Intel® VTune™ Amplifier XE 2013

Optimization Notice

Intel® VTune™ Amplifier XE 2013

Second Generation VTune™ Analyzer

Fast, Accurate Performance Profiles

- Hotspot (Statistical call tree)
- Hardware-Event Based Sampling[1]

Thread Profiling

- Visualize thread interactions on timeline
- Balance workloads

Easy set-up

- Pre-defined performance profiles
- Use a normal production build

Compatible

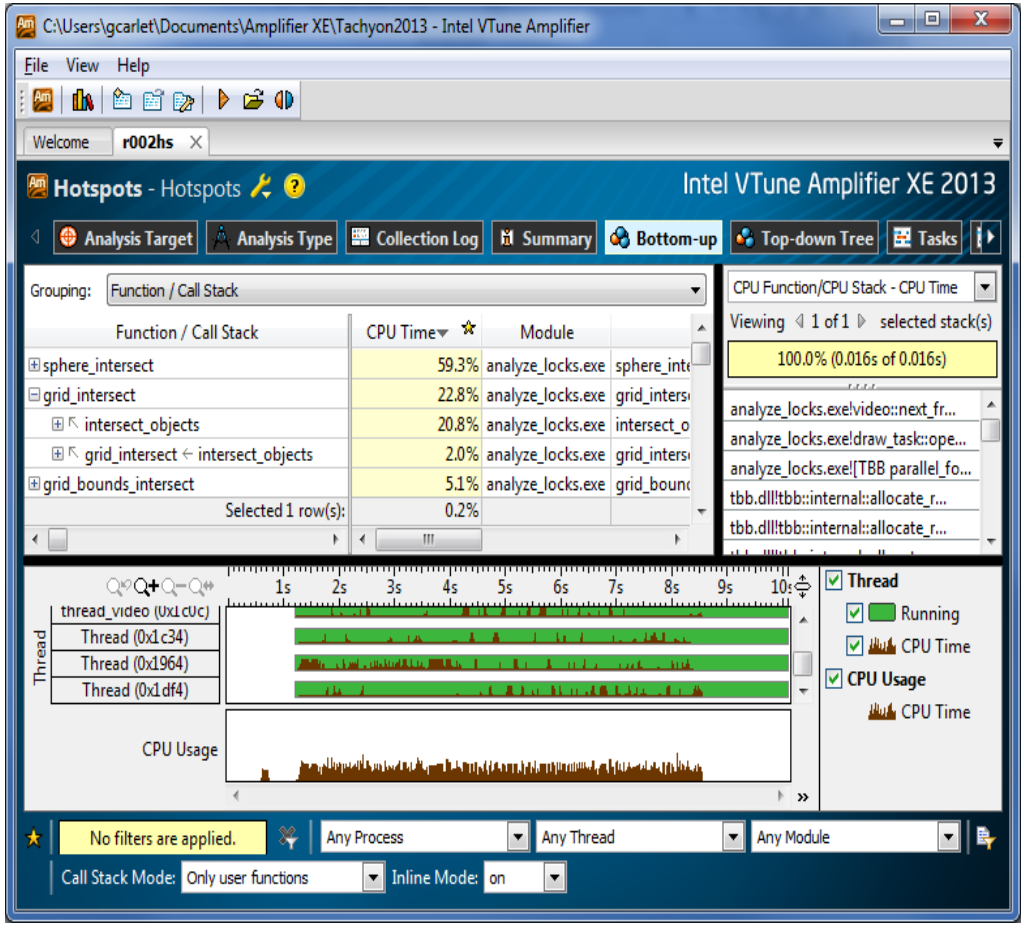
- Microsoft, GCC, Intel compilers
- C/C++, Fortran, Assembly, .NET, Java*
- Latest Intel® processors and compatible processors[1]

