ATPESC 21

Intel® VTune Profiler and Intel® Advisor Overview

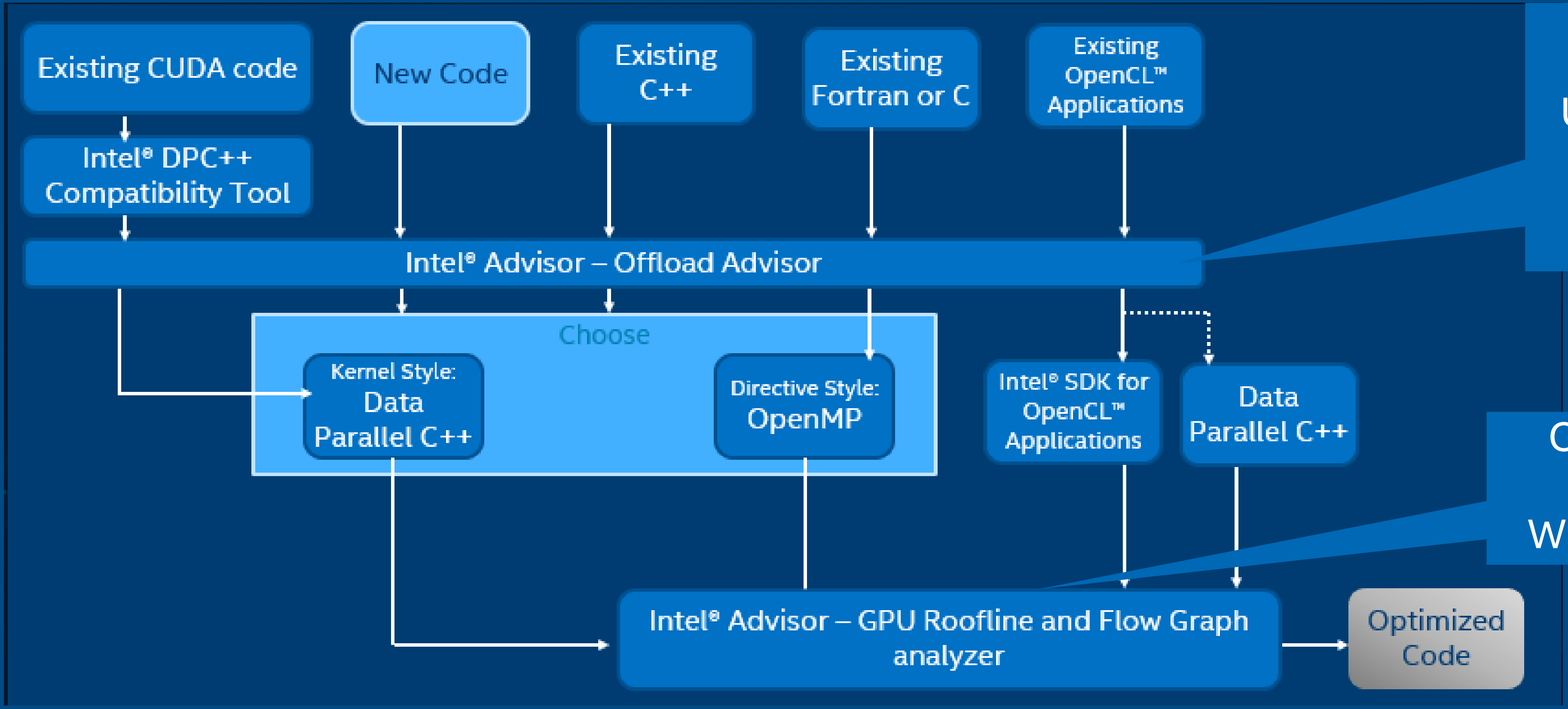
Kevin O'Leary

intel®

Agenda

1. Intel® VTune Profiler overview
2. Intel® VTune Profiler for GPU
3. Offload Modeling with Intel® Advisor
4. GPU Roofline with Intel® Advisor

Using Intel® Analyzers to increase performance


Existing CUDA code
Intel® DPC++ Compatibility Tool
New Code
Existing C++
Existing Fortran or C
Existing OpenCL™ Applications
Intel® Advisor – Offload Advisor
Choose
Kernel Style: Data Parallel C++
Directive Style: OpenMP
Intel® SDK for OpenCL™ Applications
Data Parallel C++
Intel® Advisor – GPU Roofline and Flow Graph analyzer
Optimized Code

Use Offload Advisor to find kernels to offload

Optimize your kernels

With Advisor and VTune

Intel Analysis Tools for GPU Compute

Intel® Advisor

- Offload Advisor
  - Identify high-impact opportunities to offload
  - Detect bottlenecks and key bounding factors
  - Get your code ready even before you have the hardware by modeling performance, headroom and bottlenecks
- Roofline Analysis
  - See performance headroom against hardware limitations
  - Determine performance optimization strategy by identifying bottlenecks and which optimizations will payoff the most
  - Visualize optimization progress
- Flow Graph Analyzer
  - Visualize your CPU/GPU code and get recommendations for the CPU device

Intel® VTune™ Profiler

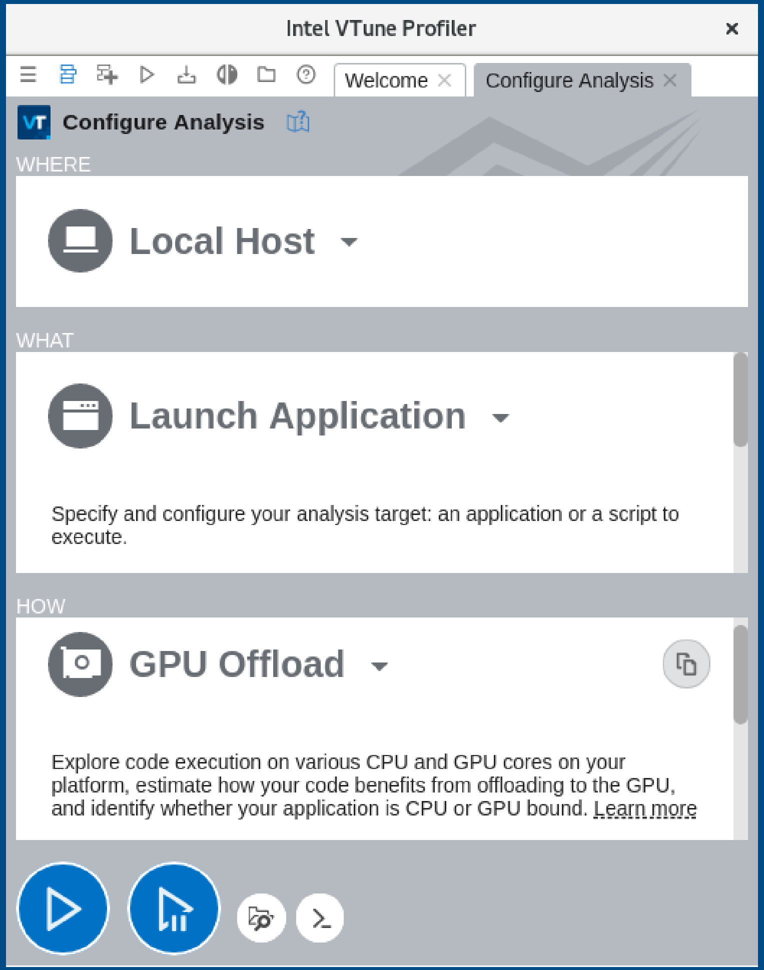
- Offload Performance Tuning
  - Explore code execution on various CPU and GPU cores on your platform
  - Correlate CPU and GPU activity
  - Identify whether your application is GPU or CPU bound
- HPC Performance Characterization
  - Identify whether the OpenMP application offloads work to GPU effectively
- GPU Compute/Media Hotspots
  - Analyze the most time-consuming GPU kernels, characterize GPU usage based on GPU hardware metrics
  - GPU code performance at the source-line level and kernel assembly level

Intel® VTune™ Profiler

Intel® VTune™ Profiler GUI: quick overview

- GUI provides 3 panes to configure the analysis:
  - **WHERE** is used to specify an analysis system
  - **WHAT** is used to specify an analysis target
  - **HOW** is used to select an analysis type

VTune Profiler documentation: WHERE: Analysis system, WHAT: Analysis Target and HOW: Analysis Types



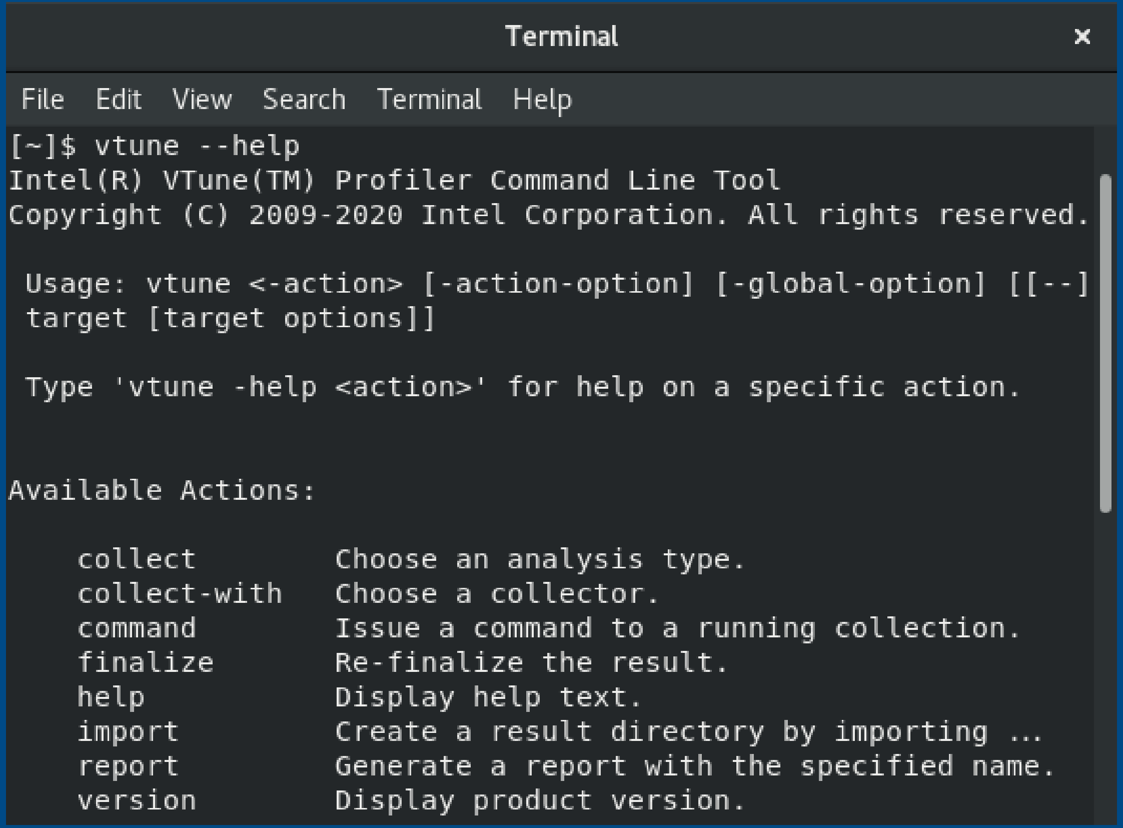
Specify and configure your analysis target: an application or a script to execute.

