Intel® C++ Composer XE 2013 SP1 for Windows* Installation Guide and Release Notes

Document number: 321414-005US
23 January 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.

Due to the nature of this comprehensive integrated software development tools solution, different Intel® C++ Composer XE 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

- Intel® C++ Compiler XE 14.0.2
- Intel® Math Kernel Library 11.1 update 2
- Intel® Integrated Performance Primitives 8.1
- Intel® Threading Building Blocks 4.2 update 3
- Microsoft Visual Studio 2013* supported
- New Intel® Cilk™ Plus STL vector reducer in Intel® C++ Composer XE 2013 SP1 update 2
- KMP_DYNAMIC_MODE Environment Variable Support for "asat" Deprecated
- Corrections to reported problems

1.1.2 Changes in Update 1

- Windows 8.1 supported
- Intel® C++ Compiler XE 14.0.1
- First 14.0 version with Japanese localization
- Intel® Math Kernel Library 11.1 update 1
- Intel® Integrated Performance Primitives 8.0 update 1
- Intel® Threading Building Blocks 4.2 update 1
- New Numeric String Conversion Library: `libistrconv`
- New compiler option /`Qopt-gather-scatter-unroll (opt-gather-scatter-unroll)` for targeting Intel® MIC Architecture
- New compiler option `/Q[a]xMIC-AVX512(-[a]xMIC-AVX512)`
- New compiler option `/Qcheck-pointers-mpx (-check-pointers-mpx)` to support the Intel® Memory Protection Extensions (Intel® MPX)
- "`uniform(this)`" is now allowed in Intel® Cilk™ Plus SIMD enabled function (e.g., `__declspec(vector(uniform(this)))`)
- New intrinsic `_allow_cpu_features`

1.1.3 Changes since Intel® C++ Composer XE 2013

- Online installation
- Intel® C++ Compiler XE 14.0.0
- Support for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Features from C++11 (-std=c++11)
- Partial OpenMP* 4.0 support
- Intel® Cilk™ Plus changes
- __INTEL_COMPILER_UPDATE predefined macro
- Pointer type alignment qualifiers
- Variable definition attributes to avoid false sharing

- Using offload code in shared libraries requires main program to be linked with –offload=mandatory or –offload=optional option
- Limitations of __Cilk_shared
- /Qopenmp-offload and /Qopenmp-simd options added for controlling the enabling/disabling of specific OpenMP* 4.0 features independently of other OpenMP features
- /QxATOM_SSE4.2 option added to support Silvermont microarchitecture
- Intel® Debugging Extension 1.0 for Intel® MIC Architecture
- Intel® Math Kernel Library updated to version 11.1
- Intel® Integrated Performance Primitives updated to version 8.0 update 1
- Intel® Threading Building Blocks updated to version 4.2
- Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

1.2 Product Contents

*Intel® C++ Composer XE 2013 SP1 Update 2 for Windows** includes the following components:

- Intel® C++ Compiler XE 14.0.2 for building applications that run on IA-32 or Intel® 64 architecture systems or Intel® Xeon Phi™ coprocessors running the Windows* operating system
- Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Intel® Integrated Performance Primitives 8.1
- Intel® Math Kernel Library 11.1 update 2
- Intel® Threading Building Blocks 4.2 update 3
- 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
- 1GB RAM (2GB 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

- Microsoft Windows XP*, Microsoft Windows 7*, Microsoft Windows 8*, Microsoft Windows 8.1*, Microsoft Windows Server 2008*, Microsoft Windows HPC Server 2008*, or Microsoft Windows Server 2012* (embedded editions not supported)
    - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2010* or Visual Studio 2008* SP1.
    - 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 [1]
    - Microsoft Visual Studio 2008* Standard Edition (or higher edition) with C++ and "X64 Compiler and Tools" components installed [1]
- 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
    - Microsoft Visual C++ 2008* 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 2008 Standard Edition installs the "x64 Compiler and Tools" component by default – the Professional and higher editions require a "Custom" install to select this. 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. Intel® C++ Composer XE 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 XE 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 Experimenting with Intel C++ Composer XE for Windows and Windows 8 Store Apps (http://intel.ly/WLeXRo).

1.3.1 Visual Studio 2008* is Deprecated

Support for Visual Studio 2008* has been deprecated and will be removed in a future release.

1.3.2 Windows XP* is Deprecated

Support for Windows XP* has been deprecated and will be removed in a future release.

