

Intel® VTune™ Amplifier XE

Vladimir Tsymbal
Performance, Analysis and Threading Lab



Agenda

- Intel® VTune Amplifier XE Overview
 - Features
 - Data collectors
 - Analysis types
- Key Concepts
- Collecting performance data in cluster environment



Intel® Cluster Studio XE

Phase	Product	Feature	Benefit
Build	 Intel® MPI Library	High Performance Message Passing (MPI) Library	<ul style="list-style-type: none"> Enabling High Performance Scalability, Interconnect Independence, Runtime Fabric Selection, and Application Tuning Capability
	 Intel® Composer XE	C/C++ and Fortran compilers and performance libraries <ul style="list-style-type: none"> Intel® Threading Building Blocks Intel® Cilk™ Plus Intel® Integrated Performance Primitives Intel® Math Kernel Library 	<ul style="list-style-type: none"> Enabling solution to achieve the application performance and scalability benefits of multicore and forward scale to many-core
Verify	 Intel® Inspector XE	Memory & threading dynamic analysis for code quality Static Security Analysis for code quality	<ul style="list-style-type: none"> Increased productivity, code quality, and lowers cost, finds memory, threading, and security defects before they happen Now MPI enabled at every cluster node
Verify & Tune	 Intel® Trace Analyzer & Collector	MPI Performance Profiler for understanding application correctness & behavior	<ul style="list-style-type: none"> Analyze performance of MPI programs and visualize parallel application behavior and communications patterns to identify hotspots
Tune	 Intel® VTune™ Amplifier XE	Performance Profiler for optimizing application performance and scalability	<ul style="list-style-type: none"> Remove guesswork, saves time, makes it easier to find performance and scalability bottlenecks Now MPI enabled at every cluster node

Software & Services Group, Developer Products Division 

Intel® VTune™ Amplifier XE Performance Profiler

Where is my application...

Spending Time?

Function - Call Stack	CPU Time
algorithm_2	3.560s
do_xform	3.560s
algorithm_1	1.412s
BaseThreadInitTh	0.000s

- Focus tuning on functions taking time
- See call stacks
- See time on source

Wasting Time?

Line		MEM_LOAD... LLC_MISS
475	float rx, ry, rz =	
476	float param1 = (AP	30,000
477	float param2 = (AP	
478	bool neg = (rz < C	

- See cache misses on your source
- See functions sorted by # of cache misses

Waiting Too Long?

	Wait Time	Wait Count
176.504s	Idle Poor Ok Ideal	18,277
84.681s	Idle Poor Ok Ideal	5,499
84.612s	Idle Poor Ok Ideal	5,489

- See locks by wait time
- Red/Green for CPU utilization during wait

- Windows & Linux
- Low overhead
- No special recompiles

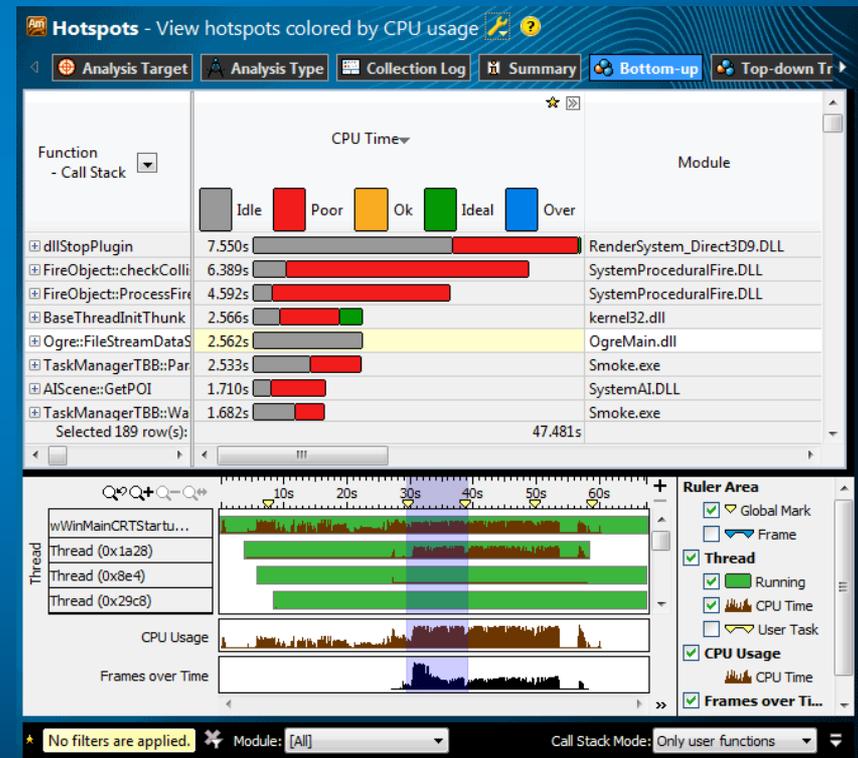
Advanced Profiling For Scalable Multicore Performance



Intel® VTune™ Amplifier XE

Tune Applications for Performance

- **Fast, Accurate Performance Profiles**
 - Hotspot (Statistical call tree)
 - Hardware-Event Based Sampling (EBS)
- **Thread Profiling**
 - Visualize thread interactions on timeline
 - Balance workloads
- **Easy set-up**
 - Pre-defined performance profiles
 - Use a normal production build
- **Compatible**
 - Microsoft, GCC, Intel compilers
 - C/C++, Fortran, Assembly, C#, .NET
 - Latest Intel® processors and compatible processors¹
- **Find Answers Fast**
 - Filter extraneous data
 - View results on the source / assembly
 - Event multiplexing
- **Windows or Linux**
 - Visual Studio Integration (Windows)
 - Standalone user i/f and command line
 - 32 and 64-bit



¹ IA32 and Intel® 64 architectures.
Many features work with compatible processors.
Event based sampling requires a genuine Intel® Processor.



Intel® VTune™ Amplifier XE

Powerful EBS Made Easier

System Wide Event Based Sampling (EBS)

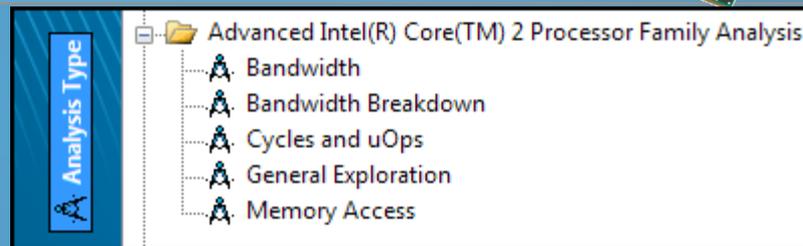
uses the on chip PMU to count performance events like cache misses, clock ticks and instructions retired.

Every Intel® Processor has an on chip Performance Monitoring Unit (PMU).



Predefined EBS Profiles

Easy EBS setup for newer processors.
No memorizing complex event names.
Profiles vary by microarchitecture.
(Full custom profiles also available)



Opportunities Highlighted

General Exploration turns the cell pink when it suspects a tuning opportunity is present. Hover gives suggestions.

/Function	PMU Event Count		CPI	Branch Mispredict
	CPU_CLK...	INST_RETIRE...		
initialize_2D_buffer	22,566,000,000	51,210,000,000	0.441	0.040
grid_intersect	11,304,000,000	10,778,000,000	1.049	0.205
sphere_intersect	11,030,000,000			
grid_bounds_intersec	1,580,000,000			

The CPI may be too high. This could be instruction starvation, branch mispredic the other hardware-related metrics to i

Pinpoint tuning opportunities

See opportunities like cache misses.
View results on the timeline, in the grid view or on your source.

Line	Source	MEM_LOAD... LLC_MISS
475	float rx, ry, rz = 1.f / (pos.z - prevPc	
476	float param1 = (AABB.zMin - prevPos.z)	30,000
477	float param2 = (AABB.zMax - prevPos.z)	
478	bool neg = (rz < 0.f);	



High-level Features



VTune™ Amplifier XE

High-level Features

- Hardware Event-based sampling (EBS)
 - Allows micro-architectural tuning
 - Improved usability
- Hotspot analysis (Software sampling)
 - Provides the time consuming regions of your application
 - Provides associated call-stacks that let you know how you got to these time consuming regions
 - Call-tree built using these call stacks
- Concurrency and Lock& Waits Analysis (Thread Profiling)
 - Visualize thread activity and lock transitions in the timeline
 - Provides lock profiling capability
 - Shows CPU/Core utilization and concurrency information



VTune™ Amplifier XE

High-level Features

- Attach to running processes
 - Hotspot and Concurrency analysis modes can attach to running processes
- System wide data collection
 - EBS modes allows system wide data collection and the tool provides the ability to filter this data
- GUI
 - Standalone GUI available on Windows* and Linux
 - Microsoft* Visual Studio integration
- Command line support
 - Comprehensive support for regression analysis and remote collection
- Platform & application support
 - Windows* and Linux
 - Microsoft* C# applications



VTune™ Amplifier XE

Improved data collection and visualization

- Timeline correlation of thread and event data
 - Populates thread active time with event data collected for that thread
 - Ability to filter regions on the timeline
- Advanced source and assembly views
 - See event data graphed in the source/assembly display
 - Visualize and analyze assembly as basic blocks
- Provides pre-defined tuning experiments



