Intel® Parallel Studio XE 2016 Suites

**Vectorizer** – Boost Performance By Utilizing Vector Instructions / Units

- Intel® Advisor XE - *Vectorizer Advisor* identifies new vectorization opportunities as well as improvements to existing vectorization and highlights them in your code. It makes actionable coding recommendations to boost performance and estimates the speedup.

**Scalable MPI Analysis** – Fast & Lightweight Analysis for 32K+ Ranks

- Intel® Trace Analyzer and Collector add *MPI Performance Snapshot* feature for easy to use, scalable MPI statistics collection and analysis of large MPI jobs to identify areas for improvement.

**Big Data Analytics** – Easily Build IA Optimized Data Analytics Application

- Intel® Data Analytics Acceleration Library (Intel® DAAL) will help data scientists speed through big data challenges with optimized IA functions.

**Standards** – Scaling Development Efforts Forward

- Supporting the evolution of industry standards of *OpenMP*, *MPI, Fortran and C++* Intel® Compilers & performance libraries.
Detail Slides
Intel® C/C++ and Fortran Compilers 16.0
Get best performance with latest standards

• More of C++14, generic lambdas, member initializers and aggregates
• More of C11, Static_assert, Generic, Noreturn, and more
• BLOCK_LOOP Pragma/Directive Syntax
• Intel® Cilk™ Plus Combined Parallel & SIMD loops
• OpenMP 4.0 C++ User Defined Reductions, Fortran Array Reductions
• Vectorization enhancements
• OpenMP 4.1 asynchronous offloading, simdlen, simd ordered
• Offload Feature Enhancements for Intel® Xeon Phi™
• F2008 Submodules, Impure Elemental Functions
• F2015 TYPE(*), DIMENSION(..), RANK intrinsic, attributes for args with BIND
C++14 Standard Support

Generic Lambdas

```
auto glambda = [ ] (auto a) { return a; };
```

Generalized lambda captures

```
int x = 4;
int z = [&r = x, y = x+1]
    {
        r += 2;   // set x to 6; "R is for Renamed Ref"
        return y+2; // return 7 to initialize z
    }();       // invoke lambda
```

Digit Separators

```
[[deprecated]] attribute
```

Function return type deduction

Member initializers and aggregates

Feature Test macros

- Test for header files existence
- Test for compiler features

```
#if __has_include("shared_mutex") // use standard header
#elif __has_include("boost/shared_mutex.h") // use from BOOST
#endif
...
#else
  // no constexpr available
#elif __cpp_constexpr == 200704
  // c++11 constexpr available
#else
  // c++14 constexpr available
#endif
```
C11 Standard Support

New Feature Support

- Unicode strings
- C11 anonymous unions

New Keyword Support

<table>
<thead>
<tr>
<th>_Alignas</th>
<th>_Alignof</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Static_assert</td>
<td>_Thread_local</td>
</tr>
<tr>
<td>_Noreturn</td>
<td>_Generic</td>
</tr>
</tbody>
</table>

_Generic : template like functionality in C w/o template support in the language
defines different actions based on different arguments types

```c
#define pow(X) _Generic((X), long double: powl, \  default: pow, \  float: powf)(X)
```

when pow() is called with a long double argument it calls powl()
when pow() is called with a float, it calls powf(),
when pow() is called with other types it calls pow()
BLOCK_LOOP Pragma/Directive Syntax

C++:

```cpp
#pragma block_loop [clause[,clause]...]
#pragma noblock_loop
```

Fortran:

```fortran
!DIR$ BLOCK_LOOP [clause[[,] clause]...] 
!DIR$ NOBLOCK_LOOP
```

clause → factor(expr), level(levels)

- Enables or disables loop blocking for the immediately following nested loops
- Works seamlessly with other directives including SIMD directive

```cpp
#pragma block_loop factor(256) level(1:2)
for( j=1; j<n ; j++){
    f = 0;
    for( i=1; i<n; i++){
        f = f + a[i] * b[i];
    }
    c[j] = c[j] + f;
}
```

```fortran
for( jj=1 ; jj<n/256+1; jj+){
    for( ii=1; ii<n/256+1; ii++){
        for( j=(jj-1)*256+1; min(jj*256, n); j++){
            f = 0;
            for ( i=(ii-1)*256+1; i<min(ii*256,n); i++){
                f = f + a[i] * b[i];
            }
            c[j] = c[j] + f;
        }
    }
}
```
Intel® Cilk™ Plus
Combined Parallel & SIMD loops

