Intel® C++ Compiler Professional Edition 11.1 for Windows* Installation Guide and Release Notes

Document number: 321414-002US
31 March 2010

Table of Contents

1 Introduction

This document describes how to install the product, provides a summary of new and changed product features and includes notes about features and problems not described in the product documentation.

1.1 Change History

This section highlights important changes in product updates. For a list of corrections to reported problems, please read Intel® Professional Edition Compilers 11.1 Fixes List, Intel® IPP Library 6.1 Fixes List, Intel® Math Kernel Library 10.2 Fixes List and Intel® Threading Building Blocks 2.2 Changes.

Update 6

- Intel® Integrated Performance Primitives updated to 6.1 Update 5
- Intel® Math Kernel Library updated to 10.2 Update 5
- Corrections to reported problems

Update 5 (11.1.060)

- Intel® Integrated Performance Primitives updated to 6.1 Update 4
- Intel® Math Kernel Library updated to 10.2 Update 4
- Intel® Threading Building Blocks updated to 2.2 Update 2
- Corrections to reported problems

Update 4 (11.1.054)

- OpenMP header file changed for improved error detection
- Intel® Integrated Performance Primitives updated to 6.1 Update 3
- Intel® Math Kernel Library updated to 10.2 Update 3
- Corrections to reported problems

Update 3 (11.1.051)

- Intel® Threading Building Blocks updated to 2.2 Update 1
- Corrections to reported problems

Update 2 Revised (11.1.048)

- The cross-compilers (IA-32 to Intel® 64 and IA-32 to IA-64) were rebuilt to correct an issue where these compilers would not run on certain Windows 7* and Windows Server 2008* systems. Correctness of generated code is not an issue.
- Intel® Threading Building Blocks (Intel® TBB) changed to version 2.2 in Update 2 – this was not noted earlier. The installation folder for Intel® TBB on the Intel® 64 architecture is now named "intel64" instead of "em64t".
- Microsoft Windows 7* was added as a supported operating system

Update 2 (11.1.046)

- Note added about new compiler options `/Qmkl` and `/QxAVX`
- Note added about removal of certain Intel® AVX architecture instructions and intrinsics
- System Requirements updated to note that Intel® Parallel Debugger Extension does not support Microsoft Visual Studio .NET 2003*
- Note added that if a Visual Studio solution includes a Visual Basic project, it must be manually "cleaned" after conversion to use Intel® C++.
- Note added about issue with diagnostics from command-line compiles on Intel® 64 architecture for filenames containing Japanese characters
- The Parallel Debugger Extension can now display taskwaits.
- Corrections to reported problems

Update 1 (11.1.038)

- Note added about change in behavior of `/Od`
- The Parallel Debugger Extension no longer fails due to a dependency on MSVCR71.DLL
- Added note that dynamic linking of the OpenMP library (default) is required to use the Parallel Debugger Extension
- Corrections to reported problems

Product Release (11.1.035)

1.2 Product Contents

*Intel® C++ Compiler Professional Edition 11.1 for Windows** includes the following components:

- Intel® C++ Compilers for building applications that run on IA-32, Intel® 64 or IA-64 architecture systems running the Windows* operating system
- Intel® Assembler for IA-64 Architecture applications
- Intel® Integrated Performance Primitives 6.1 Update 5
- Intel® Math Kernel Library 10.2 Update 5
- Intel® Threading Building Blocks 2.2 Update 2 for IA-32 and Intel® 64

- Intel® Parallel Debugger Extension for Microsoft Visual Studio 2005 and 2008*
- Integration into Microsoft* development environments
- Sample programs
- On-disk documentation

1.3 System Requirements

For an explanation of architecture names, see http://software.intel.com/en-us/articles/intel-architecture-platform-terminology/

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later, or compatible non-Intel processor), or based on an IA-64 architecture (Intel® Itanium®) processor
    - For the best experience, a multi-core or multi-processor system is recommended
- 1GB RAM (2GB recommended)
- 4GB free disk space for all product features and all architectures
- Microsoft Windows XP*, Microsoft Windows Vista*, Microsoft Windows 7*, Microsoft Windows Server 2003*, Microsoft Windows Server 2008* or Microsoft Windows HPC Server 2008* (embedded editions not supported)
    - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2008* SP1. Other versions of Visual Studio listed below are not supported on Windows Server 2008 or Windows HPC Server 2008.
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
    - Microsoft Visual Studio 2008* Standard Edition (or higher edition) with C++ and "X64 Compiler and Tools" components installed [1]
    - Microsoft Visual Studio 2005* Standard Edition (or higher edition) with C++ and "X64 Compiler and Tools" components installed [1]
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 architecture applications, one of:
    - Microsoft Visual Studio .NET 2003* with C++ component installed [2]
    - Microsoft Visual C++ .NET 2003* [2]
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-64 architecture applications, one of:
    - Microsoft Visual Studio 2008* Team System Edition with C++ and "Itanium Compiler and Tools" components installed [3] plus Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
    - Microsoft Visual Studio 2005* Team System Edition with C++ and "Itanium Compiler and Tools" components installed [3]
- To use command-line tools only to build IA-32 architecture applications, one of:
    - Microsoft Visual C++ 2008* Express Edition
    - Microsoft Visual C++ 2005* Express Edition and Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To use command-line tools only to build Intel® 64 architecture applications, one of:

    - Microsoft Windows Software Development Kit Update for Windows Vista*
    - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To use command-line tools only to build IA-64 architecture applications:
    - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2005 and 2008 Standard Edition installs the "x64 Compiler and Tools" component by default – the Professional and higher editions require a "Custom" install to select this.
2. Microsoft Visual Studio .NET 2003 is not supported on Microsoft Windows Vista or Microsoft Windows 7. The Intel® Parallel Debugger Extension is not supported in Microsoft Visual Studio .NET 2003. Support for Microsoft Visual Studio .NET 2003 will be removed in a future version of the product.
3. Microsoft Visual Studio is not supported for installation on IA-64 architecture systems
4. Development on an IA-64 architecture system supports building IA-64 architecture applications only.
5. The default for Intel® C++ Compiler is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions. A compiler option is available to generate code that will run on any IA-32 architecture processor.
6. Applications can be run on the same Windows versions as specified above for development. Applications may also run on non-embedded 32-bit versions of Microsoft Windows earlier than Windows XP, though Intel does not test these for compatibility. Your application may depend on a Win32 API routine not present in older versions of Windows. You are responsible for testing application compatibility. You may need to copy certain run-time DLLs onto the target system to run your application.

1.4 Documentation

Product documentation can be found in the `Documentation` folder as shown under Installation Folders.

1.5 Samples

Samples for each product component can be found in the `Samples` folder as shown under Installation Folders.

1.6 Japanese Language Support

Intel compilers provide support for Japanese language users. Error messages, visual development environment dialogs and some documentation are provided in Japanese in addition to English. By default, the language of error messages and dialogs matches that of your operating system language selection. Japanese-language documentation can be found in the `ja_JP` subdirectory for documentation and samples.

If you wish to use Japanese-language support on an English-language operating system, or English-language support on a Japanese-language operating system, you will find instructions

at http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-windows/

1.7 Technical Support

If you did not register your compiler during installation, please do so at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit http://www.intel.com/software/products/support/

Note: If your distributor provides technical support for this product, please contact them for support rather than Intel.

2 Installation

2.1 Pre-installation Steps

2.1.1 Configure Visual Studio for 64-bit Applications

If you are using Microsoft Visual Studio 2005* or 2008 and will be developing 64-bit applications (for the Intel® 64 or IA-64 architectures) you may need to change the configuration of Visual Studio to add 64-bit support.

If you are using Visual Studio 2005/2008 Standard Edition, no configuration is needed to build Intel® 64 architecture applications. For other editions:

1. From Control Panel > Add or Remove Programs, select “Microsoft Visual Studio 2005” (or 2008) > Change/Remove. The Visual Studio Maintenance Mode window will appear. Click Next.
2. Click Add or Remove Features
3. Under “Select features to install”, expand Language Tools > Visual C++
4. If the box “X64 Compiler and Tools” is not checked, check it, then click Update. If the box is already checked, click Cancel.

To use Microsoft Visual Studio 2005/2008 Team System Edition to build applications to run on IA-64 architecture systems, follow the above steps and ensure that the box “Itanium Compiler and Tools” is checked.

2.1.2 Installation on Microsoft Windows Vista* or Microsoft Windows 7*

On Microsoft Windows Vista* or Microsoft Windows 7*, Microsoft Visual Studio.NET 2003* is not supported. Microsoft Visual Studio 2005* users should install *Visual Studio 2005 Service Pack 1* (VS 2005 SP1) as well as the *Visual Studio 2005 Service Pack 1 Update for Windows Vista*, which is linked to from the VS 2005 SP1 page. After installing these updates, you must ensure that Visual Studio runs with Administrator permissions, otherwise you will be unable to use the

Intel compiler. For more information, please see Microsoft's Visual Studio on Windows Vista page (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx) and related documents.

Microsoft Visual Studio 2008 does not need an update for compatibility with Windows Vista or Windows 7, but you are encouraged to install the latest Microsoft Service Pack for Visual Studio 2008.

2.2 Installation

If you are installing the product for the first time, please be sure to have the product serial number available as you will be asked for it during installation. A valid license is required for installation and use.

To begin installation, insert the first product DVD in your computer's DVD drive; the installation should start automatically. If it does not, open the top-level folder of the DVD drive in Windows Explorer and double-click on `setup.exe`.

If you received your product as a downloadable file, double-click on the executable file (`.EXE`) to begin installation. Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.

You do not need to uninstall previous versions or updates before installing a newer version – the new version will coexist with the older versions

2.3 Changing, Updating and Removing the Product

Use the Windows Control Panel "Add or Remove Products" applet to change which product components are installed or to remove the product.