Data Collectors and Analysis Types



VTune™ Amplifier XE

Data Collectors

- Collectors come in two flavors
 - Event based sampling (EBS)
 - User mode sampling (Software sampling)
 - Thread profiling and stack sampling collector
 - Uses Pin based dynamic instrumentation technology
 - Application recompilation not necessary
 - Provides call-stacks with each sample
 - Statistical Call-tree constructed from call stacks



VTune™ Amplifier XE

Analysis types

- Software sampling based analysis types:
 - Hotspots
 - Concurrency
 - Locks and Waits
- Hardware EBS based analysis types:
 - Lightweight hotspots (pre-defined)
 - Advanced Hardware-level analysis (pre-defined)
 - Intel® Core™ i7 processor family
 - Intel® Xeon™ processor family
- Supports custom analysis types
 - Can be based on existing profiles



VTune™ Amplifier XE

Pre-defined Analysis Types

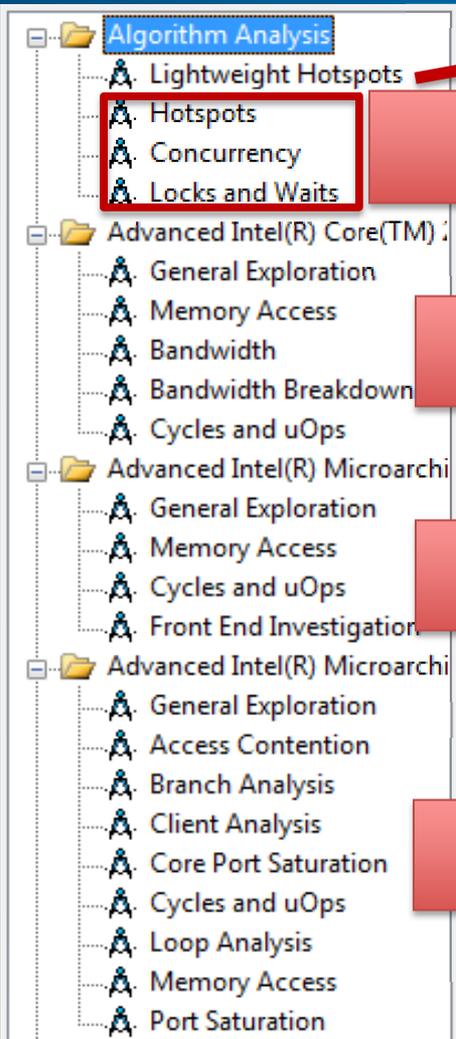
Lightweight Hotspot analysis based on the underlying architecture

User mode stack sampling
Threading, IO, Signaling API instrumentation

Core™ 2 Architecture Analysis types

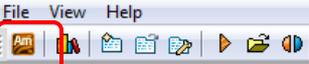
Core™ i7 (a.k.a Nehalem) Architecture analysis types

2nd Generation Core Architecture (a.k.a SandyBridge) analysis types



GUI Layout





r029hs r030hs r030hs-r029hs r031hs r031hs-r030hs New Amplifier XE Result

Choose Analysis Type

- Analysis Type
 - Algorithm Analysis
 - Lightweight Hotspots
 - Hotspots
 - Concurrency
 - Locks and Waits
 - Advanced Intel(R) Core(TM)
 - General Exploration
 - Memory Access
 - Bandwidth
 - Bandwidth Breakdown
 - Cycles and uOps
 - Advanced Intel(R) Microarch
 - General Exploration
 - Read Bandwidth
 - Write Bandwidth
 - Memory Access
 - Cycles and uOps
 - Front End Investigation
 - Advanced Intel(R) Microarch
 - General Exploration
 - Bandwidth
 - Access Contention
 - Branch Analysis
 - Client Analysis
 - Core Port Saturation
 - Cycles and uOps
 - Loop Analysis
 - Memory Access
 - Port Saturation
 - Advanced Intel(R) Atom(TM)
 - General Exploration
 - Custom Analysis
 - My Lightweight Hotspot

All available analysis types

Different ways to start the analysis

Start

Start Paused

Project Properties

Hotspots

Identify hotspots, Hotspots collects stack and call tree information. This analysis type cannot be used for attach to one. This analysis type uses user-mode sampling and tracing collection. Press F1 for more information.

Collect highly accurate CPU time

Details

CPU sampling interval, ms:	10
Detect context switches:	Yes
Collect CPU sampling data:	With stacks
Collect signalling API data:	No
Collect synchronization API data:	No
Collect I/O API data:	No
Stack unwinding mode:	After collection
Collect timeline data:	Yes

Resume

Pause

Stop

Cancel

Mark Timeline

Helps creating new analysis types

Copy from current

New CPU Event Collector analysis

New Stack-sampling Collector analysis

Copy the commandline to clipboard

Get Command Line

VTune™ Amplifier XE

GUI Layout

The screenshot displays the Intel VTune Amplifier XE 2011 GUI. The interface is divided into several key areas, each highlighted with a red box and labeled with a green callout:

- File Help**: The top menu bar.
- Hotspots**: The main title bar, indicating the current analysis target.
- Analysis Target** and **Analysis Type**: Navigation tabs for switching between different analysis targets and types.
- Menu and Tool bars**: A secondary set of navigation and tool options.
- Analysis Type**: A dropdown menu currently set to "Call Stack".
- Viewpoint currently being used**: A callout pointing to the "Bottom-up" and "Top-down Tree" buttons.
- Current grouping**: A callout pointing to the "Function" dropdown and the "Call Stack" view.
- Grid area**: A callout pointing to the central heatmap table.
- Stack Pane**: A callout pointing to the "Thread" list on the left side of the timeline.
- Filter area**: A callout pointing to the "Global Mark", "Frame", "Thread", "CPU Usage", and "Frames over Time" checkboxes on the right.
- Timeline area**: A callout pointing to the bottom section of the interface, including the "No filters are applied" status, "Module" and "Thread" filters, and the "Call Stack Mode" dropdown.

The central heatmap table shows the following data:

Function	Time	Color	Group
FireObject::checkCollision	6.542s	Red	0ms
dllStopPlugin	6.346s	Yellow	SystemProcedu...
TaskManagerTBB::WaitForSys...	6.155s	Yellow	enderSystem_D...
FireObject::ProcessFireCollisio	5.118s	Red	Smoke.exe
TaskManagerTBB::ParallelFor	2.905s	Yellow	SystemProcedu...
BaseThreadInitThunk	2.832s	Green	0ms
OpenFileStreamDataStream	2.602s	Yellow	kernel32.dll

The bottom timeline area shows a "Thread" list on the left, a "CPU Usage" graph in the middle, and a "Frames over Time" graph at the bottom. The "Thread" list includes:

- wWinMainCRTstartu...
- Thread (0x4c18)
- Thread (0x344)
- Thread (0x4f7c)
- Thread (0x537c)
- Thread (0x6f4)



VTune™ Amplifier XE

GUI Layout

Hotspots - View hotspots colored by CPU usage

Analysis Target | Analysis Type | Collection Log | **Summary** | Bottom-up | Top-down Tree

Result Summary

CPU Time: 58.024s
Frames: 649
Elapsed Time: 78.850s
Total Thread Count: 12
Overhead Time: 0.233s

Top Hotspots

Function	CPU Time
FireObject::checkCollision	6.542s
dllStopPlugin	6.346s
TaskManagerTBB::WaitForSystemTasks	6.155s
FireObject::ProcessFireCollisionsRange	5.118s
TaskManagerTBB::ParallelFor	2.905s
[Others]	30.958s

CPU Usage Histogram

This histogram represents a breakdown of the Elapsed Time. It visualizes what percentage of the wall time there were a specific number of simultaneously running CPUs. CPU Usage may be higher than the thread concurrency if a thread is spinning, or executing code on a CPU while it is logically waiting.

Simultaneously Utilized Logical CPUs	Elapsed Time
0	33.758s
1	33.758s
2	11.258s

Legend: Idle (grey), Poor (red), Ideal (green)

Collection and Platform Info

Information about this collection, including result set size and collection platform data

Command Line: E:\GDC 2010 Tutorial\bin\release\Smoke.exe
Frequency: 2.793 GHz
Logical CPU Count: 2
Operating System: Windows
Computer Name: VTOVINK-MOBL.amr.corp.intel.com
Result Size: 3 MB



VTune™ Amplifier XE

GUI Layout

Hotspots - View hotspots colored by CPU usage

Analysis Target Analysis Type Collection Log Summary Bottom-up **Top-down Tree**

Call Stack	CPU Time	CPU Time:Total
LdrGetProcedureAddressEx	0.998s	1.7%
[TBB parallel_for on class ParallelForBoc	0.034s	9.7%
[fmodex.dll]	0.014s	0.0%
ParallelForBody::operator()	0s	9.6%
FireTask::UpdateCallback	0s	7.9%
ChangeManager::DistributionCall	0s	1.4%
FireObject::FireCollisionCallback	0s	0.2%
FireObject::ProcessFireCollision	0.062s	0.2%
FireObject::checkCollision	0.038s	0.1%
AIScene::UpdateCallback	0s	0.0%
FireObject::UpdateCallback	0s	0.2%
FireObject::UpdateRange	0s	0.2%
FireObject::EmitterCollision	0s	0.1%
FireObject::ProcessFireCo	0.028s	0.1%
FireObject::checkCollie	0.050s	0.1%
ParticleEmitter::ParticleSyste	0s	0.1%
FireObject::BuildVerticesCallback	0s	0.0%
Ogre::ParticleSystem::_updateBounds	0.018s	0.0%
Ogre::RenderQueue::addRenderable	0.018s	0.0%
[fmodex.dll]	0.013s	0.0%
dllStopPlugin	0.012s	0.0%
Ogre::Instance::CreateBatchInsta	0.008s	0.0%
Selected 1 row(s):	0s	9.6%

