Profile DPC++ and GPU workload performance

Intel® VTune™ Profiler, Advisor

Vladimir Tsymbal, Technical Consulting Engineer, Intel, IAGS

intel®

Agenda

- Introduction to GPU programming model
- Overview of GPU Analysis in Intel® VTune Profiler
- Offload Performance Tuning
- GPU Compute/Media Hotspots
- A DPC++ Code Sample Analysis Demo

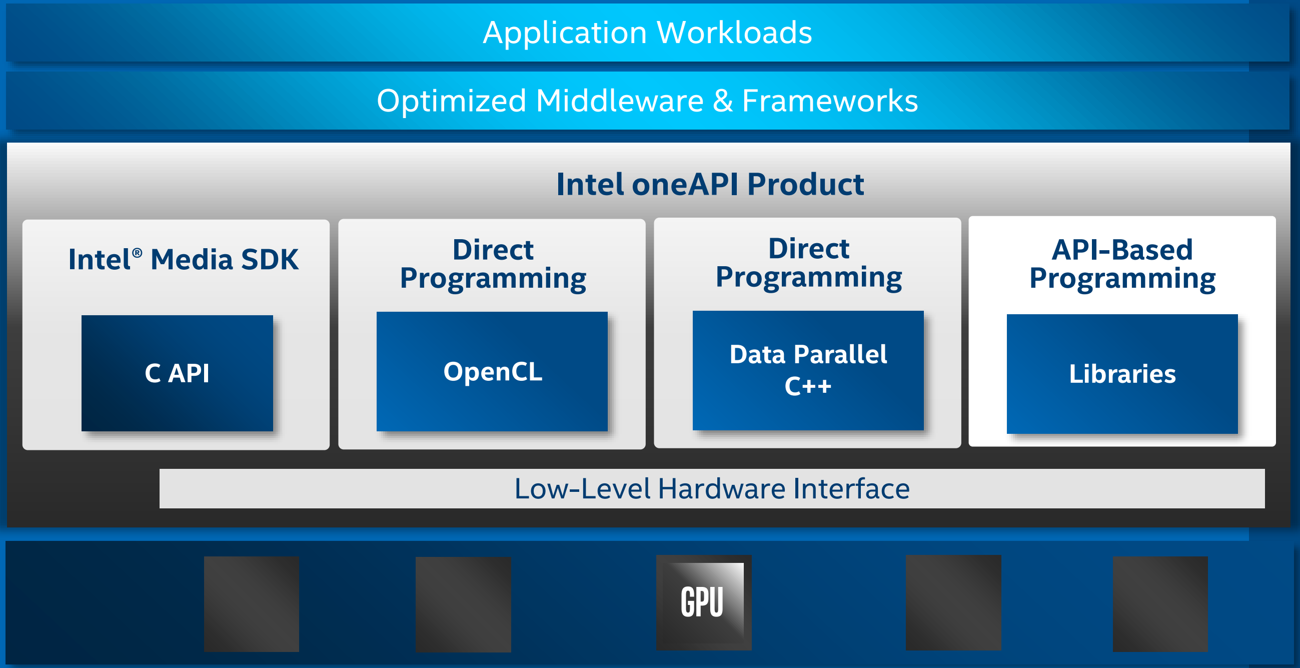
- Using Intel® Advisor to increase performance
- Offload Advisor discrete GPUs
- GPU Roofline for discrete GPUs

Intel GPUs and Programming Model

Gen9
- Most common in mobile, desktop and workstations

Gen11
- Mobile platforms with Ice Lake CPU

Gen12
- Intel Xe-LP
- Tiger Lake CPU



GPU Application Analysis

GPU Compute/Media Hotspots

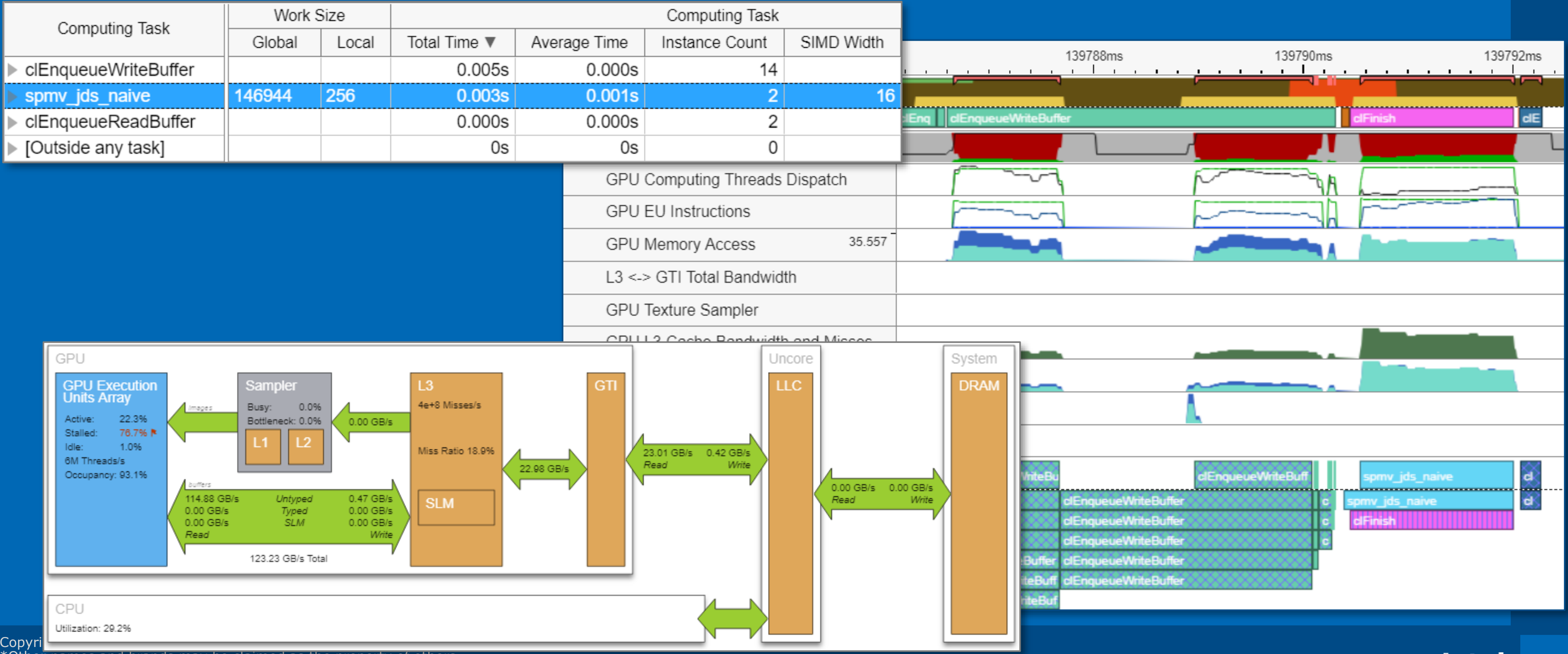
- Visibility into both host and GPU sides
- HW-events based performance tuning methodology
- Provides overtime and aggregated views

