INTEL® ARCHITECTURE AND TOOLS
JURECA - TUNING FOR THE PLATFORM II

Klaus-Dieter Oertel
Intel-SSG-Developer Products Division
FZ Jülich, 22-05-2017
ARCHITECTURE BASICS
The “Free Lunch” is over, really
Processor clock rate growth halted around 2005

Software must be parallelized to realize all the potential performance
# Changing Hardware Impacts Software

More cores → More Threads → Wider vectors

<table>
<thead>
<tr>
<th></th>
<th>Intel® Xeon® Processor 64-bit</th>
<th>Intel® Xeon® Processor 5100 series</th>
<th>Intel® Xeon® Processor 5500 series</th>
<th>Intel® Xeon® Processor 5600 series</th>
<th>Intel® Xeon® Processor E5-2600 v2 series</th>
<th>Intel® Xeon® Processor E5-2600 v3 series v4 series</th>
<th>Future Intel® Xeon® Processor¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to Core(s)</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>18-22</td>
<td>TBD</td>
</tr>
<tr>
<td>Up to Threads</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>24</td>
<td>36-44</td>
<td>TBD</td>
</tr>
<tr>
<td>SIMD Width</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Vector ISA</td>
<td>Intel® SSE3</td>
<td>Intel® SSE3</td>
<td>Intel® SSE4- 4.1</td>
<td>Intel® SSE 4.2</td>
<td>Intel® AVX</td>
<td>Intel® AVX2</td>
<td>Intel® AVX-512</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Intel® Xeon Phi™ x100 Coprocessor (KNC)</th>
<th>Intel® Xeon Phi™ x200 Processor &amp; Coprocessor (KNL)</th>
<th>Future Intel® Xeon Phi™ (KNH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to Core(s)</td>
<td>61</td>
<td>72</td>
<td>TBD</td>
</tr>
<tr>
<td>Up to Threads</td>
<td>244</td>
<td>288</td>
<td>TBD</td>
</tr>
<tr>
<td>SIMD Width</td>
<td>512</td>
<td>512</td>
<td>TBD</td>
</tr>
<tr>
<td>Vector ISA</td>
<td>IMCI 512</td>
<td>Intel® AVX-512</td>
<td>TBD</td>
</tr>
</tbody>
</table>

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

Product specification for launched and shipped products available on ark.intel.com.

1. Not launched or in planning.
## Changing Hardware Impacts Software

More cores → More Threads → Wider vectors

<table>
<thead>
<tr>
<th>Product</th>
<th>Up to Core(s)</th>
<th>Up to Threads</th>
<th>SIMD Width</th>
<th>Vector ISA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Xeon® Processor 64-bit</td>
<td>1</td>
<td>2</td>
<td>128</td>
<td>SSE3</td>
</tr>
<tr>
<td>Intel® Xeon® Processor 5100 series</td>
<td>1</td>
<td>2</td>
<td>128</td>
<td>SSE3</td>
</tr>
<tr>
<td>Intel® Xeon® Processor 5500 series</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>SSE4- 4.1</td>
</tr>
<tr>
<td>Intel® Xeon® Processor 5600 series</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>SSE4- 4.1</td>
</tr>
<tr>
<td>Intel® Xeon® Processor E5-2600 v2 series</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>SSE4- 4.1</td>
</tr>
<tr>
<td>Intel® Xeon® Processor E5-2600 v3 series</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>SSE4- 4.1</td>
</tr>
<tr>
<td>Intel® Xeon® Processor E5-2600 v4 series</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>SSE4- 4.1</td>
</tr>
<tr>
<td>Future Intel® Xeon® Processor</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>

High performance software must be both:
- Parallel (multi-thread, multi-process)
- Vectorized
Untapped Potential Can Be Huge!

Threaded + Vectorized can be much faster than either one alone

The Difference Is Growing With Each New Generation of Hardware

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.
Performance Scaling from the Core, to Multicore, to Many Core and Beyond – to Cluster

Extracting performance from the computing resources

- Core: **vectorization**, prefetching, cache utilization
- Multi-Many core (processor/socket) level **parallelization**
- Multi-socket (node) level **parallelization**
- Clusters **scaling**
Moore’s “Law”

“The number of transistors on a chip will **double** approximately **every two years.**”

[Gordon Moore]
Desktop, Mobile & Server
Tick/Tock Model

Intel® Core™

Nehalem (2008)
New Microarchitecture 45nm

Westmere (2010)
New Process Technology 32nm

Sandy Bridge (2011)
New Microarchitecture 32nm

Ivy Bridge (2012)
New Process Technology 22nm

Haswell (2013)
New Microarchitecture 22nm

Future

Broadwell (2014)
New Process Technology 14nm

Skylake
New Microarchitecture 14nm

New Process Technology 11nm

New Microarchitecture 11nm

Tick
Tock
Tick
Tock
Tick
Tock
Desktop, Mobile & Server
Your Source for Intel® Product Information

**Naming schemes:**
- **Desktop & Mobile:**
  - Intel® Core™ i3/i5/i7 processor family
  - 4 generations, e.g.:
    - 4th Generation Intel® Core™ i7-XXXX

- **Server:**
  - Intel® Xeon® E3/E5/E7 processor family
  - 3 generations, e.g.:
    - Intel® Xeon® Processor E3-XXXX v3

Information about available Intel products can be found here: [http://ark.intel.com/](http://ark.intel.com/)
# Haswell Processor at JURECA: E5-2680v3

See [ark.intel.com](http://ark.intel.com) for more Details

<table>
<thead>
<tr>
<th># Cores</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-AVX Reference Frequency</td>
<td>2500 MHz</td>
</tr>
<tr>
<td>Non-AVX Max Turbo Frequency</td>
<td>3300 MHz</td>
</tr>
<tr>
<td>AVX Reference Frequency</td>
<td>2100 MHz</td>
</tr>
<tr>
<td>AVX Max Turbo Frequency</td>
<td>3100 MHz</td>
</tr>
<tr>
<td>L3 Cache Size</td>
<td>30 MB</td>
</tr>
<tr>
<td>QPI</td>
<td>9.6 GT/s</td>
</tr>
</tbody>
</table>