Top down tree shows the statistical call-graph built from the samples



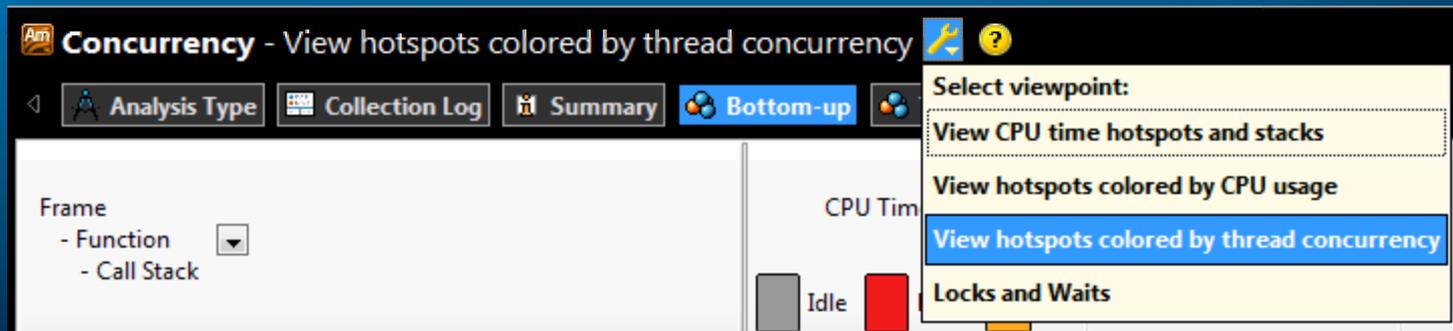
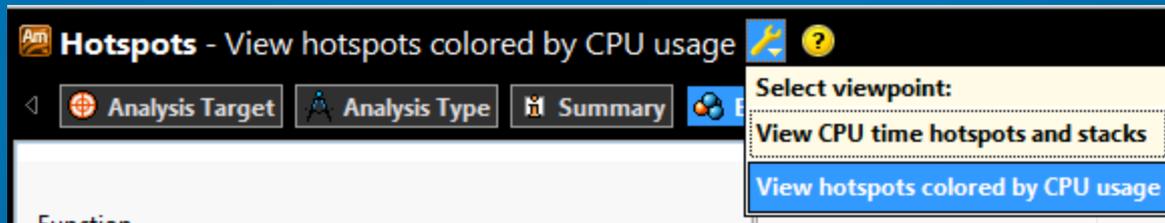
Key Result Analysis and GUI Concepts



VTune™ Amplifier XE

Key Concepts

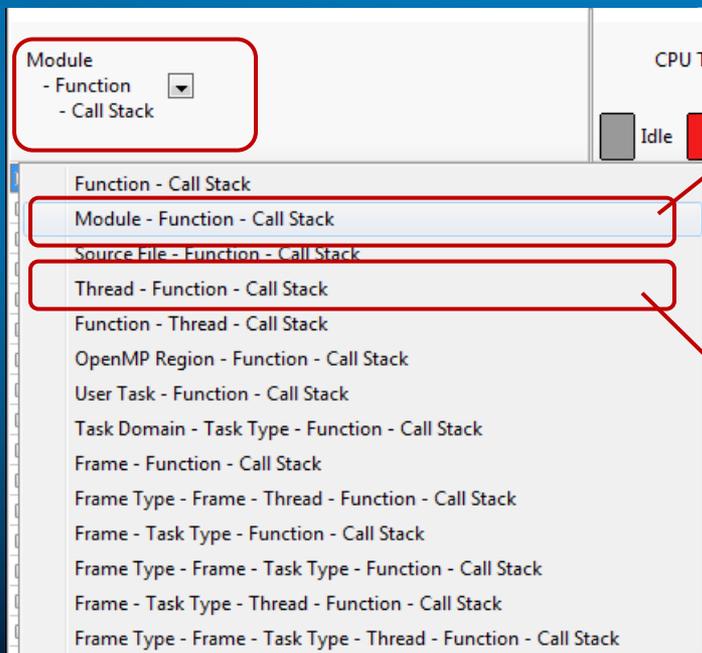
- Viewpoints
 - Example of an analysis type having multiple viewpoints



VTune™ Amplifier XE

Key Concepts

- Groupings
 - Each analysis type has many viewpoints
 - Each viewpoint has pre-defined groupings
 - Allows you to analyze the data in different hierarchies and granularities



Module	CPU Time
- Function - Call Stack	Idle Poor Ok
shaderapid9.dll	17.397s
⊕ CShaderAPI8::OcclusionQuery_GetNumPixelsRendered	6.810s
⊕ CShaderDeviceDx8::ShutdownDevice	2.267s
⊕ DestroyD3DTexture	0.802s
⊕ CVertexBuffer::Lock	0.681s
⊕ CShaderAPI8::SetSkinningMatrices	0.601s

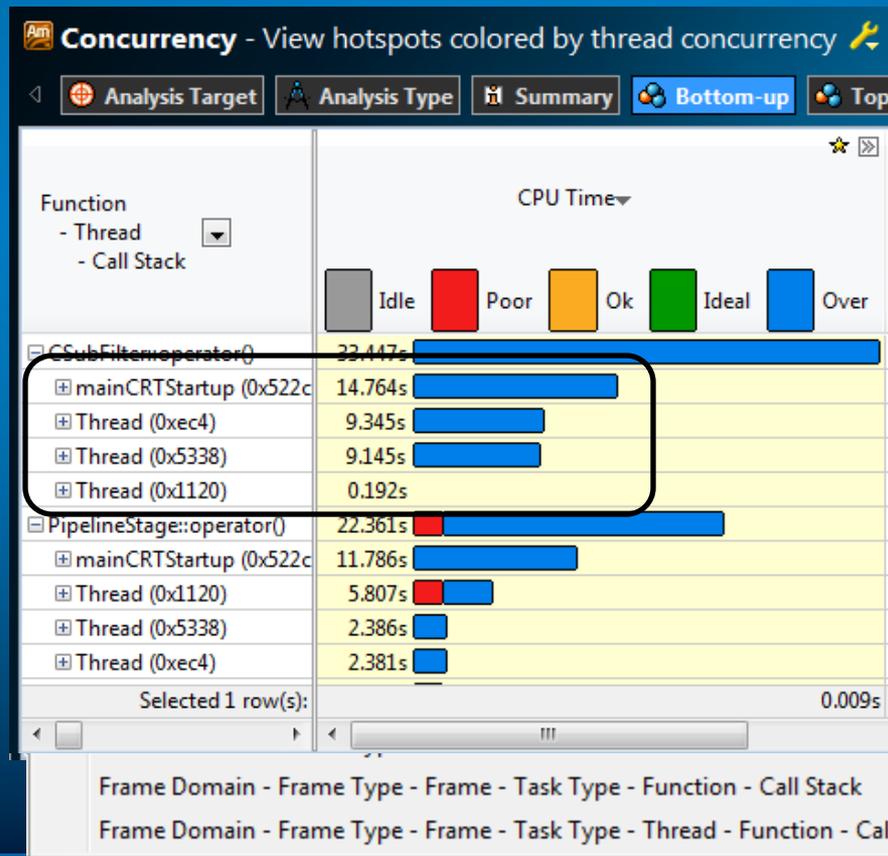
Function	CPU Time
- Thread - Call Stack	Idle Poor Ok
VCR_WaitForMultipleObjects	8.759s
⊖ CShaderAPI8::OcclusionQuery_GetNumPixelsRendered	6.810s
⊕ threadstartex (0x1c60)	3.782s
⊕ threadstartex (0x1988)	1.822s
⊕ threadstartex (0x1d38)	1.186s
⊕ WinMainCRTStartup (0x2304)	0.020s



VTune™ Amplifier XE

Key Concepts

- For example, pre-defined groupings can be used to determine load imbalance



VTune™ Amplifier XE

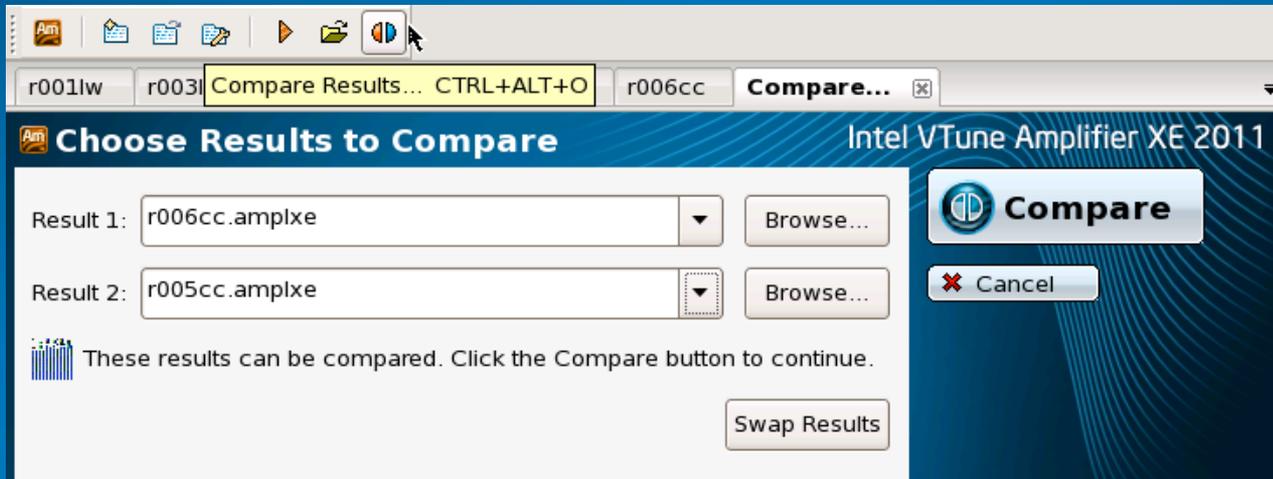
Key Concepts

- VTune™ Amplifier XE allows comparison of two similar runs
- Extremely useful for
 - Benchmarking
 - Regression analysis
 - Testing



