



oneAPI

INTRODUCTION TO OFFLOAD ADVISOR

Klaus-Dieter Oertel

Intel IAGS

oneAPI@CERN 24-Mar-2020

INTEL® ADVISOR (BETA)

DESIGN ASSISTANT – DESIGN FOR MODERN HARDWARE

Offload Advisor

- Identify opportunities for offload to an accelerator

Vectorization Advisor

- Add and optimize vectorization

Roofline Analysis

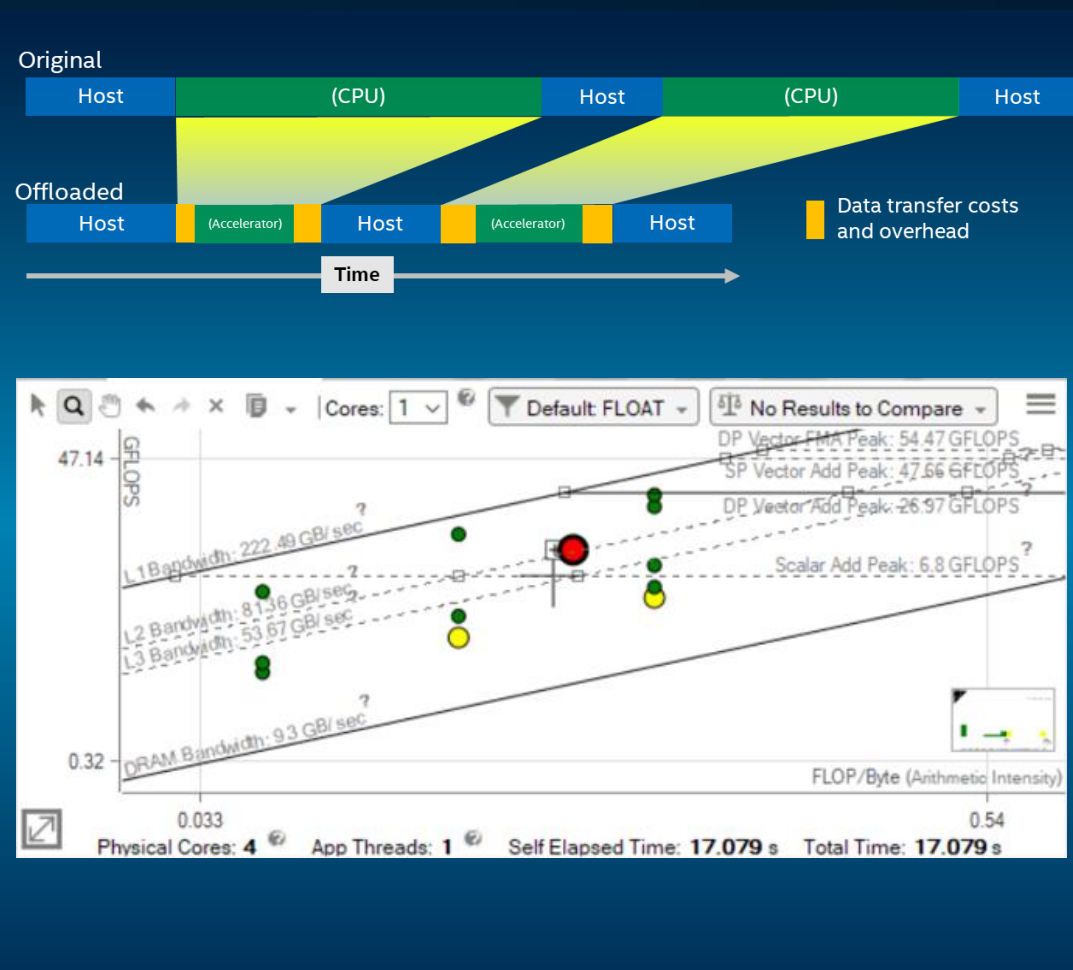
- Optimize CPU/GPU code for memory and compute

Threading Advisor

- Add effective threading to unthreaded applications

Flow Graph Analyzer

- Create and analyze efficient flow graphs



AGENDA

- Introduction to Offload Advisor
- Command line tips
- Understanding the performance modelization
- GPU Roofline Analysis

INTRODUCTION TO OFFLOAD ADVISOR

[Optimization Notice](#)