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

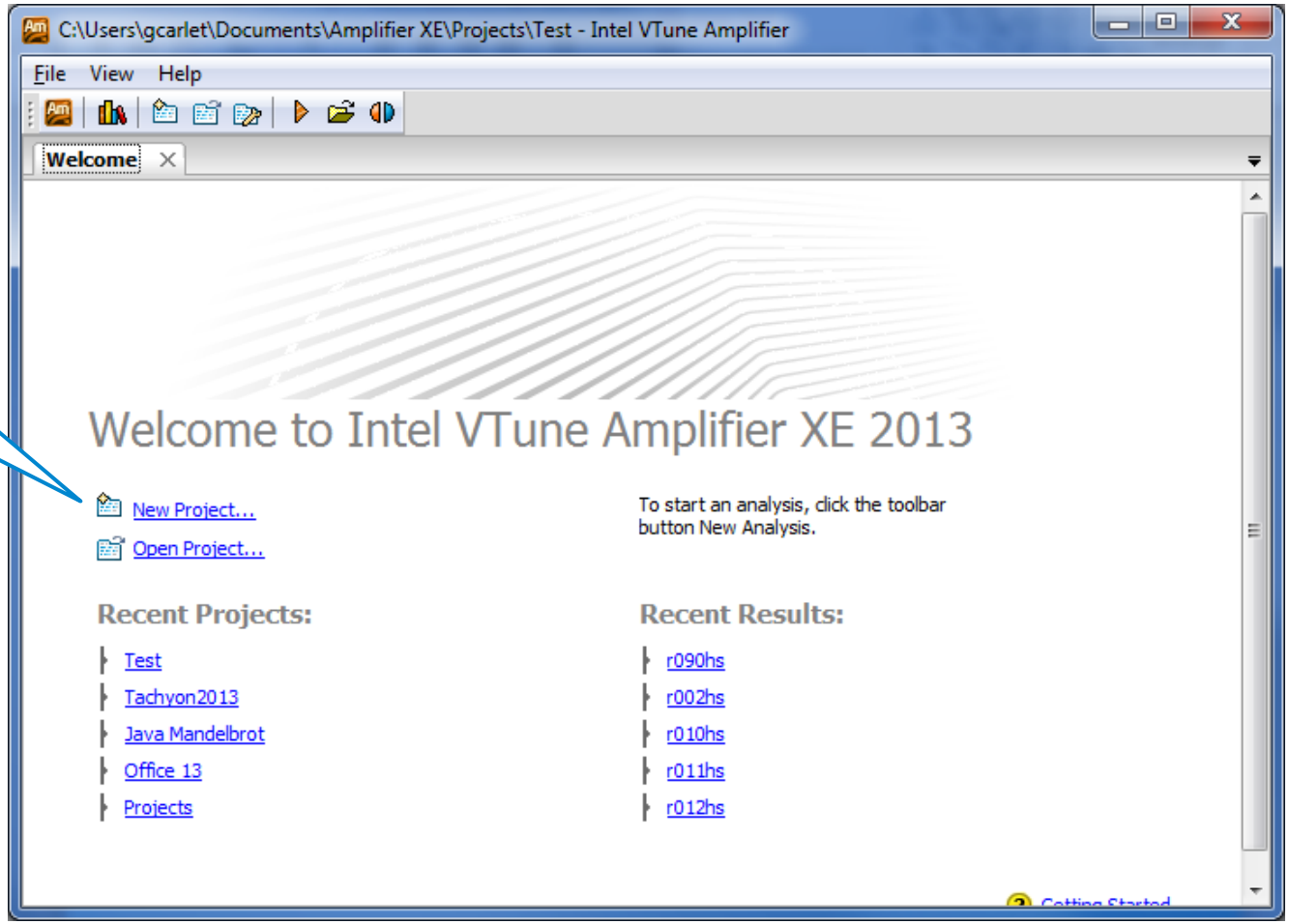


[1] IA32 and Intel® 64 architectures. Many features work with compatible processors.
Event based sampling requires a genuine Intel® Processor.