VTune™ Amplifier XE

Key Concepts



/Function	CPU Time: Difference by Utilization	CPU Time: r006cc by Utilization					CPU Time: r005cc by Utilization					
		Idle	Poor	Ok	Ideal	Over	Idle	Poor	Ok	Ideal	Over	
grid_intersect	-3.122s											
sphere_intersect	-2.388s											
Raypnt	0s											
pos2grid	-0.070s											



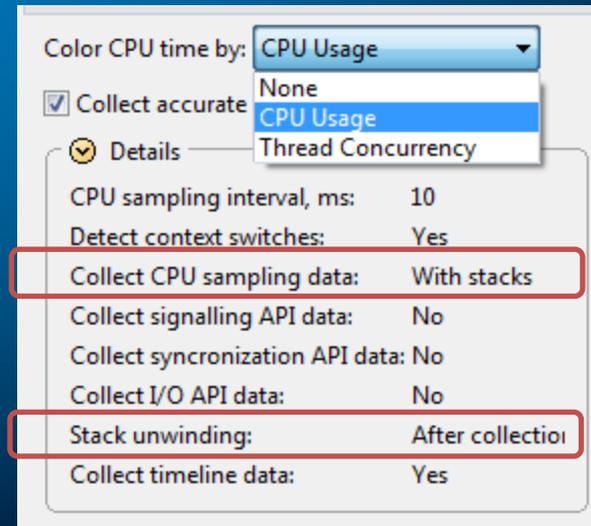
Demo/Lab Activities Analysis Types Revisited



Analysis Types

Hotspots

- For each sample, capture execution context, time passed since previous sample and thread CPU time
- Allows time spent in system calls to be attributed to user functions making the call
- Provides additional knobs:
 - Collect accurate CPU information requires “Administrator” privileges and uses ETW data to determine context switches
 - Timeline does not reflect CS
 - If “Thread Concurrency” is selected for coloring CPU time, then Pin Probes is used to capture wait data (slightly higher overhead)
 - The defaults for Hotspot analysis are configurable and can be done so by creating a custom analysis type inherited from Hotspots



VTune™ Amplifier XE

Terminology used

- CPU Time
 - The amount of time a thread spends executing on a logical processor
 - For multi-threaded applications, the CPU time is aggregated over all threads for the given level of granularity
- Wait Time
 - The amount of time that a given thread waited for some event to occur, such as: synchronization waits and I/O waits



VTune™ Amplifier XE

Terminology used



- **Elapsed Time:** 6 seconds
- **CPU Time:** T1 (4s) + T2 (2s) + T3 (2s) = 8 seconds
- **Wait Time:** T1(2s) + T2(3s) + T3 (2s) = 7 seconds



Hotspots analysis

The screenshot displays the Intel VTune Amplifier XE 2011 interface for analyzing CPU time hotspots. The main workspace is divided into several sections:

- Call Stack Table:** A table showing function hotspots with columns for Function, CPU Time, and Module. The top entry is `initialize_2D_buffer` with a CPU time of 6.383s.
- Thread Timeline:** A horizontal bar chart showing the execution of threads over time. The threads listed are `WinMainCRTStartup (0...`, `Thread (0x1df8)`, and `thread_video (0x468)`.
- CPU Usage Graph:** A bar chart showing CPU usage over time, with a peak of 100.0% (0.016s of 0.016s).

Callouts highlight key features:

- Function hotspot:** Points to the `initialize_2D_buffer` function in the call stack table.
- Call stack:** Points to the `grid_intersect` function in the call stack table.
- Function CPU time:** Points to the CPU Time column in the call stack table.
- Thread timeline:** Points to the thread execution bars in the thread timeline section.

Function	CPU Time	Module
initialize_2D_buffer	6.383s	tachyon_find_hotspots.exe
grid_intersect	5.268s	tachyon_find_hotspots.exe
intersect_objects	4.914s	tachyon_find_hotspots.exe
grid_intersect ← intersect_objects	0.354s	tachyon_find_hotspots.exe
sphere_intersect	2.768s	tachyon_find_hotspots.exe
video::main	0.910s	tachyon_find_hotspots.exe
grid_boun	0.232s	tachyon_find_hotspots.exe
chader	0.200s	tachyon_find_hotspots.exe



Hotspots analysis – Source View

Hotspots - View CPU time hotspots and stacks

Analysis Type | Collection Log | Summary | Bottom-up | Top-down Tree | sphere.cpp

Line	Source	CPU Time	Address	Assembly	CPU Time
110			0xedc8	fldq 0x24(%ebx), %st0	
111	VSUB(spr->ctr, ry->o, V);	2.560s	0xedcb	fld %st2, %st0	
112	VDOT(b, V, ry->d);	3.900s	0xedcd	fld %st2, %st0	
113	VDOT(temp, V, V);	0.578s	0xedcf	fxch %st2, %st0	0.241s
114			0xedd1	fsubrq 0x20(%esi), %st0	0.330s
115	disc=b*b + spr->rad*spr->rad - temp;	1.641s	0xedd4	fxch %st2, %st0	0.120s
116			0xedd6	fmulq 0x34(%ebx), %st0	0.250s
117	if (disc<=0.0) return;	1.500s	0xedd9	fxch %st1, %st0	0.050s
118	disc=sqrt(disc);	0.050s	0xeddb	fmulq 0x2c(%ebx), %st0	0.241s
119			0xedde	fxch %st2, %st0	1.490s
120	t2=b+disc;		0xede0	fstq %st0, -0x18(%ebp)	0.430s

Selected (1 row(s)): 3.900s | Highlighted 13 row(s): 3.900s

Thread (0x7fffff) | tbb::internal::rml... | CPU Usage

No filters are applied. Module: [All] Thread: [All] Call Stack Mode: Only user functions

Before starting demo/lab activities
let's quickly check how to use VTune™
Amplifier XE

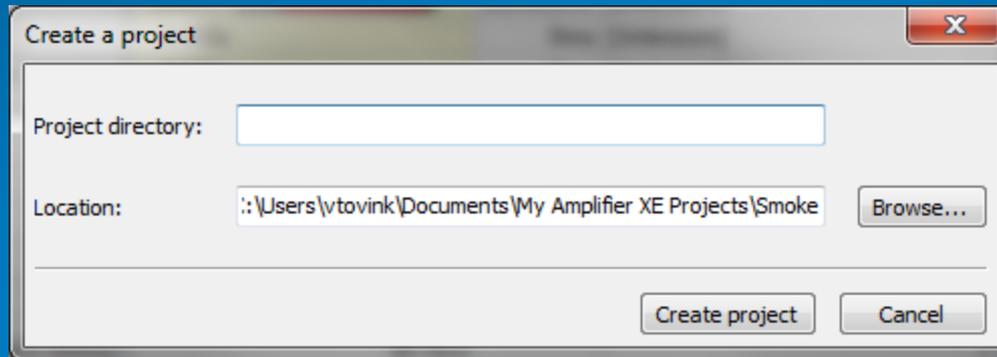


Creating a New Project

- Project management support provided is minimal
 - Allows users to create new projects
 - Each project is defined by an executable of interest (or system wide for EBS)
 - Associated command line arguments that describe the workload, if applicable

Steps for creating a new Project

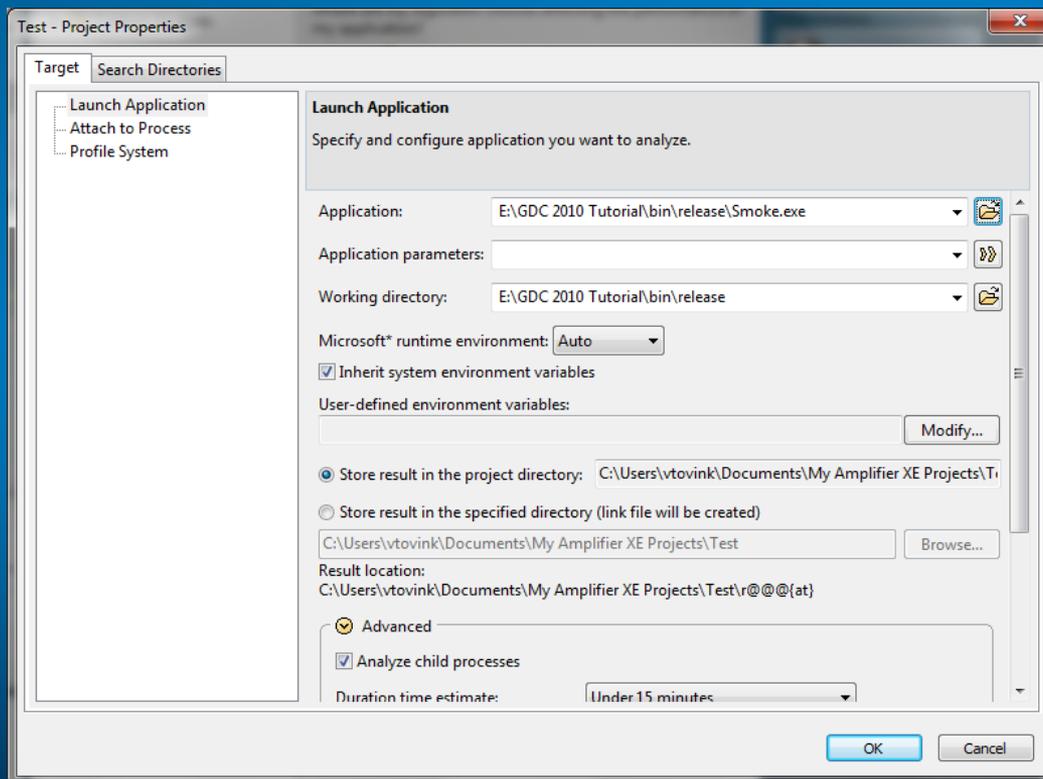
- Select the “New Project” menu attribute