1.3.3 IA-64 Architecture (Intel® Itanium®) Development Not Supported

This product version does not support development on or for IA-64 architecture (Intel® Itanium®) systems. The version 11.1 compiler remains available for development of IA-64 architecture applications.

1.3.4 Windows Server 2003* and Windows Vista* Not Supported

Support has been removed for installation and use on Windows Server 2003 and Windows Vista. Intel recommends migrating to a newer version of these operating systems.

1.3.5 Visual Studio 2005* Not Supported

Support has been removed for installation and use with Visual Studio 2005. Intel recommends migrating to a newer version of Visual Studio*.

1.4 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

1.5 Samples

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

1.6 Japanese Language Support

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

Japanese language support will be available in an update on or after the release of Intel® C++ Composer XE 2013 SP1.

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 in Intel® Composer XE 2013 SP1

The default electronic installation package for Intel® Composer XE 2013 SP1 now consists of 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.

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® Composer XE 2013 SP1 for Windows* including Intel® MIC Architecture product.

Using the latest version of Intel® MPSS available is recommended.

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

2.4.1 Configure Visual Studio for 64-bit Applications

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

If you are using Visual Studio 2008* Standard Edition, or Visual Studio 2010* Professional Edition or higher, no configuration is needed to build Intel® 64 architecture applications. For other editions:

1. From Control Panel > Add or Remove Programs, select “Microsoft Visual Studio 2008” > Change/Remove. The Visual Studio Maintenance Mode window will appear. Click Next.
2. Click Add or Remove Features
3. Under “Select features to install”, expand Language Tools > Visual C++

4. If the box “X64 Compiler and Tools” is not checked, check it, then click Update. If the box is already checked, click Cancel.

Note that Visual C++ Express Edition does not support 64-bit development.

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.

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

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

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

2.5.1 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

2.5.3 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 2013 SP1
  - bin
    - ia32
    - ia32_gfx
    - ia32_intel64
    - intel64
    - intel64_mic
    - sourcechecker
  - compiler
    - include
      - cilk
      - ia32
      - intel64
      - mic
    - lib
      - ia32
      - intel64
      - mic
        - locale
          - en_US
          - ja_JP
    - perf_headers
      - C++
  - Debugger
    - gdb
      - LICENSES
      - src
      - target
        - mic
          - bin
          - lib
          - share
            - man
              - man1

- w64_mic
  - bin
  - include
    - gdb
  - lib
  - share
    - gdb
      - python
        - gdb
          - command
          - function
      - syscalls
    - info
    - locale
    - man
      - man1
- debuggerextension
  - mic
    - scripts

- Documentation
  - en_US
    - compiler_c
      - cl
    - debugger
      - gdb
        - pdf
    - gs_resources
    - ipp
      - get_started_files
      - ipp_userguide
      - tutorials
    - mkl
      - get_started_files
      - mkl_userguide
      - tutorials
    - ssadiag_docs
    - tbb
      - get_started_files
      - html
      - tutorial
    - tutorials
      - cmp_gap_c
      - cmp_thd_c

- cmp_vec_c
- msvhelp
  - 1033
    - compiler_c
    - ipp
    - mkl
    - ssadiag
    - tbb
  - vshelp
    - intel.cppprodocs
    - intel.cprocompilerdocs
    - intel.ippdocs
    - intel.mkldocs
    - intel.sssadiag
    - intel.tbbdocs
- Help
- ipp
  - bin
  - examples
  - include
  - interfaces
  - lib
  - tools
- mkl
  - benchmarks
  - bin
  - examples
  - include
  - interfaces
  - lib
  - tests
  - tools
- redist
  - ia32
    - compiler
      - 1033
      - irml
      - irml_c
    - ipp
      - 1033
    - mkl
      - 1033
    - tbb

                - vc_mt
                - vc9
                - vc10
                - vc11
        - intel64
            - compiler
                - 1033
                - irml
                - irml_c
            - ipp
                - 1033
            - mkl
                - 1033
            - tbb
                - vc_mt
                - vc9
                - vc10
                - vc11
- Samples
    - en_US
        - C++
        - Ipp
        - mkl
- tbb
    - bin
    - examples
    - include
        - serial
            - tbb
        - tbb
            - compat
            - internal
            - machine
    - lib
        - ia32
            - vc_mt
            - vc9
            - vc10
            - vc11
        - intel64
            - vc_mt
            - vc9
            - vc10

- vc11
- VS Integration
  - C++
    - VS2008

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 2013.`*`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

C++ Composer XE 2013 SP1 now contains Intel® C++ Compiler XE 14.0. The following features are new or significantly enhanced in this version. For more information on these features, please refer to the documentation.