GPU In-kernel Profiling

- GPU source/instruction level profiling
- SW instrumentation
- Two modes: Basic Block latency and memory access latency

Identify GPU occupancy and which kernel to profile. Tune a kernel on a fine grain level

GPU Analysis: Aggregated and Overtime Views

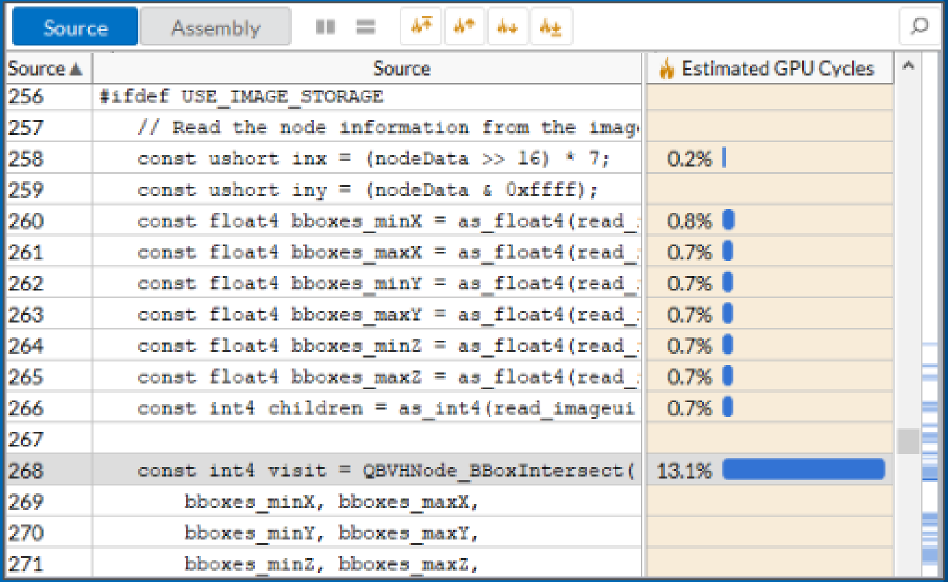


Computing Task	Work Size		Computing Task			
	Global	Local	Total Time ▼	Average Time	Instance Count	SIMD Width
▶ clEnqueueWriteBuffer			0.005s	0.000s	14	
▶ spmv_jds_naive	146944	256	0.003s	0.001s	2	16
▶ clEnqueueReadBuffer			0.000s	0.000s	2	
▶ [Outside any task]			0s	0s	0	

GPU Analysis: In-kernel Profiling

Analyze GPU Kernel Execution

- Find memory latency or inefficient kernel algorithms
- See the hotspot on the DPC++ or OpenCL™ source & assembly code
- Analyze DMA packet execution
  - Packet Queue Depth histogram
  - Packet Duration histogram
- GPU-side call stacks



Offload Performance Tuning

GPU Offload Analysis

Heterogeneous applications

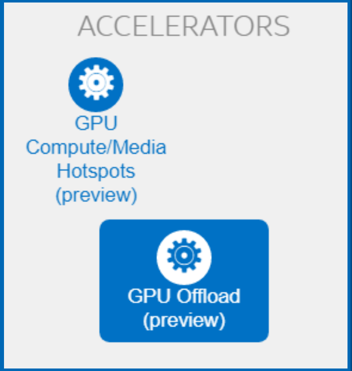
- Off-load models: OpenCL, SYCL, DPC++, OpenMP

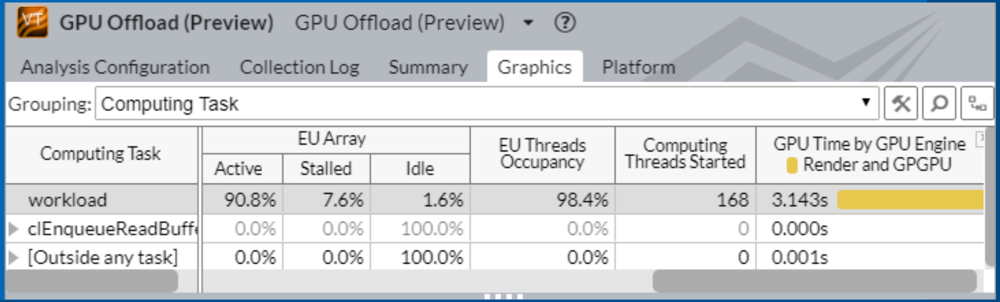
All execution resources in focus

- Explore code execution on various CPU and GPU cores
- Correlate CPU and GPU activity
- Identify whether your application is GPU or CPU bound

Find kernels for further analysis

- Task level analysis
- Kernel efficiency
- Data transfer rates





Computing Task	EU Array			EU Threads Occupancy	Computing Threads Started	GPU Time by GPU Engine Render and GPGPU
	Active	Stalled	Idle			
workload	90.8%	7.6%	1.6%	98.4%	168	3.143s
clEnqueueReadBuff	0.0%	0.0%	100.0%	0.0%	0	0.000s
[Outside any task]	0.0%	0.0%	100.0%	0.0%	0	0.001s

DPC++ Sample app

DPC++ is an extension to SYCL leveraging addition features like:

- Unified shared memory (USM)
- ND-range subgroups
- Ordered queue, etc.