- New combined Parallel + Vector loop

```c
_Cilk_for _Simd (int i = 0; i < N; ++i)
    // Do something
```

or

```c
#pragma simd
_Cilk_for (int i = 0; i < N; ++i)
    // Do something
```

- Combined loop gives both thread-parallelism and vectorization

- Behaves approximately like this pair of nested loops

```c
_Cilk_for (int i_1 = 0; i_1 < N; i_1 += M)
    for _Simd (int i = i_1; i < i_1 + M; ++i)
        // Do something
```

- The chunk size, M, is determined by the compiler and runtime
Improvements in Vectorization

Intel® Cilk™ Plus and OpenMP* 4.0

**simdlen** (i.e. `vectorlength`) and **safelen** for loops
- Usable with **#pragma simd** (Intel Cilk™ Plus) and **omp simd** (OpenMP*)

Array reductions
- Fortran only (available in Beta update)

User-defined reductions
- Supported for parallel in C/C++ for POD types. No support for Fortran, SIMD, or non-POD types (C++)

**omp-simd** `collapse(N)` clause
- Available in a Beta update

FP-model honoring for simd loops
- No more undesirable speculations and transformations

Adjacent gathers optimization
- Replaces series of gathers with series of vector loads & sequence of permutes

Ordered blocks in SIMD context
- `ordered with simd` specifies structured block simd loop or SIMD function

```c
#pragma simd safelen(9) simdlen(4)
for (i = 0; i < N; i++) {
    a[i] = a[i] + a[i-9];
}
```

```c
#pragma omp simd safelen(4,8)
for (i = 0; i < N; i++) {
    foo(a[i]); //versions for VL=4,8
}
```

```c
#pragma omp simd collapse(2)
for (i = 0; i < N; i++)
for (j = 0; j < 4; j++) {
    a[5*i+j] += da[i]*cf[j];
}
```

```c
#pragma omp ordered [simd]
structured code block
#pragma simdoff
structured code block
```
OpenMP* 4.1 Extensions

Support for Features in OpenMP* 4.1 Technical Report 3

- Non-structured data allocation
  - `omp target [enter | exit ] data`
  - Begins/ends data region for the target
  - `map` specifies data to be mapped/unmapped with motion

- Asynchronous offload
  - `nowait clause on omp task`
  - Enables asynchronous offload using existing tasking model with `nowait clause`

- Dependence (signal)
  - `depend clause on omp task`
  - Enables offload with dependence

- Map clause extensions
  - Modifiers `always` and `delete`

Available for C/C++ and Fortran
Offload Feature Enhancements for Intel® Xeon Phi™

New Features available for C/C++

- Offload of structures with pointer members
  - Enabling offload inside member function
  - ‘pointer->field’ is now allowed in offload constructs
  - Structured objects are copied without pointer fields. This makes them bitwise copyable
  - Pointer fields transfer must be pointed out in the pragma explicitly

- Offload with MIC-only memory allocation using new modifiers
  - `targetptr` for MIC-only memory allocation by offload runtime
  - `preallocated targetptr` for informing offload runtime about user allocated memory on MIC

- Offload using Streams - new `stream` clause and associated APIs
  - Offload multiple concurrent computations to a coprocessor from a single CPU thread
  - `stream` is a logical queue of offloads
  - Offloads in any one stream complete in the order they were issued to the stream

Performance Improvements for C/C++ and Fortran

- Asynchronous offload
- Memory Allocation and Data Transfers
Annotated Source Listing (ASL)

A modified copy of a source code file with each line numbered and compiler diagnostics inserted after correspondent lines

The listing file can be generated in either a plain text or html format

Example:

```c
int* foo(int* a, int* b, int upperbound)
{
    int* c = new int[upperbound];
    #pragma omp parallel for
    OpenMP DEFINED LOOP WAS PARALLELIZED
    for (int i = 0; i < upperbound; ++i) {
        LOOP BEGIN at Test/library.cpp(5,2)
        <Peeled>
        LOOP END
        LOOP BEGIN at Test/library.cpp(5,2)
            remark #25460: No loop optimizations reported
        LOOP END
        c[i] = a[i] + b[i];
    }
    return c;
}
```
New library targeting data analytics market

- **Customers**: analytics solution providers, system integrators, and application developers (FSI, Telco, Retail, Grid, etc.)

