Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux* Installation Guide and Release Notes

6 August 2015

Table of Contents

1 Introduction

This document describes how to install the product, provide 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 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 changes 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 5

- Intel® Fortran Compiler updated to version 15.0.4
- Intel® Math Kernel Library 11.2 Update 4
- Corrections to reported problems

1.1.2 Changes in Update 4

- No update 4 for this product

1.1.3 Changes in Update 3

- Intel® Fortran Compiler updated to version 15.0.3
- Intel® Math Kernel Library 11.2 Update 3
- Corrections to reported problems

1.1.4 Changes in Update 2

- New Library Support for Assignment with Allocatable Polymorphic Components
- Support for Fedora 21* added
- Intel® Fortran Compiler updated to version 15.0.2
- Intel® Math Kernel Library 11.2 Update 2
- GNU* Project Debugger (GDB*) 7.8 for IA-32/Intel® 64 architecture
- Corrections to reported problems

1.1.5 Changes in Update 1

- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- First update with Japanese Localization
- Support for SuSE Linux Enterprise Server 12* has been added
- MIN/MAX Reductions supported in SIMD Loop Directive
- Intel® Fortran Compiler updated to version 15.0.1
- Intel® Math Kernel Library 11.2 Update 1
- Corrections to reported problems

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

- Intel® Fortran Compiler updated to version 15.0
  - New Optimization Report interface, structure, and options (users of existing options -opt-report, -vec-report, -openmp-report, and -par-report are strongly encouraged to consult the Intel Compiler User's Guide for additional details)
- Intel® Math Kernel Library updated to version 11.2
- Python* no longer a requirement to use GNU* Project Debugger
- GNU* GDB 7.7 with improved Fortran Support
- Intel® Debugger has been removed
- Static analysis is deprecated
- Compiler options starting with –o are deprecated
- binutils 2.24 supported
- Red Hat Enterprise Linux* 7 now supported
- Support for Ubuntu* 13.10, 14.04 LTS, Fedora* 20 added.
- Support for the following Linux distributions has been dropped:
  - Fedora 18, 19*
  - Ubuntu 13.04*
  - SuSE Linux Enterprise Server 10*
- Additional OpenMP* 4.0 features
- Select custom installation configurations with the online installer
- Enable threadsafe profile generation with PGO
- New INTEL_PROF_DYN_PREFIX environment variable to add custom prefix to PGO .dyn filenames
- Corrections to reported problems

1.2 Product Contents

Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux Update 5* includes the following components:

- Intel® Fortran Compiler XE 15.0.4 for building applications that run on IA-32, Intel® 64 architecture systems and Intel® Xeon Phi™ coprocessors running the Linux* operating system
- GNU* Project Debugger (GDB*) 7.7 Intel® Xeon Phi™ coprocessor debugger
- GNU* Project Debugger (GDB*) 7.8 IA-32/Intel® 64 debugger
- Intel® Math Kernel Library 11.2 Update 4
- 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

For an explanation of architecture names, see Intel® Architecture Platform Terminology

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later, or compatible non-Intel processor)
    - Development of 64-bit applications, and those that offload work to Intel® Xeon Phi™ coprocessors, is supported on a 64-bit version of the OS only. Development of 32-bit applications is supported on either 32-bit or 64-bit versions of the OS.
    - Development for a 32-bit on a 64-bit host may require optional library components (ia32-libs, lib32gcc1, lib32stdc++6, libc6-dev-i386, gcc-multilib, g++-multilib) to be installed from your Linux distribution.
- For the best experience, a multi-core or multi-processor system is recommended
- 2GB of RAM (4GB recommended)
- 4GB free disk space for all features
- For Intel® Xeon Phi™ coprocessor development/testing:
    - Intel® Xeon Phi™ processor
    - Intel® Manycore Platform Software Stack (Intel® MPSS)
- For development of IA-32 or Intel® 64 architecture applications, one of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to Technical Support if you have questions):
    - Debian* 6.0, 7.0
    - Fedora* 20, 21
    - Red Hat Enterprise Linux* 5, 6, 7
    - SuSE LINUX Enterprise Server* 11,12
    - Ubuntu* 12.04 LTS (64-bit only), 13.10, 14.04 LTS
    - Intel® Cluster Ready

Note: Support for Red Hat Enterprise Linux 5* and Debian 6.0* is deprecated and will be removed in a future release.

- Linux Developer tools component installed, including gcc, g++ and related tools. (this is the list of component versions tested by Intel; other versions may or may not work and are not recommended - please refer to Technical Support if you have questions
    - gcc 4.1-4.8
    - binutils 2.17-2.24
- Library `libunwind.so` is required in order to use the `-traceback` option. Some Linux distributions may require that it be obtained and installed separately.

Notes

- The Intel® compilers are tested with a number of different Linux distributions, with different versions of gcc. Some Linux distributions may contain header files different from those we have tested, which may cause problems. The version of glibc you use must be consistent with the version of gcc in use. For best results, use only the gcc versions as supplied with distributions listed above.
- 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, Intel® MKL requires Intel® SSE2 as a minimum instruction set.
- Compiling very large source files (several thousands of lines) using advanced optimizations such as -O3, -ipo and -qopenmp, may require substantially larger amounts of RAM.
- Some optimization options have restrictions regarding the processor type on which the application is run. Please see the documentation of these options for more information.

1.5 Documentation

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

1.6 Optimization Notice

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

1.7 Japanese Language Support

Intel® compilers optionally provide support for Japanese language users when the combined English-Japanese product is installed. 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 is not provided with every update of the product.

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 Changing Language Setting to see English on a Japanese OS environment or Vice Versa on Linux*

1.8 Technical Support

Register your license at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

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

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

2 Installation

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

After downloading the product, first unpack the downloaded file into a writeable directory of your choice using the command:

```
tar -xzvf name-of-downloaded-file
```

Then change the directory (`cd`) to the directory containing the unpacked files and begin the installation using the command:

```
./install.sh
```

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.