- New Intel® Cilk™ Plus STL vector reducer in Intel® C++ Composer XE 2013 SP1 update 2
- New intrinsic _allow_cpu_features in Intel® C++ Composer XE 2013 SP1 update 1
- The `this` pointer is now allowed in the Intel® Cilk™ Plus SIMD-enabled function `uniform` clause (i.e. `__declspec(vector(uniform(this)))` in Intel® C++ Composer XE 2013 SP1 update 1
- New Numeric String Conversion Library libistrconv in Intel® C++ Composer XE 2013 SP1 update 1
- Support for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Features from C++11 (-std=c++11)

    - Complete (instead of partial) implementation of initializer lists. See N2672 and N3217.
    - Complete implementation of inline namespaces. See N2535.
    - Complete implementation of non-static data member initializers. See N2756.
    - Complete implementation of generalized constant expressions. See N2235.
    - Complete implementation of unrestricted unions. See N2544.
    - Delegating constructors. See N1986.
    - Rvalue references for *this. See N2439.
    - Raw string literals. See N2442.
    - Conversions of lambdas to function pointers.
    - Implicit move constructors and assignment operators. See N3053.
    - __bases and __direct_bases type traits.
    - The context-sensitive keyword "final" can now be used on a class definition, and "final" and "override" can be used on member function declarations. See N2928, N3206, and N3272.
    - Complete implementation of the "noexcept" specifier and operator. See N3050. Includes the late instantiation of noexcept per core issue 1330.
- Partial OpenMP* 4.0 support
- Intel® Cilk™ Plus changes
- __INTEL_COMPILER_UPDATE predefined macro
- Pointer type alignment qualifiers
- Variable definition attributes to avoid false sharing

3.2.1 New Intel® Cilk™ Plus STL vector reducer in Intel® C++ Composer XE 2013 SP1 update 2

In update 2, a `reducer_vector` class is now provided. The header file `"cilk/reducer_vector.h"` will need to be included. The reducer type is `cilk::reducer< cilk::op_vector<type> >`. See the header file comments for further specifics.

3.2.2 New intrinsic `_allow_cpu_features` in Intel® C++ Composer XE 2013 SP1 update 1

This new intrinsic `_allow_cpu_features([xxx][,xxx])` is added to `immintrin.h`. It tells the compiler that the code region following it may be targeted for processors with the specified feature(s) so some specific optimizations may be performances.

Note: support of this intrinsic is preliminary, not all of the compilers optimization phases can be taken place in the code region.

Please reference Compiler documentation for detailed information with code sample, and the article New intrinsic _allow_cpu_features support for additional information.

3.2.3 The `this` pointer is now allowed in the Intel® Cilk™ Plus SIMD-enabled function uniform clause (i.e. `__declspec(vector(uniform(this)))`) in Intel® C++ Composer XE 2013 SP1 update 1

When a uniform class object calls a SIMD-enabled class member function, explicitly specifying "`uniform(this)`" clause in the callee's SIMD declaration may improve performance (how much depends on how the "`this`" keyword is used inside the callee). The usage model is the same as the uniform clause applied to formal parameters.

Please reference Compiler documentation for detailed information.

3.2.4 New Numeric String Conversion Library `libistrconv` in Intel® C++ Composer XE 2013 SP1 update 1

This New Numeric String Conversion Library, `libistrconv`, provides a collection of routines for converting between ASCII strings and C data types, which are optimized for performance. The new APIs are declared in the header file "`istrconv.h`".

Please reference Compiler documentation for detailed information.

