Intel® Parallel Studio XE

**Optimization Notice**

Copyright © 2018, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.*
Faster, Scalable Code, Faster
Intel® VTune™ Amplifier Performance Profiler

Accurate Data - Low Overhead
- CPU, GPU, FPU, threading, bandwidth...

Meaningful Analysis
- Threading, OpenMP region efficiency
- Memory access, storage device

Easy
- Data displayed on the source code
- Easy set-up, no special compiles

"Last week, Intel® VTune™ Amplifier helped us find almost 3X performance improvement. This week it helped us improve the performance another 3X."

Claire Cates
Principal Developer
SAS Institute Inc.

http://intel.ly/vtune-amplifier-xe
Intel® VTune™ Amplifier
Tune Applications for Scalable Multicore Performance

Agenda

- Data Collection –
  Rich set of performance data
- Data Analysis -
  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Compare results
  - Remote collection
- Advanced Features
- What’s New
- Summary
## Two Great Ways to Collect Data

**Intel® VTune™ Amplifier**

<table>
<thead>
<tr>
<th>Software Collector</th>
<th>Hardware Collector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses OS interrupts</td>
<td>Uses the on chip Performance Monitoring Unit (PMU)</td>
</tr>
<tr>
<td>Collects from a single process tree</td>
<td>Collect system wide or from a single process tree.</td>
</tr>
<tr>
<td>~10ms default resolution</td>
<td>~1ms default resolution (finer granularity - finds small functions)</td>
</tr>
<tr>
<td>Either an Intel® or a compatible processor</td>
<td>Requires a genuine Intel® processor for collection</td>
</tr>
<tr>
<td>Call stacks show calling sequence</td>
<td>Optionally collect call stacks</td>
</tr>
<tr>
<td>Works in virtual environments</td>
<td>Works in a VM only when supported by the VM (e.g., vSphere*, KVM)</td>
</tr>
<tr>
<td>No driver required</td>
<td>Requires a driver</td>
</tr>
<tr>
<td></td>
<td>- Easy to install on Windows</td>
</tr>
<tr>
<td></td>
<td>- Linux requires root (or use default perf driver)</td>
</tr>
</tbody>
</table>

No special recompiles - C, C++, C#, Fortran, Java, Assembly
# A Rich Set of Performance Data

**Intel® VTune™ Amplifier**

<table>
<thead>
<tr>
<th>Software Collector</th>
<th>Hardware Collector</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Hotspots</strong></td>
<td><strong>Advanced Hotspots</strong></td>
</tr>
<tr>
<td>Which functions use the most time?</td>
<td>Which functions use the most time?</td>
</tr>
<tr>
<td></td>
<td>Where to inline? – Statistical call counts</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td><strong>General Exploration</strong></td>
</tr>
<tr>
<td>Tune parallelism.</td>
<td>Where is the biggest opportunity?</td>
</tr>
<tr>
<td>Colors show number of cores used.</td>
<td>Cache misses? Branch mispredictions?</td>
</tr>
<tr>
<td><strong>Locks and Waits</strong></td>
<td><strong>Advanced Analysis</strong></td>
</tr>
<tr>
<td>Tune the #1 cause of slow threaded performance:</td>
<td>Memory-access, HPC Characterization, etc…</td>
</tr>
<tr>
<td>– waiting with idle cores.</td>
<td></td>
</tr>
<tr>
<td>Any IA86 processor, any VM, no driver</td>
<td>Higher res., lower overhead, system wide</td>
</tr>
</tbody>
</table>

**No special recompiles - C, C++, C#, Fortran, Java, Python, Assembly**

*Other names and brands may be claimed as the property of others.*
Example: Hotspots Analysis

Summary View

**General Exploration**
Hotspots viewpoint (change)

- **Elapsed Time**: 5.554s
  - CPU Time: 10.504s
  - Instructions Retired: 21,698,000,000
  - CPI Rate: 1.257
  - CPU Frequency Ratio: 1.041
  - Total Thread Count: 9
  - Paused Time: 0s

**Top Hotspots**
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

<table>
<thead>
<tr>
<th>Function</th>
<th>Module</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid_internet</td>
<td>_tachyon_omp.exe</td>
<td>5.539s</td>
</tr>
<tr>
<td>sphere_internet</td>
<td>_tachyon_omp.exe</td>
<td>3.247s</td>
</tr>
<tr>
<td>func@0x1002a59d</td>
<td>libomp5md.dll</td>
<td>0.149s</td>
</tr>
<tr>
<td>shader</td>
<td>_tachyon_omp.exe</td>
<td>0.117s</td>
</tr>
<tr>
<td>KdDelayExecutionThread</td>
<td>ntoskm.exe</td>
<td>0.081s</td>
</tr>
<tr>
<td>[Others]</td>
<td>N/A*</td>
<td>1.561s</td>
</tr>
</tbody>
</table>

*N/A is applied to non-runnable metrics.

