POLYTECHNIQUE MONTRÉAL

TECHNOLOGICAL UNIVERSITY

AMD ROCm GPU profiling in Trace Compass

Arnaud Fiorini with Pr. Michel Dagenais

May 8th, 2020

Polytechnique Montreal

DORSAL Laboratory

Agenda

I. Introduction

   1. GPU Development

   2. Optimization Problems

II. Tracing and profiling of CPU-GPU systems

   1. ROC Platform

   2. Tracing GPUs

   3. Profiling GPUs

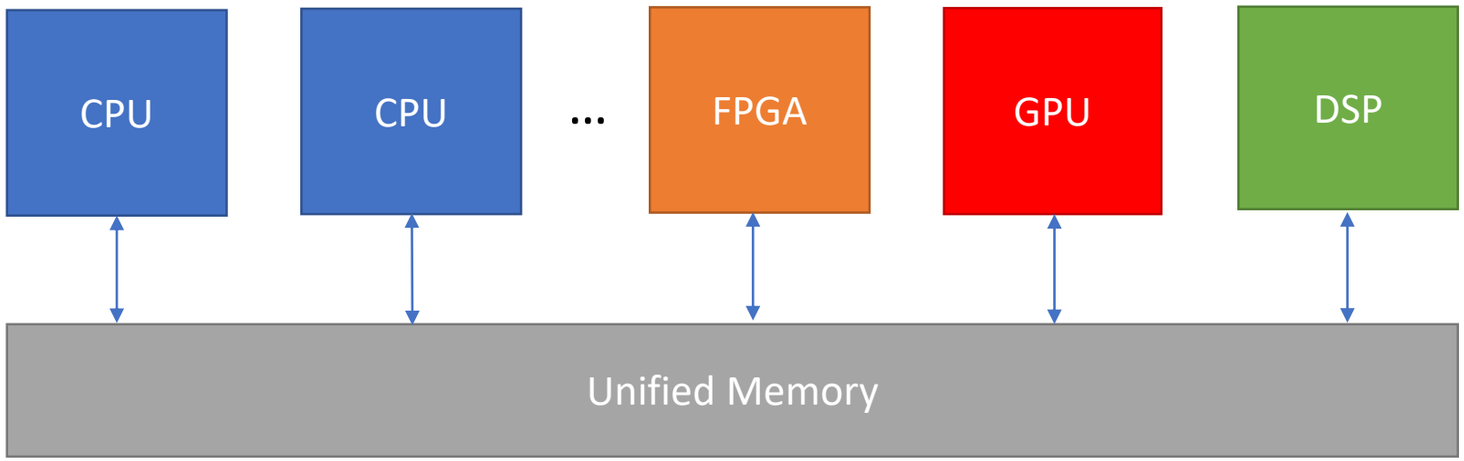
GPU Development - Introduction

- A few definitions :
  - Kernel : A small piece of code executed on the device.

```
__kernel void saxpy(__global float *src, __global float *dst, float factor)
{
  long i = get_global_id(0);
  dst[i] += src[i] * factor;
}
```

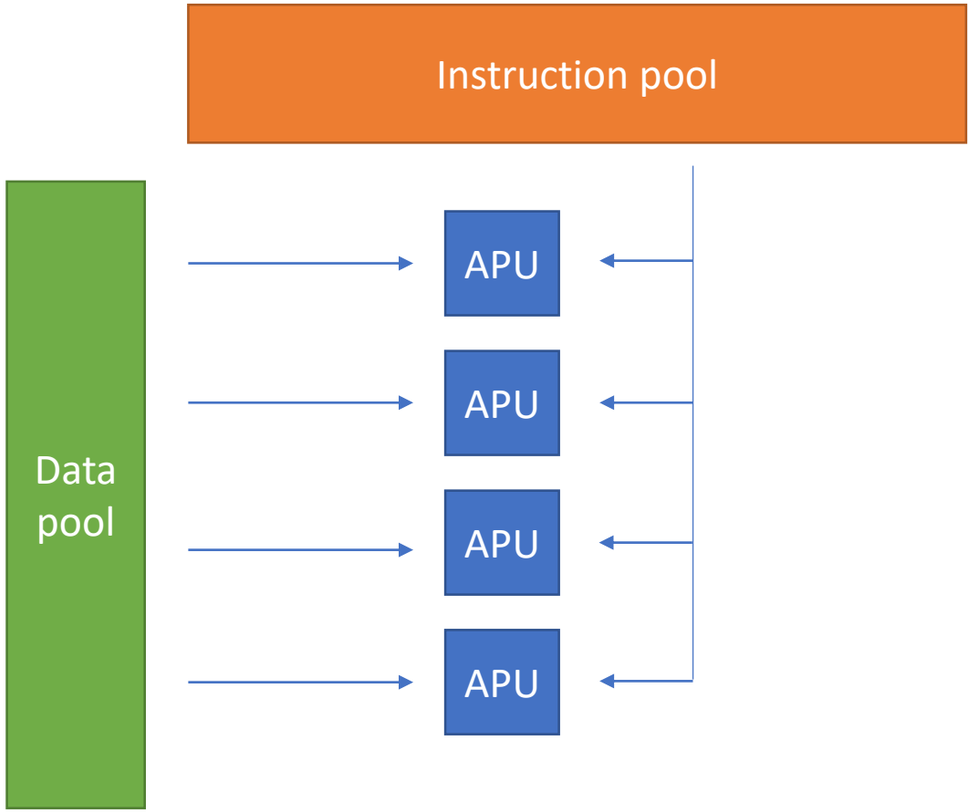
GPU Development - Introduction

- A few definitions :
  - Kernel : A small piece of code executed on the device.
  - Heterogeneous system : system mixing multiple types of processors

