

Profiling/Optimizing GPU Offload Application using Intel Analyzer Tools

Cory Levels – Technical Consulting Engineer

Xiao Zhu– Technical Consulting Engineer

Rupak Roy– Technical Consulting Engineer



Agenda

1

Introduction

Overview of Analysis Tools

2

Intel® Advisor

GPU Offload Modeling, GPU Roofline Analysis

3

VTune™ Profiler

Work Scheduling Balancing, Data Transfer, GPU Occupancy

4

Recommended Workflow

When are Advisor and VTune most helpful?

5

Q&A

Introduction

Overview of Analysis Tools

Intel® Advisor



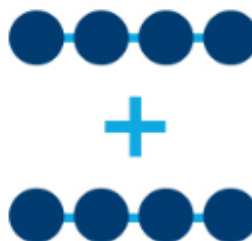
Offload Advisor

Design your code for efficient GPU offload, even before you have the hardware



Automated Roofline Analysis

See performance headroom against hardware limitations. Get insights for an effective optimization roadmap.



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

Optimize Performance

Intel® VTune™ Profiler

Get the Right Data to Find Bottlenecks

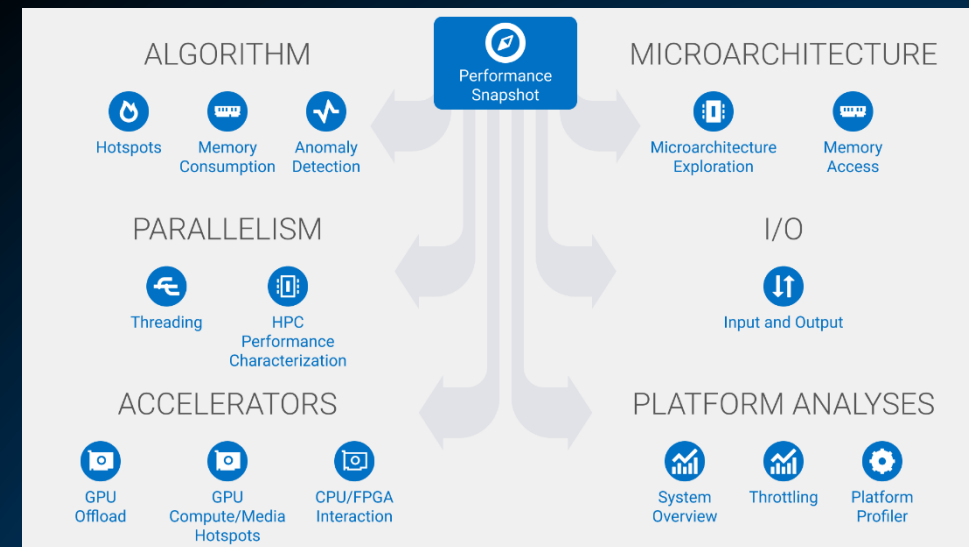
- A suite of profiling for CPU, GPU, FPGA, threading, memory, cache, storage, offload, power...
- Application or system-wide analysis
- DPC++, C, C++, Fortran, Python*, Go*, Java*, or a mix
- Linux, Windows, FreeBSD, Android, Yocto and more
- Containers and VMs

Analyze Data Faster

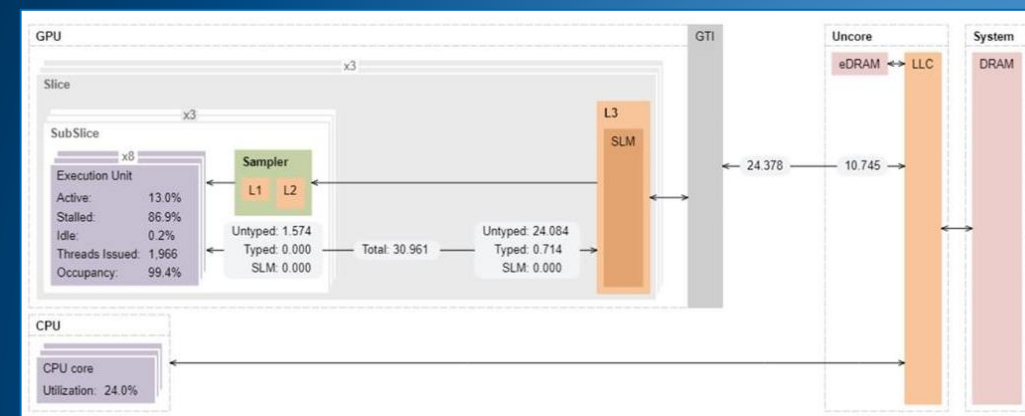
- Collect data HW/SW sampling and tracing w/o re-compilation
- See results on your source, in architecture diagrams, as a histogram, on a timeline...
- Filter and organize data to find answers

Work Your Way

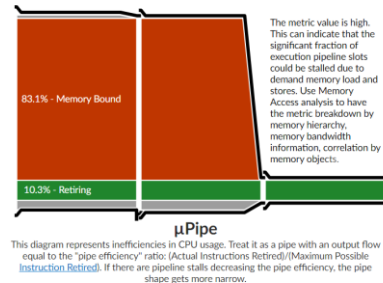
- User interface or command line
- Profile locally and remotely
- GUI (desktop or web) or command line



Source		Assembly		GPU Instructions Executed by Instruction T...	
...		=		Control Flow Send & Wait Int32 & SP Float Int64 & DP Float Other	
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	



Intel® VTune™ Profiler



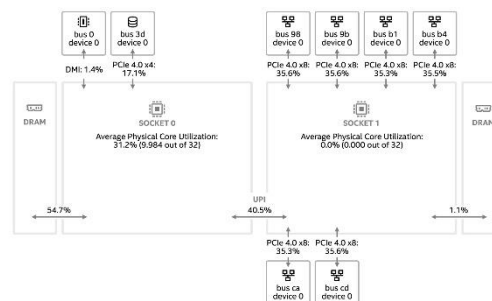
- ✓ Hotspots
- ✓ Anomaly Detection
- ✓ Memory Consumption