Explore code execution on various CPU and GPU cores on your platform, estimate how your code benefits from offloading to the GPU, and identify whether your application is CPU or GPU bound. Learn more

VTune CLI: quick overview

- CLI has its own help with several levels:
  - vtune -help
  - vtune -help collect
  - vtune -help collect gpu-offload
- Run collection:
  - vtune -collect <*analysis_type*> <*target*>
- Generate a report:
  - vtune -report <*report_name*> -r <*result_dir*>

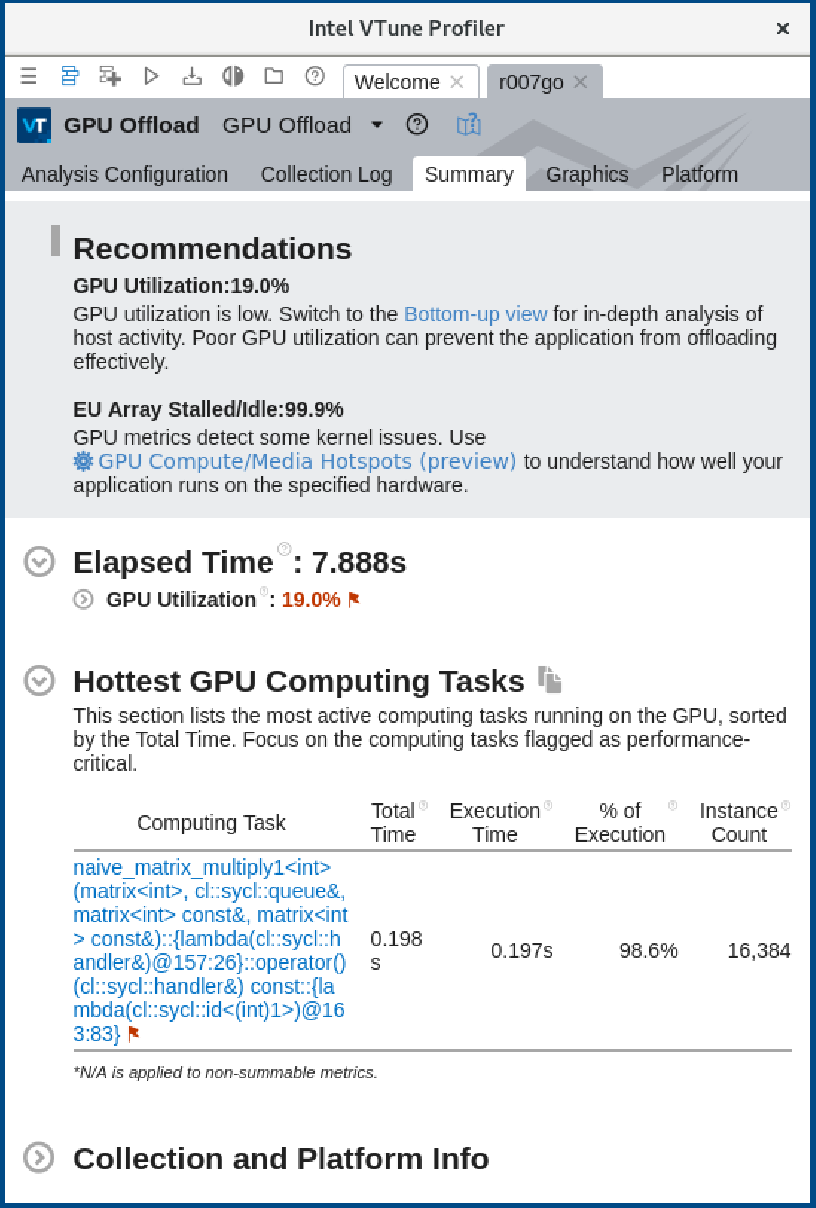
VTune Profiler documentation: Command Line Interface



```
[~]$ vtune --help
Intel(R) VTune(TM) Profiler Command Line Tool
Copyright (C) 2009-2020 Intel Corporation. All rights reserved.

 Usage: vtune <-action> [-action-option] [-global-option] [[--]
 target [target options]]

 Type 'vtune -help <action>' for help on a specific action.


Available Actions:

    collect        Choose an analysis type.
    collect-with   Choose a collector.
    command        Issue a command to a running collection.
    finalize       Re-finalize the result.
    help           Display help text.
    import         Create a result directory by importing ...
    report         Generate a report with the specified name.
    version        Display product version.
```

GPU offload

- Helps to identify whether the application offloads work to GPU effectively.
- Can be used to profile OpenCL, Level0 and Intel Media SDK based applications or DPC++ and OpenMP applications that offload work on Intel GPU.



Recommendations

GPU Utilization:19.0%

GPU utilization is low. Switch to the Bottom-up view for in-depth analysis of host activity. Poor GPU utilization can prevent the application from offloading effectively.

EU Array Stalled/Idle:99.9%

GPU metrics detect some kernel issues. Use GPU Compute/Media Hotspots (preview) to understand how well your application runs on the specified hardware.

Elapsed Time: 7.888s

GPU Utilization: 19.0% ⚑

Hottest GPU Computing Tasks

This section lists the most active computing tasks running on the GPU, sorted by the Total Time. Focus on the computing tasks flagged as performance-critical.

Computing Task	Total Time	Execution Time	% of Execution	Instance Count
naive_matrix_multiply1<int>(matrix<int>, cl::sycl::queue&, matrix<int> const&, matrix<int> const&)::{lambda(cl::sycl::handler&)@157:26}::operator()(cl::sycl::handler&) const::{lambda(cl::sycl::id<(int)1>)@163:83} ⚑	0.198 s	0.197s	98.6%	16,384

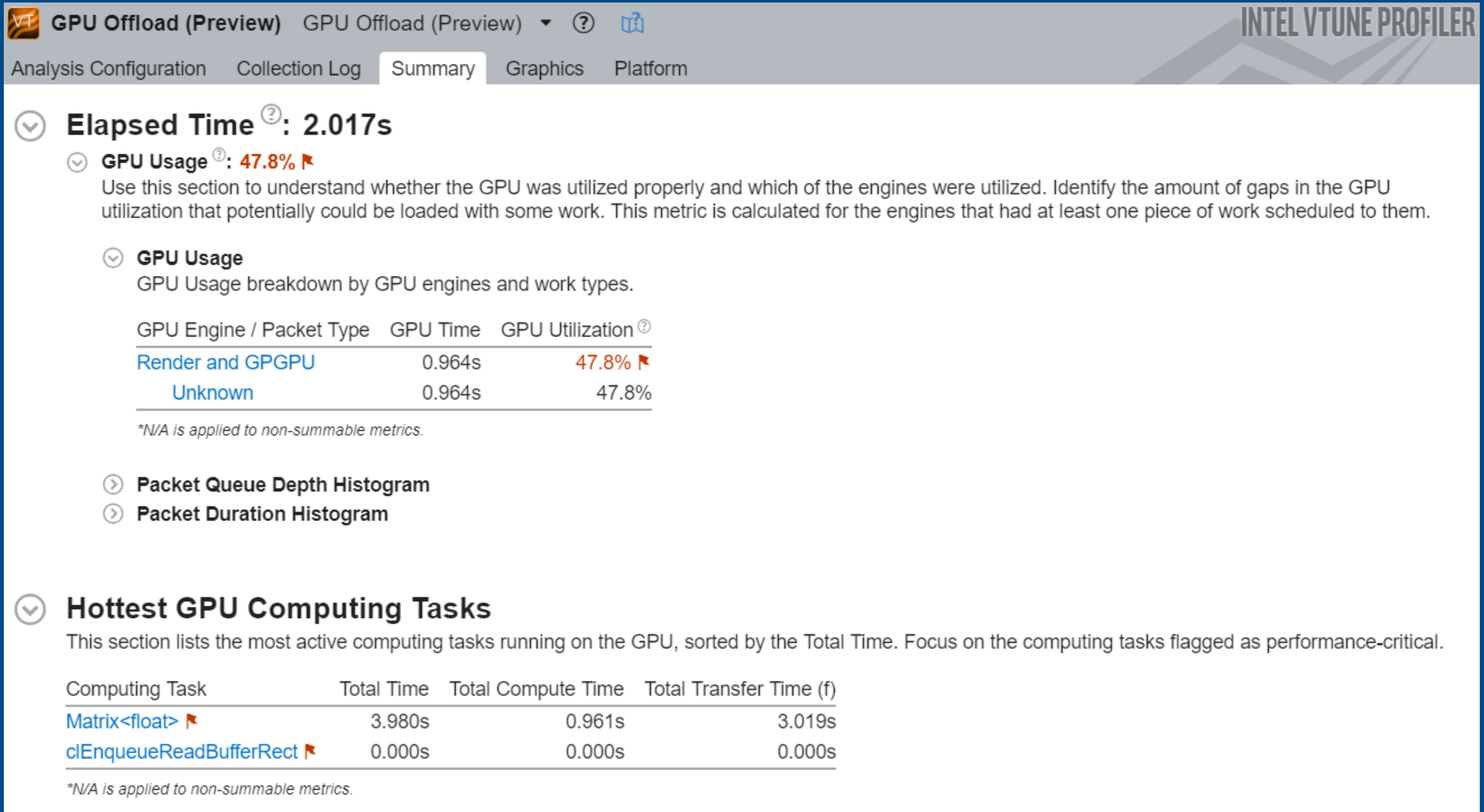
**N/A is applied to non-summable metrics.*

Collection and Platform Info

VTune Profiler documentation: GPU Offload Analysis

Optimize your GPU usage using Intel® VTune Profiler

GPU offload



Elapsed Time: 2.017s

GPU Usage: 47.8% ⚑

Use this section to understand whether the GPU was utilized properly and which of the engines were utilized. Identify the amount of gaps in the GPU utilization that potentially could be loaded with some work. This metric is calculated for the engines that had at least one piece of work scheduled to them.

GPU Usage

GPU Usage breakdown by GPU engines and work types.

GPU Engine / Packet Type	GPU Time	GPU Utilization
Render and GPGPU	0.964s	47.8% ⚑
Unknown	0.964s	47.8%

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

Packet Queue Depth Histogram

Packet Duration Histogram

Hottest GPU Computing Tasks

This section lists the most active computing tasks running on the GPU, sorted by the Total Time. Focus on the computing tasks flagged as performance-critical.

Computing Task	Total Time	Total Compute Time	Total Transfer Time (f)
Matrix<float> ⚑	3.980s	0.961s	3.019s
clEnqueueReadBufferRect ⚑	0.000s	0.000s	0.000s

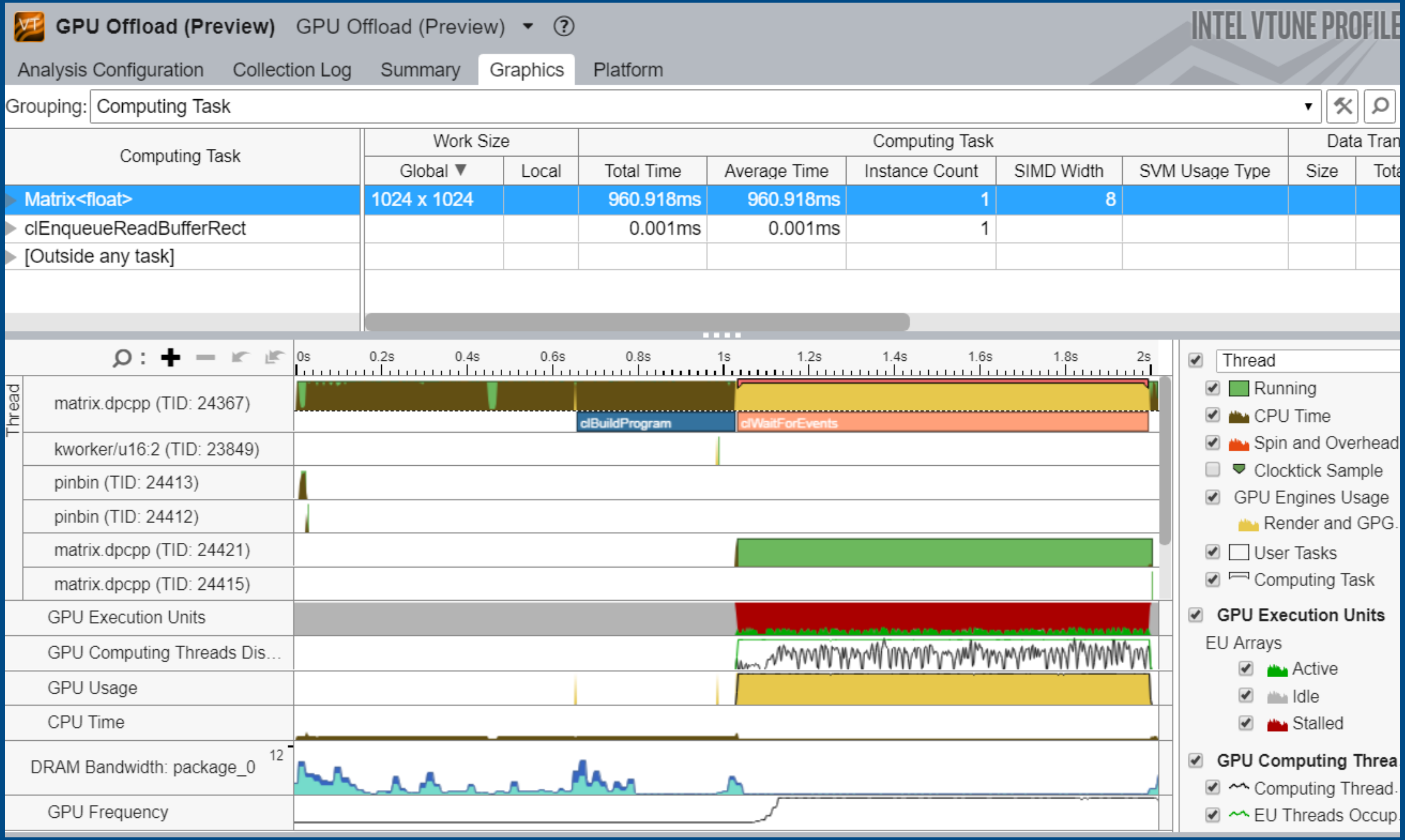
**N/A is applied to non-summable metrics.*

This analysis enables you to:

Identify how effectively your application uses DPC++ or OpenCL kernels.

