Intel® Parallel Studio XE 2017
Create Faster Code - Faster
Intel® Parallel Studio XE

**Performance Libraries**
- Intel® Math Kernel Library
  Optimized Routines for Science, Engineering & Financial
- Intel® Data Analytics Acceleration Library
  Optimized for Data Analytics & Machine Learning

**Profiling, Analysis & Architecture**
- Intel® VTune™ Amplifier
  Performance Profiler
- Intel® Inspector
  Memory & Threading Checking

**Cluster Tools**
- Intel® Cluster Checker
  Cluster Diagnostic Expert System
- Intel® Trace Analyzer & Collector
  MPI Profiler

**Intel® C/C++ & Fortran Compilers**

**Intel® MPI Library**
- Intel® Integrated Performance Primitives
  Image, Signal & Data Processing
- Intel® Threading Building Blocks
  Task Based Parallel C++ Template Library

**Intel® Distribution & for Python**
- Performance Scripting

**Intel® Advisor**
- Vectorization Optimization & Thread Prototyping

**Intel® Inspector**
- Memory & Threading Checking

**Intel® VTune™ Amplifier**
- Performance Profiler

**Intel® Math Kernel Library**
- Optimized Routines for Science, Engineering & Financial

**Intel® Data Analytics Acceleration Library**
- Optimized for Data Analytics & Machine Learning

**Intel® Cluster Checker**
- Cluster Diagnostic Expert System

**Intel® Trace Analyzer & Collector**
- MPI Profiler

**Intel® MPI Library**
- Intel® Integrated Performance Primitives
  Image, Signal & Data Processing
- Intel® Threading Building Blocks
  Task Based Parallel C++ Template Library

**Intel® C/C++ & Fortran Compilers**

**Intel® Distribution for Python**
- Performance Scripting

*Other names and brands may be claimed as the property of others.
## What’s Inside
Intel® Parallel Studio XE 2017

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® C++ Compiler</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Fortran Compiler</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Distribution for Python*</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Math Kernel Library – fast math library</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Integrated Performance Primitives – image, signal &amp; data processing</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Threading Building Blocks – threading library</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Data Analytics Acceleration Library – machine learning &amp; analytics</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® VTune™ Amplifier XE – performance profiler</td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Intel® Advisor – vectorization optimization and thread prototyping</td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Intel® Inspector – memory and thread debugging</td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® MPI Library – message passing interface library</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Intel® Trace Analyzer and Collector – MPI Tuning and Analysis</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Intel® Cluster Checker – cluster diagnostic expert system</td>
<td></td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

| Rogue Wave IMSL® Library – Fortran numerical analysis | Bundle or Add-on | Add-on | Add-on |

Additional configurations including, floating and academic, are available at: [http://intel.ly/perf-tools](http://intel.ly/perf-tools)
Staying current with Support for the Latest Standards, Operating Systems & Processors

Enhanced C11 and C++14 standards support

- Sized deallocation
- Relaxed constexpr restrictions
- Variable templates
- Single-Quotation-Mark as a digit separator

Operating systems

- Debian* 7.0, 8.0; Fedora* 23, 24; Red Hat Enterprise Linux* 6, 7; SuSE LINUX Enterprise Server* 11,12; Ubuntu* 14.04 LTS 16.04 LTS, 16.04
- macOS* 10.11

Enhanced Fortran 2008 and draft 2015 standards support

- Implied-shape PARAMETER arrays
- 2008 bind C internal procedures
- Extended EXIT for all named blocks
- Pointer initialization

Latest processors

- Support and tuning added for the latest Intel® Xeon Phi™ codenamed Knights Landing and AVX-512
Intel® Compilers for Parallel Studio XE 2017

What’s new in Intel® C++ 17.0 and Intel® Fortran 17.0

Productive language-level vectorization & parallelism models for advanced developers driving application performance

Common updates

- Enhanced support for the newest AVX2 and AVX512 instruction sets for the latest Intel® processors (including Intel® Xeon Phi)
- Enhanced optimization/vectorization reports – register allocation
- Tight integration with Intel® Advisor
- Initial support for OpenMP* 4.5, offering improved vectorization control, new SIMD instructions, and much more

