Intel® Parallel Studio XE 2015 Composer Edition for C++ Windows* Installation Guide and Release Notes

30 October 2014

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. For the most current update to these release notes, see the release notes posted at the Intel® Software Development Products Registration Center where you downloaded this product.

Due to the nature of this comprehensive integrated software development tools solution, different Intel® Parallel Studio XE Composer Edition components may be covered by different licenses. Please see the licenses included in the distribution as well as the Disclaimer and Legal Information section of these release notes for details.

1.1 Change History

This section highlights important from the previous product version and changes in product updates. For information on what is new in each component, please read the individual component release notes.

1.1.1 Changes in Update 1

- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- Tools->Options and Project Menu Labels Changed in 2015 Update 1
- First update with Japanese Localization
- Intel® C++ Compiler 15.0.1
- Intel® Math Kernel Library 11.2 Update 1
- Intel® Integrated Performance Primitives 8.2 Update 1
- Intel® Threading Building Blocks 4.3 Update 1

1.1.2 Changes since Intel® C++ Composer XE 2013 SP1 (New in Intel® Parallel Studio XE 2015 Composer Edition)

- Compiler offload to Intel® Graphics Technology is supported
- Support for native code generation for Intel® Graphics Technology
- New Optimization Report interface, structure, and options (strongly recommended to read for users of existing options /Qopt-report, /Qvec-report, /Qopenmp-report, and /Qpar-report)
- New IDE integration for optimization reports showing report information integrated with source with hyperlinking to relevant areas. See the User's Guide for details.
- Full C++11 language support
- Additional OpenMP* 4.0 support
- MIN/MAX Reductions supported in #pragma simd
- Intel® Cilk™ Plus changes
- Microsoft vectorcall calling convention supported

- Select custom installation configurations with the online installer
- Enforce same code to be executed regardless of data alignment with /Qopt-dynamic-align-
- Enable threadsafe profile generation with PGO
- Control diagnostic strictness of Pointer Checker for problems with pointers to structure fields
- aligned_new header
- Improved debugging of lambda functions
- Extended offload syntax to allow copying of non-contiguous memory
- SIMD data types (for example, __m128) updated to allow use of arithmetic and logical operators
- New pragma directives to control inlining behavior per function
- New INTEL_PROF_DYN_PREFIX environment variable to add custom prefix to PGO .dyn filenames
- Added Microsoft Visual Studio property “Base Platform Toolset” in the Intel IDE integration to explicitly specify which Visual Studio toolset to use with the Intel® C++ Compiler
- Added Microsoft Visual Studio property “Use MPI Library” in the Intel IDE integration to explicitly specify which MPI library to use with Intel® Math Kernel Library cluster configurations
- Improvements to the Intel® Performance Guide for giving guidance for applications with flat performance profiles
- Static Analysis is deprecated
- Windows XP* not supported
- Microsoft Visual Studio 2008* not supported
- Intel® C++ Compiler 15.0.0
- Intel® Debugging Extension for Intel® MIC Architecture updated
- Intel® Math Kernel Library updated to version 11.2
- Intel® Integrated Performance Primitives 8.2
- Intel® Threading Building Blocks 4.3

1.2 Product Contents

*Intel® Parallel Studio XE 2015 Update 1 Composer Edition for C++ Windows** includes the following components:

- Intel® C++ Compiler 15.0.1 for building applications that run on IA-32, Intel® 64 architecture systems, Intel® Xeon Phi™ coprocessors, or Intel® Graphics Technology running the Windows* operating system
- Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Intel® Math Kernel Library 11.2 Update 1
- Intel® Integrated Performance Primitives 8.2 Update 1
- Intel® Threading Building Blocks 4.3 Update 1

- Integration into Microsoft* development environments
- Sample programs
- On-disk documentation

1.3 System Requirements

For an explanation of architecture names, see http://intel.ly/q9JVjE

- 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
    - For the best experience, a multi-core or multi-processor system is recommended
- 2GB RAM (4GB recommended)
- 4GB free disk space for all product features and all architectures
- For Intel® Many Integrated Core Architecture (Intel® MIC Architecture) development/testing:
    - Intel® Xeon Phi™ processor
    - Intel® Manycore Platform Software Stack (Intel® MPSS)
    - Debugging of offload code requires Microsoft Visual Studio* 2012 or 2013
- For offload to Intel® Graphics Technology development/testing:
    - The following processor models are supported:
        - Intel® Xeon® Processor E3-1285 v3 and E3-1285L v3 (Intel® C226 Chipset) with Intel® HD Graphics P4700
        - 4th Generation Intel® Core™ Processors with Intel® Iris™ Pro Graphics, Intel® Iris™ Graphics or Intel® HD Graphics 4200+ Series
        - 3rd Generation Intel® Core™ Processors with Intel® HD Graphics 4000/2500

          Please note: Intel® Xeon® processors are only supported with the chipsets listed. Intel® Xeon® configurations with other chipsets are not supported. Previous generations of Intel® Core™ processors are not supported. Intel® Celeron and Intel® Atom™ processors are also not compatible.
    - The latest 32-bit or 64-bit graphics driver with support for Intel® Graphics Technology (available at http://downloadcenter.intel.com)
    - A version of binutils (specifically 2.24.51.20131210) for Windows (available at http://intel.ly/1fHX7xO)
        - Note that after installing binutils, you will need to set your `PATH` to include the directory containing `ld.exe`.
- Microsoft Windows 7*, Microsoft Windows 8*, Microsoft Windows 8.1*, Microsoft Windows Server 2008* SP2 (IA-32 only), Microsoft Windows Server 2008 (R2 SP1), Microsoft Windows HPC Server 2008*, or Microsoft Windows Server 2012* (embedded editions not supported)

    - On Microsoft Windows 8, Microsoft Windows 8.1, and Microsoft Windows Server 2012, the product installs into the “Desktop” environment. Development of “Windows 8 UI” applications is not supported. [4]
- 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 2013* Professional Edition (or higher edition) with C++ component installed
    - Microsoft Visual Studio 2012* Professional Edition (or higher edition) with C++ component installed
    - Microsoft Visual Studio 2010* Professional Edition (or higher edition) with C++ and “X64 Compiler and Tools” components installed
- To use command-line tools only to build IA-32 architecture applications, one of:
    - Microsoft Visual C++ Express 2013 for Windows Desktop*
    - Microsoft Visual C++ Express 2012 for Windows Desktop*
    - Microsoft Visual C++ 2010* Express Edition
- To use command-line tools only to build Intel® 64 architecture applications, one of:
    - Microsoft Visual C++ Express 2013 for Windows Desktop*
    - Microsoft Visual C++ Express 2012 for Windows Desktop*
    - Microsoft Windows* Software Development Kit for Windows 8*
- To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2010 includes x64 support by default.
2. The default for the Intel® compilers is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions - for example, the Intel® Pentium® 4 processor. A compiler option is available to generate code that will run on any IA-32 architecture processor. However, if your application uses Intel® Integrated Performance Primitives or Intel® Threading Building Blocks, executing the application will require a processor supporting the Intel® SSE2 instructions.
3. 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.
4. The Intel® C++ Compiler does not support development of Windows 8* UI apps. We are always interested in your comments and suggestions. For example, if you want to use the Intel® C++ Compiler or other Intel software development capabilities in Windows 8 UI apps, please file a request at Intel® Premier Support (https://premier.intel.com/).
   If you are interested in experimenting with unsupported Intel software development capabilities for developing Windows 8 UI apps, please read the article at http://intel.ly/WLeXRo .