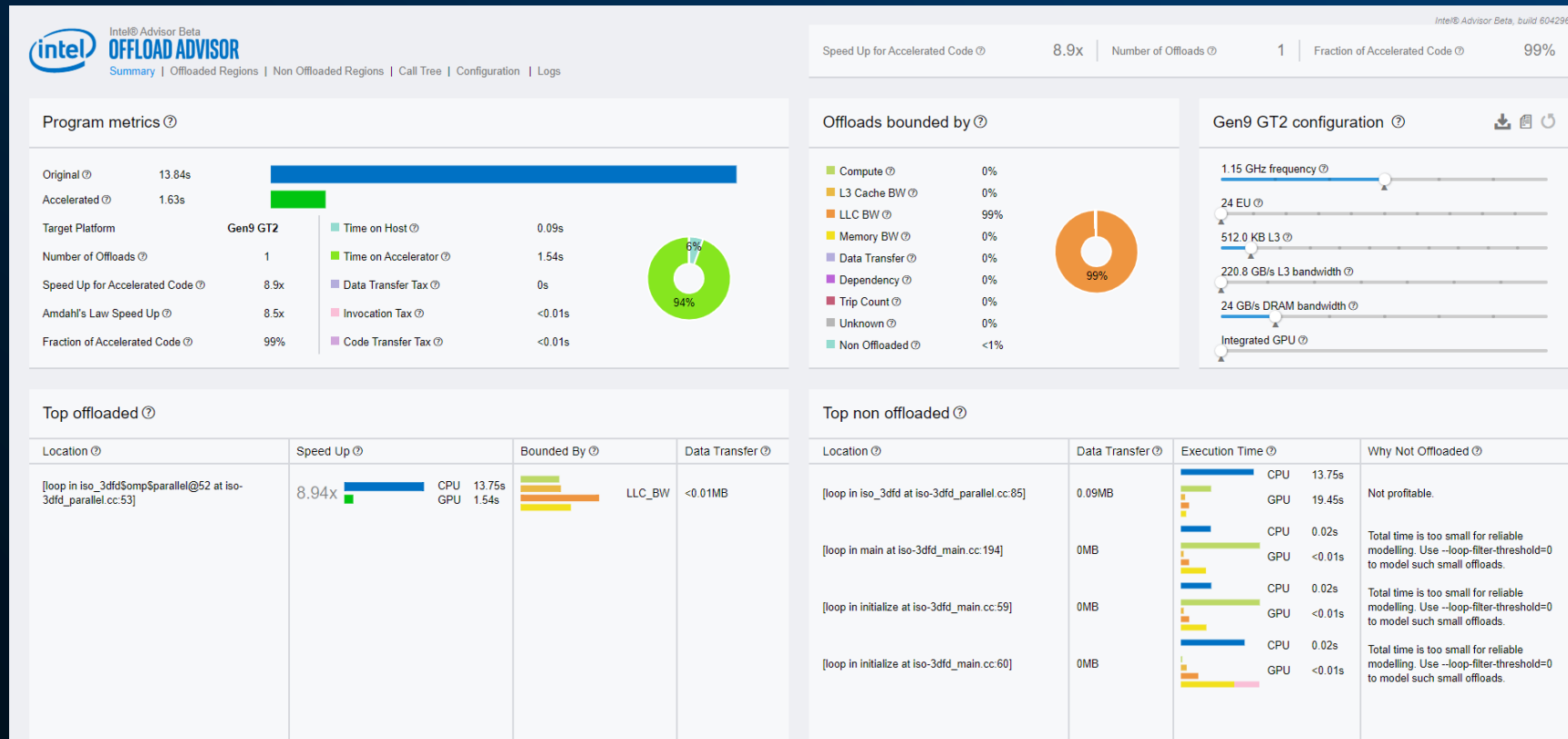
Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.

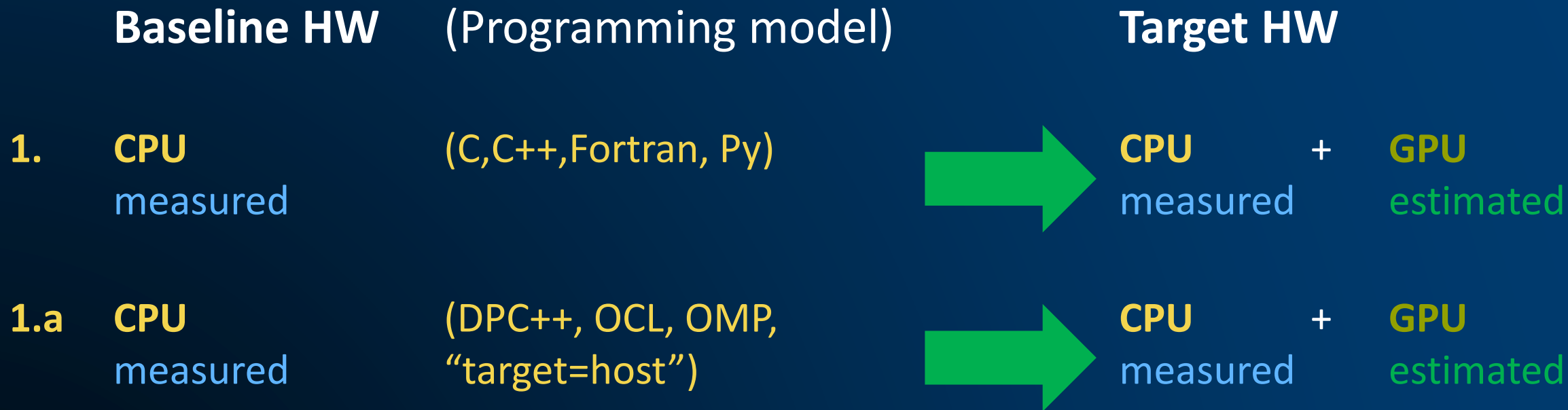


INTEL OFFLOAD ADVISOR (BETA)




- Starting from a baseline binary (running on CPU):
 - Helps defining which sections of the code should run on a given accelerator
 - Provides performance projection on accelerators (currently gen9 and gen11)