Intel® C++ Compiler

- SIMD Data Layout Template to facilitate vectorization for your C++ code
- Virtual function vectorization capability
- Improved compiler loop and function alignment
- Full support for the latest C11 and C++14 standards

Intel® Fortran Compiler

- Substantial coarray performance improvement – up to twice as fast as previous versions on non-trivial coarray Fortran programs
- Almost complete Fortran 2008 support
- Further interoperability with C (part of draft Fortran 2015)
Boost application performance on Windows* and Linux* Intel® C++ and Fortran Compilers

Boost C++ application performance on Windows* & Linux* using Intel® C++ Compiler (higher is better)

Boost Fortran application performance on Windows* & Linux* using Intel® Fortran Compiler (higher is better)

Windows Estimated SPECfp®_rate_base2006 | Linux Estimated SPECint®_rate_base2006
---|---
PGI 15.10 | Visual C++ 2015 | Intel C++ 17.0 | GCC 6.0 | Intel C++ 17.0 | Intel C++ 17.0
1.05 | 1.39 | 1.03 | 1.28 | 1.13 | 1.71

Windows Relative geomean performance, SPEC® benchmark – higher is better

Windows Linux
PGI Fortran 15.10 | Absoft 15.0 | Intel Fortran 17.0 | Open64 4.5.2 | PGI Fortran 16.4 | gFortran 6.1.0 | Intel Fortran 17.0
1.00 | 1.29 | 1.86 | 1.64 | 1.14 | 1.26 | 1.87

Configuration: Windows hardware: Intel® (R) Xeon® (R) CPU E3-1245 v5 @ 3.50GHz; HT enabled; TB enabled; 32 GB RAM; Linux hardware: Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.80GHz, 256 GB RAM; Hyperthreading enabled.
Software: Intel compilers: 15.0, Microsoft® VS C++ + optimizations Compiler Version 19.0.23194 for x86_64; GCC 6.1.0; PGI 15.10, Clang/LLVM 3.8
Linux OS: Red Hat Enterprise Linux Server release 7.1 (Maipo); kernel 3.10.0-293.17.5.68, 64, Windows OS: Windows 10 Pro 15063.1034 N/A Build 10240.
SPEC benchmark: Maipo
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. Software and workloads used in performance tests may have been optimized. Some optimizations may not run on older microprocessors. Some optimizations are intended for use with 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 #20170804.

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 non-Intel microarchitectures. Compiler optimizations in this product may vary by compiler version. The optimization notice for this product can be viewed at https://software.intel.com/content/www/us/en/develop/tools/compilers/optimization-guide.html. Other optimized versions may be available. Notice revision #20170804.

Other brands and names are the property of their respective owners. Performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. Software and workloads used in performance tests may have been optimized. Some optimizations may not run on older microprocessors. Some optimizations are intended for use with 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 #20170804.

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 non-Intel microprocessors. These optimizations may not run on older microprocessors. Some optimizations are intended for use with 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 #20170804.
Impressive Performance Improvement
Intel® Compiler OpenMP* Explicit Vectorization

- Two lines added that take full advantage of both SSE or AVX
- Pragmas ignored by other compilers so code is portable

```c
typedef float complex fcomplex;
const uint32_t max_iter = 3000;
#pragma omp declare simd uniform(max_iter), simdlen(16)
uint32_t mandel(fcomplex c, uint32_t max_iter)
{
    uint32_t count = 1; fcomplex z = c;
    while (cabsf(z) < 2.0f) && (count < max_iter) {
        z = z * z + c; count++;
    }
    return count;
}
```

```
for (int32_t y = 0; y < ImageHeight; ++y) {
    float c_im = max_imag - y * imag_factor;
    #pragma omp simd safelen(16)
    for (int32_t x = 0; x < ImageWidth; ++x) {
        fcomplex in_vals_tmp = (min_real + x * real_factor) + (c_im * 1.0iF);
        count[y][x] = mandel(in_vals_tmp, max_iter);
    }
```

Mandelbrot calculation speedup
Normalized performance data – higher is better

<table>
<thead>
<tr>
<th></th>
<th>Serial</th>
<th>SSE 4.2</th>
<th>Core-AVX2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup</td>
<td></td>
<td>2.48</td>
<td>4.27</td>
</tr>
</tbody>
</table>

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.
Impressive performance improvement
Intel C++ Explicit Vectorization using OpenMP* SIMD

