# Intel® Network Builders Insights Series

## Analyze & Optimize FlexRAN, DPDK and Other Network Workloads Using Intel® oneAPI

- Xiaojun (Shawn) Li, Sales Director, Next Wave OEM & eODM
- Abhinav Singh, Software Technical Consulting Engineer
- Ashish Gupta, Business Development Manager

# Notices & Disclaimers

## Today's Plan

- Intel® oneAPI overview
- How to obtain Intel compilers for FlexRAN
- How IPP helps with FlexRAN
- Vtune™ Overview & Demo
- Q&A

# Intel® oneAPI Base & IoT Toolkits

## What is it?

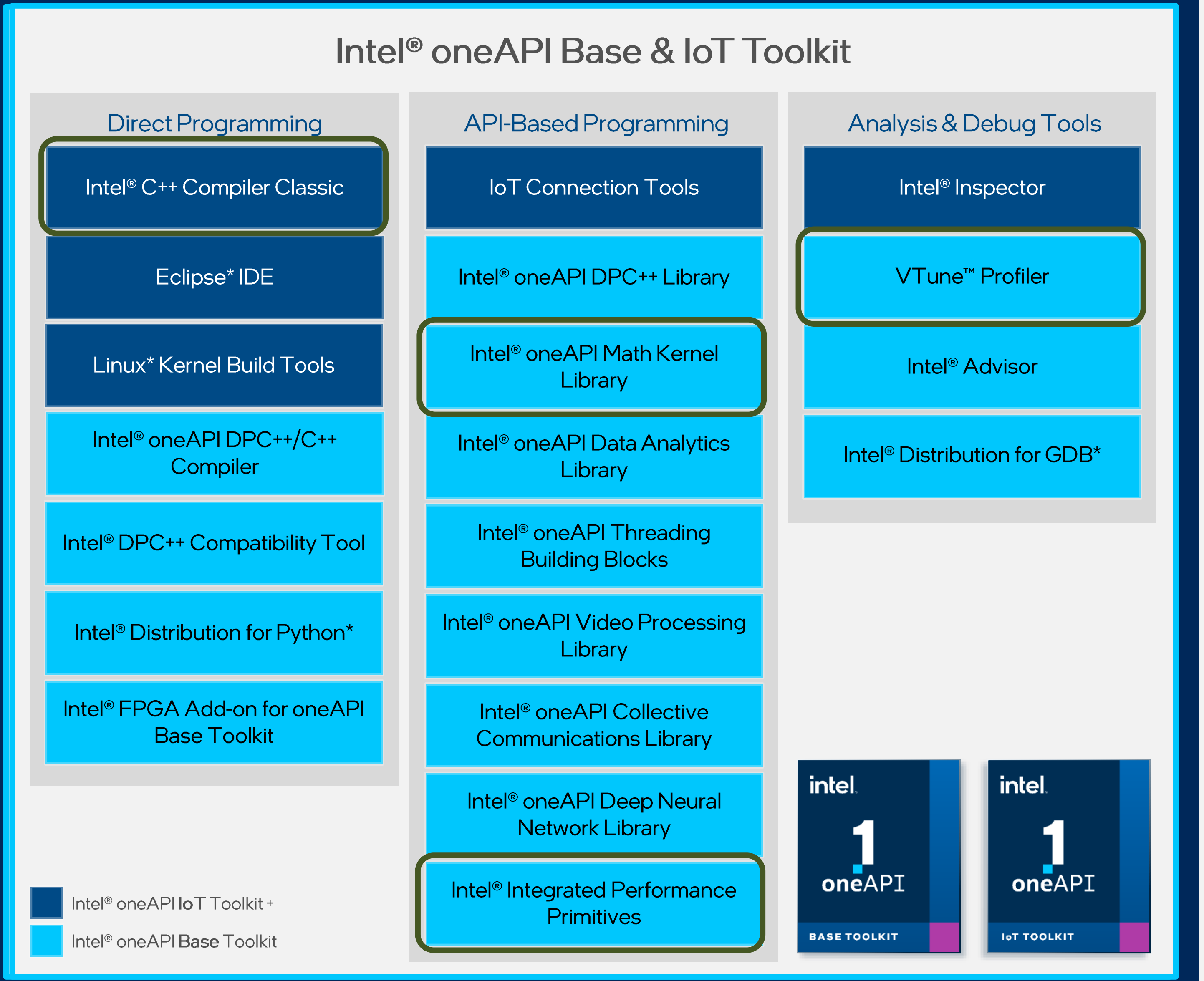
The **Intel® oneAPI Base** and **IoT Toolkits** help developers build efficient, reliable solutions that run at network's edge for healthcare, smart homes, industrial, retail, aerospace, and more. With this set of cross-architecture tools, performance libraries, and compiler, efficiently develop IoT applications and simplify deployment across Intel CPUs, GPUs, FPGAs.

## Who needs this product?

- OEMs, ODMs, SIs, ISVs
- DPC++, C, C++, OpenMP, Python* Developers

## Why is this important?

- Leverage more cores & built-in technologies in Intel® architecture through optimized compilers & libraries
- Develop with confidence with powerful analysis tools to identify threading, memory & offloading optimization opportunities
- DPC++ compatibility tool helps migrate existing code written in CUDA



### Intel® oneAPI Base & IoT Toolkit

**Direct Programming**
- 🟦 Intel® C++ Compiler Classic
- 🟦 Eclipse* IDE
- 🟦 Linux* Kernel Build Tools
- Intel® oneAPI DPC++/C++ Compiler
- Intel® DPC++ Compatibility Tool
- Intel® Distribution for Python*
- Intel® FPGA Add-on for oneAPI Base Toolkit

**API-Based Programming**
- 🟦 IoT Connection Tools
- Intel® oneAPI DPC++ Library
- Intel® oneAPI Math Kernel Library
- Intel® oneAPI Data Analytics Library
- Intel® oneAPI Threading Building Blocks
- Intel® oneAPI Video Processing Library
- Intel® oneAPI Collective Communications Library
- Intel® oneAPI Deep Neural Network Library
- Intel® Integrated Performance Primitives

**Analysis & Debug Tools**
- 🟦 Intel® Inspector
- VTune™ Profiler
- Intel® Advisor
- Intel® Distribution for GDB*

🟦 Intel® oneAPI **IoT** Toolkit +
▢ Intel® oneAPI **Base** Toolkit

Learn More: intel.com/oneAPI-IOTKit

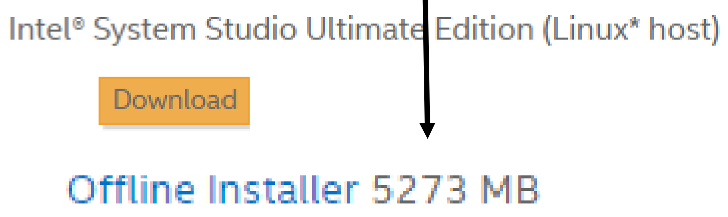
# FlexRAN v21.07 requires ICC 2019.3

**Step 1:** Get a commercial license for Intel® oneAPI Base & IoT Toolkit:

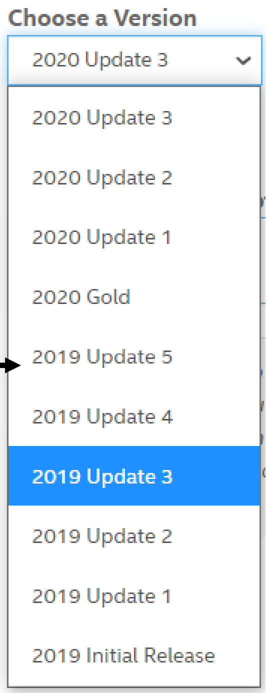
- Ask your Intel account manager (preferred)
- Alternatively, email: intel.software.sales@intel.com

**Step 2:** Register your serial number at https://registrationcenter.intel.com

**Step 3:** Download Intel System Studio Ultimate Edition (Linux host) 2019 Update 3



Intel® System Studio Ultimate Edition (Linux* host)
Download
Offline Installer 5273 MB



**Step 4A:** Install the entire package

Extract the Intel System Studio installation package

```
cd system_studio_2019_update_3_ultimate_edition_offline
Open and edit silent.cfg
ACCEPT_EULA=accept
PSET_INSTALL_DIR=/opt/intel (default dir) or change to [/your_chosen_directory]
COMPONENTS=DEFAULTS
ACTIVATION_TYPE=serial_number
ACTIVATION_SERIAL_NUMBER= XXXX-YYYYYYYY
-bash-4.1$ sudo ./install.sh -s silent.cfg
```

**OR**

**Step 4B:** Install only ICC

```
Open and edit silent.cfg
ACCEPT_EULA=accept
PSET_INSTALL_DIR =/opt/intel (default dir) or change to [/your_chosen_directory]
COMPONENTS=intel-icc-64bit-meta-linux-2019.3-206__noarch
ACTIVATION_TYPE=serial_number
ACTIVATION_SERIAL_NUMBER= XXXX-YYYYYYYY
-bash-4.1$ sudo ./install.sh -s silent.cfg
```

**Note: FlexRAN 21.11 will require ICX 2021.4 (or ICC 2019.3)**

# Intel® Ingenuity Partner Program (Intel® IPP)
# Your Building Blocks for Image, Signal & Data Processing Applications

## What is Intel® IPP?

Intel IPP developers with ready-to-use, processprovidesor- optimized functions to accelerate ***Image & Signal processing, Data Compression & Cryptography computation tasks***

## Why use Intel® IPP?

- High Performance
- Easy to Use APIs
- Faster Time to Market (TTM)
- Production Ready
- Cross-Platform API

## How to get Intel® IPP?

- Intel® oneAPI Base Toolkit

**Optimized for**

**Supports**

Windows Linux MacOS

**Addresses**

Data Center Internet of Things Embedded Systems Cloud Computing

***http://software.intel.com/intel-ipp***

## Image Processing Uses

- Medical Imaging
- Computer Vision
- Digital Surveillance
- ADAS
- Automated Sorting
- Biometric Identification Visual Search

## Signal Processing Uses

- Games (audio control or effects)
- Echo Cancellation
- Telecommunication/Vector Math
- Energy

## Data Compression & Crypto Uses

- Data centers
- Enterprise data management
- ID Verification
- Smart Cards/wallets
- Electronic Signature
- Information Security/Cybersecurity

# Intel® IPP Signal Processing Functions – 5G

| Intel® IPP Functions (5G) | Description |
|---|---|
| Copy | Copies the contents of one vector into another. It also works for bit copy. For example, ippsCopyLE_1u: work for 5G PDSCH/PUSCH rate match block ippsCopyLE_8u: work for 5G PDSCH/PUSCH rate de-match block |
| CRCs | Computes up to 32-bit CRC checksum for the source data buffer. IPP supports all the CRCs for 5G specification. The APIs work for 5G signal processing, for example, calculating CRC of TB, CB in data channel as well as control channels |
| Fourier Transform Functions | The functions perform the fast Fourier transform (FFT) and the discrete Fourier transform (DFT) of signal samples. They can speedup 5G PUSCH/PUCCH receiver for transform precoding, and work for FFT/DFT based channel estimation. |
| MaxIndx | Returns the maximum value of a vector and the index of the maximum element. It works for 5G PRACH peak detection |
| SortRadixIndexAscend/SortIndexAscend | Rearranges elements of the vector and their indexes. It works for 5G beamforming weight matrix generation |

# Sample Code – Intel® IPP FP16/single float32 FFT/DFT

| 5G call single float FFT/DFT | 5G call FP16 FFT/DFT |
|---|---|
| _/* step 1: Compute the sizes of the specification structure for specific length  complex FFT */_ status = ippsFFTGetSize_C_32f(3, IPP_FFT_DIV_INV_BY_N, ippAlgHintNone, &sizeSpec, &sizeInit, &sizeBuf); | _/* step 1: Performs the specific length complex FFT or DFT */_ **ippsDFTFwd_Direct_CToC_16fc**(pSrcFP16, pDstFP16, length); |
| _/* step 2: memory allocation with the sizes of the specification structure */_ pSpecMem = (Ipp8u*) ippMalloc( sizeSpec ); pBuffer  = (Ipp8u*) ippMalloc( sizeBuf ); pMemInit = (Ipp8u*) ippMalloc( sizeInit ); | |
| _/* step 3: Initialize the specification structure which contains such data as tables of twiddle factors */_ status = ippsFFTInit_C_32f(&pSpec, 3, IPP_FFT_DIV_INV_BY_N, ippAlgHintNone, pSpecMem, pMemInit); | |
| _/* step 4: Performs the specific length complex FFT or DFT */_ status = ippsFFTFwd_CToCCS_32f(pSrc, pDst, pSpec, NULL); | |
| _/* step 5:  Free memory */_ ippFree( pMemInit ); ippFree( pSpec ); ippFree( pBuffer ); | |

# Intel® VTune™ Profiler

## Get the Right Data to Find Bottlenecks

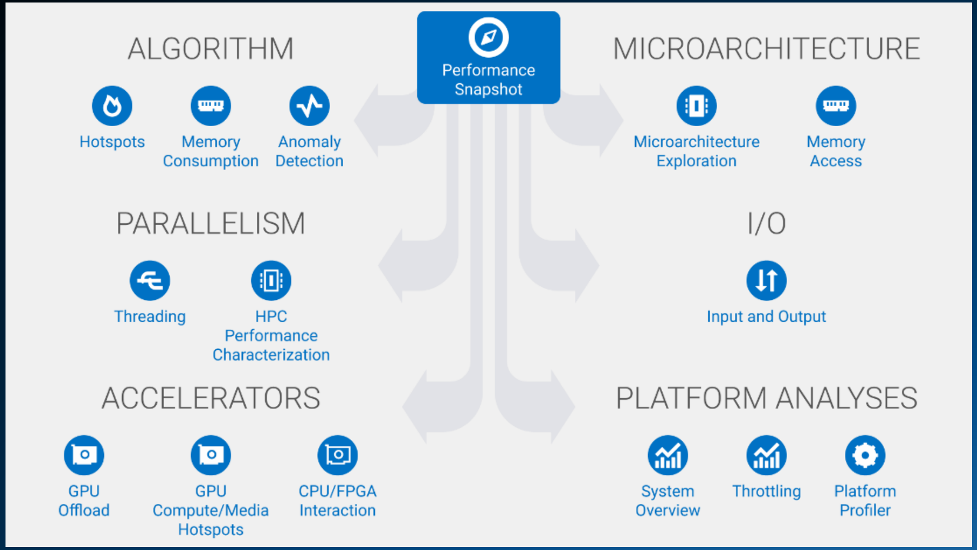
- A suite of profiling for CPU, GPU, FPGA, threading, memory, cache, storage, offload, power…
- DPC++, C, C++, Fortran, Python*, Go*, Java*, or a mix
- Linux, Windows, FreeBSD, Android, Yocto and more

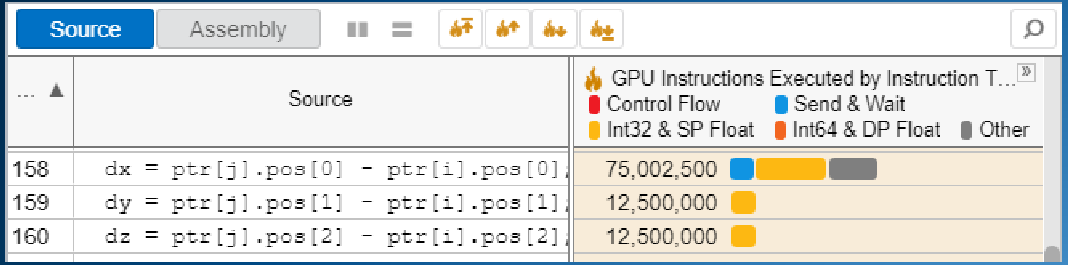
## Quickly Analyze Data

- See data on your source, in architecture diagrams, as a histogram, on a timeline…
- Filter and organize data to find answers

## Work Your Way

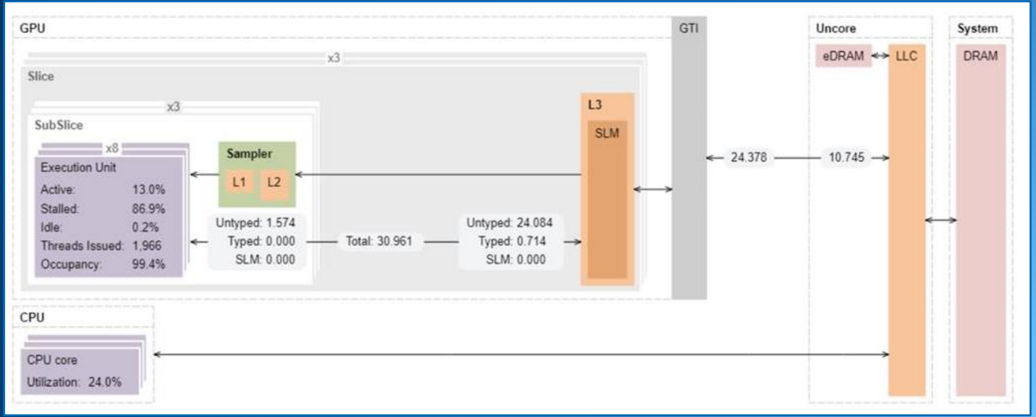
- User interface or command line
- Profile locally and remotely
- Install as an application
- Install as a server accessible with a web browser