Explore GPU usage and analyze a software queue for GPU engines at each moment of time

Optimize your GPU usage using Intel® VTune™ Profiler
GPU offload



GPU Offload (Preview) GPU Offload (Preview)

Grouping: Computing Task

Computing Task	Work Size		Computing Task					Data Tran	
	Global ▼	Local	Total Time	Average Time	Instance Count	SIMD Width	SVM Usage Type	Size	Tota
Matrix<float>	1024 x 1024		960.918ms	960.918ms	1	8			
clEnqueueReadBufferRect			0.001ms	0.001ms	1				
[Outside any task]									

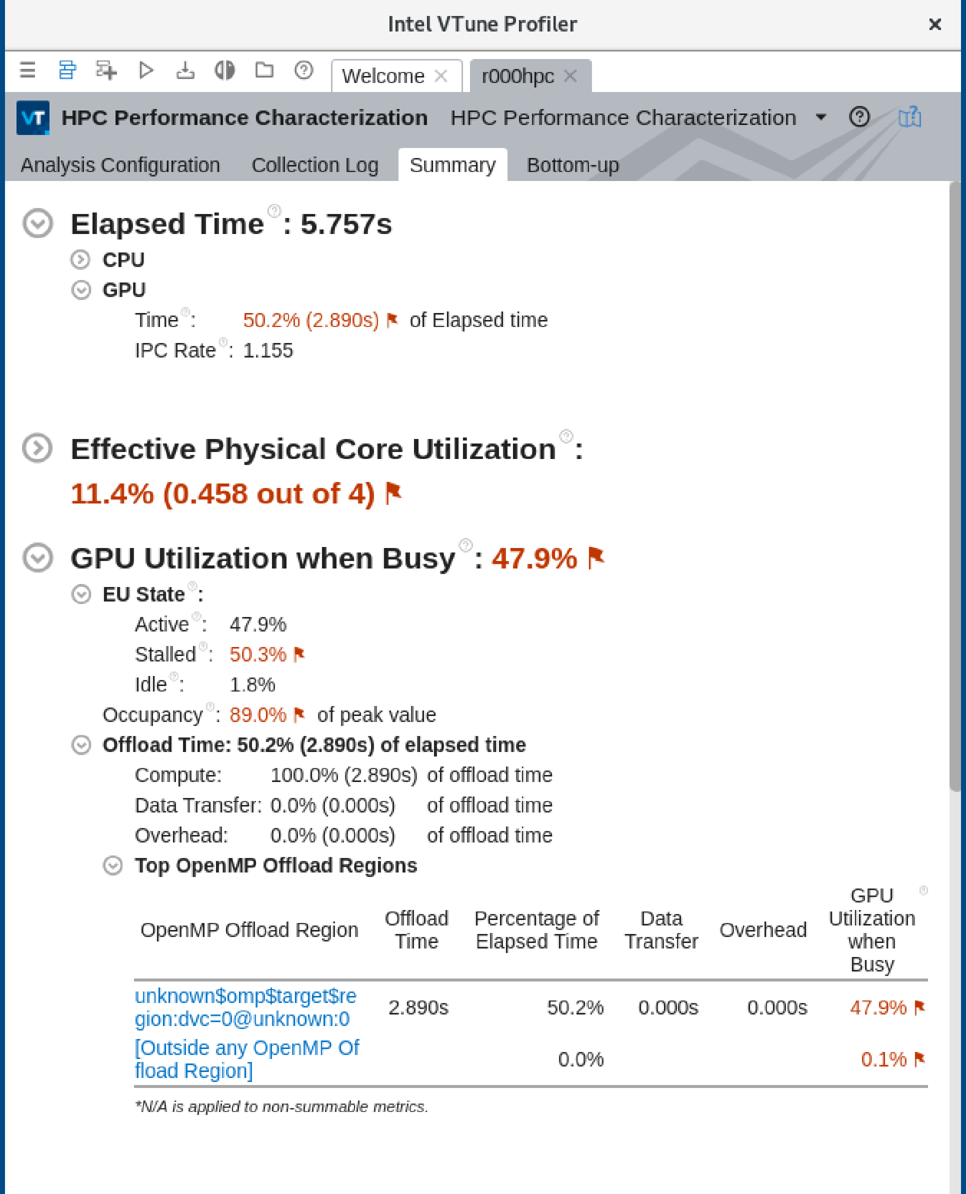
Use the GPU offload features Intel® VTune™ Profiler to see how effectively we are using our GPU.

VTune Profiler shows a synchronized time line between the CPU and GPU. GPU offload does indicate that our GPU execution units are stalling as indicated by the dark read bar in our timeline.

HPC Performance Characterization

- Helps to identify whether the OpenMP application offloads work to GPU effectively.
- The main difference with GPU Offload analysis is that the data is collected through OMPT interface.

VTune Profiler documentation: HPC Performance Characterization Analysis



Elapsed Time: 5.757s

CPU

GPU

Time: 50.2% (2.890s) ⚑ of Elapsed time

IPC Rate: 1.155

Effective Physical Core Utilization:

11.4% (0.458 out of 4) ⚑

GPU Utilization when Busy: 47.9% ⚑

EU State:

Active: 47.9%

Stalled: 50.3% ⚑

Idle: 1.8%

Occupancy: 89.0% ⚑ of peak value

Offload Time: 50.2% (2.890s) of elapsed time

Compute: 100.0% (2.890s) of offload time

Data Transfer: 0.0% (0.000s) of offload time

Overhead: 0.0% (0.000s) of offload time

Top OpenMP Offload Regions

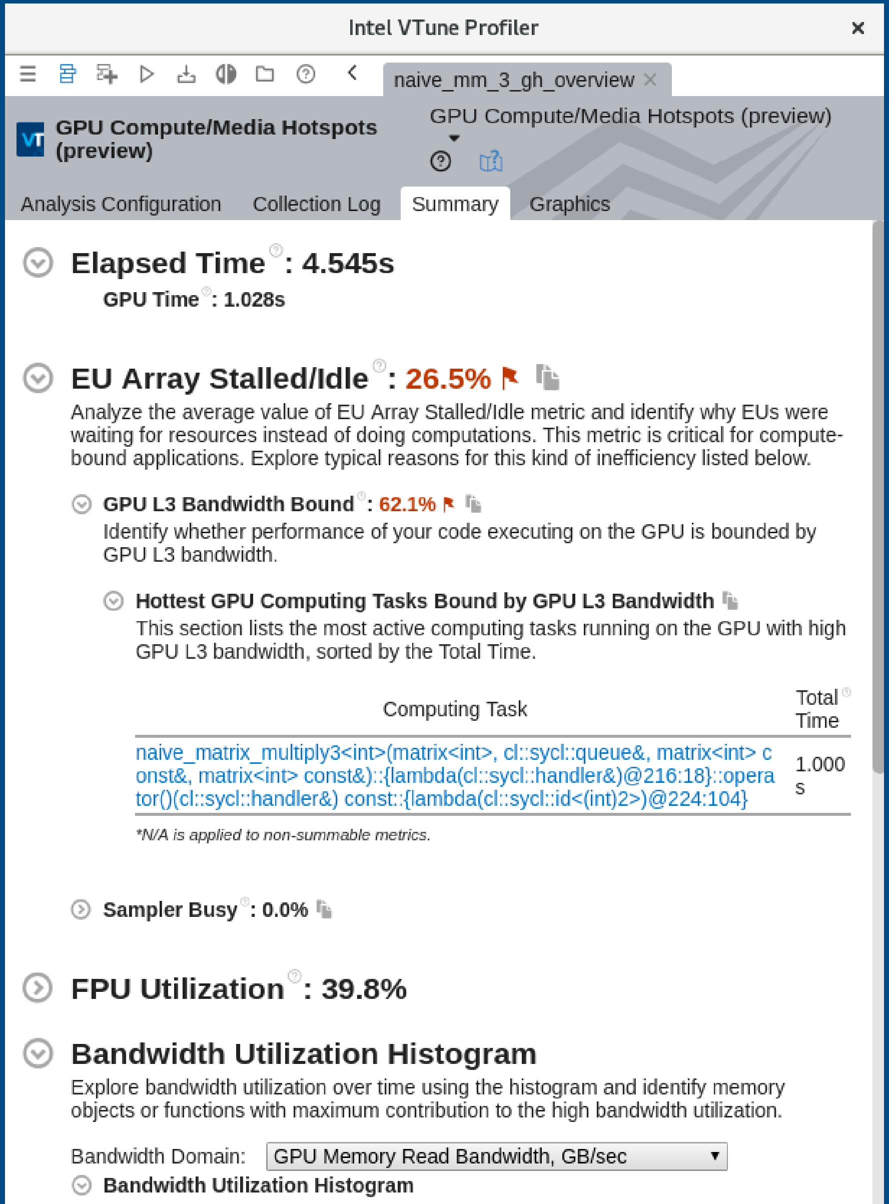
OpenMP Offload Region	Offload Time	Percentage of Elapsed Time	Data Transfer	Overhead	GPU Utilization when Busy
unknown$omp$target$region:dvc=0@unknown:0	2.890s	50.2%	0.000s	0.000s	47.9% ⚑
[Outside any OpenMP Offload Region]		0.0%			0.1% ⚑

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

GPU Compute/Media Hotspots

- Allows to analyze the most time-consuming GPU kernels, characterize GPU usage based on GPU hardware metrics, identify performance issues caused by memory latency or inefficient kernel algorithms, and analyze GPU instruction frequency per certain instruction types.

VTune Profiler documentation: [GPU Compute/Media Hotspots Analysis]()



Elapsed Time: 4.545s

GPU Time: 1.028s

EU Array Stalled/Idle: 26.5%

Analyze the average value of EU Array Stalled/Idle metric and identify why EUs were waiting for resources instead of doing computations. This metric is critical for compute-bound applications. Explore typical reasons for this kind of inefficiency listed below.

GPU L3 Bandwidth Bound: 62.1%

Identify whether performance of your code executing on the GPU is bounded by GPU L3 bandwidth.

Hottest GPU Computing Tasks Bound by GPU L3 Bandwidth

This section lists the most active computing tasks running on the GPU with high GPU L3 bandwidth, sorted by the Total Time.

Computing Task	Total Time
naive_matrix_multiply3<int>(matrix<int>, cl::sycl::queue&, matrix<int> const&, matrix<int> const&)::{lambda(cl::sycl::handler&)@216:18}::operator()(cl::sycl::handler&) const::{lambda(cl::sycl::id<(int)2>)@224:104}	1.000 s

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

Sampler Busy: 0.0%

FPU Utilization: 39.8%

Bandwidth Utilization Histogram

Explore bandwidth utilization over time using the histogram and identify memory objects or functions with maximum contribution to the high bandwidth utilization.