3.2.5 Updated Support for Upcoming OpenMP* features added in Composer XE 2013 SP1

Composer XE 2013 SP1 adds partial support for OpenMP* 4.0 features. The features supported as defined in the OpenMP* 4.0 specifications available from http://openmp.org are:

- TEAMS pragmas, directives and clauses
- DISTRIBUTE pragmas, directive and clauses
- SIMD pragmas, directives, and clauses
- TARGET pragmas, directives and clauses for attached coprocessors (or devices)
- `#pragma omp taskgroup` construct
- Atomic clause `seq_cst`
- Six new forms of atomic capture and update:
    - `Atomic swap: {v = x; x = expr;}`
    - `Atomic update: x = expr binop x;`
    - `Atomic capture 1: v = x = x binop expr;`
    - `Atomic capture 2: v =x = expr binop x;`
    - `Atomic capture 3: {x = expr binop x; v = x;}`
    - `Atomic capture 4: {v = x; x = expr binop x;}`
- `proc_bind(<type>)` clause where <type> is "spread", "close", or "master"
- OMP_PLACES environment variable
- OMP_PROC_BIND environment variable
- `omp_get_proc_bind()` API

For more information, see http://intel.ly/W7CHjb.

3.2.6 Intel® Cilk™ Plus changes in Intel® C++ Composer XE 2013 SP1

Please note the following new features for Intel® Cilk™ Plus in Intel C++ Composer XE 2013 SP1:

- SIMD enabled function implementation has changed to be more compatible with other vector function implementations in gcc and OpenMP*. This breaks binary compatibility with previous Intel® C++ Compiler versions (13.1 and earlier). You should either rebuild all codes using SIMD enabled functions with the version 14.0 compiler, or use the /Qvecabi:legacy compiler option to use the previous implementation.
- New multiply reducer defined in `cilk/reducer_opmul.h`
- Three new array notation reduction intrinsics have been added to support bitwise reduction operations:
  - `__sec_reduce_and`
  - `__sec_reduce_or`
  - `__sec_reduce_xor`

3.2.7 New attribute for pointers and pointer types to specify assumed data alignment in Composer XE 2013 SP1

__declspec(align_value(N)) and __attribute__((align_value(N))) have been added to indicate to the compiler it can assume the specified alignment “N” when using the attributed pointer type. For example:

```
typedef float float_a16
__attribute__((align_value (16)));

void foo(float_a16 *restrict dest, float_a16 *restrict src){
```

Let’s the compiler know that the src and dest arguments should be aligned by the user on 16-byte boundaries.

3.2.8 New attribute to variable declarations to avoid false sharing in Composer XE 2013 SP1

__declspec(avoid_false_share)/__attribute__((avoid_false_share)) and __declspec(avoid_false_share(identifier))/__attribute__((avoid_false_share(identifier))) have been added to indicate to the compiler that the variable attributed should be suitably padded or aligned to avoid false sharing with any other variable. If an identifier is specified, then any variables attributed with that identifier will be padded or aligned to avoid false sharing with any other variables except those others with the same identifier. These attributes must be on variable definitions in function, global, or namespace scope. If in function scope, the scope of the identifier is the current function. If the variable definition is in global or namespace scope, the scope of the identifier is in the current compilation unit.

3.2.9 New __INTEL_COMPILER_UPDATE predefined macro in Composer XE 2013 SP1

A new __INTEL_COMPILER_UPDATE predefined macro can now be used to obtain the minor update number for the Intel® Compiler being used. For example, for a compiler version 14.0.2, the macro would preprocess to “2”.

3.2.10 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.

3.2.11 Intel® C++ Project File Compatibility

The Intel C++ project file (`.icproj`) format changed in version 14.0 (Composer XE 2013 SP1). 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 14.0 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 compiler option `/Qcheck-pointers-mpx (-check-pointers-mpx)` to support the Intel® Memory Protection Extensions (Intel® MPX) (Update 1)

This option will cause the compiler to generate code which uses the Intel® Memory Protection Extensions (Intel® MPX) for performance acceleration of Pointer Checker. If the target platform does not support Intel® MPX, Pointer Checker features will operate as no-ops. See the Introduction to Intel® Memory Protection Extensions for some details about Intel® MPX.

Please see Compiler Options section of the documentation for detailed information about this new option.

3.3.2 New compiler option /Q[a]xMIC-AVX512(-[a]xMIC-AVX512) for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions support (Update 1)

This option may generate Intel(R) AVX-512 Foundation instructions, Intel(R) AVX-512 Conflict Detection instructions, Intel(R) AVX-512 Exponential and Reciprocal instructions, Intel(R) AVX-512 Prefetch instructions for Intel(R) processors and the instructions enabled with CORE-AVX2. It lets compiler to optimize for Intel(R) processors that support Intel(R) AVX-512 instructions.

Please see Compiler Options section of the documentation for detailed information, and see AVX-512 instructions article for more about the new instructions.