Please do not run the install script as a background process (i.e. running "./install.sh &"). This is not supported.

2.1 GUI installation

If on a Linux* system with GUI support, the installation will now provide a GUI-based installation. In environments where a GUI is not supported (for example if running from an ssh terminal), the installation defaults to a command-line installer.

2.2 Online Installer

The default electronic installation package 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. The online installer may be downloaded and saved as shell script which can then be launched from the command line.

2.2.1 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/<UID>. This location may be changed with the online installer command line option "--download-dir [FOLDER]". The online installer also supports a download only mode which allows the user to create a package without installation. This mode is enabled with the "--download-only" command line option.

2.2.2 `http_proxy` is set, but sudo installation still fails to connect

Most sudo profiles are set to not inherit certain settings like `http_proxy` from the original user. Make sure your /etc/sudoers file contains a line like the following to allow your proxy settings to propagate:

```
Defaults   env_keep += "http_proxy"
```

2.3 Cluster Installation

If a license for Intel® Parallel Studio XE Cluster Edition is present, and the installation detects that the installing system is a member of a cluster, you will have the option of installing on multiple nodes of the cluster.

To install on multiple nodes, follow these steps:

1. Passwordless ssh must be configured among the nodes of the cluster
2. During install step 4, “Options”, select “Cluster installation”.
3. You will be prompted to provide the path to a `machines.LINUX` file with IP addresses, hostnames, or Fully Qualified Domain Names (FQDNs) of the cluster nodes, one per line. The first line is expected to be the current (master) node.
4. Once the `machines.LINUX` file is found, additional options will appear, including “Number of parallel installations” and “Check for shared installation directory”. Select the desired options.
5. Once all options are configured and the install is started, the installation will check connectivity to all the nodes; if successful, it will attempt the install on all indicated nodes.

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

The Intel® Manycore Platform System Software (Intel® MPSS) should be installed only if you plan to build applications that target an Intel® Xeon Phi™ coprocessor. Intel® MPSS may be installed before or after installing the *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux** product.

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

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

2.5 Intel® Software Manager

The installation provides the 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 Intel® Software Improvement Program.

2.6 Silent Install

For information on automated or “silent” install capability, please see Intel® Compilers for Linux* Silent Installation Guide.

2.6.1 Support of Non-Interactive Custom Installation

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

- `--duplicate config_file_name (or -d config_file_name)`: it specifies the configuration file name. If full path name is specified, "`--download-dir`" is ignored and the installable package will be created under the same directory as the configuration file.
- `--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/<UID>/<package_id>`

For example:

l_ccompxe_online_2015.0.0XX.sh --duplicate=icc15_install_config.ini ↵
--download-dir "/temp/custom_pkg_ic15"

The configuration file and installable package will be created under "`/temp/custom_pkg_ic15`".

2.7 Using a License Server

If you have purchased a "floating" license, see Licensing: Setting Up the Client for a Floating License. This article also provides a source for the Intel® License Manager for FLEXlm* product that can be installed on any of a wide variety of systems.

2.8 Known Installation Issues and changes

- When uninstalling a more recent major version of *Intel® Parallel Studio XE Composer Edition* , the symbolic links will be removed even if earlier versions of *Intel® Parallel Studio XE Composer Edition* exist on the system. For example, uninstalling *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux** will remove the links if this is the only 2015 update installed on the system, regardless if Composer XE 2013 SP1 is also installed. As a workaround, refer to the older Composer XE's directories explicitly (for example, `composer_xe_2013_sp1.<n>.<pkg>`)
- Remote offline activation via a code has been removed. Use of a license file or a license manager remain as options for alternative activation.
- Default permissions of online installer bootstrap scripts depend on default permissions based on the umask procedure and security setup of the downloading tool. If the downloaded online installer bootstrap scripts do not have execution permission, users should do chmod +x "installer bootstrap scripts"
- The Intel® Software Manager is not supported on Red Hat Enterprise Linux* 5.0. This limitation will be corrected in a future update.

2.9 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, and can also perform a "non-root" install in the location of your choice.

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
- `mkl` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `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 for Fortran Linux** release
- `composer_xe_2015.<n>.<pkg>` - physical directory containing files for a specific update 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 for Fortran Linux** product:

- `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
- `mkl` – symbolic link to the `mkl` directory
- `debugger` – symbolic link to the `debugger` directory
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `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 for Fortran Linux** compiler:

- `bin` – all executables
- `compiler` – shared libraries and include/header files
- `debugger` – debugger files
- `Documentation` – documentation files
- `man` – `man` pages
- `mkl` – Intel® Math Kernel Library libraries and header files
- `mpirt` – Intel® MPI Library run-time files used by Fortran coarray support
- `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 product update, no matter which version, the latest update of the *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux** product, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh [.csh]` script, which will always get the latest product installed. This layout should remain stable for future releases.

2.10 Removal/Uninstall

Removing (uninstalling) the product should be done by the same user who installed it (root or a non-root user). If `sudo` was used to install, it must be used to uninstall as well. It is not possible to remove the compiler while leaving any of the performance library components installed.

1. Open a terminal window and set default (`cd`) to any folder outside `<install-dir>`
2. Type the command: `<install-dir>/uninstall.sh`
3. Follow the prompts
4. Repeat steps 2 and 3 to remove additional platforms or versions

If you also have the same-numbered version of Intel® C++ Compiler installed, it may also be removed.

3 Intel® Fortran Compiler

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

3.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel Fortran Compiler for Linux* (8.0 and later) may be used in a build with version 15. Exceptions include:

- Sources that use the CLASS keyword to declare polymorphic variables and which were built with a compiler version earlier than 12.0 must be recompiled.
- Objects built with the multi-file interprocedural optimization (-ipo) option must be recompiled with the current version.
- Objects that use the REAL(16), REAL*16, COMPLEX(16) or COMPLEX*32 datatypes and which were compiled with versions earlier than 12.0 must be recompiled.
- Objects built for the Intel® 64 architecture with a compiler version earlier than 10.0 and that have module variables must be recompiled. If non-Fortran sources reference these variables, the external names may need to be changed to remove an incorrect leading underscore.
- Modules that specified an ATTRIBUTES ALIGN directive outside of a derived type and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.
- Modules that specified an ATTRIBUTES ALIGN directive inside a derived type declaration cannot be used by compilers older than 13.0.1.