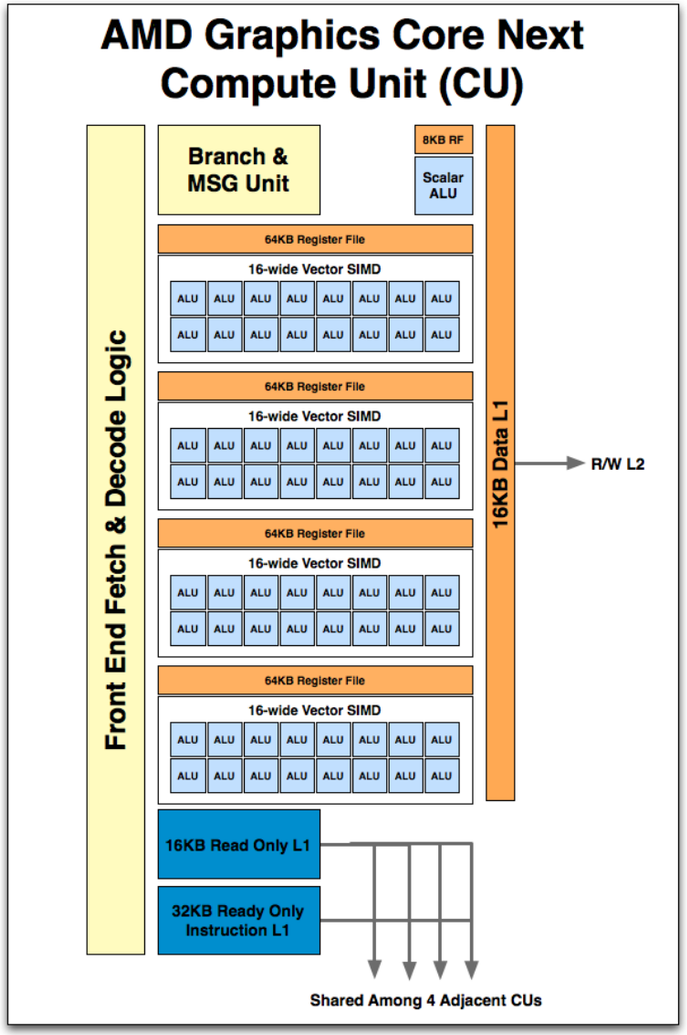


GPU Development - Introduction

- SIMD Architecture :



GPU Development - Introduction



© 2019 AMD Corporation

Optimization Problems - Introduction

- Communication Overhead :
  - Memory synchronisation
  - Interprocessor Communication

- Scheduling and load balancing :
  - Benchmarking
  - Load characteristics of kernels

- Shared Cache :
  - Cache misses, thrashing

Optimization Problems - Introduction

- Communication Overhead :
  - Memory synchronisation
  - Interprocessor Communication

Tracing

- Scheduling and load balancing :
  - Benchmarking
  - Load characteristics of kernels

- Shared Cache :
  - Cache misses, thrashing

Profiling
(Performance Counters)