3.3.3 /Qopt-gather-scatter-unroll(-opt-gather-scatter-unroll) for targeting Intel® MIC Architecture (Update 1)

This option lets you specify an alternative loop unroll sequence for gather and scatter loops on Intel® MIC Architecture. It is only available on Intel® 64 architecture targeting Intel® MIC Architecture.

Please see Compiler Options section of the documentation for detailed information.

3.3.4 New and Changed in Composer XE 2013 SP1

- /Qopenmp-offload[-]
- /Qopenmp-simd[-]
- /QxATOM_SSE4.2
- /QxATOM_SSSE3
- /Qvecabi:<arg>
- /Qpar
- /pdbfile[:filename]
- /nopdbfile
- /Wpch-messages[-]
- /Qmic
- /Qoffload[:keyword]
- /Qoffload-attribute-target:target-name
- /Qoffload-option, target, tool, “option-list”
- /Qopt-assume-safe-padding
- /Qopt-prefetch-distance:n1[,n2]
- /Qopt-streaming-cache-evict[:n]
- /Qopt-threads-per-core:n

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

3.3.5 /Qopenmp-offload[-] and /Qopenmp-simd[-] added to Composer XE 2013 SP1

These two options allow you to enable/disable the TARGET and SIMD features of OpenMP* 4.0 independently of support of the rest of OpenMP* (enabled with /Qopenmp). When /Qopenmp is specified, /Qopenmp-offload and /Qopenmp-simd are set as well, allowing the use of these features.

3.3.6 /Wpch-messages[-] added to Composer XE 2013 SP1

Functionality to enable or disable diagnostics related to pre-compiled headers has been added to Composer XE 2013 SP1.

3.3.7 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 2013 > Command Prompt > Parallel Studio XE with Intel Compiler XE v14.0 [Update xx] > IA-32/Intel® 64 Visual Studio xxx mode
2) Run command:

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

3.4 Other Changes

3.4.1 KMP_DYNAMIC_MODE Environment Variable Support for "asat" Deprecated

Support for "`asat`" (automatic self-allocating threads) by the environment variable `KMP_DYNAMIC_MODE` is now deprecated, and will be removed in a future release.

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`

`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`
- `vs2008`

If you also have Intel® Visual Fortran Composer XE 2013 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 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.`"

3.5.1.2 Missing documentation for functions to check decimal floating-point status

To detect exceptions occurring during decimal floating-point arithmetic, use the following floating-point exception functions:

Function	Brief Description
fe_dec_feclearexcept	Clears the supported floating-point exceptions
fe_dec_fegetexceptflag	Stores an implementation-defined representation of the states of the floating-point status flags
fe_dec_feraiseexcept	Raises the supported floating-point exceptions
fe_dec_fesetexceptflag	Sets the floating-point status flags
fe_dec_fetestexcept	Determines which of a specified subset of the floating point exception flags are currently set

The decimal floating-point exception functions are defined in the fenv.h header file.

Similar binary floating-point exception functions are described in ISO C99.

To compile the source using DFP, use the preprocessor macro __STDC_WANT_DEC_FP__.

3.5.1.3 Command-Line Diagnostic Issue for Filenames with Japanese Characters

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

3.5.2 Visual Studio Known Issues

3.5.3 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`".

3.5.3.1 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.

3.5.3.2 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.

3.5.3.3 Language packs of Visual Studio 2010

If you install a new language pack of Visual Studio 2010 after installing the Intel C++ Composer XE 2013, 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 14.0\1033`" exists, copy all files to "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 14.0\<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 C++ Composer XE 2013 and reinstall the Intel C++ Composer XE 2013.

3.5.4 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`.

3.5.5 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`.

3.5.6 Static Analysis Known Issues

3.5.6.1 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 Developing Applications for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

This section summarizes changes, new features and late-breaking news about the Intel Composer XE 2013 for Windows* including Intel® MIC Architecture.

4.1 About Intel® Composer XE 2013 SP1 for Windows* including Intel® MIC Architecture

The Intel® Composer XE 2013 SP1 for Windows* including Intel® MIC Architecture extends the feature set of the Intel® C++ Composer XE 2013 and the Intel® Fortran Composer XE 2013 products by enabling predefined sections of code to execute on an Intel® Xeon Phi™ coprocessor.

These sections of code run on the coprocessor if it is available. Otherwise, they run on the host CPU.