MODELING FLOWS SUPPORTED: NOW

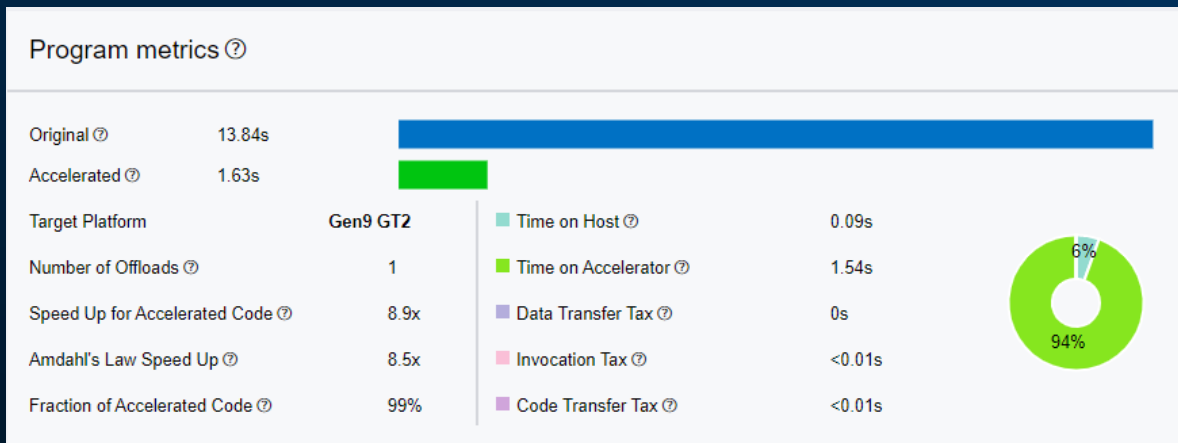


MODELING FLOWS SUPPORTED: NOW + COMING SOON

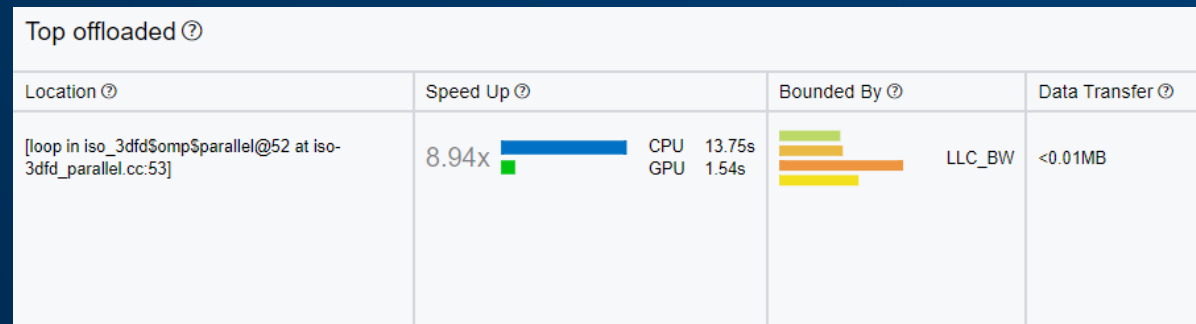
	Baseline HW	(Programming model)		Target HW	
1.	CPU measured	(C,C++,Fortran, Py)		CPU + measured	GPU estimated
1.a	CPU measured	(DPC++, OCL, OMP, "target=host")		CPU + measured	GPU estimated
2	CPU+iGPU measured	(C, C++, Fortran, DPC++, OCL, OMP)		CPU + measured	GPU estimated

FROM YOUR CPU APPLICATION, YOU WONDER:

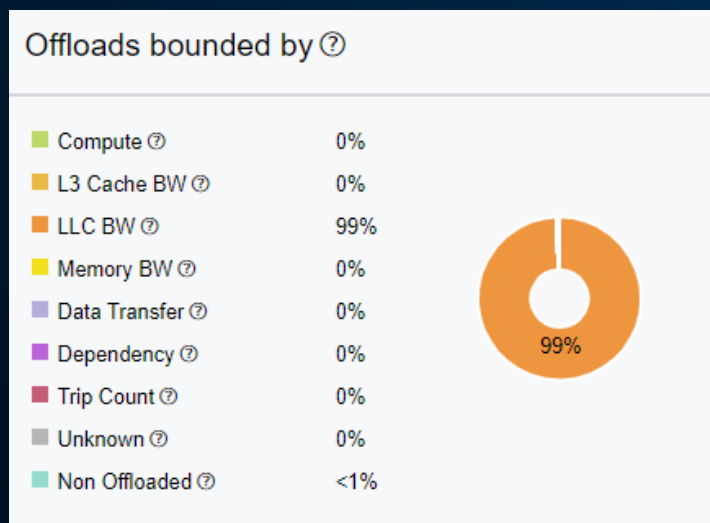
- How your code might perform on an accelerator ?



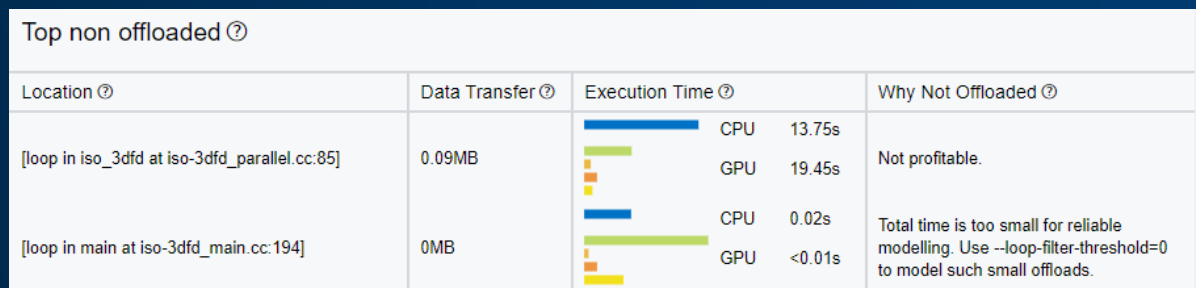
- What should you offload ?



- What might be limiting your performance on the accelerator ?



- What are the bad candidates for offload and Why ?



TOP OFFLOADED IN DEPTH

- Provides a detailed description of each loop interesting for offload
 - 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 run on the accelerator

This is where you will use DPCPP or OMP target for offload

The screenshot displays the Intel Advisor Beta Offload Advisor interface. At the top, it shows performance metrics: Speed Up for Accelerated Code at 8.9x, Number of Offloads at 1, and Fraction of Accelerated Code at 99%. Below this is a table with columns for Hierarchy, Total Data Transferred from GPU to CPU (MB), Trip Counts (Average Trip Count, Call Count), L3 Cache (Total L3 Traffic), LLC (Total LLC Access), Memory (Total Memory Traffic), Instruction & Traffic Counts (FPU Util, FLOP per Cycle), and Diagnostics. The first row is highlighted in blue, showing a loop with 57600 average trip counts and 102 call counts. To the right, a code snippet is shown with a red arrow pointing to a line containing a pragma omp parallel directive. Below the code, it states 'No memory objects data' and 'No memory object tracked for selected row.'