E5-2680v3: Turbo bins in GHz for number of cores being used (see [here](http://ark.intel.com) for more)

<table>
<thead>
<tr>
<th>Cores</th>
<th>1-2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-AVX</td>
<td>3.3</td>
<td>3.1</td>
<td>3</td>
<td>2.9</td>
<td>2.9</td>
<td>2.9</td>
<td>2.9</td>
<td>2.9</td>
<td>2.9</td>
<td>2.9</td>
</tr>
<tr>
<td>AVX</td>
<td>3.1</td>
<td>2.9</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
<td>2.8</td>
</tr>
</tbody>
</table>
## Desktop, Mobile & Server Performance

### Following Moore's Law:

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Instruction Set</th>
<th>SP FLOPs per Cycle per Core</th>
<th>DP FLOPs per Cycle per Core</th>
<th>L1 Cache Bandwidth (bytes/cycle)</th>
<th>L2 Cache Bandwidth (bytes/cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nehalem</td>
<td>SSE (128-bits)</td>
<td>8</td>
<td>4</td>
<td>32 (16B read + 16B write)</td>
<td>32</td>
</tr>
<tr>
<td>Sandy Bridge</td>
<td>Intel® AVX (256-bits)</td>
<td>16</td>
<td>8</td>
<td>48 (32B read + 16B write)</td>
<td>32</td>
</tr>
<tr>
<td>Haswell</td>
<td>Intel® AVX2 (256-bits)</td>
<td>32</td>
<td>16</td>
<td>96 (64B read + 32B write)</td>
<td>64</td>
</tr>
</tbody>
</table>

### Example of **theoretic peak** FLOP rates:

- Intel® Core™ i7-2710QE (Sandy Bridge): 2.1 GHz * 16 SP FLOPs * 4 cores = **134.4 SP GFLOPs**

- Intel® Core™ i7-4765T (Haswell): 2.0 GHz * 32 SP FLOPs * 4 cores = **256 SP GFLOPs**
Desktop, Mobile & Server
Caches

Cache hierarchy:

<table>
<thead>
<tr>
<th>Level</th>
<th>Latency (cycles)</th>
<th>Bandwidth (per core per cycle)</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1-D</td>
<td>4</td>
<td>2x 16 bytes</td>
<td>32KiB</td>
</tr>
<tr>
<td>L2 (unified)</td>
<td>12</td>
<td>1x 32 bytes</td>
<td>256KiB</td>
</tr>
<tr>
<td>L3 (LLC)</td>
<td>26-31</td>
<td>1x 32 bytes</td>
<td>varies (≥ 2MiB per core)</td>
</tr>
<tr>
<td>L2 and L1 D-Cache in other cores</td>
<td>43 (clean hit), 60 (dirty hit)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Processor Architecture Basics

Core vs. Uncore

• **Core:**
  Processor core's logic:
  - Execution units
  - Core caches (L1/L2)
  - Buffers & registers
  - ...

• **Uncore:**
  All outside a processor core:
  - Memory controller/channels (MC) and Intel® QuickPath Interconnect (QPI)
  - L3 cache shared by all cores
  - Type of memory
  - Power management and clocking
  - Optionally: Integrated graphics

⇒ Only uncore is differentiation within same processor family!
Processor Architecture Basics
NUMA - Memory, Bandwidth & Latency

Memory allocation:
• Differentiate: implicit vs. explicit memory allocation
• Explicit allocation with NUMA aware libraries, e.g. libnuma (Linux*) or tbbmalloc
• Bind memory ⇔ (SW) thread, and (SW) thread ⇔ processor
• More information on optimizing for performance: https://software.intel.com/de-de/articles/optimizing-applications-for-numa

Performance:
• Remote memory access latency ~1.7x greater than local memory
• Local memory bandwidth can be up to ~2x greater than remote
Processor Architecture Basics
NUMA - Thread Affinity & Enumeration

Non-NUMA:
Thread affinity might be beneficial (e.g. cache locality) but not required

NUMA:
Thread affinity is required:
• Improve accesses to local memory vs. remote memory
• Ensure 3rd party components support affinity mapping, e.g.:
  ▪ Intel® OpenMP* via $KMP_AFFINITY or $OMP_PLACES
  ▪ Intel® MPI via $I_MPI_PIN_DOMAIN (default may be OK)
    tool cpuinfo provides additional information
  ▪ ...

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.
Documentation

Intel® 64 and IA-32 Architectures Software Developer Manuals: https://software.intel.com/en-us/articles/intel-sdm

- Intel® 64 and IA-32 Architectures Software Developer's Manuals
  - Volume 1: Basic Architecture
  - Volume 2: Instruction Set Reference
  - Volume 3: System Programming Guide
- Software Optimization Reference Manual
- Related Specifications, Application Notes, and White Papers

Intel® Processor Numbers (how type names are encoded): http://www.intel.com/products/products/processor_number
Intel® Parallel Studio XE

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

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

**Intel® C/C++ and Fortran Compilers**

**Intel® MPI Library**
- Intel® Integrated Performance Primitives
  Image, Signal, and Compression Routines
- Intel® Threading Building Blocks
  Task-Based Parallel C++ Template Library

**Intel® Advisor**
Vectorization Optimization and Thread Prototyping

**Cluster Tools**
- Intel® Cluster Checker
  Cluster Diagnostic Expert System
- Intel® Trace Analyzer and Collector
  MPI Profiler
- Intel® Inspector
  Memory and Threading Checking
- Intel® Trace Analyzer and Collector
  Memory and Threading Checking
- Intel® Advisor
  Performance Profiler
- Intel® Inspector
  Performance Profiler

**Intel® Distribution for Python**
Performance Scripting
WHICH TOOL SHOULD I USE?
Tools for High-Performance Implementation
Intel® Parallel Studio XE

- Intel® MPI Library
- Intel® MPI Benchmarks
- Intel® Compiler
- Intel® Math Kernel Library
- Intel® IPP – Media & Data Library
- Intel® Data Analytics Library
- Intel® Cilk™ Plus
- Intel® OpenMP*
- Intel® TBB – Threading Library

Flowchart:
- Cluster Scalable?
  - Y: Thread
  - N: Tune MPI
- Effective threading?
  - Y: Vectorize
  - N: Tune MPI
- Memory Bandwidth Sensitive?
  - Y: Optimize Bandwidth
  - N: Vectorize

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

Intel® Parallel Studio XE

Cluster Scalable?