Microarch.&Memory Bottlenecks

- ✓ Microarchitecture Exploration
- ✓ Memory Access

Accelerators / xPU

- ✓ GPU Offload
- ✓ GPU Compute / Media Hotspots
- ✓ CPU/FPGA Interaction

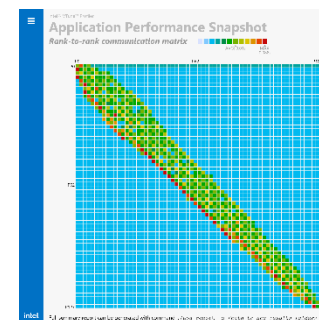


Parallelism

- ✓ Threading
- ✓ HPC Performance Characterization

Platform & I/O

- ✓ Input and Output
- ✓ System Overview
- ✓ Platform Profiler



Multi-Node

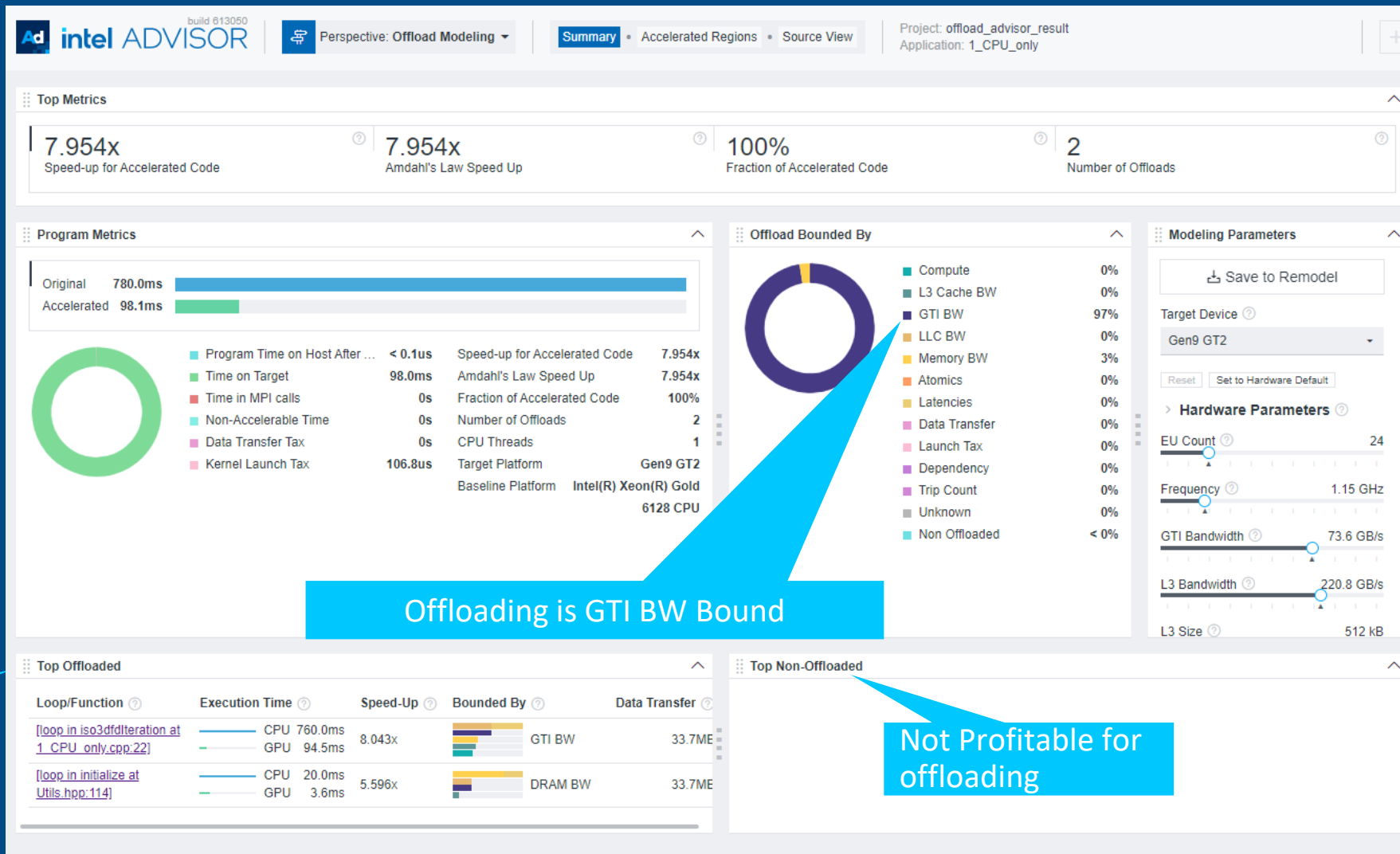
- ✓ Application Performance Snapshot

Intel® Advisor

Offload Advisor

Intel® Advisor

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



In-Depth Analysis of Top Offload Regions

Intel® Advisor

Loop at 1_CPU_only.cpp:22 is recommended for offloading

- GTI BW Bound
- Estimated to run on GPU in 94.5ms

Ad intel ADVISOR build 613050

Perspective: Offload Modeling

Summary Accelerated Regions Source View

7.954x Speed-up ...

7.954x Amdahl's ...

100% Fraction ...

2 Num...

All

Code Regions

Loop/Function	Performance Issues	Measured			Basic Estimated Metrics			
		Time	Region	Iteration Space	Speed-Up	Time	Bounded By	Offload Summary
[loop in iso3dfdIteration at 1_CPU_only.cpp:22]	Code regio...	760.0ms		CC 20 TC 128	8.043x	94.5ms 96.4%	GTI BW	Offloaded
[loop in initialize at Utils.hpp:114]	Code regio...	20.0ms		CC 1 TC 144	5.596x	3.6ms 3.6%	DRAM BW	Offloaded

Source Top Down Recommendations

Line	Source	Is Offloaded	Speed-Up
1	//=====		
2	// Copyright © 2022 Intel Corporation		
3	//		
4	// SPDX-License-Identifier: MIT		
5	// =====		
6			
7	#include <chrono>		
8	#include <string>		
9			
10	#include "Utils.hpp"		
11			
12	void inline iso3dfdIteration(float* ptr_next_base, float* ptr_prev_base,		
13	float* ptr_vel_base, float* coeff, const size_t n1,		
14	const size_t n2, const size_t n3) {		
15	auto dimn1n2 = n1 * n2;		
16			
17	// Remove HALO from the end		
18	auto n3_end = n3 - kHalfLength;		
19	auto n2_end = n2 - kHalfLength;		
20	auto n1_end = n1 - kHalfLength;		
21			
22	for (auto iz = kHalfLength; iz < n3_end; iz++) {	Yes	8.043x
23	for (auto iy = kHalfLength; iy < n2_end; iy++) {		
24	// Calculate start pointers for the row over X dimension		
25	float* ptr_next = ptr_next_base + iz * dimn1n2 + iy * n1;		
26	float* ptr_prev = ptr_prev_base + iz * dimn1n2 + iy * n1;		

Details Data Transfer Estimation

[loop in iso3dfdIteration at 1_CP...

ESTIMATED SPEED-UP: 8.043X

Estimated Time Measured Time

94.5ms 760.0ms

BOUNDED BY: GTI BW

Compute 27.2ms

DRAM BW 33.6ms

L3 BW 30.9ms

LLC BW 52.1ms

Load Latency 42.3ms

Data Transfer Tax 0s

Kernel Launch Tax 100.6us

Estimated Time 94.5ms

Identifying Good Optimization Candidates

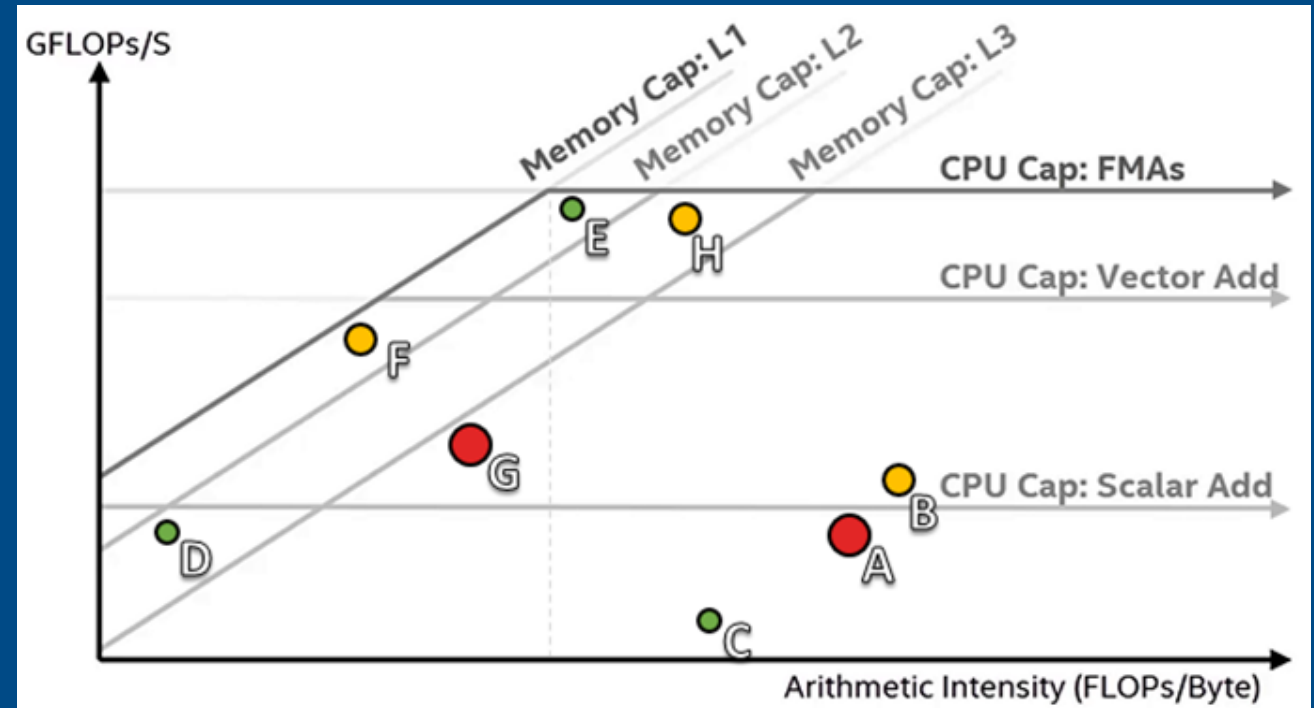


Focus optimization effort where it makes the most difference

- 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



GPU Analysis Flow

Intel® Advisor

Survey Analysis

#1 app run

- GPU Kernels list, timings
- MDAPI collection (hardware counters)

Characterization Analysis

#2 app run

- Measure GPU hardware maximum capabilities
- Binary instrumentation with GTPIN collection