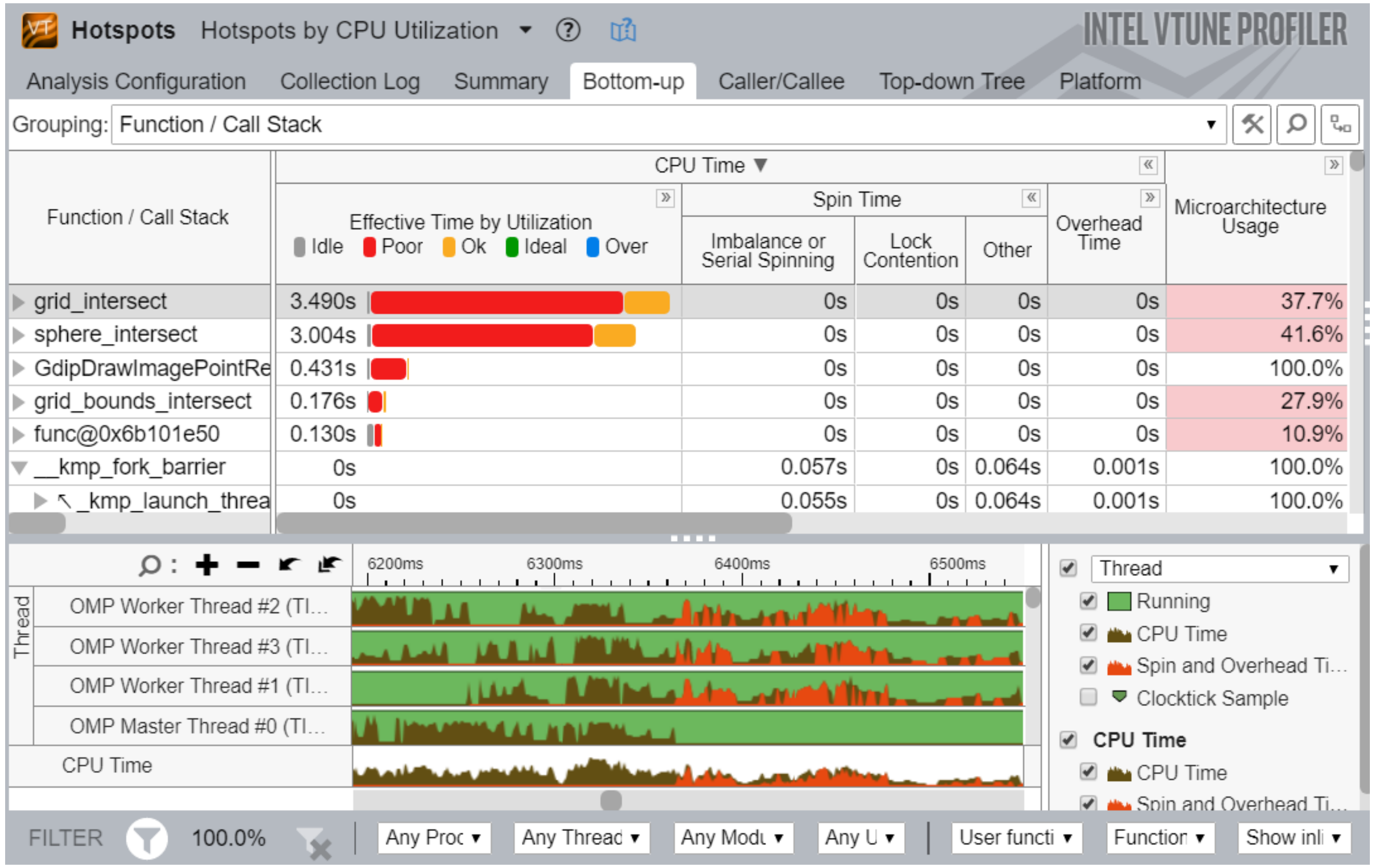



Source | Assembly

GPU Instructions Executed by Instruction T…
🟥 Control Flow 🟦 Send & Wait
🟨 Int32 & SP Float 🟧 Int64 & DP Float ▮ Other

| … ▲ | Source | GPU Instructions Executed by Instruction T… |
|---|---|---|
| 158 | `dx = ptr[j].pos[0] - ptr[i].pos[0];` | 75,002,500 |
| 159 | `dy = ptr[j].pos[1] - ptr[i].pos[1];` | 12,500,000 |
| 160 | `dz = ptr[j].pos[2] - ptr[i].pos[2];` | 12,500,000 |





Hotspots | Hotspots by CPU Utilization | INTEL VTUNE PROFILER

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

| Function / Call Stack | CPU Time ▼ | | | | | Microarchitecture Usage |
|---|---|---|---|---|---|---|
| | Effective Time by Utilization (▮ Idle ▮ Poor ▮ Ok ▮ Ideal ▮ Over) | Spin Time | | | Overhead Time | |
| | | Imbalance or Serial Spinning | Lock Contention | Other | | |
| ▶ grid_intersect | 3.490s | 0s | 0s | 0s | 0s | 37.7% |
| ▶ sphere_intersect | 3.004s | 0s | 0s | 0s | 0s | 41.6% |
| ▶ GdipDrawImagePointRe | 0.431s | 0s | 0s | 0s | 0s | 100.0% |
| ▶ grid_bounds_intersect | 0.176s | 0s | 0s | 0s | 0s | 27.9% |
| ▶ func@0x6b101e50 | 0.130s | 0s | 0s | 0s | 0s | 10.9% |
| ▼ __kmp_fork_barrier | 0s | 0.057s | 0s | 0.064s | 0.001s | 100.0% |
| ▶ ↖ _kmp_launch_threa | 0s | 0.055s | 0s | 0.064s | 0.001s | 100.0% |

FILTER 100.0% | Any Proc | Any Thread | Any Modu | Any U | User functi | Function | Show inli

# Live Demo:

## Intel® VTune™ Amplifier

## Questions?

Xiaojun (Shawn) Li, Sales Director, Next Wave OEM & eODM

**Xiaojun.Li@intel.com**

Abhinav Singh, Software Technical Consulting Engineer

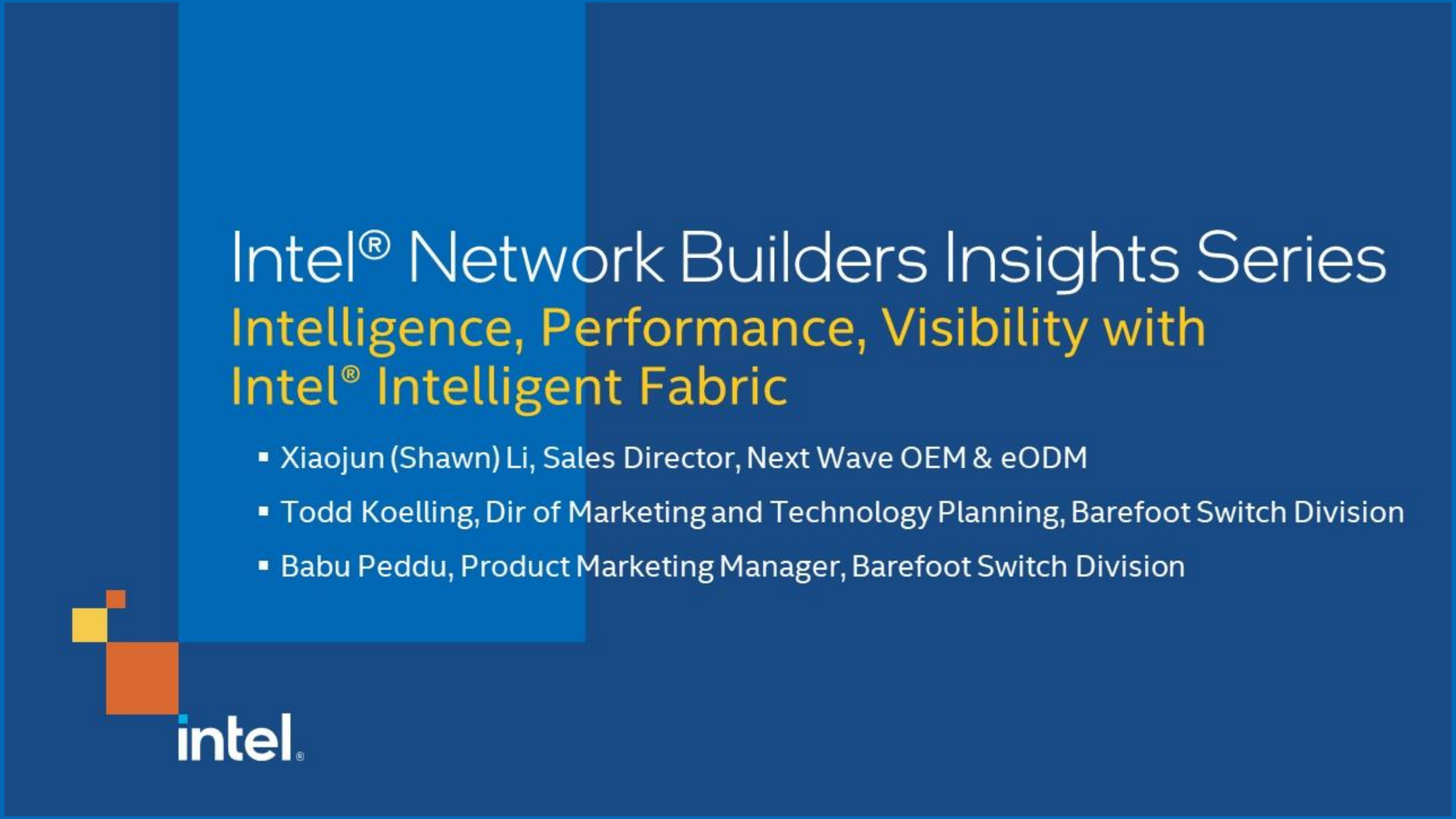
Abhinav.Singh@intel.com

Ashish Gupta, Business Development Manager

Ashish.Gupta@intel.com

# Join Us Next Time

December 1st @ 8am PDT



Intel® Network Builders Insights Series

Intelligence, Performance, Visibility with Intel® Intelligent Fabric

- Xiaojun (Shawn) Li, Sales Director, Next Wave OEM & eODM
- Todd Koelling, Dir of Marketing and Technology Planning, Barefoot Switch Division
- Babu Peddu, Product Marketing Manager, Barefoot Switch Division

intel®

## Backup

## Intel® VTune™ Profiler Overview

# Rich Set of Profiling Capabilities for Multiple Markets

Intel® VTune™ Profiler

**Single Thread**

Optimize single-threaded performance.

**Multithreaded**

Effectively use all available cores.

**System**

See a system-level view of application performance.

**Media & OpenCL™ Applications**

Deliver high-performance image and video processing pipelines.

**HPC & CLoud**

Access specialized, in-depth analyses for HPC and cloud computing.

**Memory & Storage Management**

Diagnose memory, storage, and data plane bottlenecks.

**Analyze & Filter Data**

Mine data for answers.

**Environment**

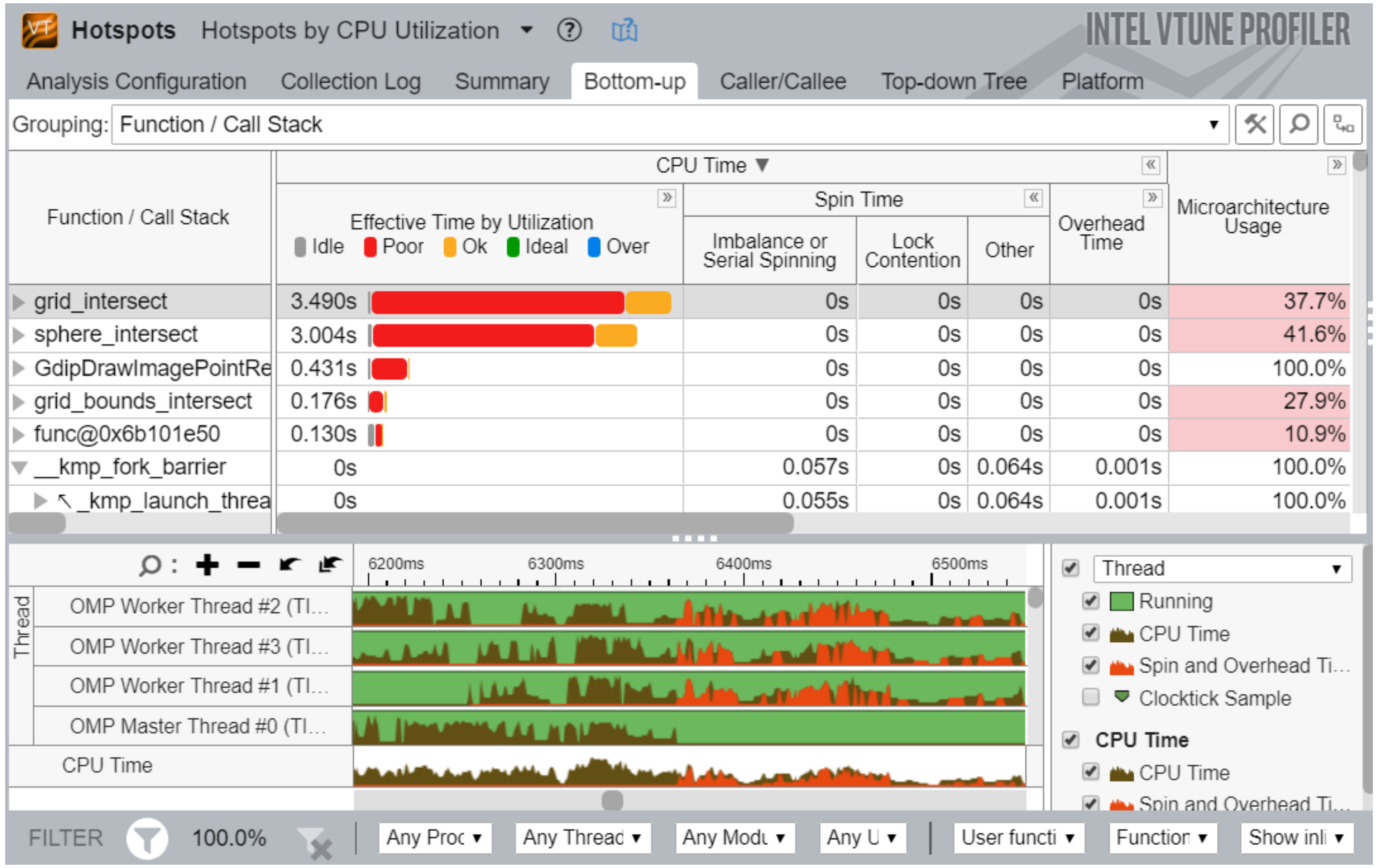
Fits your environment and workflow.

