INTEL® VTUNE™ AMPLIFIER

PERFORMANCE PROFILER

Faster, Scalable Code, Faster

Intel® VTune™ Amplifier Performance Profiler

Accurate Data - Low Overhead

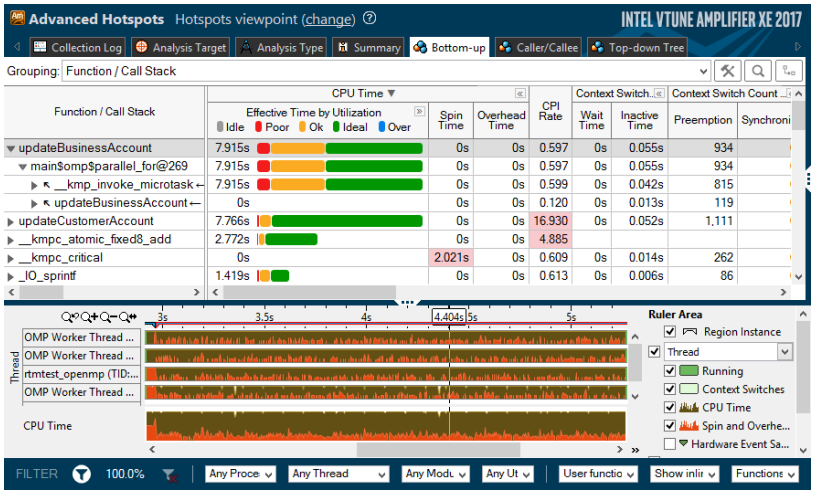
- CPU, GPU, FPU, threading, bandwidth…

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

Two Great Ways to Collect Data

Intel® VTune™ Amplifier

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

No special recompiles – C, C++, C#, Fortran, Java, Assembly

A Rich Set of Performance Data

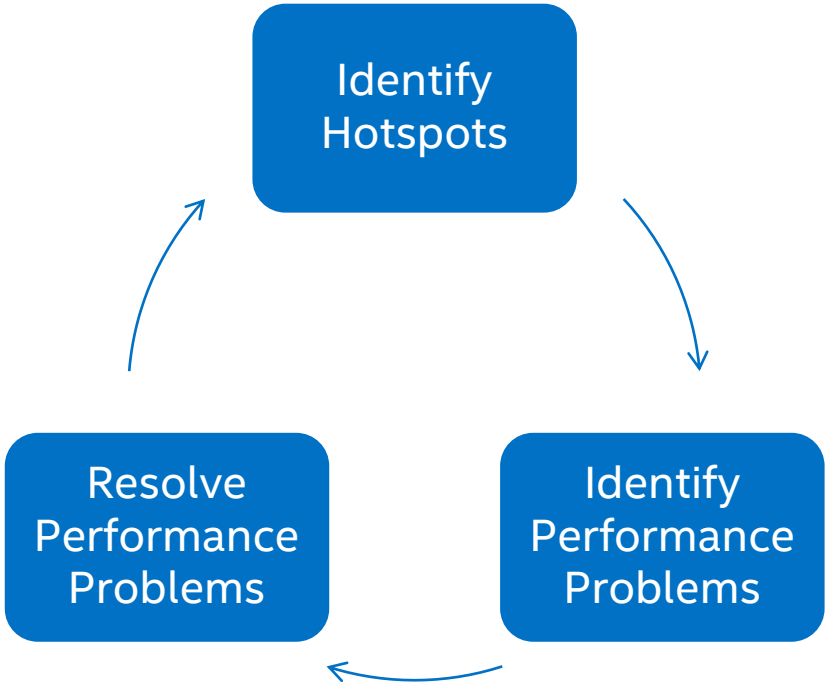
Intel® VTune™ Amplifier

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

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

PERFORMANCE ANALYSIS WITH INTEL® VTUNE™ AMPLIFIER

Analysis Workflow



VTUNE BY EXAMPLE

VTUNE FEATURES

Example: Hotspots Analysis

Summary View

General Exploration Hotspots viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up

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.

Function	Module	CPU Time
grid_intersect	3_tachyon_omp.exe	5.539s
sphere_intersect	3_tachyon_omp.exe	3.247s
func@0x1002e59d	libiomp5md.dll	0.148s
shader	3_tachyon_omp.exe	0.117s
KeDelayExecutionThread	ntoskrnl.exe	0.091s
[Others]	N/A*	1.361s

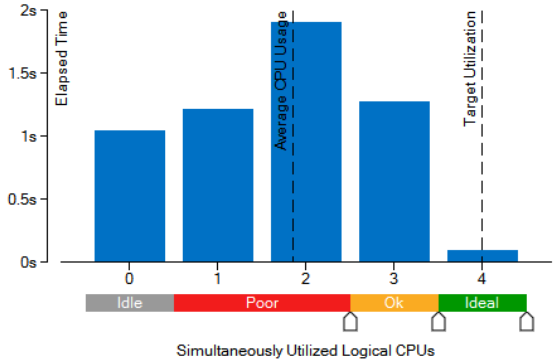
*N/A is applied to non-summable metrics.

Average Bandwidth

Package	Total, GB/sec	Read, GB/sec	Write, GB/sec
package_0	5.715	3.504	2.212

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.