Performance Modeling

No run

- Analyze performance metrics

GPU Roofline Perspective

- Performance Metrics
- Roofline
- Kernel Details
- Guidance
- Recommendations

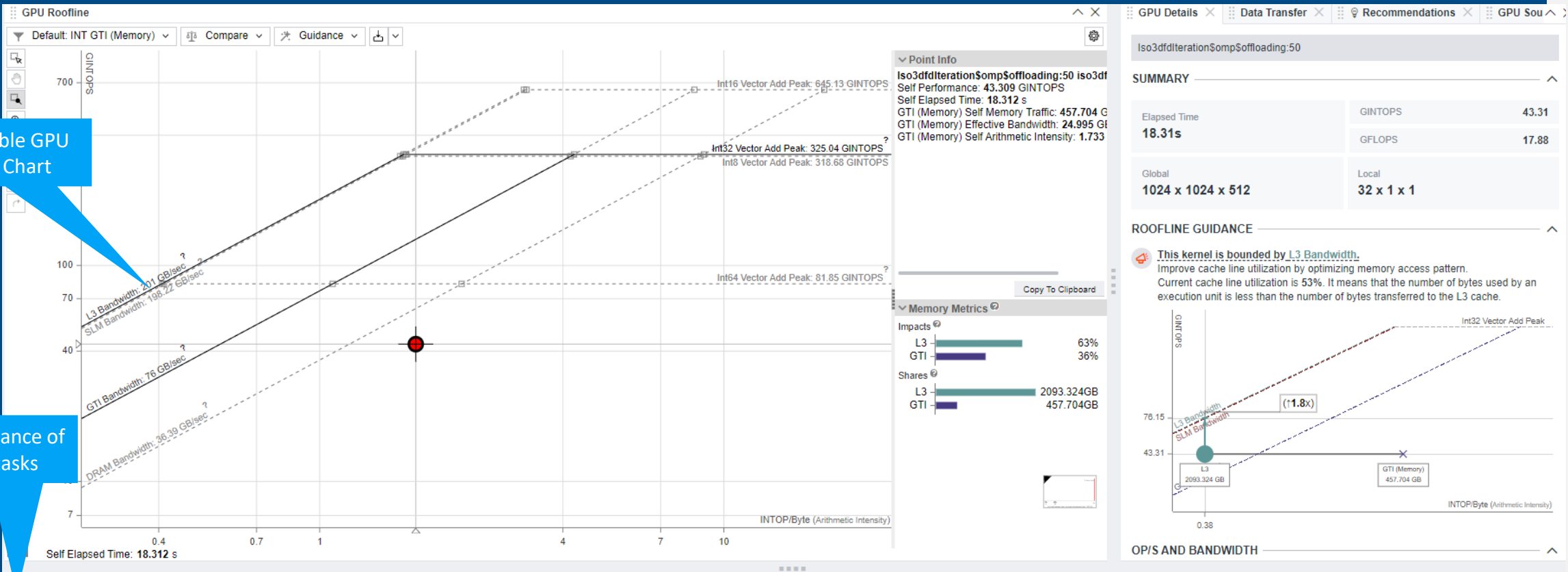
GPU Roofline Chart

Intel® Advisor

View GPU Details,
Data Transfer, GPU
Source and Assembly
info

Customizable GPU
Roofline Chart

GPU performance of
compute tasks



Kernel	Elapsed Time	Performance Issues	GPU Compute Performance		Work Size		Kernel Type	Data Transferred		Kernel Details	
			FLOAT Operations	INT Operations	Global	Local		Total Size	Total Time	Average Time	SIMD Width
Iso3dfdIterationSompSoffloading:50	18.312s	Private memory usage is present	327.491 GOP 17.884 GOP/s AI 0.716	793.059 GOP 43.309 GOP/s AI 1.733	1024 x 1024 x 512	32 x 1 x 1	Compute	0s	1.831s	0s	32
zeCommandListAppendMemoryCopy	0.000s		0.000 GOP 0.000 GOP/s AI 0.000	0.000 GOP 0.000 GOP/s AI 0.000			Host-to-De...	5.1 KB		0.000s	
[Outside any task]	31.765s		0.000 GOP 0.000 GOP/s AI 0.000	0.000 GOP 0.000 GOP/s AI 0.000			[Unknown]		1.021s	0s	

Profiling/Optimizing GPU Offload Application using Intel® VTune™ Profiler

Rupak Roy– Technical Consulting Engineer

Xiao Zhu– Technical Consulting Engineer

Cory Levels – Technical Consulting Engineer

Agenda

- GPU Profiling Overview
- GPU Performance Issues
- OpenMP Offload Case Study
- Conclusion

Intel® VTune™ Profiler

HPC Performance Characterization

Analysis Configuration

Collection Log

Summary

Bottom-up

Platform Diagram

Platform Diagram

Socket 0: Average Physical Core Utilization: 41.5% (9,950 out of 24)

Socket 1: Average Physical Core Utilization: 39.0% (9,360 out of 24)

UPI: 27.4%

Effective Physical Core Utilization: 40.3% (19,329 out of 48)

Effective Logical Core Utilization: 39.0% (37,439 out of 96)

Serial Time (outside parallel regions): 8.300s (54.6%)

- Top Serial Hotspots (outside parallel regions)

Parallel Region Time: 6.902s (45.4%)

- Estimated Ideal Time: 5.960s (39.2%)
- OpenMP Potential Gain: 0.942s (6.2%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was executed sequentially, assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain (s)	OpenMP Potential Gain (%)	OpenMP Region Time (s)
_Z22Iso3dfdVerifyIterationPFS_S_S_iiimmm.DIR.OMP.PARALLEL.2\$omp\$parallel:96@/home/gta/iso3dfd_omp_offload/src/iso3dfd_verify.cpp:25:25	0.942s	6.2%	6.902s

Memory Bound: 44.1% of Pipeline Slots

- Cache Bound: 14.8% of Clockticks
- DRAM Bound: 40.7% of Clockticks
 - DRAM Bandwidth Bound: 39.6% of Elapsed Time
- NUMA: % of Remote Accesses: 65.0%
- Bandwidth Utilization Histogram

Vectorization: 99.9% of Packed FP Operations

- Instruction Mix
 - SP FLOPs: 17.9% of uOps
 - Packed: 99.9% from SP FP
 - 128-bit: 99.9% from SP FP
 - 256-bit: 0.0% from SP FP
 - 512-bit: 0.0% from SP FP
 - Scalar: 0.1% from SP FP
 - DP FLOPs: 0.0% of uOps
 - Packed: 0.0% from DP FP
 - Scalar: 100.0% from DP FP
 - x87 FLOPs: 0.0% of uOps
 - Non-FP: 82.1% of uOps
- FP Arith/Mem Rd Instr. Ratio: 0.562
- FP Arith/Mem Wr Instr. Ratio: 3.945

- Top Loops/Functions with FPU Usage by CPU Time

This section provides information for the most time consuming loops/functions with floating point operations.

Function	CPU Time (s)	% of FP Ops	FP Ops: Packed	FP Ops: Scalar	Vector Instruction Set
[Loop at line 38 in _Z22Iso3dfdVerifyIterationPFS_S_S_iiimmm.DIR.OMP.PARALLEL.2]	532.905s	23.3%	100.0%	0.0%	SSE(128); SSE2(128)
[Loop at line 34 in _Z22Iso3dfdVerifyIterationPFS_S_S_iiimmm.DIR.OMP.PARALLEL.2]	16.680s	2.5%	100.0%	0.0%	SSE(128); SSE2(128)
[Loop at line 103 in WithinEpsilon]	0.460s	9.5%	0.0%	100.0%	SSE(128)

GPU Utilization when Busy: 25.0%

EU State

- Active: 25.0%
- Stalled: 71.9%
- Idle: 3.0%

Occupancy: 97.9% of peak value

Offload Time: 34.1% (5.190s) of elapsed time

- Compute: 84.4% (4.381s) of offload time
- Data Transfer: 14.0% (0.728s) of offload time
- Overhead: 1.6% (0.081s) of offload time

Top OpenMP Offload Regions

OpenMP Offload Region	Offload Time (s)	Percentage of Elapsed Time (%)	Data Transfer (s)	Overhead (s)	GPU Utilization when Busy (%)
iso3dfdIteration\$omp\$target\$region:dvc=0@/home/gta/iso3dfd_omp_offload/src/iso3dfd.cpp:50	4.382s	28.8%	0s	0.000s	0.0%
iso3dfd\$omp\$target\$region:dvc=0@/home/gta/iso3dfd_omp_offload/src/iso3dfd.cpp:332	0.808s	5.3%	0.728s	0.081s	0.0%
[Outside any OpenMP Offload Region]		0.0%			25.0%