Agenda

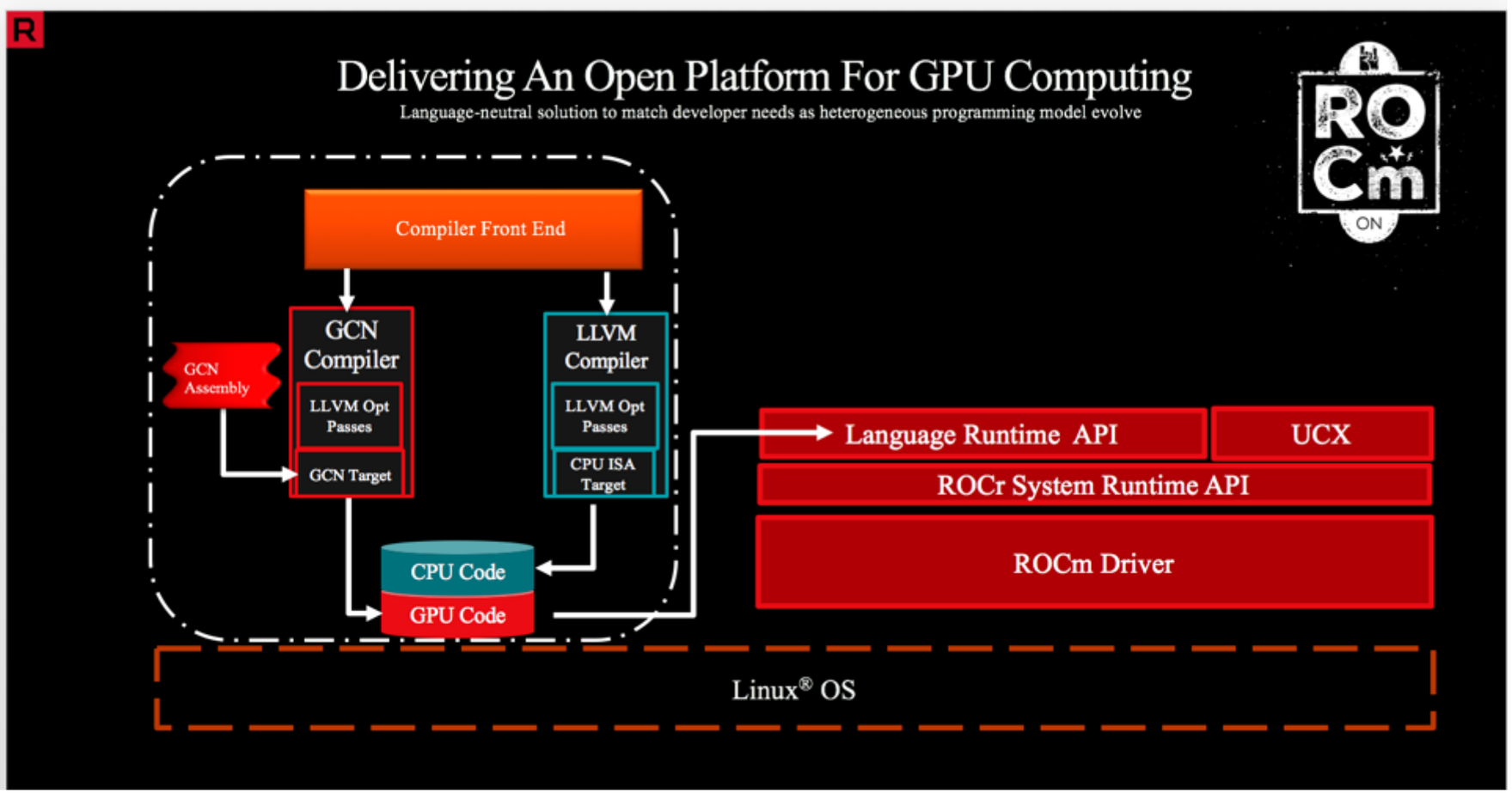
I. Introduction

1. GPU Development
2. Optimization Problems

II. Tracing and profiling of CPU-GPU systems

1. ROC Platform
2. Tracing GPUs
3. Profiling GPUs

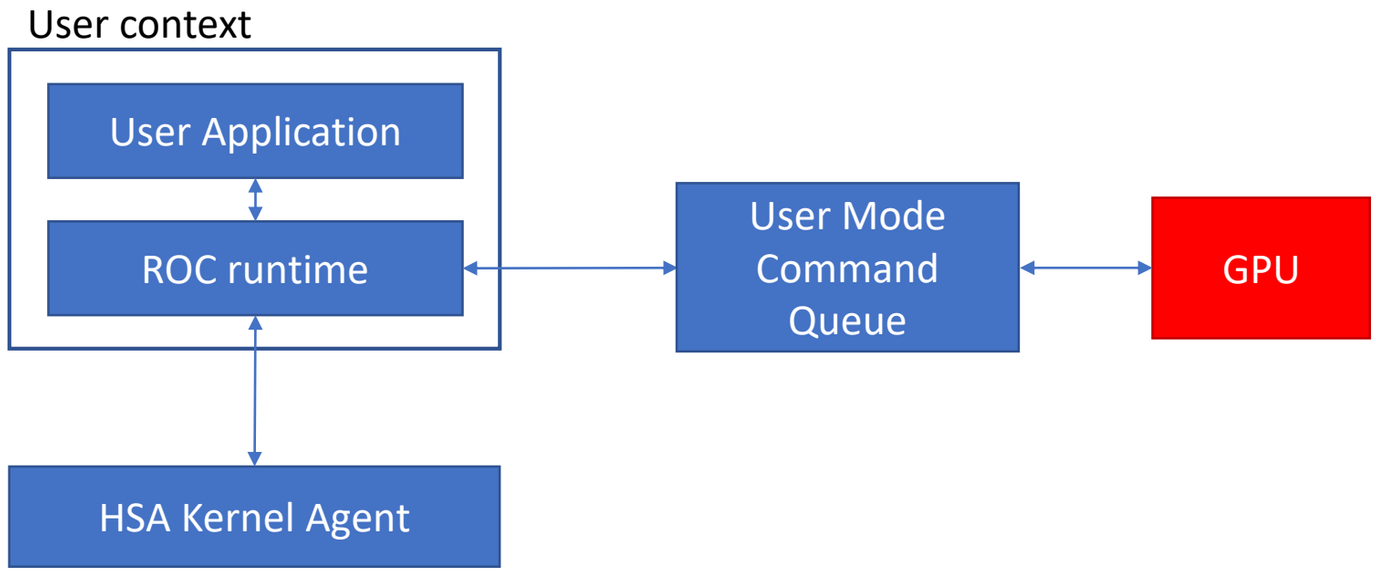
ROC Platform - Tracing and profiling of CPU-GPU systems



© 2019 AMD Corporation https://rocm.github.io/

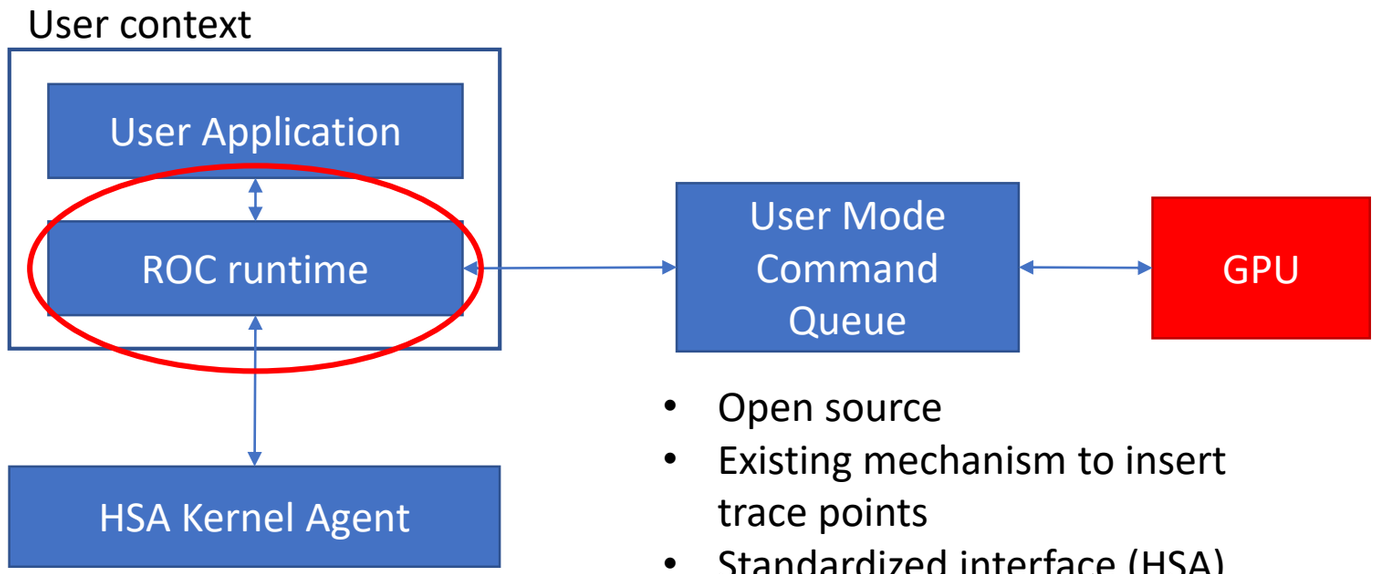
ROC Platform - Tracing and profiling of CPU-GPU systems

