Intel® Visual Fortran Composer XE 2013 for Windows* Installation Guide and Release Notes

Document number: 321417-004US
19 June 2013

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.

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 Product Updates

Update 5 – June 2013

- Intel® Visual Fortran Compiler updated to 13.1.3
- Intel® Math Kernel Library updated to 11.0 Update 5
- Corrections to reported problems
  - Compiler fix list
  - Intel® MKL fix list

Update 4 – May 2013

- Intel® Visual Fortran Compiler updated to 13.1.2
  - Including the fix for 0_10711 Internal Compiler Error with Composer XE 2013 Update 3
- Intel® Math Kernel Library updated to 11.0 Update 4
- Corrections to reported problems
  - Compiler fix list
  - Intel® MKL fix list

Update 3 – March 2013

- Intel® Visual Fortran Compiler updated to 13.1.1
- Intel® Math Kernel Library updated to 11.0 Update 3
- Corrections to reported problems
  - Compiler fix list
  - Intel® MKL fix list

Update 2 – January 2013

- Intel® Visual Fortran Compiler updated to 13.1.0
  - Added support for additional directives, clauses and procedures from OpenMP* 4.0
  - Added the KMP_PLACE_THREADS run-time environment variable
  - Added (in Update 1) a temporary option that can improve performance of coarray applications
  - Added /Qvec-report7 compiler option
  - Added description of change in how sequential, unformatted files are buffered during READs
  - ProcessorInfo sample added in Win32.zip to illustrate use of GetLogicalProcessorInformation
  - DynamicLoad sample in Dll.zip updated to use more standard Fortran features
- Intel® Math Kernel Library updated to 11.0 Update 2
- Corrections to reported problems
  - Compiler fix list: http://intel.ly/S5uIAb
  - Intel® MKL fix list: http://intel.ly/S5uw3R

Update 1 – October 2011

- Intel® Visual Fortran Compiler updated to 13.0.1
  - ATTRIBUTES ALIGN may now be specified for an ALLOCATABLE or POINTER component of a derived type
  - Declarations to support use of the Windows API routines GetLogicalProcessorInformation and GetLogicalProcessorInformationEx have been added to modules IFWINTY and KERNEL32
- Intel® Math Kernel Library updated to 11.0 Update 1
- Note added about problems viewing documentation in Visual Studio 2012* on Windows Server 2012*
- Corrections to reported problems
  - Compiler fix list: http://intel.ly/S5uIAb
  - Intel® MKL fix list: http://intel.ly/S5uw3R

1.2 Changes since Intel® Visual Fortran Composer XE 2011

- Intel® Visual Fortran Compiler updated to version 13.0
  - The static form of the Intel® OpenMP* library, libiomp5mt.lib, is no longer provided.
  - On Windows systems with a long value for the PATH system environment variable, a reboot after install may be required.
- Intel® Math Kernel Library updated to version 11.0
  - Intel® Math Kernel Library (Intel® MKL) removed support for Intel® Pentium® III processors. The minimum instruction set supported by Intel® MKL is Intel® SSE2. See the Intel® MKL section for additional changes.

    - The path to Intel® MKL run-time DLLs is no longer added to the system PATH environment variable. If you will be running programs using the MKL DLL libraries, you will need to add the appropriate folder to the system PATH environment variable.
- The Intel® Parallel Debugger Extension is no longer provided
- Support for Microsoft Windows 8*, Microsoft Windows Server 2012* and Microsoft Visual Studio 2012* has been added
    - This release was finalized before the Microsoft products were released. We do not expect that the final releases of the Microsoft products will change in ways that affect Intel® Visual Fortran Composer XE, but if they do we will address them in the earliest possible update.
- Support for the following versions of Windows has been dropped:
    - Windows Vista*
    - Windows Server 2003*
- Support for Microsoft Visual Studio 2005* has been dropped
- Support for Windows XP* has been deprecated
- The Tools > Options > Intel Visual Fortran dialog in Visual Studio is now Tools > Options > Intel Composer XE > Visual Fortran
- The Intel® Software Manager has been added to help you manage product updates and license activation
- Corrections to reported problems

1.3 Product Contents

*Intel® Visual Fortran Composer XE 2013 for Windows** includes the following components:

- Intel® Visual Fortran Compiler XE 13.1.3 for building applications that run on IA-32 and Intel® 64 architecture systems
- Intel® Math Kernel Library 11.0 Update 5
- Integration into Microsoft* development environments
- Microsoft Visual Studio 2010* Shell and Libraries (not included with Evaluation licenses)
- Sample programs
- On-disk documentation

1.4 System Requirements

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

- 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)
- 2GB free disk space required for all product features and all architectures