**Average Bandwidth**

<table>
<thead>
<tr>
<th>Package</th>
<th>Total GB/sec</th>
<th>Read GB/sec</th>
<th>Write GB/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>package 0</td>
<td>5.715</td>
<td>3.504</td>
<td>2.212</td>
</tr>
</tbody>
</table>

**CPU Usage Histogram**
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the idle CPU usage value.

**Collection and Platform Info**
This section provides information about this collection, including result set size and collection platform data.
Example: Concurrency Analysis
Bottom-up View
## Example: General Exploration

**Bottom-up View**

<table>
<thead>
<tr>
<th>Function / Call Stack</th>
<th>Clockticks</th>
<th>Instructions Retired</th>
<th>CPI Rate</th>
<th>Front-End Bound</th>
<th>Bad Speculation</th>
<th>Back-End Bound</th>
<th>Retiring</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid_intersect</td>
<td>14,293,700,000</td>
<td>11,109,300,000</td>
<td>1.287</td>
<td>7.1%</td>
<td>4.2%</td>
<td>0.4%</td>
<td>27.0%</td>
</tr>
<tr>
<td>sphere_intersect</td>
<td>8,569,000,000</td>
<td>7,985,000,000</td>
<td>1.087</td>
<td>6.2%</td>
<td>0.0%</td>
<td>7.2%</td>
<td>18.2%</td>
</tr>
<tr>
<td>func@bx1002a59d</td>
<td>374,300,000</td>
<td>311,600,000</td>
<td>1.201</td>
<td>5.1%</td>
<td>0.0%</td>
<td>15.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>shader</td>
<td>271,700,000</td>
<td>195,700,000</td>
<td>1.388</td>
<td>10.5%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>31.6%</td>
</tr>
<tr>
<td>KeDelayExecutionThread</td>
<td>220,400,000</td>
<td>110,200,000</td>
<td>2.000</td>
<td>17.2%</td>
<td>0.0%</td>
<td>17.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>func@bx14015140</td>
<td>212,900,000</td>
<td>26,500,000</td>
<td>8.000</td>
<td>31.3%</td>
<td>0.0%</td>
<td>22.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>tm_intersect</td>
<td>195,700,000</td>
<td>210,900,000</td>
<td>0.928</td>
<td>4.9%</td>
<td>0.0%</td>
<td>9.7%</td>
<td>0.0%</td>
</tr>
<tr>
<td>__kmp_x86_pause</td>
<td>144,400,000</td>
<td>43,700,000</td>
<td>3.304</td>
<td>19.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>34.2%</td>
</tr>
<tr>
<td>[wow64cpu.dll]</td>
<td>104,500,000</td>
<td>41,800,000</td>
<td>2.500</td>
<td>18.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>63.6%</td>
</tr>
<tr>
<td>light_intersect</td>
<td>87,400,000</td>
<td>57,000,000</td>
<td>1.533</td>
<td>10.9%</td>
<td>0.0%</td>
<td>10.9%</td>
<td>0.0%</td>
</tr>
<tr>
<td>func@bx10013010</td>
<td>81,700,000</td>
<td>247,000,000</td>
<td>0.331</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>32.6%</td>
</tr>
<tr>
<td>VNorm</td>
<td>66,500,000</td>
<td>34,200,000</td>
<td>1.944</td>
<td>14.3%</td>
<td>0.0%</td>
<td>14.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>func@bx1009c00</td>
<td>58,900,000</td>
<td>125,400,000</td>
<td>0.470</td>
<td>0.0%</td>
<td>0.0%</td>
<td>16.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>SleepEx</td>
<td>58,900,000</td>
<td>17,100,000</td>
<td>3.444</td>
<td>0.0%</td>
<td>0.0%</td>
<td>16.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>__ibm_sse2_pow</td>
<td>55,100,000</td>
<td>26,600,000</td>
<td>2.071</td>
<td>17.2%</td>
<td>0.0%</td>
<td>17.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>func@bx140159900</td>
<td>55,100,000</td>
<td>11,400,000</td>
<td>4.833</td>
<td>0.0%</td>
<td>0.0%</td>
<td>17.2%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Find real CPU stalls due to cache misses, instruction fetch misses, branch misprediction, and a lot more.
Intel® VTune™ Amplifier
Tune Applications for Scalable Multicore Performance