Tune MPI

Intel® Trace Analyzer & Collector (ITAC)
Intel® MPI Snapshot
Intel® MPI Tuner

Effective threading?

Vectorize

Optimize Bandwidth

Memory Bandwidth Sensitive?

Thread

Optimize

Bandwidth

Intel® VTune™ Amplifier

Intel® Advisor

Intel® VTune™ Amplifier

Performance Analysis Tools for Diagnosis

Intel® Parallel Studio XE

Cluster Scalable?

Tune MPI

Intel® Trace Analyzer & Collector (ITAC)
Intel® MPI Snapshot
Intel® MPI Tuner

Effective threading?

Vectorize

Optimize Bandwidth

Memory Bandwidth Sensitive?

Thread

Optimize

Bandwidth

Intel® VTune™ Amplifier

Intel® Advisor

Intel® VTune™ Amplifier

Performance Analysis Tools for Diagnosis

Intel® Parallel Studio XE

Cluster Scalable?

Tune MPI

Intel® Trace Analyzer & Collector (ITAC)
Intel® MPI Snapshot
Intel® MPI Tuner

Effective threading?

Vectorize

Optimize Bandwidth

Memory Bandwidth Sensitive?

Thread

Optimize

Bandwidth

Intel® VTune™ Amplifier

Intel® Advisor

Intel® VTune™ Amplifier

Performance Analysis Tools for Diagnosis

Intel® Parallel Studio XE

Cluster Scalable?

Tune MPI

Intel® Trace Analyzer & Collector (ITAC)
Intel® MPI Snapshot
Intel® MPI Tuner

Effective threading?

Vectorize

Optimize Bandwidth

Memory Bandwidth Sensitive?

Thread

Optimize

Bandwidth

Intel® VTune™ Amplifier

Intel® Advisor

Intel® VTune™ Amplifier

Performance Analysis Tools for Diagnosis

Intel® Parallel Studio XE

Cluster Scalable?

Tune MPI

Intel® Trace Analyzer & Collector (ITAC)
Intel® MPI Snapshot
Intel® MPI Tuner

Effective threading?

Vectorize

Optimize Bandwidth

Memory Bandwidth Sensitive?

Thread

Optimize

Bandwidth

Intel® VTune™ Amplifier

Intel® Advisor

Intel® VTune™ Amplifier

Performance Analysis Tools for Diagnosis

Intel® Parallel Studio XE

Cluster Scalable?

Tune MPI

Intel® Trace Analyzer & Collector (ITAC)
Intel® MPI Snapshot
Intel® MPI Tuner

Effective threading?

Vectorize

Optimize Bandwidth

Memory Bandwidth Sensitive?

Thread

Optimize

Bandwidth

Intel® VTune™ Amplifier

Intel® Advisor

Intel® VTune™ Amplifier
LIBRARIES

Intel® Math Kernel Library
Intel® Data Analytics Acceleration Library
Intel® Integrated Performance Primitives
INTEL® MATH KERNEL LIBRARY (MKL)
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>
</tr>
</thead>
<tbody>
<tr>
<td>• BLAS</td>
<td>• Multidimensional</td>
<td>• Trigonometric</td>
<td>• Kurtosis</td>
<td>• Splines</td>
</tr>
<tr>
<td>• LAPACK</td>
<td>• FFTW interfaces</td>
<td>• Hyperbolic</td>
<td>• Variation</td>
<td>• Interpolation</td>
</tr>
<tr>
<td>• ScaLAPACK</td>
<td>• Cluster FFT</td>
<td>• Exponential</td>
<td>• coefficient</td>
<td>• Trust Region</td>
</tr>
<tr>
<td>• Sparse BLAS</td>
<td></td>
<td>• Log</td>
<td>• Order</td>
<td>• Fast Poisson</td>
</tr>
<tr>
<td>• Sparse Solvers</td>
<td></td>
<td>• Power</td>
<td>statistics</td>
<td>Solver</td>
</tr>
<tr>
<td>• Iterative</td>
<td></td>
<td>• Root</td>
<td>• Min/max</td>
<td></td>
</tr>
<tr>
<td>• PARDISO*</td>
<td></td>
<td>• Vector RNGs</td>
<td>• Variance-</td>
<td></td>
</tr>
<tr>
<td>• Cluster Sparse Solver</td>
<td></td>
<td></td>
<td>covariance</td>
<td></td>
</tr>
</tbody>
</table>

And More…

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver
## Intel® Math Kernel Library is Parallelized

<table>
<thead>
<tr>
<th>Domain</th>
<th>SIMD</th>
<th>OpenMP</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAS 1, 2, 3</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FFTs</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LAPACK (dense LA solvers)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(relies on BLAS 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScALAPACK (cluster dense LA solvers)</td>
<td>X</td>
<td>(hybrid)</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARDISO (sparse solver)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>(relies on BLAS 3)</td>
<td></td>
<td>(Cluster PARDISO)</td>
</tr>
<tr>
<td>VML / VSL</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cluster FFT</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
What’s New: Intel® MKL 2017

