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

22 January 2015

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 2

- New string conversion functions added to libistrconv in 15.0.2
- -xCOMMON-AVX512 and -axCOMMON-AVX512 options added in 15.0.2
- -auto-p32 supported on OS X* and both –auto-p32 and –auto-ilp32 require –no-pie
- Intel® C++ Compiler 15.0.2

- GNU* Project Debugger (GDB*) 7.8
- Intel® Math Kernel Library 11.2 Update 2
- Intel® Threading Building Blocks 4.3 Update 3

1.1.2 Changes in Update 1

- OS X* 10.10 now supported
- Xcode* 6.0 and 6.1 now supported
- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- 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

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

- New Optimization Report interface, structure, and options New Optimization Report interface, structure, and options (strongly recommended to read for users of existing options -opt-report, -vec-report, -openmp-report, and -par-report)
- Full C++11 language support
- –stdlib=libc++ is now enabled by default
- Additional OpenMP* 4.0 features
- MIN/MAX Reductions supported in #pragma simd
- Intel® Cilk™ Plus changes
- Compiler options starting with –o are deprecated
- Select custom installation configurations with the online installer
- Enforce same code to be executed regardless of data alignment with –no-opt-dynamic-align
- Enable threadsafe profile generation with PGO
- aligned_new header
- Improved debugging of lambda functions
- Debug information now in DWARF Version 3 format by default
- 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
- _bittest and _bittestandcomplement intrinsics supported in Intel C++ Compiler 15.0
- Python* no longer a requirement to use GNU* Project Debugger
- OS X* 10.8 not supported
- Xcode* 4.6 not supported
- Intel® Debugger (IDB) is removed
- Intel® C++ Compiler 15.0.0

- GNU* Project Debugger (GDB*) 7.7
- Intel® Math Kernel Library 11.2
- Intel® Integrated Performance Primitives 8.2
- Intel® Threading Building Blocks 4.3

1.2 Product Contents

*Intel® Parallel Studio XE 2015 Update 2 Composer Edition for C++ OS X** includes the following components:

- Intel® C++ Compiler 15.0.2 for building applications that run on Intel-based Mac systems running the OS X* operating system
- GNU* Project Debugger (GDB*) 7.8
- Intel® Math Kernel Library 11.2 Update 2
- Intel® Integrated Performance Primitives 8.2 Update 1
- Intel® Threading Building Blocks 4.3 Update 3
- Integration into the Xcode* development environment
- On-disk documentation

1.3 Intel® Debugger (IDB) is removed from this release

The Intel Debugger (IDB) has been removed from this release. A debugger based on the GNU* Project Debugger (GDB*) is now provided for debugging.