# Intel® VTune™ Profiler

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
  - Remote collection
- Key features
- Summary



Grouping: Function / Call Stack

| Function / Call Stack | CPU Time ▼ | | | | | Microarchitecture Usage |
|---|---|---|---|---|---|---|
| | Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over) | Spin Time | | | Overhead Time | |
| | | Imbalance or Serial Spinning | Lock Contention | Other | | |
| ▶ grid_intersect | 3.490s | 0s | 0s | 0s | 0s | 37.7% |
| ▶ sphere_intersect | 3.004s | 0s | 0s | 0s | 0s | 41.6% |
| ▶ GdipDrawImagePointRe | 0.431s | 0s | 0s | 0s | 0s | 100.0% |
| ▶ grid_bounds_intersect | 0.176s | 0s | 0s | 0s | 0s | 27.9% |
| ▶ func@0x6b101e50 | 0.130s | 0s | 0s | 0s | 0s | 10.9% |
| ▼ __kmp_fork_barrier | 0s | 0.057s | 0s | 0.064s | 0.001s | 100.0% |
| ▶ ↖ _kmp_launch_threa | 0s | 0.055s | 0s | 0.064s | 0.001s | 100.0% |

## Two Great Ways to Collect Data

Intel® VTune™ Profiler

| 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 | Uses Intel driver or perf if driver not installed |

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

# Intel® VTune™ Profiler

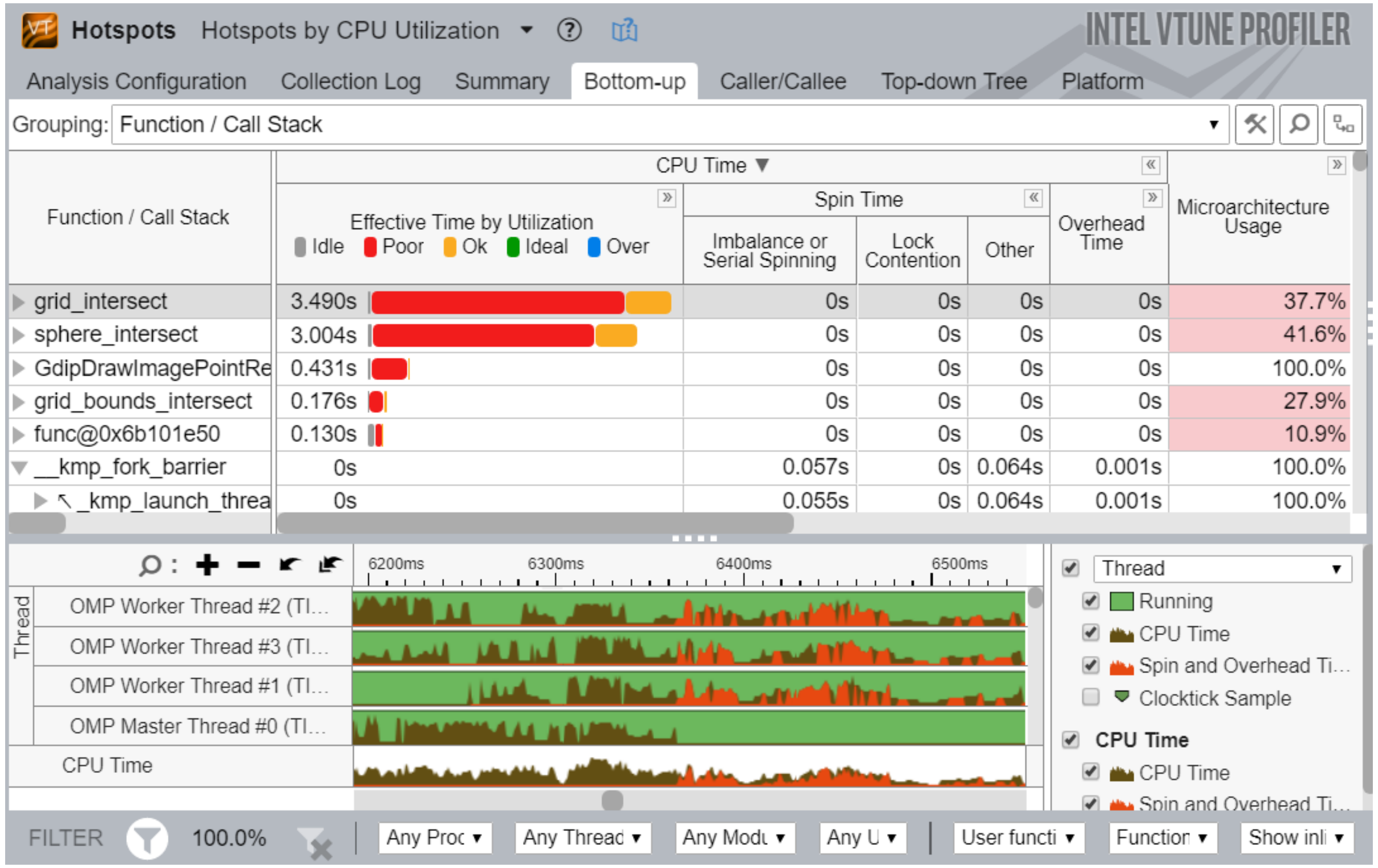
## Tune Applications for Scalable Multicore Performance

### Agenda

- Data Collection –

  Rich set of performance data
- Data Analysis -

  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Remote collection
- Key features
- Summary


Hotspots  Hotspots by CPU Utilization ▾  INTEL VTUNE PROFILER

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

| Function / Call Stack | CPU Time ▼: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over) | Spin Time: Imbalance or Serial Spinning | Spin Time: Lock Contention | Spin Time: Other | Overhead Time | Microarchitecture Usage |
|---|---|---|---|---|---|---|
| ▸ grid_intersect | 3.490s | 0s | 0s | 0s | 0s | 37.7% |
| ▸ sphere_intersect | 3.004s | 0s | 0s | 0s | 0s | 41.6% |
| ▸ GdipDrawImagePointRe | 0.431s | 0s | 0s | 0s | 0s | 100.0% |
| ▸ grid_bounds_intersect | 0.176s | 0s | 0s | 0s | 0s | 27.9% |
| ▸ func@0x6b101e50 | 0.130s | 0s | 0s | 0s | 0s | 10.9% |
| ▾ __kmp_fork_barrier | 0s | 0.057s | 0s | 0.064s | 0.001s | 100.0% |
| ▸ ↖ _kmp_launch_threa | 0s | 0.055s | 0s | 0.064s | 0.001s | 100.0% |

# Find Answers Fast

## Intel® VTune™ Profiler

**Adjust Data Grouping**

Function / Call Stack

Source Function / Function / Call Stack

Sync Object / Function / Call Stack

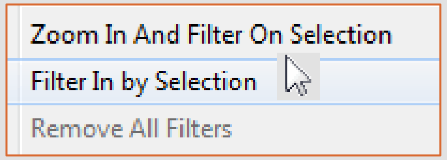
Sync Object / Thread / Function / Call Stack

... (Partial list shown)

**Double Click Function to View Source**

**Click [▶] for Call Stack**

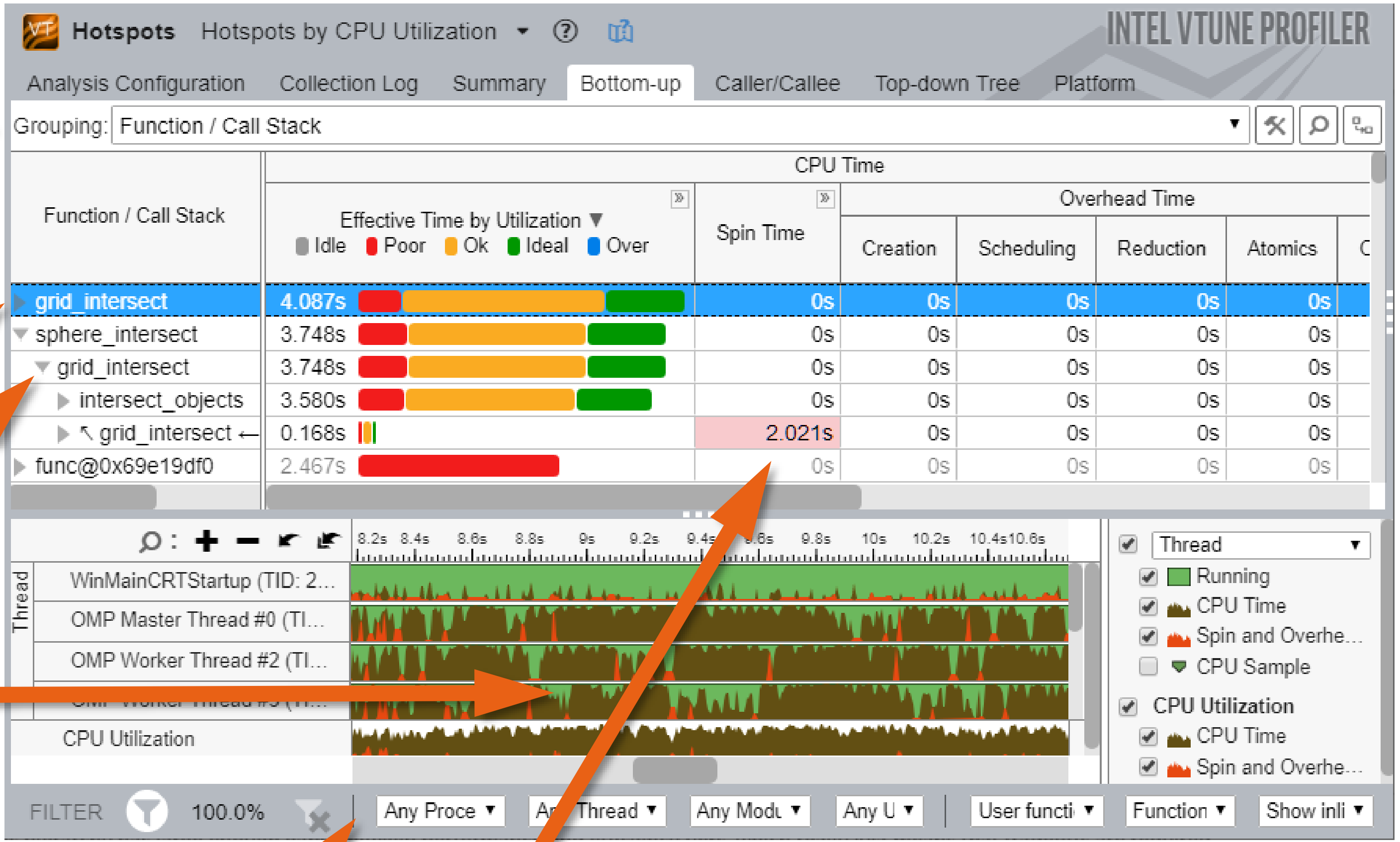
**Filter by Timeline Selection (or by Grid Selection)**



Zoom In And Filter On Selection

Filter In by Selection

Remove All Filters



Hotspots Hotspots by CPU Utilization

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

| Function / Call Stack | CPU Time: Effective Time by Utilization (▨ Idle 🟥 Poor 🟧 Ok 🟩 Ideal 🟦 Over) | Spin Time | Overhead Time: Creation | Scheduling | Reduction | Atomics | C |
|---|---|---|---|---|---|---|---|
| ▶ grid_intersect | 4.087s | 0s | 0s | 0s | 0s | 0s | |
| ▼ sphere_intersect | 3.748s | 0s | 0s | 0s | 0s | 0s | |
| ▼ grid_intersect | 3.748s | 0s | 0s | 0s | 0s | 0s | |
| ▶ intersect_objects | 3.580s | 0s | 0s | 0s | 0s | 0s | |
| ▶ ↖ grid_intersect ← | 0.168s | 2.021s | 0s | 0s | 0s | 0s | |
| ▶ func@0x69e19df0 | 2.467s | 0s | 0s | 0s | 0s | 0s | |

FILTER 100.0% | Any Proce ▾ | Any Thread ▾ | Any Modu ▾ | Any U ▾ | User functi ▾ | Function ▾ | Show inli ▾

**Filter by Process & Other Controls**

**Tuning Opportunities Shown in Pink. Hover for Tips**

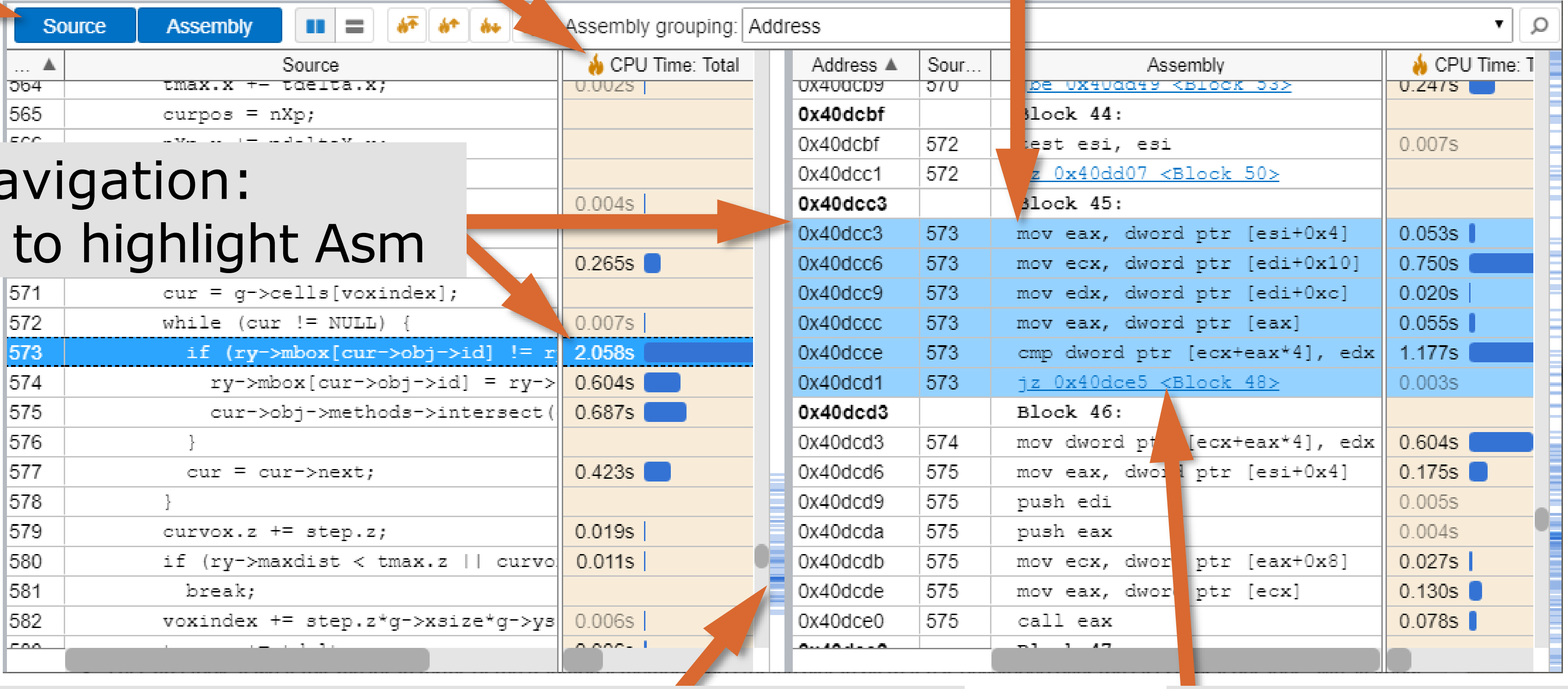
# See Profile Data On Source / Asm

## Double Click from Grid or Timeline

View Source / Asm or both

CPU Time

Right click for instruction reference manual



| ... | Source | CPU Time: Total |
|---|---|---|
| 564 | tmax.x += tdelta.x; | 0.002s |
| 565 | curpos = nXp; | |
| | | 0.004s |
| | | 0.265s |
| 571 | cur = g->cells[voxindex]; | |
| 572 | while (cur != NULL) { | 0.007s |
| 573 | if (ry->mbox[cur->obj->id] != r | 2.058s |
| 574 | ry->mbox[cur->obj->id] = ry-> | 0.604s |
| 575 | cur->obj->methods->intersect( | 0.687s |
| 576 | } | |
| 577 | cur = cur->next; | 0.423s |
| 578 | } | |
| 579 | curvox.z += step.z; | 0.019s |
| 580 | if (ry->maxdist < tmax.z \|\| curvo | 0.011s |
| 581 | break; | |
| 582 | voxindex += step.z*g->xsize*g->ys | 0.006s |

| Address | Sour... | Assembly | CPU Time: T |
|---|---|---|---|
| 0x40dcb9 | 570 | jbe 0x40dd49 <Block 53> | 0.247s |
| **0x40dcbf** | | Block 44: | |
| 0x40dcbf | 572 | test esi, esi | 0.007s |
| 0x40dcc1 | 572 | jz 0x40dd07 <Block 50> | |
| **0x40dcc3** | | Block 45: | |
| 0x40dcc3 | 573 | mov eax, dword ptr [esi+0x4] | 0.053s |
| 0x40dcc6 | 573 | mov ecx, dword ptr [edi+0x10] | 0.750s |
| 0x40dcc9 | 573 | mov edx, dword ptr [edi+0xc] | 0.020s |
| 0x40dccc | 573 | mov eax, dword ptr [eax] | 0.055s |
| 0x40dcce | 573 | cmp dword ptr [ecx+eax*4], edx | 1.177s |
| 0x40dcd1 | 573 | jz 0x40dce5 <Block 48> | 0.003s |
| **0x40dcd3** | | Block 46: | |
| 0x40dcd3 | 574 | mov dword ptr [ecx+eax*4], edx | 0.604s |
| 0x40dcd6 | 575 | mov eax, dword ptr [esi+0x4] | 0.175s |
| 0x40dcd9 | 575 | push edi | 0.005s |
| 0x40dcda | 575 | push eax | 0.004s |
| 0x40dcdb | 575 | mov ecx, dword ptr [eax+0x8] | 0.027s |
| 0x40dcde | 575 | mov eax, dword ptr [ecx] | 0.130s |
| 0x40dce0 | 575 | call eax | 0.078s |