Hierarchy	Total Data Transferred from GPU to CPU (MB)	Average Trip Count	Call Count	Total L3 Traffic (GB)	Total LLC Access (GB)	Total Memory Traffic (GB)	FPU Util (GFLOP/s)	FLOP per Cycle	Diagnostics
[loop in iso_3dfd\$omp\$parallel@52 at i	<0.01	57600	102	174.250	113.259	23.637	7.896	7.896	In whole loop
[loop in iso_3dfd\$omp\$parallel@52 at	0	30	5875200	173.894	113.257	23.637	7.947	7.947	
[loop in iso_3dfd\$omp\$parallel@52		<1	<1	0	0	0	0	0	Aggregated e

```
51 #pragma omp parallel for OMP_SCHEDULE num_threads(1) c
52 for(int iz=HALF_LENGTH; iz<n3-HALF_LENGTH; iz++) {
53     for(int iy=HALF_LENGTH; iy<n2-HALF_LENGTH; iy++) {
54         #pragma omp simd
55             for(int ix=HALF_LENGTH; ix<n1-HALF_LENGTH; ix+
56                 int offset = iz*dimn1n2 + iy*n1 + ix;
57                 float value = 0.0;
58                 value += ptr_prev[offset]*coeff[0];
59                 for(int ir=1; ir<=HALF_LENGTH; ir++) {
60                     value += coeff[ir] * (ptr_prev[offset
61                     value += coeff[ir] * (ptr_prev[offset
62                     value += coeff[ir] * (ptr_prev[offset
```

NON OFFLOADED IN DEPTH

- Explains why Advisor doesn't recommend a given loop for offload
 - Dependency issues
 - Not profitable
 - Total time is too small

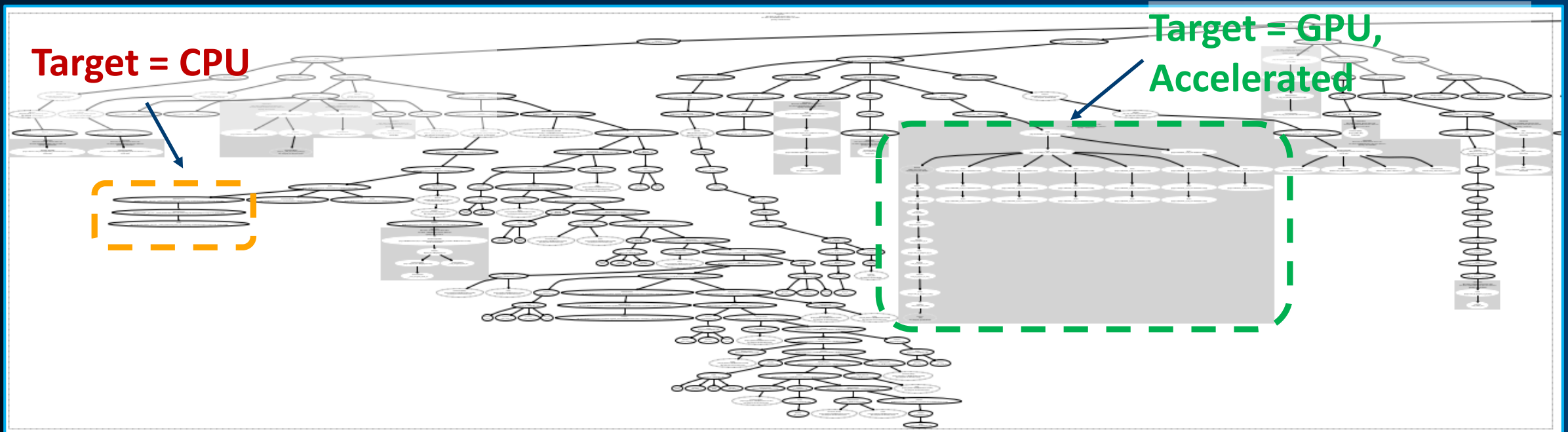
Intel® Advisor Beta
OFFLOAD ADVISOR
Summary | Offloaded Regions | **Non Offloaded Regions** | Call Tree | Configuration | Logs

Speed Up for Accelerated Code 8.9x | Number of O

Hierarchy	Information >			Potential Offload <		Potential Speed Up for Whole Region	Column configurator
	Loop identified?	Estimated Execution Time on Accelerator (+Host) (s)	Bounded By	Fraction Offloaded (%)	Why Not Offloaded		
[loop in iso_3dfd at iso-3dfd_parallel.cc]		1.539		100.00	Not profitable.	0.7068x	
[loop in main at iso-3dfd_main.cc:194]		0.020		0	Total time is too small for reliable modelling. Use --loop-filter-threshold=0 to model such small offloads.	41338.1565x	
> [loop in initialize at iso-3dfd_main.cc:59]					Total time is too small for reliable modelling. Use --loop-filter-threshold=0 to model such small offloads.	640.4016x	Custom filter

PROGRAM TREE

The program tree offers another view of the proportion of code that can be offloaded to the accelerator.



COMMAND LINE TIPS

[Optimization Notice](#)

Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.



BEFORE YOU START TO USE OFFLOAD ADVISOR

- The only strict requirement for compilation and linking is full debug information:
 - g:** Requests full debug information (compiler and linker)
- Offload Advisor supports any optimization level, but the following settings are considered the optimal requirements:
 - O2:** Requests moderate optimization
 - no-ipo:** Disables inter-procedural optimizations that may inhibit Offload Advisor to collect performance data (Intel® C++ & Fortran Compiler specific)

SOURCE OFFLOAD ADVISOR

- To set up the Intel® Advisor Beta environment, run one of the shell script:

```
source <ONEAPI_INSTALL_DIR>/setvars.sh
```

or

```
source <ADV_INSTALL_DIR>/env/vars.sh
```

- This script sets all required Intel Advisor environment variables, including APM, which points to **<ADV_INSTALL_DIR>/perfmodes**
- This is the location of the Offload Advisor scripts in the Intel® Advisor Beta installation directory



The performance modeling functionality is available on Linux* OS only

HOW DOES IT WORK ?