SIMD Speedup on Intel® Xeon® Processor
Normalized performance data – higher is better

Configuration: Intel® Xeon® CPU E3-1270 @ 3.50 GHz Haswell system (4 cores with Hyper-Threading On), running at 3.50GHz, with 32.0GB RAM, L1 Cache 256KB, L2 Cache 1.0MB, L3 Cache 8.0MB, 64-bit Windows* Server 2012 R2 Datacenter. Compiler options: -O3 -Qopenmp -simd -QxSSE4.2 or AVX2: -O3 -Qopenmp -simd -QxCORE-AVX2. For more information go to http://www.intel.com/performance

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source Intel Corporation

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 .
INTEL® DISTRIBUTION FOR PYTHON*
Boost NumPy/SciPy performance with Intel® MKL
Intel® Distribution for Python*

Easy access to High performance Python

- NumPy/SciPy/Scikit-Learn/pandas accelerated with Intel® MKL
  - Close to 100X performance speedups on select functions
  - Includes Python optimized modules for Intel® TBB, Intel® DAAL
- Includes numba, Cython, pyDAAL
- Integrated Distribution, Out-of-the-Box access to performance
- Python 2.7 & 3.5. Windows, Linux, macOS
- Latest Optimizations for Intel Xeon and Intel Xeon Phi™ Processors
- Available as free standalone, via conda* and Intel® Parallel Studio XE 2017
Close to 100X faster for select functions
Profile Python & Go using Intel® VTune™ Amplifier
And Mixed Python / C++ / Fortran

Low Overhead Sampling
- Accurate performance data without high overhead instrumentation
- Launch application or attach to a running process

Precise Line Level Details
- No guessing, see source line level detail
- Mixed Python / native C, C++, Fortran...
- Optimize native code driven by Python
LIBRARIES

Intel® Math Kernel Library
Intel® Data Analytics Acceleration Library
Intel® Integrated Performance Primitives
Intel® Threading Building Blocks
Intel® Math Kernel Library

- Speeds math processing for machine learning, scientific, engineering financial and design applications
- Includes functions for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics and more
- De facto standard APIs for easy switching from other math libraries
- Highly optimized, threaded and vectorized to maximize processor performance
# Components of Intel MKL 2017

<table>
<thead>
<tr>
<th>Linear Algebra</th>
<th>Fast Fourier Transforms</th>
<th>Vector Math</th>
<th>Summary Statistics</th>
<th>And More...</th>
<th>Deep Neural Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>• BLAS</td>
<td>• Multidimensional</td>
<td>• Trigonometric</td>
<td>• Kurtosis</td>
<td>• Splines</td>
<td>• Convolution</td>
</tr>
<tr>
<td>• LAPACK</td>
<td>• FFTW interfaces</td>
<td>• Hyperbolic</td>
<td>• Variation</td>
<td>• Interpolation</td>
<td>• Pooling</td>
</tr>
<tr>
<td>• ScaLAPACK</td>
<td>• Cluster FFT</td>
<td>• Exponential</td>
<td>coefficient</td>
<td>• Trust Region</td>
<td>• Normalization</td>
</tr>
<tr>
<td>• Sparse BLAS</td>
<td></td>
<td>• Log</td>
<td>• Order</td>
<td>• Fast Poisson</td>
<td>• ReLU</td>
</tr>
<tr>
<td>• Sparse Solvers</td>
<td></td>
<td>• Power</td>
<td>statistics</td>
<td>Solver</td>
<td>• Softmax</td>
</tr>
<tr>
<td>• Iterative</td>
<td></td>
<td>• Root</td>
<td>• Min/max</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• PARDISO*</td>
<td></td>
<td>• Vector RNGs</td>
<td>• Variance-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Cluster Sparse Solver</td>
<td></td>
<td></td>
<td>covariance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**New**

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver
- Convolution
- Pooling
- Normalization
- ReLU
- Softmax

---

**Optimization Notice**

Copyright © 2016, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.*
What’s New: Intel® MKL 2017

- Optimized math functions to enable neural networks (CNN and DNN) for deep learning
- Improved ScaLAPACK performance for symmetric eigensolvers on HPC clusters
- New data fitting functions based on B-splines and monotonic splines
- Improved optimizations for newer Intel processors, especially Knight’s Landing Xeon Phi
- Extended TBB threading layer support for all BLAS level-1 functions
INTEL® DATA ANALYTICS ACCELERATION LIBRARY
Intel DAAL Overview

Industry leading performance, C++/Java/Python library for machine learning and deep learning optimized for Intel® Architectures.