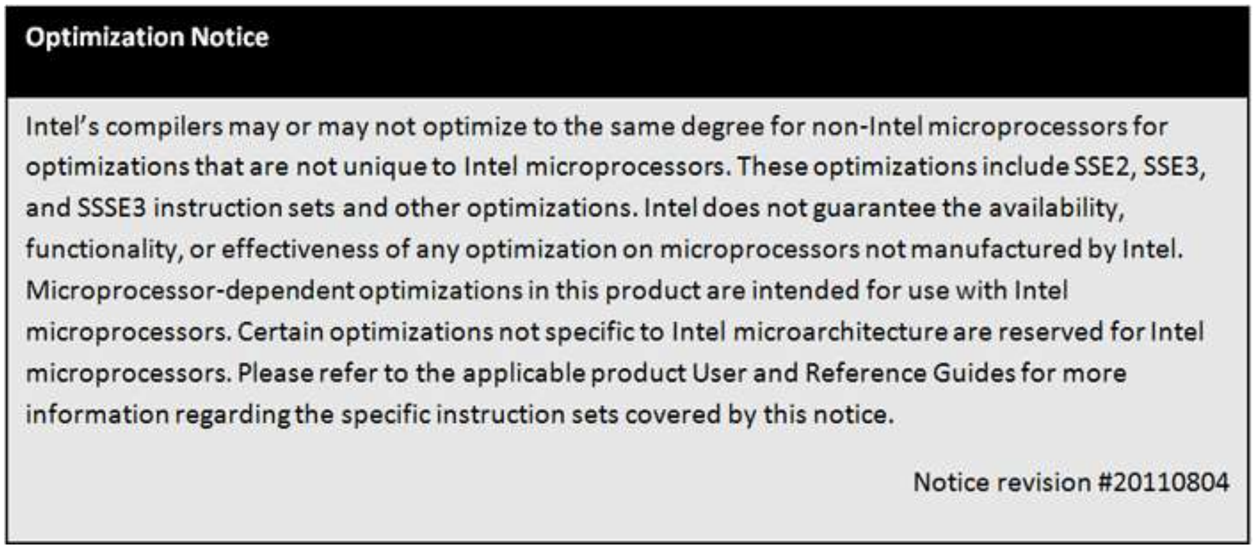
1.4 System Requirements

- A 64-bit Intel®-based Apple* Mac* system host (development for 32-bit is still supported)
- 2GB RAM minimum, 4GB RAM recommended
- 3GB free disk space
- One of the following combinations of OS X*, Xcode* and the Xcode SDK:
    - OS X 10.10 and Xcode* 6.0 or 6.1
    - OS X 10.9 and Xcode* 5.0 or 5.1
- If doing command line development, the Command Line Tools component of Xcode* is required

Note: Advanced optimization options or very large programs may require additional resources such as memory or disk space.

1.5 Documentation

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


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

1.6 Samples

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

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

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.

If you will be using Xcode*, please make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you must reinstall the Intel C++ Compiler afterwards.

The Command Line Tools component, required for command-line development, is not installed by default. It can be installed using the Components tab of the Downloads preferences panel.

You will need to have administrative or “sudo” privileges to install, change or uninstall the product.

To begin installation, double-click the downloaded .dmg file.

Follow the prompts to complete 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.

2.1 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 /tmp/intel/downloads.   This location may be changed with the INTEL_SWTOOLS_DOWNLOAD_DIR environment variable. 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 INTEL_SWTOOLS_DOWNLOAD_DIR environment variable.

2.2 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.

2.3 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.

2.4 Silent Install

For information on automated or “silent” install capability, please see http://intel.ly/1gcW0Bl

2.4.1 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:

- `export INTEL_SWTOOLS_DUPLICATE_MODE=config_file_name`: it specifies the configuration file name. If full path file name is specified, the INTEL_SWTOOLS_DOWNLOAD_DIR environment variable is ignored and the installable package will be created under the directory where the configuration file is.
- `export INTEL_SWTOOLS_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:
  ```
  /tmp/intel/downloads/<package_id>
  ```

2.5 Installation Folders

The compiler installs, by default, under `/opt/intel` – this is referenced as `<install-dir>` in the remainder of this document. You are able to specify a different location.

Under `<install-dir>` are the following directories:

- `bin` – contains symbolic links to executables for the latest installed version
- `lib` – symbolic link to the lib directory for the latest installed version
- `include` – symbolic link to the include directory for the latest installed version
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `ipp` – symbolic link to the directory for the latest installed version of Intel® Integrated Performance Primitives
- `mkl` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `tbb` – symbolic link to the directory for the latest installed version of Intel® Threading Building Blocks
- `ism` – contains files for Intel® Software Manager
- `composerxe` – symbolic link to the `composer_xe_2015` directory
- `composer_xe_2015` – directory containing symbolic links to subdirectories for the latest installed Intel® Parallel Studio XE 2015 Composer Edition compiler release
- `composer_xe_2015.<n>.<pkg>` - physical directory containing files for a specific compiler version. `<n>` is the update number, and `<pkg>` is a package build identifier.

Each `composer_xe_2015` directory contains the following directories that reference the latest installed Intel® Parallel Studio XE 2015 Composer Edition:

- `bin` – directory containing scripts to establish the compiler environment and symbolic links to compiler executables for the host platform
- `pkg_bin` – symbolic link to the compiler `bin` directory
- `include` – symbolic link to the compiler `include` directory
- `lib` – symbolic link to the compiler `lib` directory
- `ipp` – symbolic link to the `ipp` directory
- `mkl` – symbolic link to the `mkl` directory
- `tbb` – symbolic link to the `tbb` directory
- `debugger` – symbolic link to the `debugger` directory
- `man` – symbolic link to the `man` directory
- `Documentation` – symbolic link to the `Documentation` directory
- `Samples` – symbolic link to the `Samples` directory

Each `composer_xe_2015.<n>.<pkg>` directory contains the following directories that reference a specific update of the Intel® Parallel Studio XE 2015 Composer Edition :

- `bin` – all executables
- `pkg_bin` – symbolic link to `bin` directory
- `compiler` – shared libraries and header files
- `debugger` – debugger files
- `Documentation` – documentation files
- `man` – symbolic link to the `man` directory
- `ipp` – Intel® Integrated Performance Primitives libraries and header files
- `mkl` – Intel® Math Kernel Library libraries and header files
- `tbb` – Intel® Threading Building Blocks libraries and header files
- `Samples` – Product samples and tutorial files

If you have both the Intel C++ and Intel Fortran compilers installed, they will share folders for a given version and update.