- Easy to collect data and generate output with batch mode:

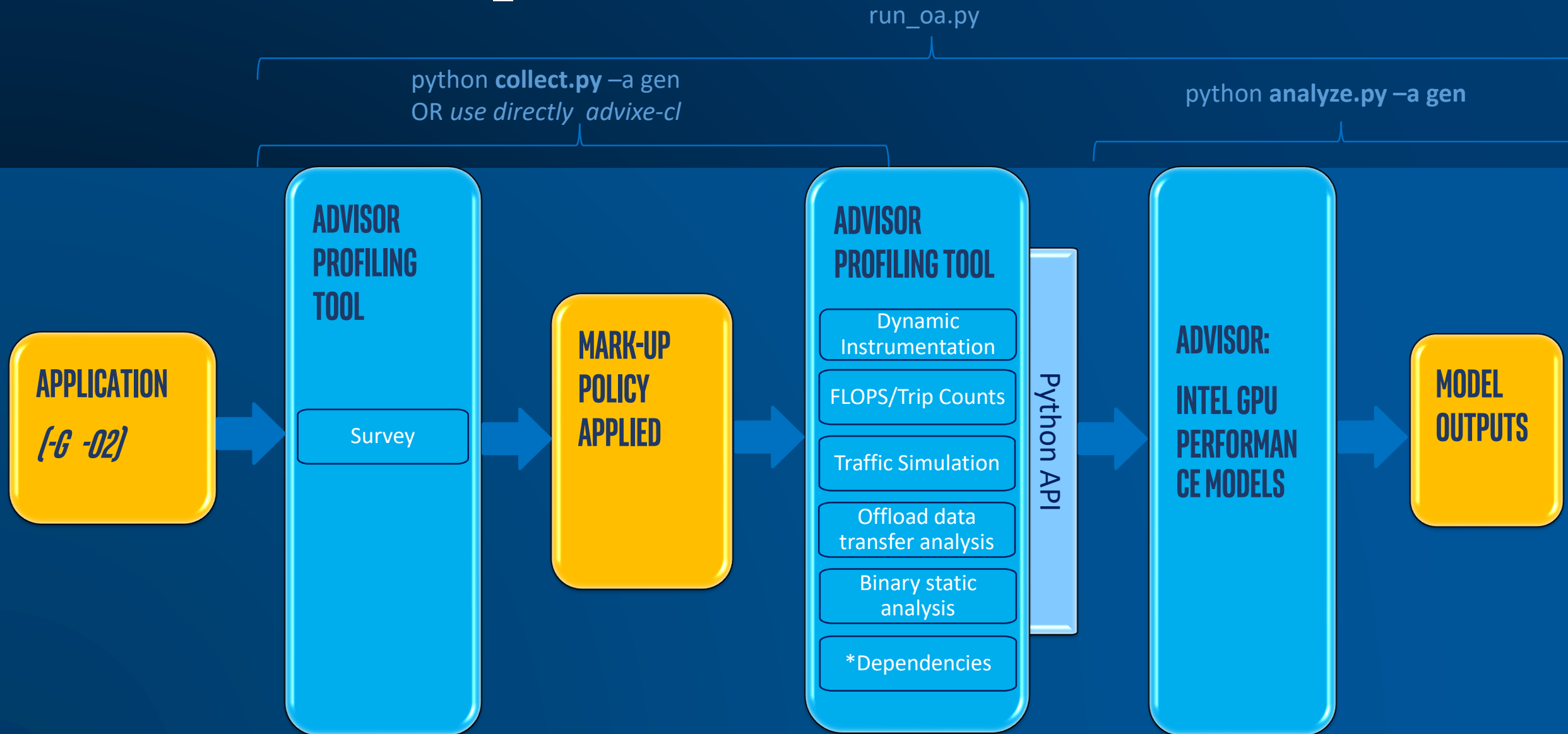
```
advixe-python <ADV_INSTALL_DIR>/perfmodels/run_oa.py <project_dir>  
  --config gen9 --out-dir <project_dir> [--options] -- <app> <app_args>
```

- By default, **run_oa.py** marks up all regions and only selects the most profitable ones for analysis
- To generate the report.html, uses the following command:

```
advixe-python $APM/analyze.py <project_dir> --config gen9  
  --out-dir <project_dir> [--options]
```

```
u31313@s001-n004:/opt/intel/inteloneapi/advisor/latest/perfmodels$ ls  
accelerators    analyze.py      collect.py      debug.so        environ.py     oa_wrapper.so  shared.so      toml  
analyze_impl.so collect_impl.so compute_stats.py dot_graph.so    helpers        run_oa.py      template       tree.so
```

RUN_OA.PY: WHAT IS RUNNING BEHIND?



[Optimization Notice](#)

Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.



OFFLOAD ADVISOR OUTPUT OVERVIEW

- **report.html**: Main report in HTML format
- **report.csv** and **whole_app_metric.csv**: Comma-separated CSV files
- **program_tree.dot**: A graphical representation of the call tree showing the offloadable and accelerated regions
- **program_tree.pdf**: A graphical representation of the call tree
Generated if the DOT(GraphViz*) utility is installed
1:1 conversion from the **program_tree.dot** file
- **JSON** and **LOG** files that contain data used to generate the HTML report and logs, primarily used for debugging and reporting bugs and issues

WANT TO AVOID DEPENDENCY CHECKING ?