*NA is applied to non-summable metrics

A starting point for performance optimization

- CPU/GPU usage, Memory efficiency, and Floating-point utilization

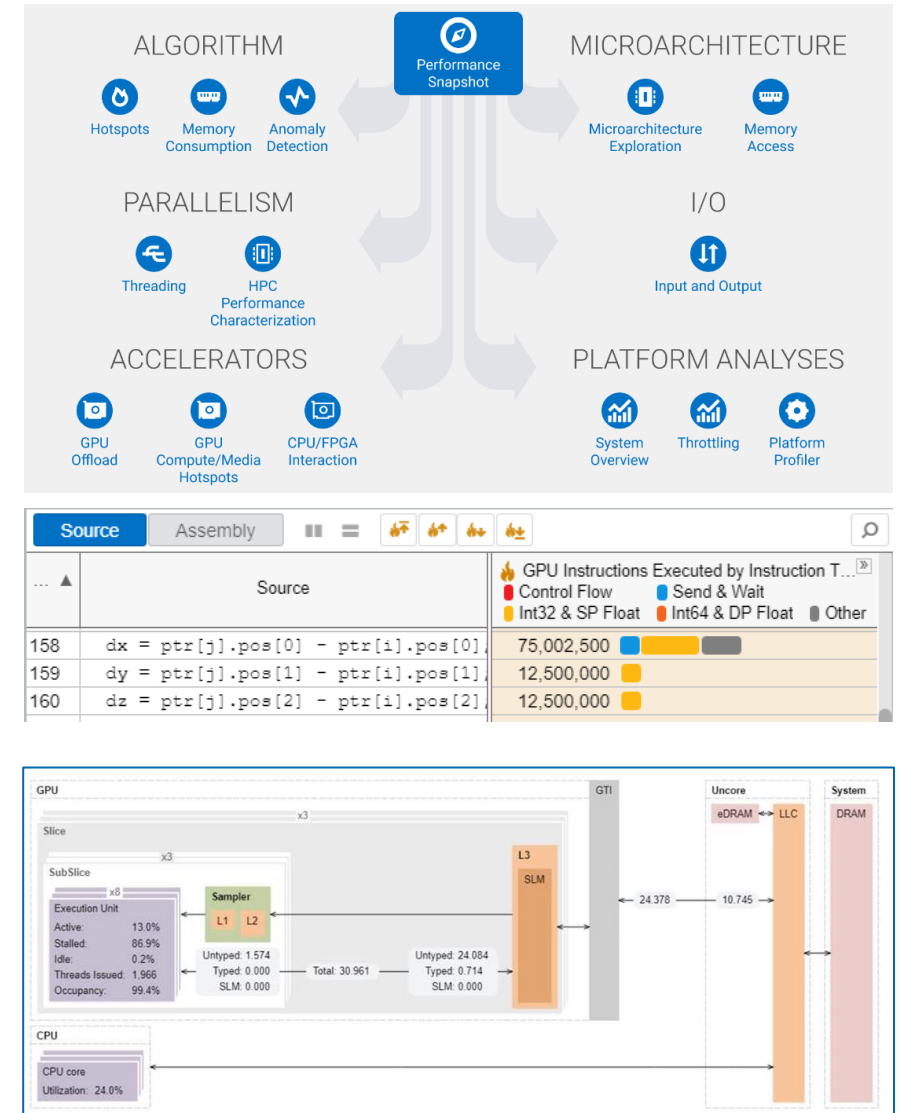
intel

15

GPU Performance Problems

Addressing performance issues with dynamic analysis tools

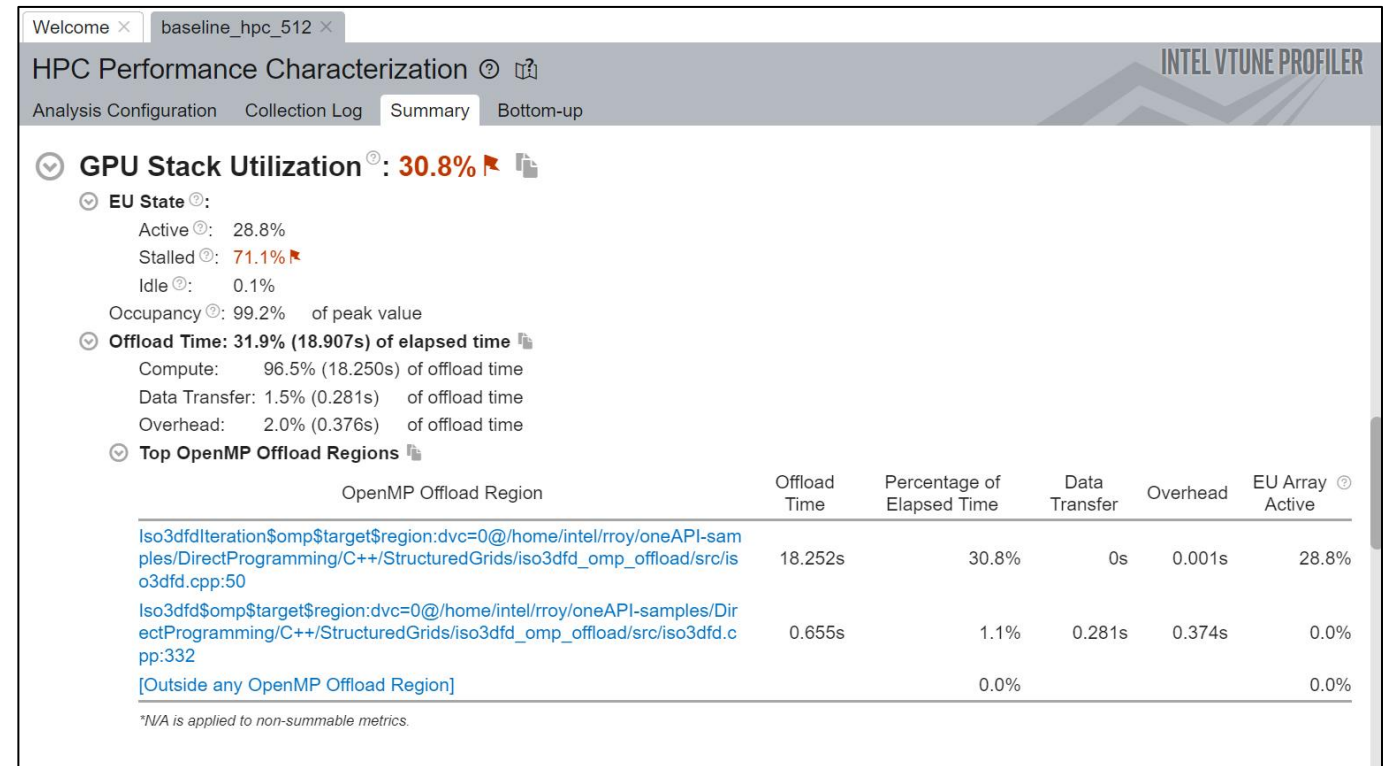
- Work Distribution
- Data transfer
- GPU occupancy
- Memory access
- Kernel inefficiencies
- Non-scaling implementations
- ...



Work Distribution

Work distribution among computing resources

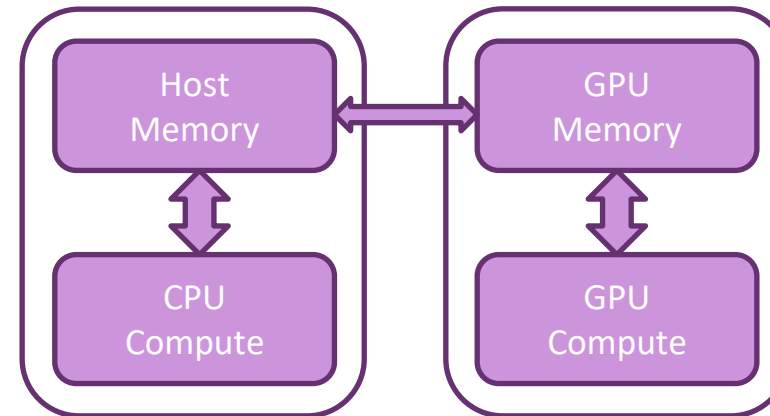
- CPU or GPU bound?
- GPU Utilization for OpenMP regions
- EU/XVEs efficiency (Active, Stalled, Idle)
- Offload Time characterization
 - Compute
 - Data Transfer
 - Overhead