When installing an updated version of the product, you do not need to remove the older version first. You can have multiple versions of the compiler installed and select among them. If you remove a newer version of the product you may have to reinstall the integrations into Microsoft Visual Studio from the older version.

2.4 Installation Folders

The 11.1 product installs into a different arrangement of folders than in previous versions. The new arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- `C:\Program Files\Intel\Compiler\11.1\xxx`
  - `bin`
    - `ia32`
    - `ia32_intel64`
    - `ia32_ia64`
    - `intel64`
    - `ia64`
  - `Documentation`

- help
- include
- ipp
  - em64t
  - ia32
  - ia64
- lib
  - ia32
  - intel64
  - ia64
- mkl
  - benchmarks
  - em64t
  - examples
  - ia32
  - ia64
  - include
  - interfaces
  - tests
  - tools
- perf_headers
- Samples
- tbb
  - examples
  - ia32
  - include
  - intel64
- setup_c
- vsDebuggerExtension

Where *xxx* is the three-digit build number and the folders under `bin`, `include` and `lib` are used as follows:

- `ia32`: Files used to build applications that run on IA-32
- `intel64` or `em64t`: Files used to build applications that run on Intel® 64
- `ia64`: Files used to build applications that run on IA-64
- `ia32_intel64`: Compiler that runs on IA-32 to build applications that run on Intel®64
- `ia32_ia64`: Compiler that runs on IA-32 (or Intel® 64) to build applications that run on IA-64

If you are installing on a system with a non-English language version of Windows, the name of the `Program Files` folder may be different. On Intel® 64 and IA-64 architecture systems, the folder name is `Program Files (X86)` or the equivalent.

3 Intel® C++ Compiler

This section summarizes changes, new features and late-breaking news about the Intel C++ Compiler.

3.1 Compatibility

In version 11, the IA-32 architecture default for code generation has changed to assume that Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions are supported by the processor on which the application is run. See below for more information.

3.2 New and Changed Features

Please refer to the compiler documentation for details

- Additional features from C++0x
- C++ lambda functions
- Language extensions for parallel execution
- Asynchronous I/O extensions
- Decimal floating point
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- valarray implementation using IPP option
- Parallel debug code instrumentation useful for thread data sharing and reentrant call detection

3.2.1 Intel® C++ Project File Compatibility

The Intel C++ project file (`.icproj`) format changed in version 11.0. If you open a project created with an earlier version of Intel C++, you will get a message indicating that the project needs to be converted. A version 11 project cannot be used by an earlier version of the Intel C++ integration (but you can use older versions of the compiler that you have installed through Tools > Options > Intel C++ > Compilers.)

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- /hotpatch
- /Qdiag-enable:sc-parallel
- /Qmkl[:lib]
- /QxAVX
- /Yd

3.3.1 /Od no longer implies /Op

In version 11.1, the `/Od` option for disabling optimizations no longer implies `/Op` for maximizing floating-point precision. The `/Op` switch is deprecated, so we recommend using an explicit `/fp` option for applications that are sensitive to floating-point precision changes.

3.3.2 Deprecated Options

The `/YX` option, which specifies that precompiled headers should be automatically generated, is deprecated and will be removed in a future major version of Intel® C++ Compiler. Microsoft removed this option in Microsoft Visual Studio 2005. Use `/Yu` and/or `/Yc` as replacements.

For a list of additional deprecated compiler options, see the Compiler Options section of the documentation.

3.4 Other Changes and Known Issues

3.4.1 Optimization Reports Disabled by Default

As of version 11.1, the compiler no longer issues, by default, optimization report messages regarding vectorization, automatic parallelization and OpenMP threaded loops. If you wish to see these messages you must request them by specifying `/Qdiag-enable:vec`, `/Qdiag-enable:par` and/or `/Qdiag-enable:openmp`, or by using `/Qvec-report`, `/Qpar-report` and/or `/Qopenmp-report`.

Also, as of version 11.1, optimization report messages are sent to `stderr` and not `stdout`.

3.4.2 Build Environment Command Script Change

The command window script used to establish the build environment changed in version 11.1 to allow the optional specification of the version of Microsoft Visual Studio to use. If you are not using the predefined Start menu shortcut to open a build environment window, use the following command to establish the proper environment:

`"C:\Program Files\Intel\Compiler\11.1\`*`xxx`*`\Bin\iclvars.bat"` *`arch [vs]`*

Where *`xxx`* is the update number and *`arch`* is one of `ia32`, `ia32_intel64`, `intel64`, `ia32_ia64`, or `ia64` as described above under Installation Folders. *`vs`* is optional and can be one of `vs2008` or `vs2005`. If *`vs`* is not specified, the version of Visual Studio specified at installation time for command-line integration is used by default. Note that Microsoft Visual Studio .NET 2003 cannot be specified using the *`vs`* argument.

If you have installed the compiler into a different path, make the appropriate adjustments in the command.