A bunch of sample apps can be found in the OneAPI Toolkit on Github

- We pick-up one: matrix_multiply
- Multiple kernels to select for execution the MM op
- Not fully offload example

```
void multiply1(int msize, int tidx, int numt, TYPE a[][NUM], TYPE b[][NUM], TYPE c[][NUM], TYPE t[][NUM]) {
  int i, j, k;

  default_selector device;
  queue q(device, exception_handler);

  range<2> matrix_range{NUM, NUM};

  buffer<TYPE, 2> bufferA((TYPE*)a, matrix_range);
  buffer<TYPE, 2> bufferB((TYPE*)b, matrix_range);
  buffer<TYPE, 2> bufferC((TYPE*)c, matrix_range);

  q.submit([&](cl::sycl::handler& h) {
    auto accessorA = bufferA.get_access<sycl_read>(h);
    auto accessorB = bufferB.get_access<sycl_read>(h);
    auto accessorC = bufferC.get_access<sycl_read_write>(h);

    h.parallel_for<class Matrix1<TYPE> >(matrix_range,[=](cl::sycl::id<2> ind) {
      int k;
      for (k = 0; k < NUM; k++) {
        accessorC[ind[0]][ind[1]] += accessorA[ind[0]][k] * accessorB[k][ind[1]];
      }
    });
  }).wait_and_throw();
}
```

Declare a deviceQueue

Declare a 2 dimensional range

Declare 3 buffers and Initialize them

Submit our job to the queue

Declare 3 accessors to our buffers. 2RD, 1 WR

Execute matrix multiply in parallel over our matrix_range

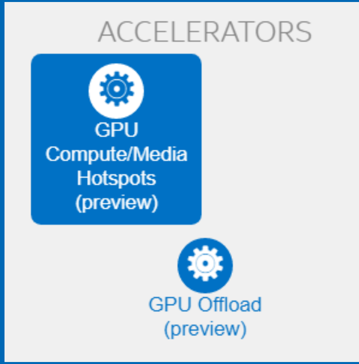
ind is an index into this range

Perform computation ind[0] is row, ind[1] is col

Demo: GPU Offload Analysis

GPU Compute/Media Hotspots

GPU Compute/Media Hotspots



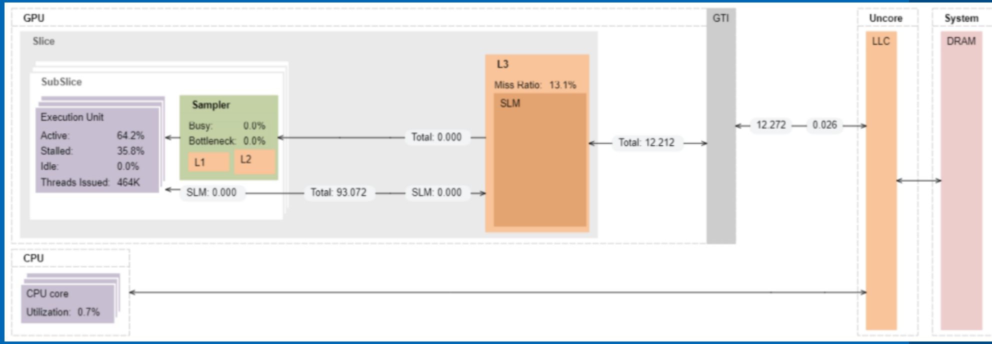
A purely GPU bound analysis

- Although some metrics to SoC are measured

How to gain max performance on GPU

- In an "ideal world" you'd be using optimized IP blocks - "Performance Libraries"

- High level models like DPC++ still give ability to tweak workload layout that better match to GPU architecture
- Need to know GPU blocks as VTune is providing HW level metrics



intel

My GPU architecture

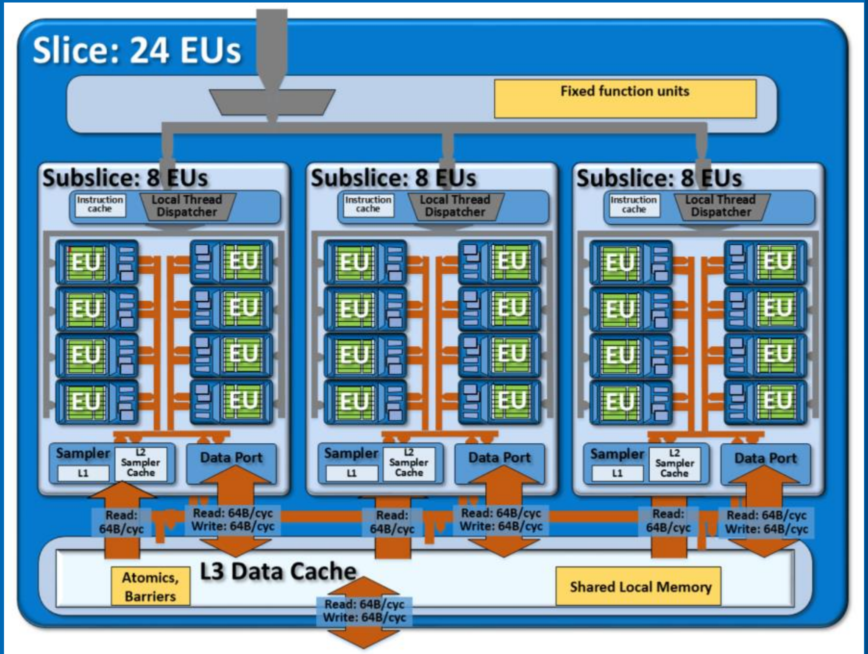
Collection and Platform Info

GPU

Name:	Intel(R) UHD Graphics 620
Vendor:	Intel Corporation
Driver:	27.20.100.8187
EU Count:	24
Max EU Thread Count:	7
Max Core Frequency:	1.1 GHz

GPU OpenCL Info

Version:	OpenCL C 2.0
Max Compute Units:	24
Max Work Group Size:	256
Local Memory:	64 KB
SVM Capabilities:	Fine-grained buffer with atomics



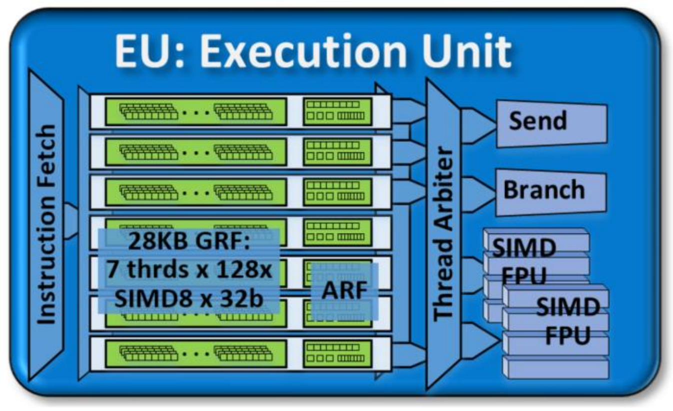
Quickly learn your GPU architecture details from VTune Profiler Summary page

Gen9 GPU EU Details

EU compute architecture includes:

- 24EU x 7thr =168 threads to make busy
- 128 GRF of 32 Byte (accessible as vector-8 of 32-bit data), flexible
- 2 SIMD-4 FPUs of 32-bit FP or INT data
- 16 MAD/cycle (ADD + MUL) x 2 FPUs x SIMD-4)
- 2 additional units: Branch and Send