• 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® MKL: Performance Benefit to Applications

DGEMMM Performance
On Intel® Xeon® Processor E5-2699 v4

The latest version of Intel® MKL unleashes the performance benefits of Intel architectures.
INTEL® DATA ANALYTICS ACCELERATION LIBRARY (DAAL)
Intel® DAAL Overview

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

Pre-processing
(De-)Compression

Transformation
PCA
Statistical moments
Variance matrix
QR, SVD, Cholesky
Apriori

Analysis
Linear regression
Naïve Bayes
SVM
Classifier boosting
Kmeans
EM GMM

Modeling
Collaborative filtering
Neural Networks

Validation

Decision Making
Example Performance: Intel® DAAL vs. Spark* MLLib

Intel® DAAL vs. Spark* Mlib

K-means Performance Comparison on Eight-node Cluster

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® INTEGRATED PERFORMANCE PRIMITIVES (IPP)
## Intel® IPP Domain Applications

<table>
<thead>
<tr>
<th>Image Processing</th>
<th>Signal Processing</th>
<th>Data Compression and Cryptography</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Medical imaging</td>
<td>• Games (sophisticated audio content or effects)</td>
<td>• Data centers</td>
</tr>
<tr>
<td>• Computer vision</td>
<td>• Echo cancellation</td>
<td>• Enterprise data managements</td>
</tr>
<tr>
<td>• Digital surveillance</td>
<td>• Telecommunications</td>
<td>• ID verification</td>
</tr>
<tr>
<td>• Biometric identification</td>
<td></td>
<td>• Smart cards/wallets</td>
</tr>
<tr>
<td>• Automated sorting</td>
<td>• Energy</td>
<td>• Electronic signature</td>
</tr>
<tr>
<td>• ADAS</td>
<td></td>
<td>• Information security/cybersecurity</td>
</tr>
<tr>
<td>• Visual search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Other names and brands may be claimed as the property of others.*
Intel® IPP Data Compression and Decompression Performance Boost vs. ZLIB Library

<table>
<thead>
<tr>
<th>Speedup</th>
<th>Compression (Level=1)</th>
<th>Compression (Level=6)</th>
<th>Compression (Level=9)</th>
<th>Compression (Level=IPP Fastest*)</th>
<th>Decompression</th>
</tr>
</thead>
<tbody>
<tr>
<td>zlib 1.2.8</td>
<td>1</td>
<td>1.3x</td>
<td>2.1x</td>
<td>2.7x</td>
<td>3.3x</td>
</tr>
<tr>
<td>Intel® IPP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.9x</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.
INTEL® DISTRIBUTION FOR PYTHON®
Python* Landscape

Challenge#1:
Domain specialists are not professional software programmers.

Challenge#2:
Python performance limits migration to production systems

Adoption of Python continues to grow among domain specialists and developers for its productivity benefits
Challenge#1: Domain specialists are not professional programmers.

Challenge#2: Python performance limits migration to production systems.

Intel’s solution is to...

- Accelerate Python performance
- Enable easy access
- Empower the community

Adoption of Python continues to grow among domain specialists and developers for its productivity benefits.
Access Multiple Options for Faster Python*
Included in Intel® Distribution for Python

Accelerate with native libraries
- NumPy, SciPy, Scikit-Learn, Theano, Pandas, pyDAAL
- Intel® MKL, Intel® DAAL

Exploit vectorization and threading
- Cython + Intel C++ compiler
- Numba + Intel LLVM

Better/Composable threading
- Cython, Numba, Pyston
- Threading composability for MKL, CPython, Blaze/Dask, Numba

Multi-node parallelism
- Mpi4Py, Distarray
- Intel native libraries: Intel MPI

Integration with Big Data, ML platforms and frameworks
- Spark, Hadoop, Trusted Analytics Platform

Better performance profiling
- Extensions for profiling mixed Python & native/JIT codes

“I expected Intel's numpy to be fast but it is significant that plain old python code is much faster with the Intel version too.”

Dr. Donald Kinghorn, Puget Systems Review
A Two-Pronged Approach for Faster Python Performance

Accelerated Python Distribution plus Performance Profiling

Step 1: Use Intel® Distribution for Python*

- Leverage optimized native libraries for performance.
- Drop-in replacement for your current Python*.
- Latest optimizations for Intel® processors and compilers.

Step 2: Use Intel® VTune™ Amplifier for Profiling

- Get detailed summary of entire application execution profile.
- Auto-detects and profiles Python/C/C++ mixed code and extensions.
- Accurately detect hotspots. Line-level analysis helps make smart optimization decisions fast.
- Available in Intel® Parallel Studio XE 2017 suite.
INTEL® THREADING BUILDING BLOCKS (TBB)
Intel® Threading Building Blocks (Intel® TBB)

What
- Widely used C++ template library for task parallelism.
- Features
  - Parallel algorithms and data structures.
  - Threads and synchronization primitives.
  - Scalable memory allocation and task scheduling.

Benefit
- Rich feature set for general purpose parallelism.
- Available as an open source and a commercial license.
- Supports C++, Windows*, Linux*, OS X*, other OS's.
- Commercial support for Intel® Atom™, Core™, Xeon® processors, and for Intel® Xeon Phi™ coprocessors.