This document uses the terms *coprocessor* and *target* to refer to the target of an offload operation.

The current components of Intel® Composer XE 2013 SP1 that support Intel® MIC Architecture are the:

- Intel® C++ and Fortran Compilers
- Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Intel® Math Kernel Library (Intel® MKL)
- Intel® Threading Building Blocks (Intel® TBB)
- Visual Studio* IDE Integration

4.2 Getting Started

There is only one compiler that generates code both for Intel® 64 architecture and for Intel® MIC Architecture. Ensure you are using the compiler for Intel® 64 architecture from the command line or via the pre-provided shell in the Start Menu, or using an “x64” configuration in the Microsoft Visual Studio IDE*. Refer to the Notes section below for further changes.

4.3 Product Documentation

Documentation concerning the Intel® MIC Architecture for Composer XE 2013 SP1 is currently undergoing change. For the latest documentation updates, please go to our web site at http://intel.ly/MxPFYx.

4.4 Intel® Math Kernel Library (Intel® MKL)

For details on Intel® MIC Architecture support, see the section on Intel MKL.

4.5 Notes

4.6 Intel C++ Compiler

4.6.1 Limitations of `_Cilk_shared` in Composer XE 2013 SP1

- 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)

4.6.2 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.

4.6.3 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"
```

4.6.4 *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;
```

4.6.5 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.

4.6.6 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.

4.6.7 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`"

4.7 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).

4.7.1 Features

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

4.7.2 Using the Intel® Debugger Extension

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

Before starting Microsoft Visual Studio IDE, additional environment variables need to be set in order to debug offload enabled Intel® Xeon Phi™ applications. Depending on the version of the Intel® Manycore Platform Software Stack (Intel® MPSS) set the following variables:

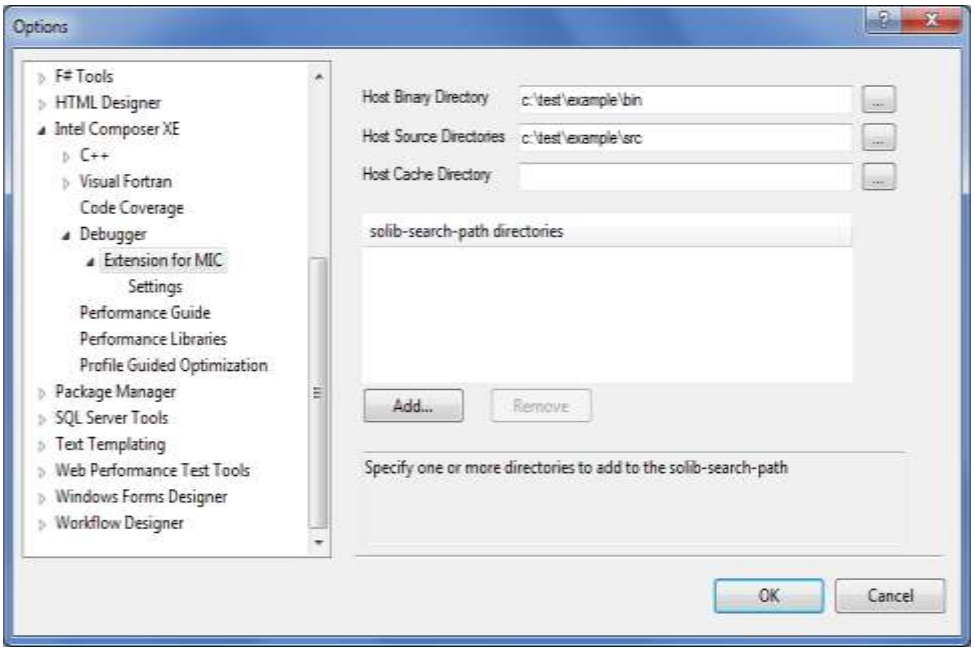
- Intel® MPSS 3.1:
  ```
  AMPLXE_COI_DEBUG_SUPPORT=TRUE
  MYO_WATCHDOG_MONITOR=-1
  ```
- Intel® MPSS 2.1:
  ```
  COI_SEP_DISABLE=FALSE
  MYO_WATCHDOG_MONITOR=-1
  ```

4.7.2.1 Loading and executing an application for Intel® Xeon Phi™ Coprocessor

This is only possible for applications using any of the two the offload models. Open a project for Intel® Xeon Phi™ coprocessor and start a debug session via the menu `Debug->Start Debugging`. The scope of coprocessors and information shown during the debug session (threads, breakpoints, etc.) depends on the use of the offload model. Hence, if the offload model is not used, no code is executed on the coprocessor.