Pre-processing > Transformation > Analysis > Modeling > Validation > Decision Making

(De-)Compression

PCA
Statistical moments
Variance matrix
QR, SVD, Cholesky
Apriori

Linear regression
Naïve Bayes
SVM
Classifier boosting
Kmeans
EM GMM

Collaborative filtering
Neural Networks
Example Performance: Intel DAAL vs. Spark* MLLib

PCA (correlation method) on an 8-node Hadoop* cluster based on Intel® Xeon® Processors E5-2697 v3

Table size

<table>
<thead>
<tr>
<th>Table size</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M x 200</td>
<td>4X</td>
</tr>
<tr>
<td>1M x 400</td>
<td>6X</td>
</tr>
<tr>
<td>1M x 600</td>
<td>6X</td>
</tr>
<tr>
<td>1M x 800</td>
<td>7X</td>
</tr>
<tr>
<td>1M x 1000</td>
<td>7X</td>
</tr>
</tbody>
</table>

Configuration Info - Versions: Intel® Data Analytics Acceleration Library 2016, CDH v5.3.1, Apache Spark* v1.2.0; Hardware: Intel® Xeon® Processor E5-2699 v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 128GB of RAM per node; Operating System: CentOS 6.6 x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

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.
What’s New: Intel DAAL 2017

• Neural Networks
• Python API (a.k.a. PyDAAL)
  – Easy installation through Anaconda or pip
• New data source connector for KDB+
• Open source project on GitHub

Fork me on GitHub: https://github.com/01org/daal
INTEL® THREADING BUILDING BLOCKS
## Rich Feature Set for Parallelism

**Intel® Threading Building Blocks (Intel® TBB)**

<table>
<thead>
<tr>
<th>Generic Parallel Algorithms</th>
<th>Flow Graph</th>
<th>Concurrent Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient scalable way to exploit the power of multi-core without having to start from scratch.</td>
<td>A set of classes to express parallelism as a graph of compute dependencies and/or data flow</td>
<td>Concurrent access, and a scalable alternative to containers that are externally locked for thread-safety</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task Scheduler</th>
<th>Concurrent Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sophisticated work scheduling engine that empowers parallel algorithms and the flow graph</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow Graph</th>
<th>Concurrent Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A set of classes to express parallelism as a graph of compute dependencies and/or data flow</td>
<td>Concurrent access, and a scalable alternative to containers that are externally locked for thread-safety</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synchronization Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic operations, a variety of mutexes with different properties, condition variables</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concurrent Containers</th>
<th>Timers and Exceptions</th>
<th>Threads</th>
<th>Thread Local Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent access, and a scalable alternative to containers that are externally locked for thread-safety</td>
<td>Thread-safe timers and exception classes</td>
<td>OS API wrappers</td>
<td>Efficient implementation for unlimited number of thread-local variables</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalable memory manager and false-sharing free allocators</td>
</tr>
</tbody>
</table>
What’s new: Intel® Threading Building Blocks 2017

static_partitioner class
- Helps minimizing overhead of parallel loops

streaming_node class
- Enables heterogeneous streaming computations within the flow graph.

Added method to isolate execution of a group of tasks or an algorithm from other tasks submitted to the scheduler. A preview feature for 2017.

Python* module is added to replace Python's thread pool class.

Graph/stereo example is added.