3.4.3 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2)

When compiling for the IA-32 architecture, `/arch:SSE2` (formerly `/QxW`) is the default as of version 11.0. Programs built with `/arch:SSE2` in effect require that they be run on a processor that supports the Intel® Streaming SIMD Extensions 2 (Intel® SSE2), such as the Intel® Pentium® 4 processor and some non-Intel processors. No run-time check is made to ensure compatibility – if the program is run on a processor that does not support the instructions, an invalid instruction fault may occur. Note that this may change floating point results since the Intel® SSE instructions will be used instead of the x87 instructions and therefore computations will be done in the declared precision rather than sometimes a higher precision.

All Intel® 64 architecture processors support Intel® SSE2.

To specify the older default of generic IA-32, specify `/arch:IA32`

3.4.4 OpenMP* Libraries Default to "compat"

In version 10.1, a new set of OpenMP* libraries was added that allowed applications to use OpenMP code from both Intel and Microsoft compilers. These "compatibility" libraries can provide higher performance than the older "legacy" libraries. Beginning in version 11.0, the compatibility libraries are used by default for OpenMP applications, equivalent to `/Qopenmp-lib:compat`. If you wish to use the older libraries, specify `/Qopenmp-lib:legacy`

The "legacy" libraries (libguide.lib, libguide40.lib, etc.) will be removed in a future release of the Intel compilers.

3.4.5 Samples Provided as ZIP Archives

As of version 11.1, the compiler programming samples are provided as ZIP archives. Unpack each ZIP archive to a writable folder. All samples are now provided as Visual Studio* solutions; command-line build instructions are also provided. Please read the `samples.htm` file for more information.

3.4.6 Error Viewing Documentation in Visual Studio .NET 2003

If you are using Microsoft Visual Studio .NET 2003 but have not installed the Microsoft MSDN Library feature, you will get an error "Help Is Not Installed for Visual Studio" when using the Help menu item for *Intel® C++ Compiler Professional Edition Help*, or when using F1 context-sensitive help. As a workaround, select Help > Contents and click OK on the error message box which is then displayed. You will then be able to access the product help in the Contents pane. This issue does not affect Microsoft Visual Studio 2005 or 2008.

3.4.7 Using Intel C++ Projects with a Source Control System

If your project is managed under a source control system, for example, Microsoft Visual Source Safe* or Microsoft Visual Studio Team Foundation Server*, there are additional steps you must follow in order to use the Intel C++ project system with your project. A detailed article on this topic is available at http://software.intel.com/en-us/articles/tips-on-using-the-intel-c-compiler-with-source-code-control-software/

3.4.8 New Option to Clean Project after Conversion to use Intel C++

When the option is selected to use Intel C++ for a project, a new dialog box appears offering to "clean" the project (remove results of previous builds). This step is recommended and is selected by default; you may choose to disable the "clean" operation so that you may perform the "clean" manually.

However, if you are using a non-English version of Microsoft Visual Studio, or the solution contains a Microsoft Visual Basic* project, this dialog does not appear and you must "clean" the project manually.

3.4.9 Command-Line Diagnostic Issue for Filenames with Japanese Characters

The filename in compiler diagnostics for filenames containing Japanese characters may be displayed incorrectly when compiled within a Windows command shell using the native Intel® 64 architecture compiler. It is not a problem when using Visual Studio or when using the Intel® 64 architecture cross-compiler or IA-32 architecture compiler.

3.4.10 Certain Intel® AVX Architecture Instructions and Intrinsics Removed

The compiler does not support the `VPERMIL2PD` and `VPERMIL2PS` instructions in the assembler, and does not support the corresponding intrinsics `_mm256_permute2_pd`, `_mm_permute2_pd`, `_mm256_permute2_ps`, and `_mm_permute2_ps`. These instructions and intrinsics were removed from the Intel® AVX architecture but not from the compiler documentation. For more information, please see http://software.intel.com/en-us/blogs/2009/01/29/recent-intelr-avx-architectural-changes/

3.4.11 OpenMP Header File Changed

The OpenMP header file `omp.h` has been improved with additional error checking in version 11.1 Update 4.

The definitions of `omp_lock_t` and `omp_nest_lock_t` types have changed. With this release, the compiler distinguishes these types at compile time. This change will not affect OpenMP programs written in conformance with the OpenMP specification. However, a non-conforming OpenMP application may generate a compiler warning. For example:

```
> type sample.c
#include <omp.h>
int main() {
    omp_lock_t lk;
    omp_init_nest_lock( & lk );
    return 0;
} // main

> icl /Qopenmp sample.c

sample.c(4): warning #167: argument of type "omp_lock_t *" is
incompatible with parameter of type "omp_nest_lock_t *"

      omp_init_nest_lock( & lk );
                          ^
```

4 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about the Intel® Integrated Performance Primitives (Intel® IPP) as part of Intel C++ Compiler Professional Edition. For detailed information about Intel® IPP see the following links:

- **New features**: see the information below and visit the main Intel IPP product page on the Intel web site at: http://software.intel.com/en-us/intel-ipp.

