Overview of HPCToolkit

Performance Tools for GPU-accelerated Supercomputing

John Mellor-Crummey, Rice University
ALCF Hands-on HPC Workshop
Oct. 11, 2023

HPCToolkit Project Team

- **Rice University**
  - HPCToolkit PI: Prof. John Mellor-Crummey
  - Research staff: Laksono Adhianto, Mark Krentel, Wil Phan, Matt Barnett
  - Contractor: Marty Itzkowitz
  - Students: Jonathon Anderson, Dragana Grbic, Vladimir Indjic, Yumeng Liu
- **University of Wisconsin – Madison**
  - Dyninst PI: Prof. Barton Miller

Outline

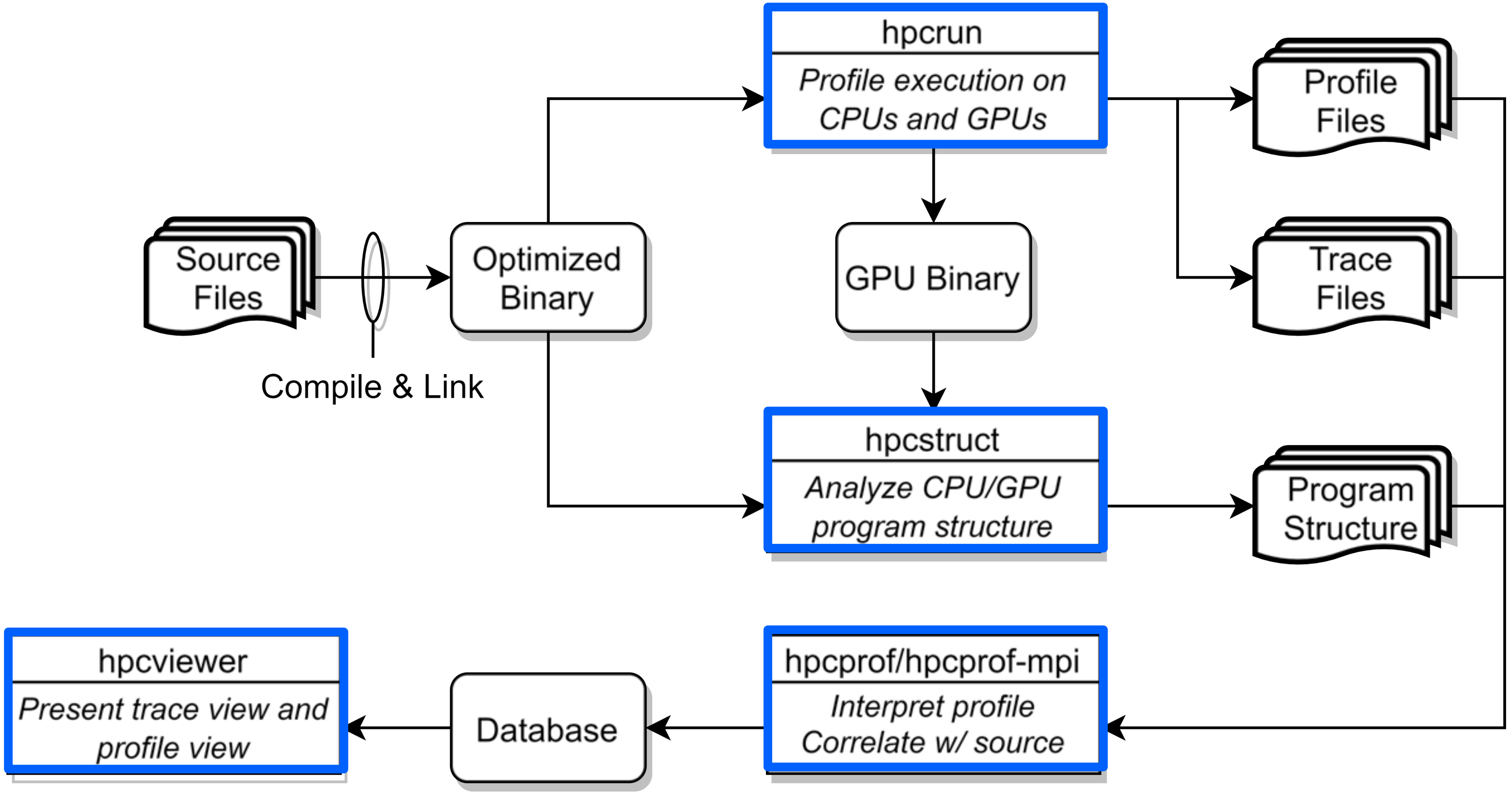
- **HPCToolkit performance tools for CPU and GPU-accelerated applications**
  - Overview of HPCToolkit components and their workflow
  - HPCToolkit's graphical user interfaces
- **Analyzing the performance of GPU-accelerated codes with HPCToolkit**
  - GAMESS
  - Quicksilver
  - LAMMPS at Exascale
- **Status**
- **Resources**
- **Hands-on directions**