- Microsoft Windows XP* SP3, Microsoft Windows 7*, Microsoft Windows 8*, Microsoft Windows Server 2012*, Microsoft Windows Server 2008* or Microsoft Windows HPC Server 2008* (embedded editions not supported)
    - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2012* or Visual Studio 2010* or Visual Studio 2010* Shell or Visual Studio 2008* SP1 or Visual Studio 2008* Shell with Visual Studio 2008 SP1 update applied.
    - On Microsoft Windows 8, the product installs into the "Desktop" environment. Development of "New Windows 8* UI" applications is not supported.
    - Support for Microsoft Windows XP* is deprecated – a future major release of Intel® Visual Fortran Composer XE will not support Windows XP
- 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 2012* with C++ component installed
    - Microsoft Visual Studio 2010* with C++ and "X64 Compiler and Tools" components installed [1]
    - Microsoft Visual Studio 2008* Standard Edition or higher with C++ and "X64 Compiler and Tools" components installed [1]
    - Intel® Visual Fortran development environment based on Microsoft Visual Studio 2010 Shell (included with some license types of Intel® Fortran Compiler) [2]
    - Intel® Visual Fortran development environment based on Microsoft Visual Studio 2008 Shell (included with compiler versions 11.0, 11.1 and Intel® Visual Fortran Composer XE 2011 through Update 5.)
- To use command-line tools only to build IA-32 architecture applications, one of:
    - Microsoft Visual C++ 2010* Express Edition [2]
    - Microsoft Visual C++ 2008* Express Edition
- To use command-line tools only to build Intel® 64 architecture applications:
    - Microsoft Windows Software Development Kit for Windows 7 and .NET Framework 4.0*
- Installation of the included Microsoft Visual Studio 2010 Shell has the following limitations:
    - Windows XP 64-bit is not supported. Microsoft Visual Studio 2008 Shell from earlier versions of Intel® Visual Fortran can be used on Windows XP 64-bit.
    - Installation on Windows XP requires prior installation of Microsoft .NET 4.0* Framework. See the Installation section of the Intel® Visual Fortran Composer XE 2013 Release Notes for details.
- 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 installs this component by default in all editions.

2. Intel® Visual Fortran development environment based on Microsoft Visual Studio 2010* Shell is included with Academic and Commercial licenses for Intel® Visual Fortran Composer XE. It is not included with Evaluation licenses. This development environment provides everything necessary to edit, build and debug Fortran applications. Some features of the full Visual Studio product are not included, such as:
    - Resource Editor (see ResEdit* (http://www.resedit.net/), a third-party tool, for a substitute)
    - Automated conversion of Compaq* Visual Fortran projects
    - Microsoft language tools such as Visual C++* or Visual Basic*
3. Microsoft Visual C++ 2010 Express Edition will coexist with the Intel® Visual Fortran Composer XE 2013 installation of Microsoft Visual Studio 2010 Shell. Note that the C++ and Fortran development environments will be separate.
4. The default for Intel® Visual Fortran is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions. A compiler option is available to generate code that will run on any IA-32 architecture processor. Note, however, that applications calling Intel® MKL require a processor supporting the Intel® SSE2 instructions.
5. 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 Windows API routine not present in older versions of Windows. You are responsible for testing application compatibility. You may need to copy certain run-time DLLs onto the target system to run your application.

1.5 Documentation

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

1.5.1 Documentation on Creating Windows-based Applications Now on the Web

The chapter in the compiler documentation on “Using Intel® Visual Fortran to Create and Build Windows-based Applications” has been moved to the “Software Documentation from Intel” web site. You can download this documentation from http://intel.ly/WinApp (PDF).

1.5.2 Delay on First Access of Intel® Composer XE Help in Visual Studio 2008*

The first time you access the Intel-installed help documentation in Microsoft Visual Studio 2008, a substantial delay may occur. Prior to displaying the help, Visual Studio must merge the new help into the collection and re-index the collection. Depending on whether you have Visual Studio help content already present and the size of the installed help, the delay in viewing help the first time may be several minutes or more.

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 Samples

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

1.8 Japanese Language Support

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

Japanese language support is not provided in all product updates.

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/pla2A5

1.9 Technical Support

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

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

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

2 Installation

2.1 Pre-Installation Steps

2.1.1 Install Prerequisite Software

If you will be installing the included Microsoft Visual Studio 2010 Shell, additional Microsoft software may be required to be installed before beginning the installation of Intel® Visual Fortran Composer XE 2013. Note that installation of Microsoft Visual Studio 2010 Shell is not supported on Windows XP 64-bit.

Microsoft .NET 4.0 Framework* is required. If you do not already have this installed, you can download the installer:

- .NET 4.0 Framework 32-bit and 64-bit

If you are installing on Windows 8*, Windows 7* or Windows Server 2008, the installation of the Shell will attempt to download and install .NET Framework 4.0 automatically if it is not already present. If this fails, the Intel® Visual Fortran Composer install will fail with a message that may not indicate the exact problem. If you find that the installation of the Shell fails, please download .NET 4.0 Framework from the above link and try again.

If you are installing Intel® Visual Fortran Composer XE 2013 from DVD or from the full product downloadable that includes Visual Studio 2010 Shell, it will try to install Visual Studio 2010 Shell if you do not already have Visual Studio 2010 installed. If you do not want Visual Studio 2010 Shell to install, you can choose a Custom install and deselect it, or choose the "_novsshell.exe" downloadable installer.