1.3.1 Visual Studio 2008* is Not Supported

Support has been removed for installation and use with Visual Studio 2008. Intel recommends migrating to a newer version of Visual Studio*.

1.3.2 Windows XP* is Not Supported

Support has been removed for installation and use on Windows XP. Intel recommends migrating to a newer version of these operating systems.

1.4 Documentation

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

1.4.1 Changes in Online Help format in Microsoft Visual Studio*

The online help format is now browser-based. When you view Intel documentation from the Microsoft Visual Studio Help menu, or when you view context-sensitive help using F1 or a help button in a dialog box or other GUI element, your default browser shows the corresponding help topic. You may encounter some minor functionality issues depending on your default browser. Known issues include:

- When `Set Help Preference` is set to `Launch in Browser` and you hit F1 in `Tools>Options>F#` Tools or `Tools>Options>Intellitrace`, the browser appears twice.
- **Chrome*:** When arriving at a topic from Search or Index, the Table of Contents (TOC) does not sync, nor does the `Sync TOC` link work.
- **Firefox*:** The TOC loses context easily. Search is case sensitive.
- **Safari*:** Response on Windows* is slow.

1.4.2 Documentation Viewing Issue with Microsoft Internet Explorer* 10 and Windows Server* 2012

If on Windows Server 2012 you find that you cannot display help or documentation from within Internet Explorer 10, modifying a security setting for Microsoft Internet Explorer* usually corrects the problem. From Tools > Internet Options > Security, add “about:internet” to the list of trusted sites. Optionally, you can remove “about:internet” from the list of trusted sites after you finish viewing the documentation.

1.4.3 Documentation viewing Issue with Visual Studio 2012 and Windows Server 2012

If on Windows Server 2012* you find that you cannot display help or documentation from within Visual Studio 2012, modifying a security setting for Microsoft Internet Explorer* usually corrects the problem. From Tools > Internet Options > Security, change the settings for Internet Zone to allow “MIME Sniffing” and “Active Scripting”.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

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 (when the combined Japanese-English installation is used). 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.

Japanese language support will be available in an update on or after the release of Intel® C++ Parallel Studio XE 2015 Composer Edition.

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://intel.ly/oZjpZs

1.7 Technical Support

If you did not register your compiler during installation, please do so at the Intel® Software Development Products Registration Center at http://registrationcenter.intel.com. 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 Online Installation now available

The electronic installation package for Intel® Parallel Studio XE now offers as an alternative a smaller installation package that dynamically downloads and then installs packages selected to be installed. This requires a working internet connection and potentially a proxy setting if you are behind an internet proxy. Full packages are provided alongside where you download this online install package if a working internet connection is not available. The online installer may be downloaded and saved as an executable file which can then be launched from the command line.

2.1.1 Storing Online Installer Download Content

The online installer stores the downloaded content in the form-factor of the standard install package which can then be copied and reused offline on other systems. The default download location is <Program Files>\Intel\Download. This location may be changed with the online installer command line option "--download-dir [FOLDER]". The online installer also supports a download only mode which allows the user to create a package without installation. This mode is enabled with the "--download-only" command line option.

2.2 Installation of Intel® Manycore Platform Software Stack (Intel® MPSS)

The Intel® Manycore Platform Software Stack (Intel® MPSS) may be installed before or after installing the Intel® Parallel Studio XE for Windows* product.

Using the latest version of Intel® MPSS available is recommended. It is available from the Intel® Software Development Products Registration Center at http://registrationcenter.intel.com as part of your Intel® Parallel Studio XE for Windows* registration.

Refer to the Intel® MPSS documentation for the necessary steps to install the user space and kernel drivers.

2.3 Intel® Software Manager

The installation now provides an Intel® Software Manager to provide a simplified delivery mechanism for product updates and provide current license status and news on all installed Intel® software products.

You can also volunteer to provide Intel anonymous usage information about these products to help guide future product design. This option, the Intel® Software Improvement Program, is not enabled by default – you can opt-in during installation or at a later time, and may opt-out at any time. For more information please see http://intel.ly/SoftwareImprovementProgram.

2.4 Pre-installation Steps

2.5 Installation

The installation of the product requires a valid license file or serial number. If you are evaluating the product, you can also choose the "Evaluate this product (no serial number required)" option during installation.

To begin installation, double-click on the executable file (.EXE). 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

2.5.1 Changes to system PATH may cause temporary in-operation of command shell (cmd.exe)

On Windows* 7 or 8, if the installation's additions to the system PATH cause the PATH length to consist of between 2000-4000 characters, this could cause the Windows command prompt (cmd.exe) to not work until the next reboot. If you observe such behavior after installation, reboot and if the symptom persists, contact Technical Support.

2.5.2 Silent Install

For information on automated or "silent" install capability, please see http://intel.ly/nKrzhv

2.5.2.1 Support of Non-Interactive Custom Installation

Intel® Parallel Studio XE 2015 supports the saving of user install choices during an 'interactive' install in a configuration file that can then be used for silent installs. This configuration file is created when the following option is used from the command line install:

- `--duplicate=config_file_name`: it specifies the configuration file name. If full path file name is specified, the "`--download-dir`" is ignored and the installable package will be created under the directory where configuration file is.
- `--download-dir=dir_name`: optional, it specifies where the configuration file will be created. If this option is omitted, the installation package and the configuration file will be created under the default download directory:

```
%Program Files%\Intel\Download\<package_id>
```

For example: `w_ccompxe_online_2015.0.0XX.exe --duplicate=ic15_install_config.ini --download-dir="C:\temp\custom_pkg_ic15"`
The configuration file and installable package will be created under "`C:\temp\custom_pkg_ic15`".

2.5.3 Cluster Installation

If Microsoft Compute Cluster Pack* is present, and the installation detects that the installing system is a member of a cluster, the product will be installed on all visible nodes of the cluster

when a “Full” installation is requested. If a “Custom” installation is requested, you will be given the option to install on the current node only.

2.5.4 Using a License Server

If you have purchased a “floating” license, see http://intel.ly/pjGfwC for information on how to install using a license file or license server. This article also provides a source for the Intel® License Server that can be installed on any of a wide variety of systems.

2.6 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.

