



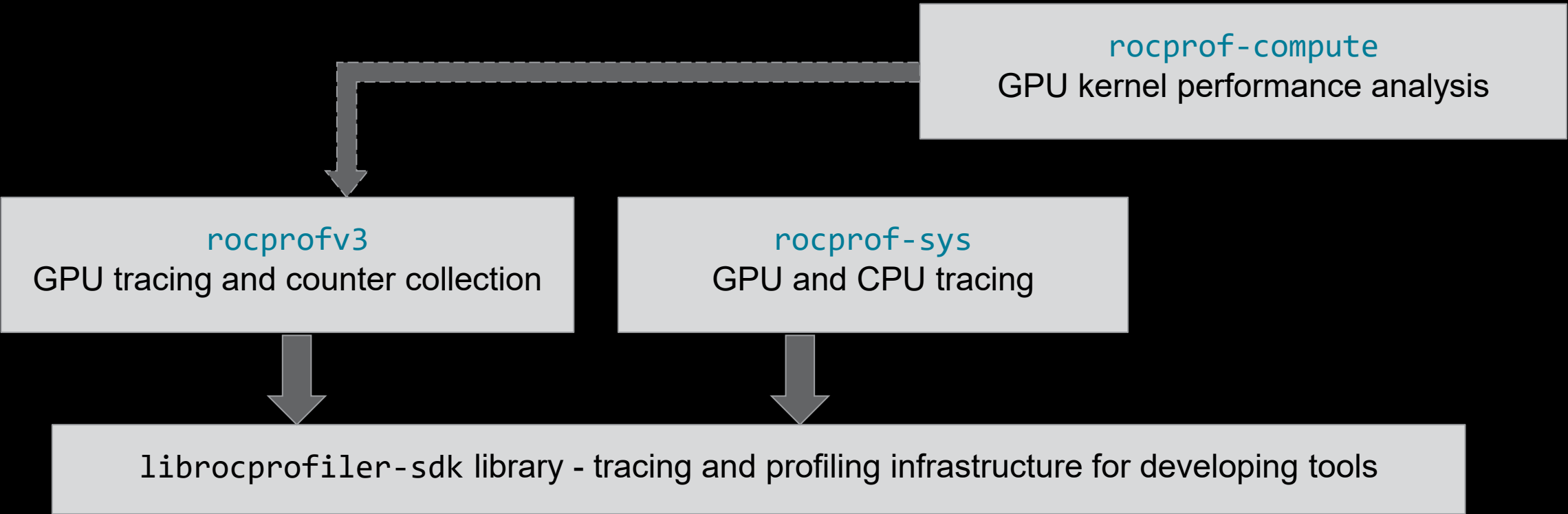
ROCprofv3

Basic Profiling and Timelines

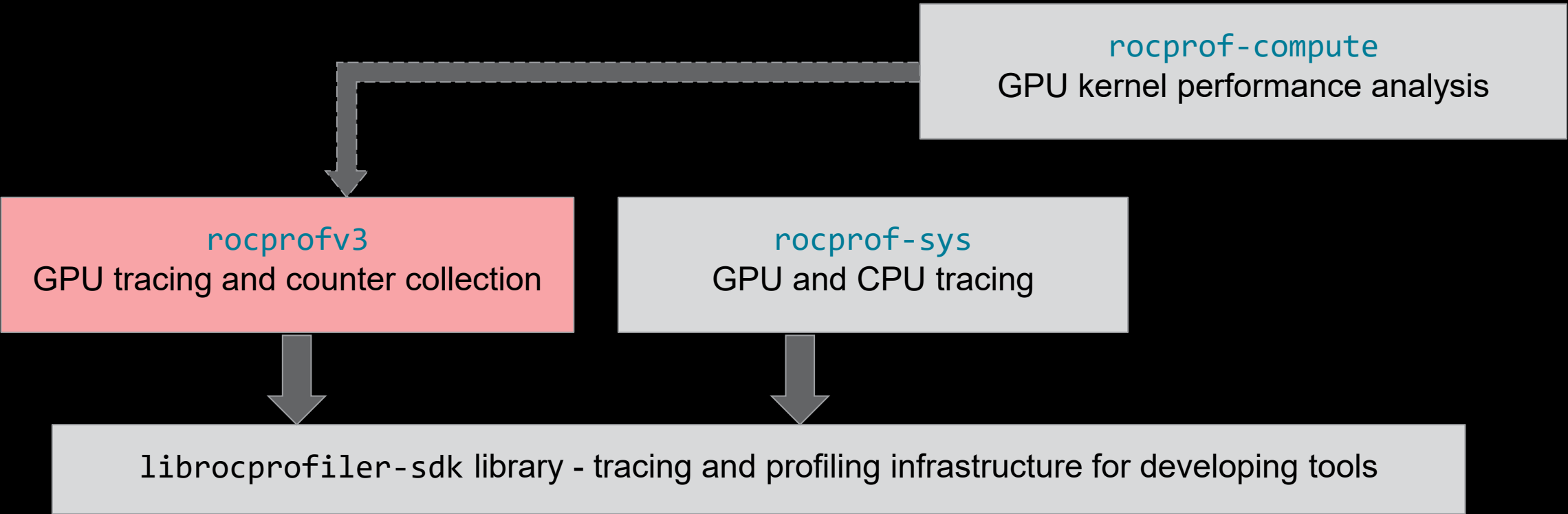
Presenter: Bob Robey
Oct 21-23, 2025
AMD @ Tsukuba University