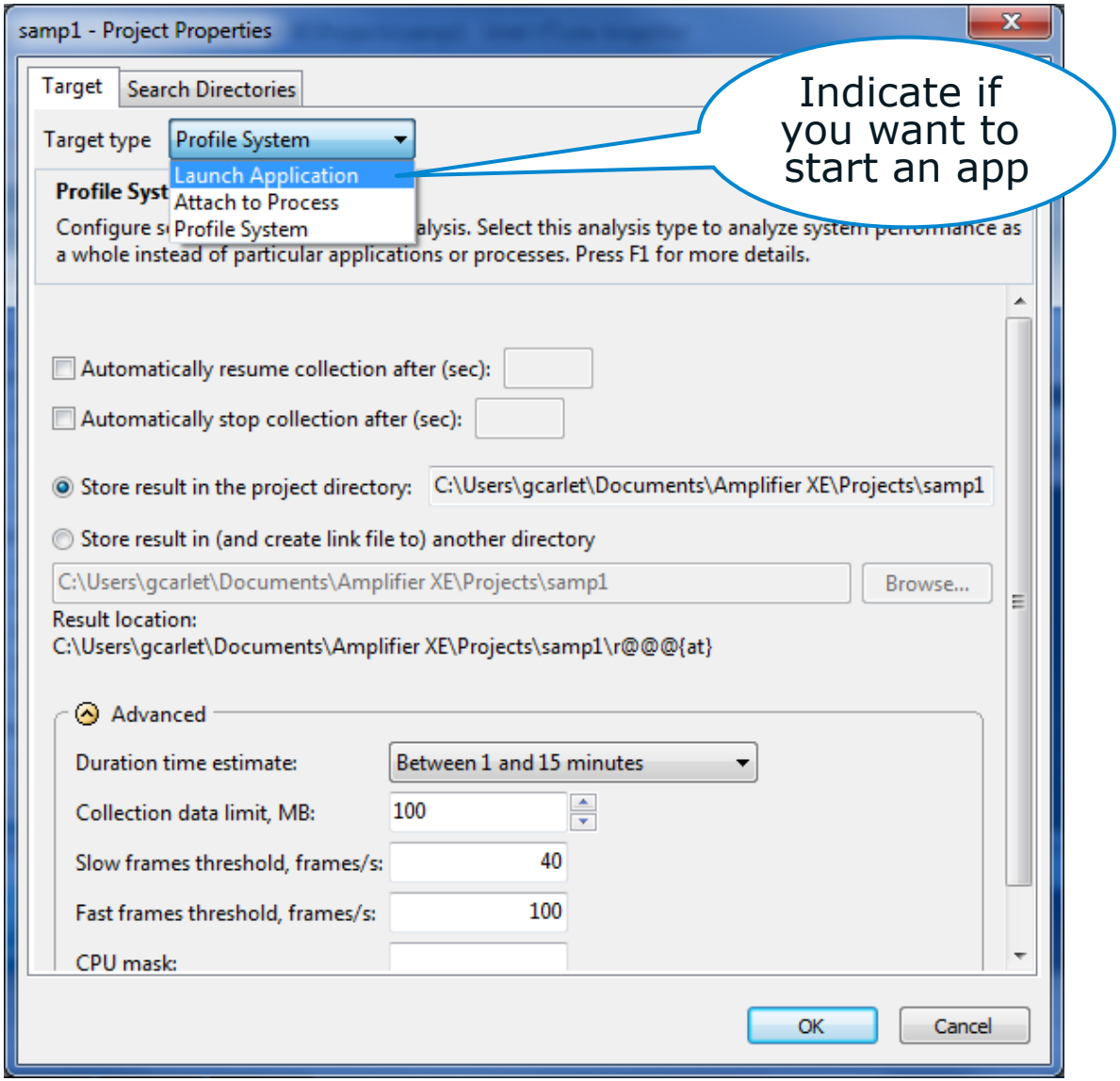
Optimization Notice

Starting VTune™ Amplifier XE - The First Time

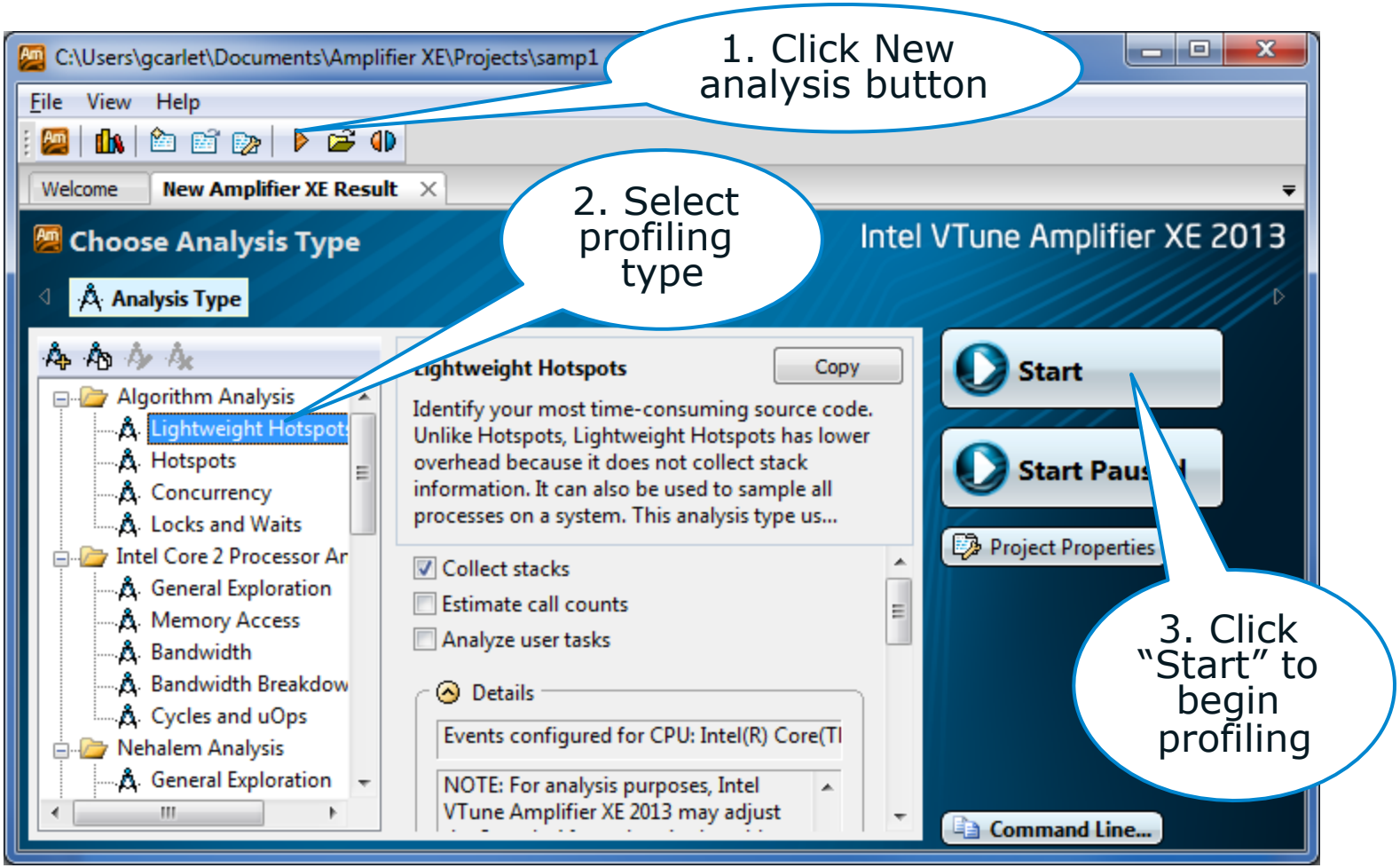
First create a project



Specify optional app to launch



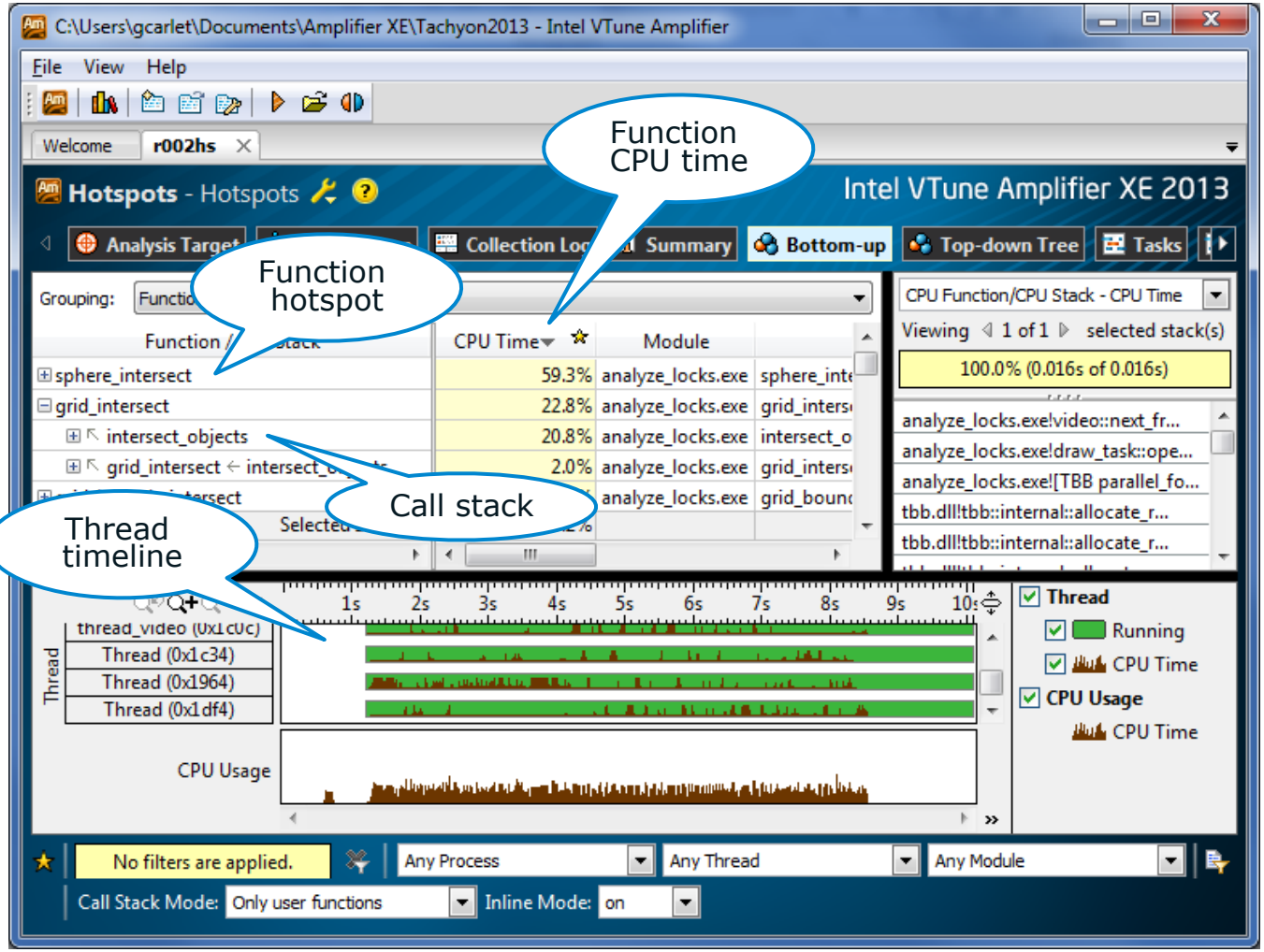
Indicate type of profiling (ex: Lightweight Hotspots)