2.7 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- C:\Program Files\Intel\Composer XE 2015
  - bin
    - ia32
    - ia32_gfx
    - ia32_intel64
    - intel64
    - intel64_gfx
    - intel64_mic
    - sourcechecker
  - compiler
    - include
      - cilk
      - gfx
      - ia32
      - intel64
      - mic
    - lib
      - ia32
      - ia32_gfx
      - intel64
      - intel64_gfx
      - mic

        - perf_headers
            - c++
    - debugger
        - gdb
            - LICENSES
            - src
            - target
            - w64_mic
        - debuggerextension
            - mic
    - Documentation
        - en_US
            - compiler_c
            - debugger
            - gs_resources
            - ipp
            - mkl
            - ssadiag_docs
            - tbb
            - tutorials
        - ja_JP
            - compiler_c
            - debugger
            - gs_resources
            - ipp
            - mkl
            - ssadiag_docs
            - tbb
            - tutorials
        - msvhelp
            - 1033
    - ipp
        - bin
            - ia32
            - intel64
        - examples
        - include
        - interfaces
        - lib
            - ia32
            - intel64

            - mic
        - tools
            - ia32
            - intel64
- mkl
    - benchmarks
        - linpack
        - mp_linpack
    - bin
        - ia32
        - intel64
    - examples
    - include
        - fftw
        - ia32
        - intel64
        - mic
    - interfaces
        - fftw2xc
        - fftw3xc
    - lib
        - ia32
        - intel64
        - mic
    - tests
    - tools
        - builder
- redist
    - ia32
        - compiler
        - ipp
        - mkl
        - tbb
    - intel64
        - compiler
        - ipp
        - mkl
        - tbb
- Samples
    - en_US
        - C++

            - ipp
            - mkl
        - ja_JP
            - C++
            - ipp
            - mkl
    - tbb
        - bin
        - examples
            - common
            - concurrent_hash_map
            - concurrent_priority_queue
            - GettingStarted
            - graph
            - parallel_do
            - parallel_for
            - parallel_reduce
            - pipeline
            - task
            - task_group
            - task_priority
            - test_all
        - include
            - serial
            - tbb
        - lib
            - ia32
            - intel64
            - mic

Where the folders under `bin, include,` and `lib` are used as follows:

- `ia32`: Files used to build applications that run on IA-32
- `intel64`: Files used to build applications that run on Intel® 64
- `ia32_intel64`: Compilers that run on IA-32 to build applications that run on Intel®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 architecture systems, the folder name is `Program Files (X86)` or the equivalent.

By default, updates of a given version will replace the existing directory contents. When the first update is installed, the user is given the option of having the new update installed alongside the

previous installation, keeping both on the system. If this is done, the top-level folder name for the older update is changed to `Composer XE 2015.nnn` where *nnn* is the update number.

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

This product now contains Intel® C++ Compiler 15.0. The following features are new or significantly enhanced in this version. For more information on these features, please refer to the documentation.

- Support for offload to Intel® Graphics Technology
- Support for native code generation for Intel® Graphics Technology
- New Optimization Report interface, structure, and options
- New IDE integration for optimization reports showing report information integrated with source with hyperlinking to relevant areas. See the User's Guide for details.
- Full C++11 language support (includes these feature new to 15.0) (/Qstd=c++11)
    - Value categories (N3055)
    - alignas and alignof (N2341)
    - decltype extensions (N3049, N3276)
    - Inheriting constructors (N2540)
    - User-defined literals (N2765)
    - thread_local (N2659)
    - The version of header files and libraries that are provided as part of the Intel® Manycore Platform Software Stack distribution on Intel® Many Integrated Core Architecture is an experimental 4.7.0 version. This version lacks full support for some gcc* features that are available in the final release of the 4.7.0 gcc libraries. In particular, this version of gcc lacks the support for allocator traits.
- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- Additional OpenMP* 4.0 support
- MIN/MAX Reductions supported in #pragma simd
- Intel® Cilk™ Plus changes
- Microsoft vectorcall calling convention supported
- Static Analysis is deprecated
- aligned_new header
- Improved debugging of lambda functions

- Extended offload syntax to allow copying of non-contiguous memory
- SIMD data types (for example, __m128) updated to allow use of arithmetic and logical operators
- New pragma directives to control inlining behavior per function
- New INTEL_PROF_DYN_PREFIX environment variable to add custom prefix to PGO .dyn filenames
- Added Microsoft Visual Studio property “Base Platform Toolset” in the Intel IDE integration to explicitly specify which Visual Studio toolset to use with the Intel® C++ Compiler
- Added Microsoft Visual Studio property “Use MPI Library” in the Intel IDE integration to explicitly specify which MPI library to use with Intel® Math Kernel Library cluster configurations
- Improvements to the Intel® Performance Guide for giving guidance for applications with flat performance profiles

3.2.1 Support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions for IA-32 and Intel® 64 architectures in 15.0.1

The Intel® Compiler 15.0.1 now supports Intel® AVX-512 instructions for processors based on IA-32 and Intel® 64 architectures that support that instruction set. The instructions are supported via inline assembly, intrinsics (using the zmmintrin.h include file), and/or the /Q[a]xCORE-AVX512 (Windows*) or –[a]xCORE-AVX512 (Linux*/OS X*) compiler options. This is in addition to the current support for Intel® AVX-512 instructions for Intel® Many Integrated Core Architecture.

3.2.2 MIN/MAX Reductions supported in #pragma simd

Starting with the Intel® Compilers version 15.0, #pragma simd now supports MIN/MAX reductions, like so:

```
#pragma simd reduction(max:simdMax)
   for(int i = 0; i < size; ++i)
      if(x[i] > simdMax)
         simdMax = x[i];

#pragma simd reduction(min:simdMin)
   for(int i = 0; i < size; ++i)
      if(x[i] < simdMin)
         simdMin = x[i];
```

3.2.3 Support for native code generation for Intel® Graphics Technology

By default, the compiler will generate virtual ISA code for the kernels to be offloaded to Intel® Graphics Technology. The vISA is target independent and will run on processors that have the Intel graphics processor integrated on the platform and that have the proper Intel® HD Graphics driver installed. The Intel HD Graphics driver contains the offload runtime support and a Jitter (just-in-time compiler) that will translate the virtual ISA to the native ISA at runtime for the

platform on which the application runs and do the offload to the processor graphics. The Jitter gets the current processor graphics information at runtime. The new feature allows generation of native ISA at link time by using the new option /Qgpu-arch:<arch> for Windows and –mgpu-arch=<arch> for Linux. The option is described in detail in the User's Guide.

3.2.4 Static Analysis is deprecated

Support for Static Analysis is deprecated and will be removed in a future release. If you have concerns or feedback, please comment. Note that the Microsoft Visual Studio* IDE menus have been disabled for Static Analysis. To re-enable them, set the environment variable "__INTEL_STATIC_ANALYSIS" and restart Visual Studio.

3.2.5 Support for offload to Intel® Graphics Technology

Support is provided via either a synchronous (with `#pragma offload target(gfx)` and a `cilk_for` parallel loop) or asynchronous (with `#pragma offload target(gfx_kernel)` and APIs provided in the provided `gfx_rt.h` header) offload implementation. Please see the Intel Compiler User's Guide under `Key Features->Intel® Graphics Technology` for information. Known limitations are documented in the release notes.