3.1.1 Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes

In versions prior to 12.0, when a REAL(16) or COMPLEX(16) (REAL*16 or COMPLEX*32) item was passed by value, the stack address was aligned at 4 bytes. For improved performance, the version 12 and later compilers align such items at 16 bytes and expects received arguments to be aligned on 16-byte boundaries. This change is also compatible with gcc.

This change primarily affects compiler-generated calls to library routines that do computations on REAL(16) values, including intrinsics. If you have code compiled with earlier versions and link it with the version 12 libraries, or have an application linked to the shared version of the Intel run-time libraries, it may give incorrect results.

In order to avoid errors, you must recompile all Fortran sources that use the REAL(16) and COMPLEX(16) datatypes if they were compiled by compiler versions earlier than 12.0.

3.1.2 New Library Support for Assignment with Allocatable Polymorphic Components (15.0.2)

In order to resolve incorrect behavior in some circumstances when an assignment is done, either explicitly or implicitly, of a derived type containing components that are allocatable and polymorphic, a new compiler support library routine was created for version 15.0.2 and is called by the generated code. This means that if a Fortran source that has such an assignment is compiled with the 15.0.2 (or newer) compiler, it must be linked with the Fortran libraries from 15.0.2 or later. Failure to do this may lead to link errors regarding a missing routine `for_alloc_assign_v2`. In general you should always link with a library version no older than the compiler.

3.2 New and Changed Features

Some language features may not yet be described in the compiler documentation. Please refer to the Fortran 2003 Standard (PDF)) and Fortran 2008 Standard (PDF) if necessary.

3.2.1 Features from Fortran 2003

- Parameterized Derived Types

3.2.2 Features from Fortran 2008

- BLOCK construct
- intrinsic subroutine EXECUTE_COMMAND_LINE

3.2.3 Features from OpenMP*

The following directives, clauses and procedures, from OpenMP 4.0, are supported by the compiler. Some of these features were supported in Intel® Fortran Composer XE 2013 Update 2 based on a preliminary specification, some syntax supported earlier (DECLARE TARGET MIRROR, DECLARE TARGET LINKABLE, MAPTO, MAPFROM, SCRATCH) is no longer supported, and some syntax has changed its meaning since the earlier specification.

For more information, see the compiler documentation or the link to the OpenMP Specification above.

SIMD Directives:

- OMP SIMD
- OMP DECLARE SIMD
- OMP DO SIMD
- OMP PARALLEL DO SIMD

Coprocessor Directives:

- OMP TARGET DATA
- OMP TARGET
- OMP TARGET UPDATE
- OMP DECLARE TARGET

Other Directives:

- OMP PARALLEL PROC_BIND
- OMP TASKGROUP
- OMP CANCEL
- OMP CANCELLATION POINT

Clauses:

- MAP
- DEPEND

Procedures:

- OMP_GET_DEVICE_NUM
- OMP_GET_PROC_BIND
- OMP_SET_DEVICE_NUM

3.2.3.1 KMP_PLACE_THREADS Environment Variable (13.1.0)

This environment variable allows the user to simplify the specification of the number of cores and threads per core used by an OpenMP application, as an alternative to writing explicit affinity settings or a process affinity mask.

3.2.3.2 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.

3.2.4 Coarrays and Intel® Xeon Phi™ Coprocessors

Support has been added to support development of applications using coarrays that run either natively on Intel® Xeon Phi™ coprocessors or run in a mixed mode with Intel® Xeon Phi coprocessors and an Intel® 64 architecture host system

For more information, see Coarrays with Intel® Xeon Phi™ Coprocessors.

3.2.5 New and Changed Directives

The following compiler directives are new or changed in *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux** – please see the documentation for details:

- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-TOTAL-SIZE=N
- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-PER-ROUTINE=N

3.2.5.1 ATTRIBUTES STDCALL now allowed with BIND(C)

As of compiler version 15.0, the ATTRIBUTES STDCALL directive may be specified for an interoperable procedure (a procedure whose declaration includes the BIND(C) language binding attribute.)

No other effects from STDCALL, such as pass-by-value, are provided. The Fortran standard VALUE attribute (not ATTRIBUTES VALUE) may be used if desired. For all other platforms, specifying STDCALL with BIND(C) has no effect.

3.2.6 Other Features

For information on these features, please see the compiler documentation.

- New environment variable INTEL_PROF_DYN_PREFIX. Allows the user to have some control over the naming PGO generated ".dyn" files to make it easy to distinguish files from different runs. By setting this environment variable to the desired character string prior to starting the instrumented application, the string will be included as prefix to the .dyn file names.
- New intrinsic __intel_simd_lane() that represents the 'lane id' within a SIMD vector. This feature supports writing short-vector hyperobject reducer implementation. It also enables the performing of reduction operations inside SIMD-enabled functions.

3.2.7 Change in File Buffering Behavior (13.1)

In product versions prior to Intel® Fortran Composer XE 2013 (compiler version 13.0), the Fortran Runtime Library buffered all input when reading variable length, unformatted sequential file records. This default buffering was accomplished by the Fortran Runtime Library allocating an internal buffer large enough to hold any sized, variable length record in memory. For extremely large records this could result in an excessive use of memory, and in the worst cases could result in available memory being exhausted. The user had no ability to change this default buffering behavior on such READs. There was always the ability to request or deny buffering of these records when writing them, but not when reading them.

This default buffering behavior was changed with the release of Intel® Fortran Composer XE 2013. Beginning with this version, all such records are **not** buffered by default, but rather read directly from disk to the user program's variables. This change helped programs that needed to conserve memory, but could in fact result in a performance degradation when reading records that are made of many small components. Some users have reported this performance degradation.