Optimization Notice

Demo: Hotspot Collector

Hotspots analysis



Hotspots analysis – Source View

C:\Users\gcarlet\Documents\Amplifier XE\Tachyon2013 - Intel VTune Amplifier

File View Help

Welcome | r002hs

Hotspots - Hotspots — Intel VTune Amplifier XE 2013

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

Source | Assembly

Sour...	Source	CPU Time
578	else if (tmax.z < tmax.y) {	0.0%
579	cur = g->cells[voxindex];	1.2%
580	while (cur != NULL) {	0.0%
581	if (ry->mbox[cur->obj->id] != ry->serial) {	5.9%
582	ry->mbox[cur->obj->id] = ry->serial;	5.5%
	Selected 1 row(s):	5.9%

CPU Function/CPU Stack

Viewing 1 of 23 select

29.3% (0.516s of 1.765s)

analyze_locks.exe!grid...
analyze_locks.exe!inte...
analyze_locks.exe!sha...
analyze_locks.exe!trac...
analyze_locks.exe!ren...

1s 2s 3s 4s 5s 6s 7s 8s 9s 10s

Thread: thread_video (0x1c0c), Thread (0x1c34), Thread (0x1964), Thread (0x1df4)

CPU Usage

Thread: Running, CPU Time

CPU Usage: CPU Time

No filters are applied. | Any Process | Any Thread | Any Module

Call Stack Mode: Only user functions | Inline Mode: on

Types of Performance Data Collection

Hardware Event-based Sampling Analysis

- Collection Types: Lightweight Hotspots, Advanced Processor Analysis
- System wide analysis
  - Kernel mode code, device drivers, OS, …
- Measures cache misses, branch mispredictions, …
- Uses the Performance Monitoring Unit of each Intel CPU Core.

User Mode Sampling and Tracing Analysis

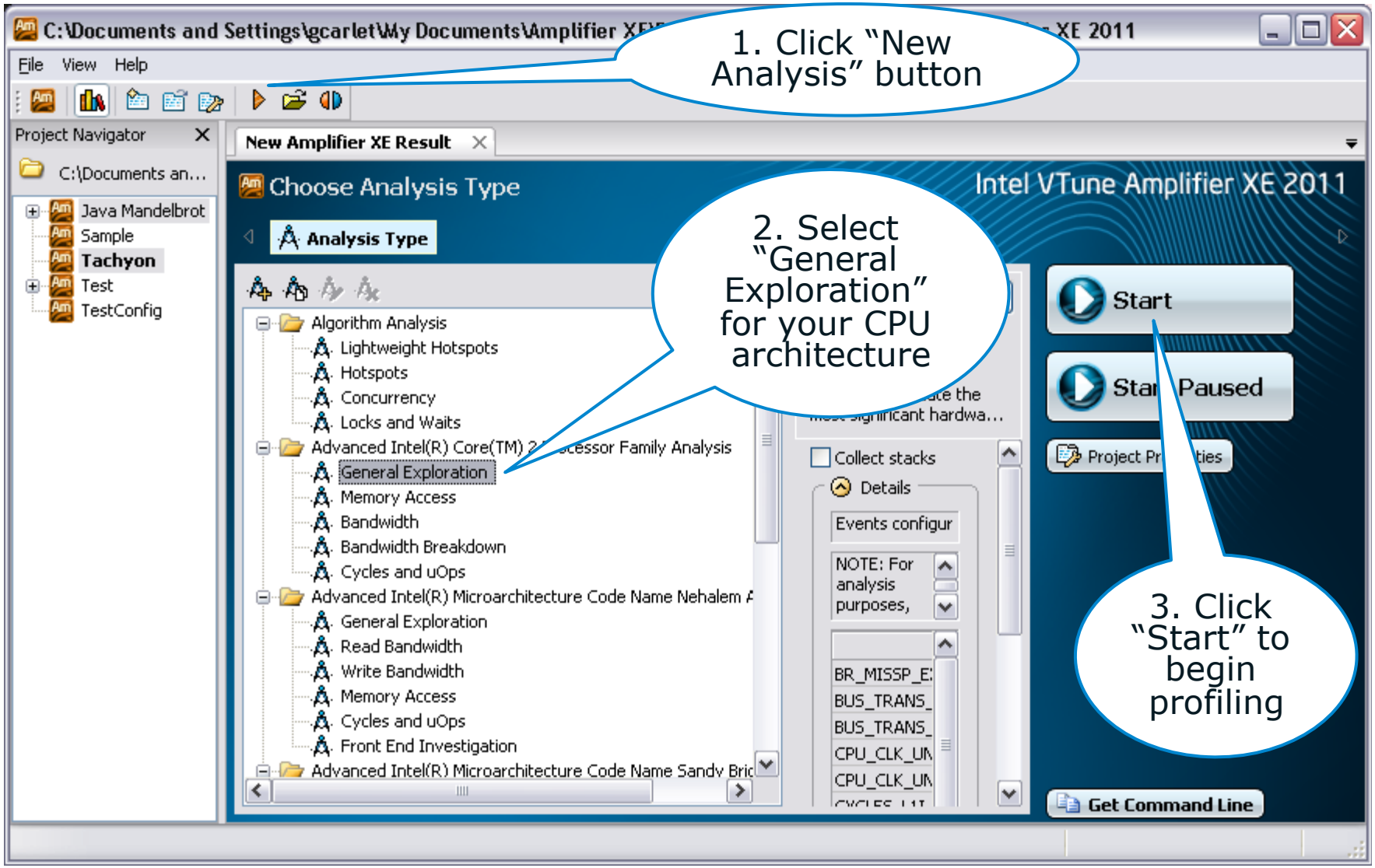
- Collection Types: Hotspots, Concurrency, & Lock and Waits
  - Dynamically instruments binary
    - Minimal = Hotspots
    - More = Concurrency, & Locks and Waits
- 1 process only
- User mode SW only
- Uses OS Timer Service for each thread to collects a sample