3.2.6 New Optimization Report interface, report structure, and options in Intel® C++ Compiler 15.0

The four kinds of optimization reports (/Qopt-report, /Qvec-report, /Qopenmp-report, and /Qpar-report) have been consolidated under one /Qopt-report interface in this version of the Intel C++ Compiler. This consolidated optimization report has been rewritten to improve the presentation, content, and precision of the information provided so that users better understand what optimizations were performed by the compiler and how they may be tuned to yield the best performance.

The output of this report no longer defaults to stderr due to issues with parallel builds. Instead, by default an output file (extension .optrpt) containing the report for each corresponding object file is generated to the target directory of the compilation process (i.e. the same directory where object files would be generated). /Qopt-report-file (for example: /Qopt-report-file:stderr) can be used to change this behavior.

The /Qvec-report, /Qopenmp-report, and /Qpar-report options have been deprecated, but they remain and map to corresponding values of the /Qopt-report option. However, the report information and formatting, and the default to reporting to a file, will follow the new opt-report model.

It is strongly recommended that you read the documentation for full details. See the Intel Compiler User's Guide under `Compiler Reference->Compiler Option Categories and Descriptions->Optimization Report Options`.

3.2.7 Updated Support for Upcoming OpenMP* features added in Intel® C++ Compiler 15.0

The Intel C++ Compiler 15.0 adds the following OpenMP* 4.0 features:

- `cancel` and `cancellation point` directives
- `depend` clause for `task` directives

OpenMP* 4.0 user defined reductions are not supported.

3.2.8 Intel® Cilk™ Plus changes in Intel® C++ Compiler 15.0

Please note the following new features for Intel® Cilk™ Plus in the Intel C++ Compiler 15.0:

- Ability to implement explicit vector programming with keywords as an alternative to #pragma simd syntax. The keywords are _Simd, _Safelen, and _Reduction. See the compiler User's Guide for more details.
- __intel_simd_lane() intrinsic to represent the "lane id" within a SIMD vector function (__declspec(vector))
- Intel Cilk Plus documentation can now be generated using Doxygen*. See the compiler/include/cilk/ReadMe.html for more details.
- New __declspec(vector_variant(…)) to define a vector-specific overload of a scalar function.

3.2.9 Microsoft vectorcall calling convention supported

Support is now available for the Microsoft vectorcall calling convention introduced in Microsoft Visual Studio 2013*.

3.2.10 aligned_new header provides way to correctly dynamically allocate data with class types with alignment specifications

C++11 allows the programmer to specify increased data alignment for class types, but there is no standard mechanism for actually allocating data with that alignment based on those types. aligned_new provides overloads of new and delete that will properly align such data.

3.2.11 New pragma directives to control inlining behavior per function

Intel® C++ Compiler 15.0 has added two new pragma directives, inline-max-per-routine and inline-max-total-size, to control the inlining behavior per function. See the C++ Compiler User's Guide under Compiler Reference->Pragmas->Intel-specific Pragma Reference for more information.

3.2.12 Static Analysis Feature (formerly "Static Security Analysis" or "Source Checker") Requires Intel® Inspector XE

The "Source Checker" feature, from compiler version 11.1, has been enhanced and renamed "Static Analysis". The compiler options to enable Static Analysis remain the same as in compiler version 11.1 (for example, `/Qdiag-enable:sc`), but the results are now written to a file that is interpreted by Intel® Inspector XE rather than being included in compiler diagnostics output.

3.2.13 Intel® C++ Project File Compatibility

The Intel C++ project file (`.icproj`) format changed in version 15.0. 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 15.0 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

For details on these and all compiler options, see the Compiler Options section of the on-disk documentation.

3.3.1 New and Changed in Intel® C++ Compiler 15.0

- /QxCORE-AVX512
- /QaxCORE-AVX512
- /Qmic
- /Qgpu-arch:<arch>[,<arch>]
- /Qopt-report-names:[mangled|unmangled]
- “jit” added as a <tool> for /Qoffload-option for offload to Intel® Graphics Technology
- /Qno-builtin-<func>
- /Gv (make vectorcall default calling convention)
- /Qopt-dynamic-align[-]
- /Qprof-gen:threadsafe
- /Qopt-report adds levels 4 and 5
- /Qopt-report-file:{stdout | stderr | <file>}
- /Qopt-report-per-object
- /Qopt-report-filter:<string>
- /Qopt-report-format:[text|vs]
- /Qopt-report-embed[-]
- /Qcheck-pointers-narrowing[-]
- /Qicl-
- /Zc:trigraphs[-]
- /fast updated to include /fp:fast=2
- /Qeliminate-unused-debug-types[-]
- /I-

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

3.3.2 Use /I- to control if search path is used for include files with angle brackets

Use of /I- effectively splits the command line include paths specified. Any directories specified with -I options before /I- are searched only for headers included with the "`#include "file"`" directive. They are not searched for headers included with angle brackets as in "`#include <file>`". If additional directories are specified with /I options after the /I- in the command line, those directories are searched for all #include directives. In addition, /I- inhibits the use of the current file directory as the first search directory for includes with quotes as in "`#include "file"`".

3.3.3 Enforce same code to be executed regardless of data alignment with /Qopt-dynamic-align-

By default, the compiler may generate multiple code paths to execute depending on the alignment of data in order to improve performance which may affect the consistency of floating-point calculations. To disable this behavior, use /Qopt-dynamic-align-

3.3.4 Enable threadsafe profile generation with PGO

To enable the safe generation of profile information in multithreaded applications, use the /Qprof-gen:threadsafe option.

3.3.5 Control diagnostic strictness of Pointer Checker for problems with pointers to structure fields

The /Qcheck-pointers-narrowing- option can be used to disable Pointer Checker diagnostics for problems with pointers to structure fields.

3.3.6 Deprecated Options

Use following method to find all the deprecated compiler options:

1) Open a command prompt from Start menu: Start > All Programs > Intel Parallel Studio XE 2015 > Command Prompt > Parallel Studio XE with Intel Compiler XE v15.0 [Update xx] > IA-32/Intel® 64 Visual Studio xxxx mode
2) Run command:

```
>> icl /? deprecate
```

3.4 Other Changes

3.4.1 Tools->Options and Project Menu Labels Changed in 2015 Update 1

Beginning in Intel® Parallel Studio XE 2015 update 1, some of the labels used to identify the Intel® Compiler have changed. Specifically:

- Under the `Tools->Options` menu, the label “Intel Parallel Studio XE” on the left is now called “Intel Compilers and Tools”. The settings available to be set there have not changed (for example, include directories, Code Coverage settings, or Performance Libraries settings).
- Under the `Project` menu, or when you right-click on a project, the menu entry “Intel Compiler XE 15.0” is now called “Intel Compiler”.

3.4.2 Build Environment Command Script Change

The command window script used to establish the build environment allows 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:

```
"<install-dir>\bin\compilervars.bat" arch [vs]
```

Where *arch* is one of followings as appropriate for the target architecture you want to build for:

- `ia32`
- `ia32_intel64`
- `intel64`

If you are targeting Intel® Many Integrated Core Architecture, you will need to use either `ia32_intel64` or `intel64` depending on whether your development system is 32-bit or 64-bit, respectively.