The Intel® Fortran Composer XE 2013 Update 2 (compiler version 13.1) release of the Fortran Runtime Library now provides a method for a user to choose whether or not to buffer these variable length, unformatted records. The default behavior remains as it was in 13.0; these records are not buffered by default. If you experience performance degradation when using 13.1 with this type of I/O, you can enable buffering of the input the same way that you choose whether to enable buffering of the output of these records – one of the following:

- specifying BUFFERED="YES" on the file's OPEN statement
- specifying the environment variable FORT_BUFFERED to be YES, TRUE or an integer value greater than 0
- specifying `-assume buffered_io` on the compiler command line

In the past, these mechanisms applied only when issuing a WRITE of variable length, unformatted, sequential files. They can now be used to request that the Fortran Runtime Library buffer all input records from such files, regardless of the size of the records in the file.

Using these mechanisms returns the READing of such records to the pre-13.0 behavior.

3.2.8 Static Analysis has been deprecated

Static analysis is deprecated. It may be removed in a future major release. If you have concerns or feedback, please comment.

3.2.9 New run-time routines to get Fortran library version numbers

- FOR_IFCORE_VERSION returns the version of the Fortran run-time library (ifcore).
- FOR_IFPORT_VERSION returns the version of the Fortran portability library (ifport).

3.2.10 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, and/or the -[a]xCORE-AVX512 compiler options. This is in addition to the current support for Intel® AVX-512 instructions for Intel® Many Integrated Core Architecture.

3.2.11 MIN/MAX Reductions supported in SIMD Loop Directive

Starting with the Intel® Compilers version SIMD Loop Directive now supports MIN/MAX reductions, like so:

```
!DIR$ SIMD REDUCTION(MAX:SIMDMAX)
      DO I = 1, SIZE
```

```
         IF (X(I) > SIMDMAX) SIMDMAX = X(I)
      END DO


!DIR$ SIMD REDUCTION(MIN:SIMDMIN)
      DO I = 1, SIZE
         IF (X(I) < SIMDMIN) SIMDMIN = X(I)
      END DO



!DIR$ SIMD REDUCTION(MAX:XMAX)
      DO I = 1, SIZE
         XMAX = MAX (XMAX, X(I))
      END DO


!DIR$ SIMD REDUCTION(MIN:XMIN)
      DO I = 1, SIZE
         XMIN = MIN (XMIN, X(I))
      END DO
```

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- -assume [no]std_value
- -assume ieee_fpe_flags
- –[no-]opt-dynamic-align
- -f[no-]fat-lto-objects
- -f[no-]eliminate-unused-debug-types
- -fast
- -init=snan
- -qopt-report
- -prof-gen=[no]threadsafe

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

3.3.1 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.

3.3.2 -assume std_value is now the default

As of compiler version 15.0, the Fortran standard VALUE attribute, (not ATTRIBUTES VALUE), when specified for a dummy argument of a non-interoperable procedure (a procedure whose declaration does not include the BIND(C) language binding attribute), applies Fortran standard semantics by default. The standard specifies that for a non-interoperable procedure, VALUE causes a temporary, redefinable copy of the actual argument to be passed using the default passing mechanism. In earlier compiler versions, VALUE always caused the actual argument to be passed by value. Compiler version 14.0 introduced –assume std_value to specify the standard-conforming semantics and this was enabled if -standard-semantics was specified.

3.3.3 –assume ieee_fpe_flags enabled with -standard-semantics and –fp-model strict or –fp-model except

As of compiler version 15.0, if -standard-semantics and one of –fp-model strict or –fp-model except is specified, -assume ieee_fpe_flags is also enabled. This option causes the state of floating point exceptions to be saved on entry to a procedure and restored on exit. The save and restore operation has a significant performance penalty so it should be used only by applications that manipulate or query the floating point exception environment. Note that Intel Fortran requires that you specify –fp-model strict if you are using the Fortran standard intrinsic modules IEEE_ARITHMETIC, IEEE_EXCEPTIONS and/or IEEE_FEATURES.

3.3.4 New –[no-]opt-dynamic-align Compiler Option

When this option is set the compiler implements conditional optimizations based on dynamic alignment of the input data for maximum performance of vectorized code especially for long trip count loops. This, however, may result in different bitwise results for aligned and unaligned data with the same values. When unset the compiler will not perform these optimizations providing bitwise reproducibility.

3.3.5 Change to –fast option

-fp-model fast=2 has been added to the –fast option. This improves gcc compatibility and makes it easier to tune for performance.

3.3.6 New –init=snan Compiler Option

This is a new command line option to help find a class of uninitialized variables at run-time by initializing floating-point variables to signaling NaNs which can then be trapped if their values are fetched before being set.

3.3.7 New Optimization Report interface, report structure, and options in *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux**

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 *Intel® Parallel Studio XE Composer Edition* . 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 created in 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 options. 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.

3.3.8 New mode for PGO instrumentation -prof-gen=[no]threadsafe

This change adds a mode to the PGO instrumentation that allows for the collection of PGO data on applications that use a high level of parallelism, such as from OpenMP 3.1. This functionality improves PGO usage for the IA-32 and Intel® 64 architectures, as well as enables the support of PGO with the native Intel® MIC Architecture programming model.

3.4 Establishing the Compiler Environment

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 and gdb-mic`), Intel® Performance Libraries and, if present, Intel® C++ Compiler.

3.5 Known Issues

3.5.1 Certain uses of length type parameters in parameterized derived types are not yet fully implemented

The following uses of length type parameters in parameterized derived types (PDTs) are not yet fully implemented:

- PDT parameter constants with length type parameters
- %RE and %IM are not yet implemented

3.5.2 source ifortvars.csh failed on Ubuntu 13.04 CSH environment

On Ubuntu systems, the default shells are bash and tcsh. The compiler environment scripts (compilervars.sh, compilervars.csh, respectively) require an argument, either 'ia32' or 'intel64', and will work as expected. However, if you are using the BSD-based version of csh (which is

not installed by default), you will see the following error when the script is invoked with one the required arguments: ERROR: Unknown switch ''. Accepted values: ia32, intel64 The workaround is to use the one of the default, supported shells.