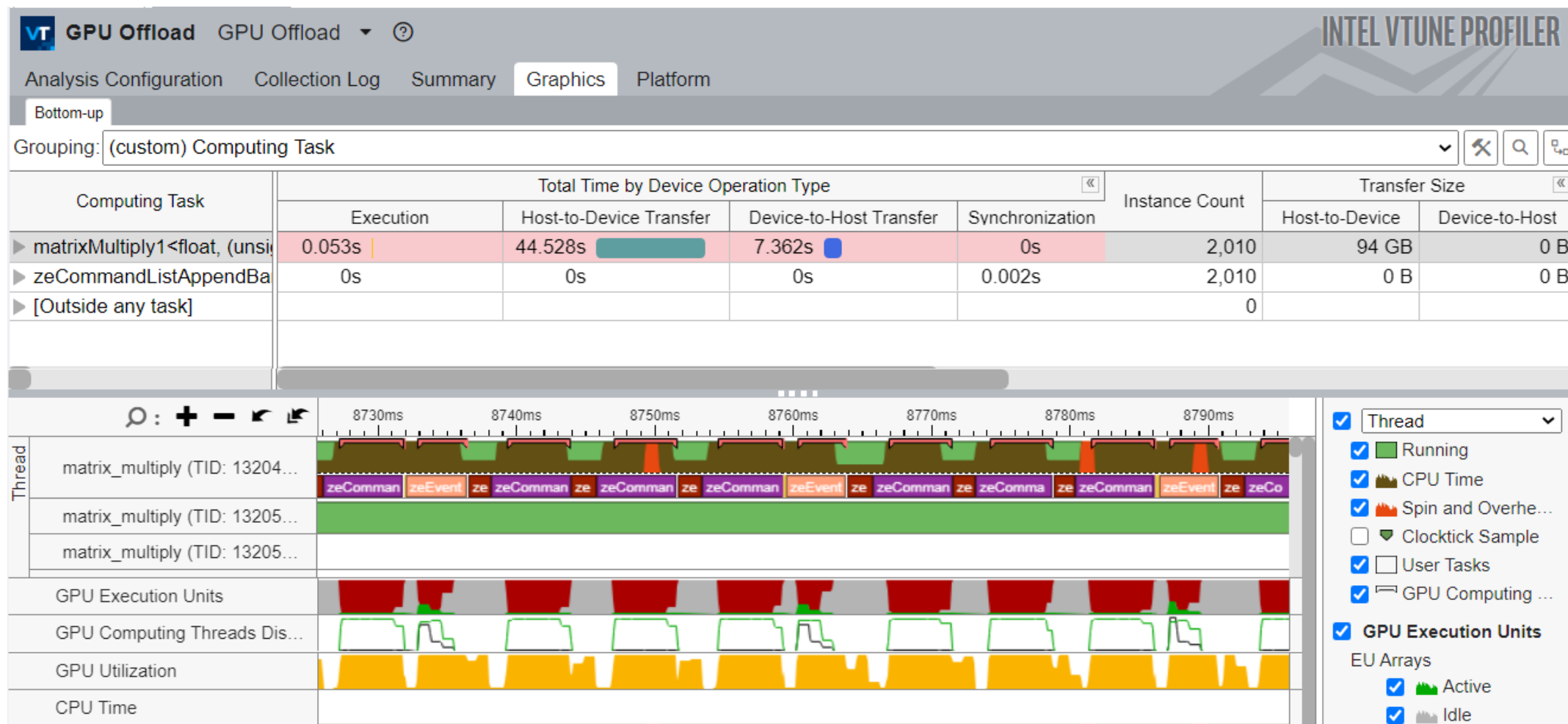
Host and GPU Data Transferring

A commonly known problem of host-to-device transfer performance

- Data transfer time
- Amount of transferred data
- Transfer direction
- Execution time



Graphics View of GPU Offload



Achieving High XVE Threads Occupancy

Occupancy analysis helps identifying ground problems with work mapping

- Detecting workgroups by global and local sizes
- SIMD Width
- Barriers usage
- Tiny/huge kernels scheduling issues

Occupancy 80.4%

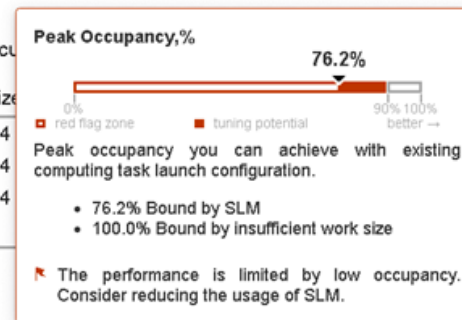
Identify too large or too small computing tasks with low occupancy that make the EU array idle while waiting for the scheduler. Note that frequent SLM accesses and barriers may affect the maximum possible occupancy.

Hottest GPU Computing Tasks with Low Occupancy

This section lists the most active computing tasks running on the GPU with a low Occu

Computing Task	Total Time	Global Size	Local Size
kernel_ocl_path_trace_shadow_blocked_dl	32.492s	128 x 185	64
kernel_ocl_path_trace_shader_sort	21.426s	128 x 185	64
kernel_ocl_path_trace_shader_eval	17.506s	128 x 185	64
[Others]	14.209s		

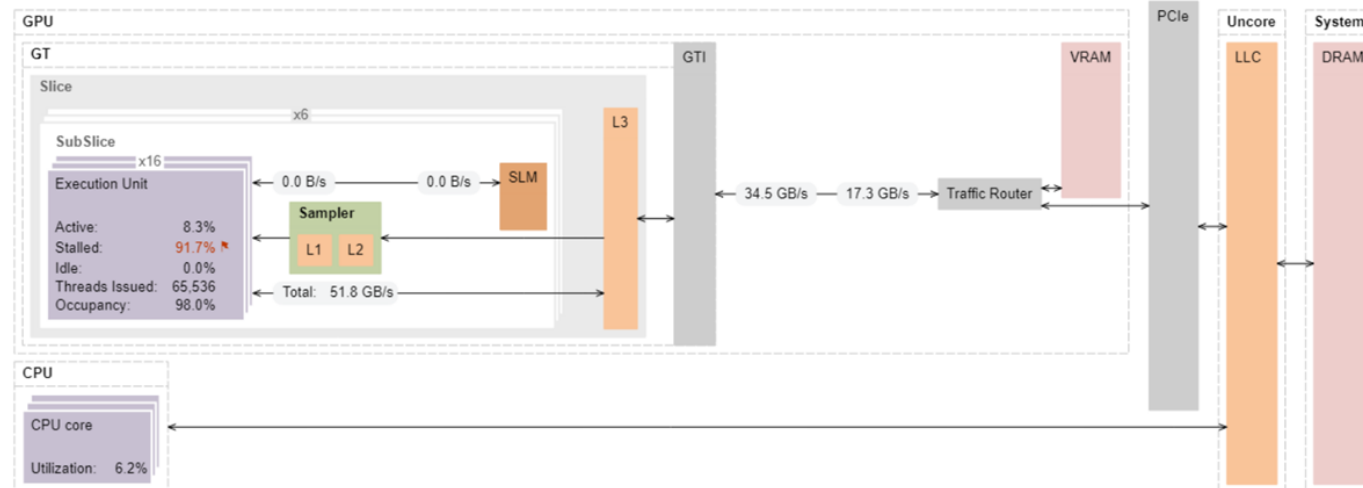
*N/A is applied to non-summable metrics.



Occupancy	Occupancy
100.0%	88.6%
76.2%	60.5%
100.0%	78.1%
0.0%	73.2%

Memory Access problems

- Global memory access penalty
- Cache memory resource limit
- Which code is responsible for latency?
- Per Basic Block and latencies per individual instructions



Source level in-kernel profiling

GPU Compute/Media Hotspots (preview) ⓘ ⓘ		
Analysis Configuration Collection Log Summary Graphics iso3dfd_kernels.cpp ×		
Source Assembly ⏸ = 🔍 ⚙️ ⚡ ⚡ ⚡ ⚡		
Source...	Source	🔥 Estimated GPU Cycles
428	for (auto iter = 0; iter < kHalfLength; iter++) {	
429	front[iter] = front[iter + 1];	
430	}	
431		
432	// Only one new data-point read from global memory	
433	// in z-dimension (depth)	
434	front[kHalfLength] = prev[gid + kHalfLength * nxy];	8.573e+8
435		
436	// Stencil code to update grid point at position given by global id (gid)	
437	float value = c[0] * front[0];	3.429e+8
438	#pragma unroll(kHalfLength)	
439	for (auto iter = 1; iter <= kHalfLength; iter++) {	
440	value += c[iter] * (front[iter] + back[iter - 1] + prev[gid + iter] +	1.367e+10
441	prev[gid - iter] + prev[gid + iter * nx] +	1.097e+10
442	prev[gid - iter * nx]);	2.358e+10
443	}	
444		
445	next[gid] = 2.0f * front[0] - next[gid] + value * vel[gid];	1.929e+9
446		
447	gid += nxy;	
448	begin_z++;	3.429e+8
449	}	
450	}	
451		
452	/*	
453	* Host-side SYCL Code	
454	*	
455	* Driver function for ISO3DFD SYCL code	
456	* Uses ptr_next and ptr_prev as ping-pong buffers to achieve	
457	* accelerated wave propagation	