`vs` is optional and can be one of followings. If *`vs`* is not specified, the version of Visual Studio specified at installation time for command-line integration is used by default.

- `vs2013`
- `vs2012`
- `vs2010`

If you also have the Intel® Visual Fortran Compiler 15.0 installed, this command will also establish the environment for using that compiler.

The script file names iclvars.bat and ifortvars.bat have been retained for compatibility with previous releases.

3.4.3 OpenMP* Static Libraries Removed

The static versions of the OpenMP* runtime libraries have been removed from this release. You must link these runtimes dynamically.

3.4.4 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://intel.ly/pImnpo

3.5 Known Issues

3.5.1 Compiler Known Issues

3.5.1.1 Pointer Checker requires a dynamic runtime library

When using the `/Qcheck-pointers` option, the runtime library libchkp.dll must be linked in. When using options like `/MT` with `/Qcheck-pointers`, be aware that this dynamic library will be linked in regardless of your settings. See the article at http://intel.ly/1jV0eWD for more information.

3.5.1.2 Mixed Microsoft / Intel compilation of 256 vector bit type parameters may generate code that causes an access violation at runtime*

A general protection fault caused by an unaligned data access may occur when an application is built using two different compilers - the Microsoft Visual C++ 2013* compiler and the Intel® C++ Compiler 15.0. The problem may arise when 256 vector bit type parameters are passed by

reference in a call, when the caller is built with Visual C++*, and the parameters are accessed by functions built with the Intel C++ Compiler.

The problem arises due to a mismatch in the alignment of the 256 vector bit type parameters.

This problem will not occur when the /Qx<code> compiler option is used with <code> equal to AVX or with a newer code value, such as CORE-AVX-I, CORE-AVX2, etc., due to the fact that unaligned access instructions are used in these instances unless __mm256_stream_* (non-temporal data load/store intrinsics) are used explicitly in the application source code.

3.5.1.3 Displaying online documentation with Internet Explorer 10*

A script used in the documentation may be blocked in Internet Explorer 10*. When this occurs, the documentation will display as a blank page or the error message "Internet Explorer restricted this page from running scripts or ActiveX controls" will appear. Click `"Allow blocked content"` in order to display the documentation. If the error message does not appear, go to `Tools > Internet Options > Security` in the Internet Explorer settings and set it to `"prompt before downloading potentially unsafe content."`

3.5.2 Visual Studio Known Issues

3.5.2.1 Showing Documentation Issue with Microsoft Visual Studio 2012* or 2013* and Windows Server 2012*

If on Windows Server 2012* you find that you cannot display help or documentation from within Visual Studio 2012* or 2013*, correcting a security setting for Microsoft Internet Explorer* usually corrects the problem. From `Tools > Internet Options > Security`, change the settings for `Internet Zone` to allow "`MIME Sniffing`" and "`Active Scripting`".

3.5.2.2 MSVCP90D.dll (or other Microsoft runtime DLL) is missing

There are situations where the sample projects provided (or any Microsoft Visual C++* project potentially) could run into a runtime System Error where the application cannot find a Microsoft Visual Studio* runtime DLL. This is related to manifest files and SXS assemblies potentially missing. The simplest solution is to go to your redist directory for the version of Microsoft Visual Studio* you are using (default location would be c:\program files[(x86)]\Microsoft Visual Studio X.X\VC\redist). There will be several subdirectories under this location, sorting files out by amd64, x86 or Debug_NonRedist (if you have D in the runtime name, this usually indicates a Debug library found in this folder). Find the appropriate folder that contains the runtime you are looking for, and then copy the entire contents of that folder (including a .manifest file) to the directory where the .exe you are trying to run is located.

3.5.2.3 Visual Studio 2010 sets default of `/fp:precise`

A project created in or converted to Visual Studio 2010 will have the command line option `/fp:precise` set by default. This option sets the "floating point model" to improve consistency for floating point operations by disabling certain optimizations, reducing performance. To set the option back to the Intel default of `/fp:fast`, change the project property C++ > Optimization > Floating Point Model to Fast.

3.5.2.4 Language packs of Visual Studio 2010

If you install a new language pack of Visual Studio 2010 after installing the Intel Parallel Studio XE 2015, you may not see the Intel C++ Compiler specific options in the Project Property dialog. Please try the following to fix the issue:

1) If directory "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 15.0\1033`" exists, copy all files to "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 15.0\<locale-ID>`".

2) If directory "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\1033\`" exists, copy all files to "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\<locale-ID>`".

* The `<locale-ID>` is the language pack.

Another method is to uninstall the Intel Parallel Studio XE 2015 and reinstall the Intel Parallel Studio XE 2015.

3.5.3 Known Issues for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

3.5.3.1 Limitations of `_Cilk_shared`

- A virtual base class may not have the `_Cilk_shared` attribute
- A class with the `_Cilk_shared` attribute may not be derived from multiple base classes (multiple inheritance is disallowed)
- A class with the `_Cilk_shared` attribute may not define a virtual destructor
- A class with the _Cilk_shared attribute, if used as a base class of another `_Cilk_shared` class, must be rounded up to be a multiple of 8 in size by the programmer (by adding dummy fields as needed)
- _Cilk_offload may not be used in a program that uses shared libraries (DLLs)

3.5.3.2 Using offload code in shared libraries requires main program to be linked with /Qoffload:mandatory or /Qoffload:optional option

There is initialization required for offload that can only be done in the main program. For offload code in shared libraries, this means that the main program must also be linked for offload so that the initialization happens. This will happen automatically if the main code or code statically linked with the main program contains offload constructs. If that is not the case, you will need to link the main program with the `/Qoffload:mandatory` or `/Qoffload:optional` compiler options.

3.5.3.3 Missing symbols not detected at link time

In the offload compilation model, the binaries targeting the Intel® MIC Architecture are generated as dynamic libraries (.so). Dynamic libraries do not need all referenced variables or routines to be resolved during linking as these can be resolved during load time. This behavior could mask some missing variable or routine in the application resulting in a failure during load time. In order to identify and resolve all missing symbols at link time, use the following command line option to list the unresolved variables.

```
/Qoffload-option,mic,compiler,"-z defs"
```

3.5.3.4 *MIC* tag added to compile-time diagnostics

The compiler diagnostics infrastructure is modified to add an additional offload *MIC* tag to the output message to allow differentiation from the Target (Intel® MIC Architecture) and the host CPU compilations. The additional tag appears only in the Target compilation diagnostics issued when compiling with offload extensions for Intel® MIC Architecture.

In the examples below the sample source programs trigger identical diagnostics during both the host CPU and Target Intel® MIC Architecture compilations; however, some programs will generate different diagnostics during these two compilations. The new tag permits easier association with either the CPU or Target compilation.

```
$ icl -c sample.c
sample.c(1): warning #1079: *MIC* return type of function "main" must
be "int"
  void main()
       ^

sample.c(5): warning #120: *MIC* return value type does not match the
function type
      return 0;
             ^

sample.c(1): warning #1079: return type of function "main" must be
"int"
  void main()
       ^

sample.c(5): warning #120: return value type does not match the
function type
      return 0;
```

3.5.3.5 Runtime Type Information (RTTI) not supported

Runtime Type Information (RTTI) is not supported under the Virtual-Shared memory programming method; specifically, use of `dynamic_cast<>` and `typeid()` is not supported.