AMD 
together we advance_

AMD has three GPU profiling tools



AMD has three GPU profiling tools



What is rocprofv3?

- Performance analysis tool for ROCm based applications (single or multi process)
- Main capabilities:
 - GPU Hotspot analysis – identify performance bottlenecks
 - Device activity tracing – visualize HIP, HSA , GPU kernels, and data transfers in a GUI
 - Performance counter collection – analyze kernel performance further
- Supports Python™ and OpenMP® offload profiling
- Documented at: <https://rocm.docs.amd.com/projects/rocprofiler-sdk/en/latest/how-to/using-rocprofv3.html>

Running rocprofv3 with single and multiple processes

- rocprofv3 requires double-hyphen (--) before the application to be executed:
 - `$ rocprofv3 [<rocprofv3-options> ...] -- <application> [<application-args> ...]`
 - `$ rocprofv3 --runtime-trace -- ./myapp -n 1`
- For MPI applications or if using Slurm, place rocprofv3 inside the job launcher:
 - `$ mpirun -np 4 rocprofv3 --runtime-trace -- ./mympiapp`
 - Output files will be generated for each MPI process automatically

NOTE: rocprofv3 can run with MPI but it does not trace MPI calls

rocprofv3: Collecting application traces

- rocprofv3 can collect a variety of trace event types and generate timelines
- Can combine options in a run

Trace Event	rocprofv3 Trace Mode
Collect HIP, Kernels, Memory Copy, Marker API	--runtime-trace
GPU Kernels	--kernel-trace
HIP API call	--hip-trace
Host <-> Device Memory copies	--hip-trace or --memory-copy-trace
User code markers	--marker-trace
CPU HSA Calls	--hsa-trace
Kokkos calls	--kokkos-trace
Collect HSA, HIP, Kernels, Memory Copy, Marker API	--sys-trace

- Pftrace output format (`--output-format pftrace`) currently recommended

rocprofv3: Collecting GPU hotspots

```
$ rocprofv3 --stats --kernel-trace --truncate-kernels --summary -- <app with arguments>
```

Combine `--stats`
with `--kernel-trace`

Truncate kernel
names, remove args

Print hotspot summary
on the console

ROCPROFV3 SUMMARY:

NAME	DOMAIN	CALLS	DURATION (nsec)	AVERAGE (nsec)	PERCENT (INC)	MIN (nsec)	MAX (nsec)	STDDEV
JacobiIterationKernel	KERNEL_DISPATCH	1000	527366199	5.274e+05	40.714858	512643	548644	5.592e+03
NormKernel1	KERNEL_DISPATCH	1001	410842848	4.104e+05	31.718772	399042	418723	2.666e+03
LocalLaplacianKernel	KERNEL_DISPATCH	1000	329779380	3.298e+05	25.460336	323842	350722	2.095e+03
HaloLaplacianKernel	KERNEL_DISPATCH	1000	15328759	1.533e+04	1.183444	14400	16480	3.597e+02
__amd_rocclr_copyBuffer	KERNEL_DISPATCH	1001	7706765	7.699e+03	0.594994	7040	10080	3.180e+02
NormKernel2	KERNEL_DISPATCH	1001	4236348	4.232e+03	0.327064	4000	5120	1.169e+02
__amd_rocclr_fillBufferAligned	KERNEL_DISPATCH	1	6880	6.880e+03	0.000531	6880	6880	0.000e+00