This directory layout allows you to choose whether you want the latest compiler, no matter which version, the latest update of the Intel® Parallel Studio XE 2015 Composer Edition compiler, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh [.csh]` script, which will always get the latest compiler installed.

2.6 Relocating Product After Install

The Xcode integration is relocatable simply by dragging and dropping the Xcode directory tree to another location.

2.7 Removal/Uninstall

It is not possible to remove the compiler while leaving any of the performance library components installed.

1. Open Terminal and set default (`cd`) to any folder outside `<install-dir>`
2. Type the command:
   `<install-dir>/composer_xe_2015.<n>.<pkg>/uninstall_ccompxe.sh`
3. Follow the prompts

If you are not currently logged in as `root` you will be asked for the `root` password.

3 Intel® C++ Compiler

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

3.1 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.

- New string conversion functions added to libistrconv in 15.0.2
- -xCOMMON-AVX512 and -axCOMMON-AVX512 options added in 15.0.2
- In Intel® C++ Compiler 15.0.2, -auto-p32 supported on OS X* and both –auto-p32 and –auto-ilp32 require –no-pie
- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- MIN/MAX Reductions supported in #pragma simd
- _bittest and _bittestandcomplement intrinsics supported
- New Optimization Report interface, structure, and options
- Full C++11 language support (includes these feature new to 15.0) (-std=c++11)
  - Value categories (N3055)
  - alignas and alignof (N2341)
  - decltype extensions (N3049, N3276)
  - Inheriting constructors (N2540)
  - User-defined literals (N2765)
  - thread_local (N2659)
- Additional OpenMP* 4.0 features
- Intel® Cilk™ Plus changes
- aligned_new header
- Improved debugging of lambda functions
- Debug information now in DWARF Version 3 format by default
- 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

3.1.1 New string conversion functions added to libistrconv in 15.0.2

The following functions have been added to Intel's Numeric String Conversion Library (libistrconv) for the Intel® C++ Compiler 15.0.2. For more details, see the User's Guide.

- `__IML_f_to_str`
- `__IML_d_to_str`
- `__IML_f_to_str_f`
- `__IML_d_to_str_f`
- `__IML_f_to_str_e`
- `__IML_d_to_str_e`
- `__IML_i_to_str`
- `__IML_u_to_str`
- `__IML_ll_to_str`
- `__IML_ull_to_str`

3.1.2 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.