- This opens a dialog window for you to create a *new* project directory
 - Browse to the directory under which you choose to place the new project
 - Give a name for the project in the “Project Directory” field
 - Press the “Create Project” button



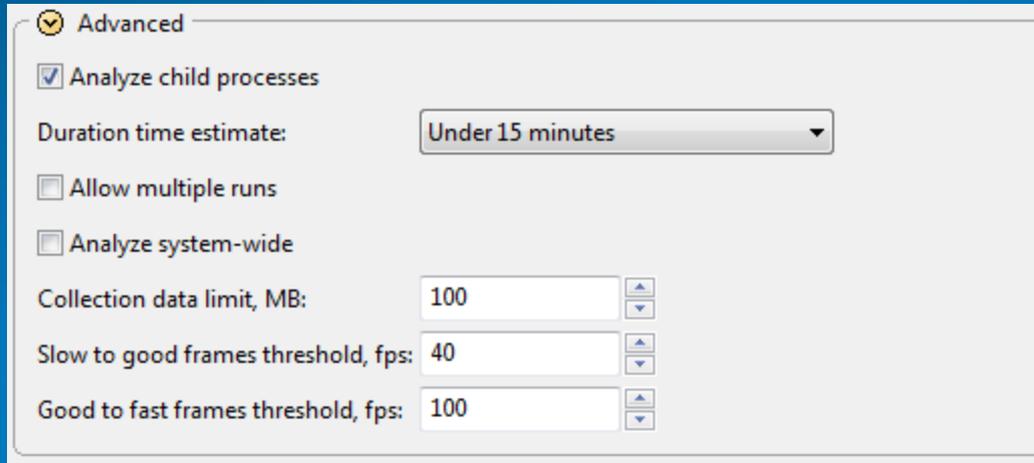
Associating an Application and Workload to the Project



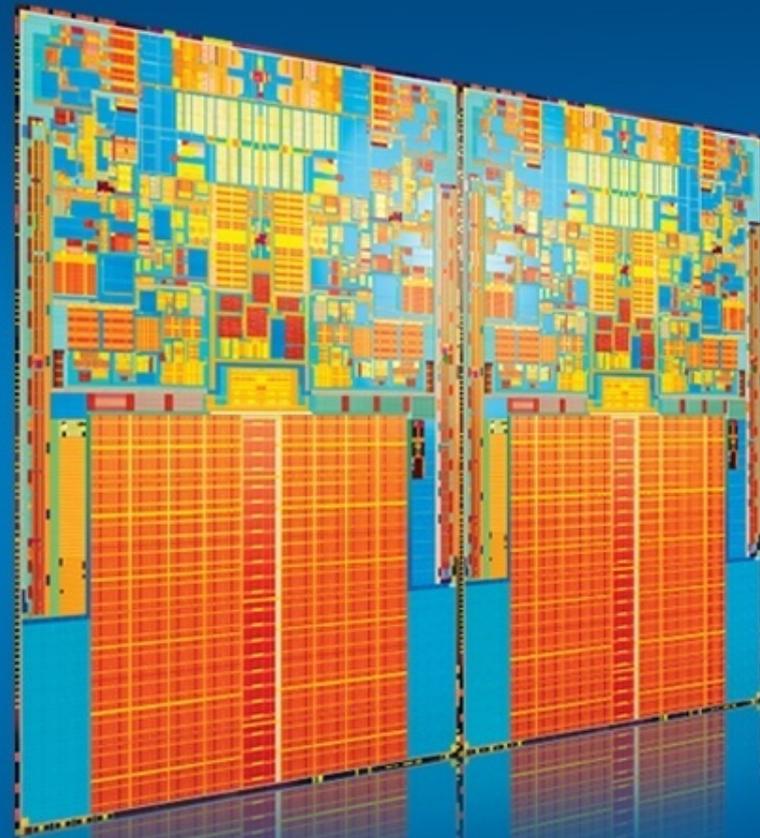
- Select the application
- Specify the command line arguments
- Select the working directory, if different from the parent directory of the application binary
- Select the “Ok” button



Advanced Project Configuration



- At the bottom of the “project Properties” dialog, other knobs are provided
 - User can specify if child process have to followed and analyzed
 - In the case of EBS, checking the “Allow multiple runs” will disable event multiplexing
 - There is also a data collection limit, which defaults to 100MB