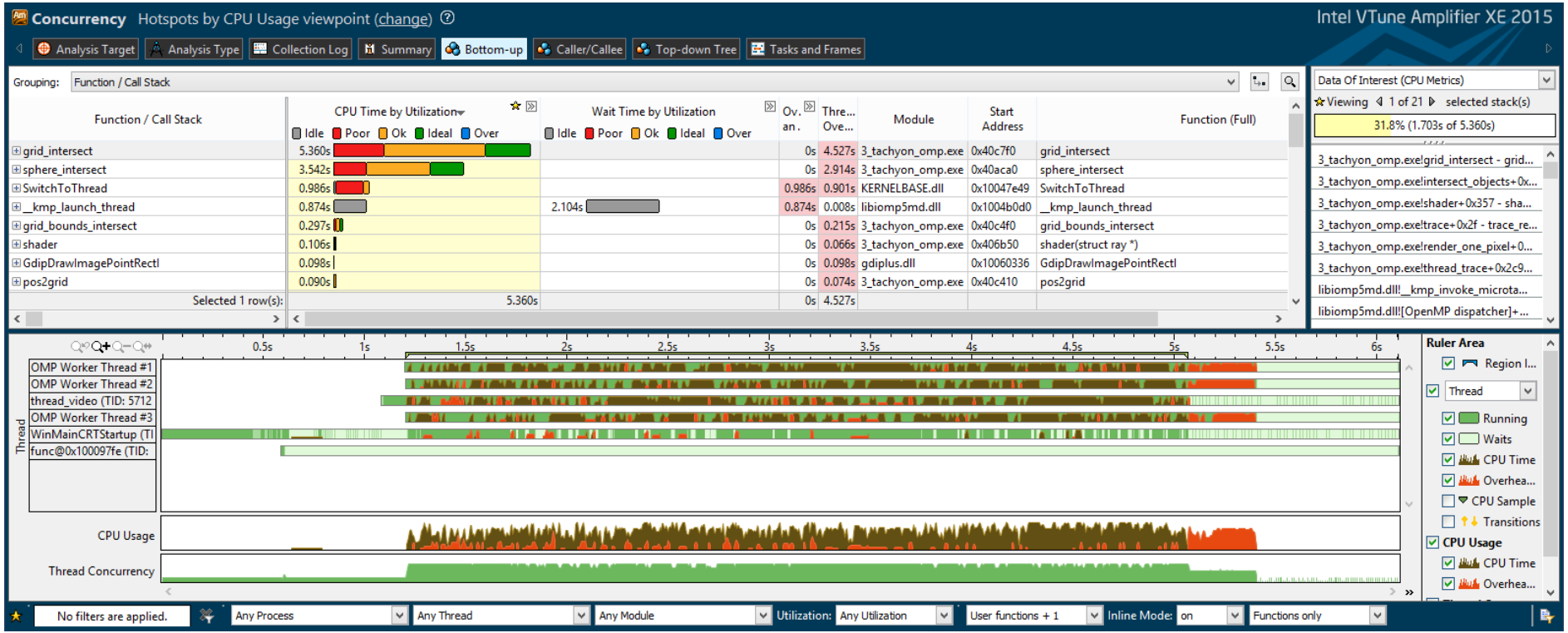


Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

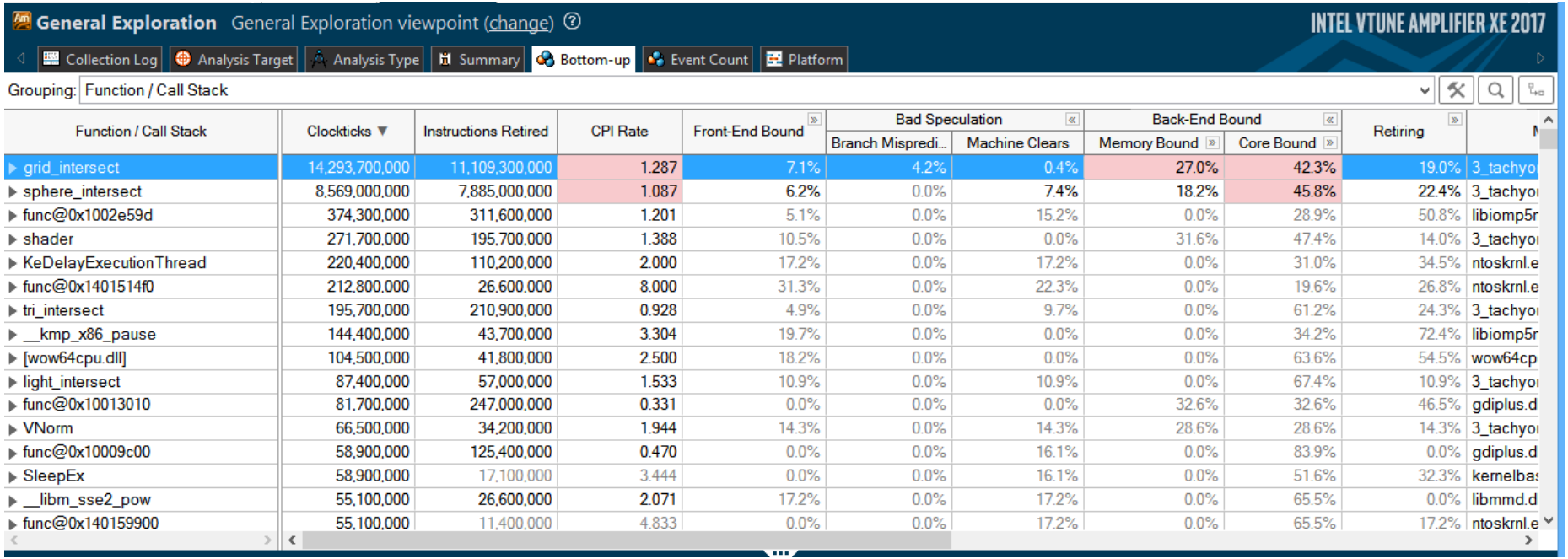
Example: Concurrency Analysis

Bottom-up View



Example: General Exploration

Bottom-up View



Function / Call Stack	Clockticks ▼	Instructions Retired	CPI Rate	Front-End Bound	Bad Speculation		Back-End Bound		Retiring	
					Branch Mispredi...	Machine Clears	Memory Bound	Core Bound		
grid_intersect	14,293,700,000	11,109,300,000	1.287	7.1%	4.2%	0.4%	27.0%	42.3%	19.0%	3_tachyo
sphere_intersect	8,569,000,000	7,885,000,000	1.087	6.2%	0.0%	7.4%	18.2%	45.8%	22.4%	3_tachyo
func@0x1002e59d	374,300,000	311,600,000	1.201	5.1%	0.0%	15.2%	0.0%	28.9%	50.8%	libiomp5r
shader	271,700,000	195,700,000	1.388	10.5%	0.0%	0.0%	31.6%	47.4%	14.0%	3_tachyo
KeDelayExecutionThread	220,400,000	110,200,000	2.000	17.2%	0.0%	17.2%	0.0%	31.0%	34.5%	ntoskrnl.e
func@0x1401514f0	212,800,000	26,600,000	8.000	31.3%	0.0%	22.3%	0.0%	19.6%	26.8%	ntoskrnl.e
tri_intersect	195,700,000	210,900,000	0.928	4.9%	0.0%	9.7%	0.0%	61.2%	24.3%	3_tachyo
__kmp_x86_pause	144,400,000	43,700,000	3.304	19.7%	0.0%	0.0%	0.0%	34.2%	72.4%	libiomp5r
[wow64cpu.dll]	104,500,000	41,800,000	2.500	18.2%	0.0%	0.0%	0.0%	63.6%	54.5%	wow64cp
light_intersect	87,400,000	57,000,000	1.533	10.9%	0.0%	10.9%	0.0%	67.4%	10.9%	3_tachyo
func@0x10013010	81,700,000	247,000,000	0.331	0.0%	0.0%	0.0%	32.6%	32.6%	46.5%	gdiplus.d
VNorm	66,500,000	34,200,000	1.944	14.3%	0.0%	14.3%	28.6%	28.6%	14.3%	3_tachyo
func@0x10009c00	58,900,000	125,400,000	0.470	0.0%	0.0%	16.1%	0.0%	83.9%	0.0%	gdiplus.d
SleepEx	58,900,000	17,100,000	3.444	0.0%	0.0%	16.1%	0.0%	51.6%	32.3%	kernelbas
__libm_sse2_pow	55,100,000	26,600,000	2.071	17.2%	0.0%	17.2%	0.0%	65.5%	0.0%	libmmd.d
func@0x140159900	55,100,000	11,400,000	4.833	0.0%	0.0%	17.2%	0.0%	65.5%	17.2%	ntoskrnl.e