4.7.2.2 Attaching to Process on Intel® Xeon Phi™ Coprocessor

This is only possible for applications running natively on the Intel® Xeon Phi™ coprocessor. The process to debug needs to run already on the coprocessor in order to continue.
Next, the process binary needs to be specified for Microsoft Visual Studio* IDE. This is done via menu `Tools->Options…`:



For “Host Binary Directory” specify the directory containing (a copy of) the binary that is to be debugged on the coprocessor. The root directory for finding the sources can be specified by “Host Source Directory”. Optionally search paths to shared libraries for the coprocessor application can be added in the “solib-search-path directories” text field. The semantic is the same as for GNU* GDB.

4.7.3 Documentation

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

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

4.7.4 Known Issues and Limitations

- Offload debugging is only supported in Microsoft Visual Studio 2012* or 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.
- Under certain conditions, the Thread Window does not show all threads running on the coprocessor.
- 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 a 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.
- Under certain conditions, debugging into offloaded code can stop the execution. The source position seems to be the offload directive but the execution is actually inside a library routine. You should be able to continue execution from that point.
- 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.

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 8.0 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/OmWI4d.

- **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

5.2 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages at http://intel.ly/pnsHxc

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

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/OeHQqf

6.1 Notices

Please refer to the Knowledge Base article on Deprecations for more information on the following notices

- Intel® MKL now provides a choice of components to install. Components necessary for PGI* compiler, Compaq Visual Fortran* Compiler, SP2DP interface, BLAS95 and LAPACK95 interfaces, Cluster support (ScaLAPACK and Cluster DFT) and Intel® Many Integrated Core Architecture (Intel® MIC Architecture) support are not installed unless explicitly selected during installation
- Unaligned Conditional Numerical Reproducibility (CNR) is not available for Intel MKL Cluster components (ScaLAPACK and Cluster DFT)
- Examples for using Intel MKL with Boost* uBLAS and Java* have been removed from product distribution and placed in the following articles:
    - How to use Intel MKL with Java*
    - How to use Boost* uBLAS with Intel MKL

6.2 Changes in This Version

6.2.1 What's New in Intel MKL 11.1 update 2

- Introduced support for Intel® Atom™ processors
- BLAS:
    - Improved performance of ?GEMM for m==1 or n==1 on all Intel architectures
    - Improved MP LINPACK performance for systems using Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
    - 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 ?SYMM on Intel MIC Architecture
    - Improved {S/D}GEMM single thread performance on small matrices for 64-bit processors supporting Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Improved DGEMV performance for 64-bit processors supporting Intel AVX2
    - Improved threaded performance of {S,D,C,Z}GEMV for notrans:n>>m and trans:m>>n on all Intel architectures
    - Improved DSYR2K performance for 64-bit processors supporting Intel AVX and Intel AVX2
    - Improved DTRMM performance on small matrices (A matrix size <= 10) for 64-bit processors supporting Intel AVX and Intel AVX2
    - Reduced stack usage for ZHEMM and ZSYRK
    - Added more detailed error messages for running Offload MP LINPACK scripts with unsupported configurations
- LAPACK:
    - Improved performance of (S/D)SYRDB and (D/S)SYEV for large dimensions and UPLO=L when eigenvectors are needed
    - Improved performance of ?GELQF,?GELS and ?GELSS for underdetermined case (M
    - Improved performance of ?GEHRD,?GEEV and ?GEES
    - Added Automatic Offload to Intel® Xeon Phi™ coprocessor for DSYRDB UPLO=L
- Sparse BLAS:
    - Optimized SpMV kernels for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set
    - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2), Intel AVX, and Intel AVX2 instruction sets
- PARDISO:
    - Improved memory estimation of out-of-core portion size for reordering algorithm leading to improved factorization-solving step performance in OOC mode
- VML:
    - Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
    - Improved performance of MRG32K3A, and MT2203 BRNGs on Intel Xeon Phi coprocessors
    - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
    - Added support for computation of group/pooled (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN) mean estimates