2.1.2 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 2008 Shell, 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.

This step is not required when using Microsoft Visual Studio 2010 or Visual Studio 2012.

2.2 Installation

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

If you 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. If you want to remove older versions, you may do so before or after installing the newer one.

Register your serial number at the Intel® Software Development Products Registration Center for access to product updates and previous versions.

2.2.1 Reboot After Install Recommended

Installation of Intel® Visual Fortran Composer XE adds to the system PATH environment variable the folders containing the compiler run-time DLLs (but not those of Intel® Math Kernel Library). On some systems, if the length of the PATH value is between 2048 and 4096 characters, command line operations may fail until the system is rebooted. Intel recommends a reboot after the first install of Intel® Visual Fortran Composer XE.

2.2.2 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.

2.2.3 Using a License Server

If you have purchased a “floating” license, see http://intel.ly/oPEdEe for information on how to install using a license file or license server. 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.

2.2.4 Additional Steps to Install Documentation for Microsoft Visual Studio 2010

When installing Intel® Visual Fortran Composer XE 2013 on a system with Microsoft Visual Studio 2010 for the first time, you will be asked to initialize the “Local Store” for documentation for Visual Studio 2010 if it was not done before. The "Help Library Manager" will register the Intel® Visual Fortran Composer XE 2013 help documentation within Visual Studio 2010. Please follow the instructions of the "Help Library Manager" installation wizard to install the Intel® Visual Fortran Composer XE 2013 help documentation for Visual Studio 2010.

This step is only needed once. When you install Intel® Visual Fortran Composer XE 2013 updates in the future, you will not be required to re-register the documentation through the "Help Library Manager".

For the more information, see http://msdn.microsoft.com/en-us/library/dd264831.aspx or search on microsoft.com for "Help Library Manager".

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 Development 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 Changing, Updating and Removing the Product

Use the Windows Control Panel "Add or Remove Products" or "Programs and Features" applet to change which product components are installed or to remove the product. Depending on which product you installed, the entry will be one of the following:

- Intel(R) Visual Fortran Composer XE 2013 for Windows*
- Intel(R) Composer XE 2013 for Windows*
- Intel(R) Parallel Studio XE 2013 for Windows*

If you also installed Microsoft Visual Studio 2010 Shell as part of the compiler install, the following additional entries may be present:

- Microsoft Visual Studio 2010 Shell (Integrated) - ENU
- Microsoft Visual Studio 2010 Files for Intel Visual Fortran
- Microsoft Visual Studio 2010 Remote Debugger – ENU

The entries for Visual Studio Shell, Files and Remote Debugger should not be removed unless you want to completely remove the product.

When installing an updated version of the product, you do not need to remove the older version first. The first time you install an update, you will have the choice to replace the older version or to keep both the older and newer versions on the system. This choice is remembered for future updates. In Microsoft Visual Studio you can select which specific compiler version to use through the Tools > Options > Intel Composer XE > Visual Fortran Compiler dialog. Compiler versions older than 12.0 (Intel® Visual Fortran Composer XE 2011) are not available to be selected through Visual Studio. All installed versions can be used from the command line.

If you remove a newer version of the product you may have to reinstall the integrations into Microsoft Visual Studio from the older version.

2.5 Silent Install and Uninstall

For information on how to install and uninstall the compiler in an automated fashion, please see http://intel.ly/qAwdvR

2.6 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation. The system environment variable `IFORT_COMPILER13` can be used to locate the most recently installed Intel® Visual Fortran Composer XE 2013.

- `C:\Program Files\Intel\Composer XE 2013`
  - `bin`
    - `ia32`
    - `ia32_intel64`
    - `intel64`
  - `compiler`
    - `include`
      - `ia32`
      - `intel64`
    - `lib`
      - `ia32`
      - `intel64`
  - `Documentation`
  - `help`
  - `mkl`
    - `benchmarks`
    - `bin`
    - `examples`
    - `include`
    - `interfaces`
    - `lib`
    - `tests`
    - `tools`
  - `redist`
  - `Samples`

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® Visual Fortran Compiler

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

3.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel Fortran Compiler (8.0 and later) may be used in a build with version 13.0. 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 (`/Qipo`) option must be recompiled.
- 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 declaration 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 (12.0)

In previous releases, 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, compiler versions 12.0 and later align such items at 16 bytes and expect received arguments to be aligned on 16-byte boundaries.

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 13 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.

3.1.2 Static Form of the Intel® OpenMP* Library is No Longer Provided

The static form of the Intel® OpenMP* library, libiomp5mt.lib, is no longer provided, and the `/Qopenmp-link:static` command line option is no longer supported. Please replace all references to libiomp5mt.lib with libiomp5md.lib, the DLL import library. This change also implies that applications using OpenMP will need to have the Intel® compiler redistributables installed if deployed on a system where an Intel® compiler is not also present. See http://intel.ly/PwYFvl for more information.

3.2 New and Changed Compiler Features

Some language features may not yet be described in the compiler documentation. Please refer to the Fortran 2003 Standard (http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf) and Fortran 2008 Standard (http://j3-fortran.org/doc/standing/links/007.pdf) if necessary.