- Dependency adds a lot of time to the collection and you might want to remove it.
- Add the option `-c basic` for the collection:

```
advixe-python <ADV_INSTALL_DIR>/perfmodels/run_oa.py <path_to_result_dir>  
-config gen9 -c basic --out-dir <path_to_result_dir> [--options] -- <app>
```

- Add the option `--assume-parallel` for the analysis:

```
advixe-python $APM/analyse.py <project_dir> --assume-parallel --config  
gen9 [--options] -- <app_binary> [app_options]
```

HARDCORE VERSION (BUT MORE CONTROL)

- You might want to run the command lines independently to tweak the parameters
- A good start is to use `run_oa.py` script with `--dry-run` to see the list of command lines and retrieve the cache configuration of the target accelerator.
- The next command will output the different command lines for doing separate analyses without running advisor collection.
- **`advixe-python <ADV_INSTALL_DIR>/perfmodels/run_oa.py <path_to_result_dir> --dry-run -config gen9 -c basic --out-dir <path_to_result_dir> [--options] -- <app>`**

HARDCORE VERSION (BUT MORE CONTROL)

- We start with the survey
- **advixe-cl --collect=survey --auto-finalize --stackwalk-mode=online --static-instruction-mix --project-dir=./oa_report - my_app**
- The survey times your application and run some static analysis on the binary without impact on the application's performance.
 - Sampling
 - Binary static analysis
 - Static code analysis (compiler and debug infos)

HARDCORE VERSION (BUT MORE CONTROL)

- We continue with the trip count and cache simulation
- **advixe-cl --collect=tripcounts -return-app-exitcode -flop -stacks -auto-finalize -ignore-checksums -enable-data-transfer-analysis -track-heap-objects -profile-jit -cache-sources -enable-cache-simulation -cache-config=1:8w:32k/1:64w:512k/1:16w:8m --project-dir=./oa_report - my_app**
- The tripcounts with `-flop` and `-cache-simulation` counts:
 - The number of iterations in your loops
 - The number of operations
 - Evaluate the data transfers between memory subsystems configured with `-cache-config`
- This analysis has usually $\approx 10x$ speeddown

HARDCORE VERSION (BUT MORE CONTROL)

- Optional step: Dependency analysis
- **advixe-cl --collect=dependencies --loops="total-time>5" --filter-reductions --loop-call-count-limit=16 --project-dir=./oa_report -- my_app**
- Detects data dependencies in your loop by checking your memory accesses
- This analysis has an important impact on the performance
- It is up to the user to define how loops will be selected for this analysis, here we use – loops="total-time>5" which select all loops impacting more than 5% of the overall time

HARDCORE VERSION (BUT MORE CONTROL)

- Last step: Generating the report
- 2 Cases:
 - You ran the dependency analysis:

```
advixe-python $APM/analyse.py ./oa_report --config gen9 --out-dir  
oa_report - my_app
```

- You didn't run the dependency analysis

```
advixe-python $APM/analyse.py ./oa_report --config gen9 --assume-  
parallel --out-dir oa_report - my_app
```

UNDERSTANDING THE PERFORMANCE MODELIZATION

THE MECHANISMS BEHIND

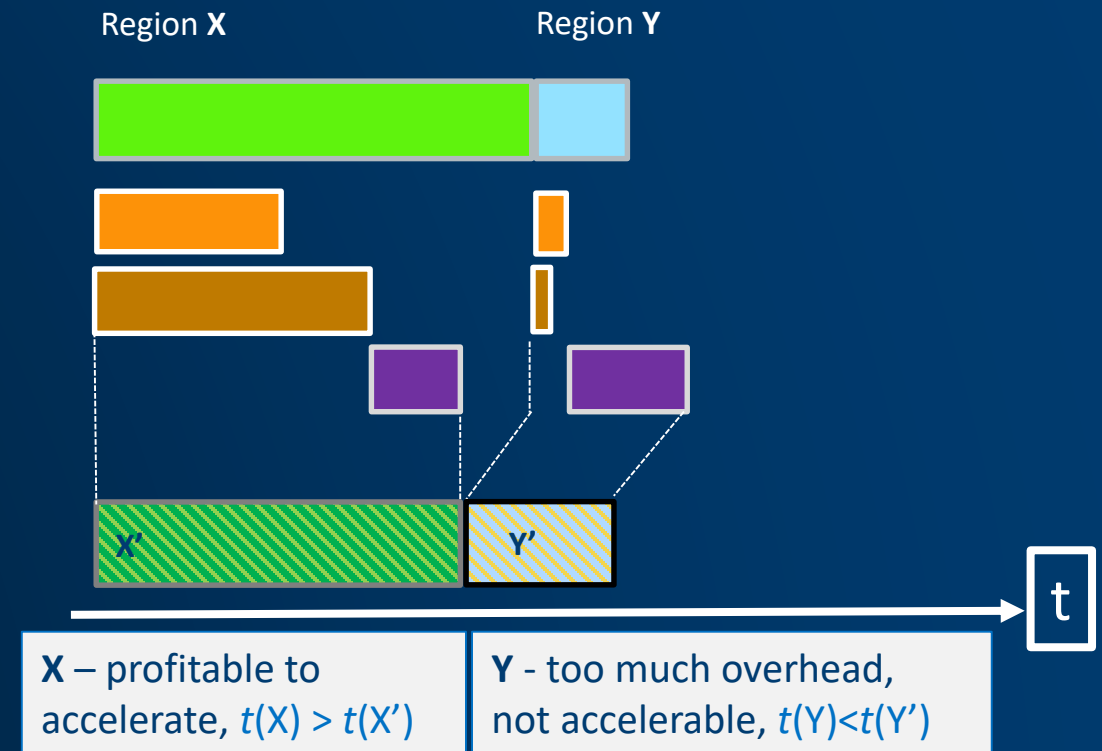
First order analytical modeling pillars:

- Compute throughput model
- Memory sub-system model
- Offload data transfer modeling

Execution time on baseline platform (CPU)

- Execution time on accelerator. Estimate assuming bound exclusively by Compute
- Execution time on accelerator. Estimate assuming bound exclusively by caches/memory
- Offload Tax estimate (data transfer + invoke)

Final estimated time on target platform (eg GPU)



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

GPU ROOFLINE ANALYSIS

[Optimization Notice](#)