Quick Asm navigation:
Select source to highlight Asm

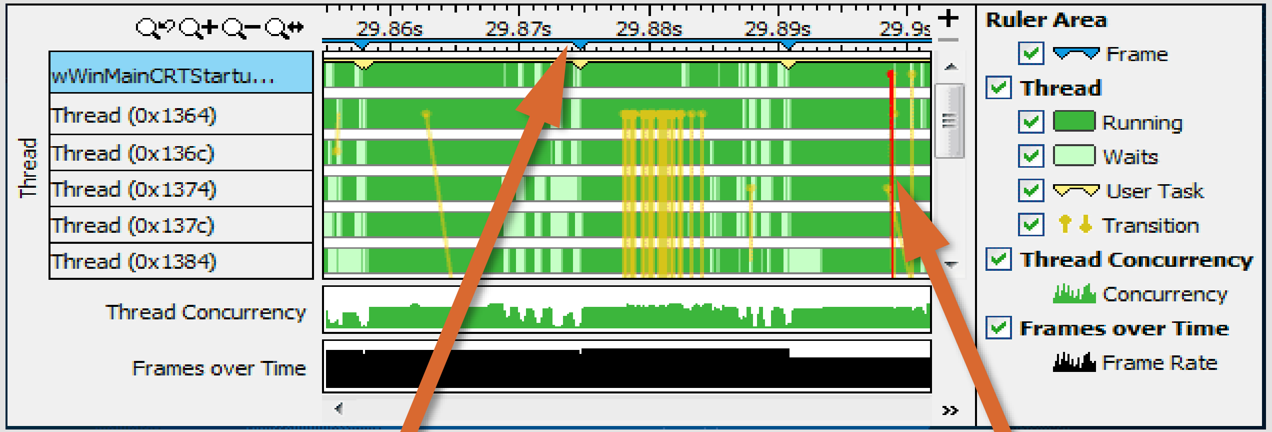
Scroll Bar “Heat Map” is an overview of hot spots

Click jump to scroll Asm

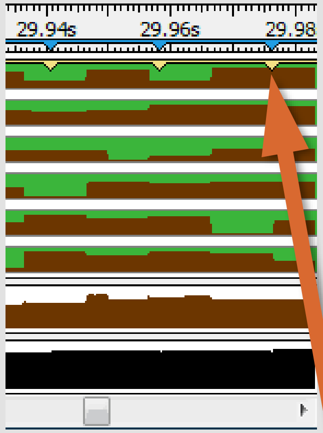
# Timeline Visualizes Thread Behavior

## Intel® VTune™ Profiler

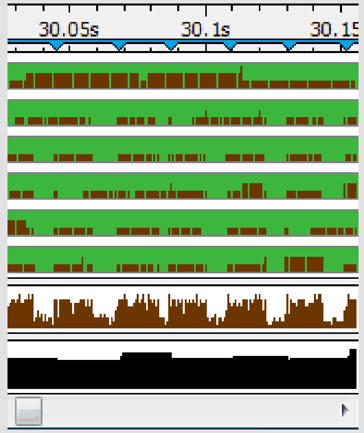
**Transitions**
Locks & Waits



**CPU Time**
Basic Hotspots



Advanced Hotspots



Hovers:

Frame

Frame
Start: 29.858s Duration: 0.017s
Frame: 72
Frame Domain: Smoke::Framework::execute()
Frame Type: Good
Frame Rate: 59.8242179

Transition

Transition
wWinMainCRTStartup (0x12d4) to Thread (0x138c) (29.899s to 29.899s)
Sync Object: TBB Scheduler
Object Creation File: taskmanagertbb.cpp
Object Creation Line: 318

User Task

User Task
Start: 29.958s Duration: 0.018s
Task Type: Smoke::FrameWork::execute()::Other
Task End Call Stack: Framework::Execute

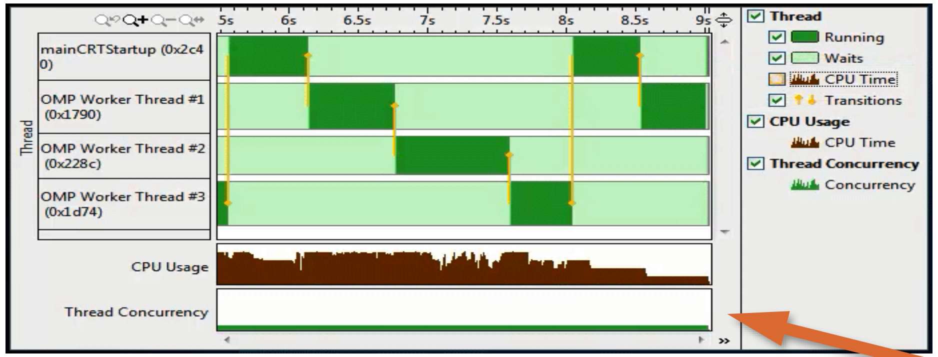
CPU Time
94.233472%

- Optional: Use API to mark frames and user tasks   Frame   User Task
- Optional: Add a mark during collection

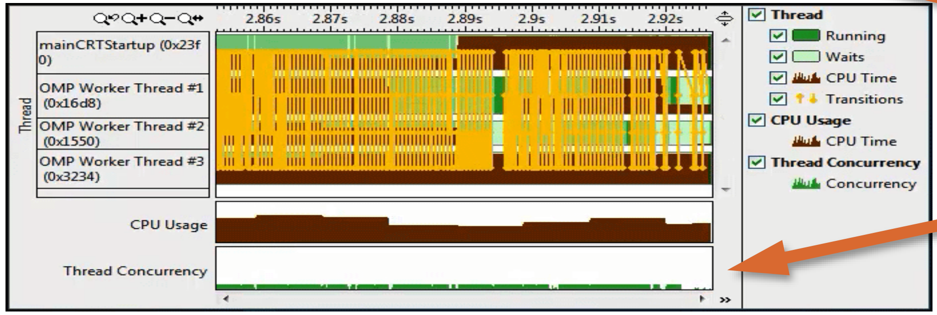
# Visualize Parallel Performance Issues

## Look for Common Patterns

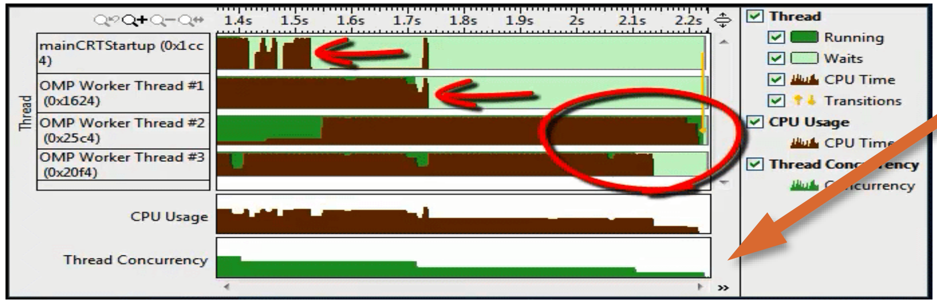
Coarse Grain Locks



High Lock Contention



Load Imbalance



Low Concurrency

# Intel® VTune™ Profiler

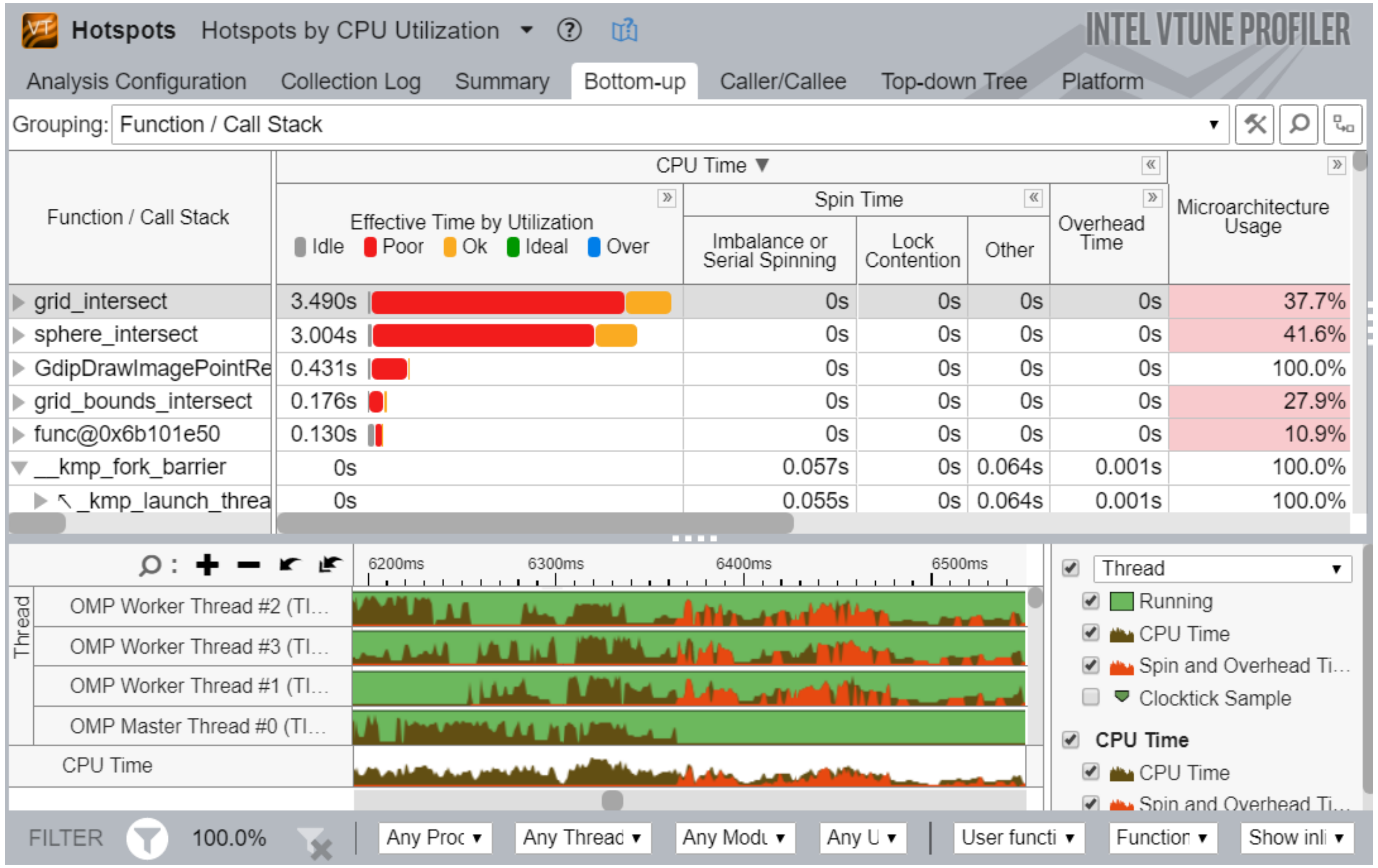
Tune Applications for Scalable Multicore Performance

## Agenda

- Data Collection –

  Rich set of performance data
- Data Analysis -

  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Remote collection
- Key features
- Summary



Grouping: Function / Call Stack

| Function / Call Stack | CPU Time ▼ | | | | | | Microarchitecture Usage |
|---|---|---|---|---|---|---|---|
| | Effective Time by Utilization (▮ Idle 🟥 Poor 🟧 Ok 🟩 Ideal 🟦 Over) | | Spin Time | | | Overhead Time | |
| | | | Imbalance or Serial Spinning | Lock Contention | Other | | |
| grid_intersect | 3.490s | 🟥🟧 | 0s | 0s | 0s | 0s | 37.7% |
| sphere_intersect | 3.004s | 🟥🟧 | 0s | 0s | 0s | 0s | 41.6% |
| GdipDrawImagePointRe | 0.431s | 🟥 | 0s | 0s | 0s | 0s | 100.0% |
| grid_bounds_intersect | 0.176s | 🟥 | 0s | 0s | 0s | 0s | 27.9% |
| func@0x6b101e50 | 0.130s | 🟥 | 0s | 0s | 0s | 0s | 10.9% |
| __kmp_fork_barrier | 0s | | 0.057s | 0s | 0.064s | 0.001s | 100.0% |
| _kmp_launch_threa | 0s | | 0.055s | 0s | 0.064s | 0.001s | 100.0% |

6200ms
6300ms
6400ms
6500ms
Thread
OMP Worker Thread #2 (TI...
OMP Worker Thread #3 (TI...
OMP Worker Thread #1 (TI...
OMP Master Thread #0 (TI...
CPU Time

- ✓ Thread
  - ✓ 🟩 Running
  - ✓ CPU Time
  - ✓ Spin and Overhead Ti...
  - ☐ Clocktick Sample
- ✓ CPU Time
  - ✓ CPU Time
  - ✓ Spin and Overhead Ti...

FILTER 100.0% | Any Proc ▾ | Any Thread ▾ | Any Modu ▾ | Any U ▾ | User functi ▾ | Function ▾ | Show inli ▾

# Command Line Interface

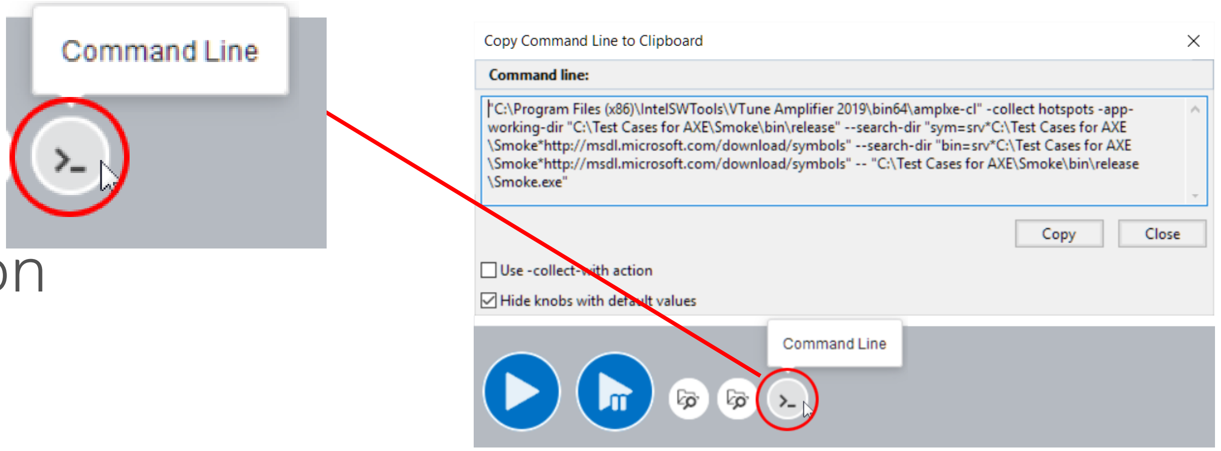
Automate analysis

- Set up the environment variables:
  - `Windows: <install-dir>\env\vars.bat`
  - `Linux: <install-dir>/env/vars.sh`

Help: `vtune -help`

Use UI to setup

1) Configure analysis in UI
2) Press "Command Line…" button
3) Copy & paste command



```
"C:\Program Files (x86)\IntelSWTools\VTune Amplifier 2019\bin64\amplxe-cl" -collect hotspots -app-
working-dir "C:\Test Cases for AXE\Smoke\bin\release" --search-dir "sym=srv*C:\Test Cases for AXE
\Smoke*http://msdl.microsoft.com/download/symbols" --search-dir "bin=srv*C:\Test Cases for AXE
\Smoke*http://msdl.microsoft.com/download/symbols" -- "C:\Test Cases for AXE\Smoke\bin\release
\Smoke.exe"
```

☐ Use -collect-with action

☑ Hide knobs with default values

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

# Default VTune Profiler Install Directories

**In Intel® oneAPI Base Toolkit:**

- Windows:  [Program Files]\Intel\oneAPI\vtune\<*version*>
- Linux:       /opt/intel/oneapi/vtune/<*version*>

**Standalone:**

- Windows:  [Program Files]\IntelSWTools\VTune Profiler <*version*>
- Linux:       /opt/intel/vtune_profiler_*version*

**On Apple* macOS* systems:**

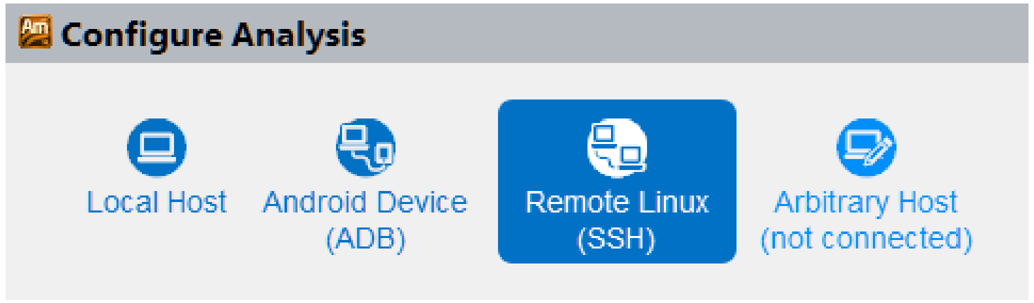
- /Applications/Intel VTune Profiler <*version*>.app