ROCm functioning summarized :



ROC Platform - Tracing and profiling of CPU-GPU systems

ROCm functioning summarized :



- Open source
- Existing mechanism to insert trace points
- Standardized interface (HSA)

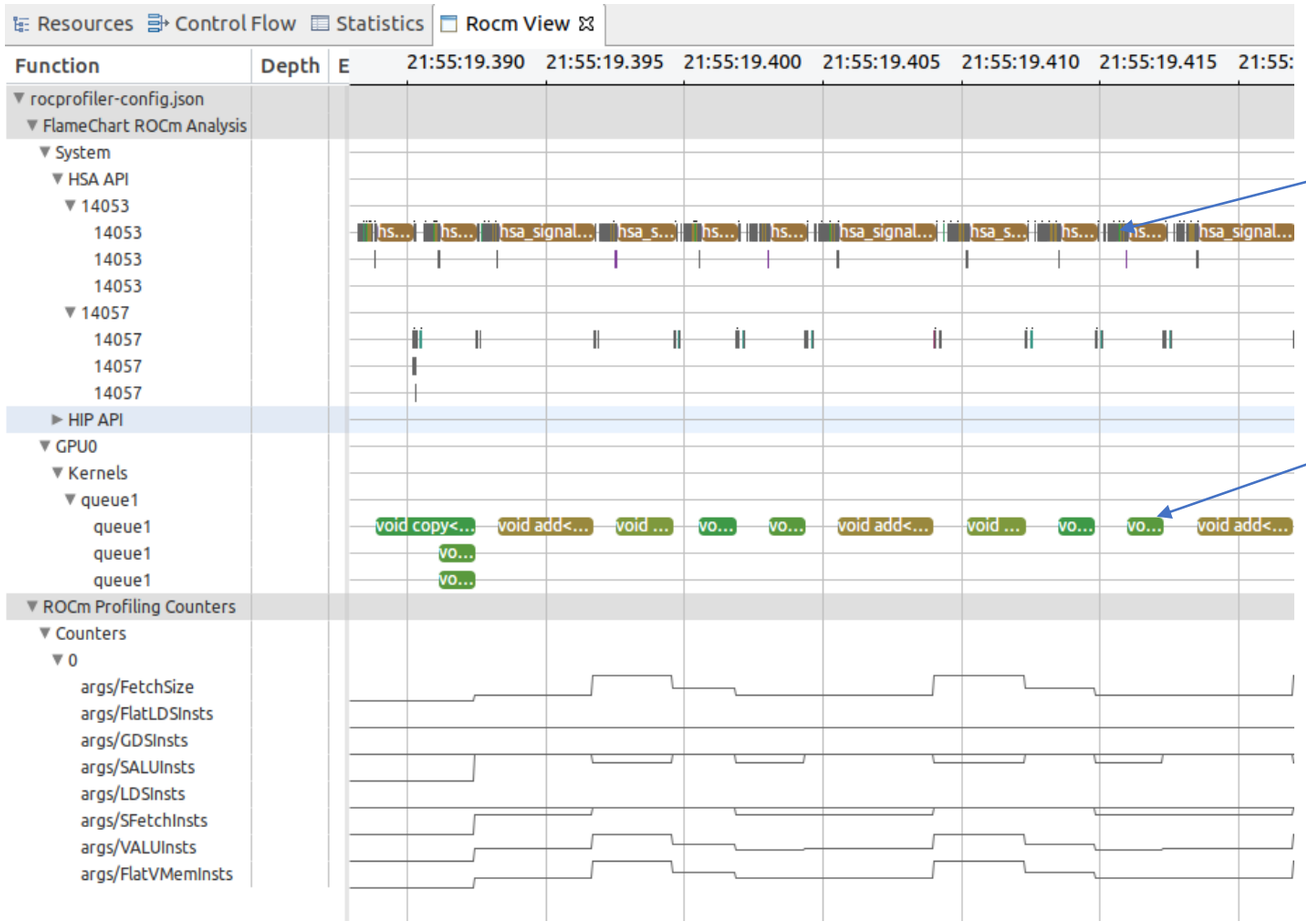
ROC Platform - Tracing and profiling of CPU-GPU systems

- This work has already been done by AMD and is open source :
  https://github.com/ROCm-Developer-Tools/rocprofiler
  https://github.com/ROCm-Developer-Tools/roctracer

- AMD has released a few other libraries and tools thanks to their Radeon Open Compute initiative.