- **Key benefits**: improved time-to-value, forward-scaling performance and parallelism on IA, advanced analytics building blocks

Key features

- Building blocks highly optimized for IA to support all data analysis stages
- Support batch, streaming, and distributed processing with easy connectors to popular platforms (Hadoop, Spark) and tools (R, Python, Matlab)
- Flexible interfaces for handling different data sources (CSV, MySQL, HDFS, RDD (Spark))
- Rich set of operations to handle sparse and noisy data
- C++ and Java APIs
Intel® Math Kernel Library (Intel® MKL) 11.3
Better performance with new two-stage API for Sparse BLAS routines

Additional Sparse Matrix Vector Multiplication API
- new two-stage API for Sparse BLAS level 2 and 3 routines

MKL MPI wrappers
- all MPI implementations are API-compatible but MPI implementations are not ABI-compatible
- MKL MPI wrapper solves this problem by providing an MPI-independent ABI to MKL

Support For Batched Small Matrix multiplication
- a single call executes multiple independent ?GEMM operation simultaneously

Support for Philox4x35 and ARS5 RNG
- two new pseudorandom number generators with a period of 2^128 are highly optimized for multithreaded environment

Sparse Solver SMP improvements
- significantly improved overall scalability for Intel Xeon Phi coprocessors and Intel Xeon processors
Intel® Integrated Performance Primitives 9.0 Beta

Improved Threading Support

Additional optimization for Intel® Quark™, Intel® Atom™, and the processors with Intel® AVX2 instructions support

- Intel® Quark™: data compression, cryptography optimization
- Intel® Atom™: computation vision, image processing optimization
- Intel® AVX2: computer vision, image processing optimization

New APIs to support external threading

Improved CPU dispatcher

- Auto-initialization. No need for the CPU initialization call in static libraries.
- Code dispatching based on CPU features

Optimized cryptography functions to support SM2/SM3/SM4 algorithm

Custom dynamic library building tool

New APIs to support external memory allocation
Intel® Threading Building Blocks 4.3
Improved cross-platform support

Fully supported tbb::task_arena

- Task arenas provide improved control over workload isolation and the degree of concurrency

Dynamic replacement of standard memory allocation routines for OS X*

- Utilize the powerful TBB scalable allocator easily on OS X

Binary files for 64-bit Android* applications were added as part of the Linux* OS package

- Improvements to the Flow Graph features

Don’t forget to check out Flow Graph Designer

- Several Improvements to examples and documentation
Intel® Advisor XE – New! Vectorization Advisor

Data Driven Vectorization Design

Have you:

- Recompiled with AVX2, but seen little benefit?
- Wondered where to start adding vectorization?
- Recoded intrinsics for each new architecture?
- Struggled with cryptic compiler vectorization messages?

Breakthrough for vectorization design

- What vectorization will pay off the most?
- What is blocking vectorization and why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?
Intel® Advisor XE – Vectorization Advisor
Provides the data you need for high impact vectorization

Compiler diagnostics + Performance Data = All the data you need in one place
- Find “hot” un-vectorized or “under vectorized” loops.
- Trip counts

Recommendations – How do I fix it?
Correctness via dependency analysis
- Is it safe to vectorize?

Memory Access Patterns analysis
- Unit stride vs Non-unit stride access, Unaligned memory access, etc.
Intel® VTune™ Amplifier XE 2016 Beta
Enhanced GPU and Microarchitecture Profiling

New OS and IDE support: Visual Studio* 2015 & Windows* 10 Threshold

GPU profiling
  - GPU Architecture Annotation Diagram
  - GPU profiling on Linux (OpenCL, Media SDK)

Microarchitecture tuning
  - General Exploration analysis with confidence indication
  - Driverless ‘perf’ EBS with stacks
Intel® VTune™ Amplifier XE 2016 Beta Improved OpenMP and Hybrid Support

Intel OpenMP analysis enhancements

- Precise trace-based imbalance calculation that is especially useful for profiling of small region instances
- Classification and issue highlighting of potential gains, e.g., imbalance, lock contention, creation overhead, etc.
- Detailed analysis of barrier-to-barrier region segments

MPI+OpenMP: multi-rank analysis on a compute node

- Per-rank OpenMP potential gain and serial time metrics
- Per-rank Intel MPI communication busy wait time detection
Improved Hybrid Profiling Support

Rank Selection/Multi-selection in ITAC for Profiling in VTune

- Automatically generate MPI command line script
- User will run command line script and automatically launch profiling of selected ranks by VTune

Run tool on specified ranks with gtool

```bash
$ mpirun -gtool "amplxe-cl -c advanced-hotspots \ -r my_dir:all=node-wide" -n 4 -ppn 2 my_mpi_app
```
MPI Performance Snapshot
Scalable profiling for MPI and Hybrid