- GPU kernel hotspots are also output in XXXXX_kernel_stats.csv

rocprofv3: Collecting hardware counters

- rocprofv3 can collect a number of hardware counters and derived counters

- \$ rocprofv3 -L

- To collect, specify counters in an input file:

- \$ rocprofv3 -i rocprof_counters.txt -- <app with args>

- \$ cat rocprof_counters.txt

```
pmc: VALUUtilization VALUBusy FetchSize WriteSize MemUnitStalled
```

```
pmc: GPU_UTIL CU_OCCUPANCY MeanOccupancyPerCU MeanOccupancyPerActiveCU
```

One pass per pmc line

Hardware limits on how many counters can be collected in one pass

- Or specify on command line:

- \$ rocprofv3 --pmc VALUUtilization FetchSize -- <app with args>

- In the <pid>_counter_collection.csv output file:

```
2,2,8,2,43371,43371,16384,18,"NormKernel1",128,1024,0,12,4,32,"FetchSize",131073.625000,116935428165636,116935428564998
```

```
2,2,8,2,43371,43371,16384,18,"NormKernel1",128,1024,0,12,4,32,"VALUUtilization",99.962183,116935428165636,116935428564998
```

Dispatch ID

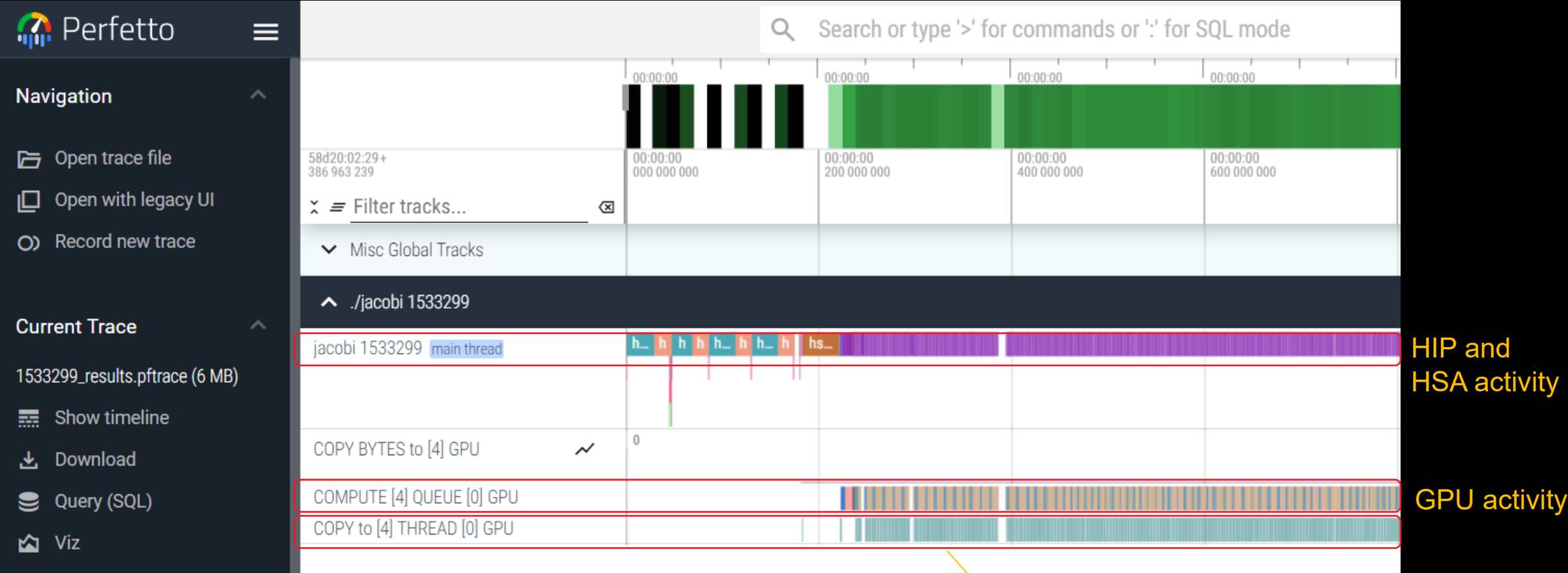
One line per metric per kernel invocation