Tracing GPUs - Tracing and profiling of CPU-GPU systems

TraceCompass ROCm plugin

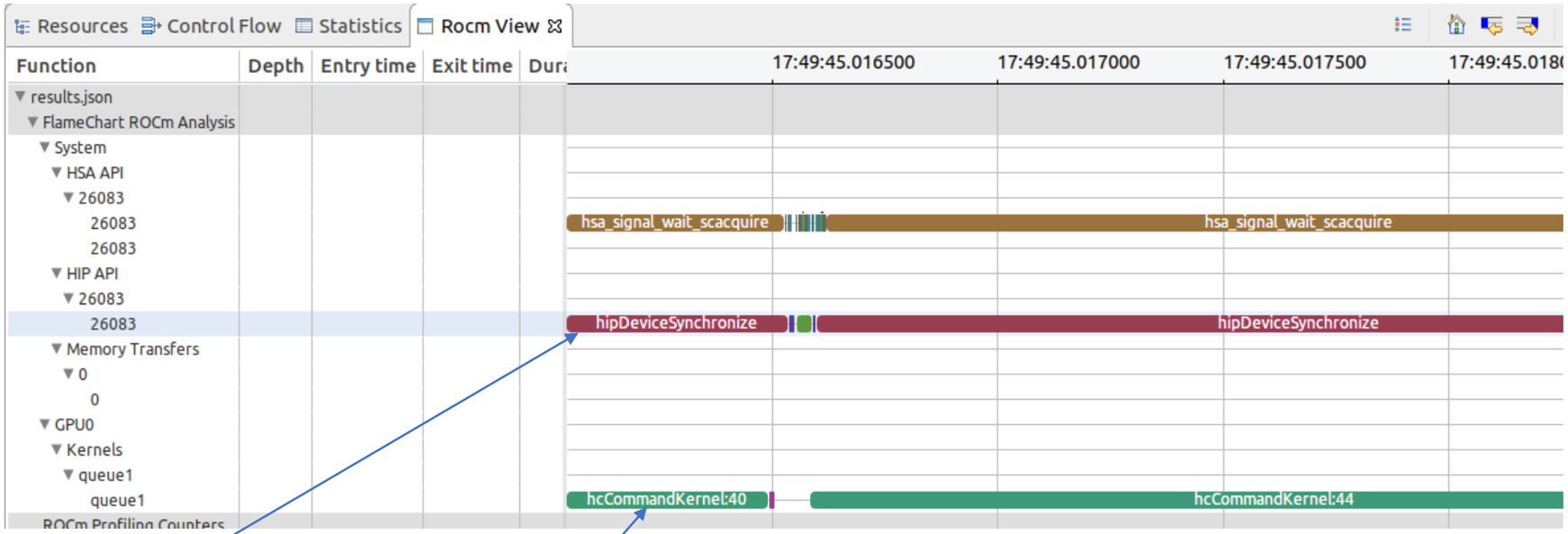


HSA function calls separated by thread

Kernel executions

Tracing GPUs - Tracing and profiling of CPU-GPU systems

TraceCompass ROCm plugin

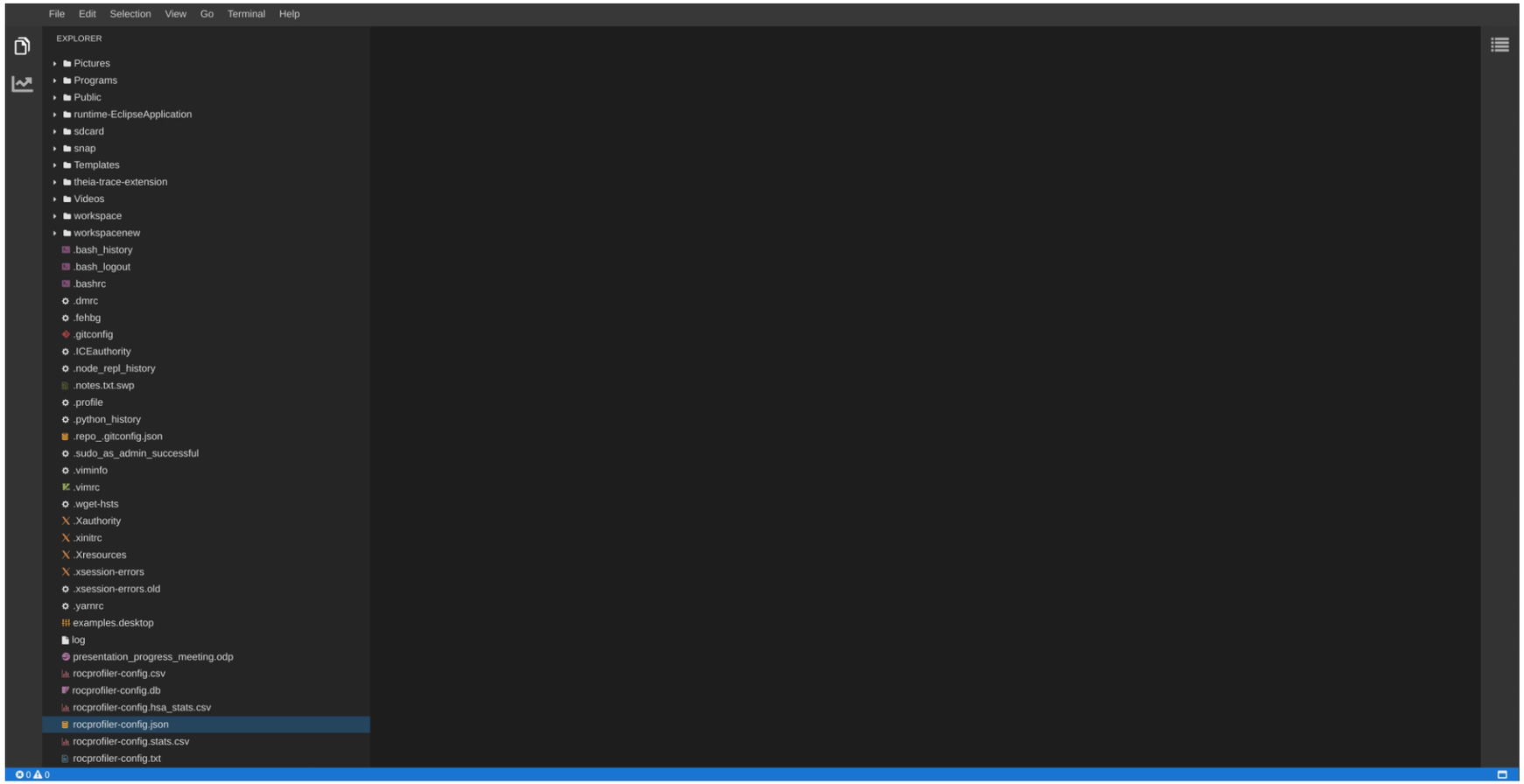


HIP function calls separated by thread

Kernel executions

Tracing GPUs - Tracing and profiling of CPU-GPU systems