Rice University’s HPCToolkit Performance Tools

Measure and analyze performance of CPU and GPU-accelerated applications

- **Easy: profile unmodified application binaries**
- **Fast: low-overhead measurement**
- **Informative: understand where an application spends its time and why**
  - call path profiles associate metrics with application source code contexts
  - optional hierarchical traces to understand execution dynamics
- **Broad audience**
  - application developers
  - framework developers
  - runtime and tool developers
- **Supported platforms**
  - CPU: x86_64, Power, ARM
  - GPU: NVIDIA, AMD, Intel

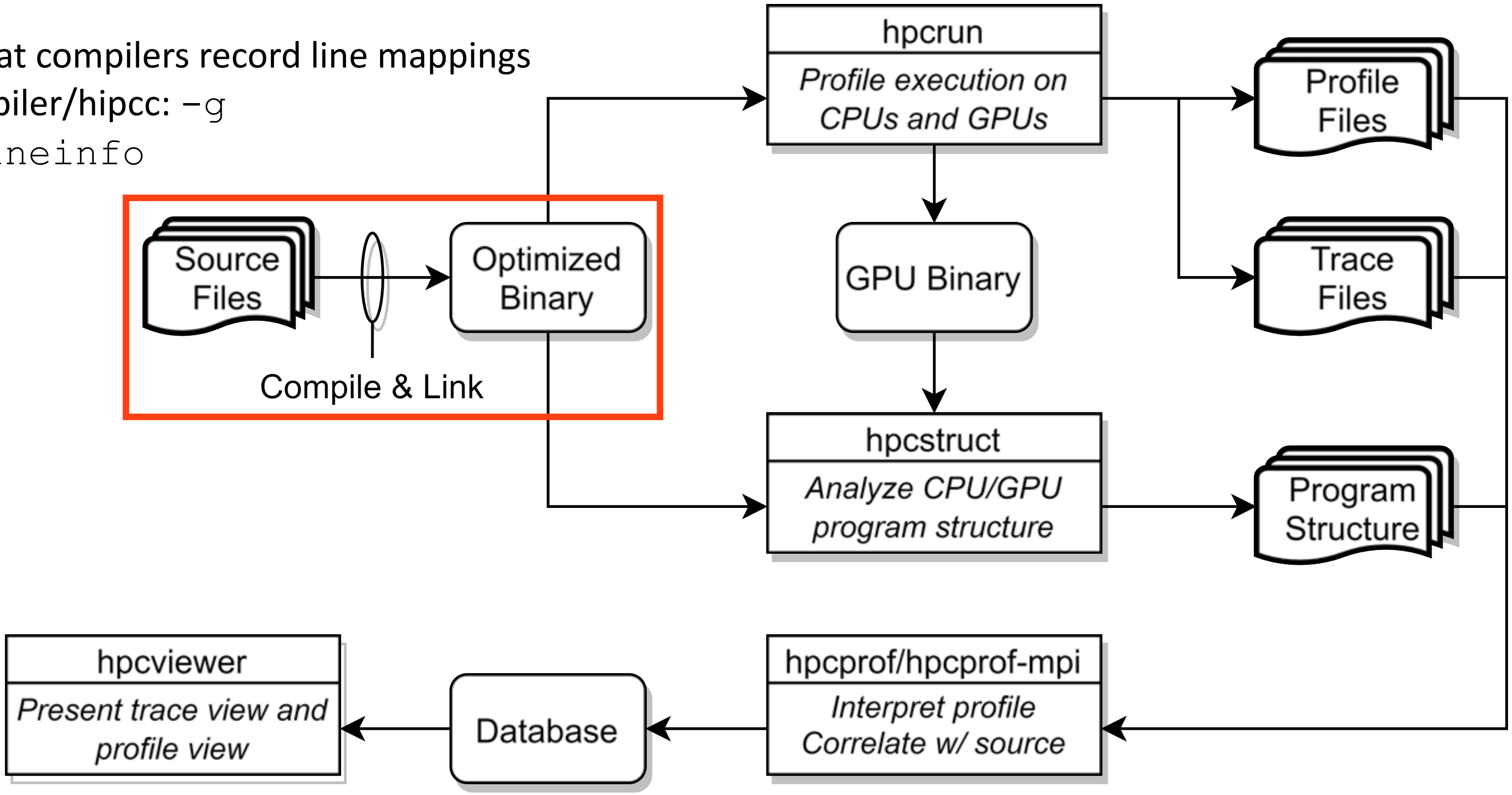
HPCToolkit's Workflow for GPU-accelerated Applications



HPCToolkit's Workflow for GPU-accelerated Applications

Step 1:

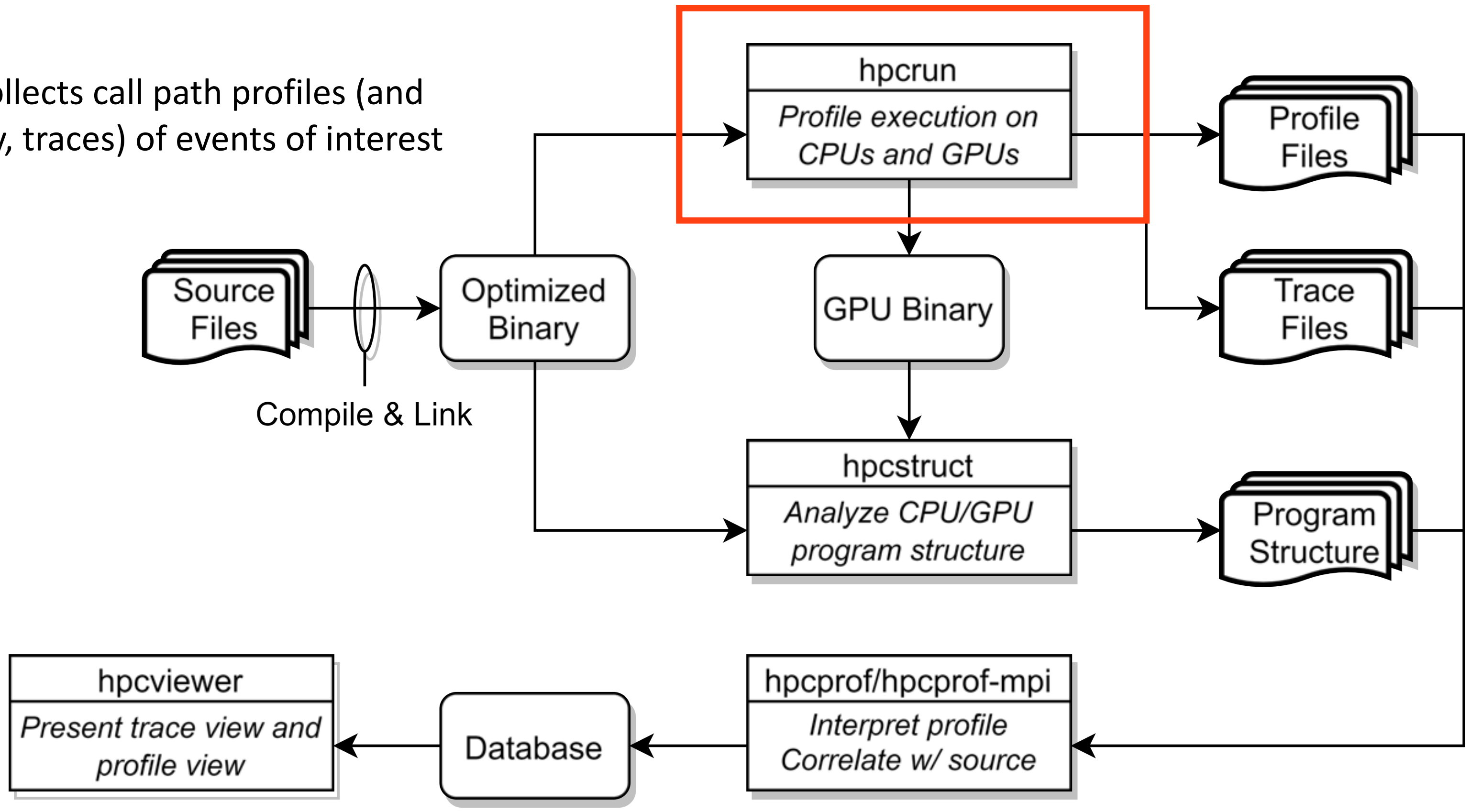
- Ensure that compilers record line mappings
- host compiler/hipcc: `-g`
- nvcc: `-lineinfo`



HPCToolkit's Workflow for GPU-accelerated Applications

Step 2:

- *hpcrun* collects call path profiles (and optionally, traces) of events of interest