Main goal is to maximize EU utilization

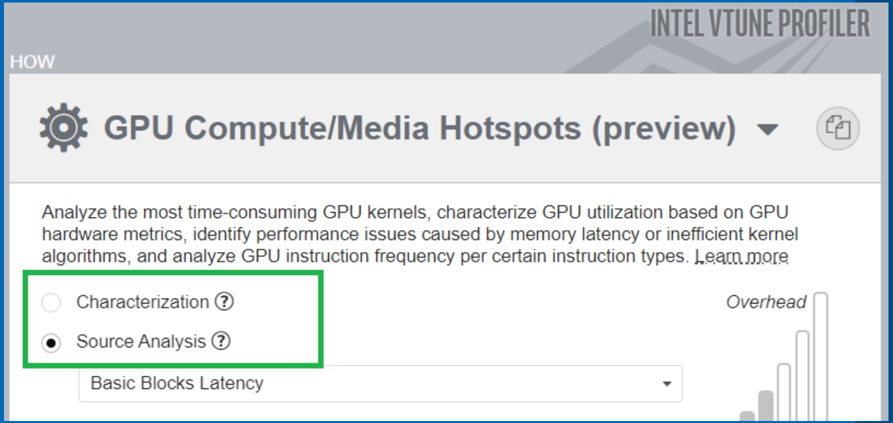


VTune Profiler Analysis

- Select either of GPU analysis configuration:
  - **Characterization** – for monitoring GPU Engine usage, effectiveness, and stalls
  - **Source Analysis** – for identifying performance-critical blocks and memory access issues in GPU kernels

Optimization strategy:

- Maximize effective EU utilization
- Maximize SIMD usage
- Minimize EU stalls due to memory issues



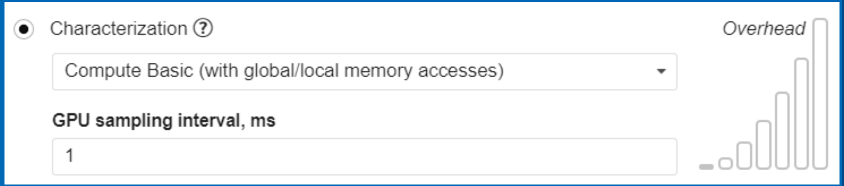
Analyze EU Efficiency and Memory issues

Use the **Characterization** configuration option

- EUs activity: EU Array Active, EU Array Stalled, EU Array Idle, Computing Threads Started, and Core Frequency

Select **Overview** or **Compute Basic** metric

- additional metrics: Memory Read/Write Bandwidth, GPU L3 Misses, Typed Memory Read/Write Transactions



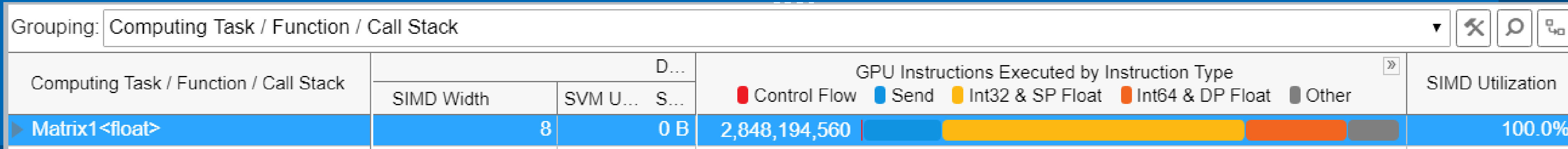
Grouping: Computing Task

Computing Task	EU Array			EU Threads Occupancy	Computing Threads Started	EU Instructions			L3 Bandwidth, GB/sec
	Active	Stalled	Idle			IPC Rate	2 FPUs active	Send active	
Matrix1<float>	62.6%	37.4%	0.0%	95.1%	131,040	1.583	34.9%	6.8%	89.763

Analyze GPU Instruction Execution

Use the **Dynamic Instruction Count** preset

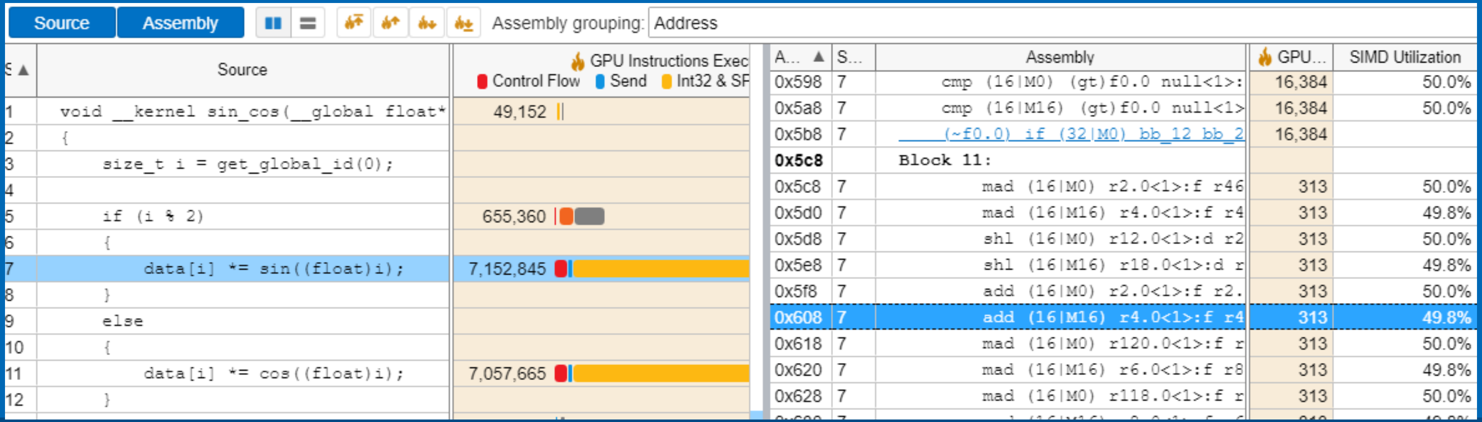
- A breakdown of instructions executed by the kernel
- Groups of instructions
  - Control flow
  - Send
  - Synchronization
  - Int16 & HP Float | Int32 & SP Float | Int64 & DP Float
  - Other



Analyze Source Code

Use the **Source Analysis** configuration option

- Analyze a kernel of interest for basic block latency or memory latency issues
- Enable both the **Source** and **Assembly** panes to get a side-by-side view