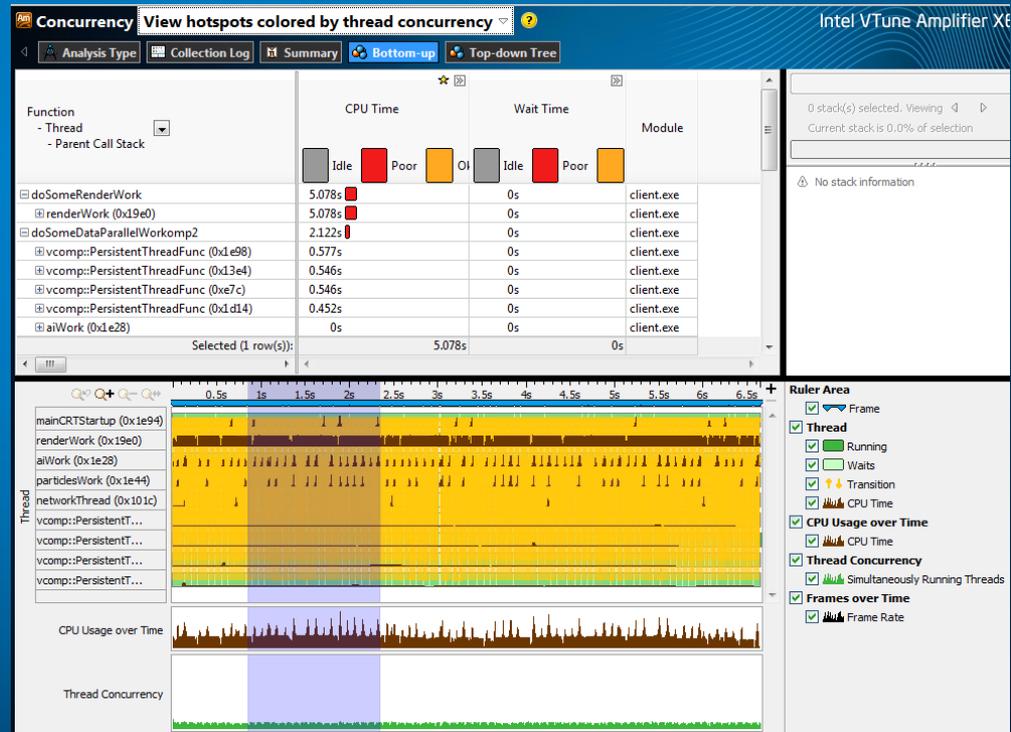
Lab 1

Finding Hotspots

Intel VTune™ Amplifier XE

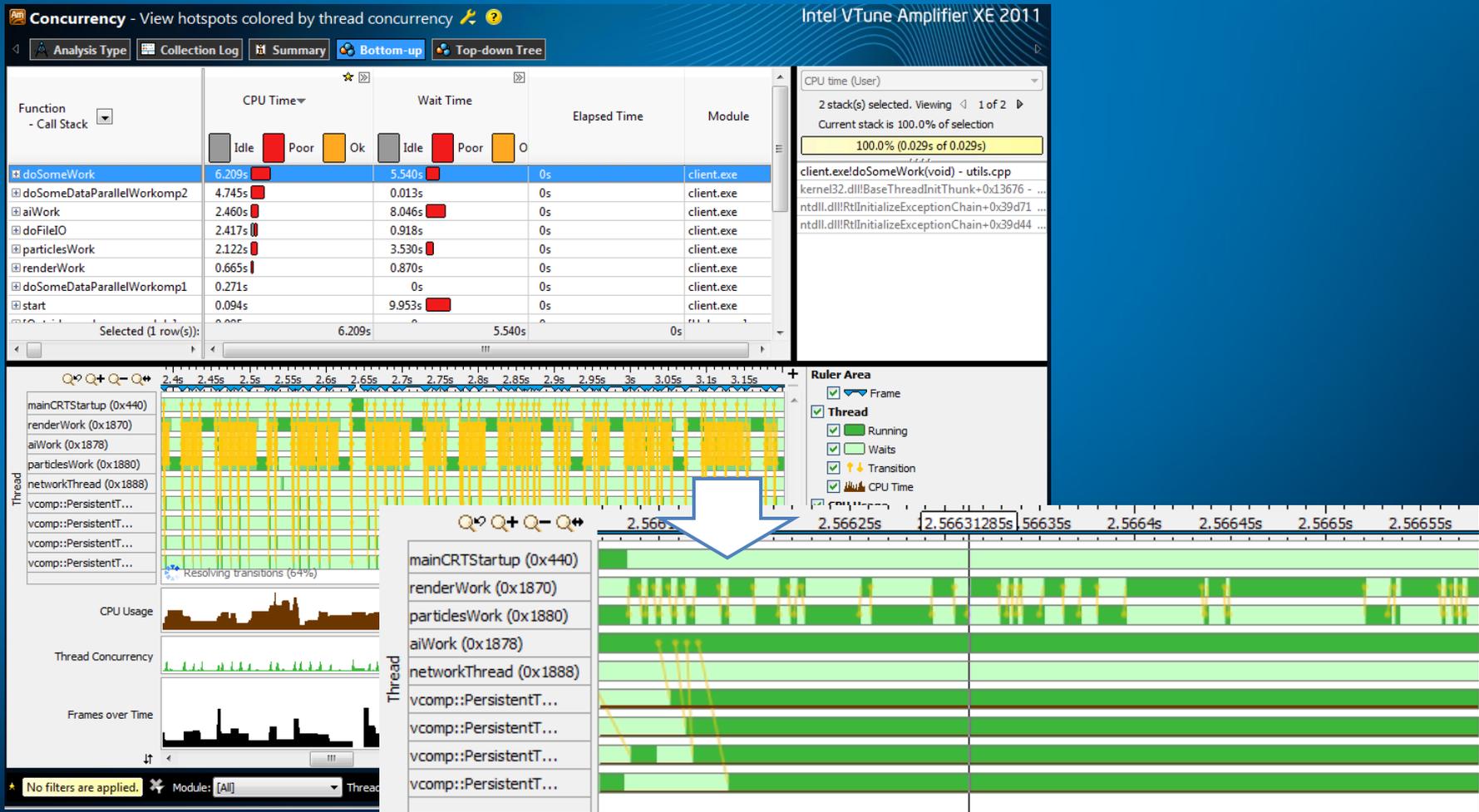
Concurrency Analysis

- For Concurrency analysis,
 - Stack sampling is done just like in Hotspots analysis type
 - Wait functions are instrumented (e.g. WaitForSingleObject, EnterCriticalSection)
 - Signal functions are instrumented (e.g. SetEvent, LeaveCriticalSection)
 - I/O functions are instrumented (e.g. ReadFile, socket)
 - Has the ability to also display utilization based on CPU Usage
 - Has newer metrics such as “overhead” and “spin” time