Advanced Processor Analysis – General Exploration

- Predefined Analysis Type that collects different types of CPU performance events
- Good for first look at whether any CPU event categories are affecting performance
- GUI highlights those events and functions that have performance problems

Demo – General Exploration Collector

Running the General Exploration collector



Optimization Notice

CPU HW Sampling results

tachyon_vtune_amp_xe - Microsoft Visual Studio

File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help

Release | Win32

r008ge | r000hs | New Amplifier XE Result | Solution Explorer

General Exploration - Hardware Issues

Analysis Target | Analysis Type | Summary | Bottom-up

/Function	PMU Event Count CPU_CLK_...	PMU Event Count INST_RETIRED....	CPI	Retire Stalls	LLC Miss	LLC Load ...	Conte... Acces...	Instru... Starva...	Branch Mispr...	Execut... Stalls	Data Sharing
initialize_2D_buffer	42,564,000,000	63,586,000,000	0.669	0.530	0.000	0.000	0.000	0.062	0.021	0.147	0.000
sphere_intersect	20,652,000,000	19,174,000,000	1.077	0.815	0.000	0.000	0.000	0.059	0.045	0.179	0.000
grid_intersect	11,816,000,000	8,086,000,000	1.461	0.847	0.015	0.000	0.000	0.273	0.221	0.227	0.000
grid_bounds_intersect	1,700,000,000	994,000,000	1.710	0.688	0.000	0.000	0.000	0.444	0.122	0.291	0.000
GdipCreateSolidFill	528,000,000	866,000,000	0.610	0.295	0.000	0.000	0.000	0.000	0.015	0.197	0.000
shader	302,000,000	184,000,000	1.641	0.603	0.000	0.000	0.000	0.000	0.013	0.000	0.000
Selected 1 row(s):	90,000,000	0									

Threads: Thread (0x33c), Thread (0x9b0), Thread (0x1458)

Hardware

Threads | Running | Hardware ... | Hardware Events | Hardware ...

No filters are appli

Output

Ready

Performance problems are highlighted

Hovering the mouse over a highlighted problem displays a tooltip with a problem definition and high level suggestions for fixes or analysis next steps

Intel® VTune™ Amplifier XE
Frame Analysis

Frame: a region executed repeatedly (non-overlapping).

API marks start and finish

Examples:

- Game – Compute next graphics frame
- Database – Query response time
- Computation – Convergence loop

Application

```
voidalgorithm_1();
voidalgorithm_2(int myid);
doubleGetSeconds();
DWORD WINAPI do_xform (void * lpmyid);
bool checkResults();
__itt_frame =
__itt_frame_createA("myDomain");
```

Region (Frame)

```
while( gRunning ) {
   __itt_frame_begin(itt_frame);
   . . .
  //Do Work
   . . .
   __itt_frame_end(itt_frame);
}

for (int k = 0; k < N; ++k) {
int ik = i*N + k;
int kj = k*N + j;
c2[ij] += a[ik]*b[kj];
}
```


Optimization Notice

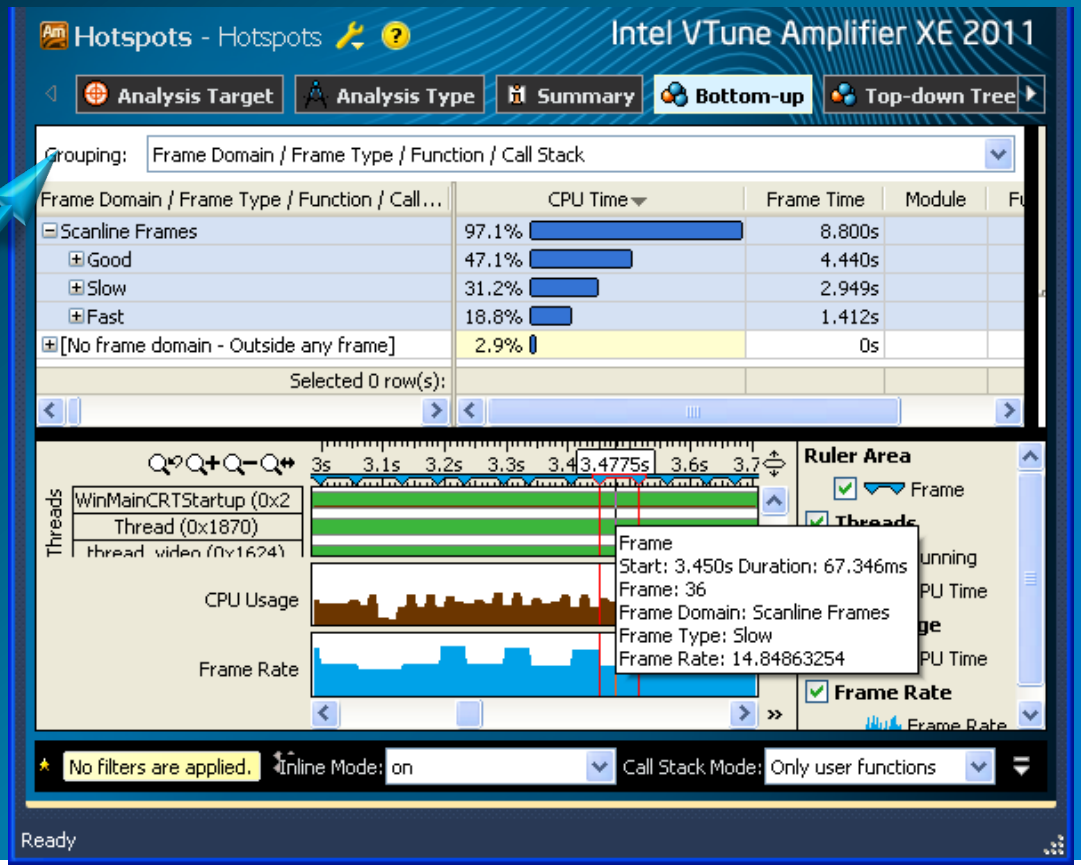
Intel® VTune™ Amplifier XE
Find Slow Frames With One Click