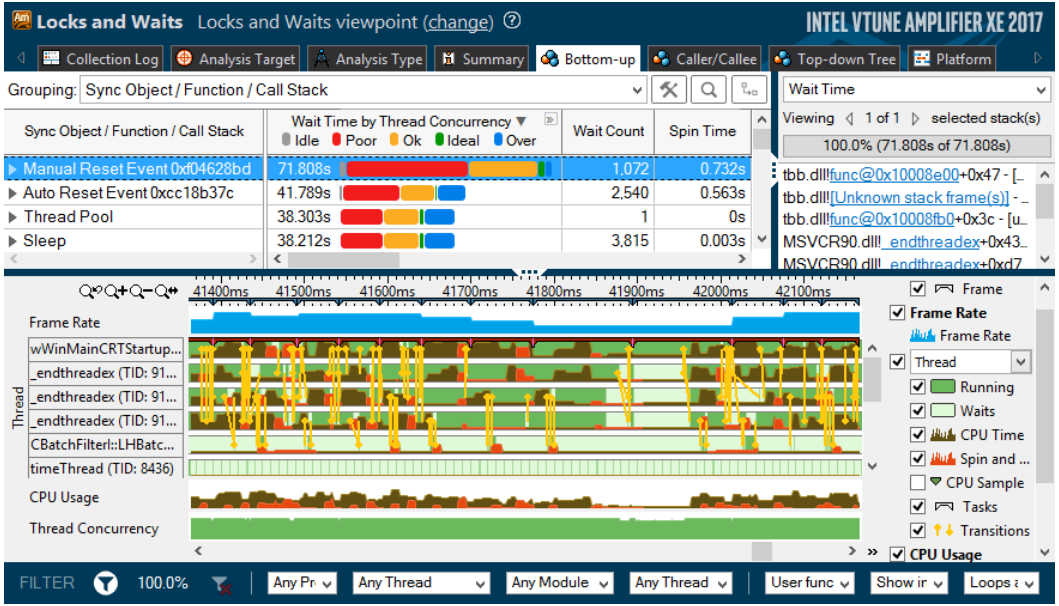
Find real CPU stalls due to cache misses, instruction fetch misses, branch misprediction, and a lot more.

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
- Additional Features
- Summary



Find Answers Fast

Intel® VTune™ Amplifier

Adjust Data Grouping

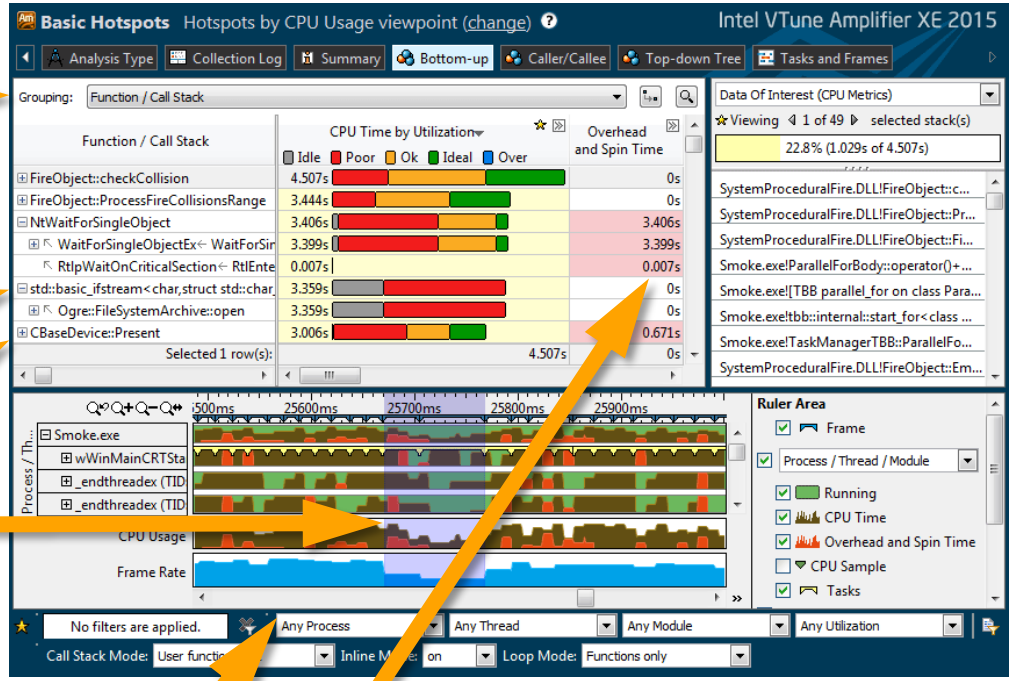
Function - Call Stack
Module - Function - Call Stack
Source File - Function - Call Stack
Thread - Function - Call Stack
... (Partial list shown)

Double Click Function to View Source

Click [+] for Call Stack

Filter by Timeline Selection (or by Grid Selection)