Improvements to graph/fgbzip example (added async_msg usage example)
INTEL® INTEGRATED PERFORMANCE PRIMITIVES
Intel® IPP Domain Applications

**Image Processing**
- Medical Imaging
- Computer Vision
- Digital Surveillance
- Biometric Identification
- Automated Sorting
- ADAS
- Visual Search

**Signal Processing**
- Games (sophisticated audio content or effects)
- Echo cancellation
- Telecommunications
- Energy

**Data Compression & Cryptography**
- Data Centers
- Enterprise data managements
- ID verification
- Smart cards/wallets
- Electronic signature
- Information security / cybersecurity
What’s new:
Intel® Integrated Performance Primitives 2017

Extended optimization for Intel® AVX-512 on KNL and Intel® Xeon® processors

Intel® IPP Platform-Aware APIs in the image and signal processing domains are added to support external threading and 64-bit data length

Significantly improved performance of zlib compression functions is Extension of IPP optimized functionality in OpenCV

Limited pre-silicon optimizations for KNH and CNL EP/XE server
**INTEL SOFTWARE ANALYSIS TOOLS**

Intel® Performance Snapshots

Intel® VTune™ Amplifier XE Performance Profiler

Intel® Inspector Memory & Thread Debugger

Intel® Advisor Vectorization Optimization and Thread Prototyping
Intel® Performance Snapshots

Three Fast Ways to Discover Untapped Performance

Is your application making good use of modern computer hardware?

- Run a test case during your coffee break.
- High level summary shows which apps can benefit most from code modernization and faster storage.

Pick a Performance Snapshot:

- **Application** – for non-MPI apps
- **MPI** – for MPI apps
- **Storage** – for systems. Servers and workstations with directly attached storage.


Also included with Intel® Parallel Studio and Intel® VTune™ Amplifier products.
INTEL® VTUNE™ AMPLIFIER XE PERFORMANCE PROFILER
Intel® VTune™ Amplifier
Faster, Scaleable Code, Faster

Get the Data You Need
- Hotspot (Statistical call tree), Call counts (Statistical)
- Thread Profiling – Concurrency and Lock & Waits Analysis
- Cache miss, Bandwidth analysis...
- GPU Offload and OpenCL™ Kernel Tracing

Find Answers Fast
- View Results on the Source / Assembly
- OpenMP Scalability Analysis, Graphical Frame Analysis
- Filter Out Extraneous Data – Organize Data with Viewpoints
- Visualize Thread & Task Activity on the Timeline

Easy to Use
- No Special Compiles – C, C++, C#, Fortran, Java, ASM
- Visual Studio® Integration or Stand Alone
- Graphical Interface & Command Line
- Local & Remote Data Collection
- Analyze Windows® & Linux® data on OS X®

1 Events vary by processor. 2 No data collection on OS X®
New for 2017! Python, FLOPS, Storage & More...

Intel® VTune™ Amplifier Performance Profiler

Profile Python and Mixed Python / C++ / Fortran
Tune Intel® Xeon Phi™ Knights Landing Processors
Quickly See 3 Keys to HPC Performance
Optimize Memory Access
Storage Analysis – I/O bound or CPU bound?
Enhanced OpenCL™ & GPU Profiling
Easier Remote and Command Line Usage
Add Custom Counters to the Timeline
Preview: Application & Storage Performance Snapshots

Intel® Advisor – optimize vectorization for AVX-512 (with or without hardware)
Intel® VTune™ Amplifier Tunes Knights Landing Processors
4 Critical Optimizations for Intel® Xeon Phi™ Processors

1) High Bandwidth Memory
   - Decide which data structures to place in MCDRAM
   - See performance problems by memory hierarchy
   - Measure DRAM and MCDRAM bandwidth

2) Scalability of MPI and OpenMP
   - Serial vs. Parallel time
   - Imbalance, overhead cost, parallel loop parameters

3) Micro Architecture Efficiency
   - See the efficiency of your code in the core pipeline
   - Zero in on details with custom PMU events

4) Vectorization Efficiency – Use Intel® Advisor
   - Optimize for AVX-512 with or without AVX-512 hardware
Optimize Memory Access
Memory Access Analysis - Intel® VTune™ Amplifier 2017

Tune data structures for performance
- Attribute cache misses to data structures (not just the code causing the miss)
- Support for custom memory allocators

Optimize NUMA latency & scalability
- True & false sharing optimization
- Auto detect max system bandwidth
- Easier tuning of inter-socket bandwidth

Easier install, Latest processors
- No special drivers required on Linux*
- Intel® Xeon Phi™ processor MCDRAM (high bandwidth memory) analysis
Storage Device Analysis (HDD, SATA or NVMe SSD)
Intel® VTune™ Amplifier

Are You I/O Bound or CPU Bound?
- Explore imbalance between I/O operations (async & sync) and compute
- Storage accesses mapped to the source code
- See when CPU is waiting for I/O
- Measure bus bandwidth to storage