# Interactive Remote Data Collection

Performance analysis of remote systems just got a lot easier

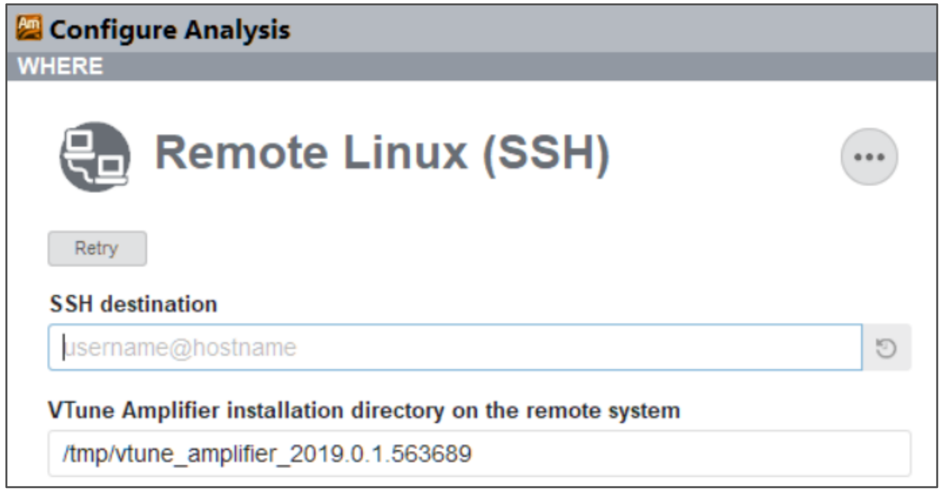
## Interactive analysis

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



SSH destination

username@hostname

VTune Amplifier installation directory on the remote system

/tmp/vtune_amplifier_2019.0.1.563689

**Conveniently use your local UI to analyze remote systems**

# Intel® VTune™ Profiler

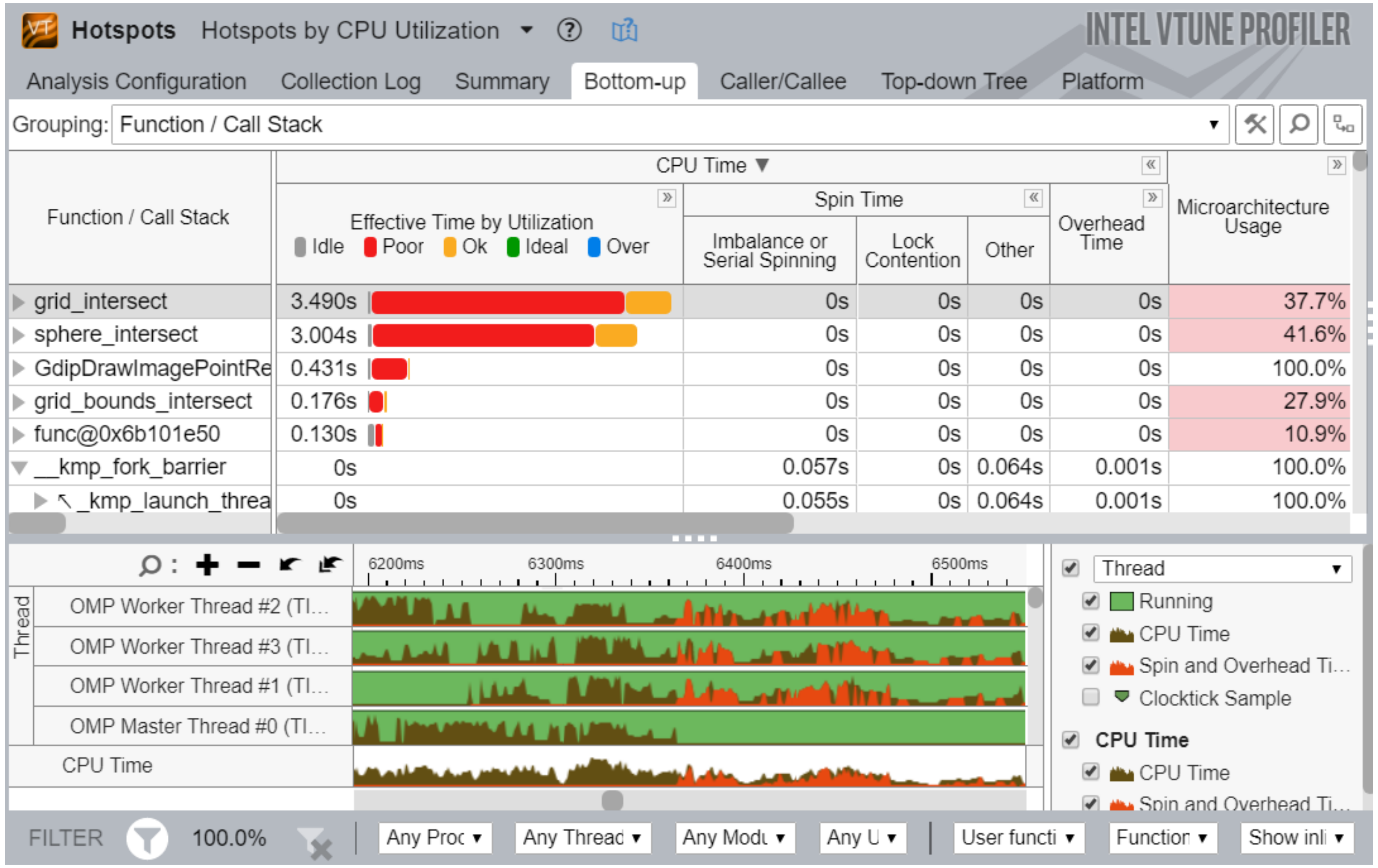
## Tune Applications for Scalable Multicore Performance

### Agenda

- Data Collection –

  Rich set of performance data
- Data Analysis -

  Find answers fast
- Flexible workflow –
  - User i/f and command line
  - Remote collection
- Key features
- Summary



| Function / Call Stack | CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over) | Spin Time: Imbalance or Serial Spinning | Spin Time: Lock Contention | Spin Time: Other | Overhead Time | Microarchitecture Usage |
|---|---|---|---|---|---|---|
| grid_intersect | 3.490s | 0s | 0s | 0s | 0s | 37.7% |
| sphere_intersect | 3.004s | 0s | 0s | 0s | 0s | 41.6% |
| GdipDrawImagePointRe | 0.431s | 0s | 0s | 0s | 0s | 100.0% |
| grid_bounds_intersect | 0.176s | 0s | 0s | 0s | 0s | 27.9% |
| func@0x6b101e50 | 0.130s | 0s | 0s | 0s | 0s | 10.9% |
| __kmp_fork_barrier | 0s | 0.057s | 0s | 0.064s | 0.001s | 100.0% |
| ↖ _kmp_launch_threa | 0s | 0.055s | 0s | 0.064s | 0.001s | 100.0% |

# Rich Set of Profiling Capabilities for Multiple Markets

## Intel® VTune™ Profiler

**Single Thread**

Optimize single-threaded performance.

**Multithreaded**

Effectively use all available cores.

**System**

See a system-level view of application performance.

**Media & OpenCL™ Applications**

Deliver high-performance image and video processing pipelines.

**HPC & CLoud**

Access specialized, in-depth analyses for HPC and cloud computing.

**Memory & Storage Management**

Diagnose memory, storage, and data plane bottlenecks.

**Analyze & Filter Data**

Mine data for answers.

**Environment**

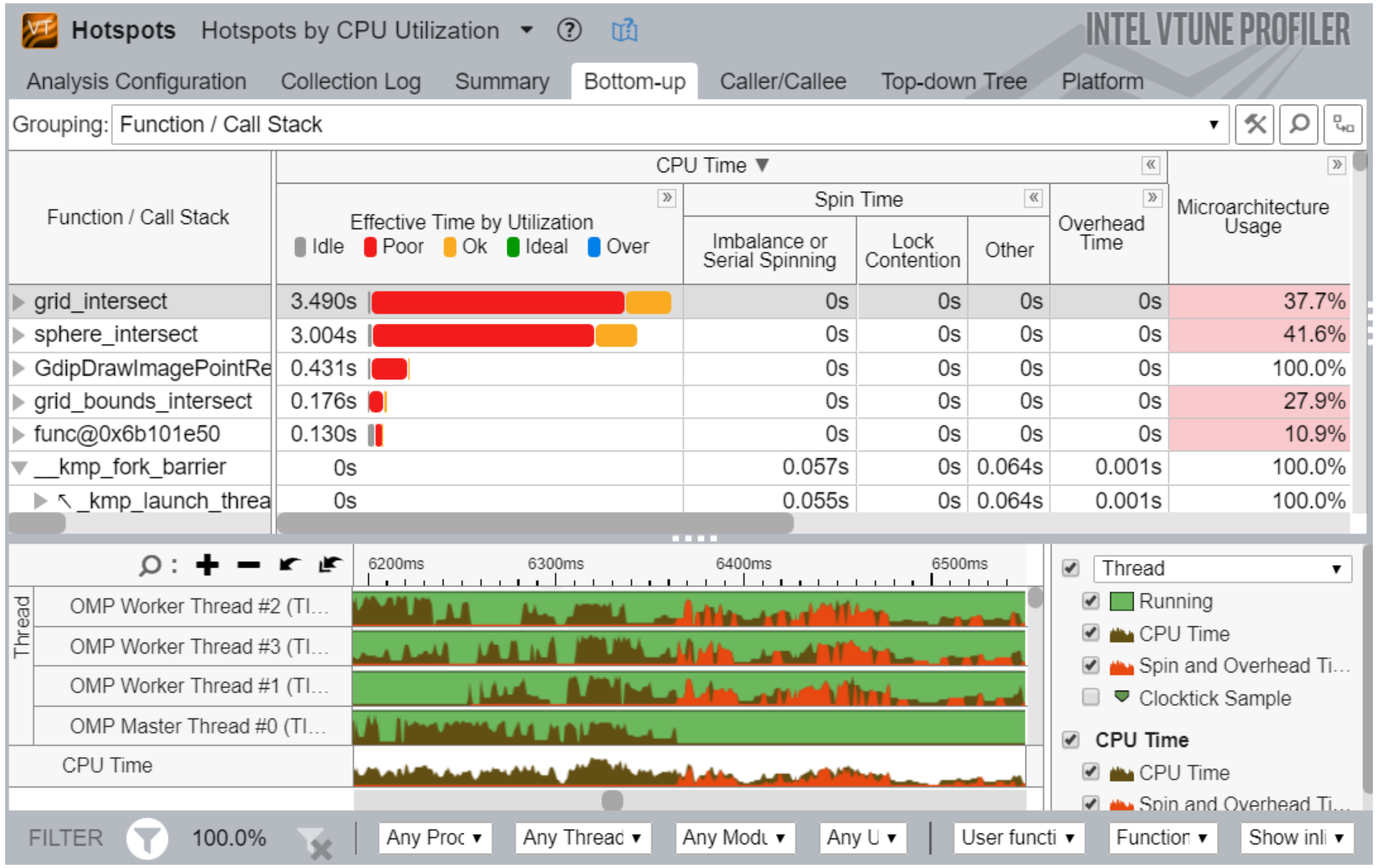
Fits your environment and workflow.

# Intel® VTune™ Profiler

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
  - Remote collection
- Key features
- Summary



Grouping: Function / Call Stack

| Function / Call Stack | CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over) | CPU Time: Spin Time: Imbalance or Serial Spinning | CPU Time: Spin Time: Lock Contention | CPU Time: Spin Time: Other | CPU Time: Overhead Time | Microarchitecture Usage |
|---|---|---|---|---|---|---|
| grid_intersect | 3.490s | 0s | 0s | 0s | 0s | 37.7% |
| sphere_intersect | 3.004s | 0s | 0s | 0s | 0s | 41.6% |
| GdipDrawImagePointRe | 0.431s | 0s | 0s | 0s | 0s | 100.0% |
| grid_bounds_intersect | 0.176s | 0s | 0s | 0s | 0s | 27.9% |
| func@0x6b101e50 | 0.130s | 0s | 0s | 0s | 0s | 10.9% |
| __kmp_fork_barrier | 0s | 0.057s | 0s | 0.064s | 0.001s | 100.0% |
| ↖ _kmp_launch_threa | 0s | 0.055s | 0s | 0.064s | 0.001s | 100.0% |

# Intel® VTune™ Profiler

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

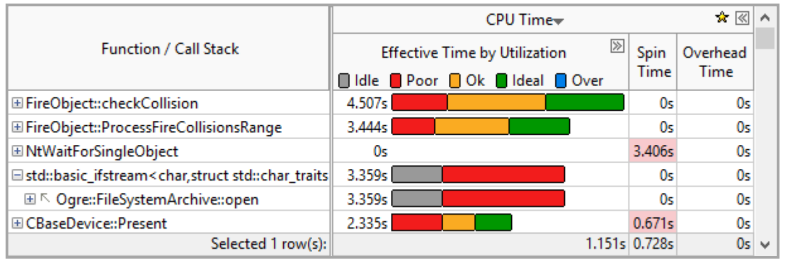
## Find Answers Fast

- 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

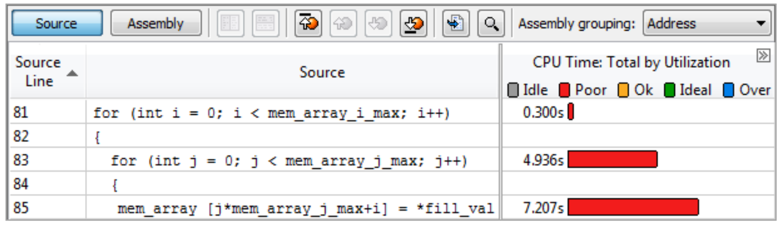
- No Special Compiles – C, C++, C#, Fortran, Java, Python, ASM
- Visual Studio* Integration or Stand Alone
- Local & Remote Data Collection, Command Line
- Analyze Windows* & Linux* data on macOS

## Quickly Find Tuning Opportunities



| Function / Call Stack | CPU Time: Effective Time by Utilization (🟥 Idle 🟥 Poor 🟧 Ok 🟩 Ideal 🟦 Over) | CPU Time: Spin Time | CPU Time: Overhead Time |
|---|---|---|---|
| ⊞ FireObject::checkCollision | 4.507s | 0s | 0s |
| ⊞ FireObject::ProcessFireCollisionsRange | 3.444s | 0s | 0s |
| ⊞ NtWaitForSingleObject | 0s | 3.406s | 0s |
| ⊟ std::basic_ifstream<char,struct std::char_traits | 3.359s | 0s | 0s |
| ⊞ ↖ Ogre::FileSystemArchive::open | 3.359s | 0s | 0s |
| ⊞ CBaseDevice::Present | 2.335s | 0.671s | 0s |
| Selected 1 row(s): | 1.151s | 0.728s | 0s |

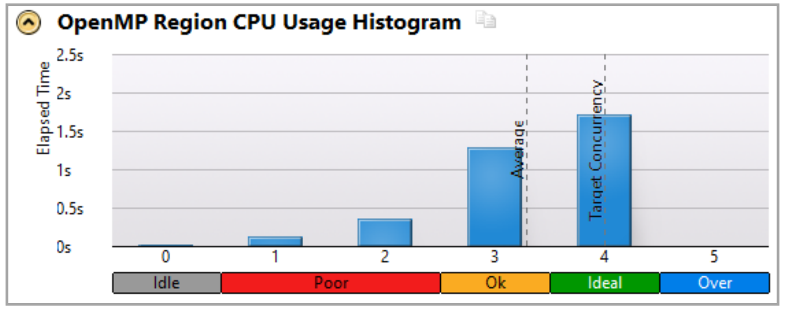
## See Results On The Source Code



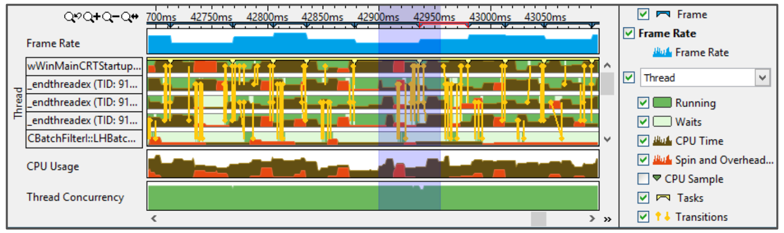
Source | Assembly | Assembly grouping: Address

| Source Line | Source | CPU Time: Total by Utilization (Idle, Poor, Ok, Ideal, Over) |
|---|---|---|
| 81 | `for (int i = 0; i < mem_array_i_max; i++)` | 0.300s |
| 82 | `{` | |
| 83 | `for (int j = 0; j < mem_array_j_max; j++)` | 4.936s |
| 84 | `{` | |
| 85 | `mem_array [j*mem_array_j_max+i] = *fill_val` | 7.207s |