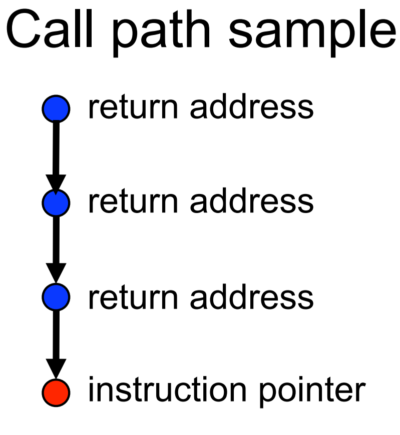


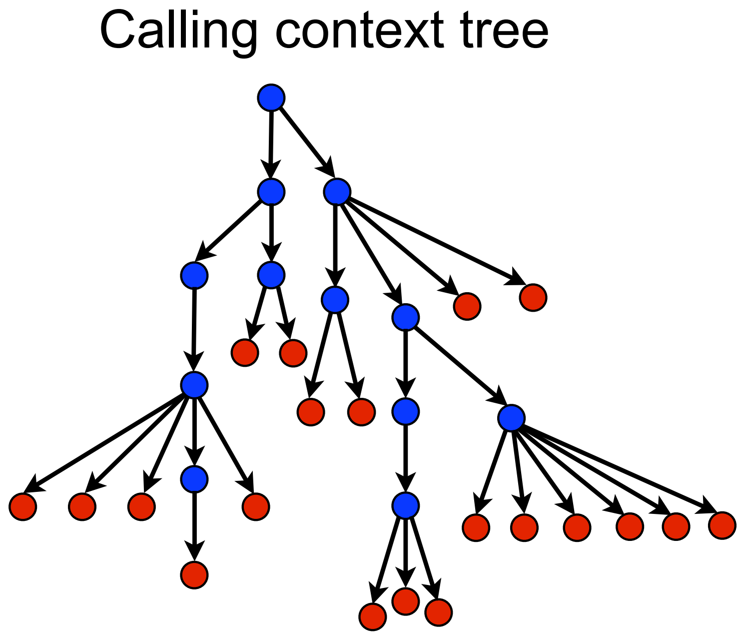
Measurement of CPU and GPU-accelerated Applications

- **Sampling using Linux timers and hardware counter overflows on the CPU**
- **Callbacks when GPU operations are launched and (sometimes) completed**
- **Event stream for GPU operations**
- **Instruction-level measurements**
  - PC Samples (NVIDIA)
  - Binary instrumentation of GPU kernels (Intel)

Call Stack Unwinding to Attribute Costs in Context

- **Unwind when timer or hardware counter overflows**
  - measurement overhead proportional to sampling frequency rather than call frequency
- **Unwind to capture context for events such as GPU kernel launches**





hpcrun: Measure CPU and/or GPU activity

- **GPU profiling**
  - `hpcrun -e gpu=xxx <app> ….`

xxx ∈ {nvidia,amd,opencl,level0}

- **GPU PC sampling (NVIDIA GPU only)**
  - `hpcrun -e gpu=nvidia,pc <app>`
- **CPU and GPU Tracing (in addition to profiling)**
  - `hpcrun -e CPUTIME -e gpu=xxx -t <app>`
- **Use hpcrun with job launchers**
  - `jsrun -n 32 -g 1 -a 1 hpcrun -e gpu=xxx <app>`

Profiles
- a calling context tree per thread
- instruction level measurements

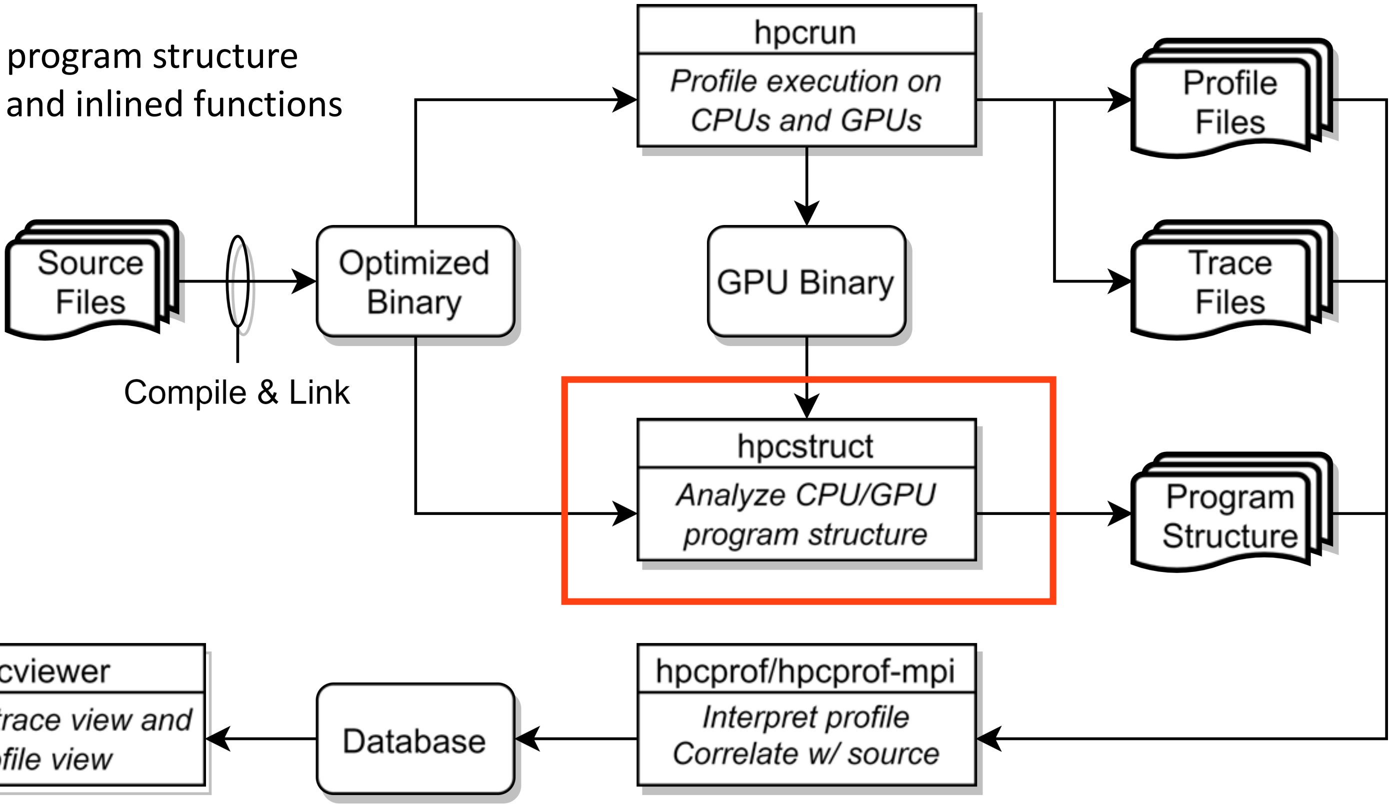
CPU traces
- trace of call stack samples