Basic Block Latency

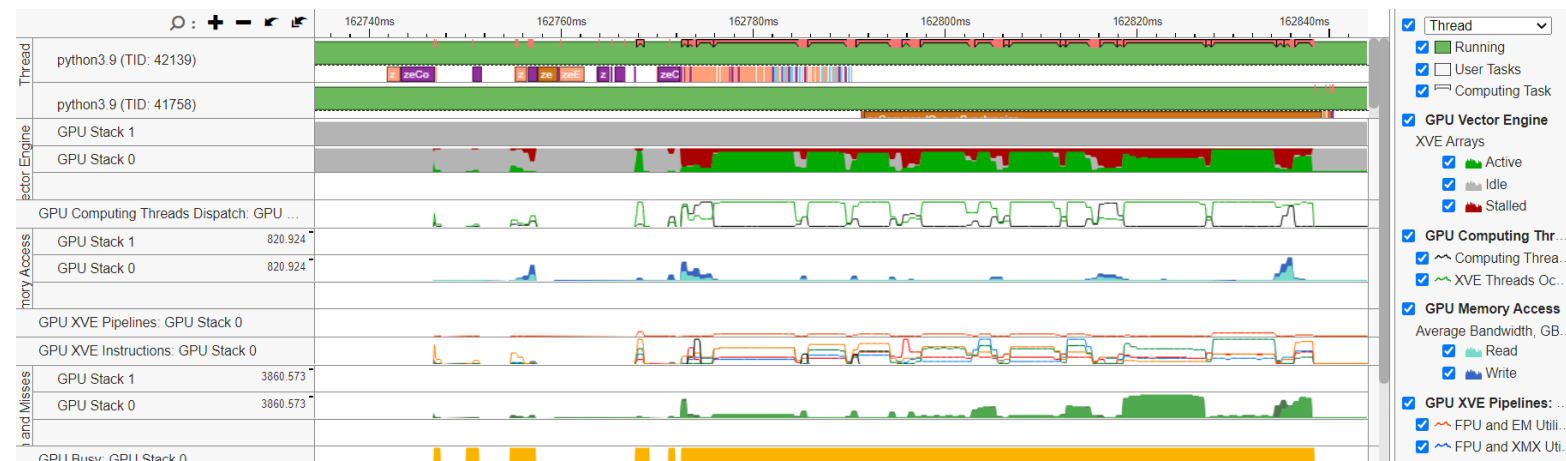
GPU Compute/Media Hotspots (preview) ⓘ ⓘ		
Analysis Configuration Collection Log Summary Graphics iso3dfd_kernels.cpp ×		
Source Assembly ⏸ = 🔍 ⚙️ ⚡ ⚡ ⚡ ⚡		
So...	Source	🔥 Average Latency, Cycles ⓘ Estimated GPU Cycles
431		
432	// Only one new data-point read from global memory	
433	// in z-dimension (depth)	
434	front[kHalfLength] = prev[gid + kHalfLength * nxy];	856 1.061e+10
435		
436	// Stencil code to update grid point at position given by global id (gid)	
437	float value = c[0] * front[0];	
438	#pragma unroll(kHalfLength)	
439	for (auto iter = 1; iter <= kHalfLength; iter++) {	
440	value += c[iter] * (front[iter] + back[iter - 1] + prev[gid + iter] +	214 7.963e+9
441	prev[gid - iter] + prev[gid + iter * nx] +	186 2.302e+10
442	prev[gid - iter * nx]);	196 1.943e+10
443	}	
444		
445	next[gid] = 2.0f * front[0] - next[gid] + value * vel[gid];	875 2.169e+10
446		
447	gid += nxy;	
448	begin_z++;	
449	}	
450	}	
451		
452	/*	
453	* Host-side SYCL Code	
454	*	
455	* Driver function for ISO3DFD SYCL code	
456	* Uses ptr_next and ptr_prev as ping-pong buffers to achieve	
457	* accelerated wave propagation	
458	*	
459	* This function uses SYCL buffers to facilitate host to device	
460	* buffer copies	

Memory Latency

Kernel code optimizations

Advanced code optimizations on kernel level

- Are FPUs and EM pipelines fully utilized?
- Instructions counting profiles
- How are the systolic instructions used in AI application?
- FPU and XMV pipeline Utilization



Source level in-kernel profiling

GPU Compute/Media Hotspots (preview) ⓘ

Analysis ConfigurationCollection LogSummaryGraphicsiso3dfd_kernels.cpp ×

SourceAssembly

Sou... ▲	Source	🔥 GPU Instructions Executed by Instruction Type				
		Control Flow	Send	Int32 & SP Float	Int64 & DP Float	Other
429	front[iter] = front[iter + 1];					
430	}					
431						
432	// Only one new data-point read from global memory					
433	// in z-dimension (depth)					
434	front[kHalfLength] = prev[gid + kHalfLength * nxy];	0.000e+0	1.239e+7 🔵	0.000e+0	4.955e+7 🔴	0.000e+0
435						
436	// Stencil code to update grid point at position given by global id (gid)					
437	float value = c[0] * front[0];	0.000e+0	0.000e+0	1.239e+7 🟡	0.000e+0	0.000e+0
438	#pragma unroll(kHalfLength)					
439	for (auto iter = 1; iter <= kHalfLength; iter++) {					
440	value += c[iter] * (front[iter] + back[iter - 1] + prev[gid + iter] +	0.000e+0	3.716e+7 🔵	4.087e+8 🟡	1.239e+8 🔴	3.716e+7 🟡
441	prev[gid - iter] + prev[gid + iter * nx] +	0.000e+0	1.239e+8 🔵	1.982e+8 🟡	2.725e+8 🔴	0.000e+0
442	prev[gid - iter * nx]);	0.000e+0	9.909e+7 🔵	3.964e+8 🟡	3.964e+8 🔴	6.689e+8 🔴
443	}					
444						
445	next[gid] = 2.0f * front[0] - next[gid] + value * vel[gid];	0.000e+0	3.716e+7 🔵	2.477e+7 🟡	4.955e+7 🔴	1.239e+7 🟡
446						
447	gid += nxy;					
448	begin_z++;	0.000e+0	0.000e+0	0.000e+0	2.477e+7 🔴	0.000e+0
449	}					
450	}					
451						
452	/*					
453	* Host-side SYCL Code					
454	*					
455	* Driver function for ISO3DFD SYCL code					
456	* Uses ptr_next and ptr_prev as ping-pong buffers to achieve					
457	* accelerated wave propagation					

Case Study (OpenMP* Offload on GPU)

- Application Name: iso3dfd OpenMP Offload
- Profile the baseline version
- Detect the bottlenecks
- Make code changes based on the findings
- Compare baseline vs optimized

HPC Characterization

Welcome
baseline_offload_512
baseline_hpc_512

HPC Performance Characterization

INTEL VTUNE PROFILER

Analysis Configuration
Collection Log
Summary
Bottom-up

GPU Stack Utilization
30.8%

EU State
Active: 28.8%
Stalled: 71.1%
Idle: 0.1%
Occupancy: 99.2% of peak value

Offload Time: 31.9% (18.907s) of elapsed time
Compute: 96.5% (18.250s) of offload time
Data Transfer: 1.5% (0.281s) of offload time
Overhead: 2.0% (0.376s) of offload time

Top OpenMP Offload Regions

OpenMP Offload Region	Offload Time	Percentage of Elapsed Time	Data Transfer	Overhead	EU Array Active
Iso3dfdIteration\$omp\$target\$region:dvc=0@/home/intel/rroy/oneAPI-samples/DirectProgramming/C++/StructuredGrids/iso3dfd_omp_offload/src/iso3dfd.cpp:50	18.252s	30.8%	0s	0.001s	28.8%
Iso3dfd\$omp\$target\$region:dvc=0@/home/intel/rroy/oneAPI-samples/DirectProgramming/C++/StructuredGrids/iso3dfd_omp_offload/src/iso3dfd.cpp:332	0.655s	1.1%	0.281s	0.374s	0.0%
[Outside any OpenMP Offload Region]		0.0%			0.0%

*NA is applied to non-summable metrics.

HPC Characterization Bottom-up View for Baseline Version

Welcome
baseline_offload_512
baseline_hpc_512

HPC Performance Characterization

INTEL VTUNE PROFILER

Analysis Configuration
Collection Log
Summary
Bottom-up

Grouping:
OpenMP Offload Region / Function / Call Stack

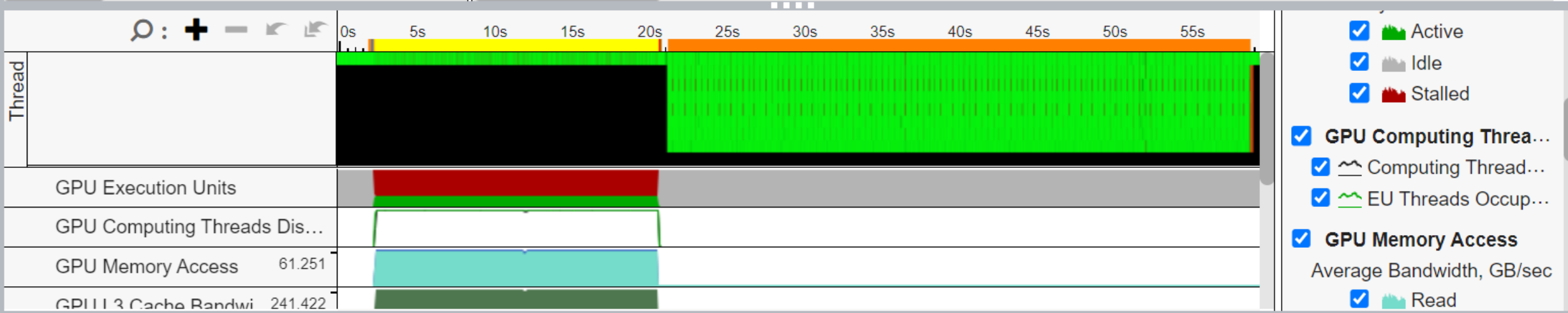
OpenMP Offload Region / Function / Call Stack	OpenMP Offload Time	Instance Count	GPU				CPU Time	Serial CPU Time
			EU State			Occupancy		
			Active	Stalled	Idle			
Iso3dfdIteration\$omp\$target\$region:dvc=0@/h	18.252s	100	28.8%	71.1%	0.1%	99.2%	17.930s	17.929s
Iso3dfd\$omp\$target\$region:dvc=0@/home/inte	0.655s	2	0.0%	0.0%	100.0%	0.0%	0.643s	0.643s
[Outside any OpenMP Offload Region]			0.0%	0.1%	99.9%	0.0%	295.220s	2.853s