3.5.3 –warn interfaces may cause intermittent build fails with parallel builds

Using –warn interfaces may cause intermittent build failures when performing parallel builds. When using –warn interfaces, it is recommended to perform a serial build.

3.6 Coarrays

No special procedure is necessary to run a program that uses coarrays in a shared-memory configuration; you simply run the executable file. The underlying parallelization implementation is Intel® MPI. Installation of the compiler automatically installs the necessary Intel® MPI run-time libraries to run on shared memory. The Intel® Cluster Toolkit product (optional) installs the necessary Intel® MPI run-time libraries to run on distributed memory. Use of coarray applications with any other MPI implementation, or with OpenMP*, is not supported.

By default, the number of images created is equal to the number of execution units on the current system. You can override that by specifying the option `-coarray-num-images <n>` on the ifort command that compiles the main program. You can also specify the number of images in an environment variable `FOR_COARRAY_NUM_IMAGES`.

3.6.1 How to Debug a Coarray Application

The following instructions describe how to debug a Coarray application.

1. Add a stall loop to your application before the area of code you wish to debug, e.g.:

```
LOGICAL VOLATILE :: WAIT_FOR_DEBUGGER
LOGICAL, VOLATILE :: TICK
   :
DO WHILE(WAIT_FOR_DEBUGGER)
  TICK = .NOT. TICK
  END DO
! Code you want to debug is here
!
```

   The use of `VOLATILE` is required to ensure that the loop will not be removed by the compiler. If the problem is only found on one image, you can wrap the loop in `IF (THIS_IMAGE() .EQ. 4) THEN` or the like.
2. Compile and link with debug enabled (`-g`).
3. Create at least N+1 terminal windows on the machine where the application will be running, where N is the number of images your application will have.
4. In a terminal window, start the application.
   `linuxprompt> ./my_app`
5. In each of the other terminal windows, set your default directory to be the same as the location of the application executable. Use the ps command in one of the windows to find out which processes are running your application:

```
linuxprompt> ps -ef | grep 'whoami' | grep my_app
```

There will be several processes. The oldest is the one you started in step 4 – it has run the MPI launcher and is now waiting for the others to terminate. Do not debug it.

The others will look like this:

```
<your-user-name>  25653 25650 98 15:06 ?        00:00:49 my_app
<your-user-name>  25654 25651 97 15:06 ?        00:00:48 my_app
<your-user-name>  25655 25649 98 15:06 ?        00:00:49 my_app
```

The first number is the PID of the process (e.g., 25653 in the first line).

Call the PIDs of these N processes running "my_app" P1, P2, P3 and so on.

6. In each window other than the first, start your debugger and set it to stop processes when attached:

   ```
   linuxprompt> gdb-ia
   ```

7. Attach to one of the processes (e.g. to P1 in window 1, to P2 in window 2, etc.)

   ```
   (gdb) attach <P1>
   ```

8. Get execution out of the stall loop:

   (gdb) set WAIT_FOR_DEBUGGER = .false.

9. You can now debug.

3.6.2 Coarrays with Intel® Xeon Phi™ Coprocessors

As of Intel® Fortran Composer XE 2013 SP1, support has been added for using the Fortran 2008 coarray feature with Intel® Xeon Phi™ coprocessors implementing the Intel® Many Integrated Core (Intel® MIC) Architecture. You can choose among three execution models:

- A coarray application that has offload regions
- A coarray application that runs on both the coprocessor and the Intel® Xeon processor (heterogeneous)
- A coarray application that runs natively on the coprocessor

For all these modes, as with any Intel® MIC Architecture application, you must copy to the coprocessor all library shared objects referenced by the application. This will include Intel® MPI libraries as well as Intel® Fortran libraries such as libicaf.so.

For example:

```
sudo scp /opt/intel/composer_xe_2015.NN/compiler/lib/mic/libicaf.so
mic0:/lib64/libicaf.so

sudo scp /opt/intel/composer_xe_2015.NN/compiler/lib/mic/libintlc.so
mic0:/lib64/libintlc.so

sudo scp /opt/intel/composer_xe_2015.NN/mpirt/lib/mic/libmpi_mt.so
mic0:/lib64/libmpi_mt.so

sudo scp /opt/intel/composer_xe_2015.NN/mpirt/bin/mic/mpiexec.hydra
mic0:/bin/mpiexec.hydra

sudo scp /opt/intel/composer_xe_2015.NN/mpirt/bin/mic/pmi_proxy
mic0:/bin/pmi_proxy
```

This needs to be done every time the coprocessor is rebooted.

3.6.2.1 Using coarrays with offload regions

Use of offload regions in a coarray application has the following restrictions:

- All accesses to coarrays within an offload region must be to the local copy of the coarray – coindexing is not allowed
- No use of SYNC ALL, SYNC MEMORY, SYNC IMAGES, or LOCK/UNLOCK is allowed in an offload region
- Coarrays must not be allocated or deallocated within an offload region

Please see the documentation on using offload regions for more general information.

3.6.2.2 Heterogeneous coarray application

You can run a coarray application where some of the images run on the Intel® 64 host system and some run on an Intel® Xeon Phi™ coprocessor. This is called “heterogeneous”.

First, you will need to build the application using the –coarray=coprocessor option as well as specifying a coarray configuration file. For example:

```
ifort –coarray=coprocessor \
–coarray-config-file=MixedPlatform.conf   \
mycoarrayprog.f90 –o mycoarrayprog \
```

In this example, we have named the configuration file `MixedPlatform.conf`, but you can choose any name. This will create two executables, `mycoarrayprog` and `mycoarrayprogMIC`

The Intel® MIC Architecture native executable, `mycoarrayprogMIC`, must be copied to the coprocessor file system.