- **Documentation, help, and samples**: see the documentation links on the IPP product page at: http://software.intel.com/en-us/intel-ipp.

4.1 New and Changed Features

4.1.1 Intel® Integrated Performance Primitives 6.1 Update 5

- This release contains no new features, only corrections to reported problems

4.1.2 Intel® Integrated Performance Primitives 6.1 Update 4

- New string processing code examples in the IPP signal processing reference manual.
- Optimizations for RSA-1024 based decryption added to the library.
- OpenSSL performance improvements and support for version 0.9.8j of OpenSSL.

4.1.3 Intel® Integrated Performance Primitives 6.1 Update 3

- New code examples in chapter 11 of the IPP signal processing reference manual.
- UMC documentation now includes motion estimation and mode decision components.
- Approximate 5% performance improvement to the BZIP2 decoder.

4.1.4 Intel® Integrated Performance Primitives 6.1 Update 2

- Support Intel® Advanced Vector Extensions (Intel® AVX)
- Support Intel® Core™ i7 processor with new optimization and threading control/optimization
- 3D Image Processing: 3D Geometric Transforms, 3D Filters
- New Data Compression Functions APIs
- New Intel IPP Crypto support to RSA_SSA1.5 and RSA_PKCSv1.5
- Unified Image Classes (UIC) to add PNG format support and new features to support DXT1, DXT3, DXT5 texture compression
- Advanced lighting functions including Spherical Harmonic and Perlin Noise generation Functions
- Windows Media* Photo Support (HD Photo): IPP PCT Functions
- New video coding areas improvement including Scene Analyzer in MPEG-2, Intensity Compensation & Overlap Smoothing in VC1
- Samples for Signal Processing, Image Processing, String Processing and C++, C# language support have been added in \Samples folder. Others can be downloadable by clicking “Free Code Samples” at http://software.intel.com/en-us/intel-ipp
- The deprecated APIs have been added more reference information in reference manual and header files.

4.2 Known Limitations

- The Visual Studio context-sensitive help may fail to work for some variants of the “generated domain” functions (functions with the “ippg” prefix).

- For a list of bug fixes, known issues, and limitations please see the following knowledge base article: http://software.intel.com/en-us/articles/intel-ipp-library-61-fixes-list/.

4.3 Intel® IPP Cryptography Libraries are Available as a Separate Download

The Intel® IPP cryptography libraries are available as a separate download. For download and installation instructions, please read
http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/

4.4 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages for Windows*, Linux* and Mac OS* at
http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/

The samples include source code for audio/video codecs, image processing and media player applications, and for calling functions from C++, C# and Java*. Instructions on how to build the sample are described in a readme file that comes with the installation package for each sample.

5 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about the Intel® Math Kernel Library (Intel® MKL) as part of Intel C++ Compiler Professional Edition.

5.1 Changes in This Version

For further information on improvements in this and previous releases, see
http://software.intel.com/en-us/articles/new-in-intel-mkl-10-2/

5.1.1 Intel® Math Kernel Library 10.2 Update 5

New Features

- Incorporated the LAPACK 3.2.1 update primarily consisting of fixes to LAPACK 3.2

Performance Improvements

- FFTs
  - Improved performance for complex FFTs, 3D and higher on the Intel® 64 architecture
- VSL
  - Improved performance of the MT19937 and MT2203 basic random number generators (BRNGs) on the 45nm Intel® Core™2 Duo processor and newer processors in 64-bit libraries

Usability and Interface Improvements

- Added support for Boost version 1.41.0 in the ublas examples
- Included Fortran 95 interfaces for the diagonally dominant solver functionality (?DTSVB, ?DTTRFB, ?DTTRSB)

- Significantly reduced the memory consumption of in-place, multi-dimensional cluster FFTs

5.1.2 Intel® Math Kernel Library 10.2 Update 4

New Features

- Introduced the single precision complex absolute value function SCABS1
- Introduced the solver ?DTSVB for diagonally dominant tri-diagonal systems which is up to 2x faster than the general solver with partial pivoting (?GTSV)
- Added routines for factorization (?DTTRFB) and the forward/backward substitution (?DTTRSB) of the diagonally dominant tri-diagonal systems

Performance improvements

- FFTs
  - Enhanced performance for transforms which are a multiple of 8 or 13
  - Optimized 1D complex cluster FFTs for non-power-of-2 vector lengths
- VSL
  - Convolution and Correlation computations that require decimation show significant improvements (re-link required, see Known Issues)