Also available as open source at threadingbuildingblocks.org

https://software.intel.com/intel-tbb

Simplify Parallelism with a Scalable Parallel Model
Task Execution

Intel TBB runtime dynamically maps tasks to threads

Automatic load balance, not fair, lock-free whenever possible
## 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>Timers and Exceptions</th>
<th>Threads</th>
<th>Thread Local Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sophisticated work scheduling engine that empowers parallel algorithms and the flow graph</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>

### Memory Allocation

Scalable memory manager and false-sharing free allocators

### Synchronization Primitives

Atomic operations, a variety of mutexes with different properties, condition variables

### Parallel algorithms and data structures

- Threads and synchronization
- Memory allocation and task scheduling
### Features and Functions List

#### Generic Parallel Algorithms
- parallel_for
- parallel_reduce
- parallel_for_each
- parallel_do
- parallel_invoke
- parallel_sort
- parallel_deterministic_reduce
- parallel_scan
- parallel_pipeline
- pipeline

#### Flow Graph
- graph
- continue_node
- source_node
- function_node
- multifunction_node
- overwrite_node
- write_once_node
- limiter_node
- buffer_node
- queue_node
- priority_queue_node
- sequencer_node
- broadcast_node
- join_node
- split_node
- indexer_node

#### Concurrent Containers
- concurrent_unordered_map
- concurrent_unordered_multimap
- concurrent_unordered_set
- concurrent_unordered_multiset
- concurrent_hash_map
- concurrent_queue
- concurrent_bounded_queue
- concurrent_priority_queue
- concurrent_vector
- concurrent_lru_cache

#### Synchronization Primitives
- atomic
- mutex
- recursive_mutex
- spin_mutex
- spin_rw_mutex
- speculative_spin_mutex
- speculative_spin_rw_mutex
- queuing_mutex
- queuing_rw_mutex
- null_mutex
- null_rw_mutex
- reader_writer_lock
- critical_section
- condition_variable
- aggregator (preview)

#### Task Scheduler
- task
- task_group
- structured_task_group
- task_group_context
- task_scheduler_init
- task_scheduler_observer
- task_arena

#### Exceptions
- tbb_exception
- captured_exception
- movable_exception

#### Memory Allocation
- tbb_allocator
- scalable_allocator
- cache_aligned_allocator
- zero_allocator
- aligned_space
- memory_pool (preview)

### Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Intel® Compilers for Intel® Parallel Studio XE 2017
Intel® C++ 17.0 and Intel® Fortran 17.0

Common Updates

- Enhanced support for the latest Intel® AVX2 and AVX512 instruction sets for the latest Intel® processors (including Intel® Xeon Phi™ processor)
- Enhanced optimization/vectorization reports make Intel compilers indispensable towards modernizing your code
- Support for OpenMP* 4.5, offering improved vectorization control, and new SIMD instructions

Intel® C++ Compiler
- SIMD Data Layout Template to facilitate vectorization for your C++ code
- Virtual function vectorization capability
- Full support for the latest C11 and C++14 standards; initial C++17 support

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)
## Common Optimization Options

<table>
<thead>
<tr>
<th>Optimization Option</th>
<th>Windows*</th>
<th>Linux*, OS* X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable optimization</td>
<td>/Od</td>
<td>-O0</td>
</tr>
<tr>
<td>Optimize for speed (no code size increase)</td>
<td>/O1</td>
<td>-O1</td>
</tr>
<tr>
<td>Optimize for speed (default, auto-vect enabled)</td>
<td>/O2</td>
<td>-O2</td>
</tr>
<tr>
<td>High-level loop optimization</td>
<td>/O3</td>
<td>-O3</td>
</tr>
<tr>
<td>Create symbols for debugging</td>
<td>/Zi</td>
<td>-g</td>
</tr>
<tr>
<td>Multi-file inter-procedural optimization</td>
<td>/Qipo</td>
<td>-ipo</td>
</tr>
<tr>
<td>Profile guided optimization (multi-step build)</td>
<td>/Qprof-gen</td>
<td>-prof-gen</td>
</tr>
<tr>
<td></td>
<td>/Qprof-use</td>
<td>-prof-use</td>
</tr>
<tr>
<td>Optimize for speed across the entire program (&quot;prototype switch&quot;)</td>
<td>/fast</td>
<td>-fast</td>
</tr>
<tr>
<td><strong>fast options definitions changes over time!</strong></td>
<td>same as: /O3 /Qipo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/Qprec-div=2 /QxHost)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-fast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>same as: Linux: -ipo -O3 -no-prec-div -static -fp-model fast=2 -xHost</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS X: -ipo -mdynamic-no-pic -O3 -no-prec-div -fp-model fast=2 -xHost</td>
<td></td>
</tr>
<tr>
<td>OpenMP support</td>
<td>/Qopenmp</td>
<td>-qopenmp</td>
</tr>
<tr>
<td>Automatic parallelization</td>
<td>/Qparallel</td>
<td>-parallel</td>
</tr>
</tbody>
</table>
AVX Vector Types (256 Bits)

Intel® AVX
- 8x single precision FP
- 4x double precision FP

Intel® AVX2
- 32x 8 bit integer
- 16x 16 bit integer
- 8x 32 bit integer
- 4x 64 bit integer
- plain 256 bit
Many Ways to Vectorize

- **Compiler:**
  - Auto-vectorization (no change of code)
  - Auto-vectorization hints (`#pragma vector, ...`)
  - Intel® Cilk™ Plus Array Notation Extensions

- **SIMD intrinsic C++ class**
  (e.g.: `F32vec`, `F64vec`, ...)

- **Vector intrinsic**
  (e.g.: `_mm_fmadd_pd(...), _mm_add_ps(...), ...`)

- **Assembler code**
  (e.g.: `[v]addps, [v]addss, ...`)

---

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.*
Basic Vectorization Switches I

• Linux*, OS X*: \(-x<\text{feature}>\), Windows*: /Qx<\text{feature}>
  ▪ Might enable Intel processor specific optimizations
  ▪ Processor-check added to “main” routine:
    Application errors in case SIMD feature missing or non-Intel processor with
    appropriate/informative message
  ▪ Example: \(-x\text{CORE-AVX2}\)