(1) Regroup Data

- Function - Call Stack
- Module - Function - Call Stack
- Source File - Function - Call Stack
- Thread - Function - Call Stack
- Function - Thread - Call Stack
- OpenMP Region - Function - Call Stack
- Task Type - Function - Call Stack
- Frame Domain - Frame - Function - Call Stack
- Frame Domain - Frame Type - Function - Call Stack

... (Partial list shown)

Result:



Optimization Notice

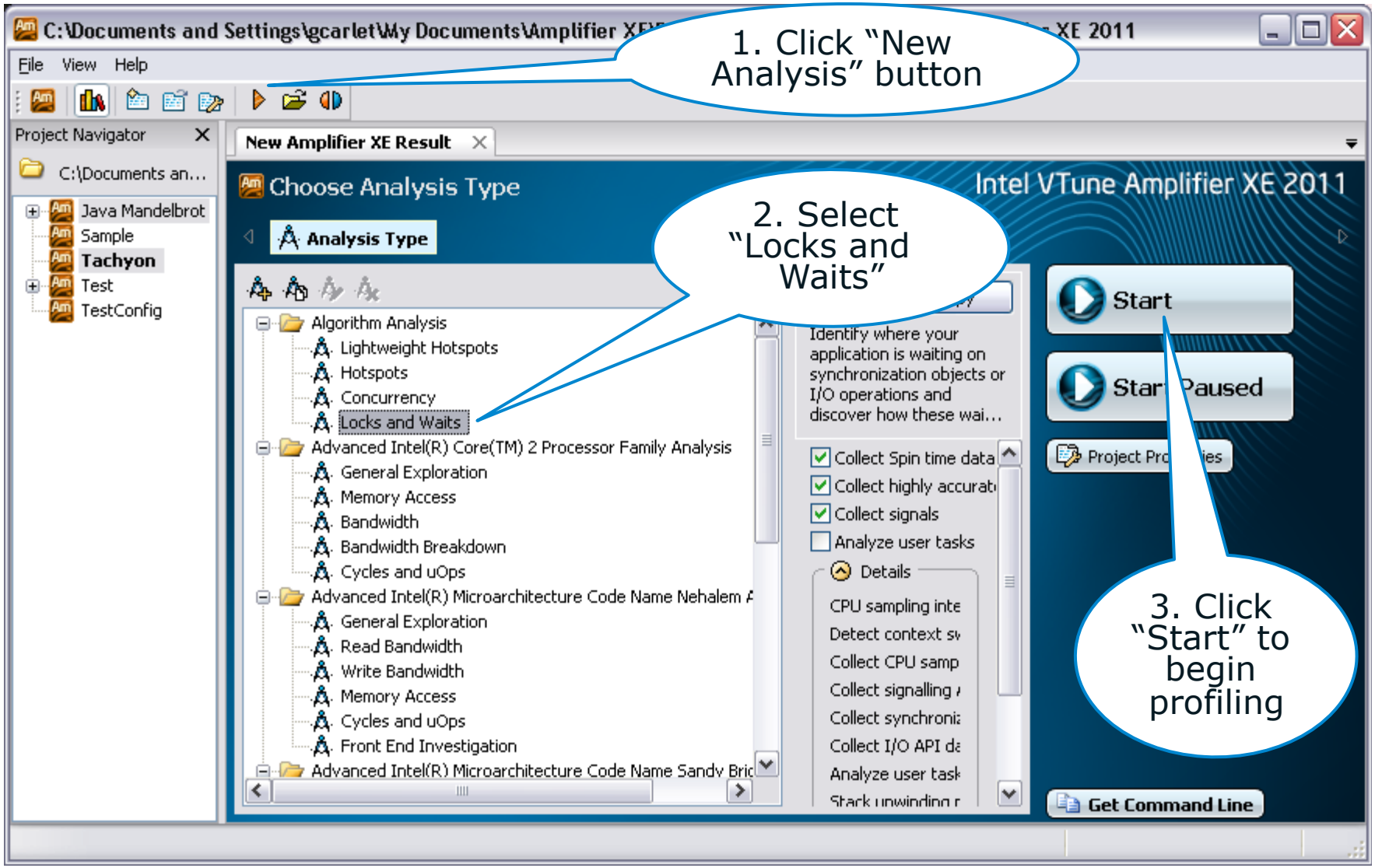
Locks and Waits Collection

Identifies those threading items that are causing the most thread block time

- Synchronization locks
- Threading APIs
- I/O

Demo:
Locks-And-Waits Collector

Running the "Locks and Waits" collector



Locks-and-Waits View

/home/vtune/intel/My Amplifier XE Projects/My Amplifier XE Project - Intel VTune Amplifier XE 2011

File Project Help

r006lw

Locks and Waits - Locks and Waits

Intel VTune Amplifier XE 2011

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

Sync Object - Function - Call Stack	Wait Time (Idle / Poor / Ok)	Wait Cou..	Module	Object Type	Object Creation F
Mutex 0x3de2bb60	25.316s	492	[Unknown]	Mutex	thread_trace
draw_task::operator(	25.316s	492	tachyon_...	Mutex	[Unknown]
TBB Scheduler	10.735s	1	[Unknown]	Constant	tbb::parallel_for<tbb::blocked_ra
Stream 0xc287898b	9.950s	828	[Unknown]	Stream	tbb::parallel_for<tbb::blocked_ra
Stream 0xbbb17798	0.664s	327	[Unknown]	Stream	[libX11.so.6.2]
Socket 0x7b5836b3	0.013s	134	[Unknown]	Socket	[libX11.so.6.2]
Mutex 0x76509717	0.010s	3	[Unknown]	Mutex	drawing_area::~drawing_area
Selected (1 row(s)):	25.316s	492			

Object Creation (Us

1 stack(s) selected. Viewing 1 of 1

Current stack is 100.0% of selection

100.0% (0.014s of 0.014s)

tachyon_analyze_locks!thread_trace(thr...
tachyon_analyze_locks!trace_shm(scen...
tachyon_analyze_locks!trace_region(sc...
tachyon_analyze_locks!renderscene(sc...
tachyon_analyze_locks!rt_renderscene(...
tachyon_analyze_locks!tachyon_video::...
tachyon_analyze_locks!video::main_loo...
tachyon_analyze_locks!main - video.cp...
tachyon_analyze_locks!_start+0x1f30 - ...