Zoom In And Filter On Selection
Filter In by Selection
Remove All Filters



Filter by Process & Other Controls

Tuning Opportunities Shown in Pink. Hover for Tips

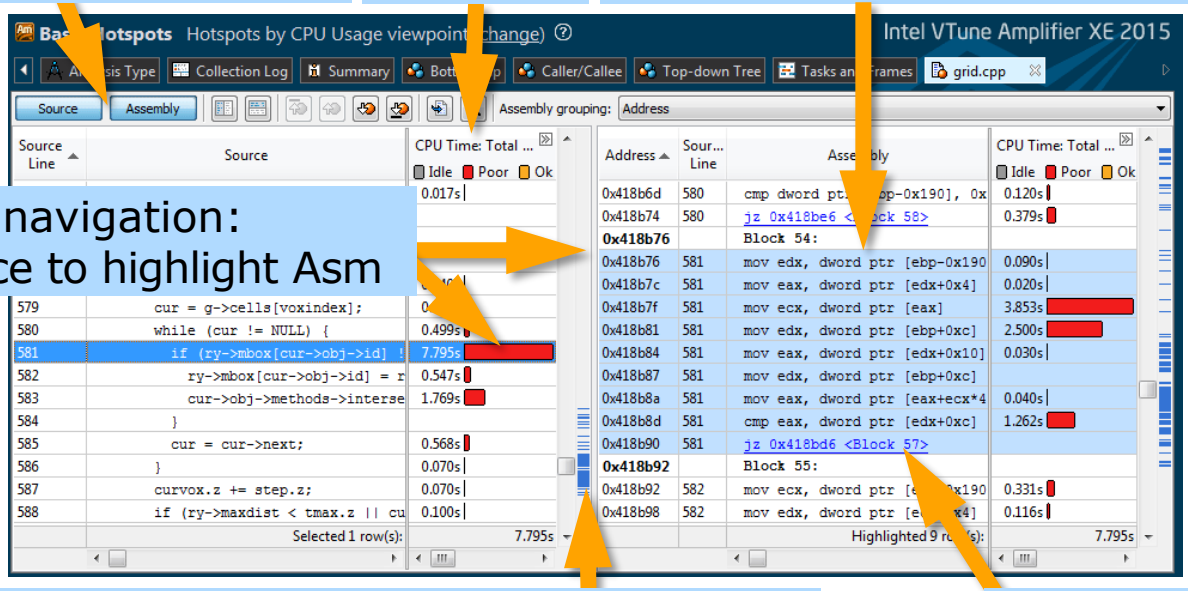
See Profile Data On Source / Asm

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

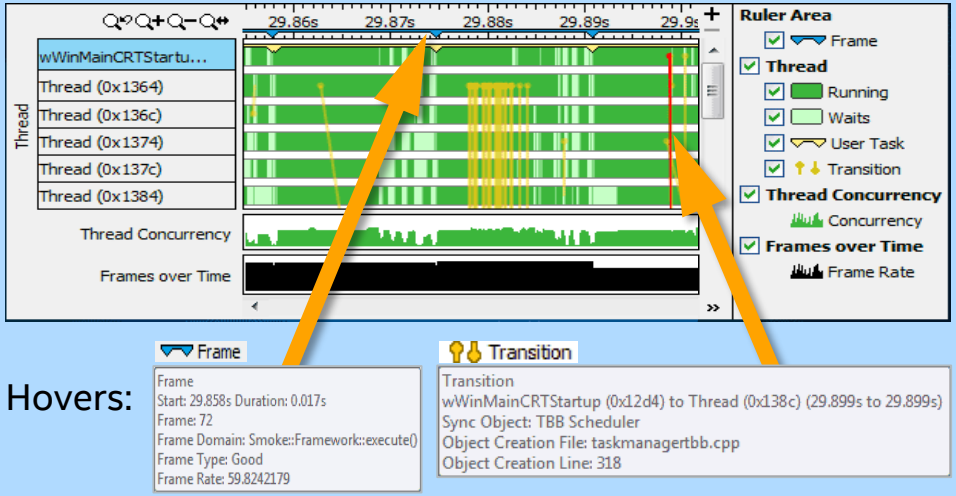
Click jump to scroll Asm

Timeline Visualizes Thread Behavior

Intel® VTune™ Amplifier

Transitions

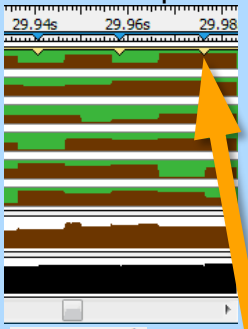
Locks & Waits



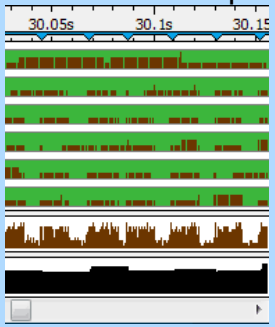
Hovers:

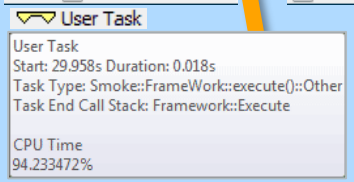
CPU Time

Basic Hotspots



Advanced Hotspots





Optional: Use API to mark frames and user tasks 

Optional: Add a mark during collection 

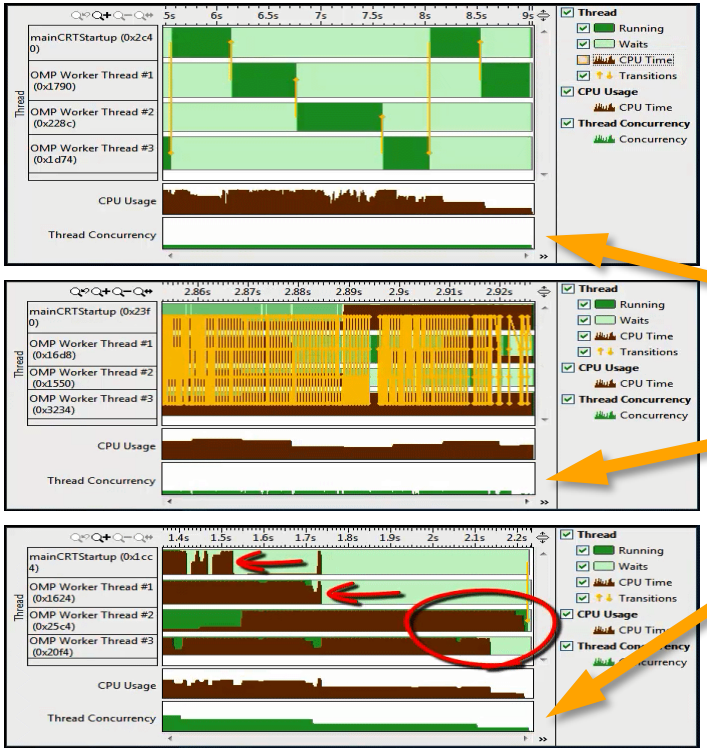
Visualize Parallel Performance Issues

Look for Common Patterns

Coarse Grain Locks

High Lock Contention

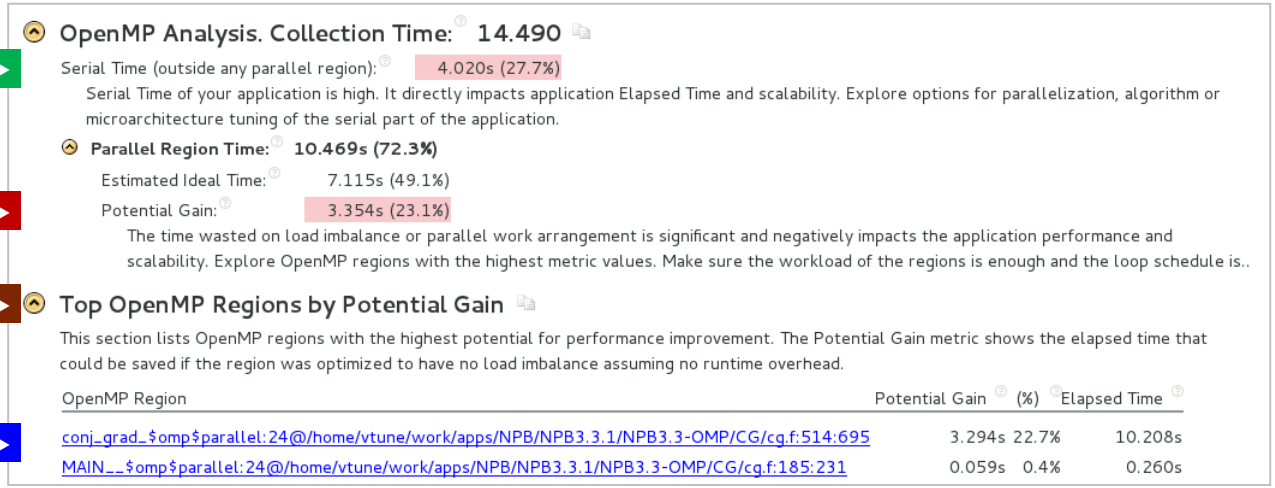
Load Imbalance



Low Concurrency

Tune OpenMP for Efficiency and Scalability

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



OpenMP Analysis. Collection Time: 14.490

1) ▶ Serial Time (outside any parallel region): 4.020s (27.7%)

Serial Time of your application is high. It directly impacts application Elapsed Time and scalability. Explore options for parallelization, algorithm or microarchitecture tuning of the serial part of the application.

Parallel Region Time: 10.469s (72.3%)

Estimated Ideal Time: 7.115s (49.1%)

2) ▶ Potential Gain: 3.354s (23.1%)

The time wasted on load imbalance or parallel work arrangement is significant and negatively impacts the application performance and scalability. Explore OpenMP regions with the highest metric values. Make sure the workload of the regions is enough and the loop schedule is..

3) ▶ Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	Potential Gain	(%)	Elapsed Time
4) ▶ conj_grad_ompparallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:514:695	3.294s	22.7%	10.208s
MAIN__ompparallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:185:231	0.059s	0.4%	0.260s

The summary view shown above gives fast answers to four important OpenMP tuning questions:

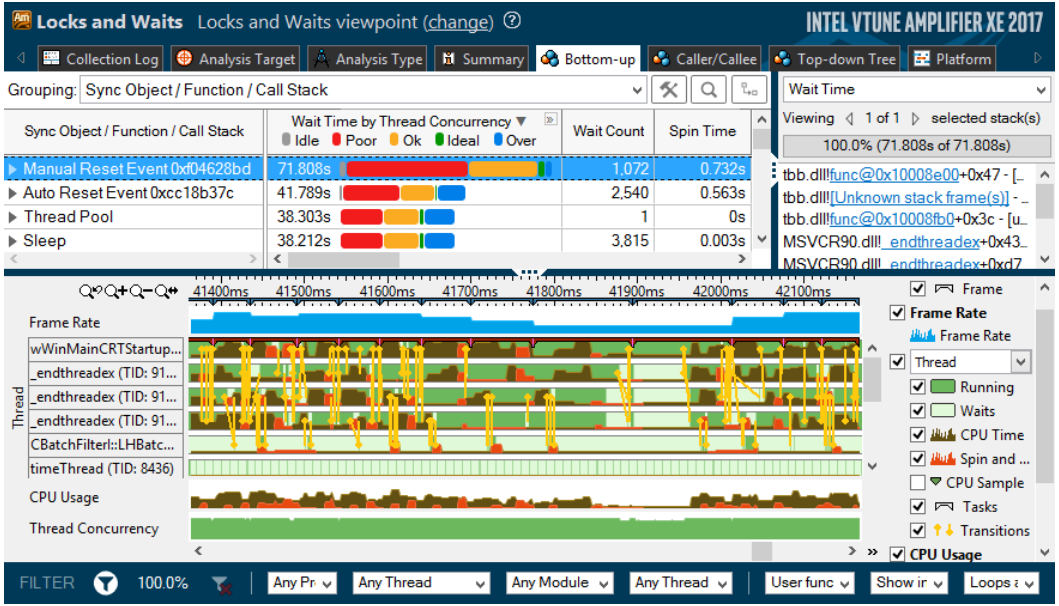
1) Is the serial time of my application significant enough to prevent scaling?
2) How much performance can be gained by tuning OpenMP?
3) Which OpenMP regions / loops / barriers will benefit most from tuning?
4) What are the inefficiencies with each region? (click the link to see details)

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
- Additional Features
- Summary



Command Line Interface

Automate analysis

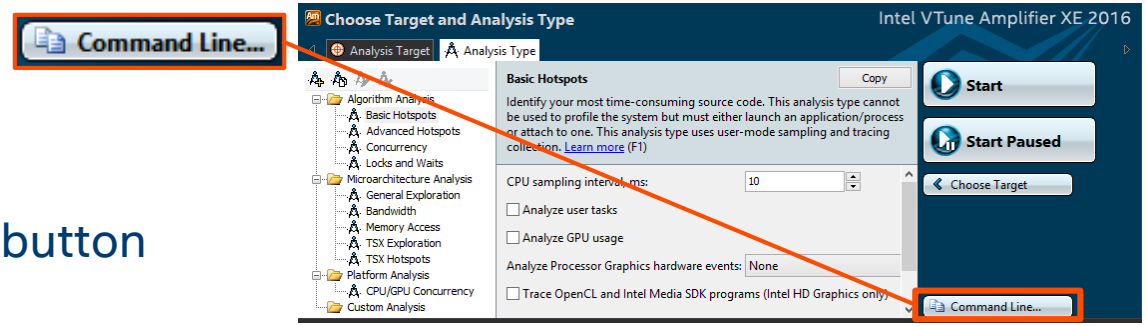
amplxe-cl is the command line:

- **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

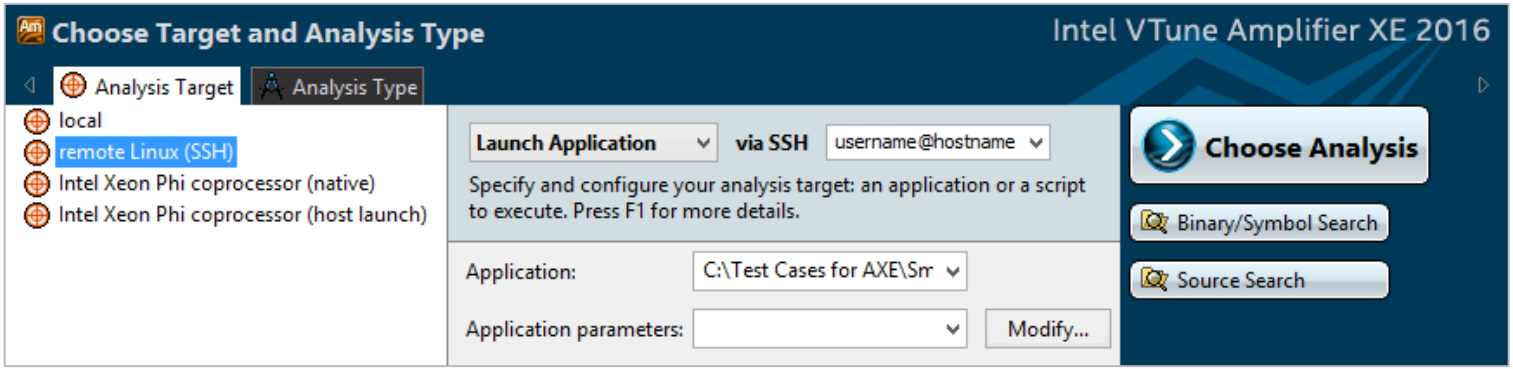
Performance analysis of remote systems just got a lot easier

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

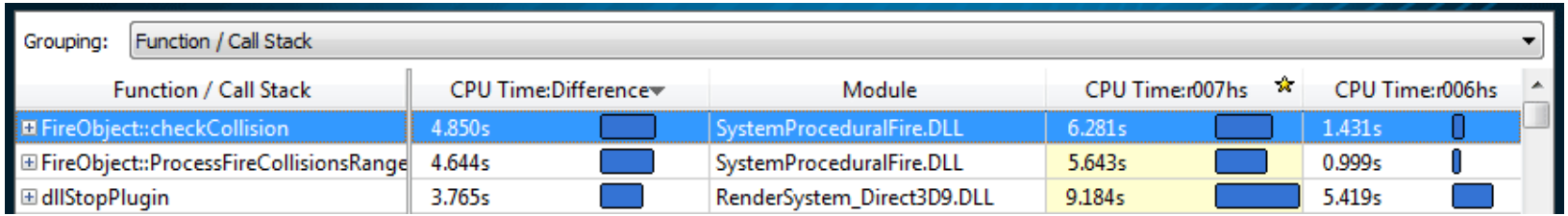
Intel® VTune™ Amplifier

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?