Latency analysis
- Tune storage accesses with latency histogram
- Distribution of I/O over multiple devices

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Find & Debug Memory & Threading Errors

Intel® Inspector – Memory & Thread Debugger

Correctness Tools Increase ROI By 12%–21%¹
- Errors found earlier are less expensive to fix
- Several studies, ROI% varies, but earlier is cheaper

Diagnosing Some Errors Can Take Months
- Races & deadlocks not easily reproduced
- Memory errors hard to find without a tool

Debugger Integration Speeds Diagnosis
- Breakpoint set just before the problem
- Examine variables & threads with the debugger

Diagnose in hours instead of months

¹ Cost Factors – Square Project Analysis
CERT: U.S. Computer Emergency Readiness Team, and Carnegie Mellon CyLab
NIST: National Institute of Standards & Technology: Square Project Results

Intel® Inspector dramatically sped up our ability to track down difficult to isolate threading errors before our packages are released to the field.

Peter von Kaenel, Director,
Software Development,
Harmonic Inc.

http://intel.ly/inspector-xe
New for 2017! New Processors, New C++ Language Features
Intel® Inspector 2017 – Memory and Thread Debugger

New C++ Language Features
- Full C++ 11 support including `std::mutex` and `std::atomic`

Easier Identification of Threading Bugs
- Variable name causing error is shown (global, static & stack) in addition to the code lines

Run Native on Intel® Xeon Phi™ Processors
- This simplifies workflow for Intel Xeon Phi processor development
- Tip: Reduce thread count to ≤ 30 for best KNL performance while running Intel Inspector
INTEL® ADVISOR

VECTORIZATION OPTIMIZATION AND THREAD PROTOTYPING
FOR SOFTWARE ARCHITECTS
Faster Code Faster with Data Driven Design

Intel® Advisor – Vectorization Optimization and Thread Prototyping

Faster Vectorization Optimization:
- Vectorize where it will pay off most
- Quickly ID what is blocking vectorization
- Tips for effective vectorization
- Safely force compiler vectorization
- Optimize memory stride

Breakthrough for Threading Design:
- Quickly prototype multiple options
- Project scaling on larger systems
- Find synchronization errors before implementing threading
- Design without disrupting development

Less Effort, Less Risk and More Impact

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Part of Intel® Parallel Studio for Windows® and Linux®
http://intel.ly/advisor-xe
Vectorize & Thread or Performance Dies
Threaded + Vectorized can be much faster than either one alone

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance Configurations at the end of this presentation.
“Automatic” Vectorization Often Not Enough
A good compiler can still benefit greatly from vectorization optimization

Compiler will not always vectorize
- Check for Loop Carried Dependencies using Intel® Advisor
- All clear? Force vectorization. C++ use: pragma simd, Fortran use: SIMD directive

Not all vectorization is efficient vectorization
- Stride of 1 is more cache efficient than stride of 2 and greater. Analyze with Intel® Advisor.
- Consider data layout changes Intel® SIMD Data Layout Templates can help

The benchmarks on the previous slides did not all “auto vectorize”. Compiler directives were used to force vectorization and get more performance.

Arrays of structures are great for intuitively organizing data, but are much less efficient than structures of arrays. Use the Intel® SIMD Data Layout Templates (Intel® SDLT) to map data into a more efficient layout for vectorization.
New for 2017! AVX-512, FLOPS, & More...

Intel® Advisor – Vectorization Optimization

Next Gen Intel® Xeon Phi™ Support
Tune for AVX-512 with or without AVX-512 hardware
Precise FLOPS calculation

Enhanced Memory Access Pattern Analysis
Easier Selection of High Impact Loops
Batch Mode Workflow Saves Time
Fast Answers with Loop Analytics
CLUSTER TOOLS

Intel® MPI Library

Intel® Trace Analyzer and Collector
Intel® MPI Library Overview

Optimized MPI application performance
- Application-specific tuning
- Automatic tuning
- New! - Support for Intel® Xeon Phi™ Processor (code named Knights Landing)
- New! – Support for Intel® Omni-Path Architecture Fabric

Lower latency and multi-vendor interoperability
- Industry leading latency
- Performance optimized support for the fabric capabilities through OpenFabrics*(OFI)