5s 10s 15s 20s 25s 30s 35s

Thread (0x7fffffff)
tbb::internal::rml:...
CPU Usage
Thread Concurr...

Thread: Running, Waits, Transition, CPU Time
CPU Usage: CPU Time
Thread Concurrency: Concurrency

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

Ready

Locks-and-Waits Source View

/home/vtune/intel/My Amplifier XE Projects/My Amplifier XE Project - Intel VTune Amplifier XE 2011

File Project Help

r006lw

Locks and Waits - Locks and Waits

Intel VTune Amplifier XE 2011

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

Line	Source	Wait Time (Idle / Poor / Ok)	Address	Assembly
162	drawing_area drawing(startx, totaly-y, stopx		0x323a	calll 0x804ae00 <.
163			0x323f	**Block 5:**
164	// Acquire mutex to protect pixel calculation		0x323f	add $0x14, %esp
165	pthread_mutex_lock (&rgb_mutex);	25.316s	0x3242	pushl $0x805d544
166	for (int x = startx; x < stopx; x++) {		0x3247	calll 0x8049b10 <.
167	color_t c = render_one_pixel (x, y, local_r		0x324c	**Block 6:**
168	drawing.put_pixel(c);		0x324c	movl 0x805d574,
169	}		0x3252	add $0x10, %esp
	Selected (1 row(s)):	25.316s		Highlighted 2 row(s):

Object Creation (U

1 stack(s) selec

Current stack is

100.0% (0.014s o

tachyon_analyze_l

Thread (0x7fffffff)

tbb::internal::rml:...

CPU Usage

Thread Concurr...

Thread: Running, Waits, Transition, CPU Time

CPU Usage: CPU Time

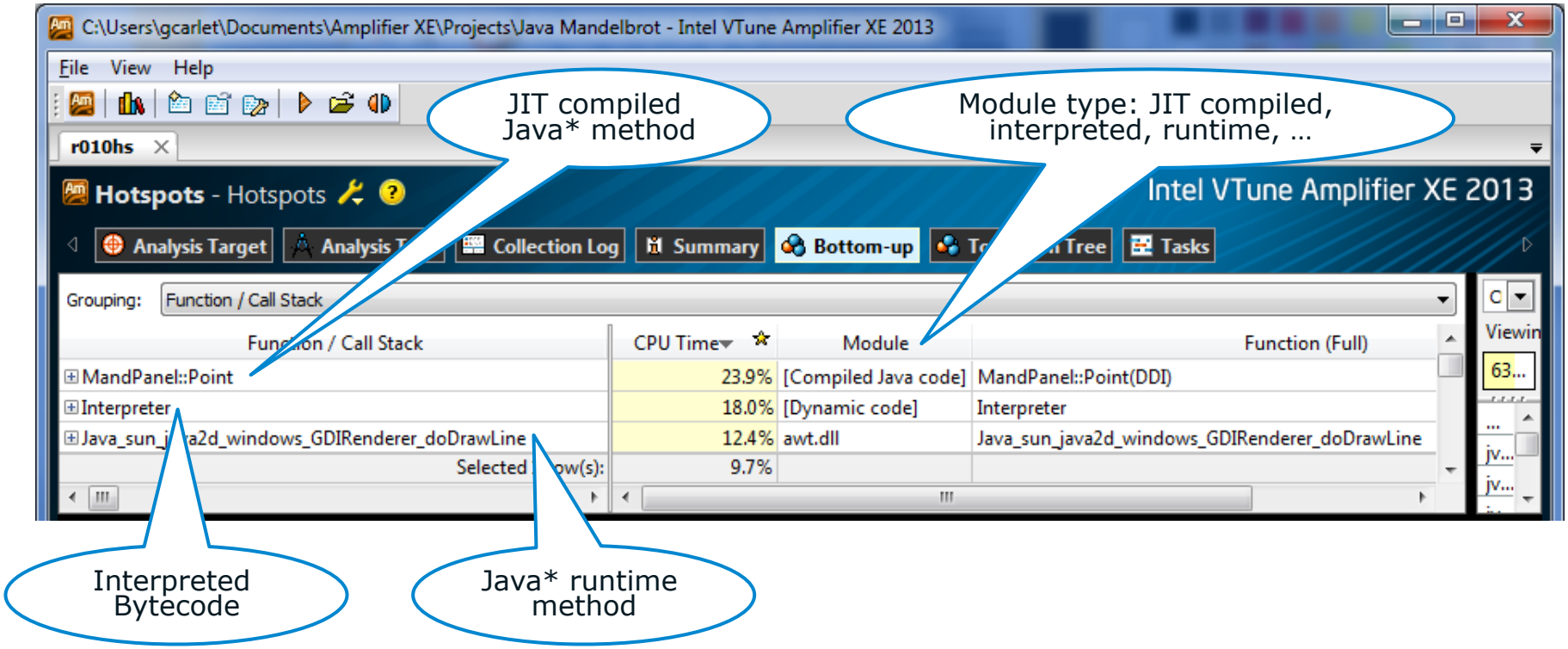
Thread Concurrency: Concurrency

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

Ready

Java* Support

- Analyze CPU usage and HW events (cache misses, …)
- Drill down to Java source code
- See how much time the Java Interpreter is consuming
- Mixed Mode Profiling: Java* and C/C++ code



Optimization Notice

Intel® VTune™ Amplifier XE 2013

Power profiling

- Profile sleep state and dynamic frequency speeds
- Identify causes of wake-ups
- See sleep state and frequency transitions in timeline view
- Determine best BIOS power settings
- Currently Linux* only

Intel® Xeon Phi™ Coprocessor support

- Hardware CPU event profiling (CPU usage, cache misses,…)
- Correlate data events across multiple cards

Intel® VTune™ Amplifier XE 2013

Analyze User Tasks via APIs

- Timeline is marked with start and stop times of your tasks
- Tasks can correspond to functions
- Supported by all collection modes
- Can be Nested

User Defined Performance Metrics

- Define new columns in sampling results displays
- Define formulas based on sampling counters
- Define tooltip text/descriptions for the new metric also
- Implemented using Python scripts
- See help under "user-defined metrics"

Summary

- The Intel® VTune Amplifier XE can be used to find:
  - Source code for performance bottlenecks
  - Characterize the amount of parallelism in an application
  - Determine which synchronization locks or APIs are limiting the parallelism in an application
  - Easily find CPU performance events that are causing additional CPU clocks

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 (Intel 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.