Intel VTune™ Amplifier XE

Parallelism/Concurrency Analysis



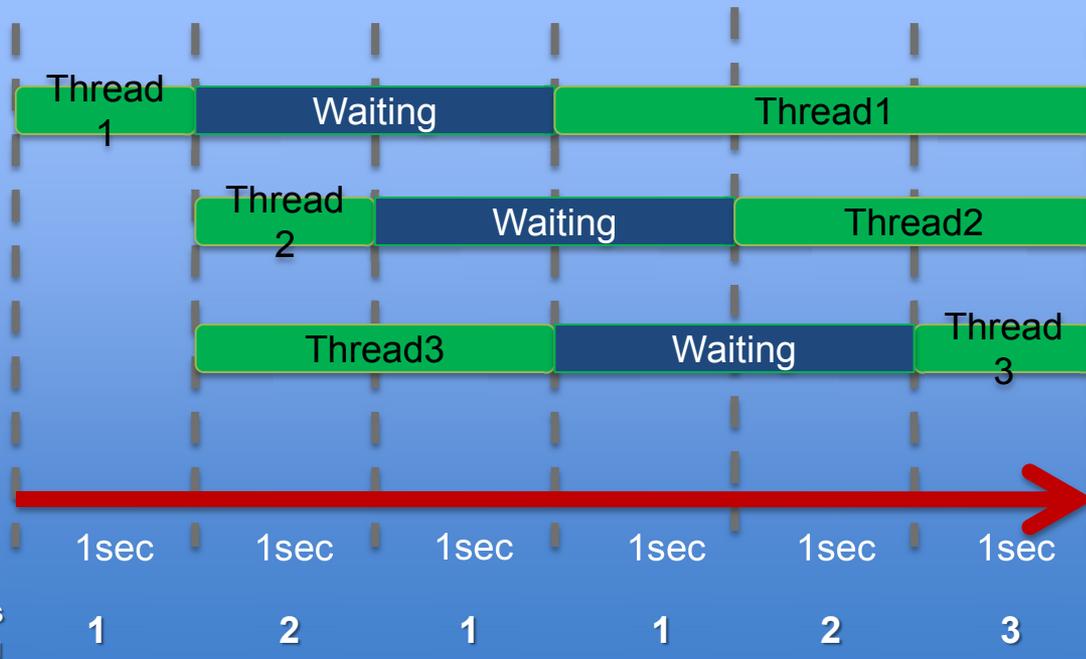
VTune™ Amplifier XE

Terminology used

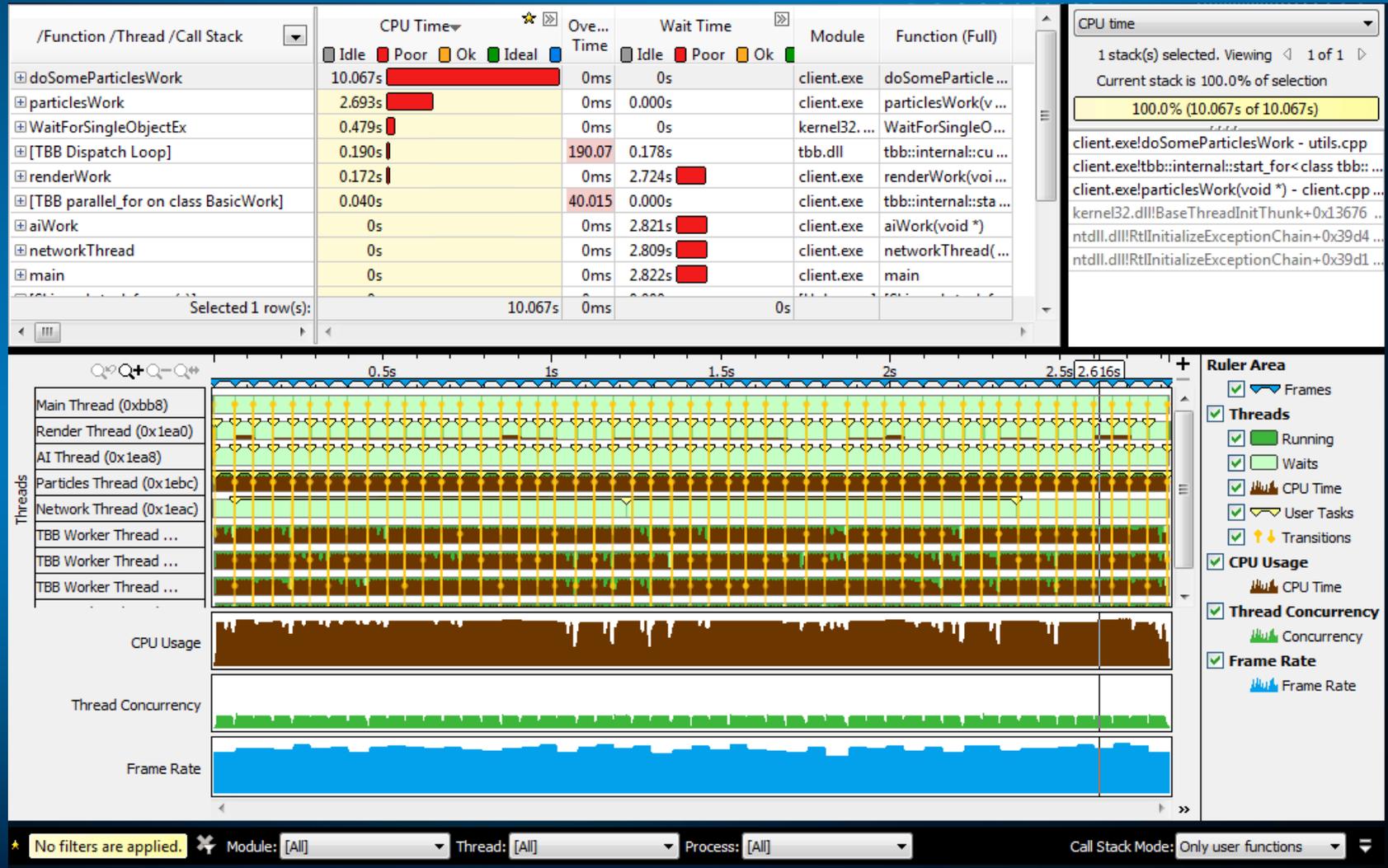
- Concurrency

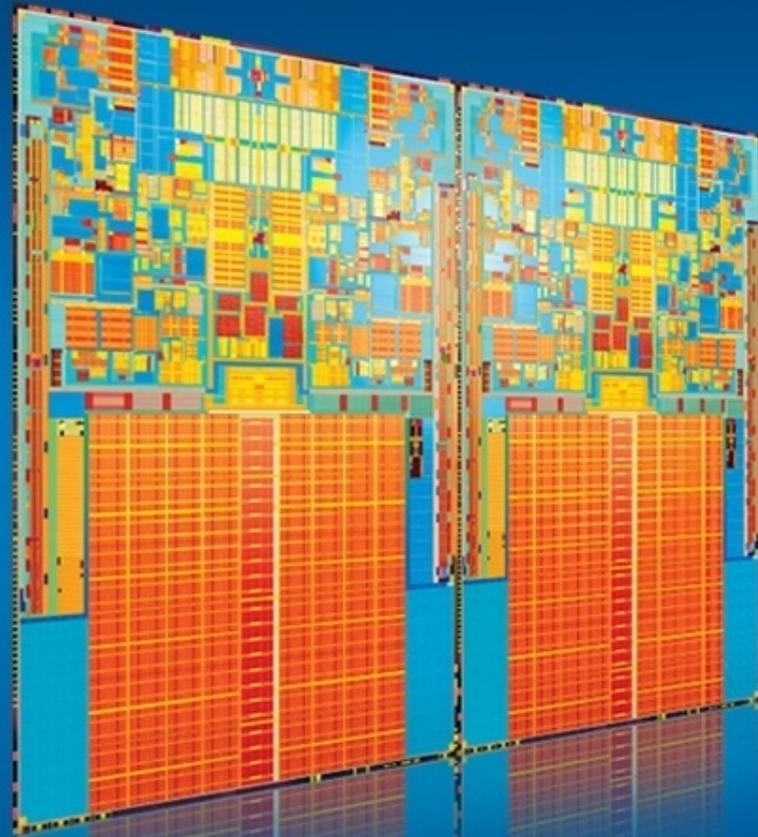
- Is a measurement of the number of active threads

 Thread running  Thread waiting



Sample Concurrency View





Lab 2

Analyzing Concurrency Issues

Analysis Types

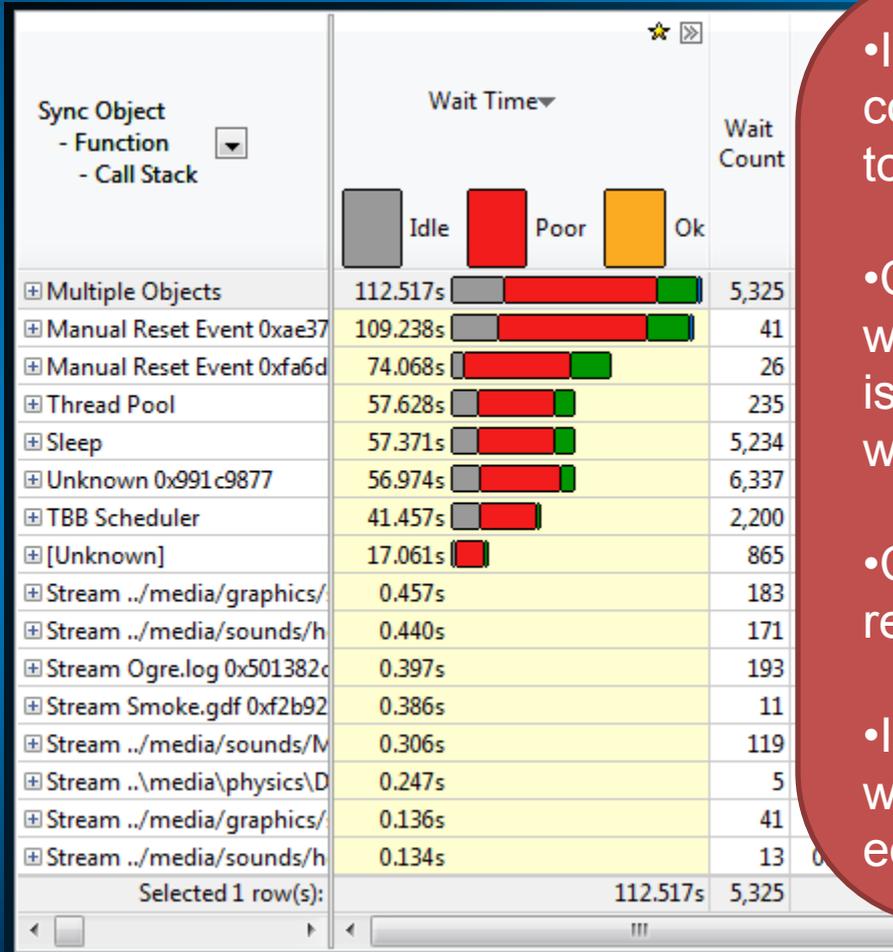
Lock and Waits

- Shows how an application is utilizing available CPU cores
- Helps identify the cause of ineffective utilization,
 - threads waiting too long on synchronization objects (locks),
 - I/O related issues
 - Timers while CPU cores are underutilized
- Overheads are higher than Hotspots analysis type



Analysis Types

Lock and Waits



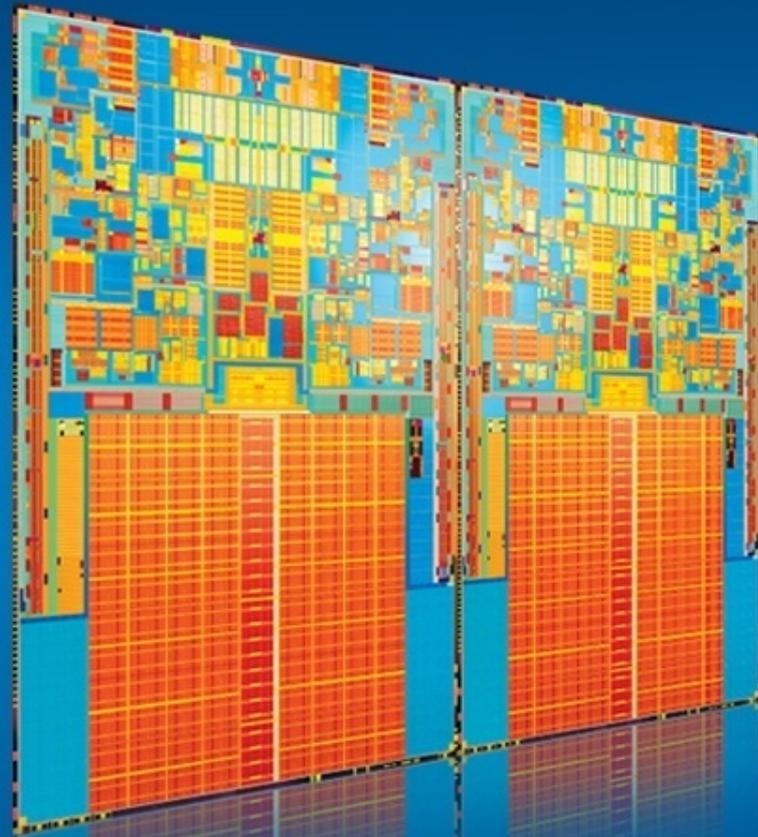
- Identify the objects that caused contention and go to the source code to fix the problem

- Concentrate your tuning on objects with long Wait time where the system is poorly utilized (red bars) during the wait

- Consider adding parallelism, rebalancing, or reducing contention

- Ideal utilization (green bars) occurs when the number of running threads equals the number of available cores





Lab 3

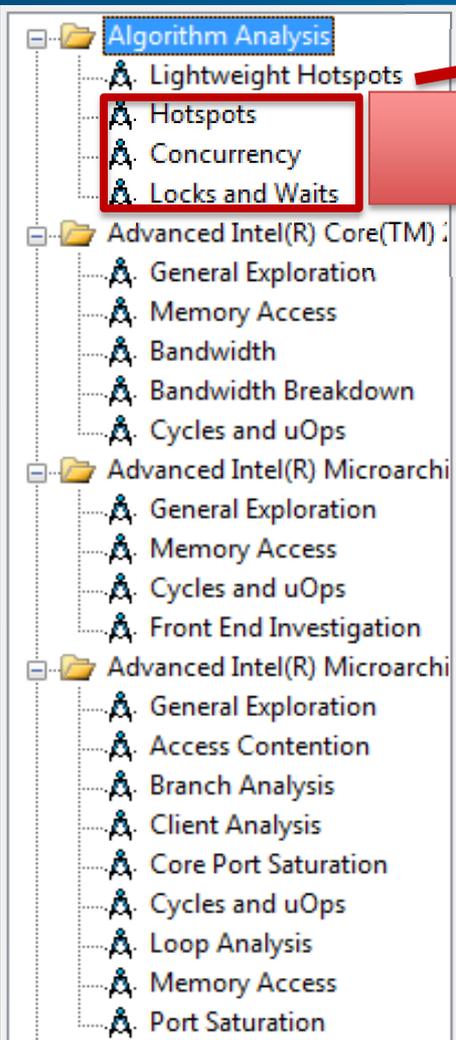
Locks and Waits Analysis

VTune™ Amplifier XE

Pre-defined Analysis Types

Lightweight Hotspot analysis based on the underlying architecture

User mode stack sampling
Threading, IO, Signaling API instrumentation



Analysis Types

Lightweight Hotspots

- New for Amplifier XE
- Similar to Hotspot Analysis
 - Sampling is performed with the SEP collector
 - Driver is required
- Stack walking is not performed
 - Only hotspots are reported
- Samples are taken more frequently, but may have less accurate timing information
- Analysis may be performed for a single application or for the entire system