Agenda

- Data Collection –
  Rich set of performance data
- Data Analysis -
  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Compare results
  - Remote collection
- Advanced Features
- What’s New
- Summary
Find Answers Fast
Intel® VTune™ Amplifier

Adjust Data Grouping
Function - Call Stack
Module - Function - Call Stack
Source File - Function - Call Stack
Thread - Function - Call Stack
... (Partial list shown)

Double Click Function to View Source
Click [+] for Call Stack
Filter by Timeline Selection (or by Grid Selection)

Filter by Process & Other Controls
Tuning Opportunities Shown in Pink. Hover for Tips
See Profile Data On Source / Asm
Double Click from Grid or Timeline

View Source / Asm or both
CPU Time
Right click for instruction reference manual

Quick Asm navigation:
Select source to highlight Asm

Scroll Bar "Heat Map" is an overview of hot spots
Click jump to scroll Asm
Timeline Visualizes Thread Behavior
Intel® VTune™ Amplifier

Optional: Use API to mark frames and user tasks
Optional: Add a mark during collection
Visualize Parallel Performance Issues
Look for Common Patterns

Coarse Grain Locks
High Lock Contention
Load Imbalance
Low Concurrency
Tune OpenMP for Efficiency and Scalability

Fast Answers: Is My OpenMP Scalable? How Much Faster Could It Be?

1) Is the serial time of my application significant enough to prevent scaling?
2) How much performance can be gained by tuning OpenMP?
3) Which OpenMP regions / loops / barriers will benefit most from tuning?
4) What are the inefficiencies with each region? (click the link to see details)
Intel® VTune™ Amplifier

Tune Applications for Scalable Multicore Performance

Agenda

- Data Collection –
  Rich set of performance data
- Data Analysis -
  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Compare results
  - Remote collection
- Advanced Features
- What’s New
- Summary
Command Line Interface

Automate analysis

`amplxe-cl` is the command line:
- Windows: `C:\Program Files (x86)\Intel\VTune Amplifier XE \bin[32|64]\amplxe-cl.exe`
- Linux: `/opt/intel/vtune_amplifier_xe/bin[32|64]/amplxe-cl`

Help: `amplxe-cl -help`

Use UI to setup
1) Configure analysis in UI
2) Press “Command Line...” button
3) Copy & paste command

Great for regression analysis – send results file to developer
Command line results can also be opened in the UI
MPI Analysis

Command line:

> mpirun -n 16 -ppn 4 -l amplxe-cl -collect advanced-hotspots -trace-mpi -result-dir my_result -- my_app.a

Or use gtool:

> mpirun -gtool "amplxe-cl -collect memory-access -result-dir my_result:7,5" my_app.a

Each process data is presented for each node they were running on:

my_result.host_name1 (rank 0-3)
my_result.host_name2 (rank 4-7)
my_result.host_name3 (rank 8-11)
my_result.host_name4 (rank 12-15)
Interactive Remote Data Collection
Performance analysis of remote systems just got a lot easier

Interactive analysis
1) Configure SSH to a remote Linux* target
2) Choose and run analysis with the UI

Command line analysis
1) Run command line remotely on Windows* or Linux* target
2) Copy results back to host and open in UI

Conveniently use your local UI to analyze remote systems
Compare Results Quickly - Sort By Difference

Intel® VTune™ Amplifier

Quickly identify cause of regressions.

- Run a command line analysis daily
- Identify the function responsible so you know who to alert

Compare 2 optimizations – What improved?

Compare 2 systems – What didn’t speed up as much?