3.2.1 Features from Fortran 2003

- Default initialization of polymorphic variables
- The keyword MODULE may be omitted from MODULE PROCEDURE in a generic interface block when referring to an external procedure

3.2.2 Features from Fortran 2008

- ATOMIC_DEFINE and ATOMIC_REF intrinsics

3.2.3 Coarrays

No special procedure is necessary to run a program that uses coarrays in a shared-memory environment; 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.

A license for Intel® Cluster Studio must be present in order to use the `/coarray:distributed` option. For details on how to run a distributed coarray application on Windows, please see http://intel.ly/oZurZS

Use of coarray applications with any MPI implementation other than Intel® MPI, or with OpenMP*, is not supported at this time.

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 `/Qcoarray-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`.

3.2.3.1 Compiler Option to Improve Coarray Performance (13.0.1)

Work is ongoing to improve performance of coarray applications. Some of this work is present in the 13.0.1 (Composer XE 2013 Update 1) compiler but is disabled by default. You can enable the optimizations implemented so far by adding the compile option:

```
/switch:coarray_opts
```

when you compile your application. At present, this improves the performance of copying coarray values from another image. Not all applications may see a significant improvement when using this option. We encourage you to try this option and to let us know if it introduces errors into your application.

In a future major version, these optimizations will be enabled by default and the option shown above will be removed.

3.2.3.2 Coarray Known Issues

The following features are known not to work in this version:

- Accessing another image's value of an ALLOCATABLE or POINTER component of a derived-type coarray. Some forms of this do work, some do not.

3.2.4 New and Changed Directives

The following compiler directives are new or changed in Intel® Composer XE 2013 – please see the documentation for details:

- ATTRIBUTES CVF
- ORDERED/END ORDERED
- SIMD VECTORLENGTHFOR

3.2.5 OpenMP 4.0 Changes (13.1.0)

The following directives and procedure, new in OpenMP* 4.0, are supported by the 13.1.0 (Composer XE 2013 Update 2) compiler. Until the product documentation is updated, please refer to http://intel.ly/VPV24X

SIMD Directives:

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

Procedure:

- OMP_GET_PROC_BIND

3.2.5.1 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.

Syntax

```
value = ( int [ "C" | "T" ] [ delim ] | delim ) [ int [ "T" ]
[ delim ] ] [ int [ "O" ] ];
```

```
delim = "," | "x";
```

Effect

Specifies the number of cores, with optional offset value and number of threads per core to use. The "C" indicates cores, "T" indicates threads, "O" is used to specify an offset. Either cores or threads should be specified. If omitted, the default value is the available number of cores (threads).

Examples

`5C,3T,1O` - use 5 cores with offset 1, 3 threads per core
`5,3,1` - same as above
`24` - use first 24 cores, all available threads per core
`2T` - use all cores, 2 threads per core
`,2` - same as above
`3x2` - use 3 cores, 2 threads per core
`4C12O` - use 4 cores with offset 12, all available threads per core

3.2.6 ATTRIBUTES ALIGN for component of derived type (13.0.1)

As of compiler version 13.0.1, the ATTRIBUTES ALIGN directive may be specified for an ALLOCATABLE or POINTER component of a derived type. The directive must be placed within the derived type declaration, and if it is an extended type, the directive must not name a component in a parent type.

If this is specified, the compiler will apply the indicated alignment when the component is allocated, either through an explicit ALLOCATE or, for ALLOCATABLE components, through implicit allocation according to Fortran language rules.

A module containing an ATTRIBUTES ALIGN directive for a derived type component cannot be used with a compiler earlier than version 13.0.1.

3.2.7 Change in File Buffering Behavior (13.1)

In product versions prior to Intel® Visual 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® Visual 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® Visual 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 Other Changes

- The output of the G format edit descriptor has been changed to more properly conform to the Fortran 2008 standard. The changes involve effects of rounding on representation of values that round to -0
- When on output using a D, E, G, EN or ES format the exponent field overflows the implicit exponent width, the output field is filled with asterisks. In earlier versions, the exponent would be displayed in a manner inconsistent with the standard
- The compiler can now vectorize references to the RANDOM_NUMBER and RANF intrinsic subroutines.
- (13.0.1) Modules IFWINTY and KERNEL32 have added declarations to support the Windows API routines GetLogicalProcessorInformation and GetLogicalProcessorInformationEx.

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details.

3.3.1 New and Changed in Composer XE 2013

- /align:array8byte
- /align:array16byte
- /align:array32byte
- /align:array64byte
- /align:array128byte
- /align:array256byte
- /assume:[no]std_intent_in

- /Qdiag-enable:sc-enums
- /Qdiag-enable:sc-{full|concise|precise}
- /Qdiag-enable:sc-single-file
- /Qguide-profile:<file|dir>[,[file|dir],…]
- /Qimf-domain-exclusion:classlist[:funclist]
- /Qvec-report6
- /Qvec-report7 (13.1.0)

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

3.3.2 New /Qvec-report7 Compiler Option (13.1.0)

The /Qvec-report7 option provides additional details about vectorization of loops, including statistics and metrics about loads, stores, type conversions and more. This information can be useful in understanding how much improvement might be expected from vectorization of a given loop.