3.5.3.6 Direct (native) mode requires transferring runtime libraries like libiomp5.so to coprocessor

The Intel® Manycore Platform Software Stack (Intel® MPSS) does not include Intel compiler libraries under /lib, for example the OpenMP* library, libiomp5.so.

When running OpenMP* applications in direct mode (i.e. on the coprocessor card), users must first upload (via scp) a copy of the Intel® MIC Architecture OpenMP* library (`<install_dir>\compiler\lib\mic\libiomp5.so`) to the card (device names will be of the format `micN`, where the first card will be named `mic0`, the second `mic1`, and so on) before running their application.

Failure to make this library available will result in a run-time failure like:

```
/libexec/ld-elf.so.1: Shared object "libiomp5.so" not found, required
by "sample"
```

This can also apply to other compiler runtimes like libimf.so. The required libraries will depend on the application and how it's built.

3.5.3.7 Calling exit() from an offload region

When calling exit() from within an offload region, the application terminates with an error diagnostic "`offload error: process on the device 0 unexpectedly exited with code 0`"

3.5.4 Known issues for offload to Intel® Graphics Technology

3.5.4.1 gfx_linker: : error : command 'ld.exe' exited with non-zero exit code -107374170

When offloading code to Intel® Graphics Technology in an x64 project, you may get a linker error with the ld.exe provided by binutils. To resolve this problem, add the binutils bin directory for the 64-bit linker to your PATH environment variable and then restart Microsoft Visual Studio*.

3.5.4.2 Host-side execution of offload code is not parallelized

The compiler will generate both a target and host version of the parallel loop under `#pragma offload`. The host version is executed when the offload cannot be performed (usually when the target system does not have a unit with Intel® Graphics Technology enabled).The parallel loop must be specified using the parallel syntax of `cilk_for` or an Array Notation statement, which has parallel semantics for offload. The target version of the loop will be parallelized for target execution, but there is a current limitation where the host-side back-up version of the parallel loop will not be parallelized. Please be aware this can affect the performance of the back-up code execution significantly when offload execution does not happen in the case of `cilk_for` use. Array notation does not currently generate parallel code on the host, so performance should not differ here in that case. This is a known issue that may be resolved in a future product release.

3.5.4.3 Other issues

- On Windows 7*, for offloading to happen, the display cannot be locked. An active display is required.
- In the offloaded code, the following are not allowed:
    - Exception handling
    - RTTI
    - longjmp/setjmp
    - VLA
    - Variable parameter lists
    - Virtual functions, function pointers, or other indirect calls or jumps
    - Shared virtual memory
    - Data structures containing pointers, such as arrays or structs
    - Globals with pointer or reference type
    - OpenMP*
    - cilk_spawn or cilk_sync
    - Intel® Cilk™ Plus reducers
    - ANSI C runtime library calls (with the exception of SVML, math.h, and mathimf.h calls and a few others)
- 64-bit float and integer operations are inefficient

3.5.5 Intel® Cilk™ Plus Known Issues

- Microsoft C++ Structured Exception Handling (SEH) will fail if an SEH exception is thrown after a steal occurs and before the corresponding `_Cilk_sync`.

3.5.6 Guided Auto-Parallel Known Issues

Guided Auto Parallel (GAP) analysis for single file, function name or specific range of source code does not work when Whole Program Interprocedural Optimization (`/Qipo`) is enabled. The workaround is to disable `/Qipo` – in Visual Studio, this is `Project > [projectname] Properties > C++ > Optimization > Interprocedural Optimization > No`.

3.5.7 Static Analysis Known Issues

3.5.7.1 Excessive false messages on C++ classes with virtual functions

Note that use of the Static Analysis feature also requires the use of Intel® Inspector XE.

Static Analysis reports a very large number of incorrect diagnostics when processing any program that contains a C++ class with virtual functions. In some cases the number of spurious diagnostics is so large that the result file becomes unusable.

If your application contains this common C++ source construct, add the following command line switch to suppress the undesired messages:`/Qdiag-disable:12020,12040` (Windows) or `-diag-disable 12020,12040` (Linux). **This switch must be added at the link step because that is when static analysis results are created.** Adding the switch at the compile step alone is not sufficient. In Microsoft Visual Studio, add this to the property page Linker > Command Line.

If you are using a build specification to perform static analysis, add the `-disable-id 12020,12040` switch to the invocation of the inspxe-runsc, for example,

```
inspxe-runsc -spec-file mybuildspec.spec -disable-id 12020,12040
```

If you have already created a static result that was affected by this issue and you are able to open that result in the Intel® Inspector XE GUI, then you can hide the undesired messages as follows:

- The messages you will want to suppress are `"Arg count mismatch"` and `"Arg type mismatch"`. For each problem type, do the following:
- Click on the undesired problem type in the Problem filter. This hides all other problem types.
- Click on any problem in the table of problem sets
- Type control-A to select all the problems
- Right click and select *Change State -> Not a problem* from the pop-up menu to set the state of all the undesired problems
- Reset the filter on problem type to All
- Repeat for the other unwanted problem type
- Set the Investigated/Not investigated filter to *Not investigated*. You may have to scroll down in the filter pane to see it as it is near the bottom. This hides all the undesired messages because the "Not a problem" state is considered a "not investigated" state.

4 Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

This section summarizes new features and changes, usage and known issues related to the Intel® Debugger Extension. This debugger extension only supports code targeting Intel® Many Integrated Core Architecture (Intel® MIC Architecture).

4.1.1 Features

- Support for both native coprocessor applications and host applications with offload extensions
- Debug multiple coprocessor cards at the same time (with offload extension)

4.1.2 Using the Intel® Debugger Extension

The Intel® Debugger Extension is a plug-in for the Microsoft Visual Studio* IDE. It transparently enables debugging of projects defined by that IDE. Applications for Intel® Xeon Phi™ coprocessors can be either loaded and executed or attached to.

Instructions on how to use Intel® Debugger Extension can be found in the documentation.