Faster MPI communication
- Optimized collectives

Sustainable scalability up to 340K cores
- Native InfiniBand* interface support allows for lower latencies, higher bandwidth, and reduced memory requirements

More robust MPI applications
- Seamless interoperability with Intel® Trace Analyzer and Collector
What’s New: Intel® MPI Library 2017

• Ready for Intel® Xeon Phi™ Processors (code named Knights Landing (KNL))
• Ready for Intel® Omni-Path Architecture fabric
• Usage of specially optimized memcpy for KNL
• Tuning of shared memory collectives on single KNL nodes
• General optimization of RMA
• General optimization and speed up startup time and MPI tune utility
Intel® Trace Analyzer and Collector Overview

Intel® Trace Analyzer and Collector helps the developer:

- Visualize and understand parallel application behavior
- Evaluate profiling statistics and load balancing
- Identify communication hotspots

Features

- Event-based approach
- Low overhead
- Excellent scalability
- Powerful aggregation and filtering functions
- Idealizer

Automatically detect performance issues and their impact on runtime
MPI Performance Snapshot
Scalable profiling for MPI and Hybrid

**Lightweight** – Low overhead profiling for 100K+ Ranks

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

**Identifying Key Metrics** – Shows MPI/OpenMP imbalances
What’s New: Intel® Trace Analyzer and Collector

- Intel Trace Analyzer and Collector will be ready for KNL
- Improved scalability of imbalance profiler by up to 10x
- Improved MPI Snapshot feature HTML output
Additional Material

- Product page

- Training materials

- Evaluation guides
### Platform Hardware and Software Configuration

<table>
<thead>
<tr>
<th>Platform</th>
<th>Unscaled Core Frequency</th>
<th>Cores/Socket</th>
<th>Num Sockets</th>
<th>L1 Data Cache</th>
<th>L2 Cache</th>
<th>L3 Cache</th>
<th>Memory</th>
<th>Memory Frequency</th>
<th>Memory Access</th>
<th>H/W Prefetchers Enabled</th>
<th>HT Enabled</th>
<th>Turbo Enabled</th>
<th>C States</th>
<th>O/S Name</th>
<th>Operating System</th>
<th>Compiler Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Xeon™ 5472 Processor</td>
<td>3.0 GHZ</td>
<td>4</td>
<td>2</td>
<td>32K</td>
<td>6 MB</td>
<td>None</td>
<td>32 GB</td>
<td>800 MHz</td>
<td>UMA</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Disabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ X5570 Processor</td>
<td>2.9 GHZ</td>
<td>4</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>8 MB</td>
<td>48 GB</td>
<td>1333 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ X5680 Processor</td>
<td>3.33 GHZ</td>
<td>6</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>12 MB</td>
<td>48 MB</td>
<td>1333 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ E5-2690 Processor</td>
<td>2.9 GHZ</td>
<td>8</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>20 MB</td>
<td>64 GB</td>
<td>1600 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ E5-2697v2 Processor</td>
<td>2.7 GHZ</td>
<td>12</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>30 MB</td>
<td>64 GB</td>
<td>1867 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>RHEL 7.1</td>
<td>3.10.0-229.el7.x86_64</td>
<td>icc version 14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ E5-2600v3 Processor</td>
<td>2.2 GHZ</td>
<td>18</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>46 MB</td>
<td>128 GB</td>
<td>2133 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>Fedora 20</td>
<td>3.13.5-202.fc20</td>
<td>14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ E5-2600v4 Processor</td>
<td>2.3 GHZ</td>
<td>18</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>46 MB</td>
<td>256 GB</td>
<td>2400 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>RHEL 7.0</td>
<td>3.10.0-123.el7.x86_64</td>
<td>icc version 14.0.1</td>
</tr>
<tr>
<td>Intel® Xeon™ E5-2600v4 Processor</td>
<td>2.2 GHZ</td>
<td>22</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>56 MB</td>
<td>128 GB</td>
<td>2133 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Enabled</td>
<td>CentOS 7.2</td>
<td>3.10.0-327.el7.x86_64</td>
<td>icc version 14.0.1</td>
</tr>
</tbody>
</table>

**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

Performance measured in Intel Labs by Intel employees.
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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