TraceCompass ROCm plugin running on Theia front-end



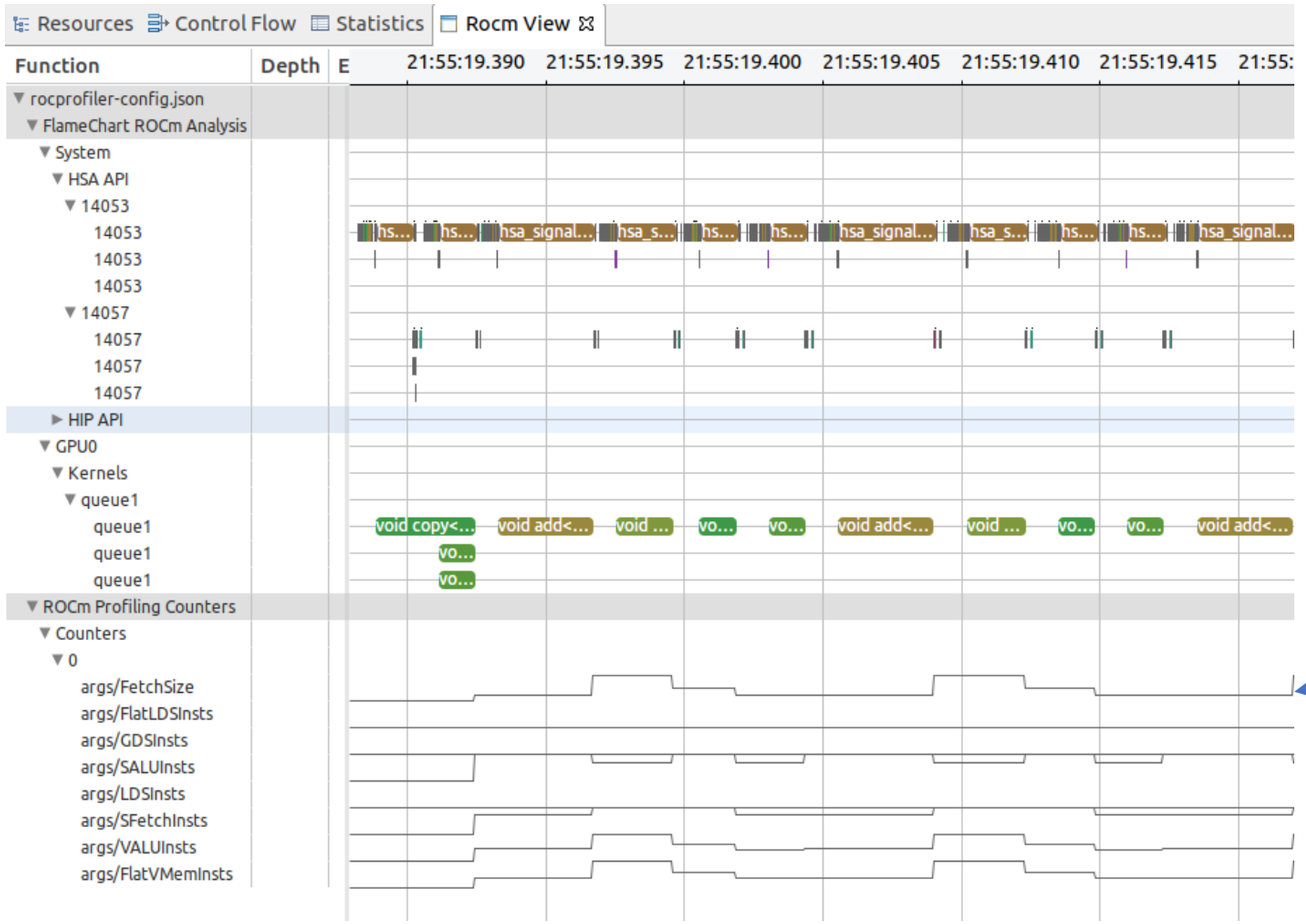
Tracing GPUs - Tracing and profiling of CPU-GPU systems

Analyzing this tracing data further, future work includes :

- Critical path analysis of CPU-GPU execution
- Determining whether the program performance is limited by the GPU or the CPU
- Extracting statistics to use in profiling analysis