Bandwidth Domain: GPU Memory Read Bandwidth, GB/sec

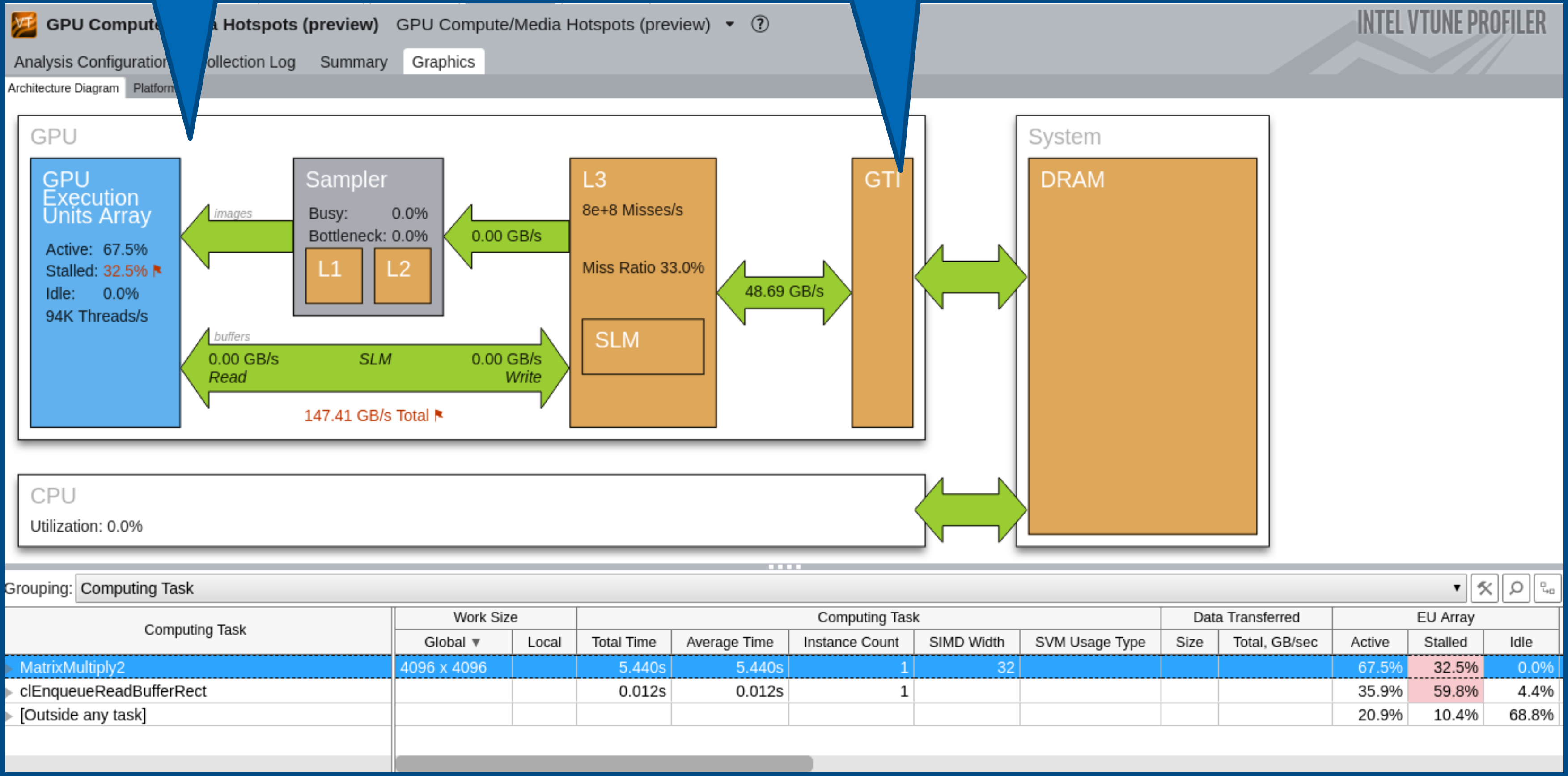
Bandwidth Utilization Histogram

Optimize your GPU usage using Intel® VTune Profiler

GPU hotspots

Active vs. Idle EU activity

Bandwidth at multiple levels



Grouping: Computing Task

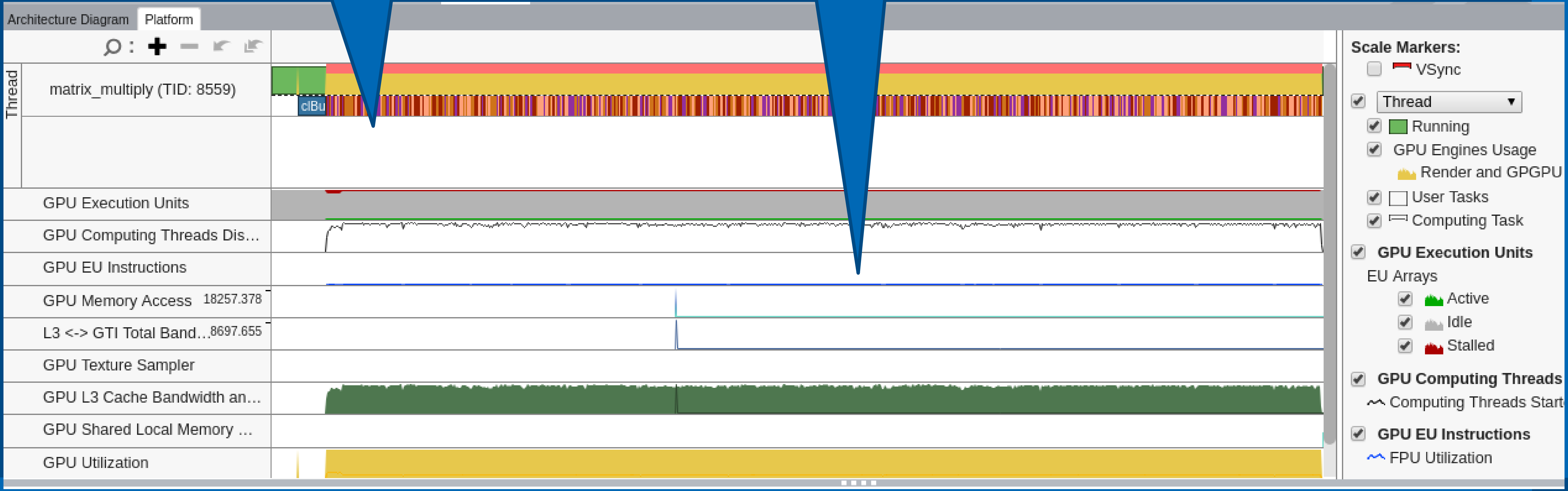
Computing Task	Work Size		Computing Task					Data Transferred		EU Array		
	Global ▼	Local	Total Time	Average Time	Instance Count	SIMD Width	SVM Usage Type	Size	Total, GB/sec	Active	Stalled	Idle
MatrixMultiply2	4096 x 4096		5.440s	5.440s	1	32				67.5%	32.5%	0.0%
clEnqueueReadBufferRect			0.012s	0.012s	1					35.9%	59.8%	4.4%
[Outside any task]										20.9%	10.4%	68.8%

Run VTune Profiler GPU Hotspots to try to identify the source of our low GPU utilization and stalls. Click on the graphics tab in GPU Hotspots and you can see a high-level diagram of your architecture.

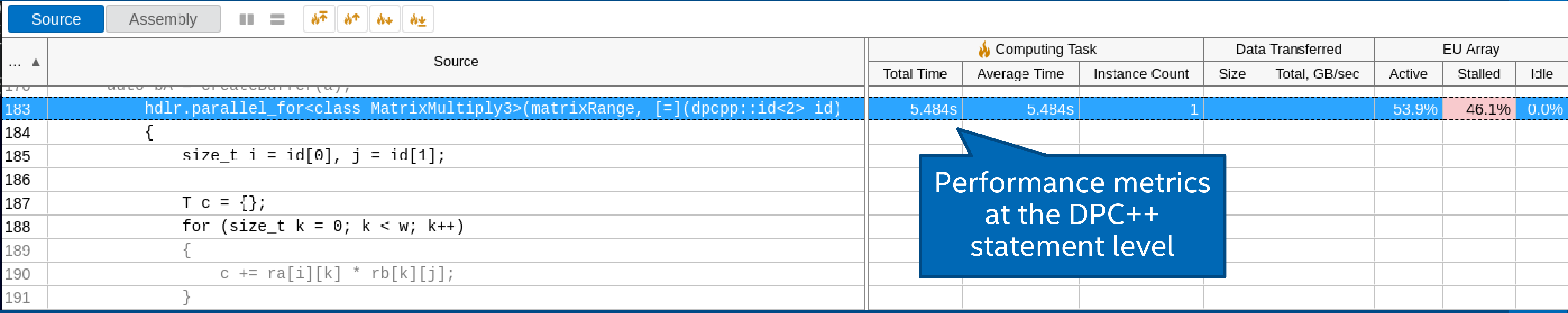
Intel ® VTune™ Profiler for Intel GPUs – Timelines for Correlation

Identify too much or too little kernel activity

Correlate GPU activity with kernels and threads

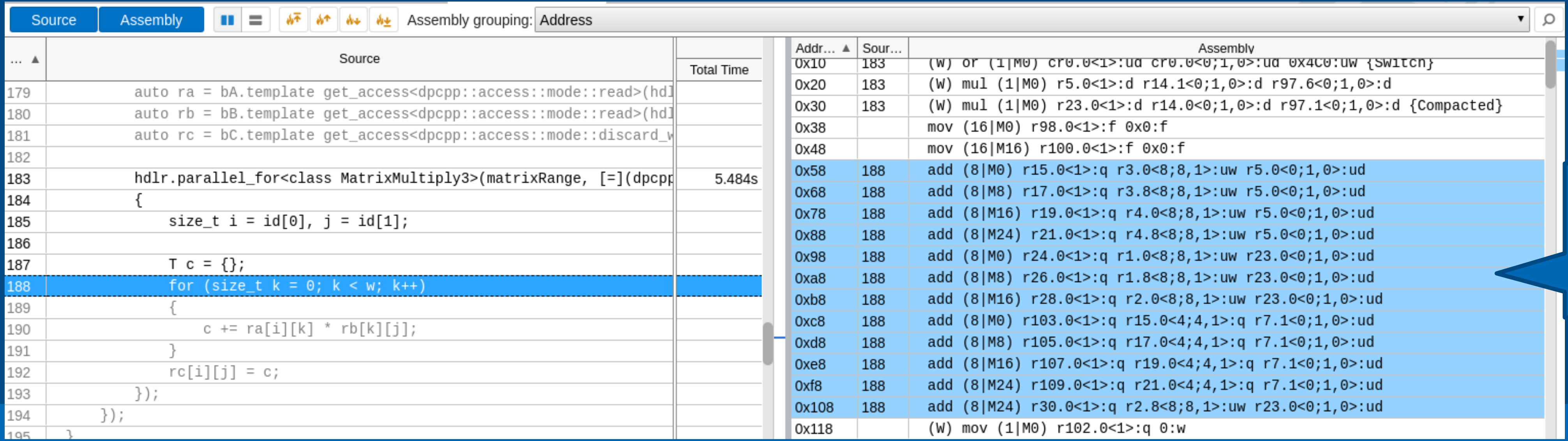


Intel® VTune™ Profiler for DPC++ Code



… ▲	Source	🔥 Computing Task			Data Transferred		EU Array		
		Total Time	Average Time	Instance Count	Size	Total, GB/sec	Active	Stalled	Idle
183	hdlr.parallel_for<class MatrixMultiply3>(matrixRange, [=](dpcpp::id<2> id)	5.484s	5.484s	1			53.9%	46.1%	0.0%
184	{								
185	size_t i = id[0], j = id[1];								
186									
187	T c = {};								
188	for (size_t k = 0; k < w; k++)								
189	{								
190	c += ra[i][k] * rb[k][j];								
191	}								