GPU traces
- trace of call stacks that initiate GPU operations

HPCToolkit's Workflow for GPU-accelerated Applications

Step 3:

- *hpcstruct* recovers program structure about lines, loops, and inlined functions



hpcstruct: Analyze CPU and GPU Binaries Using Multiple Threads

- **Usage**

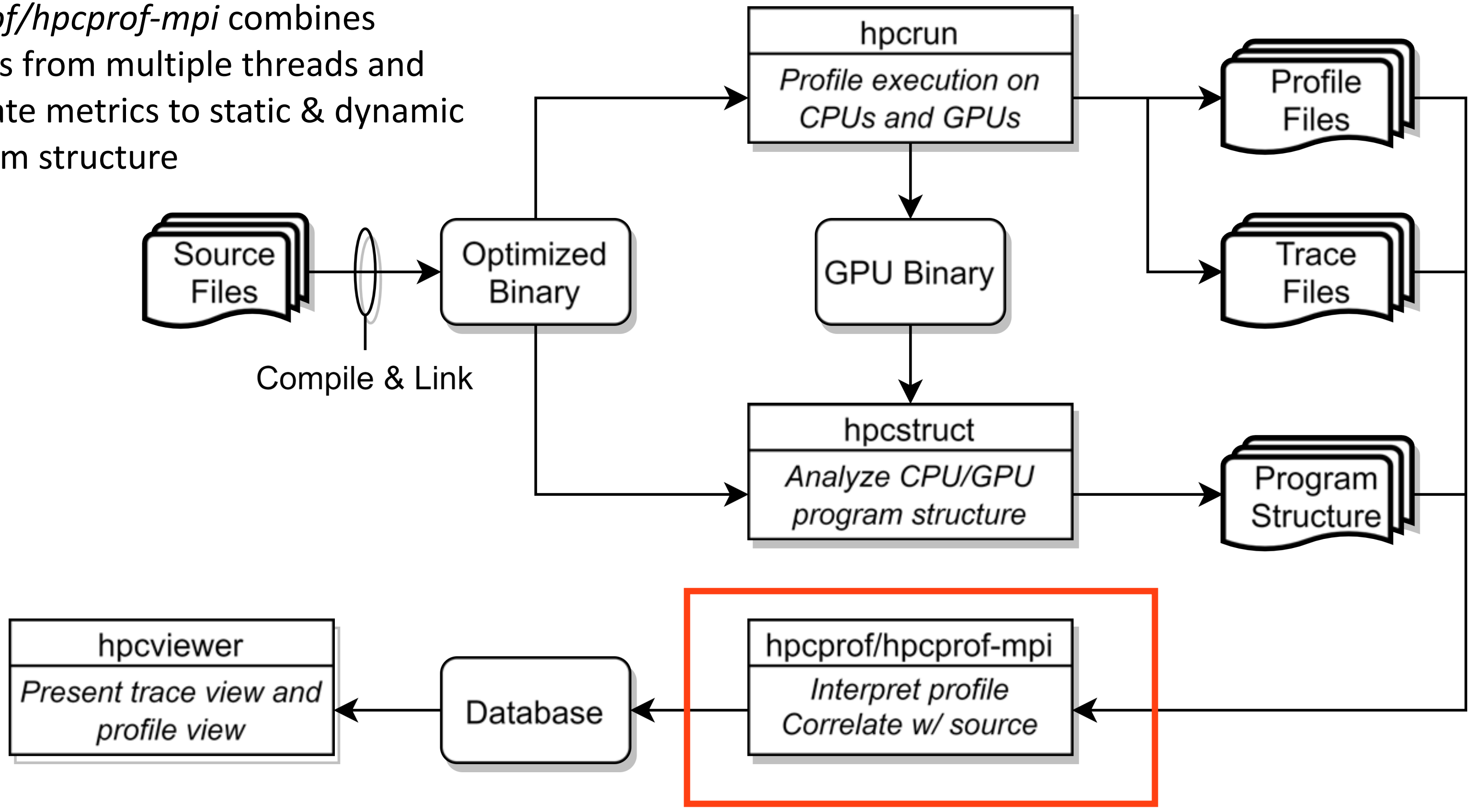
```
hpcstruct [--gpucfg yes] <measurement-directory>
```