6.2.2 What's New in Intel MKL 11.1 update 1

- Introduced support for Intel® AVX-512 instructions set with limited set of optimizations
- BLAS:
    - Improved performance of DSDOT, and added support for multiple threads, on all 64-bit Intel processors supporting Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Improved handling of denormals on the diagonal in *TRSM
    - Improved SGEMM performance for small N and large M and K on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
    - Improved parallel performance of *HEMM on all Intel processors supporting Intel® SSE4.2 and later
    - Improved parallel performance of 64-bit *SYRK/*HERK on all Intel processors supporting Intel® SSSE3 and later
    - Improved serial performance of 64-bit {D,S}SYRK on all Intel processors supporting Intel® SSE4.2 and later
    - Improved performance of DTRSM on Intel® MIC Architecture
    - Enhanced Intel® Optimized HPL Benchmark runmultiscript capabilities for Intel processors supporting Intel® AVX
    - Improved Intel® Optimized HPL Benchmark performance on Intel® MIC Architecture
- LAPACK
    - Decreased memory utilization for parallel LAPACK functions (OR/UN)M(QR/RQ/QL/LQ)
    - Decreased stack memory utilization in LAPACK functions
    - Improved performance of (S/D)SYRDB and (S/D)SYEV for large dimensions when eigenvalues are only needed
- ScaLAPACK
    - Updated PBLAS headers to mix default NETLIB and MKL complex datatypes
- DFT: Optimized complex-to-complex and real-to-complex transforms
- Transposition: Improved performance of mkl_?omatcopy routines on tall and skinny matrices
- DFTI interface and FFTW wrappers are now thread safe. setting NUMBER_OF_USER_THREADS parameter when using MKL DFT from parallel regions became optional.

6.2.3 What's New in Intel MKL 11.1

- Conditional Numerical Reproducibility : Introduced support for Conditional Numerical Reproducibility (CNR) mode on unaligned data
- Intel MKL now supports compiler assisted offload and Automatic offload programming model on Intel Xeon Phi™ coprocessors based on the Intel® Many Integrated Core Architecture (Intel® MIC Architecture) on Windows* OS*
- Improved performance of CNR=AUTO mode on recent AMD* systems
- BLAS:
    - Improved performance of [S/D]GEMV on all Intel processors supporting Intel® SSE4.2 and later
    - Optimized [D/Z]GEMM and double-precision Level 3 BLAS functions on Intel® Advanced Vector Extensions 2 (Intel® AVX2)

    - Optimized [Z/C]AXPY and [Z/C]DOT[U/C] on Intel® Advanced Vector Extensions (Intel® AVX) and Intel AVX2
    - Optimized sequential version of DTRMM on Intel MIC Architecture
    - Tuned DAXPY on Intel AVX2
- LAPACK:
    - Improved performance of (S/D)SYRDB and (S/D)SYEV for large dimensions when only eigenvalues are needed
    - Improved performance of xGESVD for small sizes like M,N<10
- VSL:
    - Added support and examples for mean absolute deviation
    - Improved performance of Weibull Random Number Generator (RNG) for alpha=1
    - Added support of raw and central statistical sums up to the 4th order, matrix of cross-products and median absolute deviation
    - Added a VSL example designed by S. Joe and F. Y. Kuo illustrating usage of Sobol QRNG with direction numbers which supports dimensions up to 21,201
    - Improved performance of SFMT19937 Basic Random Number Generator (BRNG) on Intel MIC Architecture
- DFT:
    - Improved performance of double precision complex-to-complex transforms on Intel MIC Architecture
    - Optimized complex-to-complex DFT on Intel AVX2
    - Optimized complex-to-complex 2D DFT on Intel® Xeon processor E5 v2 series
    - Improved performance for workloads specific to GENE application on Intel Xeon E5-series (Intel AVX) and on Intel AVX2
    - Improved documentation data layout for DFTI compute functions
    - Introduced scaling in large real-to-complex FFTs
- Data Fitting:
    - Improved performance of df?Interpolate1D and df?SearchCells1D functions on Intel Xeon processors and Intel MIC Architecture
    - Improved performance of df?construct1d function for linear and Hermite/Bessel/Akima cubic types of splines on Intel MIC Architecture, Intel® Xeon® processor X5570 and Intel® Xeon® processor E5-2690
- Transposition
    - Improved performance of in-place transposition for square matrices
- Examples and tests for using Intel MKL are now packaged as an archive to shorten the installation time
- Link Tool and Link Line advisor: Added support for Intel MIC Architecture on Windows* OS

6.3 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.

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

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.