The information displayed from this option can also be processed by a new vectorization analysis tool available from http://intel.ly/XeSkW6

3.4 Parallel Build Option

An enhancement to the Visual Studio build environment has been added which allows for parallel builds of sources without unresolved dependencies on multicore or multiprocessor systems. This can reduce the total time needed to build larger projects.

To enable this, open the project property page Fortran > General and set the property “Multi-processor compilation” to Yes.

3.5 Known Issues

3.5.1 Coarray Issues

For a list of known issues with Fortran 2008 Coarray support, see Coarray Known Issues.

3.5.2 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.

3.6 Microsoft Visual Studio 2010 and 2012 Notes

Microsoft Visual Studio 2010 brings several changes that primarily affect building of mixed-language applications where the main program is in C or C++. These changes were carried forward into Visual Studio 2012.

3.6.1 Configuring Microsoft Visual C++ to Reference Intel® Fortran Run-Time Libraries

In previous releases, one used the Tools > Options > Projects and Solutions > VC++ Directories dialog to make the Intel Fortran LIB folder available to C/C++ projects. In Visual Studio 2010, the method of doing this is very different.

1. In Visual Studio, with a solution open that contains a C++ project, select View > Property Manager. If you do not see Property Manager under the View menu, you will find it under View > Additional Windows. The Property Manager window will appear. Note that this is not Properties Window or Properties Pages.
2. Click on the triangles or + signs to expand the property tree under the Debug|Win32 configuration
3. Double click on Microsoft.Cpp.Win32.user
4. Select VC++ Directories
5. Click in the field to the right of "Library Directories"
6. Click the triangle that appears to the right and select <Edit...>
7. Click the New Line button or press Ctrl-Insert
8. In the new field that appears, type:

   ```
   $(IFORT_COMPILER13)\compiler\lib\ia32
   ```
9. Click OK, OK
10. In the Visual Studio toolbar, select File > Save All

If you will be building Intel® 64 (x64) configurations:

1. Back in the Property Manager, expand the Debug|x64 configuration
2. Double click on Microsoft.Cpp.x64.user
3. Select VC++ Directories
4. Click in the field to the right of "Library Directories"
5. Click the triangle that appears to the right and select <Edit...>
6. Click the New Line button or press Ctrl-Insert
7. In the new field that appears, type:

   ```
   $(IFORT_COMPILER13)\compiler\lib\intel64
   ```
8. Click OK, OK
9. In the Visual Studio toolbar, select File > Save All

Click on the Solution Explorer tab, or press Ctrl-Alt-L, to make it visible again.

If you do not see the Microsoft.Cpp.x64.user property page listed for the x64 configuration, right click on Debug|x64 and select Add Existing property Sheet. Browse to the location which contains the MsBuild 4.0 property pages. On Windows XP, this is typically:

```
C:\Documents and Settings\<username>\Local Settings\Application Data\Microsoft\MSBuild\v4.0
```

On Windows 7 and Windows 8, it is typically:

```
C:\Users\<username>\AppData\Local\Microsoft\MSBuild\v4.0
```

You may need to enable viewing of hidden files and folders to see these paths.

Select Microsoft.Cpp.x64.user.props and click Open. Now follow the steps above.

3.6.2 Adjusting Project Dependencies

If you are converting a project from an earlier version of Visual Studio and had established Project Dependencies, these are converted to References by Visual Studio 2010/2012. A Fortran project that is referenced by a C/C++ project will prevent the C/C++ project from building, with an MSB4075 error. To solve this:

1. Right click on the C/C++ project and select References.
2. If any Fortran project is shown as a reference, click Remove Reference. Repeat this for all Fortran projects shown as a reference. Click OK.
3. Repeat the above steps for any other C/C++ project

Now you have to reestablish project dependencies.

1. Right click on the C/C++ project and select Project Dependencies.
2. Check the box for each project that is a dependent of this project.
3. Click OK.
4. Repeat the above steps for any other C/C++ project that has dependencies.

Unlike earlier versions of Visual Studio, Visual Studio 2010/2012 does not automatically link in the output library of dependent projects, so you will need to add those libraries explicitly to the parent project under Linker > Additional Dependencies. You can use the Visual Studio macros $(ConfigurationName) and $(PlatformName) as required to qualify the path. For example:

```
..\FLIB\$(ConfigurationName)\FLIB.lib
```

Where $(ConfigurationName) will expand to Release or Debug, as appropriate. Similarly, $(PlatformName) will expand to Win32 or x64 as appropriate.

3.6.3 Showing Documentation Issue with Visual Studio 2012 and Windows Server 2012

If on Windows Server 2012* you find that you cannot display help or documentation from within Visual Studio 2012, 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”.