5.1.3 Intel® Math Kernel Library 10.2 Update 3

Performance Improvements

- BLAS
  - Threaded the 32-bit OS versions of the following BLAS Level 1 and 2 functions for Intel® Core™ i7 processors and Intel® Xeon® processor 5300, 5400, and 5500 series: (D,S,C,Z)COPY, (D,S,C,Z)SWAP, (D,S,C,Z)AXPY, (S,C)ROT, (S,C)DOT, CDOTC, (D,S,C,Z)GEMV, (D,S,C,Z)TRMV, (S,C)SYMV, (S,C)SYR, (S,C)SYR2
  - Improved 32-bit and 64-bit OS versions of the following BLAS level 1 functions for Intel® Xeon® processors 5300, 5400, 5500: ZAXPY, ZSCAL, ZDOT(U,C), and (D,S)ROT
  - Improved DGEMM threading efficiency for matrices with many more rows than columns for Intel® Xeon® processor 5300
- LAPACK
  - Improved scalability of the following LAPACK functions: ?POTRF, ?GEBRD, ?SYTRD, ?HETRD, and ?STEDC divide and conquer eigensolvers
- FFTs
  - Updated underlying kernels to provide widespread performance improvements in FFTs
  - Improved threading of 3D FFTs when a small number of transforms are calculated with a single function call
  - Extended threading to small size multidimensional transforms

- VML
  - Further optimization for these VML functions on Intel® Xeon® processor 5500 series: v(s,d)Asin, v(s,d)Acos, v(s,d)Ln, v(s,d)Log10, vsLog1p, v[s/d]Hypot
- VSL
  - Improved performance of viRngPoisson and viRngPoissonV random number generators

Usability and Interface Improvements

- Improved example programs for uBLAS, Java, FFTW3, LAPACK95, and BLAS95
- Some examples in the reference manual were removed where identical examples in source code form also appeared in the examples directory
- New 64-bit integer (ILP64) fftw_mpi interfaces for cluster FFTs

5.1.4 Intel® Math Kernel Library 10.2 Update 2

New Features

- LAPACK 3.2
  - 238 new LAPACK functions
  - Extra Precise Iterative Refinement
  - Non-Negative Diagonals from Householder QR factorization
  - High Performance QR and Householder Reflections on Low-Profile Matrices
  - New fast and accurate Jacobi SVD
  - Routines for Rectangular Full Packed format
  - Pivoted Cholesky
  - Mixed precision iterative refinement (Cholesky)
  - More robust DQDS algorithm
- Introduced implementation of the DZGEMM Extended BLAS function (as described at http://www.netlib.org/blas/blast-forum/). See the description of the `*gemm` family of functions in the BLAS section of the reference manual.
- PARDISO now supports real and complex, single precision data

Usability/Interface improvements

- Sparse matrix format conversion routines which convert between the following formats:
  - CSR (3-array variation) ↔ CSC (3-array variation)
  - CSR (3-array variation) ↔ diagonal format
  - CSR (3-array variation) ↔ skyline
- Fortran95 BLAS and LAPACK compiled module files (.mod) are now included
  - Modules are pre-built with the Intel Fortran Compiler and are located in the include directory (see Intel® MKL User's Guide for full path)
  - Source is still available for use with other compilers

    - Documentation for these interfaces can be found in the Intel® MKL User's Guide
- The FFTW3 interface is now integrated directly into the main libraries
    - Source code is still available to create wrappers for use with compilers not compatible with the default Intel® Fortran compiler convention for name decoration
    - See Appendix G of the Reference Manual for information
- DFTI_DESCRIPTOR_HANDLE now represents a true type name and can now be referenced as a type in user programs
- Added parameter to Jacobi matrix calculation routine in the optimization solver domain to allow access to user data (see the description of the djacobix function in the reference manual for more information)
- Added an interface mapping calls to single precision BLAS functions in Intel® MKL (functions with "s" or "c" initial letter) to 64-bit floating point precision functions has been added on 64-bit architectures (See "sp2dp" in the Intel® MKL User Guide for more information)
- Compatibility libraries (also known as "dummy" libraries) have been removed from this version of the library

Performance improvements

- Further threading in BLAS level 1 and 2 functions for Intel® 64 architecture
    - Level 1 functions (vector-vector): (CS,ZD,S,D)ROT, (C,Z,S,D)COPY, and (C,Z,S,D)SWAP
        - Increase in performance by up to 1.7-4.7 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor depending on data location in cache
        - Increase in performance by up to 14-130 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
    - Level 2 functions (matrix-vector): (C,Z,S,D)TRMV, (S,D)SYMV, (S,D)SYR, and (S,D)SYR2
        - Increase in performance by up to 1.9-2.9 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor, depending on data location in cache
        - Increase in performance by up to 16-40 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
- Introduced recursive algorithm in 32-bit sequential version of DSYRK for up to 20% performance improvement on Intel® Core™ i7 processors and Intel® Xeon® processors in 5300, 5400, and 7400 series.
- Improved LU factorization (DGETRF) by 25% over Intel MKL 10.1 Update 1 for large sizes on the Intel® Xeon® 7460 Processor; small sizes are also dramatically improved