Profiling GPUs - Tracing and profiling of CPU-GPU systems

TraceCompass ROCm plugin

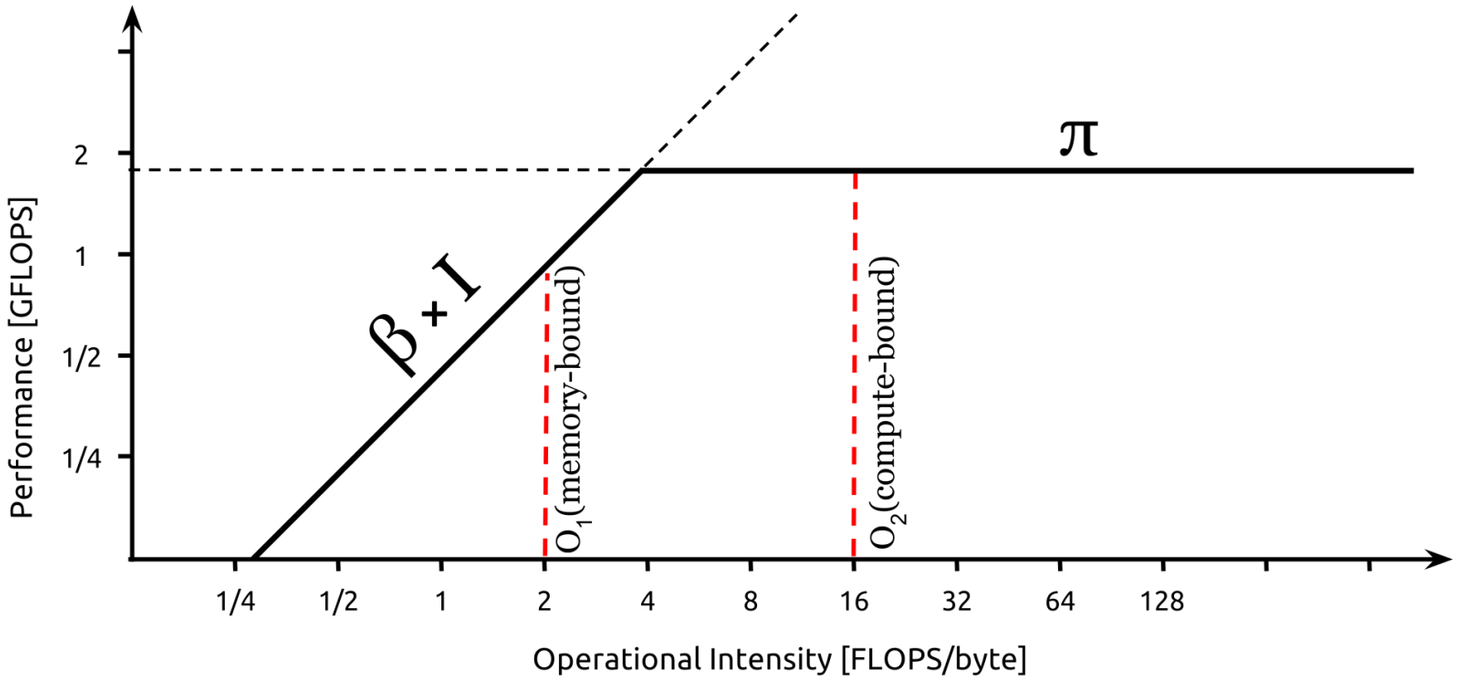


Performance counters

Profiling GPUs - Tracing and profiling of CPU-GPU systems

Using tracing and profiling data, future work includes :

- Roofline model

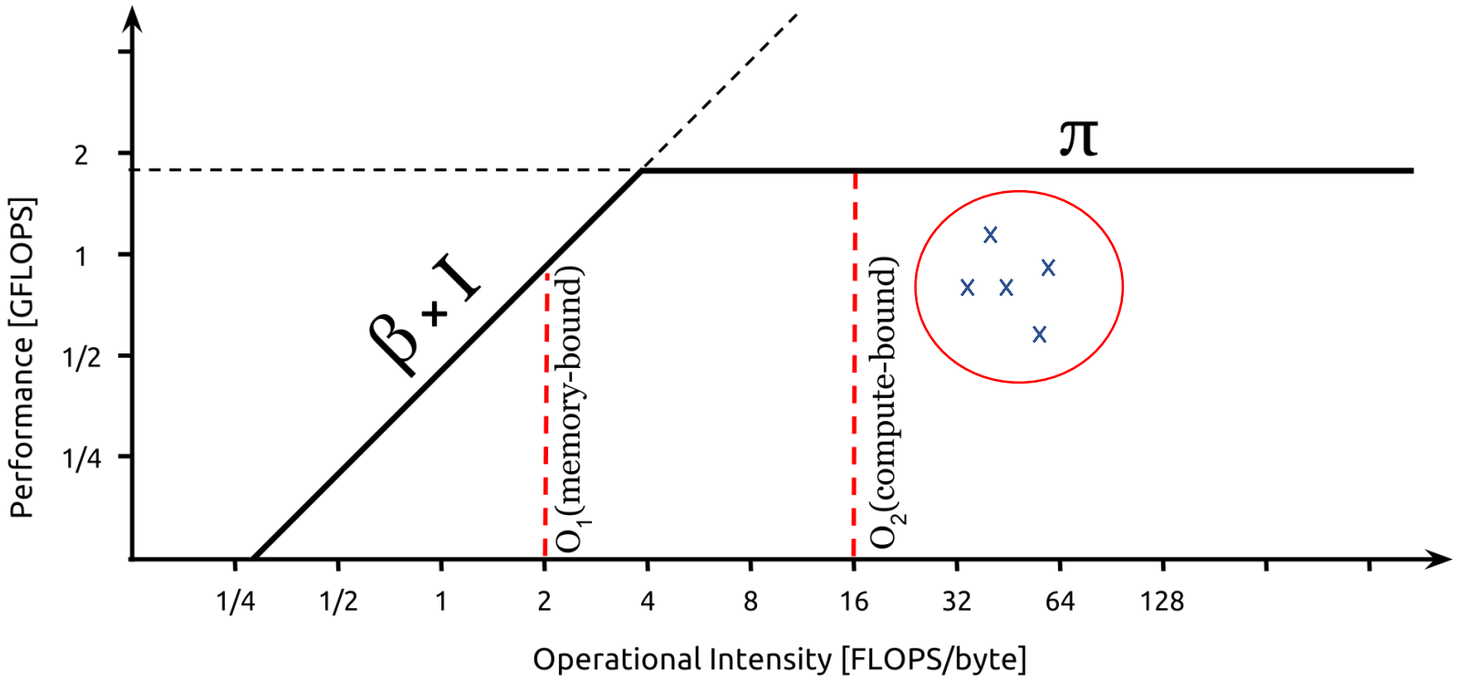


β : peak bandwidth
I : arithmetic intensity
π : peak performance

Profiling GPUs - Tracing and profiling of CPU-GPU systems

Using tracing and profiling data, future work includes :