HPC Performance Characterization

Analysis Configuration Collection Log Summary Bottom-up

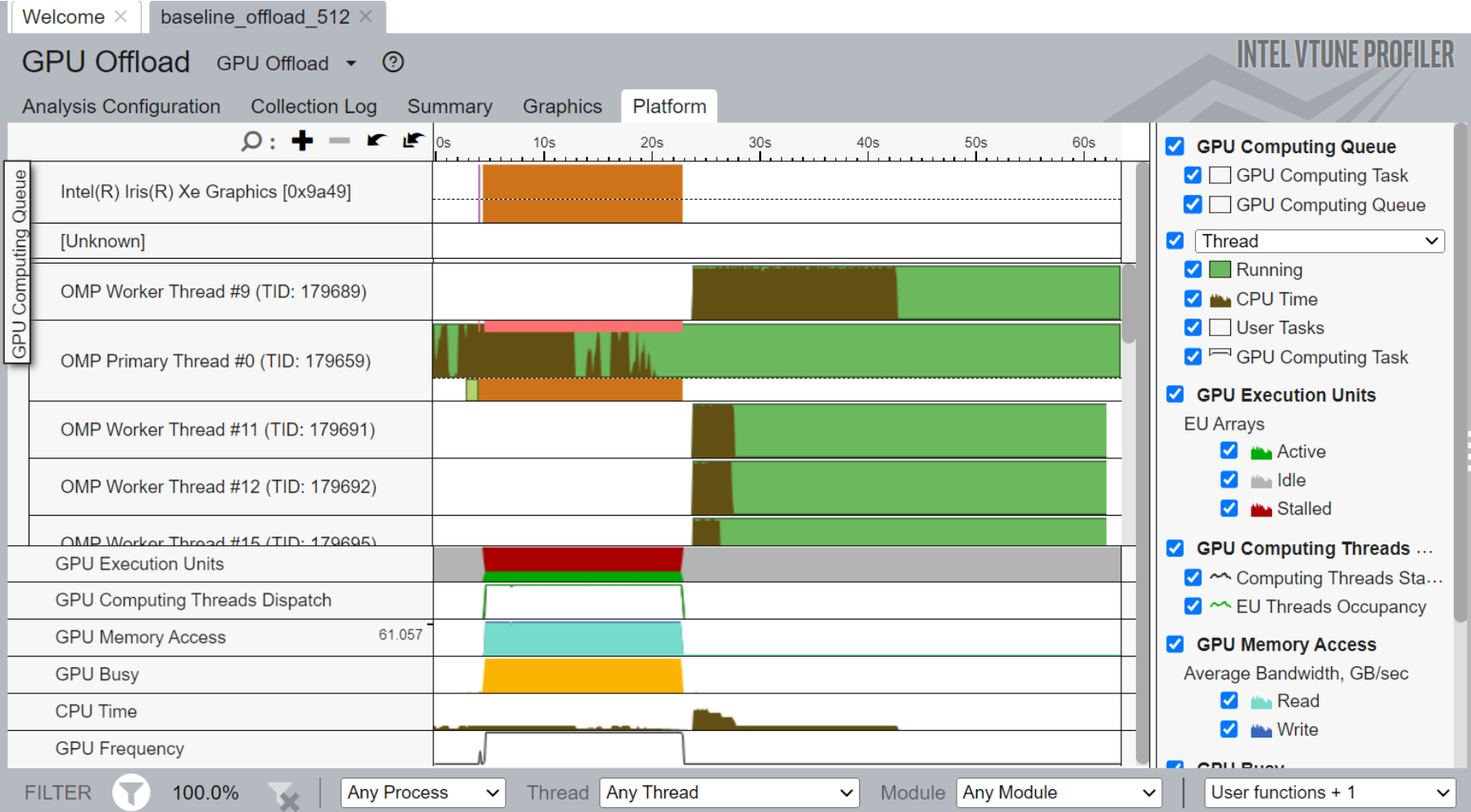
Grouping: Process / OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack

Process / OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack	Elapsed Time ▼	SP GFLOPS	Serial CPU Time	Memory Bound	OpenMP Po		
					Imbalance	Lock Contention	Creation
iso3dfd	59.330s	13.415	21.425s	28.9%	0.816s	0s	
_Z22Iso3dfdVerifyIterationPfS_S_S_iiimmm	37.481s	21.222	0s	30.3%	0.816s	0s	
Iso3dfdVerifyIteration\$omp\$loop_barrier_s	37.479s	21.222	0s	30.3%	0.816s	0s	
_INTERNALbae80b52::__kmp_for_stat		0.000	0s	0.0%			
__kmp_get_global_thread_id		0.000	0s	0.0%			
[Loop@0x167f7e in oa_buffer_check_u		0.000	0s	100.0%			
oa buffer check unlocked		0.000	0s	0.0%			

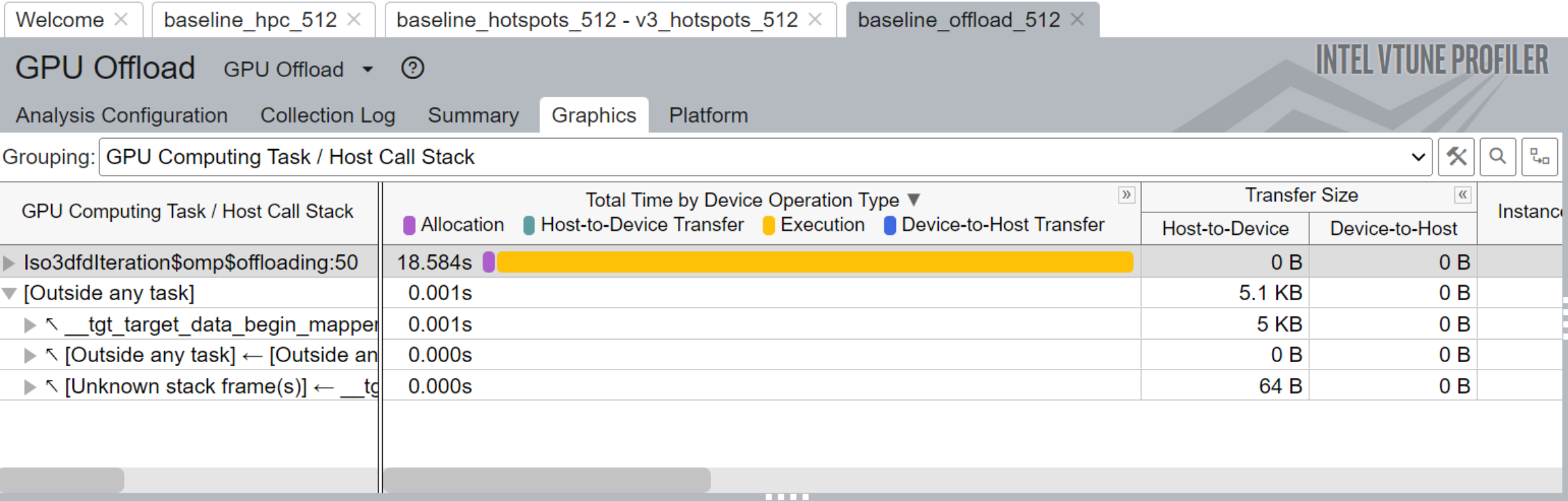


- ☒ Active
- ☒ Idle
- ☒ Stalled
- ☒ GPU Computing Threa...
- ☒ Computing Thread...
- ☒ EU Threads Occup...
- ☒ GPU Memory Access
- Average Bandwidth, GB/sec
- ☒ Read