Demo: GPU Compute/Media Hotspots

Intel Advisor for dGPU

Intel® Advisor workflows

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

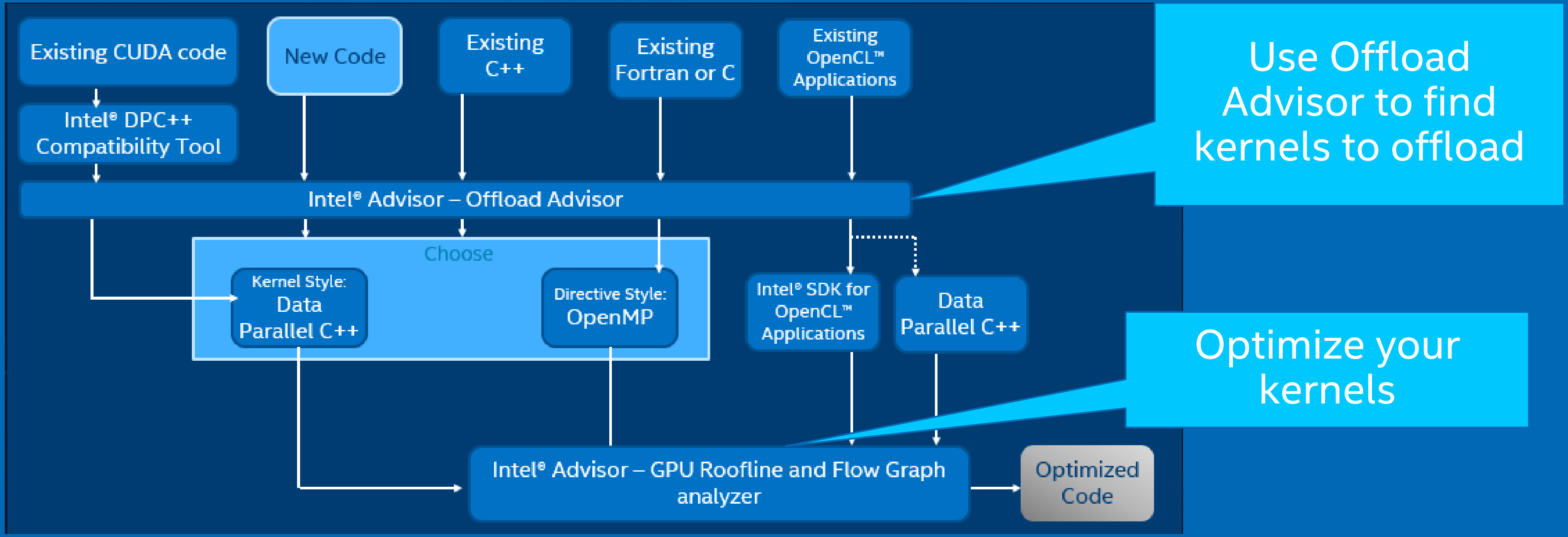
Model, tune, and test multiple threading designs.

Build Heterogeneous Algorithms

Create and analyze data flow and dependency computation graphs.

Learn More: software.intel.com/advisor

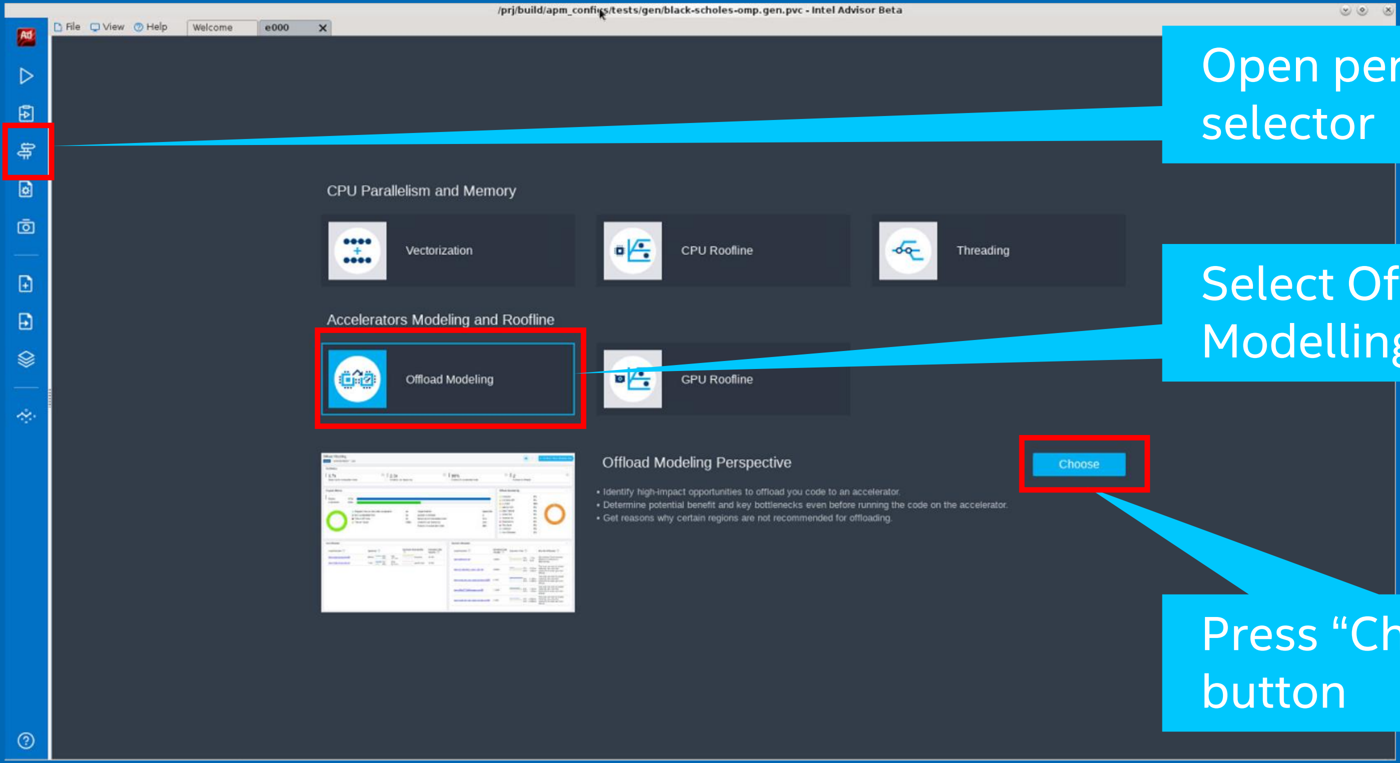
Using Intel® Advisor to increase performance