**Lightweight** – Low overhead profiling for 32K+ Ranks

**Scalability** – Performance variation at scale can be detected sooner

**Identifying Key Metrics** – Shows PAPI counters and MPI/OpenMP imbalances
Intel® Cluster Checker 3.0
Cluster system expertise packaged into a utility

Provides Assistance
- Cluster Health Checks (on-demand, background)
- Diagnoses and remedies for common issues
- Compliance with Intel® Cluster Ready spec

Simplifies Cluster Computing Platforms
- Reduces need for specialized expertise
- Enables cluster health checks by applications
- Extensible and customizable, API

Cluster
Data Collectors
Cluster Health Checks
Diagnoses and remedies for common issues
Compliance with Intel® Cluster Ready spec

Database
Analysis Checking for Issues
Expert System
Suggesting Remedies
Results

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.
## Beta Webinars

<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 14</td>
<td>Vectorize or Die - unlock code modernization performance secrets</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>April 15</td>
<td>What's New in the Intel® Parallel Studio XE 2016 Beta</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 5</td>
<td>Fast, light-weight, scalable MPI performance analysis</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 6</td>
<td>Latest Intel® VTune™ Amplifier XE Improvements</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 13</td>
<td>What's New in Intel® Fortran 16.0 for Intel® Parallel Studio XE 2016</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 20</td>
<td>New C/C++ Language Features in Intel® C/C++ Compiler 16.0</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>June 3</td>
<td>Introducing Intel® Cluster Checker 3.0</td>
<td>[REGISTER NOW]</td>
</tr>
<tr>
<td>9:00 AM Pacific</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Visit the Intel® Parallel Studio XE 2016 Beta page for details and to view all available webinars:  
[bit.ly/psxe2016beta]
Summary/Call to Action

Participate in the Beta Program today!

- Register at bit.ly/psxe2016beta

Submit Feedback via Intel® Premier Support

Tell us about your experiences using the Intel® Parallel Studio XE 2016 Beta
Back Up
Intel® Software Tools Roadmap

Data Center Tools

High Performance Computing / Enterprise

Intel® Parallel Studio XE 2015 Cluster Edition
Intel® Parallel Studio XE 2015 Professional Edition
Intel® MPI Library (IMPI) 5.0
Intel® Trace Analyzer and Collector 9.0

Cluster Edition 2016
Professional Edition NEXT
Intel® MPI Library (IMPI) NEXT
Intel® Trace Analyzer and Collector NEXT

Professional Edition 2016
Composer Edition NEXT
Intel® Inspector XE
Intel® VTune™ Amplifier XE
Intel® Advisor XE

Standards

IMPI 5.0 – MPI-3.0 Initial Support
OpenMP 4.0 Support

IMPI 5.1 (Full MPI 3.0 Support)
OpenMP 4.1 Initial Support
C++ 14 Initial Support
Fortran 2015 Initial Support

IMPI 5.1.1 (Full MPI 3.1 Support)
OpenMP 4.1 Initial Support
C++ 14 Initial Support
Fortran 2015 Initial Support

2 Supports Intel® Xeon® Phi™ coprocessors. 3 For Linux. Supports Haswell. Supports Intel® Xeon® Phi™ coprocessors. * Other names and brands may be claimed as the property of others. All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.
Intel® Parallel Studio XE 2016 Layout

Shared components
- documentation_<version>
- ide_support_<version>
- debugger_<version>
- samples_<version>
- compiler_and_libraries_<version>_<update>_<pkg>

Compiler and Libraries for specific target OSes
- bin
- compiler
  - include
    - <target_arch>
    - <clik>
  - lib
    - <target_arch>_<target_os_subset>
- <target_OS>
  - ipp
  - mkl
  - tbb
  - mpi

Symbolic links
- parallel_studio_xe_<version>_<update>_<pkg_psxe>
Licensing Changes

New License File

Intel® Parallel Studio XE 2016 employs new Feature-based names

Feature-based names require issuing a new license file

- Customer must obtain new license file to use Intel® Parallel Studio XE 2016
- New license file supports all releases
  - Contains both new Feature-based (i.e. supports Intel® Parallel Studio XE 2016) and previous Product-based feature codes (i.e. supports Intel® Parallel Studio XE 2015 and earlier)

Existing license file provide ongoing support for releases prior to Intel® Parallel Studio XE 2016
Licensing Changes

Named-User System Locked

Named-User license now a System Locked license

- This attribute is not applicable to Floating license
- License generation now delayed to product activation (during installation) from serial number registration
- User system host ID collected by installer for product activation and license file generation
- Offline activation supported
- A specific number (3 per EULA) of simultaneous activations allowed for installation on multiple systems
- Customer can release un-used/un-needed activations for reuse