`MixedPlatform.conf` is an MPI configuration file, and is required to be able to run this heterogeneous configuration. An example configuration file is:

```
-n 4 -genv FOR_ICAF_STATUS=true -host myhostname mycoarrayprog : \
-n 4 -host mic0 /home/mydir/mycoarrayprogMIC
```

The `FOR_ICAF_STATUS=true` phrase is required; this is true whenever you have a configuration file. `myhostname` is the name of your host Intel® 64 architecture system. In this example, `/home/mydir` is the path to the Intel® MIC Architecture executable on the coprocessor – change this as required.

This configuration file runs 4 images on the host, 4 images on the card. You can change the `-n` value as desired.

Before running your executable on the host system, define the environment variable `I_MPI_MIC` to be `ENABLE` . Then run the executable – it will start both on the host and on the coprocessor.

3.6.2.3 Native coarray application on the coprocessor

To build a coarray application that runs natively on the coprocessor only, build it with the `-coarray` and `-mmic` options. A configuration file is not required. For example:

```
ifort -coarray -mmic mycoarrayprog.f90 -o \
mycoarrayprog -L/opt/intel/composer_xe_2015/mpirt/lib/mic
```

This will create “mycoarrayprog” as an Intel® MIC Architecture native executable.

A convenient way to run the application is to use the `micnativeloadex` utility, for example:

```
/opt/intel/mic/coi/tools/micnativeloadex/release/micnativeloadex \
mycoarrayprog
```

This will run the application on the coprocessor, and will bring over any referenced shared images.

The tool includes a help system, found with

```
/opt/intel/mic/coi/tools/micnativeloadex/release/micnativeloadex -h
```

3.6.3 Coarrays and Intel® MPI Library compatibility

Coarrays with Intel® Fortran Compiler 14 is incompatible with Intel® MPI Library 5.0. If using coarrays, ensure that either Intel® Fortran Compiler 15 or higher is used, or use a 4.x version of Intel® MPI Library.

3.7 Fortran 2003 and Fortran 2008 Feature Summary

The Intel Fortran Compiler supports all features from the Fortran 2003 standard. The Intel® Fortran Compiler also supports many features from the Fortran 2008 standard. Additional

features will be supported in future releases. Fortran 2008 features supported by the current version include:

- Maximum array rank has been raised to 31 dimensions (Fortran 2008 specifies 15)
- Coarrays
- CODIMENSION attribute
- SYNC ALL statement
- SYNC IMAGES statement
- SYNC MEMORY statement
- CRITICAL and END CRITICAL statements
- LOCK and UNLOCK statements
- ERROR STOP statement
- ALLOCATE and DEALLOCATE may specify coarrays
- Intrinsic procedures ATOMIC_DEFINE, ATOMIC_REF, IMAGE_INDEX, LCOBOUND, NUM_IMAGES, THIS_IMAGE, UCOBOUND
- CONTIGUOUS attribute
- MOLD keyword in ALLOCATE
- DO CONCURRENT
- NEWUNIT keyword in OPEN
- G0 and G0.d format edit descriptor
- Unlimited format item repeat count specifier
- A CONTAINS section may be empty
- Intrinsic procedures BESSEL_J0, BESSEL_J1, BESSEL_JN, BESSEL_YN, BGE, BGT, BLE, BLT, DSHIFTL, DSHIFTR, ERF, ERFC, ERFC_SCALED, GAMMA, HYPOT, IALL, IANY, IPARITY, IS_CONTIGUOUS, LEADZ, LOG_GAMMA, MASKL, MASKR, MERGE_BITS, NORM2, PARITY, POPCNT, POPPAR, SHIFTA, SHIFTL, SHIFTR, STORAGE_SIZE, TRAILZ,
- Additions to intrinsic module ISO_FORTRAN_ENV: ATOMIC_INT_KIND, ATOMIC_LOGICAL_KIND, CHARACTER_KINDS, INTEGER_KINDS, INT8, INT16, INT32, INT64, LOCK_TYPE, LOGICAL_KINDS, REAL_KINDS, REAL32, REAL64, REAL128, STAT_LOCKED, STAT_LOCKED_OTHER_IMAGE, STAT_UNLOCKED
- An OPTIONAL dummy argument that does not have the ALLOCATABLE or POINTER attribute, and which corresponds to an actual argument that: has the ALLOCATABLE attribute and is not allocated, or has the POINTER attribute and is disassociated, or is a reference to the NULL() intrinsic function, is considered not present
- A dummy argument that is a procedure pointer may be associated with an actual argument that is a valid target for the dummy pointer, or is a reference to the intrinsic function NULL. If the actual argument is not a pointer, the dummy argument shall have the INTENT(IN) attribute.
- BLOCK construct
- intrinsic subroutine EXECUTE_COMMAND_LINE

4 GNU* GDB Debugger

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

4.1 Features

GNU* GDB provided with *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux** 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 are several other new features:

- Support for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Support for Intel® Transactional Synchronization Extensions (Intel® TSX)
- Register support for Intel® Memory Protection Extensions (Intel® MPX) and Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
- Data Race Detection (*pdbx*):
  Detect and locate data races for applications threaded using POSIX* thread (pthread) or OpenMP* models
- Branch Trace Store (*btrace*):
  Record branches taken in the execution flow to backtrack easily after events like crashes, signals, exceptions, etc.
- Improved Fortran support
- Moved from GNU* GDB 7.7 to GDB 7.8 with Intel® Parallel Studio XE 2015 Update 2 Composer Edition.
- Intel® Processor Trace (Intel® PT) support for 5th generation Intel® Core™ Processors:
  (gdb) record btrace pt

4.2 Using GNU* GDB

GNU* GDB provided with *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux** comes in different versions:

- IA-32/Intel® 64 debugger:
  Debug applications natively on IA-32 or Intel® 64 architecture systems with `gdb-ia` on the command line. A standard Eclipse* IDE can be used for this as well, if a graphical user interface is desired.
- Intel® Xeon Phi™ coprocessor debugger:
  Debug applications remotely on Intel® Xeon Phi™ coprocessor systems. The debugger will run on a host system and a debug agent (`gdbserver`) on the coprocessor.
  There are two options:
  - Use the command line version of the debugger with `gdb-mic`. This only works for native Intel® Xeon Phi coprocessor applications. A standard Eclipse* IDE can be used for this as well if a graphical user interface is desired.