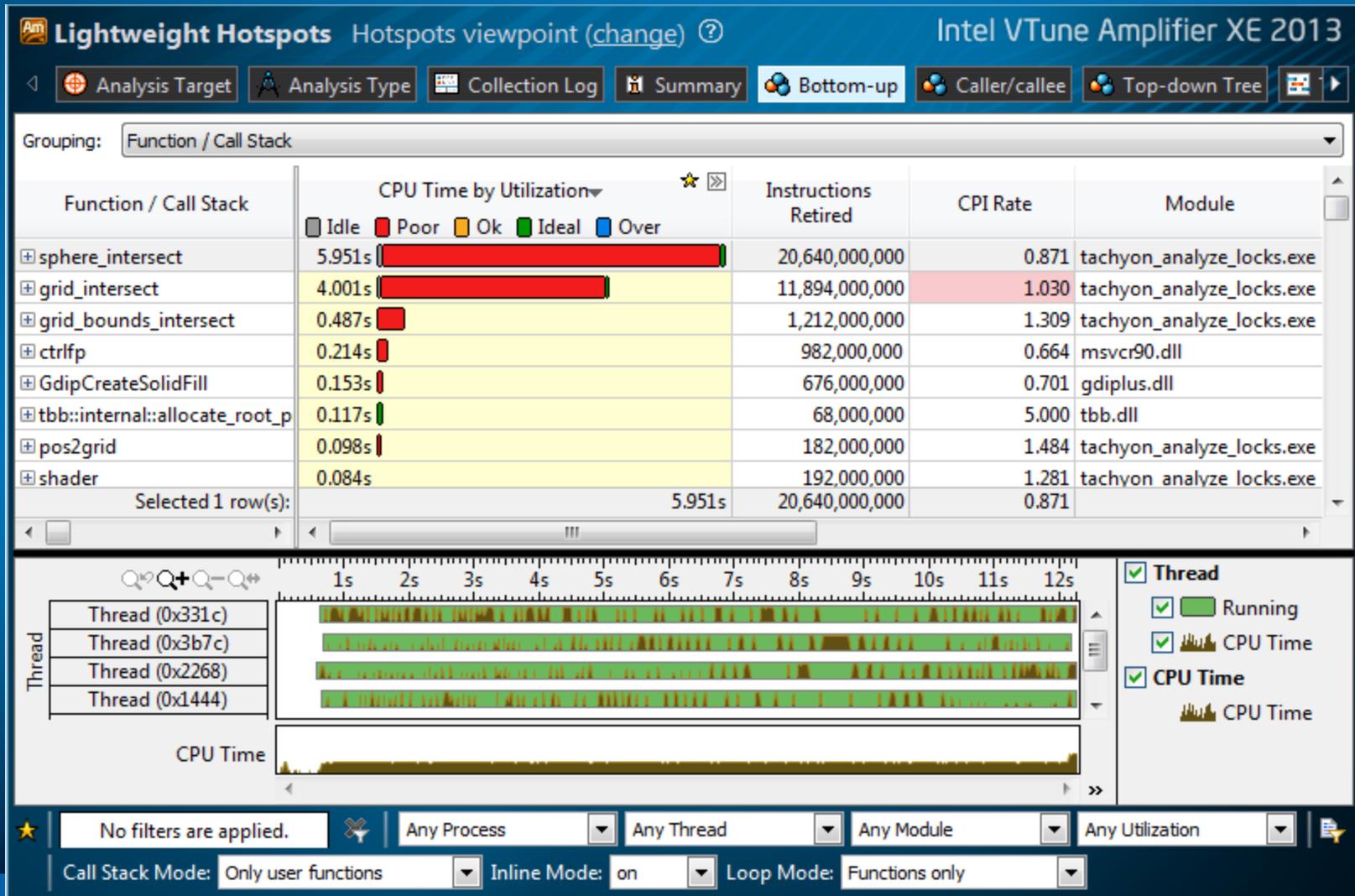


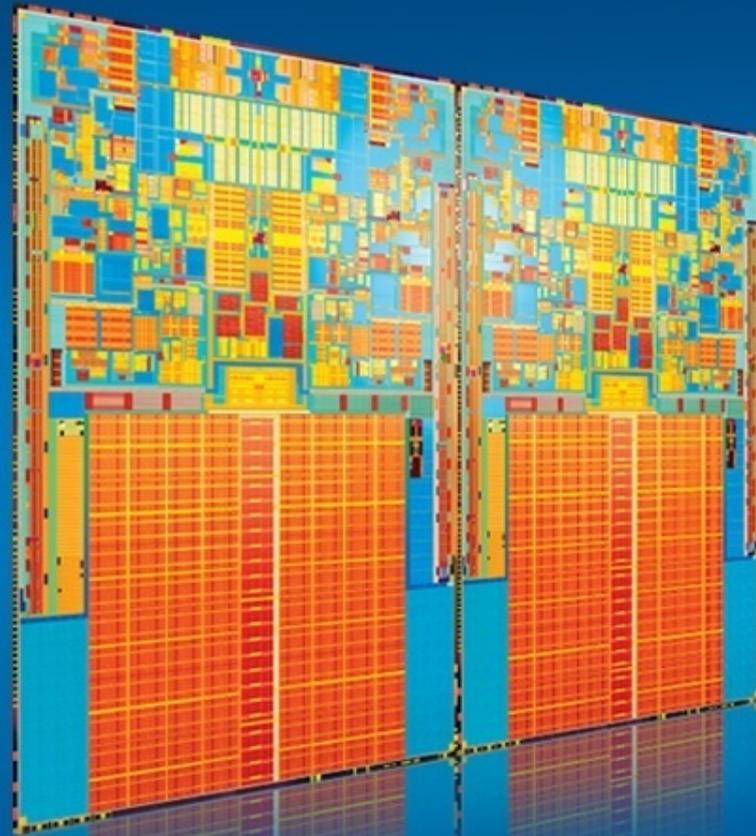
Lightweight Hotspots vs. Hotspots

- Lightweight Hotspots
 - Uses Hardware EBS
 - Lower overhead
 - System wide profiling
 - Faster finalization
 - Requires a driver/privileged mode
- Hotspots
 - Uses software sampling collector
 - Stackwalk is done post process, hence slower finalization times
 - Provides stack information and a statistical call graph



Lightweight Hotspots





Lab 4

Lightweight Hotspots

Intel® VTune™ Amplifier XE

Command Line Interface (CLI)

- **Command Line Interface use cases:**
 - Test code changes for performance regressions
 - Automate execution of performance analyses
- **Command Line Interface features:**
 - Fine-grained control of all analysis types and options
 - Text-based analysis reports
 - Analysis results can be opened in the graphical user interface



Intel® VTune™ Amplifier XE

Command Line Interface

```
$ amplxe-cl -help
```

Intel(R) VTune(TM) Amplifier Command Line Tool

Copyright (C) 2009-2013 Intel Corporation. All rights reserved.

Usage: **amplxe-cl** **<-action>** **[-action-option]** **[-global-option]** **[[--] target [target options]]**

To view the results in the IDE, double-click the `<resultname>.amplxe` file located in the result directory.

Type `amplxe-cl -help <action>` for help on a specific action.

Available actions:

- collect**
- collect-with**
- command**
- finalize**
- help**
- import**
- report**
- version**



Intel® VTune™ Amplifier XE

Command Line Interface Examples

- Display a list of available analysis types and preset configuration levels

```
amplxe-cl -help collect
```

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

```
amplxe-cl -collect hotspots myApp
```

- Run the Parallelism analysis, store the result in directory *r001par*

```
amplxe-cl -collect concurrency -result-dir r001par myApp
```



Intel® VTune™ Amplifier XE

Command Line Interface - reporting

Examples

Generate the 'hotspots' report for the result directory 'r000hs'.

```
amplxe-cl -report hotspots -r r000hs
```

Group the data by module.

```
amplxe-cl -report hotspots -r r000hs -group-by module
```

Filter the output.

```
amplxe-cl -report hotspots -r r000hs -filter module=libexample.so
```

Use '-help report <report name>' for more information about each report.

Available reports

callstacks	Display CPU or wait time for callstacks.
gprof-cc	Display CPU or wait time in the gprof-like format.
hotspots	Display CPU time.
hw-events	Display hardware events.
perf	Display CPU or wait time for each module.
perf-detail	Display CPU time for each function or synchronization object.
pmu-events	
sfdump	Specialized report to display hardware events.
summary	Display data about overall performance.
top-down	Display a call tree for your target application and provide CPU and wait time for each function.
wait-time	Display wait time.



Summary

- VTune™ Amplifier XE is the coming together of distinct tools
 - Correlation of the data from various capabilities makes it very powerful
 - Supports EBS data collection with better usability
 - Supports statistical call-graph capability
 - Combines these two types of capabilities with powerful Thread Profiling capability
- VTune™ Amplifier XE provides a standalone GUI with the same look and feel on both Windows* and Linux
- Complete re-design to make it a very extensible tool





Optimization Notice

Optimization Notice

Intel compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel compilers, associated libraries and associated development tools 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 Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (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.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20110307



Legal Disclaimer

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.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

Copyright ©2010. Intel Corporation.

<http://www.intel.com/software/products>