3.7 Fortran 2003 and Fortran 2008 Feature Summary

The Intel Fortran Compiler supports many features that are new in Fortran 2003. Additional Fortran 2003 features will appear in future versions. Fortran 2003 features supported by the current compiler include:

- The Fortran character set has been extended to contain the 8-bit ASCII characters ~ \ [ ] ` ^ { } | # @
- Names of length up to 63 characters

- Statements of up to 256 lines
- Square brackets [ ] are permitted to delimit array constructors instead of (/ /)
- Structure constructors with component names and default initialization
- Array constructors with type and character length specifications
- A named PARAMETER constant may be part of a complex constant
- Enumerators
- Allocatable components of derived types
- Allocatable scalar variables
- Deferred-length character entities
- PUBLIC types with PRIVATE components and PRIVATE types with PUBLIC components
- ERRMSG keyword for ALLOCATE and DEALLOCATE
- SOURCE= keyword for ALLOCATE
- Type extension
- CLASS declaration
- Polymorphic entities
- Inheritance association
- Deferred bindings and abstract types
- Type-bound procedures
- TYPE CONTAINS declaration
- ABSTRACT attribute
- DEFERRED attribute
- NON_OVERRIDABLE attribute
- GENERIC keyword for type-bound procedures
- FINAL subroutines
- ASYNCHRONOUS attribute and statement
- BIND(C) attribute and statement
- PROTECTED attribute and statement
- VALUE attribute and statement
- VOLATILE attribute and statement
- INTENT attribute for pointer objects
- Default initialization of polymorphic objects
- Reallocation of allocatable variables on the left hand side of an assignment statement when the right hand side differs in shape or length (requires option `/assume:realloc_lhs` if not deferred-length character)
- Bounds specification and bounds remapping on a pointer assignment
- ASSOCIATE construct
- SELECT TYPE construct
- In all I/O statements, the following numeric values can be of any kind: UNIT=, IOSTAT=
- NAMELIST I/O is permitted on an internal file
- Restrictions on entities in a NAMELIST group are relaxed
- Changes to how IEEE Infinity and NaN are represented in formatted input and output

- FLUSH statement
- WAIT statement
- ACCESS='STREAM' keyword for OPEN
- ASYNCHRONOUS keyword for OPEN and data transfer statements
- ID keyword for INQUIRE and data transfer statements
- POS keyword for data transfer statements
- PENDING keyword for INQUIRE
- The following OPEN numeric values can be of any kind: RECL=
- The following READ and WRITE numeric values can be of any kind: REC=, SIZE=
- The following INQUIRE numeric values can be of any kind: NEXTREC=, NUMBER=, RECL=, SIZE=
- Recursive I/O is allowed in the case where the new I/O being started is internal I/O that does not modify any internal file other than its own
- IEEE Infinities and NaNs are displayed by formatted output as specified by Fortran 2003
- BLANK, DECIMAL, DELIM, ENCODING, IOMSG, PAD, ROUND, SIGN, SIZE I/O keywords
- DC, DP, RD, RC, RN, RP, RU, RZ format edit descriptors
- In an I/O format, the comma after a P edit descriptor is optional when followed by a repeat specifier
- Rename of user-defined operators in USE
- INTRINSIC and NON_INTRINSIC keywords in USE
- IMPORT statement
- Allocatable dummy arguments
- Allocatable function results
- PROCEDURE declaration
- The keyword MODULE may be omitted from MODULE PROCEDURE in a generic interface block when referring to an external procedure
- Procedure pointers
- ABSTRACT INTERFACE
- PASS and NOPASS attributes
- The COUNT_RATE argument to the SYSTEM_CLOCK intrinsic may be a REAL of any kind
- Execution of a STOP statement displays a warning if an IEEE floating point exception is signaling
- MAXLOC or MINLOC of a zero-sized array returns zero if the option `/assume:noold_maxminloc` is specified.
- Type inquiry intrinsic functions
- COMMAND_ARGUMENT_COUNT intrinsic
- EXTENDS_TYPE_OF and SAME_TYPE_AS intrinsic functions
- GET_COMMAND intrinsic
- GET_COMMAND_ARGUMENT intrinsic
- GET_ENVIRONMENT_VARIABLE intrinsic

- IS_IOSTAT_END intrinsic
- IS_IOSTAT_EOR intrinsic
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC intrinsics allow CHARACTER arguments
- MOVE_ALLOC intrinsic
- NEW_LINE intrinsic
- SELECTED_CHAR_KIND intrinsic
- The following intrinsics take an optional KIND= argument: ACHAR, COUNT, IACHAR, ICHAR, INDEX, LBOUND, LEN, LEN_TRIM, MAXLOC, MINLOC, SCAN, SHAPE, SIZE, UBOUND, VERIFY
- ISO_C_BINDING intrinsic module
- IEEE_EXCEPTIONS, IEEE_ARITHMETIC and IEEE_FEATURES intrinsic modules
- ISO_FORTRAN_ENV intrinsic module

The following is a partial list of Fortran 2003 features that are unimplemented or are known not to work in this release.

- User-defined derived type I/O
- Parameterized derived types
- Transformational intrinsics, such as MERGE and SPREAD, in initialization expressions

The Intel® Fortran Compiler also supports some 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.