- Use an Eclipse* IDE plugin shipped with *Intel® Parallel Studio XE 2015 Composer Edition for Fortran Linux**. This works only for offload enabled Intel® Xeon Phi coprocessor applications..

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|ja_JP]/debugger/gdb/gdb.pdf
<install-dir>/Documentation/[en_US|ja_JP]/debugger/
              gdb/gdb_quickstart_lin.pdf
```

4.4 Known Issues and Changes

4.4.1 Safely ending offload debug sessions

To avoid issues like orphan processes or stale debugger windows when ending offload applications, manually end the debugging session before the application is reaching its exit code. The following procedure is recommended for terminating a debug session.

- Manually stop a debug session before the application reaches the exit-code.
- When stopped, press the red stop button in the toolbar in the Intel® MIC Architecture-side debugger first. This will end the offloaded part of the application.
- Next, do the same in the CPU-side debugger.
- The link between the two debuggers will be kept alive. The Intel® MIC Architecture-side debugger will stay connected to the debug agent and the application will remain loaded in the CPU-side debugger, including all breakpoints that have been set.
- At this point, both debugger windows can safely be closed.

4.4.2 Intel® MIC Architecture-side debugger asserts on setting source directories

Setting source directories in the GNU* GDB might lead to an assertion.

Resolution:
The assertion should not affect debugger operation. To avoid the assertion anyway, don't use source directory settings. The debugger will prompt you to browse for files it cannot locate automatically.

4.4.3 Using GDB within Eclipse IDE

Prior to using the Intel provided GNU* GDB within an Eclipse* IDE compilervars.sh should be sourced (see Establishing the Compiler Environment).

4.4.4 Debugging Fortran applications with Eclipse* IDE plugin for Intel® Xeon Phi™ coprocessor

If the Eclipse* IDE plugin for the Intel® Xeon Phi™ coprocessor is used for debugging Fortran applications, evaluation of arrays in the locals window might be incorrect. The underlying CDT

applies the C/C++ syntax with brackets to arrays to retrieve their contents. This does not work for Fortran.
Solution: Use a fully qualified Fortran expression to retrieve the contents of arrays (e.g. with array sections like *array(1:10)*).

4.4.5 Nested variable length arrays not working with user derived types

If using a variable length array (VLA) of user derived types that contain nested variable length arrays, the debugger fails to resolve the latter. A future version of the debugger will fix this issue.

5 Intel® Math Kernel Library (Intel® MKL)

This section summarizes changes, new features and late-breaking news about this version of the Intel® Math Kernel Library (Intel® MKL). Bug fixes can be found here.

5.1 What's New in Intel MKL 11.2 Update 4

- Reduced memory consumption when using Intel MKL in GNU OMP parallel regions
- BLAS:
    - Improved parallel and serial performance of ?TRSM on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for 64-bit Intel MKL
    - Improved ?SYRK/?HERK/?SYR2K/?HER2K performance for beta=0 on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for 64-bit Intel MKL
    - Improved serial performance of STRMM for small triangular matrices (dimension less than or equal to 10) on Intel® AVX2 for 64-bit Intel MKL
    - Improved performance of BLAS level 3 functions for second generation of Intel® Xeon Phi™ coprocessors

5.2 What's New in Intel MKL 11.2 Update 3

- Extended the Intel MKL memory manager to improve scaling on large SMP systems.
- Added new service functions to provide more control for Intel MKL Automatic Offload for Intel Xeon Phi systems. These functions include mkl_mic_get_meminfo, mkl_mic_get_cpuinfo, mkl_mic_set_flags, mkl_mic_get_flags, mkl_mic_clear_status, and mkl_mic_get_status.
- BLAS:
    - Improved parallel performance of (D/S)SYMV on all Intel Xeon processors.

- Improved (C/D/S/Z/DZ/SC)ROT performance for Intel Advanced Vector Extensions (Intel AVX) architectures in 64-bit Intel MKL.
- Improved (C/Z)ROT performance for Intel Advanced Vector Extensions 2 (Intel AVX2) architectures in 64-bit Intel MKL.
- Improved parallel performance of ?SYRK/?HERK, ?SYR2K/?HER2K, and ?GEMM for cases with large *k* sizes on Intel AVX2 architectures in 64-bit Intel MKL.
- Improved ?SYRK/?HERK and ?SYR2K/?HER2K performance on Intel Xeon Phi coprocessors.

- LAPACK:
  - Improved performance of SVD, for cases where singular vectors are computed, on multi-socket systems based on Intel AVX or Intel AVX2 architectures.
  - Added new routines for incomplete LU factorization without pivoting.

5.3 What's New in Intel MKL 11.2 Update 2

- BLAS:
  - Improved ?GEMM performance for Intel® Xeon Phi™ coprocessors for cases where k >> m, k >> n.
  - 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.
  - 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%.

- Parallel Direct Sparse Solver for Clusters:
    - Added ability to overwrite the right hand side vector with solution with the distributed CSR format.
    - Added ability to gather system solution on all compute nodes with distributed CSR format.
- Intel® MKL PARDISO:
    - Significantly improved overall scalability for Intel Xeon Phi coprocessors.
    - 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.

5.4 What's New in Intel MKL 11.2 Update 1

- Introduced support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) on Intel® Xeon® processors for Windows* and Linux* versions of Intel MKL. This is in addition to the current support for Intel® AVX-512 instructions for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- BLAS:
    - Optimized the following functions on Intel microarchitecture code name Skylake:
        - (D/Z)AXPY,(S/D/C/Z)COPY, DTRMM (for cases when the triangular matrix is on right side and with no matrix transpose)
    - Optimized following BLAS Level-1 functions on Intel AVX2 both for Intel® 64 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 Intel 64 architectures)
    - Improved DGEMM performance (serial and multithreaded) on Intel AVX (for Intel 64 architectures)
- LAPACK:
    - Introduced support for LAPACK version 3.5. New features introduced in this version are:
        - Symmetric/Hermitian LDLT factorization routines with rook pivoting algorithm
        - 2-by-1 CSD for tall and skinny matrix with orthonormal columns
    - Improved performance of (C/Z)GE(SVD/SDD) when M>=N and singular vectors are not needed
- FFT:
    - Introduced Automatic Offload mode for 1D Batch FFT on Intel MIC Architecture
    - Improved performance of Hybrid (OpenMP+MPI) Cluster FFT