Intel® Advisor

- Offload Advisor new UI (coming in beta10)
- Support for target GPU devices

Select offload modelling perspective

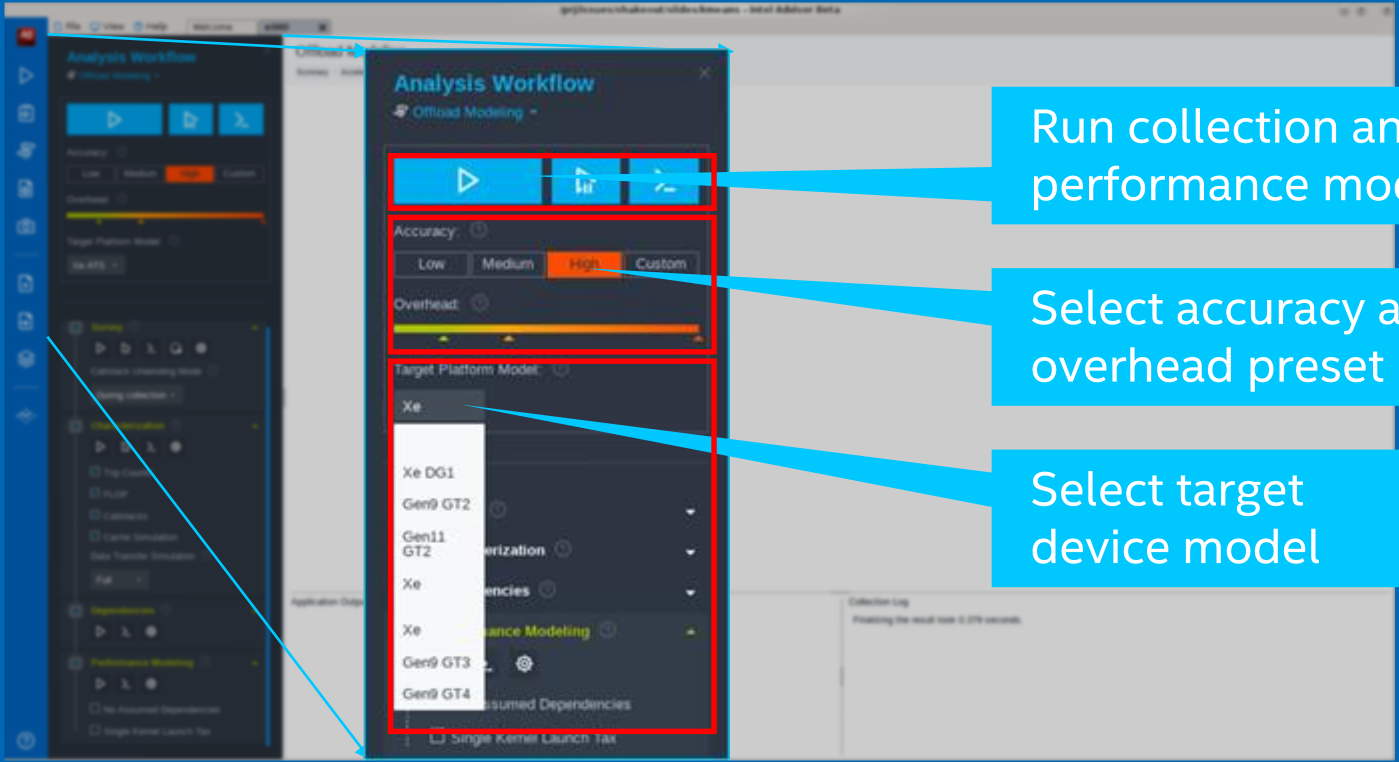


Open perspective selector

Select Offload Modelling

Press “Choose” button

Offload Modelling Workflow



Run collection and performance model

Select accuracy and overhead preset

Select target device model

Offload Advisor: CLI interface

1. Use $APM/collect.py script to run required analysis types:

```
advixe-python $APM/collect.py --config=<target> <my_project_directory>
        -- ./myapp [app_parameters]
```

2. Generate Offload report:

```
advixe-python $APM/analyze.py <my_project_directory> -o <path-to-report-dir>
```

There are lots options both for collect.py and analyze.py to tune hardware and software parameters

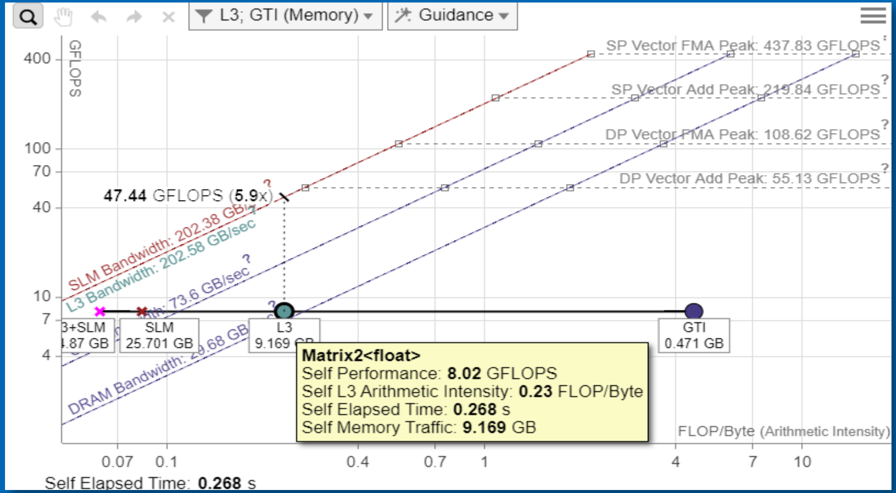
Intel® Advisor

GPU Roofline

GPU Roofline chart

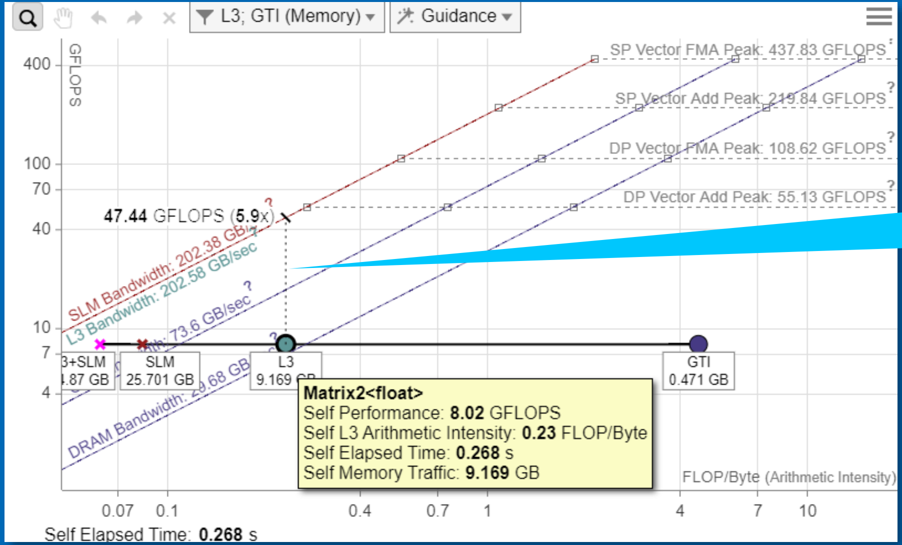
GPU Roofline Performance Insights

- Highlights poor performing loops
- Shows performance 'headroom' for each loop
  - Which can be improved
  - Which are worth improving
- Shows likely causes of bottlenecks
  - Memory bound vs. compute bound
- Suggests next optimization steps