4 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).

4.1 What's New in Intel® MKL 11.0 update 5

- Improved SMP LINPACK performance for 3rd and 4th Generation Intel® Core™ microarchitectures
- Improved matrix generation time for Intel® Optimized MP LINPACK Benchmark for Clusters
- BLAS:
  - Optimized {Z,D}GEMM and double-precision real/complex Level 3 BLAS functions on Intel® Advanced Vector Extensions 2 (Intel® AVX2)
  - Optimized *SYR2K and *HER2K on the Intel® MIC Architecture
  - Optimized DAXPY on Intel® AVX2
- LAPACK:
  - Improved performance of ?GESVD for small sizes like M,N<10
- DFT:
  - Improved documentation for DFTI compute functions data layout
  - Improved performance of workloads specific for GENE application on Intel Xeon® E5-series (Intel® AVX) and 4th generation Intel Core processors (Intel® AVX2)
  - Added scaling capability to large real-to-complex FFTs
- Added examples for Reverse Communication Interface (RCI) in Intel Extended Eigensolver
- Added live links to Intel MKL code examples:

    - The HTML version of the Intel MKL Reference Manual (available from http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/) provides hyperlinks from references to specific code examples so that when you click on an example, your Web browser displays the code. See, for example, the links from the documentation on Fourier Transform Functions and Nonlinear Optimization Problem Solvers
- **Known Limitation: MKL CTRMM may not return bitwise-identical results on some architectures**
  Running in CNR mode on all systems supporting the SSE4.2 instruction set, MKL CTRMM may not return bitwise-identical results if the input matrices contain NaN values. To get bitwise-identical results, please set the environment variable MKL_CBWR to COMPATIBLE
-

4.2 What's New in Intel® MKL 11.0 update 4

- Corrections to reported problems – bug fix list
- Included a fix for SVD multithreading issue

4.3 What's New in Intel® MKL 11.0 Update 3

- Corrections to reported problems – bug fix list
- BLAS:
    - Improved serial and multithreaded performance of DGEMM on 2nd and 3rd Generation Intel® Core™ microarchitecture
- Linpack:
    - Updated the Intel® Optimized MP LINPACK Benchmark for Clusters package to HPL 2.1
- Sparse BLAS:
    - Improved performance of DCOOMM on Intel® Advanced Vector Extensions 2 (Intel® AVX2)
- LAPACK:
    - Parallelized ?LASET, ?LACPY, ?LANGE, ?LANSY
- FFT:
    - Improved Complex-to-complex power-of-2 FFT performance on Intel® AVX2
- VSL:
    - Improved performance of SFMT19937 Basic Random Number Generator (BRNG) on Intel® AVX2
- Cluster FFT:
    - Improved hybrid mode (MPI + OpenMP) Cluster FFT performance
- Data Fitting:
    - Improved performance of df?construct1d function for linear and Hermite/Bessel/Akima cubic types of splines on Intel® Xeon® X5570 and Intel® Xeon® E5-2690 CPUs series

4.4 What's New in Intel® MKL 11.0 Update 2

- New Intel® MKL Extended Eigensolver:
  - Intel® MKL Extended Eigensolver is a high performance package for solving symmetric standard or a generalized symmetric-definite eigenvalue problems on matrices in dense, LAPACK banded, and sparse (CSR) formats. It is based on an innovative fast and stable numerical algorithm named Feast (see Attributions section below)
- Sparse BLAS:
  - Improved performance of 0-based DCSRMM significantly
- LAPACK:
  - Improved performance of parallel versions of ?(OR/UN)G(LQ/QL/QR/RQ) functions significantly
- ScaLAPACK:
  - Updated version to 2.0.2. New functions introduced include:
    - P?HSEQR: Nonsymmetric Eigenvalue Problem
    - P?SYEVR/P?HEEVR: MRRR (Multiple Relatively Robust Representations) algorithm
- Cluster FFT:
  - Implemented transposed order in multidimensional Cluster FFT transforms, including FFTW2 wrappers
- VSL:
  - Supported ICDF (Inverse cumulative distribution function) method in VSL Lognormal RNG
  - Added "const" specifier to declarations of Data Fitting and VSL Summary Statistics functions
- Data Fitting:
  - Improved performance of df[d/s]Interpolate1D, df[d/s]InterpolateEx1D, df[d/s]SearchCells1D, df[d/s]SearchCellsEx1D routines for default/quasi-uniform partition, sorted interpolation sites in scalar (number of interpolation sites is 1) and vector cases for Intel® Xeon® processor X5570 and Intel® Xeon® processor E5-2600
  - Supported DF_DISABLE_CHECK_FLAG parameter in dfiEditVal editor to improve performance for small number of interpolation sites (fewer than one dozen) by disabling checking of the correctness of parameters in Data Fitting routines
- Transposition:
  - Parallelized general out-of-place matrix transposition (mkl_?omatcopy2), improving its performance significantly
- Service functions:
  - Added mkl_peak_mem_usage function which provides information about peak memory amount used by Intel MKL Memory Allocator
  - Added mkl_calloc and mkl_realloc functions extending Intel® MKL Memory Allocator functionality to standard C library memory allocation API

- Enhanced SMP LINPACK with residual check:
    - It returns error code 1 if a failure is detected and prints conclusion if resulting residuals are ok to pass precision check or not. Please note that residuals might slightly vary from run-to-run on the same matrix if conditional numerical reproducibility mode is not turned on. The check ensures that results are reliable.