    - Improved accuracy of large 1D real-to-complex transforms
- Parallel Direct Sparse Solver for Clusters:
    - Added support for many factorization steps with the same reordering (maxfct > 1)
- Intel MKL PARDISO:
    - Added support for Schur complement, including getting explicit Schur complement matrix and solving the system through Schur complement
- Sparse BLAS:
    - Optimized SpMV on Intel microarchitecture code name Skylake
    - Added Sparse Matrix Checker functionality as standalone API to simplify validation of matrix structure and indices(see Sparse Matrix Checker Routines in 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)

5.5 What's New in Intel MKL 11.2

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

    - Added support for pure MPI mode and support for single OpenMP thread in hybrid configurations
- BLAS:
    - Improved threaded performance of ?GEMM for all 64-bit architectures supporting Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Optimized ?GEMM, ?TRSM, DTRMM for the Intel AVX-512 instruction set
    - Improved performance of ?GEMM for outer product [large m, large n, small k] and tall skinny matrices [large m, medium n, small k] on Intel MIC Architecture
    - Improved performance of ?TRSM and ?SYMM in Automatic Offload mode on Intel MIC Architecture
    - Improved performance of Level 3 BLAS functions for 64-bit processors supporting Intel AVX2
    - Improved ?GEMM performance on small matrices for all processors when MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ is defined during compilation (see the *Intel® Math Kernel Library User's Guide for more details*)
    - Improved performance of DGER and DGEMM for the beta=1, k=1 case for 64-bit processors supporting Intel SSE4.2, Intel® Advanced Vector Extensions (Intel® AVX), and Intel AVX2 instruction sets
    - Optimized (D/Z)AXPY for the Intel AVX-512 instruction set
    - Optimized ?COPY for Intel AVX2 and 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
    - Improved performance of ?POTRF UPLO=U in Automatic Offload mode on Intel MIC Architecture
    - Added Automatic Offload for ?SYRDB on Intel MIC Architecture, which speeds up ?SY(EV/EVD/EVR) when eigenvectors are not needed
- PBLAS and ScaLAPACK:
    - Enabled Automatic Offload in P?GEMM routines for large distribution blocking factors
- Sparse BLAS:
    - Optimized SpMV kernels for Intel AVX-512 instruction set
    - Added release example for diagonal format use in Sparse BLAS
    - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel SSE4.2, Intel AVX and Intel AVX2 instruction sets
- Intel MKL PARDISO:
    - Added the ability to store Intel MKL PARDISO handle to the disk for future use at any solver stage
    - Added pivot control support for unsymmetric matrices and out-of-core mode
    - Added diagonal extraction support for unsymmetric matrices and out-of-core mode
    - Added example demonstrating use of Intel MKL PARDISO as iterative solver for non-linear systems
    - Added capability to free memory taken by original matrix after factorization stage if iterative refinement is disabled
    - Improved memory estimation of out-of-core (OOC) portion size for reordering algorithm leading to improved factorization-solve performance in OOC mode
    - Improved message output from Intel MKL PARDISO
    - Added support of zero pivot during factorization for structurally symmetric cases
- Poisson library:

    - Added example demonstrating use of the Intel MKL Poisson library as a preconditioner for linear systems solves
- Extended Eigensolver:
    - Improved message output
    - Improved examples
    - Added input and output iparm parameters in predefined interfaces for solving sparse problems
- FFT:
    - Optimized FFTs for the Intel AVX-512 instruction set
    - Improved performance for non-power-of-2 length on Intel® MIC Architecture
- VML: Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
    - Added support of ntrial=0 in Binomial Random Number Generator
    - Improved performance of MRG32K3A and MT2203 BRNGs on Intel MIC Architecture
    - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
    - Added support for group/pooled mean estimates (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN)
- Data Fitting: Fixed incorrect behavior of the natural cubic spline construction function when number of breakpoints is 2 or 3
- Introduced an Intel MKL mode that ignores all settings specified by Intel MKL environment variables
    - User can set up the mode by calling mkl_set_env_mode() routine which directs Intel MKL to ignore all environment settings specific to Intel MKL so that all Intel MKL related environment variables such as MKL_NUM_THREADS, MKL_DYNAMIC, MKL_MIC_ENABLE and others are ignored; users can instead set needed parameters via Intel MKL service routines such as mkl_set_num_threads() and mkl_mic_enable()

5.6 Notes

- 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 MIC Architecture support are not installed unless explicitly selected during installation
- Unaligned CNR is not available for MKL Cluster components (ScaLAPACK and Cluster DFT)
- Examples for using Intel MKL with BOOST/uBLAS and Java have been removed from the product distribution and placed in the following articles:

    - How to use Intel® MKL with Java*
    - How to use BOOST* uBLAS with Intel® MKL
- 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)
- Important deprecations are listed in Intel® Math Kernel Library (Intel® MKL) 11.2 Deprecations

5.7 Known Issues

A full list of the known limitations can be found in the Intel® MKL Article List at Intel® Developer Zone

5.8 Attributions

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

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

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

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

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

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. 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.

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

6 Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

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

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

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

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

http://www.intel.com/design/literature.htm

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

http://www.intel.com/products/processor_number/

for details.

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

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

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