4.1.3 Documentation

The full documentation for the Intel® Debugger Extension can be found here:

```
<install-dir>\Documentation\[en_US|ja_JP]\debugger\mic\gdb_quickstart_win.pdf
```

4.1.4 Known Issues and Limitations

- Using conditional breakpoints for offload sections might stall the debugger. If a conditional breakpoint is created within an offload section, the debugger might hang when hitting it and evaluating the condition. This is currently analyzed and will be resolved in a future version of the product.
- Offload debugging is only supported in Microsoft Visual Studio 2012* and Microsoft Visual Studio 2013*.
- Data breakpoints are not yet supported within offload sections.
- Disassembly window cannot be scrolled outside of 1024 bytes from the starting address within an offload section.
- Handling of exceptions from the Intel® MIC Architecture application is not supported.
- Changing breakpoints while the application is running does not work. The changes will appear to be in effect but they are not applied.
- Starting an Intel® MIC Architecture native application is not supported. You can attach to a currently running application, though.
- The Thread Window in Microsoft Visual Studio* offers context menu actions to Freeze, Thaw and Rename threads. These context menu actions are not functional when the thread is on an Intel® Xeon Phi™ coprocessor.
- Setting a breakpoint right before an offload section sets a breakpoint at the first statement of the offload section. This only is true if there is no statement for the host between set breakpoint and offload section. This is normal Microsoft Visual Studio* breakpoint behavior but might become more visible with interweaved code from host and coprocessor. The superfluous breakpoint for the offload section can be manually disabled (or removed) if desired.
- Only Intel® 64 applications containing offload sections can be debugged with the Intel® Debugger Extension for Intel® Many Integrated Core Architecture.
- Stepping out of an offload section does not step back into the host code. It rather continues execution without stopping (unless another event occurs). This is intended behavior.
- The functionality “Set Next Statement” is not working within an offload section.
- If breakpoints have been set for an offload section in a project already, starting the debugger might show bound breakpoints without addresses. Those do not have an impact on functionality.
- For offload sections, setting breakpoints by address or within the Disassembly window won’t work.
- For offload sections, using breakpoints with the following conditions of hit counts do not work: “break when the hit count is equal to” and “break when the hit count is a multiple of”.
- The following options in the Disassembly window do not work within offload sections: “Show Line Numbers”, “Show Symbol Names” and “Show Source Code”
- Evaluating variables declared outside the offload section shows wrong values.
- Please consult the Output (Debug) window for detailed reporting. It will name unimplemented features (see above) or provide additional information required to

configuration problems in a debugging session. You can open the window in Microsoft Visual Studio* via menu `Debug->Windows->Output`.

- When debugging an offload-enabled application and a variable assignment is entered in the *Immediate Window*, the debugger may hang if assignments read memory locations before writing to them (for example, `x=x+1`). Please do not use the *Immediate Window* for changing variable values for offload-enabled applications.
- Depending on the debugger extensions provided by Intel, the behavior (for example, run control) and output (for example, disassembly) could differ from what is experienced with the Microsoft Visual Studio* debugger. This is because of the different debugging technologies implemented by each and should not have a significant impact to the debugging experience.

5 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about this version of Intel® Integrated Performance Primitives (Intel® IPP).

The latest information on Intel® IPP 8.2 can be found in the product release notes under `<install dir>\Documentation\<locale>\ipp\ReleaseNotes.htm`.

For detailed information about 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://intel.ly/OG5IF7; and the Intel IPP Release Notes at http://intel.ly/1uj984p.

- **Documentation, help, and samples**: see the documentation links on the IPP product page at: http://intel.ly/OG5IF7.

5.1 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://intel.ly/ndrGnR

6 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about this version of the Intel® Math Kernel Library (Intel® MKL). All the bug fixes can be found here: http://intel.ly/RGiGV9

6.1 Changes in This Version

6.1.1 What's New in Intel MKL 11.2 Update 1

- Introduced support of Intel® Advanced Vector Extensions 512 (Intel® AVX-512) on Intel® microarchitecture code name Skylake for Windows* and Linux* versions of Intel

MKL. This is in addition to the current support for Intel® AVX-512 instructions for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

- BLAS:
  - Optmized the following functions on Intel microarchitecture code name Skylake:
    - (D/Z)AXPY,(S/D/C/Z)COPY, DTRMM (for cases when the triangular matrix is on right side and with no matrix transpose)
  - Optimized the following BLAS Level-1 functions on Intel® Advanced Vector Extensions 2 (Intel® AVX2) both for Intel64 and IA-32 Architectures
    - (S/D)DOT,(S/D)SCAL,(S/D)ROT,(S/D)ROTM,(S/D/C/Z)SWAP,(S/D/SC/DZ)ASUM
  - Improved ?GEMM performance (serial and multithreaded) on Intel AVX2 (for IA-32 Architectures)
  - Improved ?GEMM performance for beta==0 on Intel AVX and Intel AVX2 (for Intel64 Architectures)
  - Improved DGEMM performance (serial and multithreaded) on Intel AVX (for Intel64 Architectures)
- LAPACK:
  - Introduced support for LAPACK version 3.5. New features introduced in this version are:
    - Symmetric/Hermitian LDLT factorization routines with rook pivoting algorithm
    - 2-by-1 CSD for tall and skinny matrix with orthonormal columns
  - Improved performance of (C/Z)GE(SVD/SDD) when M>=N and singular vectors are not needed
- FFT:
  - Introduced Automatic Offload mode for 1D Batch FFT on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
  - Improved performance of Hybrid (OpenMP+MPI) Cluster FFT
  - Improved accuracy of large 1D real-to-complex transforms
- Parallel Direct Sparse Solver for Clusters:
  - Added support for many factorization steps with the same reordering (maxfct > 1)
- Intel MKL PARDISO:
  - Added support for Schur complement, including getting explicit Schur complement matrix and solving the system through Schur complement
- Sparse BLAS:
  - Optimized SpMV on Intel microarchitecture code name Skylake
  - Added Sparse Matrix Checker functionality as standalone API to simplify validation of matrix structure and indices (see Sparse Matrix Checker Routines in the Intel® Math Kernel Library (Intel® MKL) Reference Manual)
  - Sparse BLAS API for C/C++ uses const modifier for constant parameters
- VML:
  - Introduced new Environment variable, MKL_VML_MODE to control the accuracy behavior. This Environment variable can be used to control VML functions behavior (analog of vmlSetMode() function)

6.1.2 What's New in Intel MKL 11.2

- Intel MKL now provides optimizations for all Intel® Atom™ processors that support Intel® Streaming SIMD Extensions 4.1 (Intel® SSE4.1) and Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2) instruction sets
- Introduced support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set with limited optimizations in BLAS, DFT and VML
- Introduced Verbose support for BLAS and LAPACK domains, which enables users to capture the input parameters to Intel MKL function calls
- Introduced support for Intel® MPI Library 5.0
- Introduced the Intel Math Kernel Library Cookbook (http://software.intel.com/en-us/mkl_cookbook) , a new document that describes how to use Intel MKL routines to solve certain complex problems
- Introduced the MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ compilation feature that provides ?GEMM small matrix performance improvements for all processors (see the Intel® Math Kernel Library User's Guide for more details)
- Added the ability to link a Single Dynamic Library (mkl_rt) on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Added a customizable error handler. See the Intel Math Kernel Library Reference Manual description of mkl_set_exit_handler() for further details
- Extended the Intel® Xeon Phi™ coprocessor Automatic Offload feature with a resource sharing mechanism.See the Intel Math Kernel Library Reference Manual for the description of mkl_mic_set_resource_limit() function and the MKL_MIC_RESOURCE_LIMIT environment variable for further details
- Parallel Direct Sparse Solver for Clusters:
    - Introduced Parallel Direct Sparse Solver for Clusters, a distributed memory version of Intel MKL PARDISO direct sparse solver
    - Improved performance of the matrix gather step for distributed matrices
    - Enabled reuse of reordering information on multiple factorization steps
    - Added distributed CSR format, support of distributed matrices, RHS, and distributed solutions
    - Added support of solving of systems with multiple right hand sides
    - Added cluster support of factorization and solving steps
    - Added support for pure MPI mode and support for single OpenMP* thread in hybrid configurations
- BLAS:
    - Improved threaded performance of ?GEMM for all 64-bit architectures supporting Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Optimized ?GEMM, ?TRSM, DTRMM for the Intel AVX-512 instruction set
    - Improved performance of ?GEMM for outer product [large m, large n, small k] and tall skinny matrices [large m, medium n, small k] on Intel MIC Architecture
    - Improved performance of ?TRSM and ?SYMM in Automatic Offload mode on Intel MIC Architecture
    - Improved performance of Level 3 BLAS functions for 64-bit processors supporting Intel AVX2