Value of metric

Visualizing application traces with Perfetto

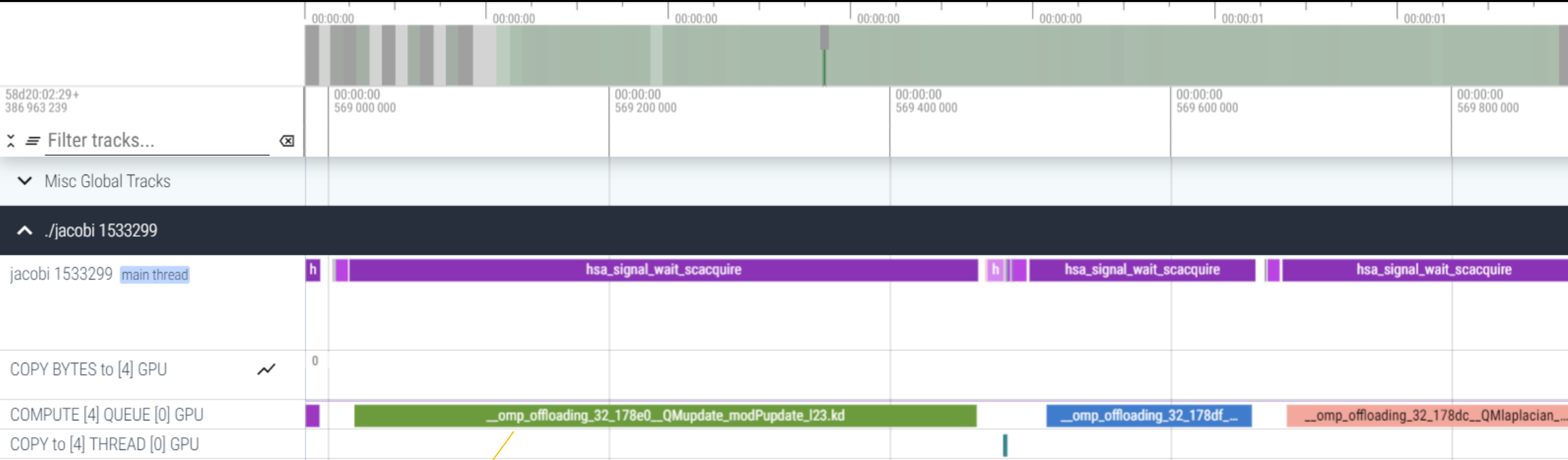
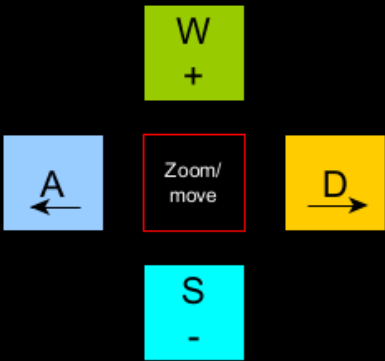
Traces from Fortran +
OpenMP® offload code

- `$ rocprofv3 --sys-trace --output-format pftrace -- <app with arguments>`
- This outputs a Pftrace file that can be visualized using Perfetto (<https://ui.perfetto.dev/>) in Chrome