• Linux*, OS X*: \(-ax<\text{features}>\), Windows*: /Qax<\text{features}>
  ▪ Multiple code paths: baseline and optimized/processor-specific
  ▪ Multiple SIMD features/paths possible, e.g.: \(-ax\text{SSE2,AVX}\)
  ▪ Baseline code path defaults to \(-\text{msse2} (/\text{arch:}\text{sse2})\)
  ▪ The baseline code path can be modified by \(-m<\text{feature}>\) or \(-x<\text{feature}>\)
    (/\text{arch:<feature>} or /Qx<\text{feature}>)
Basic Vectorization Switches II

• Special switch for Linux*, OS X*: `-xHost`, Windows*: `/QxHost`
  
  ▪ Compiler checks SIMD features of current host processor (where built on) and makes use of latest SIMD feature available
  
  ▪ Code only executes on processors with same SIMD feature or later as on build host
  
  ▪ As for `-x<feature>` or `/Qx<feature>`, if “main” routine is built with `-xHost` or `/QxHost` the final executable only runs on Intel processors
Control Vectorization I

• Disable vectorization:
  ▪ Globally via switch:
    Linux*, OS X*: `-no-vec` and `-qno-openmp-simd`, Windows*: `/Qvec-` and `/Qopenmp-simd-`
  ▪ For a single loop:
    C/C++: `#pragma novector`, Fortran: `!DIR$ NOVECTOR`
  ▪ Compiler still can use some SIMD features

• Using vectorization:
  ▪ Globally via switch (default for optimization level 2 and higher):
    Linux*, OS X*: `-vec` and `-qopenmp-simd`, Windows*: `/Qvec` and `/Qopenmp-simd`
  ▪ Enforce for a single loop (override compiler efficiency heuristic) if semantically correct:
    C/C++: `#pragma vector always`, Fortran: `!DIR$ VECTOR ALWAYS`
  ▪ Influence efficiency heuristics threshold:
    Linux*, OS X*: `-vec-threshold[n]`
    Windows*: `/Qvec-threshold[:n]`
    n: 100 (default; only if profitable) ... 0 (always)
Control Vectorization II

• Verify vectorization:
  ▪ Globally:
  ▪ Abort compilation if loop cannot be vectorized:
    C/C++: `#pragma vector always assert`
    Fortran: `!DIR$ VECTOR ALWAYS ASSERT`

• Advanced:
  ▪ *Ignore* vector dependencies (IVDEP):
    C/C++: `#pragma ivdep`
    Fortran: `!DIR$ IVDEP`
  ▪ “Enforce” vectorization using the OpenMP standard
    C/C++: `#pragma omp simd`
    Fortran: `!OMP$ SIMD`
How to detect Vectorization?

- We need feedback! Compiling with a flag below reveals a short report showing which loops are vectorized and the reason for refusing vectorisation for certain loops:
  - **-opt-report=<n>**
    - n = 0-5 increases the level of detail, 2 is the default
      - **-opt-report-phase=vec**
        - shows just vectorization
  
- Profile the code, compile without vectorization (-no-vec) and profile again. Look for code sections with different timing

- Look into assembly. Can be done by using Intel® VTune™ Amplifier XE
## Impressive Performance Improvement

Intel C++ Explicit Vectorization using OpenMP* SIMD

### SIMD Speedup on Intel® Xeon® Processor

Normalized performance data – higher is better

<table>
<thead>
<tr>
<th>Test</th>
<th>Serial</th>
<th>SSE4.2</th>
<th>Core-AVX2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AoBench</td>
<td>1.00</td>
<td>4.27</td>
<td>4.14</td>
</tr>
<tr>
<td>Collision Detection</td>
<td>1.00</td>
<td>2.27</td>
<td>4.15</td>
</tr>
<tr>
<td>Grassshader</td>
<td>1.00</td>
<td>2.26</td>
<td>4.15</td>
</tr>
<tr>
<td>Mandelbrot</td>
<td>1.00</td>
<td>2.43</td>
<td>4.83</td>
</tr>
<tr>
<td>Libor</td>
<td>1.00</td>
<td>3.51</td>
<td>6.61</td>
</tr>
<tr>
<td>RTM-stencil</td>
<td>1.00</td>
<td>3.91</td>
<td>6.06</td>
</tr>
<tr>
<td>Geomean</td>
<td>1.00</td>
<td>2.74</td>
<td>4.92</td>
</tr>
</tbody>
</table>

### 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 -QsSSE4.2` or `-O3 -Qopenmp -simd -QxCORE-AVX2`.

For more information go to [http://www.intel.com/performance](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 SY斯mark 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 SOFTWARE ANALYSIS TOOLS

Intel® Advisor XE Vectorization Optimization and Thread Prototyping
Intel® VTune™ Amplifier XE Performance Profiler
Intel® Inspector XE Memory & Thread Debugger
INTEL® ADVISOR XE
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

Part of Intel® Parallel Studio XE for Windows® and Linux®

http://intel.ly/advisor-xe
The Right Data At Your Fingertips
Get all the data you need for high impact vectorization

Filter by which loops are vectorized!

What prevents vectorization?

Focus on hot loops

What vectorization issues do I have?

Which Vector instructions are being used?

How efficient is the code?

Get Fast Code Fast!
4 Steps to Efficient Vectorization
Intel® Advisor XE – Vectorization Advisor

1. Compiler diagnostics + Performance Data + SIMD efficiency information

2. Guidance: detect problem and recommend how to fix it

3. Loop-Carried Dependency Analysis

4. Memory Access Patterns Analysis
4 Steps to Efficient Vectorization

Intel® Advisor XE – Vectorization Advisor

1. Compiler diagnostics + Performance Data + SIMD efficiency information

2. Guidance: detect problem and recommend how to fix it

Ease-of-use (e.g. for batch systems): Let the GUI create the command line, copy-paste to shell [script]
“Cache-aware” Roofline Automation in Intel Advisor
“Cache-aware” Roofline Automation in Intel Advisor

Better optimized – smaller potential (gap)

Big optimization gap

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

Classic Roofline formulated by Williams, Waterman, Patterson, (Berkeley)  
http://www.eecs.berkeley.edu/~waterman/papers/roofline.pdf