    - Improved ?GEMM performance on small matrices for all processors when MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ is defined during compilation (see the Intel® Math Kernel Library User's Guide for more details)
    - Improved performance of DGER and DGEMM for the beta=1, k=1 case for 64-bit processors supporting Intel SSE4.2, Intel® Advanced Vector Extensions (Intel® AVX), and Intel AVX2 instruction sets
    - Optimized (D/Z)AXPY for the Intel AVX-512 instruction set
    - Optimized ?COPY for Intel AVX2 and AVX512 instruction sets
    - Optimized DGEMV for Intel AVX-512 instruction set
    - Improved performance of SSYR2K for 64-bit processors supporting Intel AVX and Intel AVX2
    - Improved threaded performance of ?AXPBY for all Intel processors
    - Improved DTRMM performance for the side=R, uplo={U,L}, transa=N, diag={N,U} cases for Intel AVX-512
- LINPACK:
    - Improved performance of matrix generation in the heterogeneous Intel® Optimized MP LINPACK Benchmark for Clusters
    - Intel MIC Architecture offload option of the Intel Optimized MP LINPACK Benchmark for Clusters package now supports Intel AVX2 hosts
    - Improved performance of the Intel Optimized MP LINPACK for Clusters package for 64-bit processors supporting Intel AVX2
- LAPACK:
    - Improved performance of ?(SY/HE)RDB
    - Improved performance of ?(SY/HE)EV when eigenvectors are needed
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GELQF,?GELS and ?GELSS for underdetermined case (M less than N)
    - Improved performance of ?GEHRD,?GEEV and ?GEES
    - Improved performance of NaN checkers in LAPACKE interfaces
    - Improved performance of ?GELSX, ?GGSVP
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GETRF
    - Improved performance of (S/D)GE(SVD/SDD) when M>=N and singular vectors are not needed
    - Improved performance of ?POTRF UPLO=U in Automatic Offload mode on Intel MIC Architecture
    - Added Automatic Offload for ?SYRDB on Intel MIC Architecture, which speeds up ?SY(EV/EVD/EVR) when eigenvectors are not needed
- PBLAS and ScaLAPACK:
    - Enabled Automatic Offload in P?GEMM routines for large distribution blocking factors
- Sparse BLAS:

    - Optimized SpMV kernels for Intel AVX-512 instruction set
    - Added release example for diagonal format use in Sparse BLAS
    - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel SSE4.2, Intel AVX and Intel AVX2 instruction sets
- Intel MKL PARDISO:
    - Added the ability to store Intel MKL PARDISO handle to the disk for future use at any solver stage
    - Added pivot control support for unsymmetric matrices and out-of-core mode
    - Added diagonal extraction support for unsymmetric matrices and out-of-core mode
    - Added example demonstrating use of Intel MKL PARDISO as iterative solver for non-linear systems
    - Added capability to free memory taken by original matrix after factorization stage if iterative refinement is disabled
    - Improved memory estimation of out-of-core (OOC) portion size for reordering algorithm leading to improved factorization-solve performance in OOC mode
    - Improved message output from Intel MKL PARDISO
    - Added support of zero pivot during factorization for structurally symmetric cases
- Poisson library:
    - Added example demonstrating use of the Intel MKL Poisson library as a preconditioner for linear systems solves
- Extended Eigensolver:
    - Improved message output
    - Improved examples
    - Added input and output iparm parameters in predefined interfaces for solving sparse problems
- FFT:
    - Optimized FFTs for the Intel AVX-512 instruction set
    - Improved performance for non-power-of-2 length on Intel® MIC Architecture
- VML: Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
    - Added support of ntrial=0 in Binomial Random Number Generator
    - Improved performance of MRG32K3A and MT2203 BRNGs on Intel MIC Architecture
    - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
    - Added support for group/pooled mean estimates (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN)
- Data Fitting: Fixed incorrect behavior of the natural cubic spline construction function when number of breakpoints is 2 or 3
- Introduced an Intel MKL mode that ignores all settings specified by Intel MKL environment variables

- User can set up the mode by calling mkl_set_env_mode() routine which directs Intel MKL to ignore all environment settings specific to Intel MKL so that all Intel MKL related environment variables such as MKL_NUM_THREADS, MKL_DYNAMIC, MKL_MIC_ENABLE and others are ignored; users can instead set needed parameters via Intel MKL service routines such as mkl_set_num_threads() and mkl_mic_enable()

Known Limitations:

- Automatic Offload on Windows with large matrices may cause data corruption or crash. There is a problem in COI: HSD4868293 (critical). COI Cannot allocate a buffer with >= 2**32 bytes and 2M pages on Windows

  Workaround: Set MKL_MIC_MAX_MEMORY=3G. Note: This issue is resolved in Intel® MPSS 3.3

Note: API symbols, order of arguments and link line have changed since Intel MKL 11.2 Beta Update 2. See the Intel® Math Kernel Library User's Guide for more details.

Note: Important deprecations are listed in Intel® Math Kernel Library (Intel® MKL) 11.2 Deprecations

6.2 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 (http://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.

The Intel® MKL Extended Eigensolver functionality is based on the Feast Eigenvalue Solver 2.0 http://www.ecs.umass.edu/~polizzi/feast/

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. 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.

7 Intel® Threading Building Blocks

For information on changes in this version of Intel® Threading Building Blocks, please read the file `CHANGES` in the TBB documentation directory found in `<installdir>\Composer XE 2015\Documentation\<locale>\tbb`.

7.1 Known Issues

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

7.1.1 Library Issues

- 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.

8 Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR

PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to:

http://www.intel.com/products/processor%5Fnumber/

The Intel® C++ Compiler, Intel® Integrated Performance Primitives, Intel® Math Kernel Library, and Intel® Threading Building Blocks are provided under Intel’s End User License Agreement (EULA).

The GNU* Project Debugger, GDB is provided under the General GNU Public License GPL V3.

Please consult the licenses included in the distribution for details.

Celeron, Centrino, Intel, Intel logo, Intel386, Intel486, Atom, Core, Itanium, MMX, Pentium, VTune, Cilk, Xeon Phi, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others.