4.5 What's New in Intel® MKL 11.0 Update 1

- Sparse BLAS
    - Optimized CSRMV functionality for complex conjugate transpose and Hermitian cases
- PARDISO
    - Imaginary part of the diagonal values for Hermitian matrices are ignored
- Cluster FFT
    - Improved hybrid Cluster FFT (MPI + OpenMP*) performance up to 2 times
    - A new Cluster FFT algorithm (Segment of Interest FFT) that uses less communication was implemented for 1D FFTs and it can be enabled by setting the environment variable "MKL_CFFT_SOI_ENABLE" to "YES" or "1" — see more info in MKL documentation
- VSL
    - Added support of VSL_SS_METHOD_FAST_USER_MEAN method for computation of descriptive Summary Statistics estimates given user-provided mean
    - Improved performance of VSL_SS_METHOD_FAST method for computations of descriptive Summary Statistics estimates on Intel® Xeon® processor E5-2690 CPU
- Transposition
    - Improved performance of Out-of-place transposition on 2nd generation Intel® Core™ microarchitecture (up to 7x)
- Service functions
    - Removed seven service functions with obsolete names (see more details at http://intel.ly/OqbZEL)

4.6 What's New in Intel® MKL 11.0

- Conditional Bitwise Reproducibility (CBWR): New functionality in Intel MKL now allows you to balance performance with reproducible results by allowing greater flexibility in code branch choice and by ensuring algorithms are deterministic. See the Intel MKL User's Guide for more information. Refer to the CBWR KB Article (http://intel.ly/P4yRXR) for more information.
- Intel MKL also introduces optimizations using the new Intel® Advanced Vector Extensions 2 (AVX2) including the new FMA3 instructions. See the KB Article on support for Intel® AVX2 (http://intel.ly/PVmq3h) for more information.
- BLAS:
    - Improved DSYRK/SSYRK performance for 64-bit programs supporting Intel® Advanced Vector Extensions (Intel® AVX)

- LAPACK:
    - Optimized [S/D]GETRF, [S/D]POTRF, [S/D]GEQRF, [S/D]GELQF, [S/D]GEQLF, and [S/D]GERQF for native execution on Intel MIC Architecture
    - Introduced support for LAPACK version 3.4.1
- FFT :
    - Added configuration parameter DFTI_THREAD_LIMIT which limits the number of threads per descriptor
    - Added support for 1D real-to-complex transforms with sizes given by 64-bit prime integers
- VML /VSL:
    - Optimized MT19937, MT2203, MRG32k3a BRNGs, and discrete Uniform and Geometric RNGs for native execution on Intel MIC Architecture
    - Improved performance of viRngGeometric on Intel® Advanced Vector Extensions (Intel AVX)
    - Implemented threading in Data Fitting Integrate1d function
- Transposition: Parallelized in-place transposition of square matrices with leading dimensions greater than the matrix size for single and double precisions improving its performance significantly
- Implemented local threading control function (mkl_set_num_threads_local) which increases flexibility in threading control
- Link Line Advisor:
    - Added Help-Me functionality for selecting architecture (IA-32/Intel® 64) and interface layer (LP64/ILP64)

4.7 Deprecated and Removed Features

Please refer to the Intel® MKL Deprecations article (http://intel.ly/LkZKGL) for more information.

- Intel® MKL no longer supports execution on processors that do not support the Intel® SSE2 instruction set extensions (Intel® Pentium III and earlier.)
- Microsoft Windows* System PATH environment variable is no longer set during installation. If you will be running applications that use the DLL form of Intel® MKL, you will need to ensure that the folder containing the Intel® MKL DLLs is included in the PATH value
- Removed Intel MKL GNU Multiple Precision* (GMP) function interfaces
- Disabled timing function mkl_set_cpu_frequency() to perform useful work — use mkl_get_max_cpu_frequency(), mkl_get_clocks_frequency(), and mkl_get_cpu_frequency() as described in the Intel MKL Reference Manual
- Removed MKL_PARDISO constant — used MKL_DOMAIN_PARDISO to specify the PARDISO domain with the mkl_domain_set_num_threads() function
- Removed special backward compatibility functions for convolution and correlation functions in Intel MKL 10.2 update 4
- Removed the OpenMP* static runtime library from the Windows* version of Intel MKL and Intel® compilers
- Documentation:

- The Intel MKL Reference Manual in HTML format is no longer available with the product

4.8 Known Issues

A full list of the known limitations of this release can be found in the Knowledge Base for the Intel® MKL at http://intel.ly/ptEfAP

4.9 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/)

5 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, Itanium, MMX, Pentium, VTune, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

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