![Comparison Table]

*Other names and brands may be claimed as the property of others.*
Intel® VTune™ Amplifier
Tune Applications for Scalable Multicore Performance

Agenda

- Data Collection – Rich set of performance data
- Data Analysis - Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Compare results
  - Remote collection
- Advanced Features
- What’s New
- Summary
Java Analysis

- Multiple simultaneous JVMs
- Sampling is fast / unobtrusive
- Mixed Java / C++ / Fortran
- See results on the Java source
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

*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 & 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
Intel® VTune™ Amplifier
Tune Applications for Scalable Multicore Performance

Agenda

- Data Collection – Rich set of performance data
- Data Analysis - Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Compare results
  - Remote collection
- Advanced Features
- What’s New
- Summary
<table>
<thead>
<tr>
<th>Latest Hardware</th>
<th>Better Metrics Easier Tuning</th>
<th>Container Support</th>
<th>Enhanced Profiling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Intel® Xeon Phi™ processor (codenamed Knights Landing)</td>
<td>Better Application Snapshot: Merges MPI + Application data</td>
<td>Profile inside Docker* &amp; Mesos* containers</td>
<td>Locks &amp; waits analysis for mixed Python* and native code</td>
</tr>
<tr>
<td></td>
<td>Enhanced metrics for MPI applications: Imbalance &amp; critical path rank</td>
<td>Attach to Running Java* services &amp; daemons</td>
<td>Memory consumption analysis for Python, C/C++</td>
</tr>
</tbody>
</table>
APPLICATION PERFORMANCE SNAPSHOT

EASY PERFORMANCE SNAPSHOT FOR THREADED MPI APPLICATIONS

Quick & easy performance overview
- Does the app need performance tuning?

MPI and non-MPI Apps
- Distributed MPI with or without threading
- Shared memory applications

Popular MPI implementations supported
- Intel® MPI
- MPICH and Cray MPI

Richer metrics on computation efficiency
- CPU (processor stalls, memory access)
- FPU (vectorization metrics)

*Linux only
Optimize Private Cloud-Based Applications

Profile Enterprise Applications
- Native C, C++, Fortran*
- Attach to running Java* services (e.g., Mail)
- Profile Java daemons without restart

Accurate, Low-Overhead Data Collection
- Advanced hotspots and hardware events
- Memory analysis
- Accurate stack information for Java and HHVM*

Popular Containers Supported
- Docker*
- Mesos*

Software collectors (e.g., locks & waits) and Python* profiling are not currently available for containers.

Intel VTune Amplifier User Interface

Host
- Run analysis
- Get results

Container
- No container configuration required
- Detection of the container is automatic

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Intel® VTune™ Amplifier
Tune Applications for Scalable Multicore Performance

Agenda

- Data Collection –
  Rich set of performance data
- Data Analysis -
  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Compare results
  - Remote collection
- Advanced Features
- What’s New
- Summary
Intel® VTune™ Amplifier

Faster, Scalable 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...¹
- 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
- Local & Remote Data Collection, Command Line
- Analyze Windows* & Linux* data on OS X*²

¹ Events vary by processor. ² No data collection on OS X*
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® Distribution for Python
Performance Scripting

Cluster Tools

Intel® Advisor
Vectorization Optimization and Thread Prototyping

Intel® Cluster Checker
Cluster Diagnostic Expert System

Intel® Trace Analyzer and Collector
MPI Profiler

Intel® MPI Library

Intel® Integrated Performance Primitives
Image, Signal, and Compression Routines

Intel® Threading Building Blocks
Task-Based Parallel C++ Template Library

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Intel® Advisor – Modernize Your Code

Optimize Vectorization & Prototype Threading

Modern Performant Code
- Vectorized (uses AVX, AVX-512)
- Efficient memory access
- Threaded

Intel Advisor
- Adds & optimizes vectorization
- Analyzes memory patterns
- Quickly prototypes threading

New for 2018! (partial list)
- Roofline analysis
- Faster data collection
- More recommendations


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](http://www.intel.com/performance). Configurations for 2007-2016 Benchmarks at the end of this presentation.
Intel® Inspector – Memory & Thread Debugger
Find & Debug Memory Leaks, Corruption, Data Races, Deadlocks & more

Correctness Tools Increase ROI By 12%-21%\(^1\)
- Errors found earlier are less expensive to fix
- Races & deadlocks not easily reproduced
- Memory errors are hard to find without a tool

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

What's new in 2018
- Fewer false positives
- C++ 17 std::shared_mutex added
- Windows SRW Locks added

\(^1\) 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

Roadmap Notice: All information provided here is subject to change without notice.
Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.
Legal Disclaimer & Optimization Notice

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 complete information visit www.intel.com/benchmarks.

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.

Copyright © 2018, 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