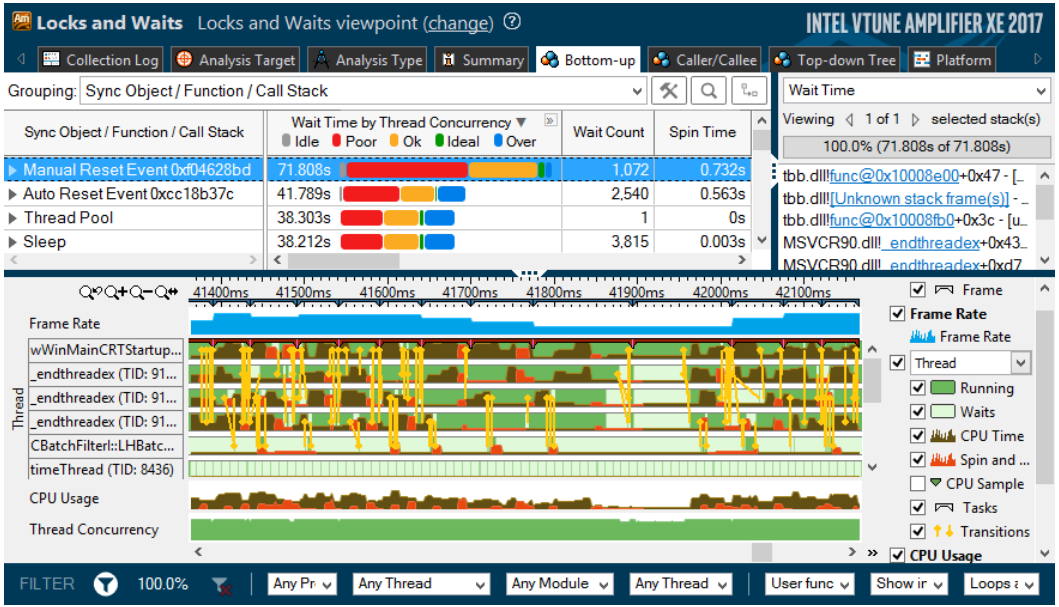


Intel® VTune™ Amplifier

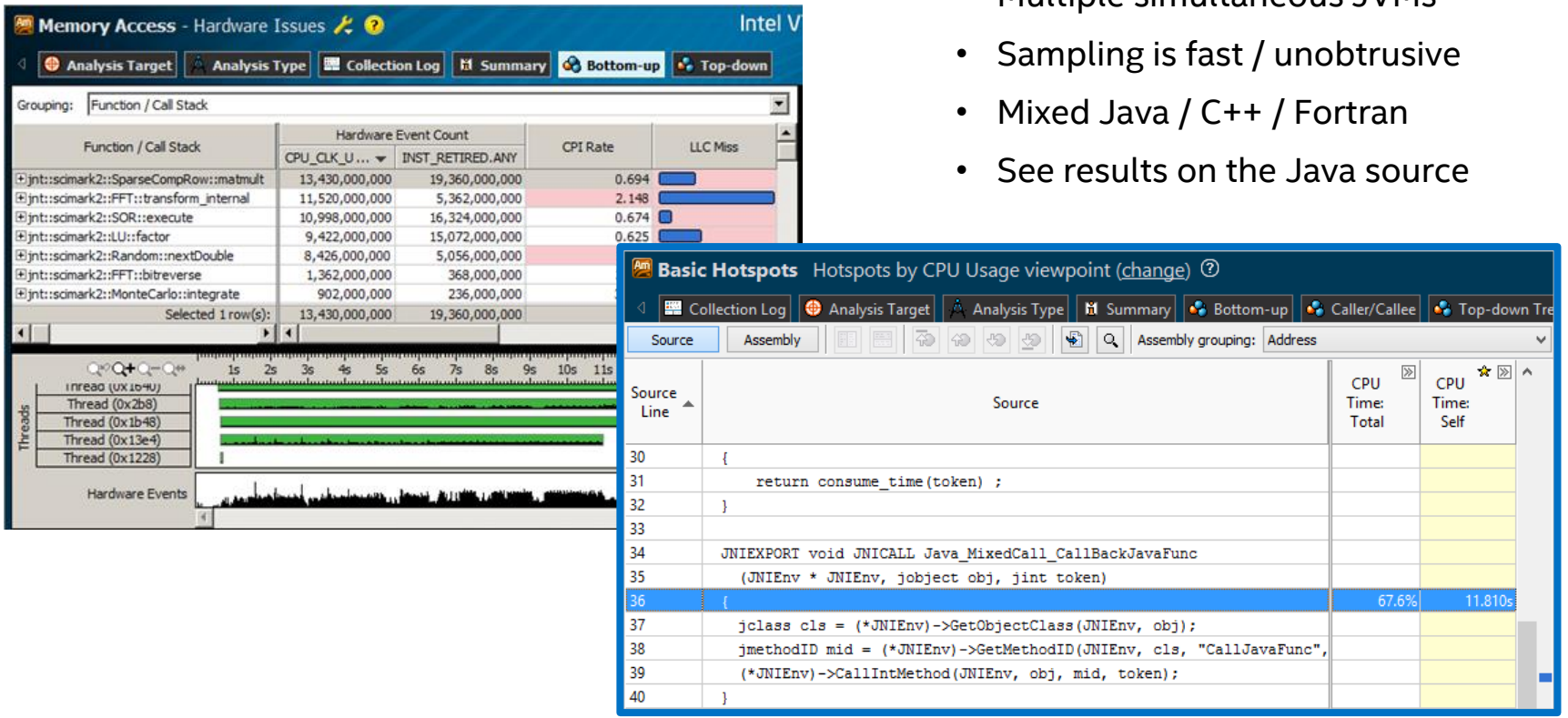
Tune Applications for Scalable Multicore Performance

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
- Additional Features
- Summary



Java Analysis



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

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

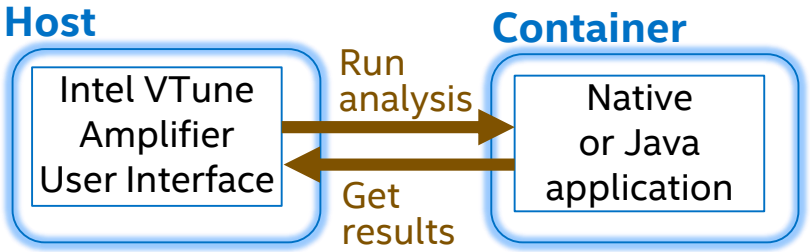
Accurate, Low-Overhead Data Collection

- Advanced hotspots and hardware events
- Memory analysis
- Accurate stack information for Java and HHVM*

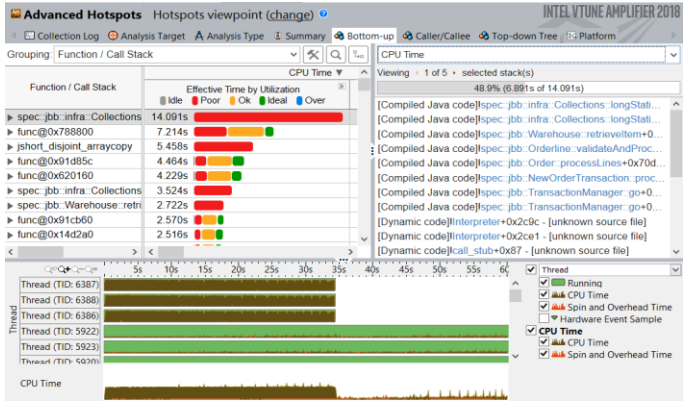
Popular Containers Supported

- Docker*
- Mesos*

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



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



Optimize Memory Access

Memory Access Analysis - Intel® VTune™ Amplifier 2017

Tune data structures for performance

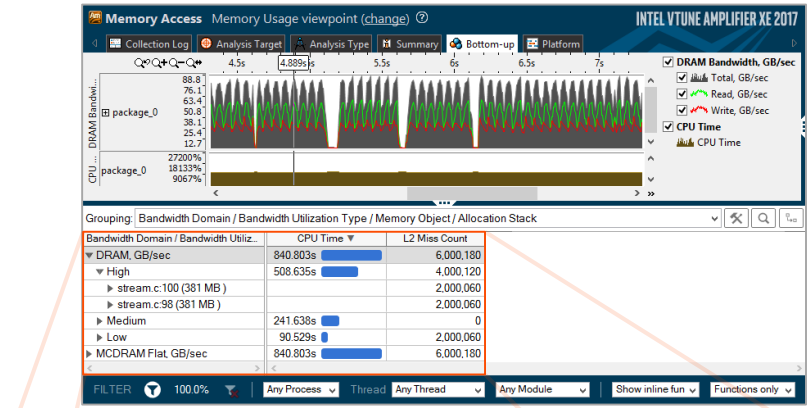
- Attribute cache misses to data structures (not just the code causing the miss)
- Support for custom memory allocators

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



Bandwidth Domain / Bandwidth Utiliz...	CPU Time ▼	L2 Miss Count
▼ DRAM, GB/sec	840.803s	6,000,180
▼ High	508.635s	4,000,120
▶ stream.c:100 (381 MB)		2,000,060
▶ stream.c:98 (381 MB)		2,000,060
▶ Medium	241.638s	0
▶ Low	90.529s	2,000,060
▶ MCDRAM Flat, GB/sec	840.803s	6,000,180

Storage Device Analysis (HDD, SATA or NVMe SSD)