3.1.3 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];
```

3.1.4 _bittest and _bittestandcomplement intrinsics supported in Intel® C++ Compiler 15.0

Support is now provided for the _bittest and _bittestandcomplement intrinsics that are currently supported for Windows*.

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

The four kinds of optimization reports (-opt-report, -vec-report, -openmp-report, and -par-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 -vec-report, -openmp-report, and -par-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`.

3.1.6 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.

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

Please note the following new features for Intel® Cilk™ Plus in 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))
- New __attribute__((vector_variant(…))) to define a vector-specific overload of a scalar function.
- Intel Cilk Plus documentation can now be generated using Doxygen*. See the compiler/include/cilk/ReadMe.html for more details.

3.1.8 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.

3.1.9 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 New and Changed Compiler Options

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

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

- -xCOMMON-AVX512
- -axCOMMON-AVX512
- -auto-p32
- -xCORE-AVX512
- -axCORE-AVX512
- -q[no-]opt-multi-version-aggressive
- -qopt-ra-region-strategy[=keyword]
- -qopt-report-names=[mangled|unmangled]
- -q[no-]opt-jump-tables=<arg>
- -qopt-block-factor=<n>
- -qopt-streaming-stores=<keyword>
- -qopt-subscript-in-range
- -q[no-]opt-mem-layout-trans[=<level>]
- -q[no-]opt-prefecth[=n]
- -q[no-]opt-dynamic-align
- -qopenmp
- -qopenmp-stubs
- -qopenmp-lib=<ver>
- -qopenmp-link=<library>
- -qopenmp-task=<arg>
- -qopenmp-threadprivate=<ver>
- -prof-gen=threadsafe
- -qopt-report
- -qopt-report-file={stdout | stderr | <file>}
- -qopt-report-per-object
- -qopt-report-phase=<phase>[,<phase>,…]
- -qopt-report-routine=<name>[,<name>,…]

- -qopt-report-filter=<string>
- -qopt-report-format=[text|vs]
- -q[no-]opt-report-embed
- -no-gcc-include-dir
- -std=gnu++11
- -debug [no]emit-column
- -fast and –Ofast now include –fp-model fast=2
- -f[no-]eliminate-unused-debug-types
- -I-

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

3.2.2 -xCOMMON-AVX512 and -axCOMMON-AVX512 options added in 15.0.2

The COMMON-AVX512 selection has been added for the -x and -ax compiler options. This allows generation of the subset of Intel® Advanced Vector Extensions 512 (Intel® AVX-512) that are supported on both Intel® Many Integrated Core Architecture and on Intel® Core™ Microarchitecture. This includes Intel AVX-512 Foundation instructions and Intel AVX-512 Conflict Detection instructions.

3.2.3 In Intel® C++ Compiler 15.0.2, -auto-p32 supported on OS X* and both –auto-p32 and –auto-ilp32 require –no-pie

Starting in the Intel® C++ Compiler 15.0.2, -auto-p32 is supported on OS X* as well as Linux*. Also, on OS X*, both the -auto-p32 and -auto-ilp32 options require Position Independent Executable code to be disabled. This is done with the -no-pie option. If -no-pie is not specified, -auto-p32 and -auto-ilp32 will be ignored.

3.2.4 Compiler options starting with –o are deprecated

All compiler options starting with –o are deprecated. These will be replaced by new options preceded with –q. For example, -opt-report should now be –qopt-report. This is to improve compatibility with third-party tools that expect –o<text> to always refer to output filenames.

3.2.5 –stdlib=libc++ is now enabled by default in Intel® C++ Compiler 15.0

The option –stdlib=libc++ to enable the use of the libc++ standard C++ library is now on by default.

3.2.6 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"`".

3.2.7 Enforce same code to be executed regardless of data alignment with –no-opt-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 –no-opt-dynamic-align

3.2.8 Enable threadsafe profile generation with PGO

To enable the safe generation of profile information in multithreaded applications, use the -prof-gen=threadsafe option.

3.3 Other Changes

3.3.1 Environment Setup Script Changed

The `compilervars.sh` script is used to establish the compiler environment.

The command takes the form:

```
source <install-dir>/bin/compilervars.sh argument
```

Where *`argument`* is either `ia32` or `intel64` as appropriate for the architecture you are building for. Establishing the compiler environment also establishes the environment for the provided GNU* GDB (gdb-ia), Intel® Performance Libraries and, if present, Intel® Fortran Compiler.

3.3.2 OpenMP* Legacy Libraries Removed

The OpenMP "legacy" libraries have been removed in this release. Only the "compatibility" libraries are provided.

3.4 Sample Notes

3.4.1 Building Tachyon

There are a couple common problems that may come up in the course of building Tachyon. If you use the provided Makefile to do a command-line build of the Tachyon sample, you may get errors about not finding directories or finding pbxcp. If you get these, go to the file common/gui/makefile.gmake, look for where the two variables `XCODE_SDK_SYSROOT` and `PBXCP` are set and change them to point to the location of your OS X 10.8 SDK and the pbxcp binary respectively.

For building from Xcode*, you may run into problems building the build_with_tbb configuration with llvm gcc*. The problem will be that the libtbb.dylib cannot be found. In this case, go to the `Summary->Linked Frameworks and Libraries` section, and manually add the libtbb.dylib library from the `composer_xe_2015.x.xxx/compiler/lib` directory.

3.5 Known Limitations

3.5.1 Incompatible with the default libc++ library

Some applications are incompatible with libc++, which is currently enabled as default C++ library in the Intel® C++ Compiler 15.0. For example, the 435.gromacs and 447.dealII from SPEC CPU2006 suite.

Please use `-stdlib=libstdc++` to compile such applications.

3.5.2 "Build" or "Run" in Xcode* 5 do not accurately detect and rebuild modified dependencies

When the Intel® C++ Compiler is selected for a project in Xcode* 5, there is an issue where if a file that other files/projects depend on is modified, for example a header file, then the dependent files or libraries will not be automatically rebuilt with the "Build" or "Run" commands. Instead, a "Clean" needs to be done first to do a full rebuild. This issue is being investigated.

3.5.3 Creating new project in Xcode* causes hardcoding of –stdlib=libc++

A new project created in Xcode* causes the hardcoding of a setting for –stdlib=libc++ even for projects that have the Intel® C++ Compiler toolset added. So setting the Intel® C++ Compiler field for the C++ Standard Library setting to libstdc++ is ineffective because libc++ overrides the setting. To change this, do the following:

1. Select the project row in the navigator area at the left of the workspace window
2. In the project editor that appears, select the row that represents the project level of build settings
3. You should see the `C++ Standard Library` setting in bold, indicating that it has a custom value in this project
4. Select that row and press the Delete key to remove the customized value
5. The `C++ Standard Library` build property should now have the value `Compiler Default`

Note that you may have to follow the above steps before adding the Intel® C++ Compiler toolset to your project build rules.

4 GNU* Project Debugger (GDB)

This section summarizes the changes, new features, customizations and known issues related to the GNU* GDB provided with Intel® Parallel Studio XE 2015.

4.1 Features

GNU* GDB provided with Intel® Parallel Studio XE 2015 Update 2 Composer Edition and above is based on GDB 7.8 with enhancements provided by Intel. This debugger replaces the Intel® Debugger from previous releases. In addition to features found in GDB 7.8, there is an additional feature:

- Support for Intel® Transactional Synchronization Extensions (Intel® TSX)

4.2 Using GNU* GDB

This debugger is designed to debug IA-32 or Intel® 64 applications natively. Its use is no different than with traditional GNU* GDB debuggers. There are some extensions, though, which can be found in the documentation.

Instructions on how to use GNU* GDB can be found in the Documentation section.

4.3 Documentation

The documentation for the provided GNU* GDB can be found here:
`<install-dir>/Documentation/en_US/debugger/gdb/gdb.pdf`
`<install-dir>/Documentation/en_US/debugger/gdb/gdb_quickstart_lin.pdf`

4.4 Known Issues

4.4.1 Debugger and debugged application required to be located on local drive

In order to use the provided GNU* GDB (`gdb-ia`), it has to be installed on a local drive. As such, the entire Intel Parallel Studio XE package has to be installed locally. Any application that is being debugged needs to be located on a local drive as well. This is a general requirement that's inherent to GNU GDB with OS X*.

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>/composer_xe_2015.x.xxx/Documentation/<locale>/ipp/ReleaseNotes.htm`.

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://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 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 2

- BLAS:
    - Improved parallel and serial performance of ?HEMM/?SYMM for on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for the 64-bit Intel MKL.
    - Improved parallel and serial performance of ?HERK/?SYRK and and ?HER2K/?SYR2K for Intel AVX2.
    - Added MKL_DIRECT_CALL support for CBLAS interfaces and ?GEMM3M routines.
    - Improved CGEMM performance for Intel® Advanced Vector Extensions 512 (Intel® AVX-512).
    - Improved SGEMM and ZGEMM performance for AMD* Opteron* 6000 series processors.
    - Small performance improvement for CGEMM and ZGEMM for Intel AVX2 for the 64-bit Intel MKL.
- LAPACK:
    - Improved symmetric eigensolvers performance by up to 3x, for the cases when eigenvectors are not needed.
    - Improved ?GESVD performance by 2-3x, for the cases when singular vectors are required.
    - Improved ?GETRF performance for Intel AVX2 by up to 14x for non-square matrices.
    - Narrowed the ?GETRF performance gap between CNR (Conditional Numerical Reproducibility)-enabled and CNR-disabled cases. The gap is now below 5%.
    - Improved Intel® Optimized LINPACK Benchmark shared memory (SMP) implementation performance for Intel AVX2 by up to 40%.
- Intel® MKL PARDISO:
    - Improved the scalability of the solving step for Intel® Xeon® processors.
    - Reduced memory footprint in the out-of-core mode.
    - Added ability to free up memory used by the input matrix after the factorization step. This helps to reduce memory consumption when iterative refinement is not needed and disabled by the user.
- Extended Eigensolver:
    - Improved performance for Intel Xeon processors
- VSL:
    - Summary Statistics:

        - Improved performance of variance-covariance matrices computation and correlation matrices computation routines for cases when the task dimension is approximately equal to the number of observations.
    - RNG:
        - Improved performance of the Sobol and the Niederreiter Quasi-RNGs for Intel Xeon processors.
- Convolution and correlation:
    - Improved 3D convolution performance.

6.1.2 What's New in Intel MKL 11.2 Update 1

- BLAS:
    - 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:
    - 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:
    - 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)

6.1.3 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 a customizable error handler. See the Intel Math Kernel Library Reference Manual description of mkl_set_exit_handler() 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 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 Intel AVX-512 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
- 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
- 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 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()

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 to Intel® Threading Building Blocks (Intel® TBB), please read the file `CHANGES` in the Intel® TBB documentation directory found in `<installdir>/composer_xe_2015.x.xxx/Documentation/<locale>/tbb`.

8 Disclaimer and Legal Information