- BLAS *TBMV/*TBSV functions now use level 1 BLAS functions to improve performance by up to 3% on Intel® Core™ i7 processors and up to 10% on Intel® Core™2 processor 5300 and 5400 series.
- Improved threading algorithms to increase DGEMM performance
    - up to 7% improvement on 8 threads and up to 50% on 3,5,7 threads on the Intel® Core™ i7 processor
    - up to 50% improvement on 3 threads on Intel® Xeon® processor 7400 series.
- Threaded 1D complex-to-complex FFTs for non-prime sizes
- New algorithms for 3D complex-to-complex transforms deliver better performance for small sizes (up to 64x64x64) on 1 or 2 threads
- Implemented high-level parallelization of out-of-core (OOC) PARDISO when operating on symmetric positive definite matrices.
- Reduced memory use by PARDISO for both in-core and out-of-core on all matrix types
- PARDISO OOC now uses less than half the memory previously used in Intel MKL 10.1 for real symmetric, complex Hermitian, or complex symmetric matrices
- Parallelized Reordering and Symbolic factorization stage in PARDISO/DSS
- Up to 2 times better performance (30% improvement on average) on Intel® Core® i7 and Intel® Core™2 processors for the following VML functions: 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
- Optimized versions of the following functions available for Intel® Advanced Vector Extensions (Intel® AVX)
    - BLAS: DGEMM
    - FFTs
    - VML: exp, log, and pow
    - See important information in the Intel® MKL User's Guide regarding the `mkl_enable_instructions()` function for access to these functions

5.2 Known Issues

A full list of the known limitations of this release can be found in the Knowledge Base for the Intel® MKL at http://software.intel.com/en-us/articles/known-limitations-in-intel-mkl-10-2

5.3 Notices

The following change is planned for future versions of Intel MKL. Please contact Technical Support if you have concerns:

- Content in the libraries containing `solver` in the filenames will be moved to the core library in a future version of Intel MKL. These `solver` libraries will then be removed.

5.4 Attributions

As referenced in the End User License Agreement, attribution requires, at a minimum, prominently displaying the full Intel product name (e.g. "Intel® Math Kernel Library") and providing a link/URL to the Intel® MKL homepage (www.intel.com/software/products/mkl) in both the product documentation and website.

The original versions of the BLAS from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/blas/index.html.

The original versions of LAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/lapack/index.html. The authors of LAPACK are E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. Our FORTRAN 90/95 interfaces to LAPACK are similar to those in the LAPACK95 package at http://www.netlib.org/lapack95/index.html. All interfaces are provided for pure procedures.

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are 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, and R. C. Whaley.

PARDISO in Intel® MKL is compliant with the 3.2 release of PARDISO that is freely distributed by the University of Basel. It can be obtained at http://www.pardiso-project.org.

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. Some FFT functions in this release of the Intel® MKL DFTI have been generated by the UHFFT software generation system under license from University of Houston. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

6 Intel® Threading Building Blocks

This section summarizes changes, new features and late-breaking news about the Intel Threading Building Blocks 2.2 as part of Intel C++ Compiler Professional Edition.

6.1 Changes in This Version

6.1.1 Intel® Threading Building Blocks 2.2 Update 2

- parallel_invoke and parallel_for_each now take function objects by const reference, not by value.
- Building TBB with /MT is supported, to avoid dependency on particular versions of Visual C++* runtime DLLs. TBB DLLs built with /MT are located in vc_mt directory.
- Class critical_section introduced.
- Improvements in exception support: new exception classes introduced, all exceptions are thrown via an out-of-line internal method.
- Improvements and fixes in the TBB allocator and malloc replacement, including robust memory identification, and more reliable dynamic function substitution on Windows*.
- Method swap() added to class tbb_thread.
- Methods rehash() and bucket_count() added to concurrent_hash_map.
- Other fixes and improvements in code, tests, examples, and docs.

6.1.2 Intel® Threading Building Blocks 2.2 Update 1

- Documentation was updated.
- TBB scheduler auto-initialization now covers all possible use cases.
- concurrent_queue: made argument types of sizeof used in padding consistent with those actually used.
- Memory allocator was improved: supported corner case of user's malloc calling scalable_malloc (non-Windows), corrected processing of memory allocation requests during tbb memory allocator startup (Linux).
- Windows malloc replacement has got better support for static objects.
- In pipeline setups that do not allow actual parallelism, execution by a single thread is guaranteed, idle spinning eliminated, and performance improved.
- RML refactoring and clean-up.
- New constructor for concurrent_hash_map allows reserving space for a number of items.
- Operator delete() added to the TBB exception classes.
- Lambda support was improved in parallel_reduce.
- gcc 4.3 warnings were fixed for concurrent_queue.
- Fixed possible initialization deadlock in modules using TBB entities during construction of global static objects.
- Copy constructor in concurrent_hash_map was fixed.
- Fixed a couple of rare crashes in the scheduler possible before in very specific use cases.
- Fixed a rare crash in the TBB allocator running out of memory.
- New tests were implemented, including test_lambda.cpp that checks support for lambda expressions.
- Fixed known exception safety issues in concurrent_vector.
- Better concurrency of simultaneous grow requests in concurrent_vector.
- TBB allocator further improves performance of large object allocation.
- Problem with source of text relocations was fixed on Linux
- Fixed bugs related to malloc replacement under Windows
- A few other small changes in code, tests, and documentation.