- **What it does**
  - Recover program structure information
    - Files, functions, inlined templates or functions, loops, source lines
  - Analyze all CPU and GPU binaries that were measured by HPCToolkit using multithreading
  - Cache binary analysis results for reuse when analyzing other executions

HPCToolkit's Workflow for GPU-accelerated Applications

Step 4:

- *hpcprof/hpcprof-mpi* combines profiles from multiple threads and correlate metrics to static & dynamic program structure



hpcprof/hpcprof-mpi: Associate Measurements with Program Structure

- **Analyze data from modest executions with multithreading**

```
hpcprof <measurement-directory>
```

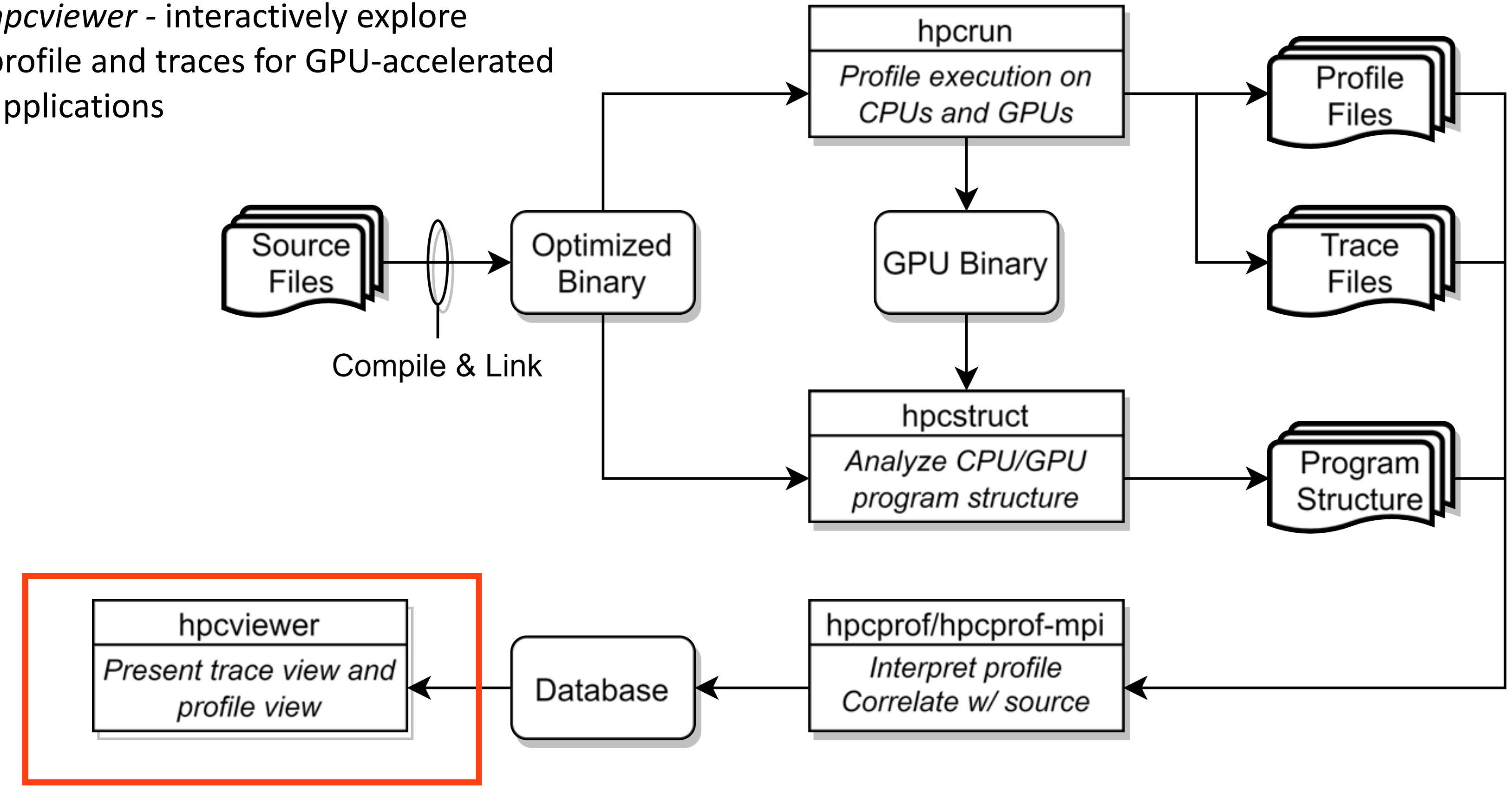
- **Analyze data from large executions with distributed-memory parallelism + multithreading**

```
jsrun -n 2 -a 1 -c 18 -b packed hpcprof-mpi <measurement-directory>
```