Perfetto: Navigating through traces

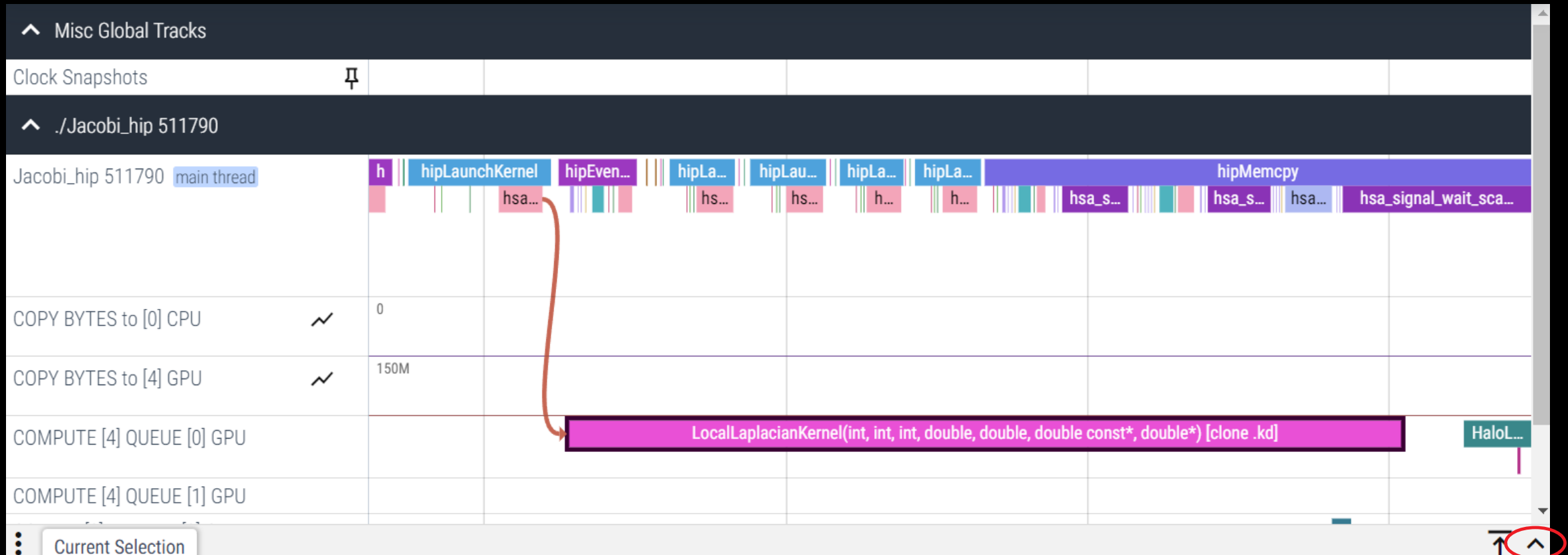
- Zoom in to see individual events
- Navigate trace using WASD keys



OpenMP® offload visualization shown here

Perfetto: Kernel information and flow events

- Zoom and select a kernel, you can see the link to the HIP call launching the kernel
- Try to open the information for the kernel (button at bottom right)



Workgroup size and grid size

rocpv3: Collecting application traces with roctx markers and regions

- Annotate code with roctx regions:

```
#include <rocpv3-sdk-roctx/roctx.h>
...
roctxRangePush("reduce_for_c");
// some code
roctxRangePop();
...
```

- Annotate code with roctx markers:

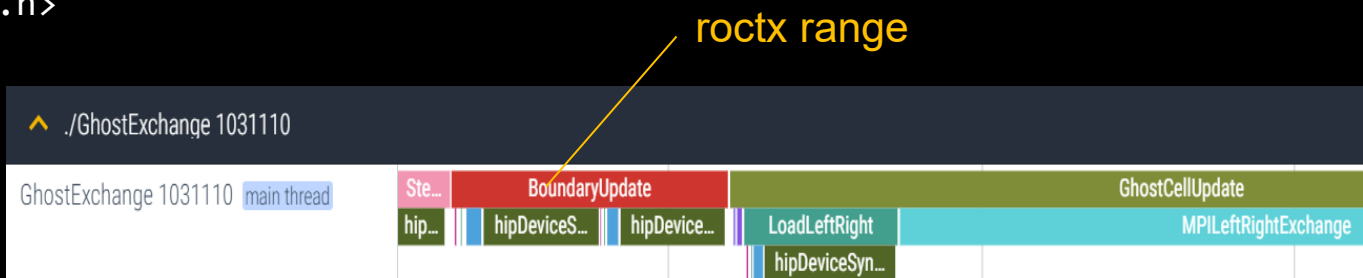
```
...
roctxMark("start of some code");
// some_code
roctxMark("end of some code");
...
```

- Link with rocpv3-sdk-roctx library:

```
-L${ROCM_PATH}/lib -lrocpv3-sdk-roctx
```

- Profile with --hip-trace --marker-trace options:

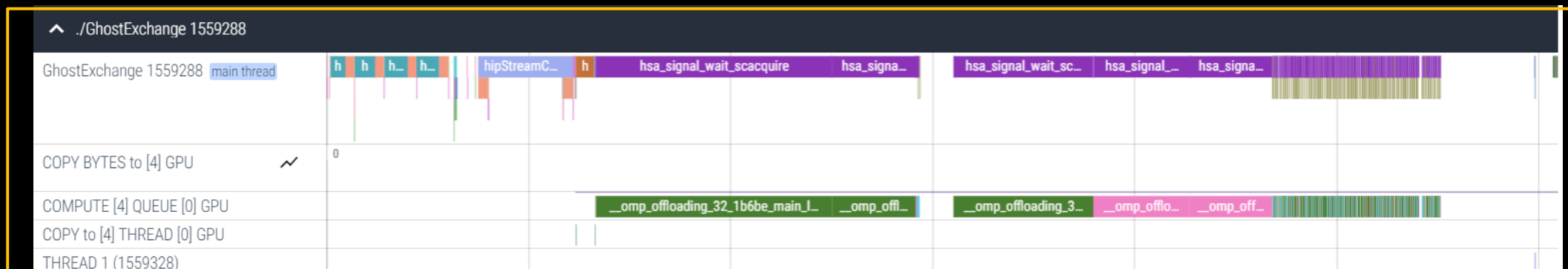
```
$ rocpv3 --hip-trace --marker-trace -- <app with arguments>
```