Performance metrics at the DPC++ statement level



… ▲	Source	Total Time
179	auto ra = bA.template get_access<dpcpp::access::mode::read>(hdl	
180	auto rb = bB.template get_access<dpcpp::access::mode::read>(hdl	
181	auto rc = bC.template get_access<dpcpp::access::mode::discard_w	
182		
183	hdlr.parallel_for<class MatrixMultiply3>(matrixRange, [=](dpcpp	5.484s
184	{	
185	size_t i = id[0], j = id[1];	
186		
187	T c = {};	
188	for (size_t k = 0; k < w; k++)	
189	{	
190	c += ra[i][k] * rb[k][j];	
191	}	
192	rc[i][j] = c;	
193	});	
194	});	
195	}	

Addr... ▲	Sour...	Assembly
0x10	183	(W) or (1|M0) cr0.0<1>:ud cr0.0<0;1,0>:ud 0x4C0:uw {Switch}
0x20	183	(W) mul (1|M0) r5.0<1>:d r14.1<0;1,0>:d r97.6<0;1,0>:d
0x30	183	(W) mul (1|M0) r23.0<1>:d r14.0<0;1,0>:d r97.1<0;1,0>:d {Compacted}
0x38		mov (16|M0) r98.0<1>:f 0x0:f
0x48		mov (16|M16) r100.0<1>:f 0x0:f
0x58	188	add (8|M0) r15.0<1>:q r3.0<8;8,1>:uw r5.0<0;1,0>:ud
0x68	188	add (8|M8) r17.0<1>:q r3.8<8;8,1>:uw r5.0<0;1,0>:ud
0x78	188	add (8|M16) r19.0<1>:q r4.0<8;8,1>:uw r5.0<0;1,0>:ud
0x88	188	add (8|M24) r21.0<1>:q r4.8<8;8,1>:uw r5.0<0;1,0>:ud
0x98	188	add (8|M0) r24.0<1>:q r1.0<8;8,1>:uw r23.0<0;1,0>:ud
0xa8	188	add (8|M8) r26.0<1>:q r1.8<8;8,1>:uw r23.0<0;1,0>:ud
0xb8	188	add (8|M16) r28.0<1>:q r2.0<8;8,1>:uw r23.0<0;1,0>:ud
0xc8	188	add (8|M0) r103.0<1>:q r15.0<4;4,1>:q r7.1<0;1,0>:ud
0xd8	188	add (8|M8) r105.0<1>:q r17.0<4;4,1>:q r7.1<0;1,0>:ud
0xe8	188	add (8|M16) r107.0<1>:q r19.0<4;4,1>:q r7.1<0;1,0>:ud
0xf8	188	add (8|M24) r109.0<1>:q r21.0<4;4,1>:q r7.1<0;1,0>:ud
0x108	188	add (8|M24) r30.0<1>:q r2.8<8;8,1>:uw r23.0<0;1,0>:ud
0x118		(W) mov (1|M0) r102.0<1>:q 0:w

GPU assembly available for compute kernels

Intel® Advisor

Rich Set of Capabilities for High Performance Code Design

Intel® Advisor

Offload Advisor

Design offload strategy and model performance on GPU.

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.

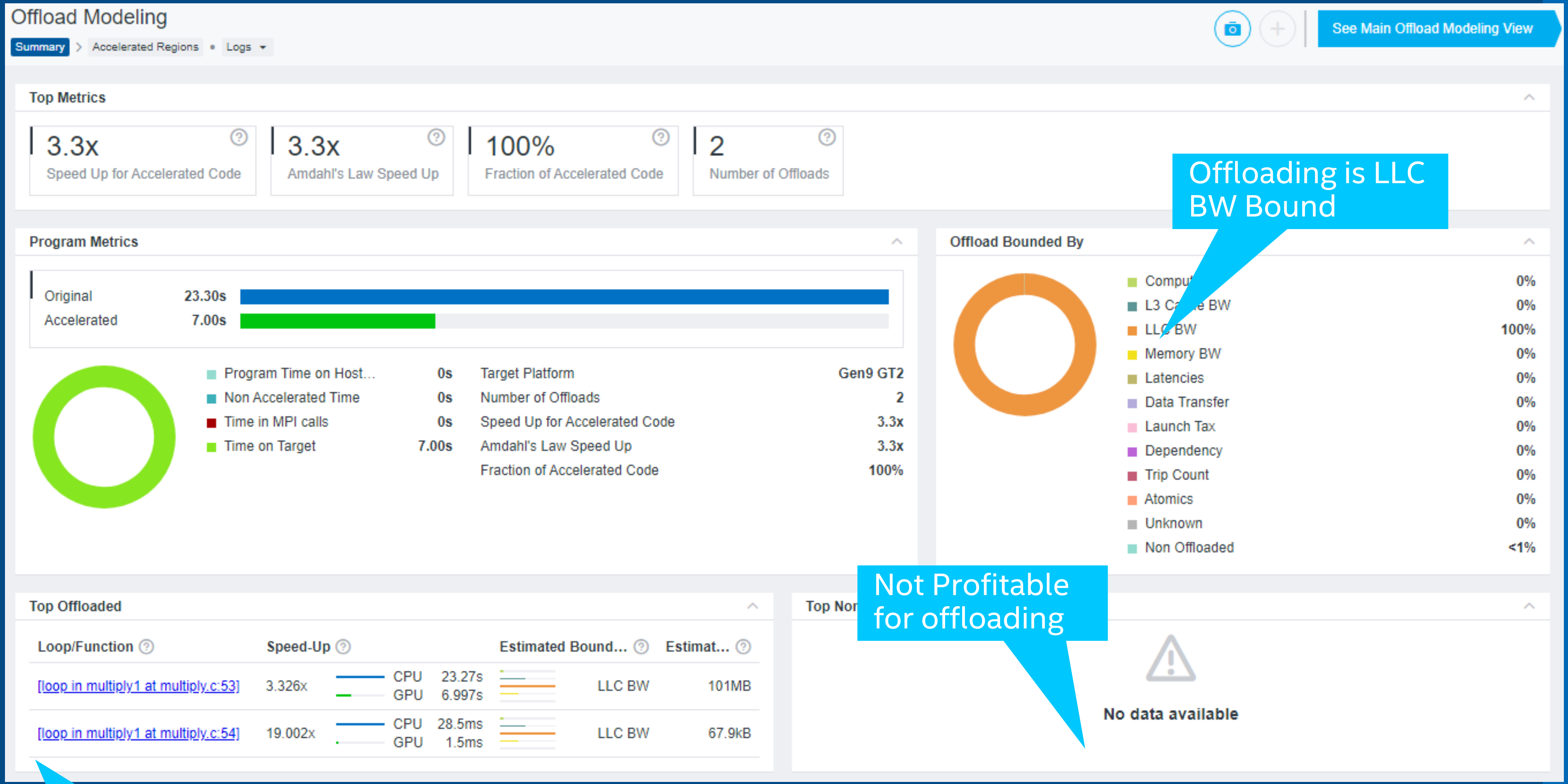
Build Heterogeneous Algorithms

Create and analyze data flow and dependency computation graphs.

Offload Modeling With Intel® Advisor

Intel® Advisor - Offload Modeling

- Run on CPU or GPU – Predict for GPU
- Helps to define which sections of the code should run on given accelerator
- Provides performance projection on accelerators



Offload Modeling
Summary > Accelerated Regions • Logs
See Main Offload Modeling View

Top Metrics

3.3x	3.3x	100%	2
Speed Up for Accelerated Code	Amdahl's Law Speed Up	Fraction of Accelerated Code	Number of Offloads

Program Metrics

Original	23.30s	
Accelerated	7.00s	

🟩 Program Time on Host...	0s	Target Platform	Gen9 GT2
🟦 Non Accelerated Time	0s	Number of Offloads	2
🟥 Time in MPI calls	0s	Speed Up for Accelerated Code	3.3x
🟩 Time on Target	7.00s	Amdahl's Law Speed Up	3.3x
		Fraction of Accelerated Code	100%

Offload Bounded By

Comput	0%
L3 Ca e BW	0%
LLC BW	100%
Memory BW	0%
Latencies	0%
Data Transfer	0%
Launch Tax	0%
Dependency	0%
Trip Count	0%
Atomics	0%
Unknown	0%
Non Offloaded	<1%

Top Offloaded

Loop/Function	Speed-Up			Estimated Bound...	Estimat...
[loop in multiply1 at multiply.c:53]	3.326x	CPU GPU	23.27s 6.997s	LLC BW	101MB
[loop in multiply1 at multiply.c:54]	19.002x	CPU GPU	28.5ms 1.5ms	LLC BW	67.9kB

Top Nor

No data available

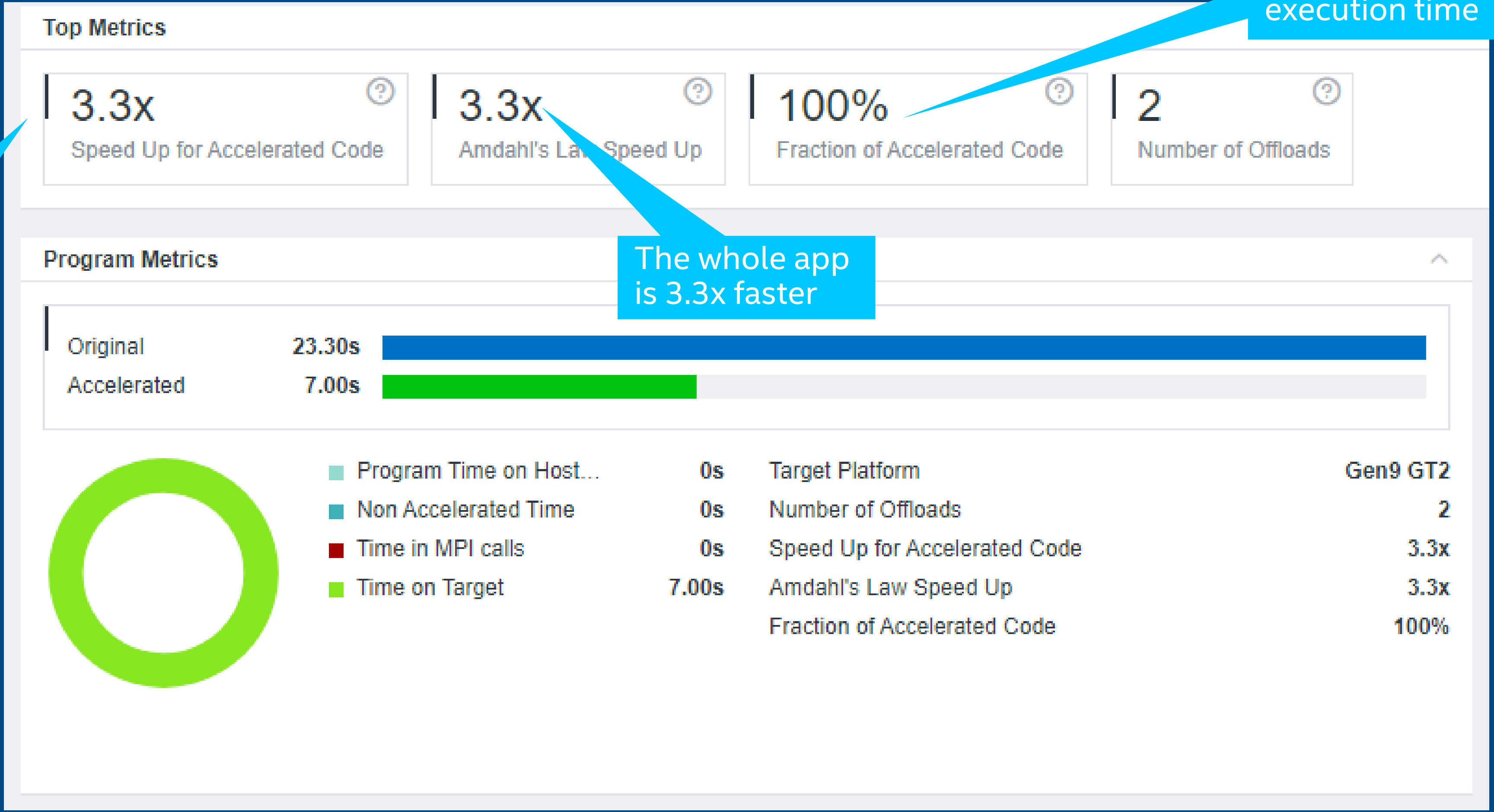
Offloading is LLC BW Bound

Not Profitable for offloading

Recommended for offloading

Intel® Advisor - Offload Modeling

Find code that can be profitably offloaded



Top Metrics

3.3x	3.3x	100%	2
Speed Up for Accelerated Code	Amdahl's Law Speed Up	Fraction of Accelerated Code	Number of Offloads