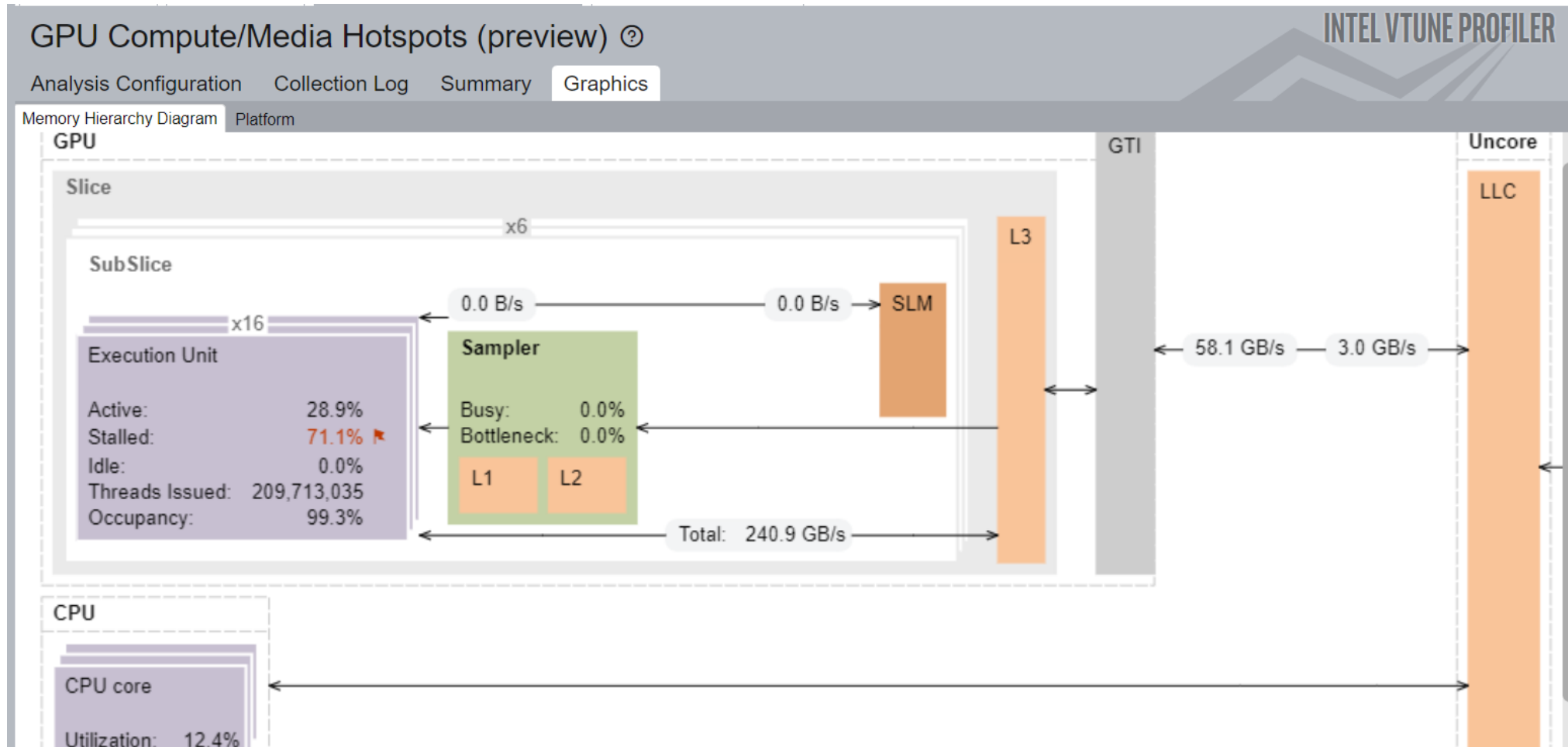
GPU-Offload Platform View for Baseline Version



GPU-Offload Graphics View for Baseline Version



GPU-Hotspots Graphics View for Baseline Version



Code Optimization

- Added 'omp target teams distribute' construct
- Distribute the work among different teams
- Store variables in registers for reuse
- Add OpenMP CPU Threads

GPU-Hotspots Summary View for Optimized Version

Welcome ×baseline_offload_512 ×baseline_hotspots_512 ×v3_hotspots_512 ×

GPU Compute/Media Hotspots (preview) ⓘ🔖INTEL VTUNE PROFILER

Analysis ConfigurationCollection LogSummaryGraphics

⌵ Elapsed Time ⓘ: 53.246s

GPU Time ⓘ: 9.424s

⌵ EU Array Stalled/Idle ⓘ: 67.2% 📈 of Elapsed time with GPU busy 📄

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 ⓘ: 79.2% 📈 of peak value 📄

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 ⓘ
iso3dfditeration\$omp\$offloading:255	9.248s

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

⌵ Occupancy ⓘ: 98.4% of peak value 📄

⌵ Sampler Busy ⓘ: 0.0% of peak value 📄

Welcome ×baseline_hpc_512 ×baseline_hotspots_512 - v3_hotspots_512 ×

GPU Compute/Media Hotspots (preview) ⓘ🔖INTEL VTUNE PROFILER

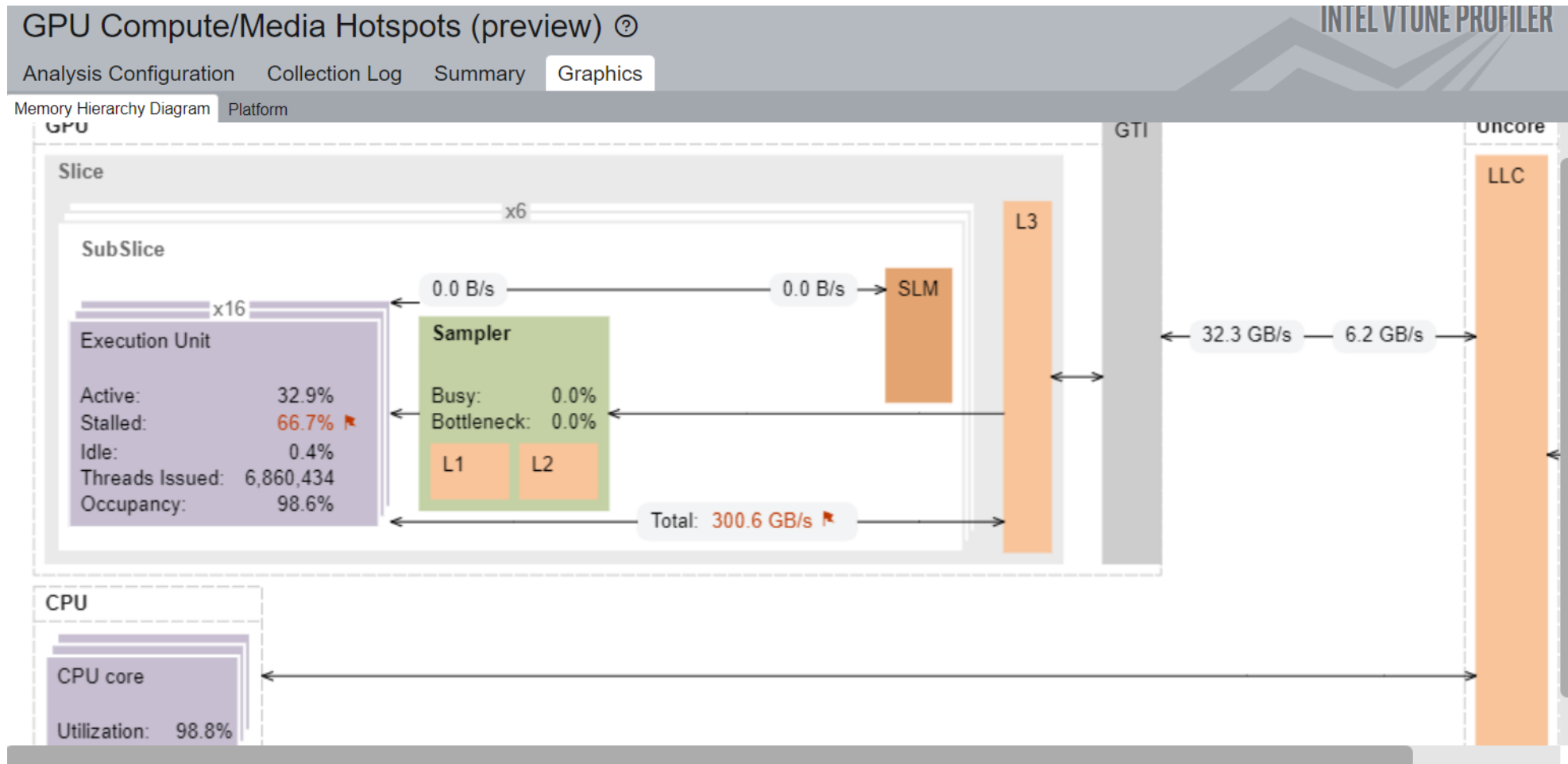
Analysis ConfigurationCollection LogSummary

⌵ Elapsed Time ⓘ: 62.197s - 53.246s = 8.951s

GPU Time ⓘ: 18.187s - 9.424s = 8.764s

➡ EU Array Stalled/Idle ⓘ: 71.1% - 67.2% = 3.9% of Elapsed time with GPU busy

GPU-Hotspots Graphics View for Optimized Version



Recommended Workflow

Using Intel® Analyzers to increase performance