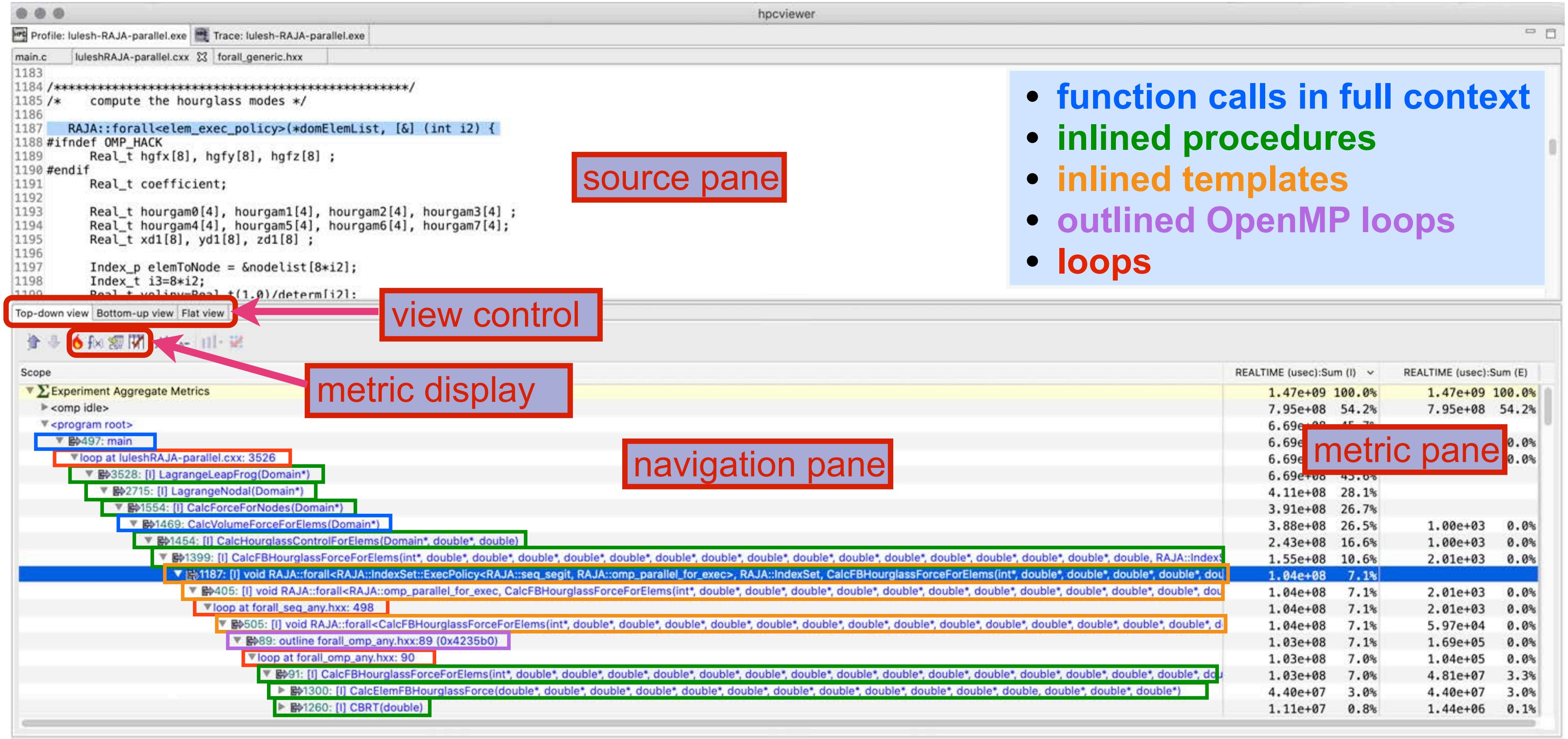
HPCToolkit’s Workflow for GPU-accelerated Applications

Step 4:

- *hpcviewer* - interactively explore profile and traces for GPU-accelerated applications



Code-centric Analysis with hpcviewer

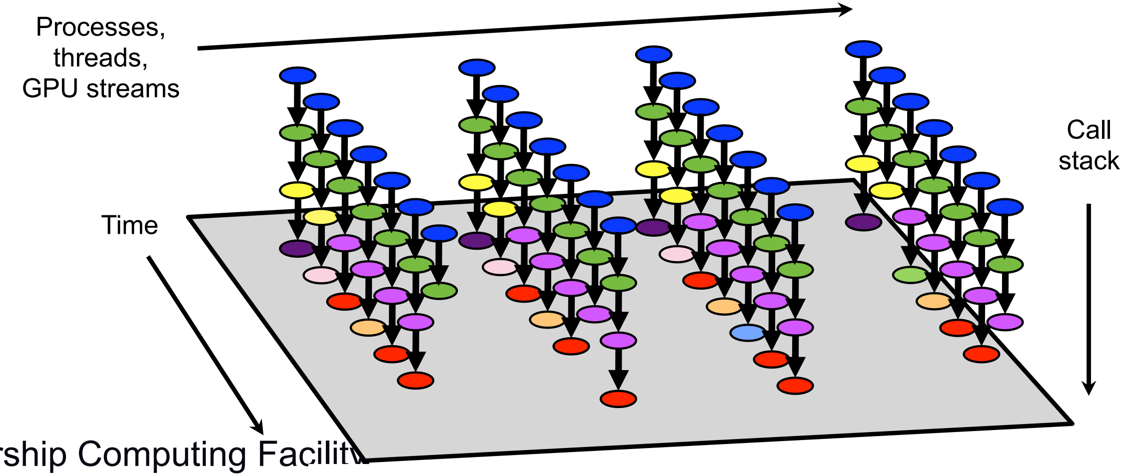


- **function calls in full context**
- **inlined procedures**
- **inlined templates**
- **outlined OpenMP loops**
- **loops**