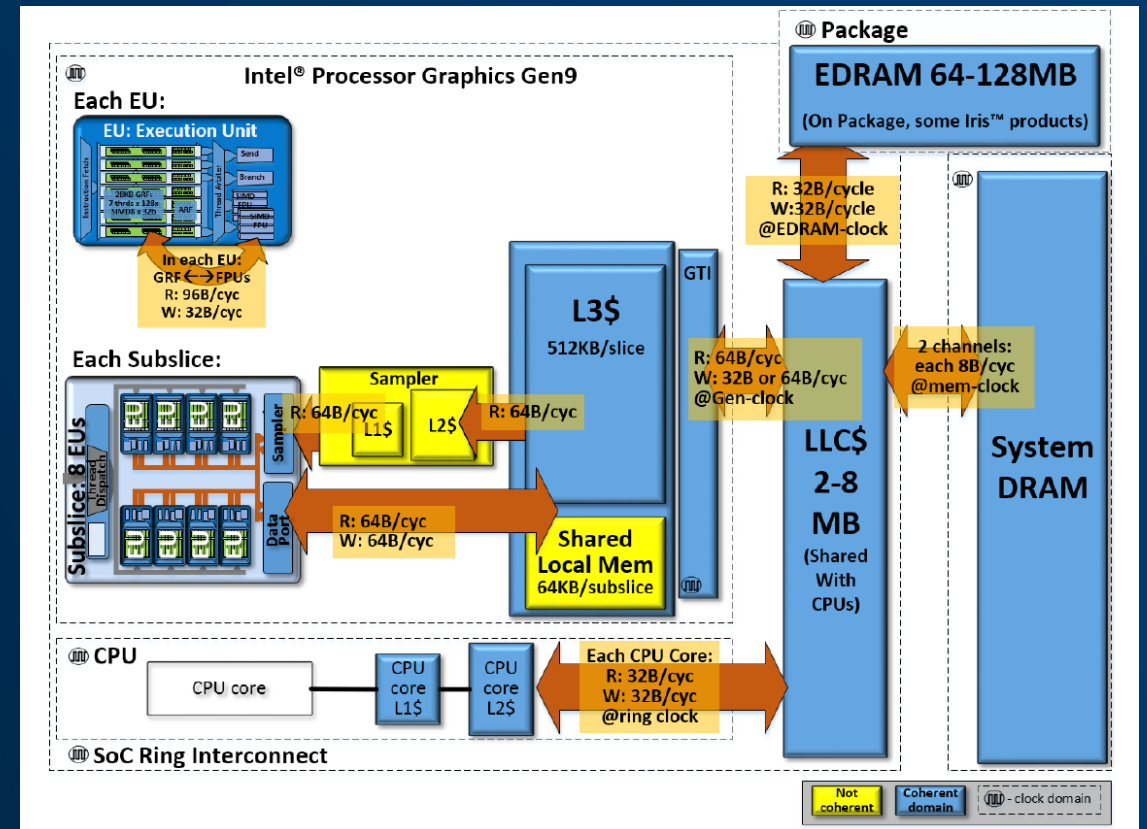
Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.



INTEL® GEN9 MEMORY HIERARCHY

- Intel® Graphics Compute Architecture uses the same DRAM with the CPU
- Level-3 (L3) data cache: slice-shared asset
- Shared Local Memory (SLM): a dedicated structure within the L3 that supports the work-group local memory address space
- Graphics Technology Interface (GTI): a dedicated interface unit connects the entire architecture interfaces to the rest of the SoC components
- The rest of SoC memory hierarchy includes the large Last-Level Cache (LLC, which is shared between CPU and GPU), possibly embedded DRAM and finally the system DRAM

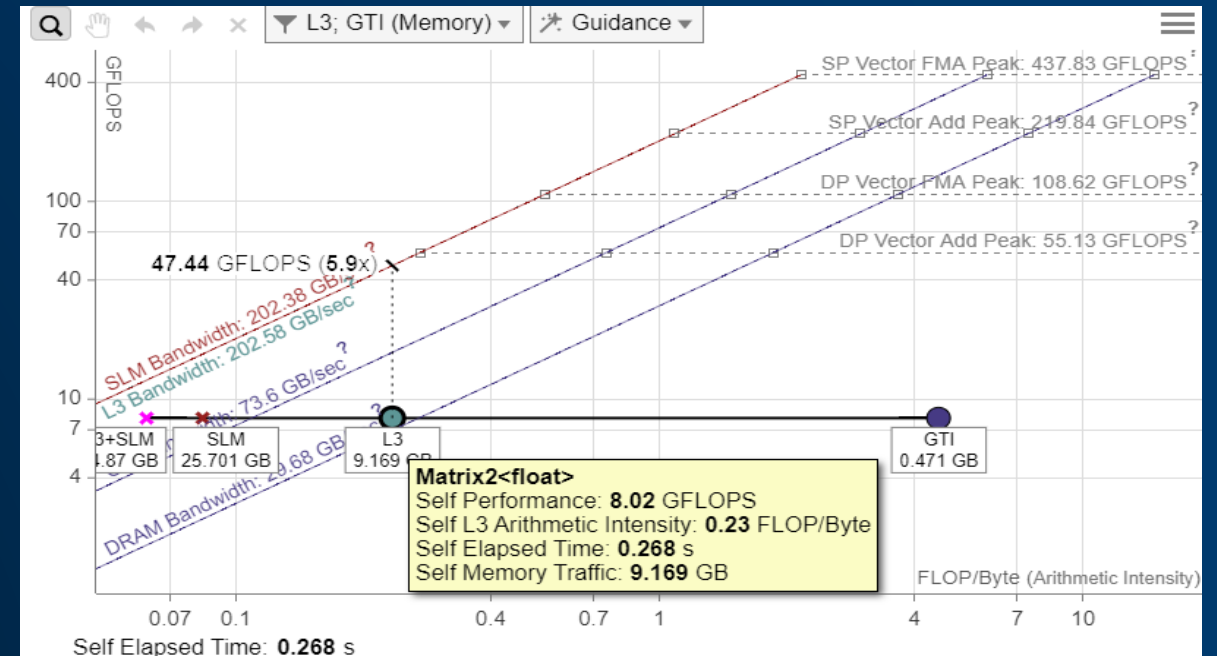


A view of the SoC chip level memory hierarchy and its theoretical peak bandwidths for the compute architecture of Intel processor graphics gen9

FIND EFFECTIVE OPTIMIZATION STRATEGIES

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



HOW TO RUN?