- Roofline model

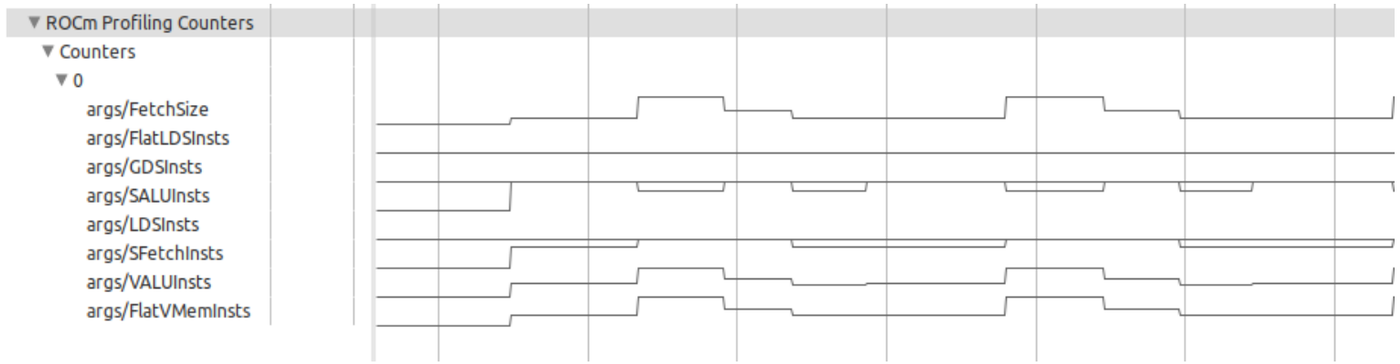


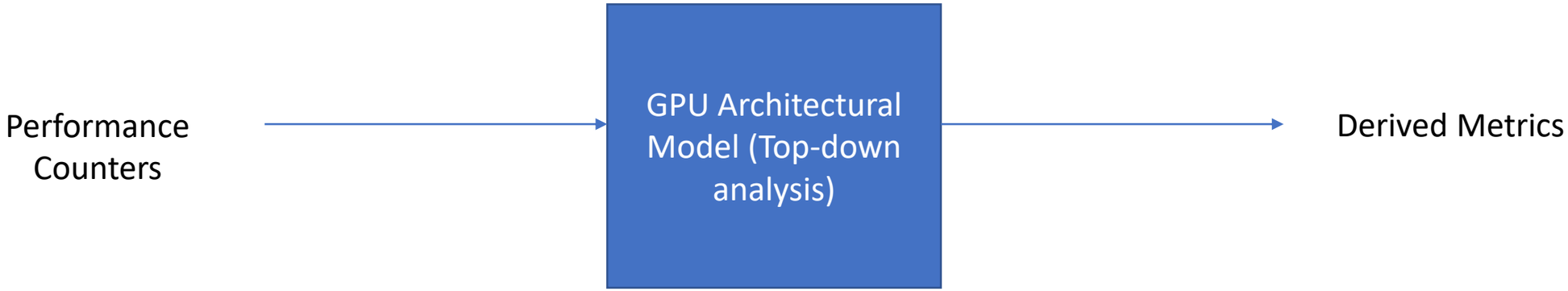
β : peak bandwidth
I : arithmetic intensity
π : peak performance

Profiling GPUs - Tracing and profiling of CPU-GPU systems

Using tracing and profiling data, future work includes :

- Top-down analysis

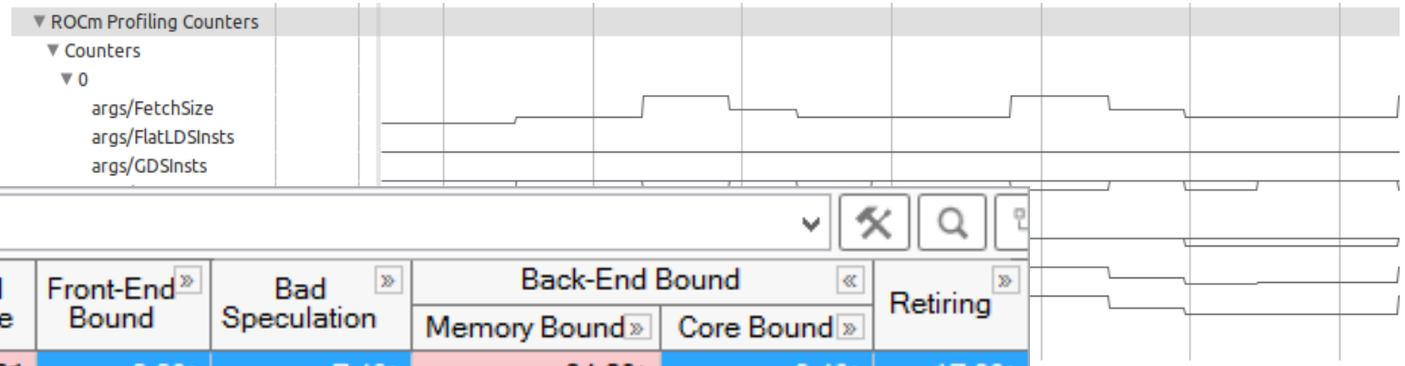




Profiling GPUs - Tracing and profiling of CPU-GPU systems

Using tracing and profiling data, future work includes :

- Top-down analysis



Grouping: Function / Call Stack

Function / Call Stack	Instructions Retired	CPI Rate	Front-End Bound	Bad Speculation	Back-End Bound		Retiring
					Memory Bound	Core Bound	
price_out_impl	62,556,093,834	1.261	2.2%	7.4%	64.2%	8.4%	17.8%
refresh_potential	17,836,026,754	3.589	3.0%	8.1%	73.2%	9.6%	6.1%
primal_bea_mpp	38,108,057,162	1.393	5.6%	24.3%	34.4%	21.0%	14.7%
update_tree	4,092,006,138	3.373	7.2%	11.5%	62.3%	11.8%	7.2%
sort_basket	12,246,018,369	1.037	20.7%	50.4%	3.8%	4.6%	20.6%
primal_iminus	5,324,007,986	2.148	7.1%	6.7%	55.0%	20.5%	10.8%
primal_net_simple:	266,000,399	2.466	17.4%	43.4%	13.1%	13.1%	13.0%

Performance Counters

rics

analysis)

Thank you for listening !

Questions ?

References

- https://github.com/RadeonOpenCompute/ROCm
- https://rocm-documentation.readthedocs.io/en/latest/
- http://www.hsafoundation.com/
- HSA Runtime Programmer’s Reference Manual, Version 1.2
- HSA Programmer's Reference Manual, Version 1.2
- HSA Platform System Architecture Specification, Version 1.2
- https://github.com/ucb-bar/opencl-kernels/blob/master/saxpy/kernel.cl
- https://medium.com/@smallfishbigsea/basic-concepts-in-gpu-computing-3388710e9239
- https://www.techpowerup.com/gpu-specs/docs/amd-gcn1-architecture.pdf
- https://software.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/methodologies/top-down-microarchitecture-analysis-method.html