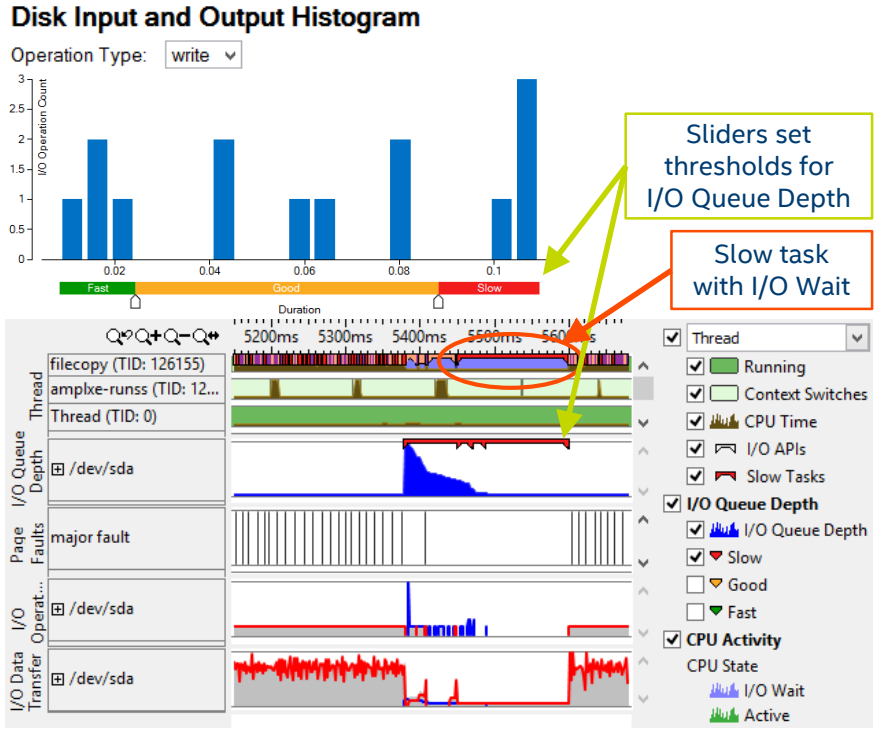
Intel® VTune™ Amplifier

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

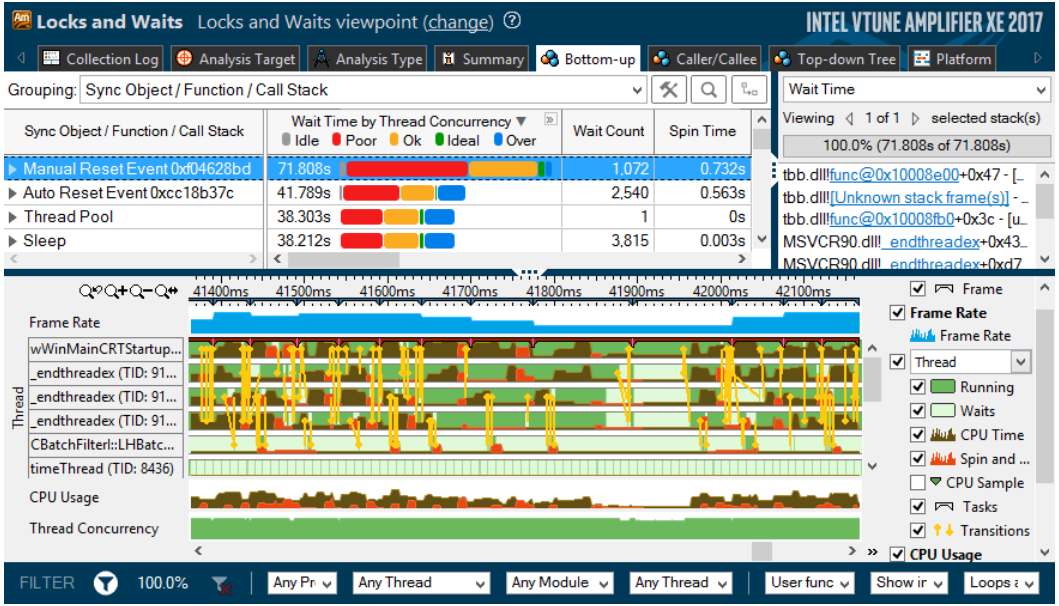


Intel® VTune™ Amplifier

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
- Additional Features
- 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

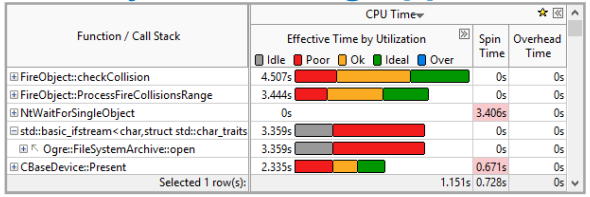
Find Answers Fast

- 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

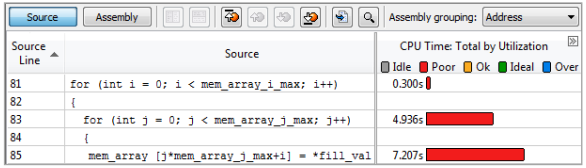
Easy to Use

- 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]

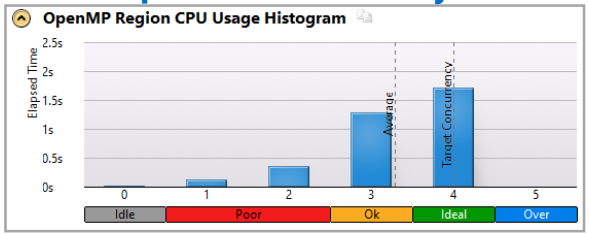
Quickly Find Tuning Opportunities



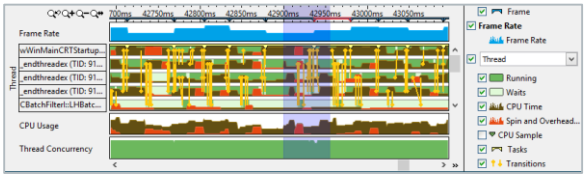
See Results On The Source Code



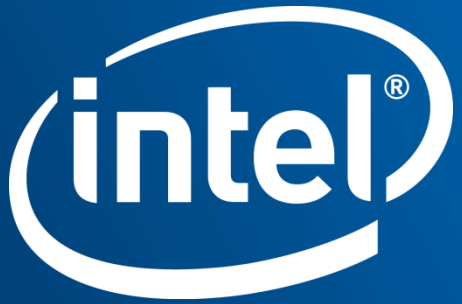
Tune OpenMP Scalability



Visualize & Filter Data



[1] Events vary by processor. [2] No data collection on OS X*


intel®

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.

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