Program Metrics

Original	23.30s
Accelerated	7.00s

🟦 Program Time on Host...	0s	Target Platform	Gen9 GT2
🟦 Non Accelerated Time	0s	Number of Offloads	2
🟥 Time in MPI calls	0s	Speed Up for Accelerated Code	3.3x
🟩 Time on Target	7.00s	Amdahl's Law Speed Up	3.3x
		Fraction of Accelerated Code	100%

Loop on GPU is 3.3x faster than on CPU

The whole app is 3.3x faster

Loop takes 100% of the whole app execution time

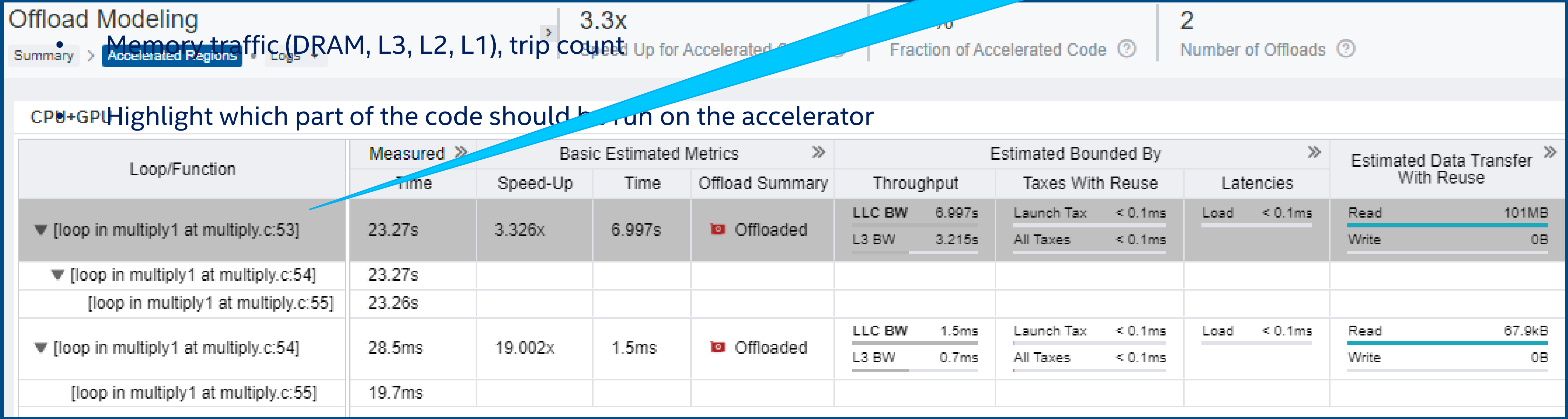
In-Depth Analysis of Top Offload Regions

Provides a detailed description of modeling for each loop

- Timings(total time, time on the accelerator, speedup)
- Offload metrics (offload tax data transfers)
- Memory traffic (DRAM, L3, L2, L1), trip count
- Highlight which part of the code should be run on the accelerator

Loop at multiply.c:53 is recommended for offloading

- LLC BW bound
- Estimated to run on GPU in 6.997s
- Transfers 101MB of data



Offload Modeling

Summary > Accelerated Regions • Logs

3.3x Speed Up for Accelerated C

% Fraction of Accelerated Code

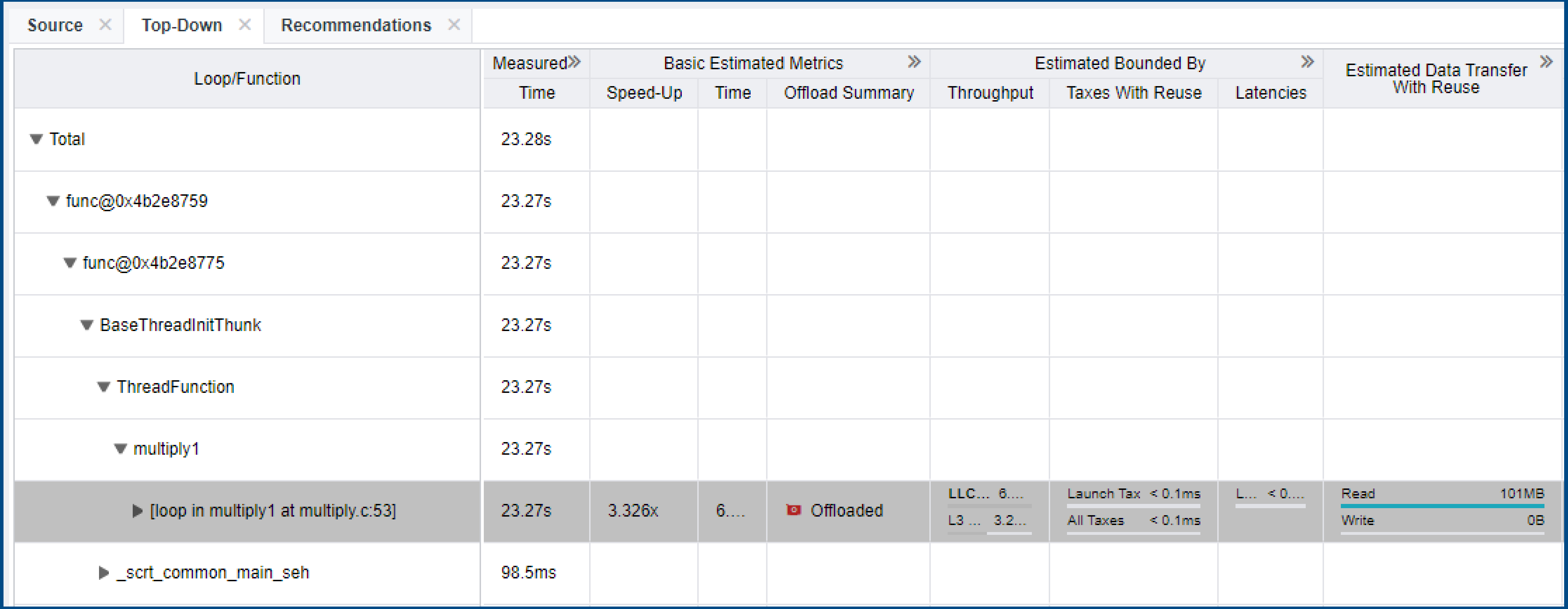
2 Number of Offloads

CPU+GPU

Loop/Function	Measured	Basic Estimated Metrics			Estimated Bounded By			Estimated Data Transfer With Reuse
	Time	Speed-Up	Time	Offload Summary	Throughput	Taxes With Reuse	Latencies	
▼ [loop in multiply1 at multiply.c:53]	23.27s	3.326x	6.997s	Offloaded	LLC BW 6.997s L3 BW 3.215s	Launch Tax < 0.1ms All Taxes < 0.1ms	Load < 0.1ms	Read 101MB Write 0B
▼ [loop in multiply1 at multiply.c:54]	23.27s							
[loop in multiply1 at multiply.c:55]	23.26s							
▼ [loop in multiply1 at multiply.c:54]	28.5ms	19.002x	1.5ms	Offloaded	LLC BW 1.5ms L3 BW 0.7ms	Launch Tax < 0.1ms All Taxes < 0.1ms	Load < 0.1ms	Read 67.9kB Write 0B
[loop in multiply1 at multiply.c:55]	19.7ms							

In-Depth Analysis of Top Offload Regions

Loop metrics are matched with Source and Call Tree



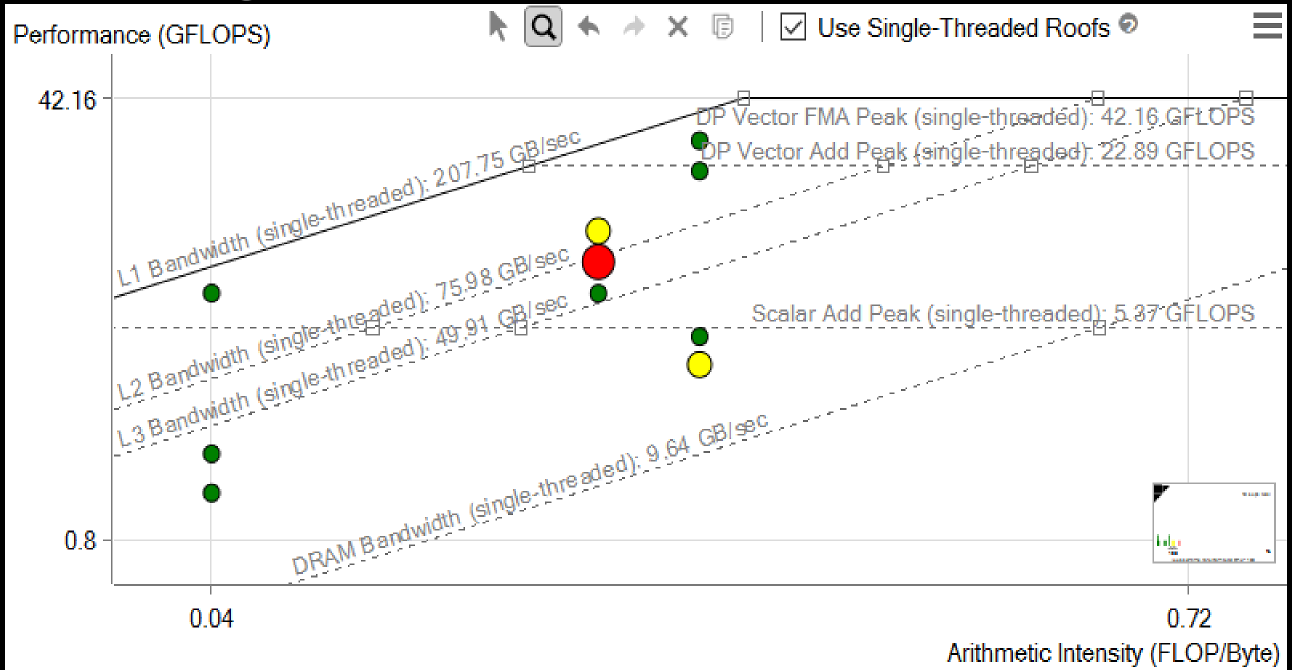
Loop/Function	Measured»	Basic Estimated Metrics »			Estimated Bounded By »			Estimated Data Transfer With Reuse »
	Time	Speed-Up	Time	Offload Summary	Throughput	Taxes With Reuse	Latencies	
▼ Total	23.28s							
▼ func@0x4b2e8759	23.27s							
▼ func@0x4b2e8775	23.27s							
▼ BaseThreadInitThunk	23.27s							
▼ ThreadFunction	23.27s							
▼ multiply1	23.27s							
▶ [loop in multiply1 at multiply.c:53]	23.27s	3.326x	6....	Offloaded	LLC... 6.... L3 ... 3.2...	Launch Tax < 0.1ms All Taxes < 0.1ms	L... < 0....	Read 101MB Write 0B
▶ _scrt_common_main_seh	98.5ms							

GPU Roofline
With Intel® Advisor

What is a Roofline Chart?

- Where are the bottlenecks?
- How much performance is being left on the table?
- Which bottlenecks can be addressed, and which *should* be addressed?
- What's the most likely cause?
- What are the next steps?