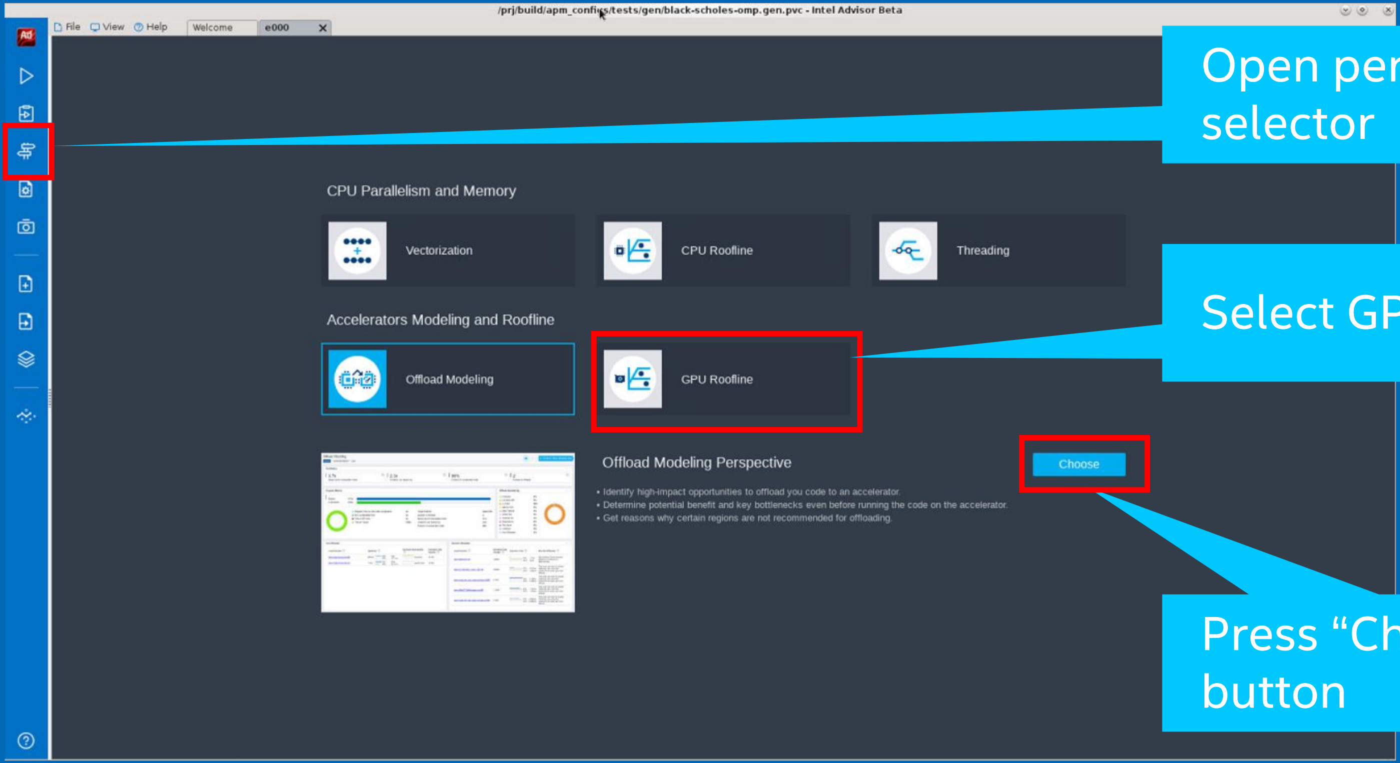
Intel® Advisor GPU Roofline

See how close you are to the system maximums (rooflines)



Roofline indicates room for improvement

Select offload modelling perspective



Open perspective selector

Select GPU Roofline

Press “Choose” button

GPU Roofline: CLI interface

Run 2 collections with **--profile-gpu** option:

```
advixe-cl -collect=survey --profile-gpu --project-dir=<my_project_directory>
-- ./myapp [app_parameters]
```

```
advixe-cl -collect=tripcounts --flop --profile-gpu --project-
dir=<my_project_directory> -- ./myapp [app_parameters]
```

Generate a GPU Roofline report:

```
advixe-cl --report=roofline --gpu --project-dir=<my_project_directory> --
report-output=roofline.html
```

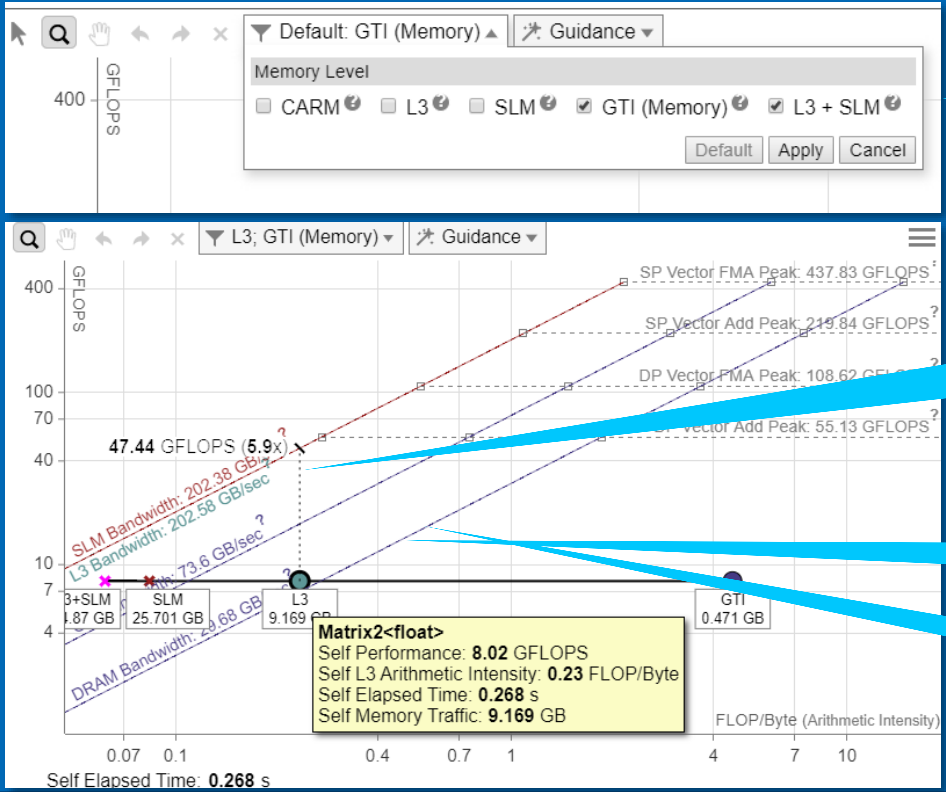
Generate a GPU Roofline report for **integer** operations:

```
advixe-cl --report=roofline --gpu -data-type=int --project-
dir=<my_project_directory> --report-output=roofline.html
```