“Cache-aware Roofline model: Upgrading the loft” (Ilic, Pratas, Sousa, INESC-ID/IST, Thec Uni of Lisbon)  
http://www.inesc-id.pt/ficheiros/publicacoes/9068.pdf
INTEL® PERFORMANCE SNAPSHOTS
LIGHT WEIGHT ANALYSIS
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
Three Keys to HPC Performance:
Threading, Memory Access, Vectorization – Intel VTune™ Amplifier

Threading: CPU Utilization
- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use hotspot OpenMP region analysis for more detail

Memory Access Efficiency
- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

Vectorization: FPU Utilization
- FLOPS† estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization

† For 3rd, 5th, 6th Generation Intel® Core™ processors and second generation Intel® Xeon Phi™ processor code named Knights Landing.
Intel® VTune™ Amplifier
Faster, Scalable Code—Faster

Get the data you need
- Hotspot (statistical call tree), call counts (statistical)
- Thread profiling – concurrency and locks and 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 and 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 and command line
- Local and remote data collection
- Analyze Windows* and Linux* data on macOS*

1 Events vary by processor. 2 No data collection on macOS*
New for 2017: Python*, FLOPS, Storage, and More...

Intel® VTune™ Amplifier Performance Profiler

- Profile Python* and Mixed Python / C++ / Fortran*
- Tune latest Intel® Xeon Phi™ processors
- Quickly see three keys to HPC performance
- Optimize memory access
- Storage analysis: I/O bound or CPU bound?
- Enhanced OpenCL* and GPU profiling
- Easier remote and command line usage
- Add custom counters to the timeline
- Preview: Application and storage performance snapshots
- Intel® Advisor: Optimize vectorization for Intel® AVX-512 (with or without 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 and scalability
- True and 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

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

Disk Input and Output Histogram
- Sliders set thresholds for I/O Queue Depth
- Slow task with I/O Wait
Command Line Interface

Examples

Display a list of available analysis types and preset configuration levels

```
amplxe-cl -collect-list
```

Run Hot Spot analysis on target *myApp* and store result in default-named directory, such as *r000hs*

```
amplxe-cl -c hotspots -- myApp
```

Run the Cuncurrency analysis, store the result in directory *r001par*

```
amplxe-cl -c concurrency -result-dir r001par
```
Command Line Interface

Examples

Display a list of available analysis types and preset configuration levels

```
amplxe-cl -collect-list
```

Run Hot Spot analysis on target *myApp* and store result in default-named directory, such as *r000hs*

```
amplxe-cl -c hotspots -- myApp
```

Run the Ease-of-use (e.g. for batch systems): Let the GUI create the command line, copy-paste to shell [script]
Find and Debug Memory and Threading Errors

Intel® Inspector: Memory and 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 and deadlocks not easily reproduced.
- Memory errors hard to find without a tool.

Debugger integration speeds diagnosis
- Breakpoint set just before the problem.
- Examine variables and 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.

Part of Intel® Parallel Studio XE Professional Ed. For Windows* and Linux* From $1,599

http://intel.ly/inspector-xe
Intel® Inspector XE
Threading Correctness

- Inspector XE simulates all possible orders of data access
- Not only errors in this run. Inspector XE finds all possible errors even if they don’t happen in this run!
- Typical race conditions in OpenMP will be due to forgotten variables in the private or reduction clause
- Following example shows a forgotten reduction clause
- Inspector can be used to find all necessary variables for the private/reduction clause!
Inspector XE – Data Races

Both threads are writing and reading to/from variable resid

Correct OpenMP pragma was commented
CLUSTER TOOLS

Intel® MPI Library

Intel® Trace Analyzer and Collector
Intel® MPI Library Overview

Optimized MPI application performance
- Application-specific tuning
- Automatic tuning
- Support for Intel® Xeon Phi™ processor (Knights Landing)
- 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
Using Intel® VTune™ Amplifier XE on MPI programs

Use -gtool flag or I_MPI_GTOOL variable to specify tools usage for restricted set of ranks

```bash
mpirun -gtool "amplxe-cl <vtune_args>:\<ranks>" <mpi_args> <application and args>

export I_MPI_GTOOL="amplxe-cl <vtune_args>:\<ranks>"
```

Hint: Let VTune create the command line arguments in the Vtune GUI

Same approach for Advisor XE!
Your application is OpenMP bound. High OpenMP imbalance has been identified.
Use Intel VTune Amplifier for further analysis.

Wallclock time
1.78 sec

Calculation
45.38%

OpenMP
30.53%

OpenMP Imbalance
12.90%

MPI
54.62%

Memory usage
25.29 MB
20.95 MB

Per-process memory usage affects the application scalability.

Cycles Per Instruction Rate
1.67
maxi 2.23
mini 1.10

This could be caused by such issues as memory stalls, instruction staling, branch misprediction or long latency instructions.

Memory Bound Coefficient
0.18
maxi 0.23
mini 0.14

It indicates that the application doesn't spend much time waiting for data. High values are usually bad. The application is not Memory Bound.