Compare application performance against hardware limitations

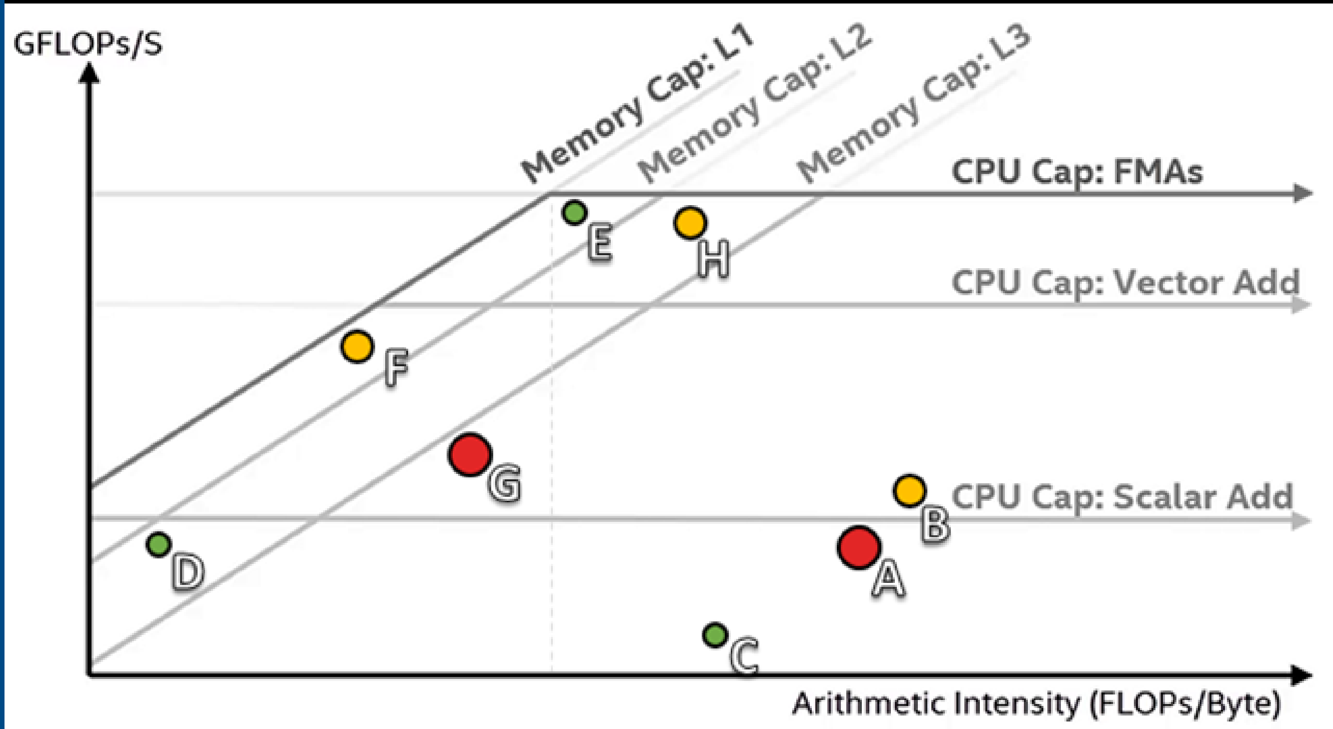

☑ Use Single-Threaded Roofs

Identifying Good Optimization Candidates

Focus optimization effort where it makes the most difference

- Large, red loops have the most impact
- Loops far from the upper roofs have more room to improve

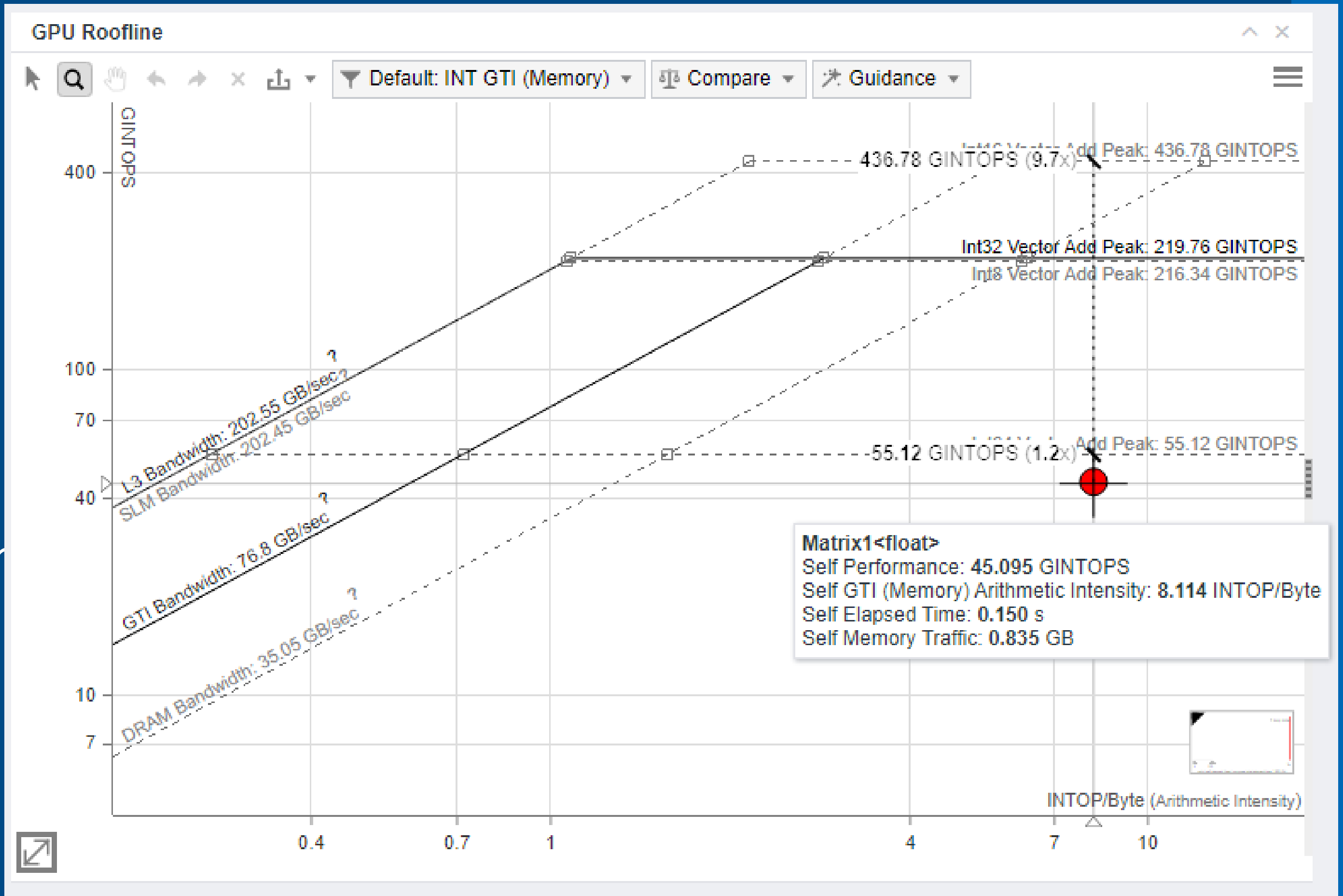
Additional roofs can be plotted for specific computation types or cache levels



Find Effective Optimization Strategies

Intel® Advisor- GPU Roofline

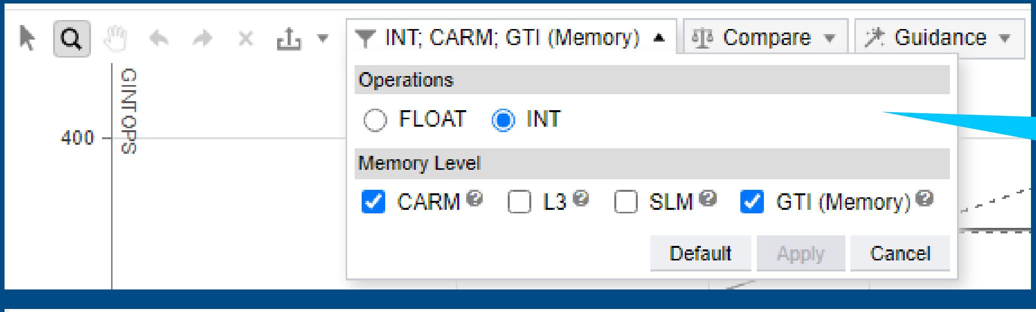
- GPU Roofline Performance
- Highlights poor performing loops
- Shows likely causes of bottlenecks
- Suggests next optimization steps
- Shows performance ‘headroom’ for each loop
- Which can be improved
- Which are worth improving


GPU Roofline
Default: INT GTI (Memory)
Compare
Guidance
GINTOPS
400
100
70
40
10
7
436.78 GINTOPS (9.7x)
Int16 Vector Add Peak: 436.78 GINTOPS
Int32 Vector Add Peak: 219.76 GINTOPS
Int8 Vector Add Peak: 216.34 GINTOPS
55.12 GINTOPS (1.2x)
Int64 Vector Add Peak: 55.12 GINTOPS
L3 Bandwidth: 202.55 GB/sec
SLM Bandwidth: 202.45 GB/sec
GTI Bandwidth: 76.8 GB/sec
DRAM Bandwidth: 35.05 GB/sec
INTOP/Byte (Arithmetic Intensity)
0.4
0.7
1
4
7
10

Matrix1<float>
Self Performance: **45.095** GINTOPS
Self GTI (Memory) Arithmetic Intensity: **8.114** INTOP/Byte
Self Elapsed Time: **0.150** s
Self Memory Traffic: **0.835** GB

Find Effective Optimization Strategies

Intel® Advisor- GPU Roofline



Operations

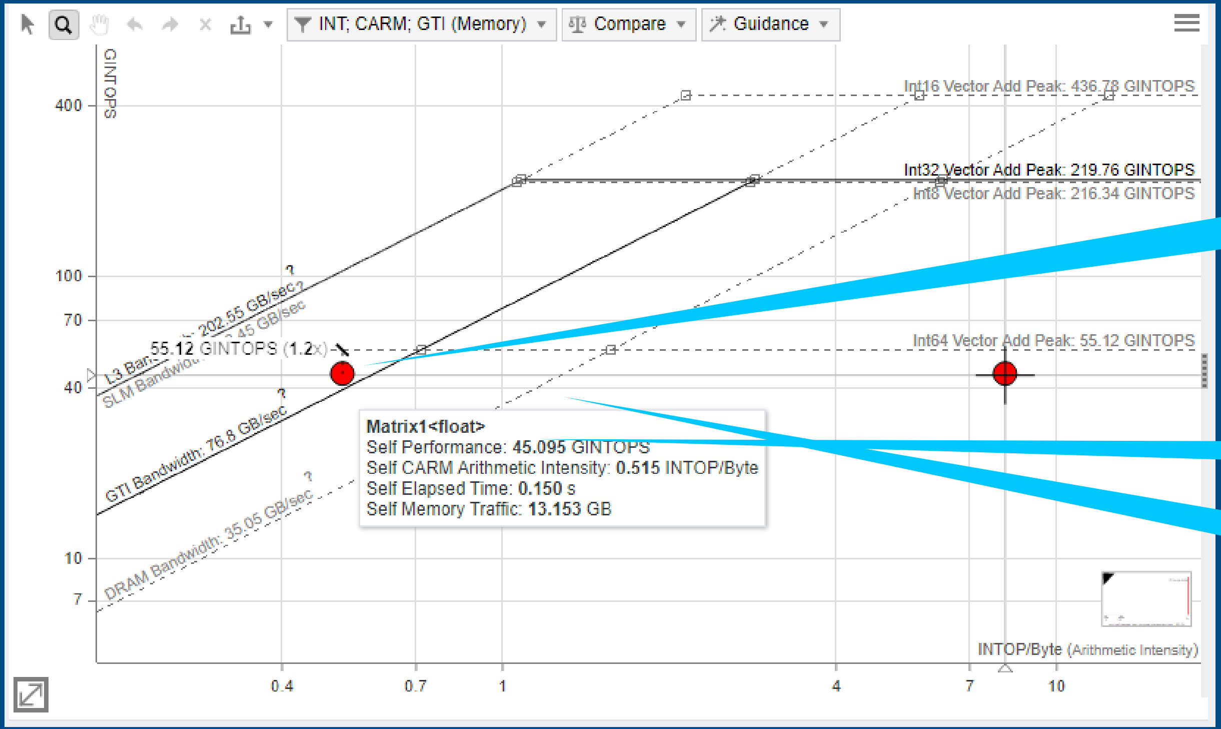
○ FLOAT ● INT

Memory Level

☑ CARM ☐ L3 ☐ SLM ☑ GTI (Memory)