## Tune OpenMP Scalability



## Visualize & Filter Data



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

# Don’t Miss Vectorization Advisor (not part of VTune™ Profiler)

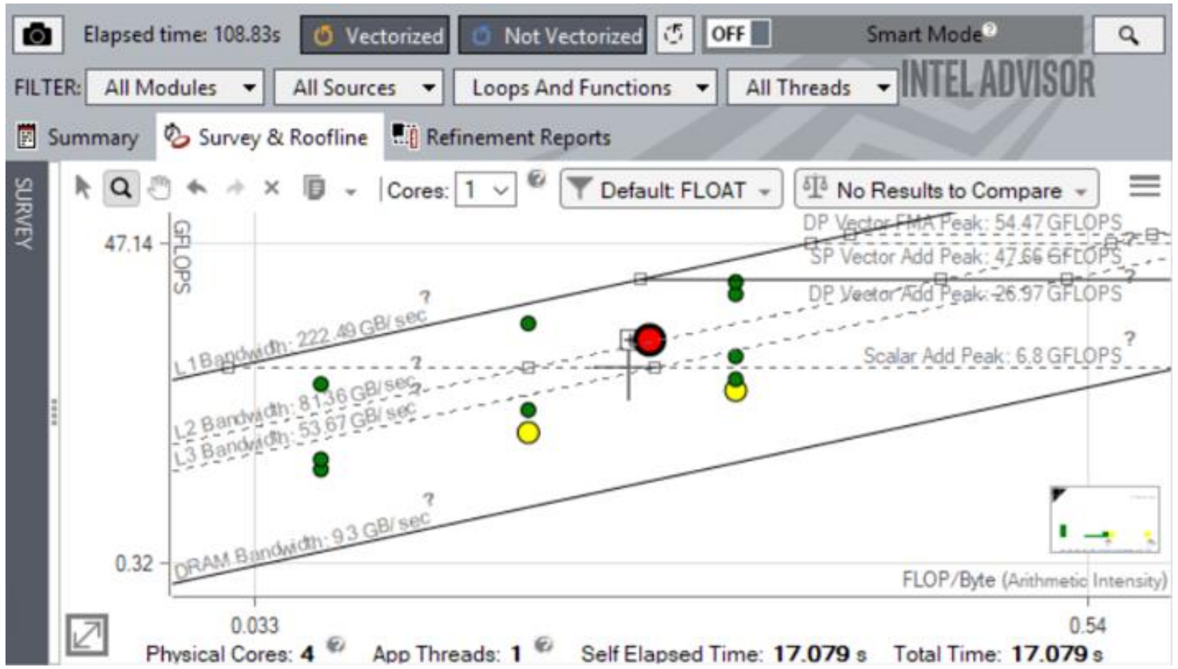
Intel® Advisor – Vectorization Optimization, Thread Prototyping, Flow Graph Analysis

## Faster Vectorization Optimization:

- Vectorize where it will pay off most
- Quickly ID what is blocking vectorization
- Tips for effective vectorization
- Safely force compiler vectorization
- Optimize memory stride

## The data and guidance you need:

- Compiler diagnostics + Performance Data + SIMD efficiency
- Detect problems & recommend fixes
- Loop-Carried Dependency Analysis
- Memory Access Patterns Analysis



**Optimize for AVX-512 with or without access to AVX-512 hardware**

http://intel.ly/advisor-xe

intel.

## Tools to Design Code for Modern Hardware

Optimize Vectorization, Prototype Threading, Create & Analyze Flow Graphs

**Roofline Analysis**

Optimize your application for memory and compute.

**Vectorization Optimization**

Enable more vector parallelism and improve its efficiency.

**Thread Prototyping**

Model, tune, and test multiple threading designs.

**Build Heterogeneous Algorithms**

Create and analyze data flow and dependency computation graphs.

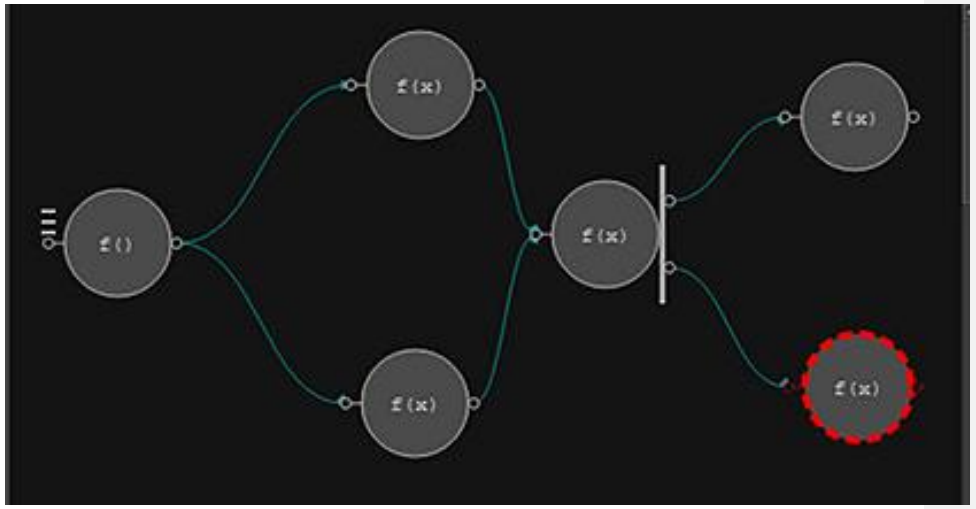
Learn More: software.intel.com/advisor

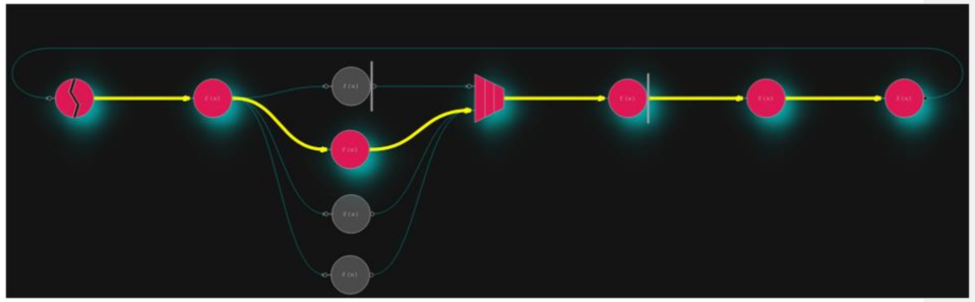
# Interactively Build, Validate & Analyze Flow Graphs

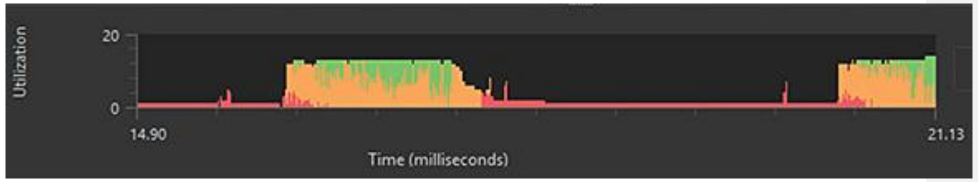
Intel® Advisor—Flow Graph Analyzer (FGA)

## Design and Analyze Flow Graphs

- Visually generate code stubs
- Generate parallel C++ programs
- Click & zoom through your algorithm's nodes & edges to understand parallel data & program flow
- Analyze load balancing, concurrency, & other parallel attributes to fine tune your program
- Use Intel® TBB or OpenMP* 5 (draft) OMPT APIs







Intel® VTune™ Profiler

## DPDK, SPDK & Network/Storage Analysis

# Tune SPDK & DPDK I/O

Measure "Empty" Polling Cycles

## Gather Key I/O Metrics – See:

- How each device performs
- I/O imbalance among SSDs
- Throughput distribution per device
- Thread activity colored by throughput
- PCIe traffic breakdown per device
- What causes I/O communication to drop
- Socket interconnect traffic

## SPDK & DPDK Use Polling, Not Interrupts

- CPU always 100%, even if unloaded
- Intel® VTune™ Profiler identifies "empty" spinning

## I/O Summary

**SPDK Info**

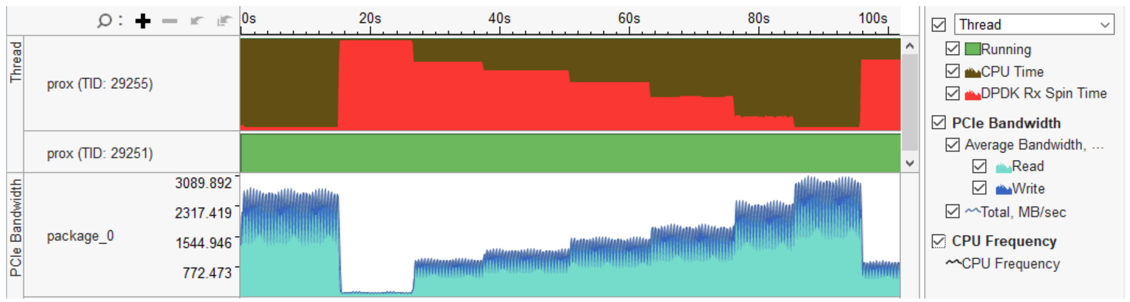
| | |
|---|---|
| Reads: | 3,595,310 |
| Read Bytes: | 28095.9 MB |
| Writes: | 3,595,563 |
| Written Bytes: | 28090.3 MB |
| SPDK Effective Time: | 66.403s |

## See Device Imbalance

**SPDK Info**

| | |
|---|---|
| Reads: | 3,595,310 |
| bdev_Nvme0n1_0x55b977f9fe50: | 2,012,685 |
| bdev_Nvme1n1_0x55b977f9aa40: | 1,582,481 |
| bdev_Nvme2n1_0x55b9780a50d0: | 73 |
| bdev_Nvme3n1_0x55b9780a52d0: | 71 |
| Read Bytes: | 28095.9 MB |
| bdev_Nvme0n1_0x55b977f9fe50: | 15727.2 MB |
| bdev_Nvme1n1_0x55b977f9aa40: | 12365.6 MB |
| bdev_Nvme2n1_0x55b9780a50d0: | 1.60547 MB |
| bdev_Nvme3n1_0x55b9780a52d0: | 1.47656 MB |

## See spin time (red)



# Get a Complete Picture of I/O Performance

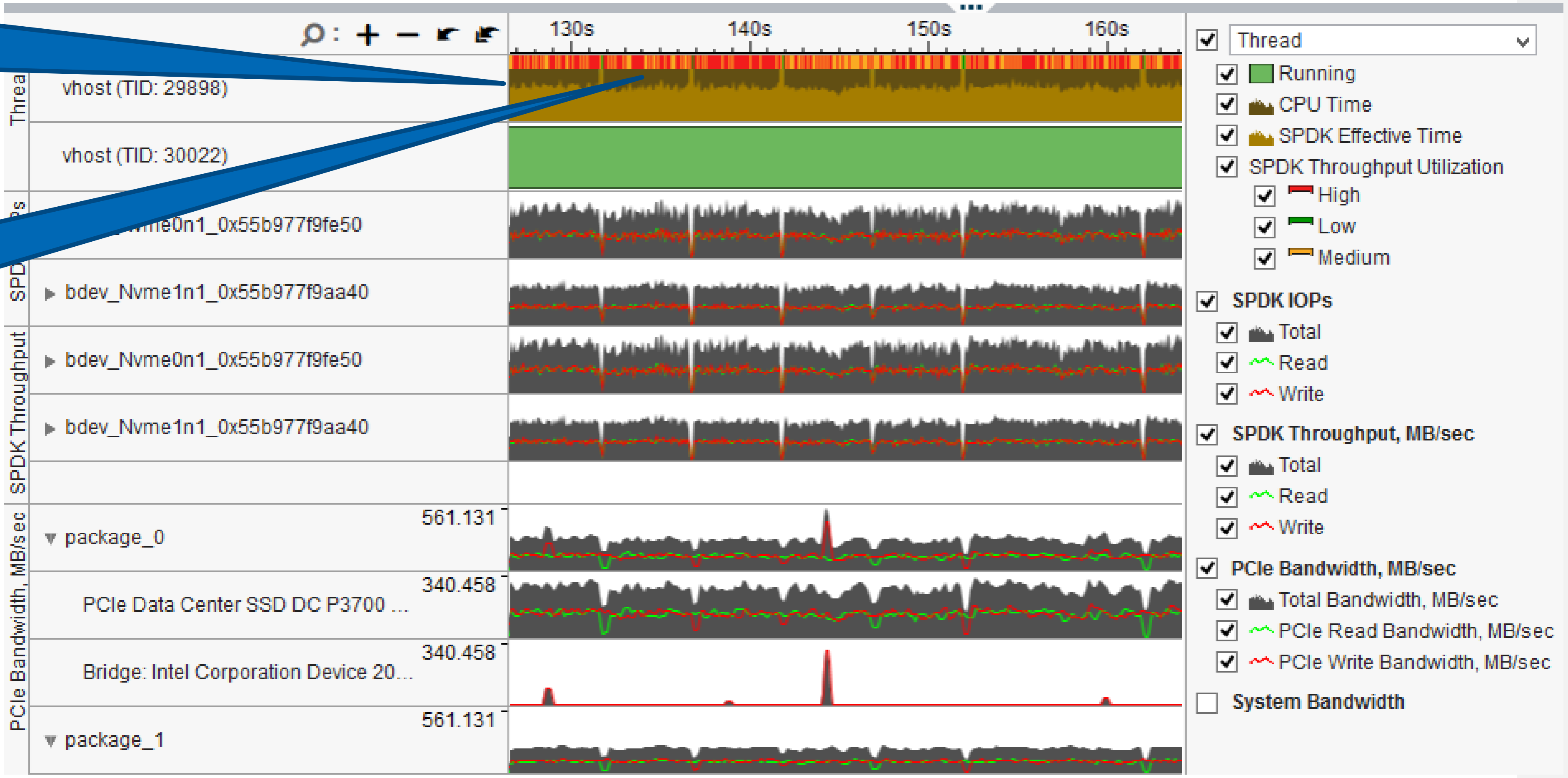
## Intel® VTune Profiler – SPDK & DPDK Performance Analysis

**SPDK Effective time:** CPU usage is accounted separately for spin-wait loops

In-**thread activity** colored according to Throughput utilization levels

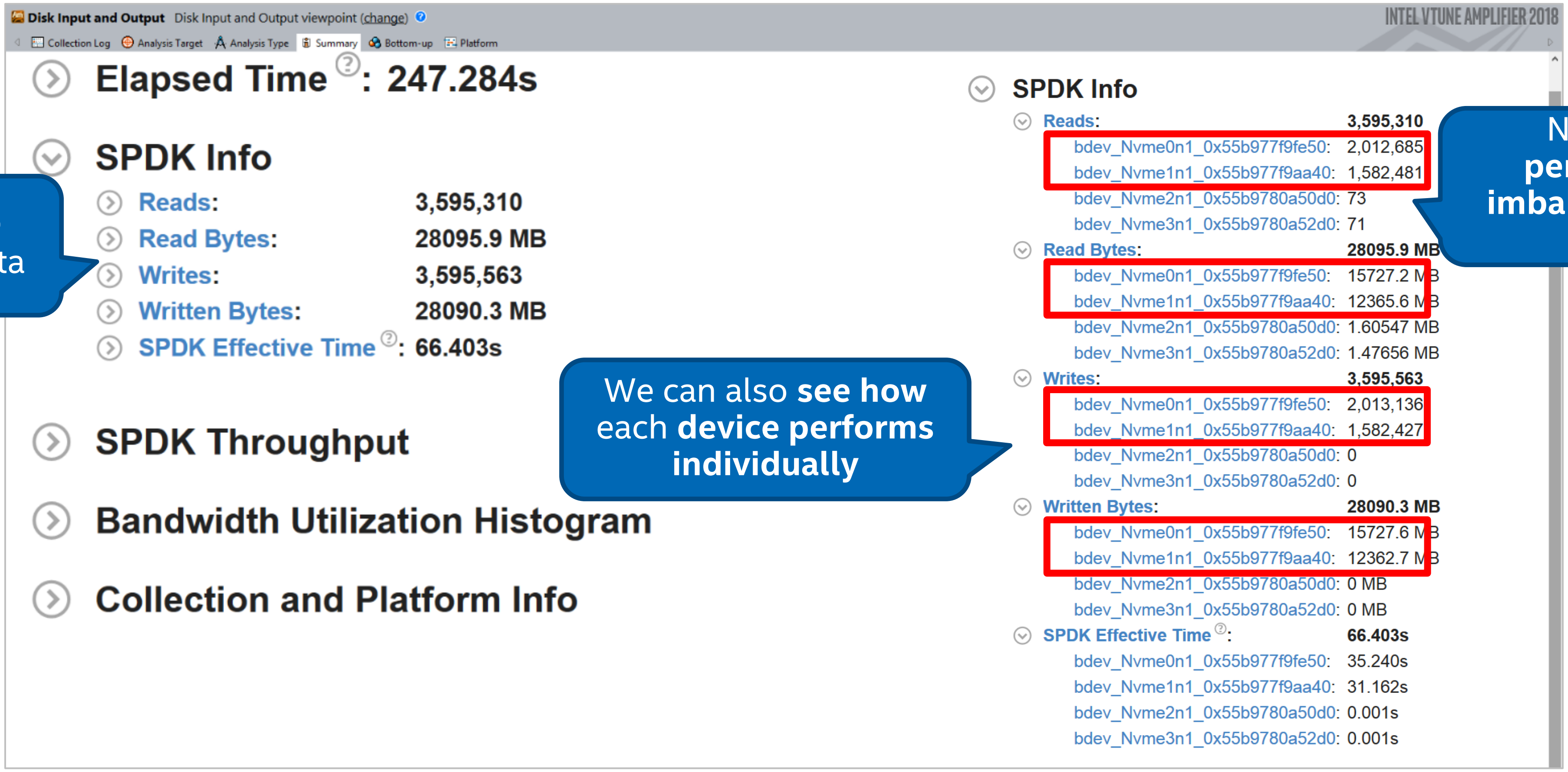
**I/O Statistic**: IOPs and Throughput

**PCIe** bandwidth with **traffic breakdown per physical device**



# See Overall I/O Performance Data

## Intel® VTune Profiler – SPDK & DPDK Performance Analysis



**Disk Input and Output** Disk Input and Output viewpoint (change)

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

INTEL VTUNE AMPLIFIER 2018

**Elapsed Time: 247.284s**

**SPDK Info**

| | |
|---|---|
| **Reads:** | **3,595,310** |
| **Read Bytes:** | **28095.9 MB** |
| **Writes:** | **3,595,563** |
| **Written Bytes:** | **28090.3 MB** |
| **SPDK Effective Time:** | **66.403s** |