Open the generated roofline.html in a web browser to visualize GPU performance.

intel

Find Effective Optimization Strategies



Configure levels to display

Shows performance headroom for each loop

Likely bottlenecks

Suggests optimization next steps

Links

Intel oneAPI

Intel® VTune™ Profiler

GPU Offload Analysis

GPU Compute/Media Hotspots Analysis

Intel Advisor

Intel Advisor Cookbooks

DPC ++ spec page

Sample apps on Github

Questions

NOTICES AND DISCLAIMERS

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.

All product plans and roadmaps are subject to change without notice.

Intel technologies may require enabled hardware, software or service activation.

Results have been estimated or simulated.

No product or component can be absolutely secure.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. Other names and brands may be claimed as the property of others. © Intel Corporation.

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.
- Copyright © 2018, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

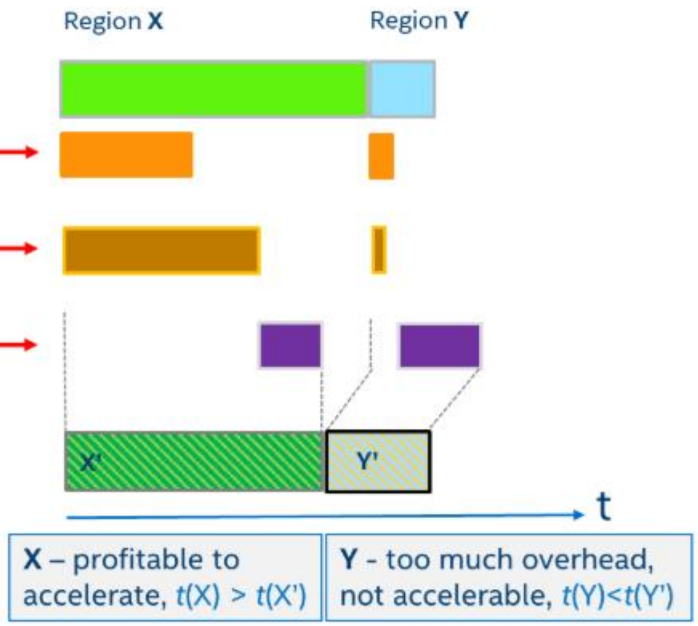
Backup

Performance model

Execution time on baseline platform (CPU)

- Estimated execution time on accelerator, assuming bound exclusively by **Compute**
- Estimated execution time on accelerator, Estimate assuming bound exclusively by caches/memory
- Offload Tax estimate (data transfer + invoke)

Final estimated time on target GPU platform



$$t_{region} = \max(t_{compute}, t_{memory\ subsystem}) + t_{data\ transfer\ tax} + t_{kernel\ launch}$$

Set Up System for GPU Analysis

Install the Intel® oneAPI software
https://software.intel.com/content/www/us/en/develop/tools/oneapi.html

Linux* systems: Linux kernel 4.14 and higher

Switch to a root mode

or

- Add your username to the *video* group
- Set the value of dev.i915.perf_stream_paranoid sysctl option to 0
- Disable Hangcheck
```
sudo sh -c "echo N> /sys/module/i915/parameters/enable_hangcheck"
```

Windows: You can install a GPU driver for your system from https://downloadcenter.intel.com.

Please visit
https://software.intel.com/content/www/us/en/develop/documentation/advisor-user-guide/top/intel-advisor-beta-gpu-roofline.html

Python API

Create Python scripts for data analysis with Python API

How to print collected data

```
advixe-python /advisor_install_dir/pythonapi/examples/survey_gpu.py
/work_dir/project_name
```

```
carm_traffic_gb_______________________________: 0.698458
computing_task________________________________: SumPointsKernel
computing_task_average_time___________________: 0.000250667
computing_task_id_____________________________: 3
computing_task_instance_count_________________: 150
computing_task_purpose________________________: Compute
computing_task_simd_width_____________________: 32
computing_task_svm_usage_type_________________:
computing_task_total_time_____________________: 0.0376
computing_threads_started_____________________: 2698
data_transferred_size_________________________:
data_transferred_total_gb_sec_________________:
elapsed_time__________________________________: 0.0376
```

```
gpu_compute_performance_fp_ai_________________: 4.00678
gpu_compute_performance_gflop_________________: 0.0384192
gpu_compute_performance_gflops________________: 1.02179
gpu_compute_performance_gintop________________: 0.135245
gpu_compute_performance_gintops_______________: 3.59693
gpu_compute_performance_gmixop________________: 0.173664
gpu_compute_performance_gmixops_______________: 4.61872
gpu_compute_performance_int_ai________________: 14.1048
gpu_compute_performance_mix_ai________________: 18.1116
gpu_memory_bandwidth_gb_sec_read______________: 0.248483
gpu_memory_bandwidth_gb_sec_write_____________: 0.00653106
gpu_memory_data_transferred_gb_read___________: 0.00934298
gpu_memory_data_transferred_gb_write__________: 0.000245568
```

```
SumPointsKernel: 3
    ?  :    : ?         : 0.0622656
    ?  :    : SYNC      : 0.0028152
    FP : 32: BASIC      : 0.0384192
    FP : 32: MOVE       : 0.0380736
    FP : 16: MOVE       : 1.44e-05
    INT: 32: BASIC      : 0.0990432
    INT: 32: FMA        : 0.000936
    INT: 32: STORE      : 0.000216
    INT: 32: LOAD       : 0.0148032
    INT: 32: SLM_STORE: 0.0238176
    INT: 32: SLM_LOAD : 0.0196704
    INT: 32: MOVE       : 0.0462312
    INT: 32: BIT        : 0.0037584
    INT: 64: BASIC      : 0.029952
    INT: 64: MOVE       : 0.0001512
    INT: 16: BASIC      : 0.0012096
    INT: 16: MOVE       : 0.0003816
    INT: 16: BIT        : 0.0003456
```