6.2 Known Issues

Please note the following with respect to this particular release of Intel Threading Building Blocks.

6.2.1 Issue When Multiple Visual Studio Versions Are Installed

Intel Threading Building Blocks (Intel® TBB) provides separate sets of dynamic-linked libraries (DLLs) for the various supported versions of Microsoft Visual Studio. If you have more than one supported Visual Studio version installed, for example, 2005 and 2008, installing Intel Threading Building Blocks will add the DLL folders for all supported Visual Studio versions to the system `PATH` environment variable; typically with the newest version listed first. Since the DLLs have the same names for the different Visual Studio versions, this means that only the first set as

found on `PATH` will be used. This may cause problems for applications built with one version of Visual Studio but run using the DLLs for a different version.

If you have more than one Visual Studio version installed, Intel recommends that you choose one version for your Intel® TBB application development and remove the folder for the other version from `PATH`. The folder names are of the form:

- VS2003: C:\Program Files\Intel\Compiler\11.1\xxx\tbb\ia32\vc7.1\bin
- VS2005: C:\Program Files\Intel\Compiler\11.1\xxx\tbb\ia32\vc8\bin
- VS2008: C:\Program Files\Intel\Compiler\11.1\xxx\tbb\ia32\vc9\bin

The folder path may vary depending on architecture and system language. Intel Threading Building Blocks does not provide static libraries.

6.2.2 Library Issues

- To allow more accurate results to be obtained with Intel® Thread Checker or Intel® Thread Profiler, download the latest update releases of these products before using them with Intel Threading Building Blocks.
- If you are using Intel Threading Building Blocks and OpenMP* constructs mixed together in rapid succession in the same program, and you are using Intel compilers for your OpenMP* code, set KMP_BLOCKTIME to a small value (e.g., 20 milliseconds) to improve performance. This setting can also be made within your OpenMP* code via the kmp_set_blocktime() library call. See the Intel compiler OpenMP* documentation for more details on KMP_BLOCKTIME and kmp_set_blocktime().
- In general, non-debug ("release") builds of applications or examples should link against the non-debug versions of the Intel Threading Building Blocks libraries, and debug builds should link against the debug versions of these libraries. On Windows systems, compile with /MD and use Intel Threading Building Blocks release libraries, or compile with /MDd and use debug libraries; not doing so may cause run-time failures. See the Tutorial in the product "Documentation" sub-directory for more details on debug vs. release libraries.

7 Intel® Parallel Debugger Extension

This section summarizes changes, new features and late-breaking news about the Intel Parallel Debugger Extension as part of Intel C++ Compiler.

7.1 Known Issues

- If you are using Microsoft Visual Studio 2005, there are three Intel-specific exceptions that must be enabled manually. Select `Debug` > `Exceptions`, expand the `Win32 Exceptions` tree, and enable items:

```
a1a01db0 Intel Parallel Debugger Extension Exception 0
a1a01db1 Intel Parallel Debugger Extension Exception 1
a1a01db2 Intel Parallel Debugger Extension Exception 2
```

This needs to be done once per project.

- Disabling the Intel Debugging exceptions during a debug session may cause Visual Studio (up to Visual Studio 2008, SP1) to hang.

- Use of the Intel Parallel Debugger Extension requires that the OpenMP library be linked dynamically, which is the default. If you wish to use the Parallel Debugger Extension, do not use `/Qopenmp-link:static` to specify static linking of the OpenMP Library.

- Be sure to enable the parallel debug instrumentation before you start parallel debugging: `Properties > Configuration Properties > C/C++ > Debug > Enable Parallel Debug Checks`. Otherwise, the debugger will not detect data sharing events nor break on re-entrant calls.

- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.

    - You can debug 64-bit applications under Visual Studio 2005 and 2008 without Service Packs only if they are linked to the low memory area. If not linked to the low memory area, you will not see any events until the debuggee terminates. After termination, all events are displayed in the event window. In order to debug 64-bit applications properly, set the base address to `0x10000` in `Project > Properties > Linker > Advanced`.

- Local variables are displayed as "`???`" in the Data Sharing Events window.

- The SSE Registers window does not work for 64-bit applications - the window shows "`???`"

- Filters on static local variables are not set correctly via context menu.

- Reentrant call detection stops in Disassembly view. It does not work correctly for static functions. When used in design mode, the function should be preceded by a suitable context operator, for example, `{,,myapp.exe} my_extern_function`

- The debugger extension windows remain empty when their placement is changed from "docked" to "floating". The workaround is to either keep them docked or to restart the debug session after the placement was changed.

- The debugger extension requires the application to be started from Visual Studio. It does not work when attaching to an existing process.

- Windows settings are restored to default (Hexadecimal) when the window is hidden or closed and reopened again.

7.2 Documentation

Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by clicking the Help button of specific dialog boxes. Context Sensitive Help is also available by clicking the function key F1 after activation of a Debugger Extension window.

8 Disclaimer and Legal Information