**SPDK Throughput**

**Bandwidth Utilization Histogram**

**Collection and Platform Info**

**SPDK Info**

| | |
|---|---|
| **Reads:** | **3,595,310** |
| bdev_Nvme0n1_0x55b977f9fe50: | 2,012,685 |
| bdev_Nvme1n1_0x55b977f9aa40: | 1,582,481 |
| bdev_Nvme2n1_0x55b9780a50d0: | 73 |
| bdev_Nvme3n1_0x55b9780a52d0: | 71 |
| **Read Bytes:** | **28095.9 MB** |
| bdev_Nvme0n1_0x55b977f9fe50: | 15727.2 MB |
| bdev_Nvme1n1_0x55b977f9aa40: | 12365.6 MB |
| bdev_Nvme2n1_0x55b9780a50d0: | 1.60547 MB |
| bdev_Nvme3n1_0x55b9780a52d0: | 1.47656 MB |
| **Writes:** | **3,595,563** |
| bdev_Nvme0n1_0x55b977f9fe50: | 2,013,136 |
| bdev_Nvme1n1_0x55b977f9aa40: | 1,582,427 |
| bdev_Nvme2n1_0x55b9780a50d0: | 0 |
| bdev_Nvme3n1_0x55b9780a52d0: | 0 |
| **Written Bytes:** | **28090.3 MB** |
| bdev_Nvme0n1_0x55b977f9fe50: | 15727.6 MB |
| bdev_Nvme1n1_0x55b977f9aa40: | 12362.7 MB |
| bdev_Nvme2n1_0x55b9780a50d0: | 0 MB |
| bdev_Nvme3n1_0x55b9780a52d0: | 0 MB |
| **SPDK Effective Time:** | **66.403s** |
| bdev_Nvme0n1_0x55b977f9fe50: | 35.240s |
| bdev_Nvme1n1_0x55b977f9aa40: | 31.162s |
| bdev_Nvme2n1_0x55b9780a50d0: | 0.001s |
| bdev_Nvme3n1_0x55b9780a52d0: | 0.001s |

**See overall** I/O **performance** data

We can also **see how** each **device performs individually**

Notice **I/O performance imbalance** among SSDs

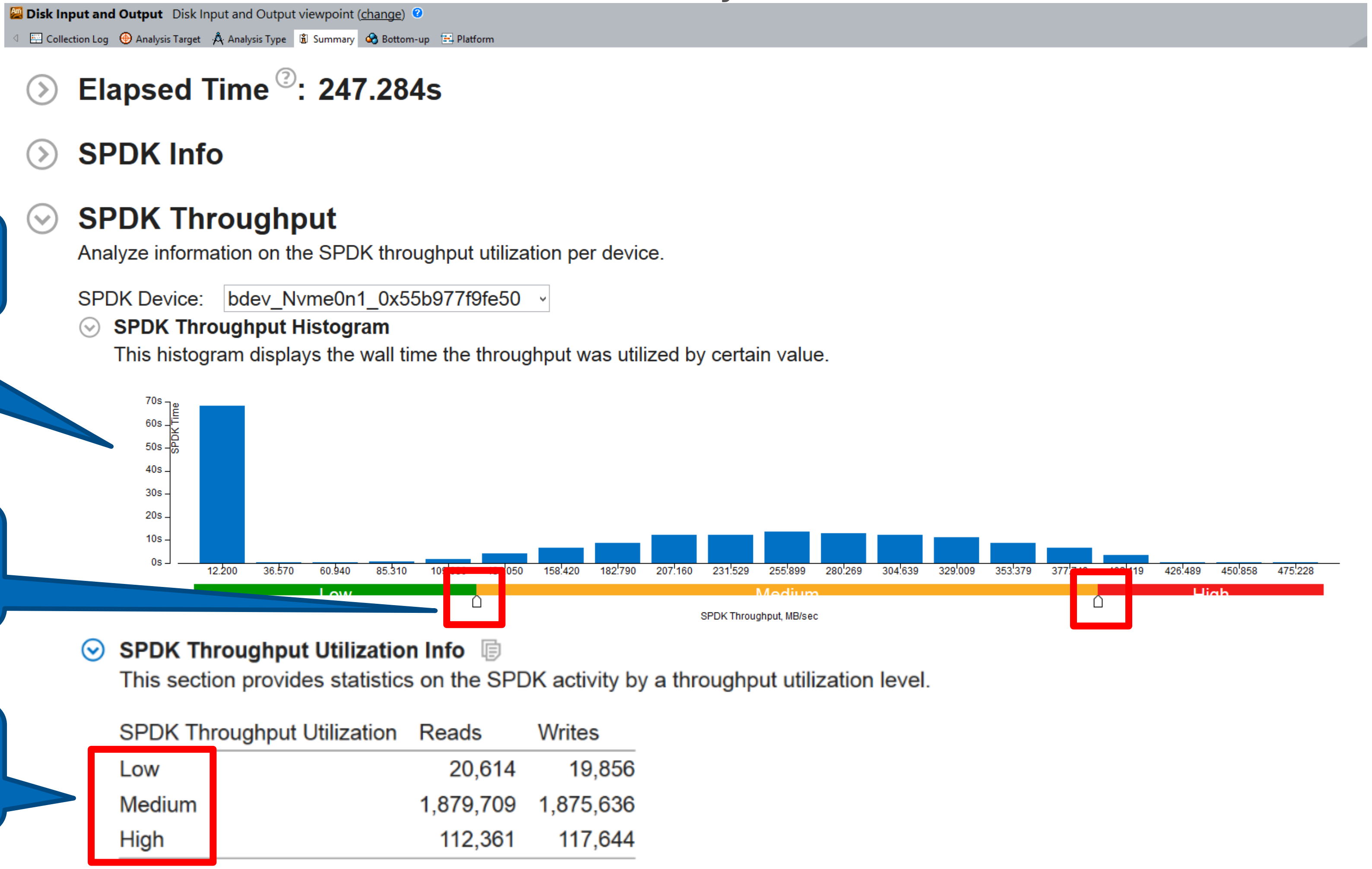
# View Results By Performance Targets

## Intel® VTune Profiler – SPDK & DPDK Performance Analysis

See **throughput distribution** for each device

Set performance targets and get **throughput utilization** for each range

All metrics can be seen on timeline within thread quanta… **see next slides**



**Elapsed Time: 247.284s**

**SPDK Info**

**SPDK Throughput**

Analyze information on the SPDK throughput utilization per device.

SPDK Device: bdev_Nvme0n1_0x55b977f9fe50

**SPDK Throughput Histogram**

This histogram displays the wall time the throughput was utilized by certain value.

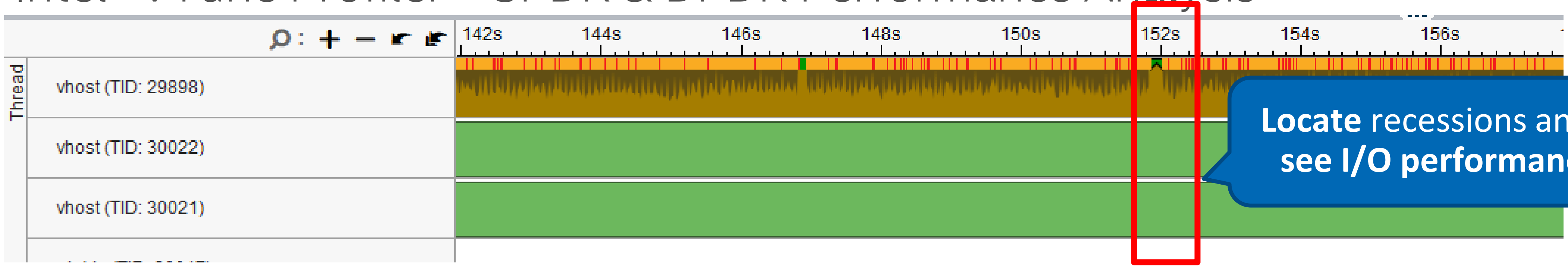
**SPDK Throughput Utilization Info**

This section provides statistics on the SPDK activity by a throughput utilization level.

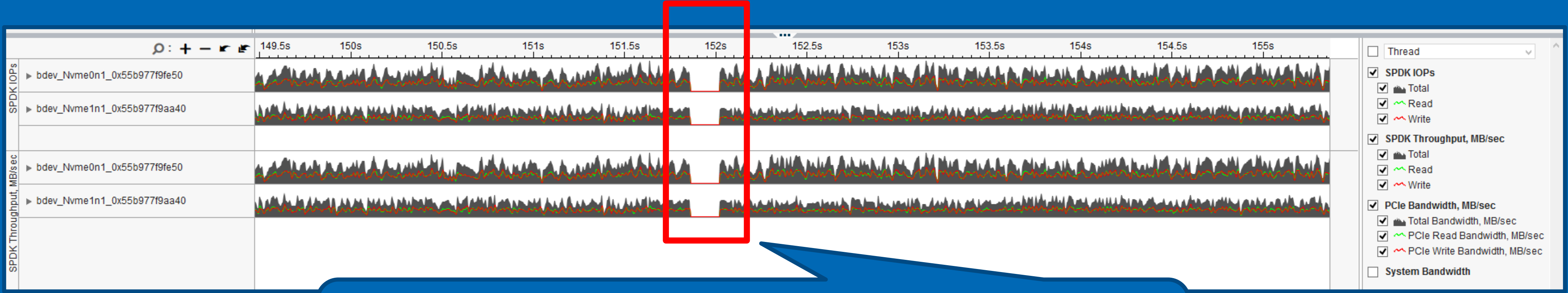
| SPDK Throughput Utilization | Reads | Writes |
|---|---|---|
| Low | 20,614 | 19,856 |
| Medium | 1,879,709 | 1,875,636 |
| High | 112,361 | 117,644 |

# Diagnose Recessions

## Intel® VTune Profiler – SPDK & DPDK Performance Analysis



**Locate** recessions and **filter in** to **see I/O performance changes**



**Check which code** was executed and **caused I/O communication being dropped for ~160ms!**

# Get a Platform Perspective

## Intel® VTune Profiler – SPDK & DPDK Performance Analysis

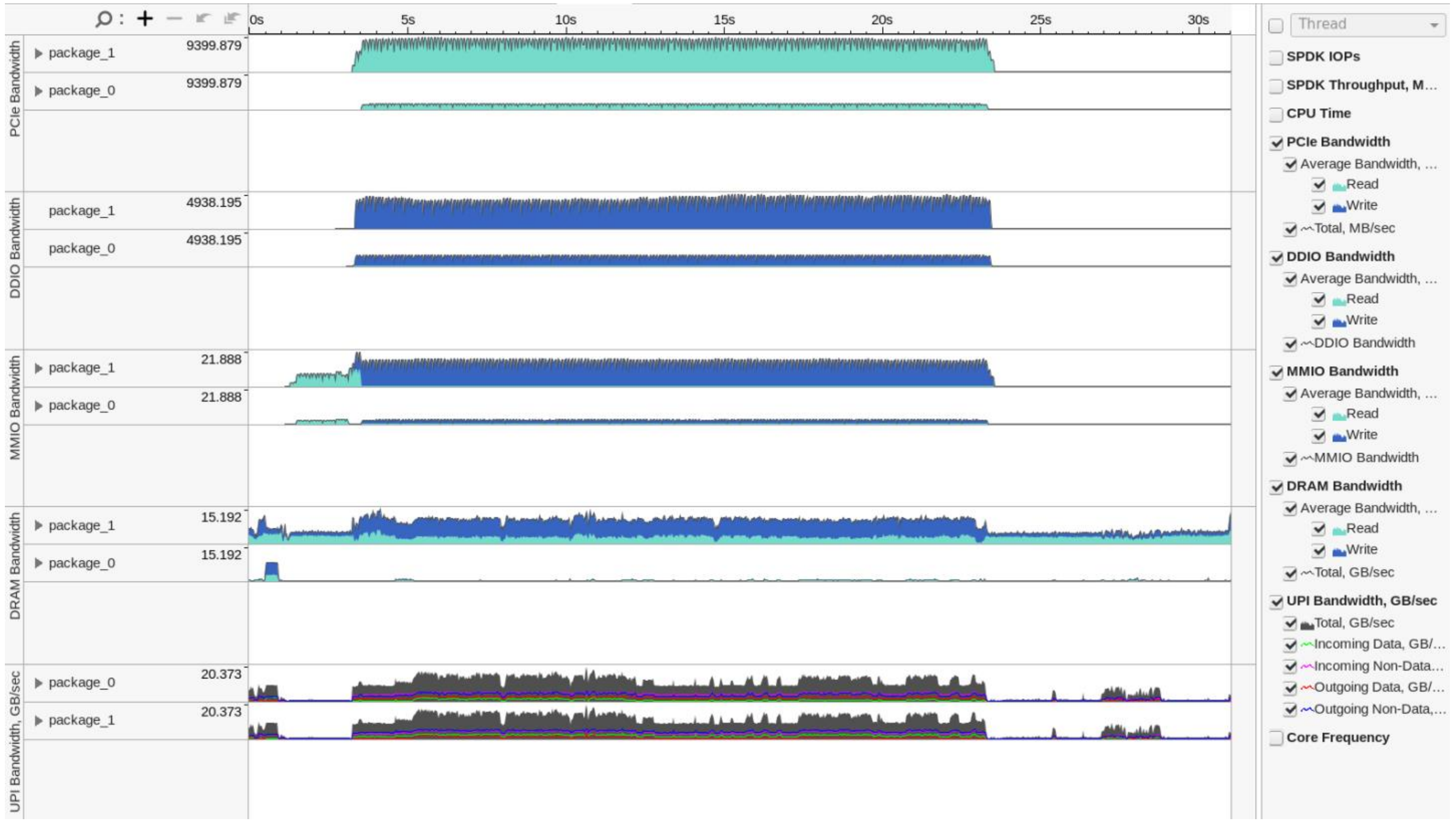
**PCIe** bandwidth with **traffic breakdown per physical device**