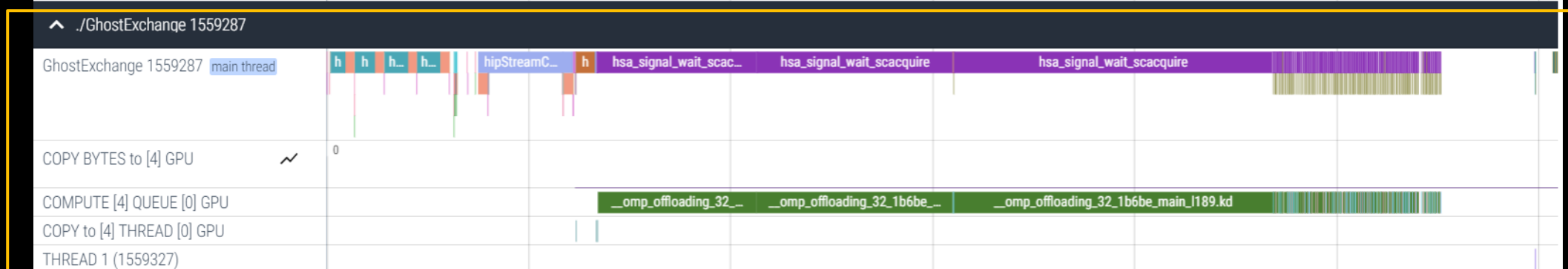
Merge traces from multiple MPI ranks

- `$ cat XXX*_results.pftrace > merged.pftrace`
- Then visualize merged.pftrace in Perfetto

Rank 0



Rank 1

Ranks
2 and 3

Profiling overhead

- rocprofv3 is designed for minimal profiling overhead
- The percentage of overhead depends on:
 - The profiling options used (e.g., tracing is faster than hardware counter collection)
 - The number of counters you collect (multiple passes for multiple pmc lines)
 - Whether roctx regions/markers are collected (can result in large traces too)
- The more data collected, the more the overhead of profiling
- To minimize collected data, profile only the small (interesting) region of the execution
 - Use `--collection-period (START_DELAY_TIME):(COLLECTION_TIME):(REPEAT)` added in ROCm 6.4
 - Possibly combined with: `--collection-period-unit {hour,min,sec,msec,usec,nsec}` (default sec)
 - `$ rocprofv3 --collection-period 3:3:1 --sys-trace -- <app with args>`

Summary

- **rocprofv3**: CLI tool to quickly analyze GPU activity on AMD GPUs
 - Hotspot analysis – identify performance bottlenecks
 - Application tracing – visualize HIP, HSA and device activity in a GUI
 - Performance counter collection – analyze kernel performance further

Hands-on exercises

- Located in our HPC Training Examples repo:
<https://github.com/amd/HPCTrainingExamples>
- A table of contents for the READMEs if available at the top-level README in the repo
- rocprofv3 exercises: [Rocprofv3/HIP/README.md](#) or [Rocprofv3/OpenMP/README.md](#)
- Log into the AAC node and clone the repo:

```
ssh <username>@aac6.amd.com -p 7000 -i <path_to_ssh_key>  
git clone https://github.com/amd/HPCTrainingExamples.git  
module load rocm/6.4.0
```

DISCLAIMERS AND ATTRIBUTIONS

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

THIS INFORMATION IS PROVIDED 'AS IS.' AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

© 2025 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD Arrow logo, Radeon™, Instinct™, EPYC, Infinity Fabric, ROCm™, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

The OpenMP name and the OpenMP logo are registered trademarks of the OpenMP Architecture Review Board
Git and the Git logo are either registered trademarks or trademarks of Software Freedom Conservancy, Inc., corporate home of the Git Project, in the United States and/or other countries