The Roofline model on GPU is a technical preview feature and is not available by default. To enable it:

```
export ADVIXE_EXPERIMENTAL=gpu-profiling
```

To run the GPU Roofline analysis in the Intel® Advisor CLI:

Run the Survey analysis with the **--enable-gpu-profiling** option:

```
advixe-cl -collect=survey --enable-gpu-profiling --project-dir=<my_project_directory> --search-dir  
src:r=<my_source_directory> -- ./myapp [app_parameters]
```

Run the Trip Counts and FLOP analysis with **--enable-gpu-profiling** option:

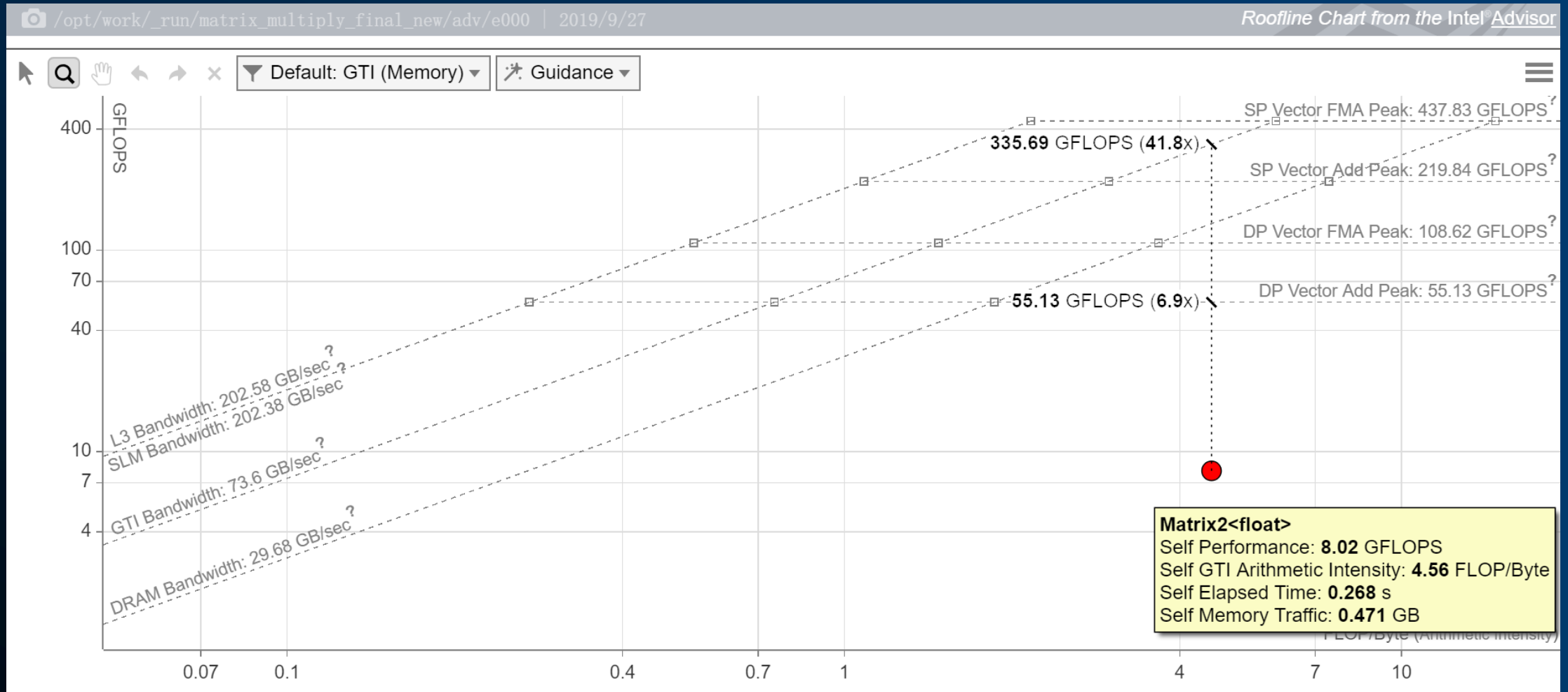
```
advixe-cl -collect=tripcounts --stacks --flop --enable-gpu-profiling --project-  
dir=<my_project_directory> --search-dir src:r=<my_source_directory> -- ./myapp [app_parameters]
```

Generate a GPU Roofline report:

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

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

ROOFLINE ANALYSIS ON INTEL® GPU



LEGAL NOTICES & DISCLAIMERS

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer. No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit <http://www.intel.com/performance>.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

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

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon, Movidius and others 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.

© 2018 Intel Corporation.

BACKUP

[Optimization Notice](#)

Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.



THE MECHANISMS BEHIND 2/2

We minimize the total time spent in this loop hierarchy by varying offload strategies U (offload/non-offload, #threads for each component $loop_i$ of loopnest)

$$\text{Objective function : } T_{all} = \min_{U=\{uf_1, uf_2, \dots\}} (\sum_i T_i + t_{data\ transfer} + t_{invoke} + T_{cpu})$$

Reject loopnests for which
 $T(x86) / T_{all}(x86+\"X\") < 1.0$

$$T_i = \max \begin{cases} T_i^{Comp_only} () \\ T_i^{M_k_only} (M_i^k) = \frac{M_i^k}{BW_k} \end{cases}$$

This is effectively “balance”
(throughput) model

Under algorithmic constraints (Dependencies and TripCount/Granularity)