**Intel DDIO** misses resulted in **write back to RAM**

**Intel MMIO** traffic. **Avoid Reads** and control Writes

**DRAM bandwidth**

**Socket interconnect traffic**



# Storage Device Analysis (HDD, SATA or NVMe SSD)

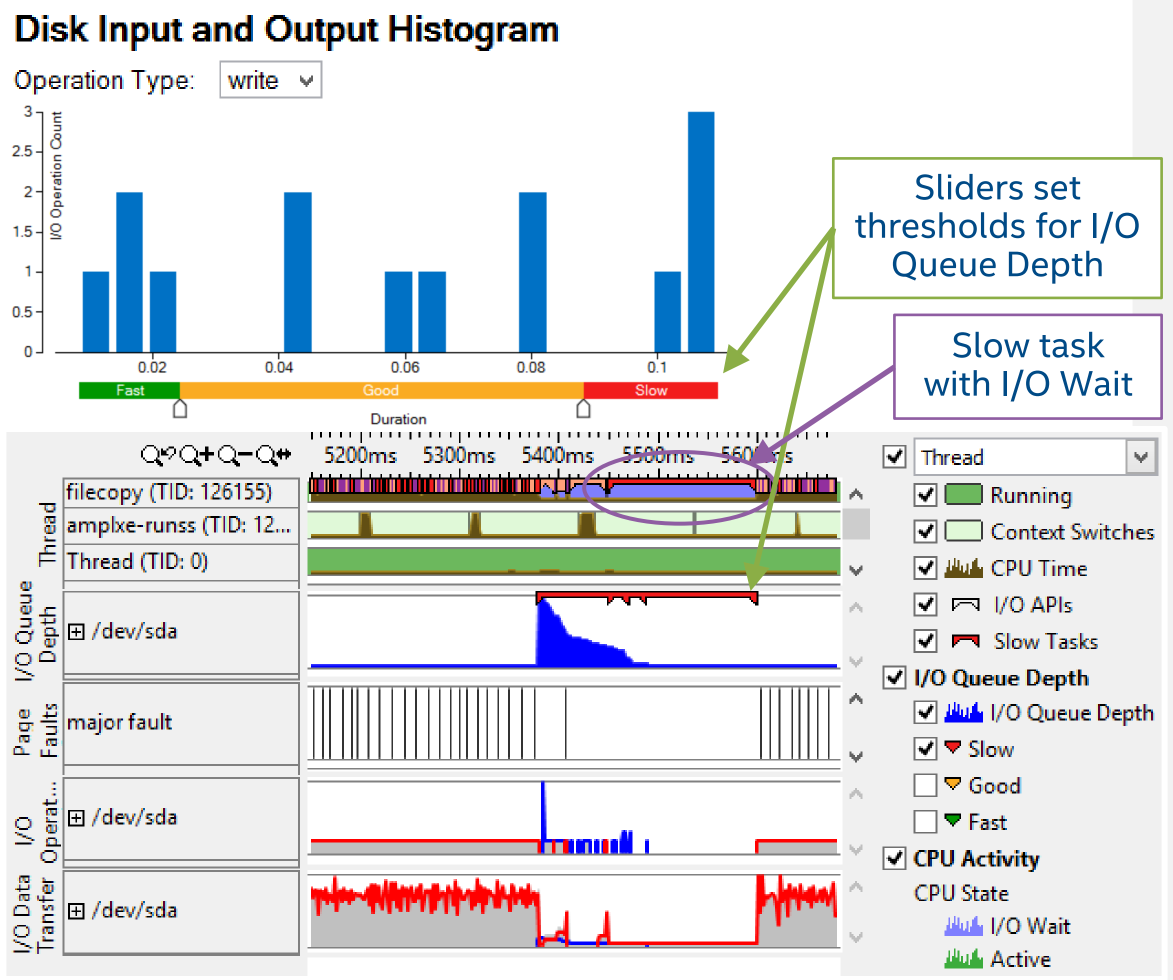
Intel® VTune™ Profiler

## 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™ Profiler

## Which Tool Should I Use?

# Performance is Many Things

CPI GHZ SIMD MULTI-CORE CACHE MEMORY IO

# Many Developer Roles

Contribute to overall performance

DESIGN

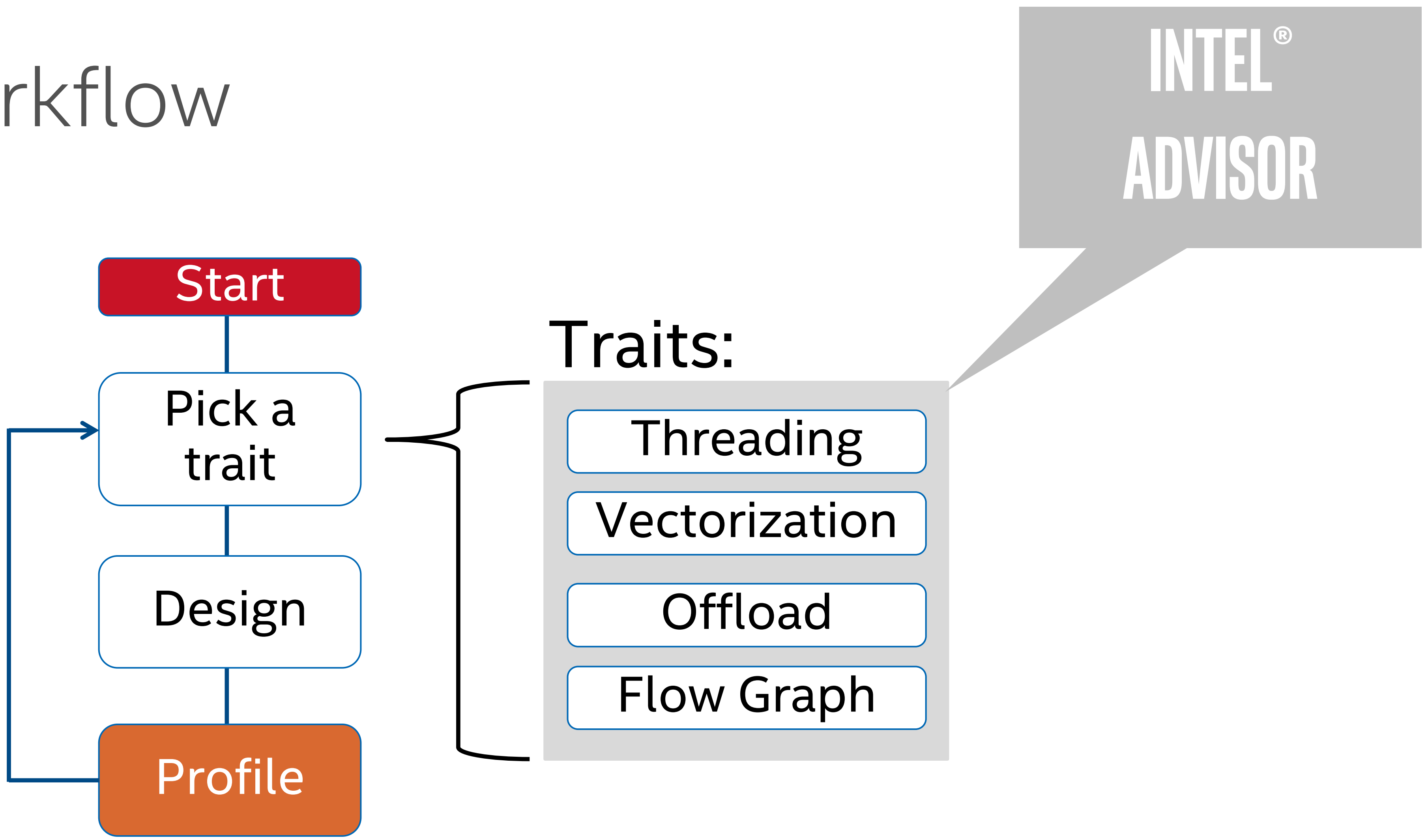
CODE

APPLICATION PERFORMANCE

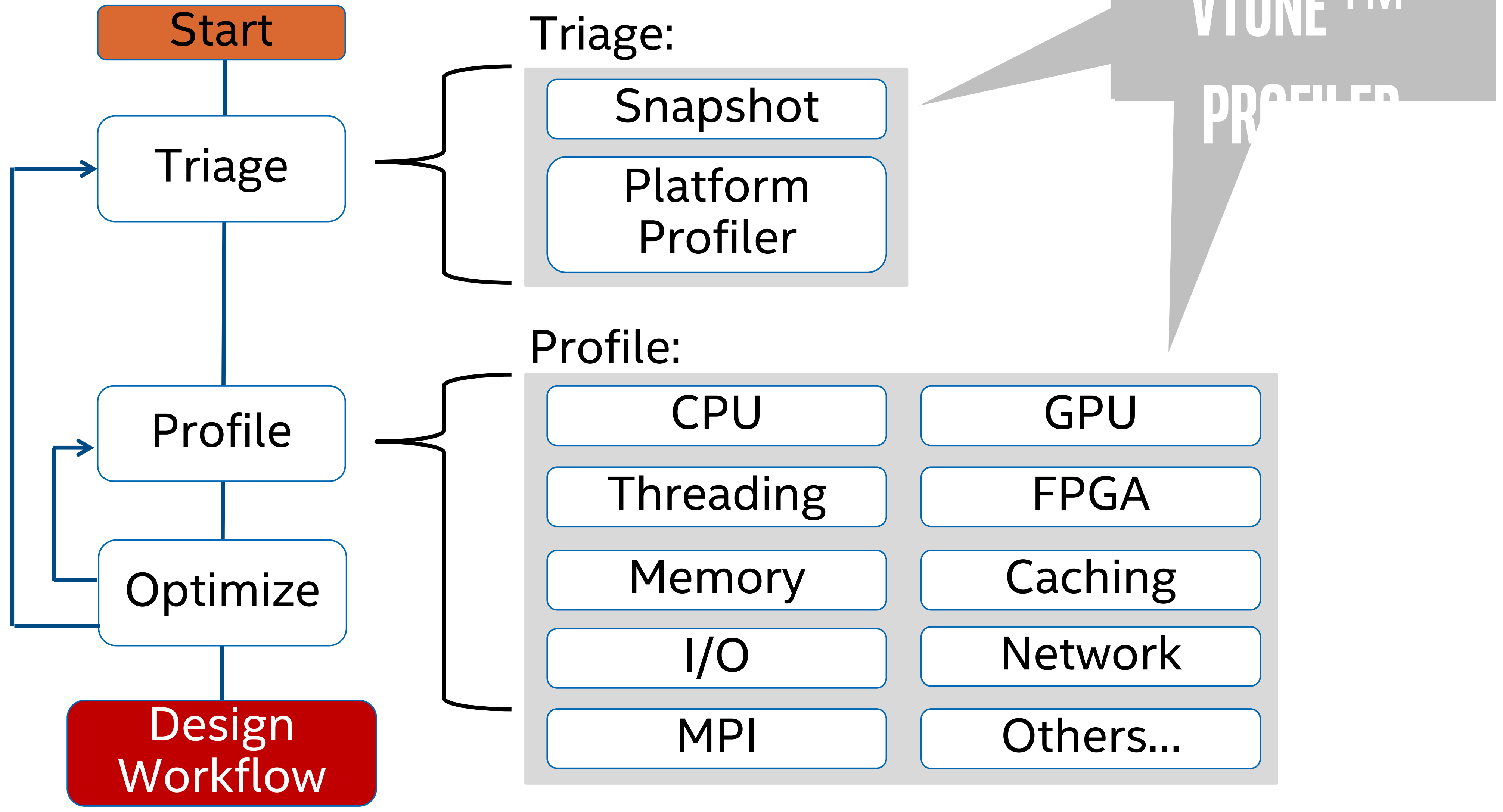
SYSTEM PERFORMANCE

- Role Specific Workflows

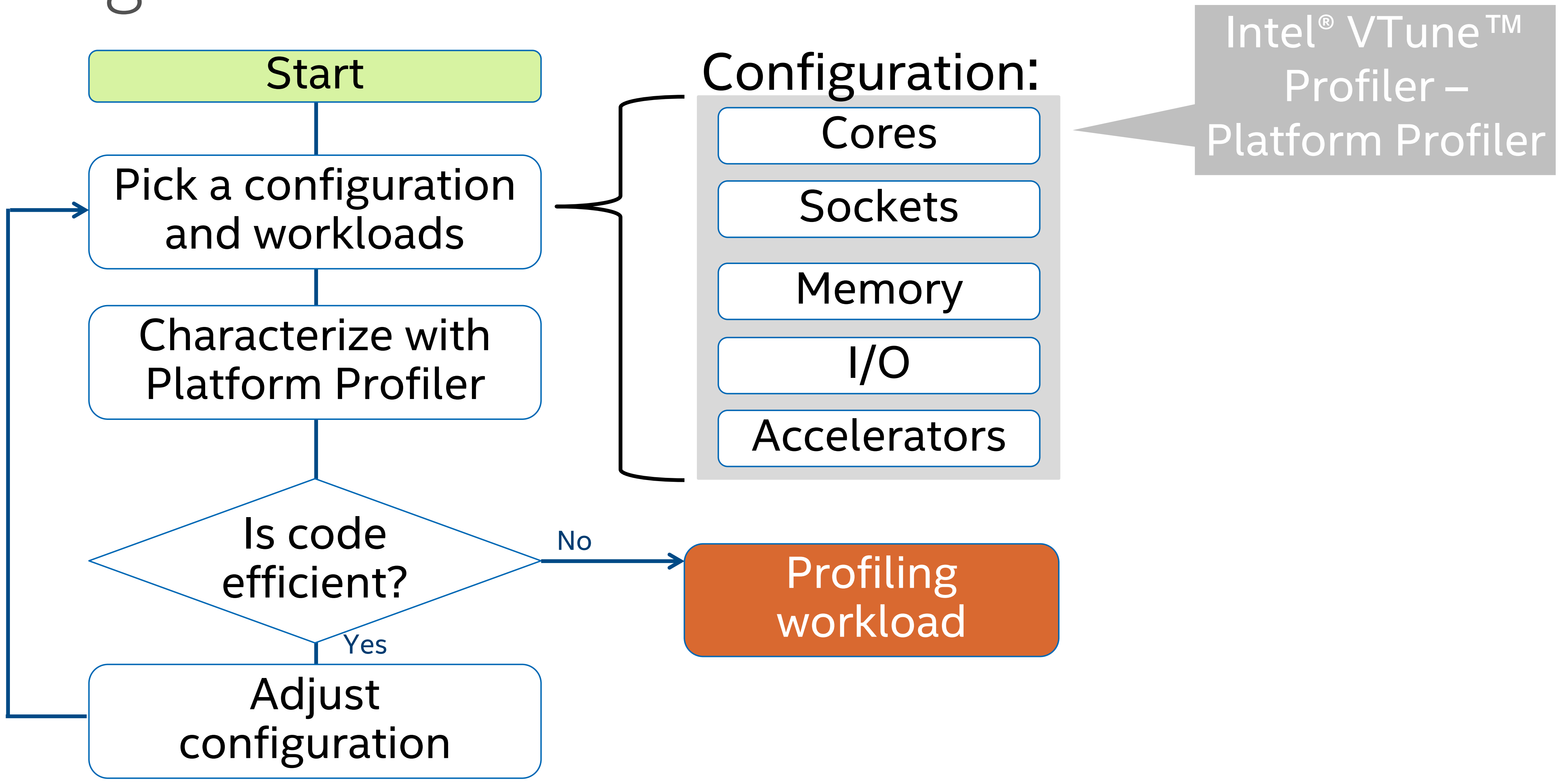
# Design Workflow



# Profiling Workflow

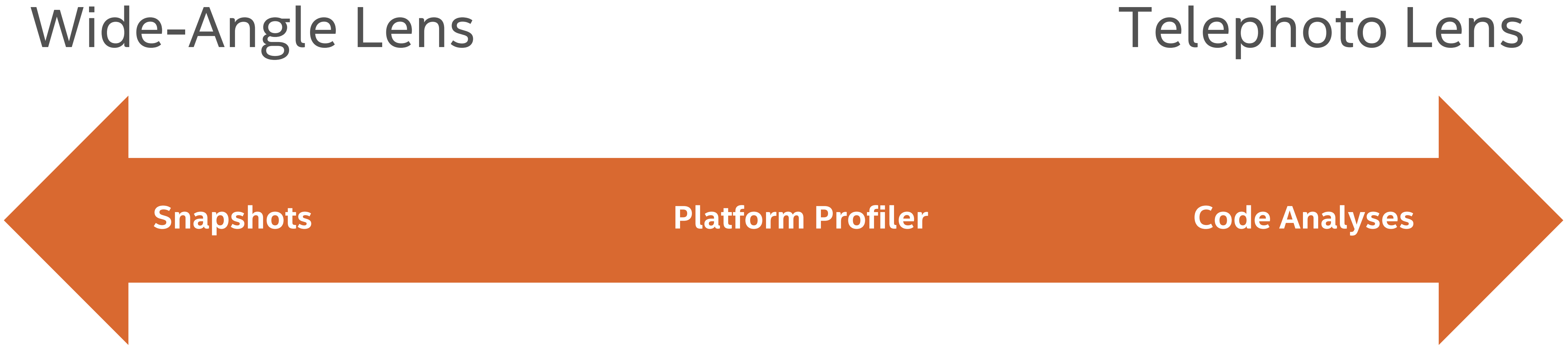


# Configuration Workflow



# Understanding Intel Analysis Toolbox

- *Design* of software aided by Intel® Advisor
- *Correctness* checked by Intel® Inspector
- *Performance* profiled, bottlenecks analyzed, and software/HW context shown by Intel® VTune™ Profiler:



# Analysis Tools Overview

**Intel® VTune™ Profiler**
Performance Profiler

**Intel® Advisor**
Design and optimize vectorization, threading, accelerator offload and flow graphs.

**Intel® Inspector**
Memory & Thread Debugger

**Intel® Trace Analyzer and Collector**
MPI Profiling for Cluster Applications

# Rich Set of Profiling Capabilities for Multiple Markets

Intel® VTune™ Profiler

**Single Thread**

Optimize single-threaded performance.

**Multithreaded**

Effectively use all available cores.

**System**

See a system-level view of application performance.

**Media & OpenCL™ Applications**

Deliver high-performance image and video processing pipelines.

**HPC & CLoud**

Access specialized, in-depth analyses for HPC and cloud computing.

**Memory & Storage Management**

Diagnose memory, storage, and data plane bottlenecks.

**Analyze & Filter Data**

Mine data for answers.

**Environment**

Fits your environment and workflow.