Default Apply Cancel

Configure levels to display



INT; CARM; GTI (Memory)
Compare
Guidance
GINTOPS
400
100
70
40
10
7
Int16 Vector Add Peak: 436.78 GINTOPS
Int32 Vector Add Peak: 219.76 GINTOPS
Int8 Vector Add Peak: 216.34 GINTOPS
Int64 Vector Add Peak: 55.12 GINTOPS
55.12 GINTOPS (1.2x)
L3 Ban... 202.55 GB/sec
SLM Bandwid... 45 GB/sec
GTI Bandwidth: 76.8 GB/sec
DRAM Bandwidth: 35.05 GB/sec
INTOP/Byte (Arithmetic Intensity)
0.4
0.7
1
4
7
10

Matrix1<float>
Self Performance: **45.095** GINTOPS
Self CARM Arithmetic Intensity: **0.515** INTOP/Byte
Self Elapsed Time: **0.150** s
Self Memory Traffic: **13.153** GB

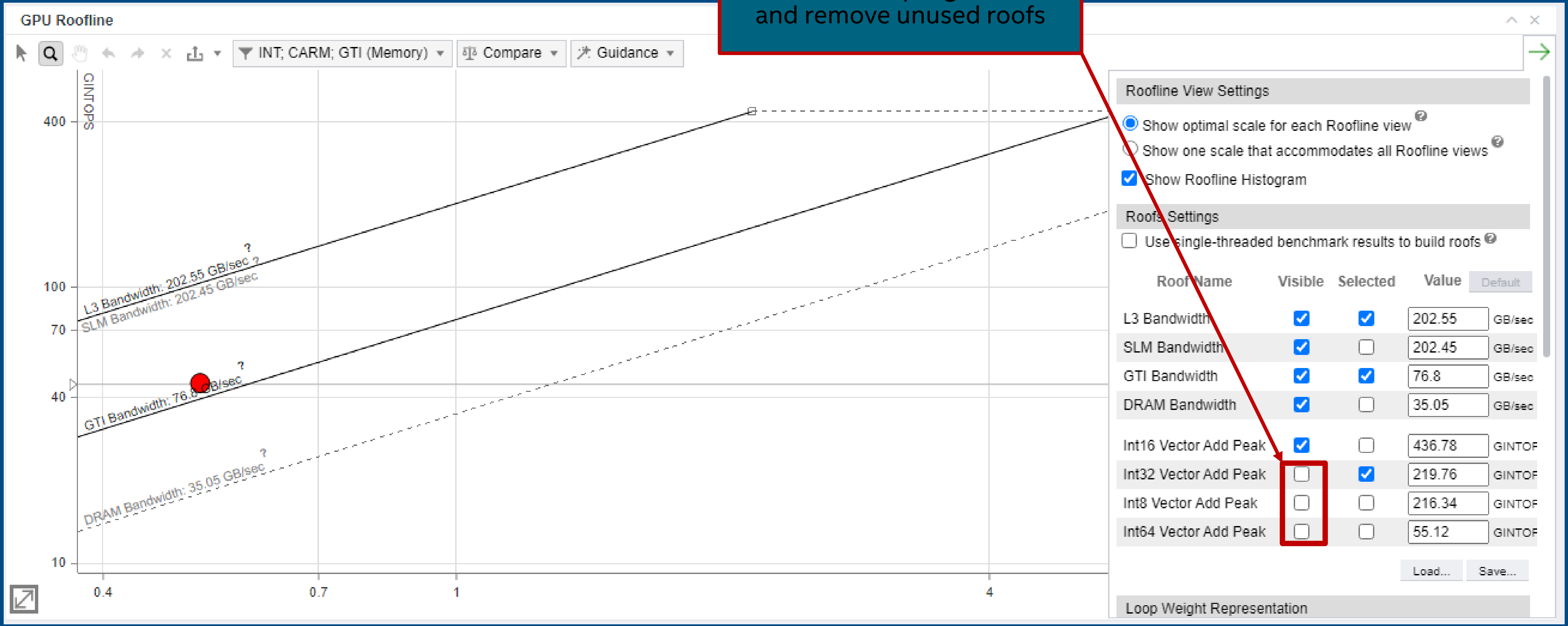
Shows performance headroom for each loop

Likely bottlenecks

Suggests optimization next steps

Customize to Display Only Desired Roofs

Click on the top-right corner and remove unused roofs


GPU Roofline
INT; CARM; GTI (Memory)
Compare
Guidance
GINTOPS
400
100
70
40
10
0.4
0.7
1
4
L3 Bandwidth: 202.55 GB/sec
SLM Bandwidth: 202.45 GB/sec
GTI Bandwidth: 76.8 GB/sec
DRAM Bandwidth: 35.05 GB/sec

Roofline View Settings

● Show optimal scale for each Roofline view

○ Show one scale that accommodates all Roofline views

☑ Show Roofline Histogram

Roofs Settings

☐ Use single-threaded benchmark results to build roofs

Roof Name	Visible	Selected	Value	Default
L3 Bandwidth	☑	☑	202.55	GB/sec
SLM Bandwidth	☑	☐	202.45	GB/sec
GTI Bandwidth	☑	☑	76.8	GB/sec
DRAM Bandwidth	☑	☐	35.05	GB/sec
Int16 Vector Add Peak	☑	☐	436.78	GINTOF
Int32 Vector Add Peak	☐	☑	219.76	GINTOF
Int8 Vector Add Peak	☐	☐	216.34	GINTOF
Int64 Vector Add Peak	☐	☐	55.12	GINTOF

Load... Save...

Loop Weight Representation

Summary metrics

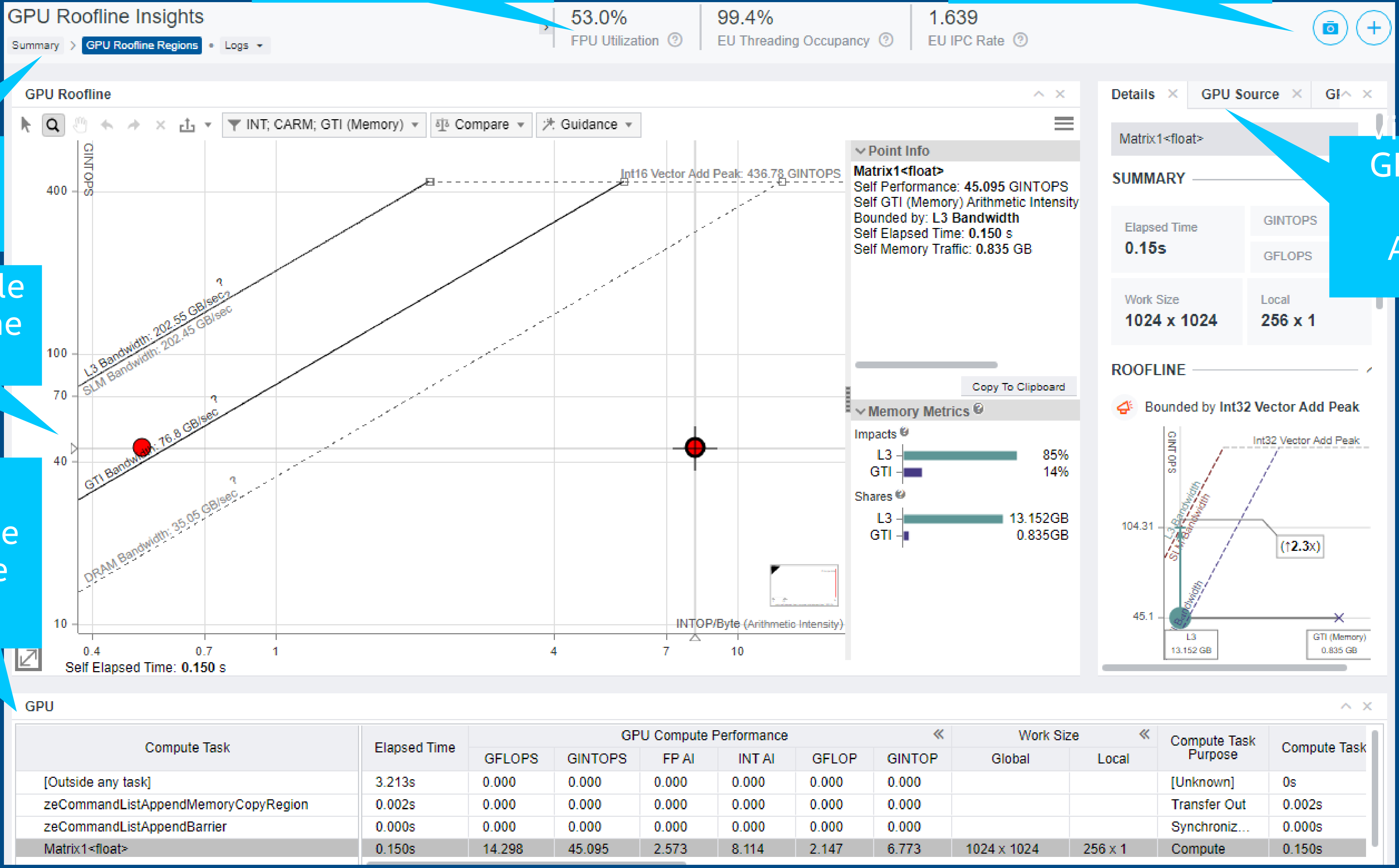
Create a snapshot

Switch between report tabs

Customizable GPU Roofline chart

GPU performance of compute tasks

View Details, GPU Source, and GPU Assembly info



GPU Roofline Insights

Summary > GPU Roofline Regions • Logs

53.0% FPU Utilization

99.4% EU Threading Occupancy

1.639 EU IPC Rate

GPU Roofline

INT; CARM; GTI (Memory) · Compare · Guidance

GINT OPS
400
100
70
40
10
0.4
0.7
1
4
7
10
Int16 Vector Add Peak: 436.78 GINTOPS
L3 Bandwidth: 202.55 GB/sec
SLM Bandwidth: 202.45 GB/sec
GTI Bandwidth: 76.8 GB/sec
DRAM Bandwidth: 35.05 GB/sec
INTOP/Byte (Arithmetic Intensity)

Self Elapsed Time: 0.150 s

Point Info

Matrix1<float>
Self Performance: **45.095** GINTOPS
Self GTI (Memory) Arithmetic Intensity
Bounded by: **L3 Bandwidth**
Self Elapsed Time: **0.150** s
Self Memory Traffic: **0.835** GB

Copy To Clipboard

Memory Metrics

Impacts

L3	85%
GTI	14%

Shares

L3	13.152GB
GTI	0.835GB

Details · GPU Source · GP

Matrix1<float>

SUMMARY

Elapsed Time **0.15s**

GINTOPS

GFLOPS

Work Size **1024 x 1024**

Local **256 x 1**

ROOFLINE

Bounded by **Int32 Vector Add Peak**

GPU

Compute Task	Elapsed Time	GPU Compute Performance						Work Size		Compute Task Purpose	Compute Task
		GFLOPS	GINTOPS	FP AI	INT AI	GFLOP	GINTOP	Global	Local		
[Outside any task]	3.213s	0.000	0.000	0.000	0.000	0.000	0.000			[Unknown]	0s
zeCommandListAppendMemoryCopyRegion	0.002s	0.000	0.000	0.000	0.000	0.000	0.000			Transfer Out	0.002s
zeCommandListAppendBarrier	0.000s	0.000	0.000	0.000	0.000	0.000	0.000			Synchroniz...	0.000s
Matrix1<float>	0.150s	14.298	45.095	2.573	8.114	2.147	6.773	1024 x 1024	256 x 1	Compute	0.150s

Summary

- You can use the Advisor and VTune GUI & CLI to run the collection and to generate the reports.
- Advisor and VTune both provide several analysis types to profile GPU workload.

Legal Disclaimer & Optimization Notice

- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.
- INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

intel®