MPI Performance Snapshot

Setup MPS with VTune Amplifier counters

```
source $VT_ROOT/bin/mpsvars.sh --vtune
```

Run MPI program with MPS enabled

```
mpirun -mps -n 4 ./application
```

Create MPS summary (text output) and HTML output

```
mps _mps_2017mmdd-HHMMSS/ stat_2017mmdd-HHMMSS/
```

Visualize HTML file mps_report.html in browser
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
## Multiple Methods for Data Collection

<table>
<thead>
<tr>
<th>Collection Mechanism</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run with –trace or preload trace collector library (LD_PRELOAD=libVT.so)</td>
<td>Automatically collects all MPI calls, requires no modification to source, compile, or link.</td>
<td>No user code collection.</td>
</tr>
<tr>
<td>Link with –trace.</td>
<td>Automatically collects all MPI calls.</td>
<td>No user code collection. Must be done at link time.</td>
</tr>
<tr>
<td>Compile with –tcollect.</td>
<td>Automatically instruments all function entries/exits.</td>
<td>Requires recompile of code.</td>
</tr>
<tr>
<td>Add API calls to source code.</td>
<td>Can selectively instrument desired code sections.</td>
<td>Requires code modification.</td>
</tr>
</tbody>
</table>

```bash
export VT_LOGFILE_FORMAT=stfsingle (generates single STF trace file)
mpirun -trace -n 4 ./application
```
MPI Correctness Checking

Recommended usage is via `check_mpi`:

```bash
mpirun -check_mpi -n 2 overlap
```

[...]

```
[0] WARNING: LOCAL:MEMORY:OVERLAP: warning
[0] WARNING: New send buffer overlaps with currently active send buffer at address 0x7fbbbffec10.
[0] WARNING: Control over active buffer was transferred to MPI at:
[0] WARNING: MPI_Isend(buf=0x7fbbbffec10, count=4, datatype=MPI_INT, dest=0, tag=103, comm=COMM_SELF [0], *request=0x508980)
[0] WARNING: overlap.c:104
[0] WARNING: Control over new buffer is about to be transferred to MPI at:
[0] WARNING: MPI_Isend(buf=0x7fbbbffec10, count=4, datatype=MPI_INT, dest=0, tag=104, comm=COMM_SELF [0], *request=0x508984)
[0] WARNING: overlap.c:105
```
Additional Material

- **Product page** – overview, features, FAQs, support...
- **Training materials** – movies, tech briefs, documentation...
- **Evaluation guides** – step by step walk through
- **Reviews**

Additional Development Products:

- **Intel® Software Development Products**
Educating with Webinar Series about 2017 Tools

- Expert talks about the new features
- Series of live webinars, September 13 – November 8, 2016
- Attend live or watch after the fact.

[https://software.intel.com/events/hpc-webinars](https://software.intel.com/events/hpc-webinars)
Educating with High-Performance Programming Book

Knights-Landing-specific details, programming advice, and real-world examples.

Intel® Xeon Phi™ Processor High Performance Programming

- Techniques to generally increase program performance on any system and prepare you better for Intel Xeon Phi processors.

Available as of June 2016

http://lotsofcores.com

"I believe you will find this book is an invaluable reference to help develop your own Unfair Advantage."

James A.
Manager
Sandia National Laboratories
More Education with software.intel.com/moderncode

- Online community growing collection of tools, trainings, support
  - Features Black Belts in parallelism from Intel and the industry

- Intel® HPC Developer Conferences
devlopers share proven techniques and best practices
  - hpcdevcon.intel.com

- Hands-on training for developers and partners with remote access to Intel® Xeon® processor and Xeon Phi™ coprocessor-based clusters.
  - software.intel.com/icmp

- Developer Access Program provides early access to Intel® Xeon Phi™ processor codenamed Knights Landing plus one-year license for Intel® Parallel Studio XE Cluster Edition.
  - http://dap.xeonphi.com/
Choices to Fit Needs: Intel® Tools

All Products with support – worldwide, for purchase.
- Intel® Premier Support - private direct support from Intel
- support for past versions
- software.intel.com/products

Most Products without Premier support – via special programs for those who qualify
- students, educators, classroom use, open source developers, and academic researchers
- software.intel.com/qualify-for-free-software

- Intel® Performance Libraries without Premier support -Community licensing for Intel performance libraries
  - no royalties, no restrictions based on company or project size
  - software.intel.com/nest
THANK YOU!
BACK-UP
Problem Size and Configuration Information
Intel® Distribution for Python* Benchmarks

<table>
<thead>
<tr>
<th>Hardware/Problem Size</th>
<th>dot</th>
<th>lu</th>
<th>det</th>
<th>inv</th>
<th>cholesky</th>
<th>fft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Xeon® processor (32 core) and Intel® Xeon Phi™ processor (64 core)</td>
<td>(20k, 10k) and (10k, 20k)</td>
<td>(35k, 35k)</td>
<td>(15k, 15k)</td>
<td>(25k, 25k)</td>
<td>(40k, 40k)</td>
<td></td>
</tr>
<tr>
<td>Intel Xeon processor (1 core)</td>
<td>(20k, 5k) and (5, 20k)</td>
<td>(20k, 20k)</td>
<td></td>
<td>(10k, 10k)</td>
<td></td>
<td>520k</td>
</tr>
<tr>
<td>Intel Xeon Phi processor (1 core)</td>
<td>(20k, 300) and (300, 20k)</td>
<td>(6k, 6k)</td>
<td>(4k, 4k)</td>
<td>(2k, 2k)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Configuration Info:** apt/atlas: installed with apt-get, Ubuntu* 16.10, Python* 3.5.2, numpy* 1.11.0, scipy* 0.17.0; pip/openblas*: installed with pip, Ubuntu 16.10, python 3.5.2, numpy 1.11.1, scipy 0.18.0; Intel Python: Intel® Distribution for Python 2017; Hardware: Intel Xeon processor: Intel Xeon processor E5-2698 v3 @ 2.30 GHz (2 sockets, 16 cores each, HT=off), 64 GB of RAM, 8 DIMMS of 8GB@2133MHz; Intel Xeon Phi processor: Intel® Xeon Phi™ processor 7210 1.30 GHz, 96 GB of RAM, 6 DIMMS of 16GB@1200MHz

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
### Configurations for 2007-2016 Benchmarks

#### 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>icc version 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>Disabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>icc version 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>Disabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>icc version 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>Disabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>icc version 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>Disabled</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>Disabled</td>
<td>Fedora 20</td>
<td>3.13.5-202.fc20</td>
<td>icc version 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>Disabled</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>Disabled</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 and 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.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Copyright © 2016, 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