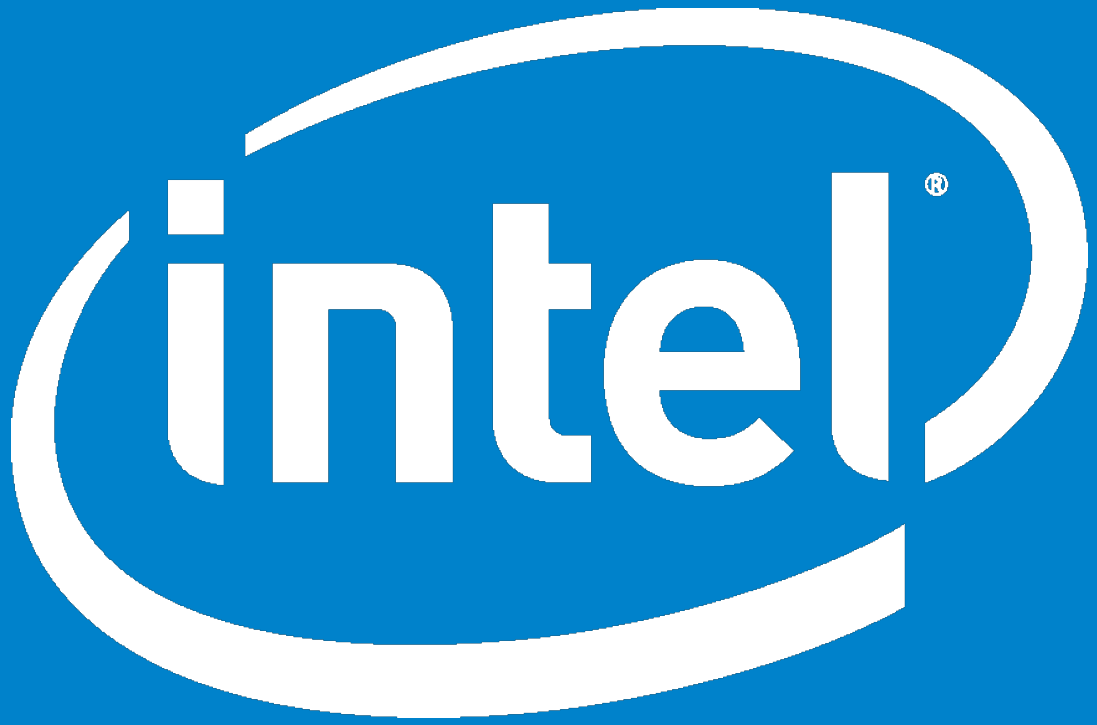
While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving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

Notice revision #20110228

Optimization Notice

Legal Disclaimer

http://intel.com/software/products


intel

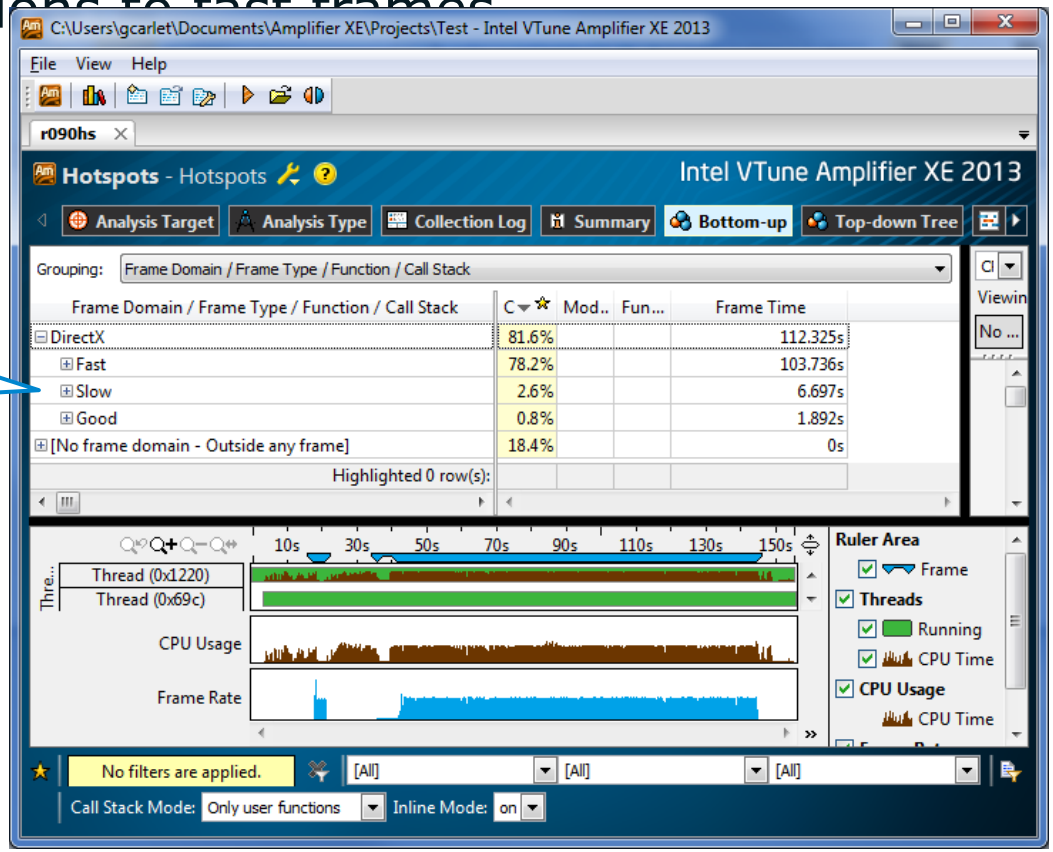
Intel® VTune™ Amplifier XE 2013

AutoDetect DirectX* Frames

- Find occasional slow video frames
  - Identify causes of intermittently slow frames by comparing slow frame functions to fast frames
  - Definition of "slow" is user configurable

2.6% of DirectX frames were too slow

- Expand "Slow" and "Fast" nodes to see the differences and identify slow frame causes



"JIT" APIs

Profiling Runtime generated code

APIs to indicate attributes of code

- Code memory address
- Symbol information
  - Function names, Line Numbers

Drill down to source code when viewing profiling analysis

APIs are defined in jitprofiling.h