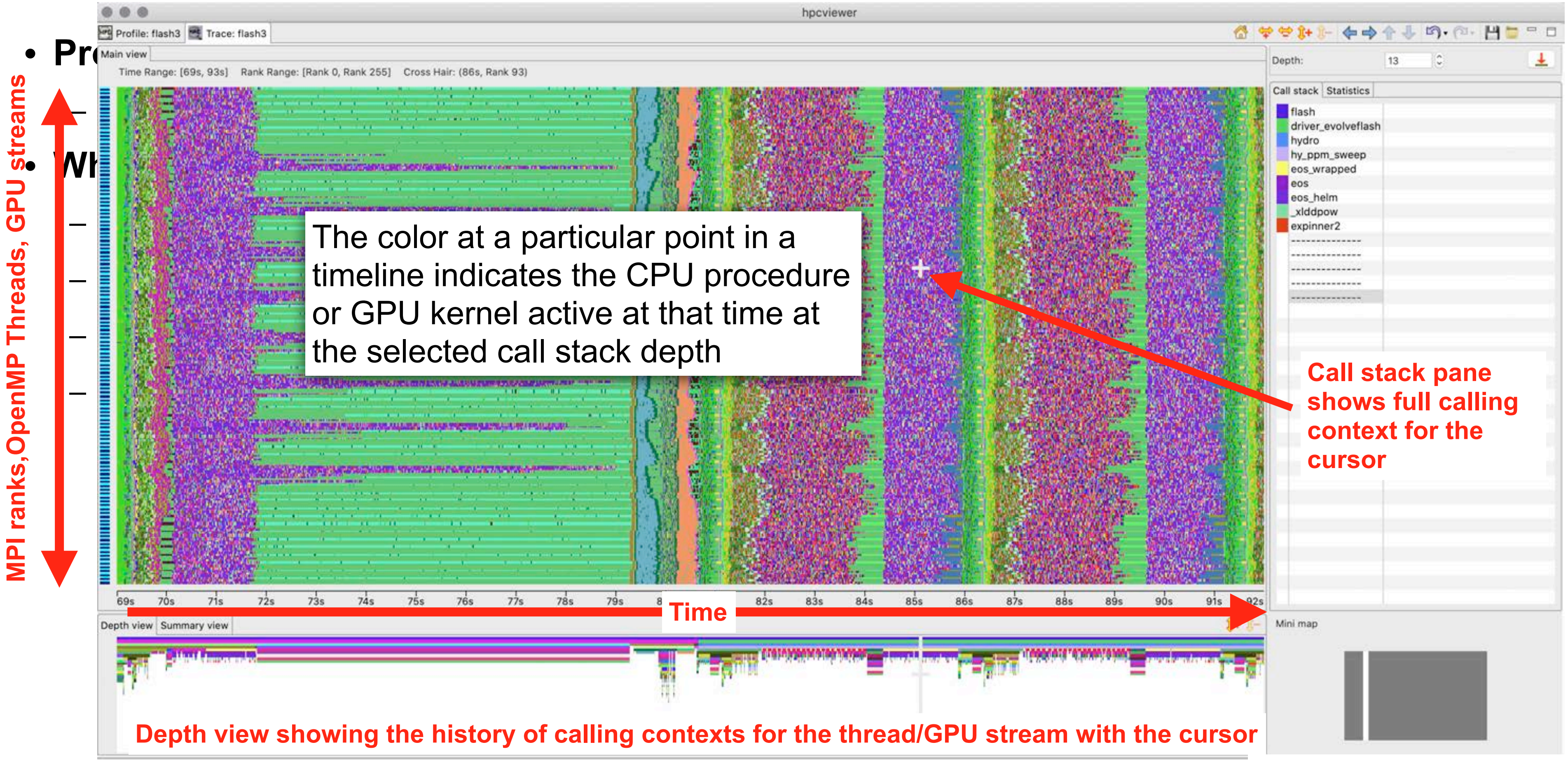
Argonne NATIONAL LABORATORY

Understanding Temporal Behavior

- **Profiling compresses out the temporal dimension**
  - Temporal patterns, e.g. serial sections and dynamic load imbalance are invisible in profiles
- **What can we do? Trace call path samples**
  - N times per second, take a call path sample of each thread
  - Organize the samples for each thread along a time line
  - View how the execution evolves left to right
  - What do we view? assign each procedure a color; view a depth slice of an execution



Time-centric Analysis with hpcviewer



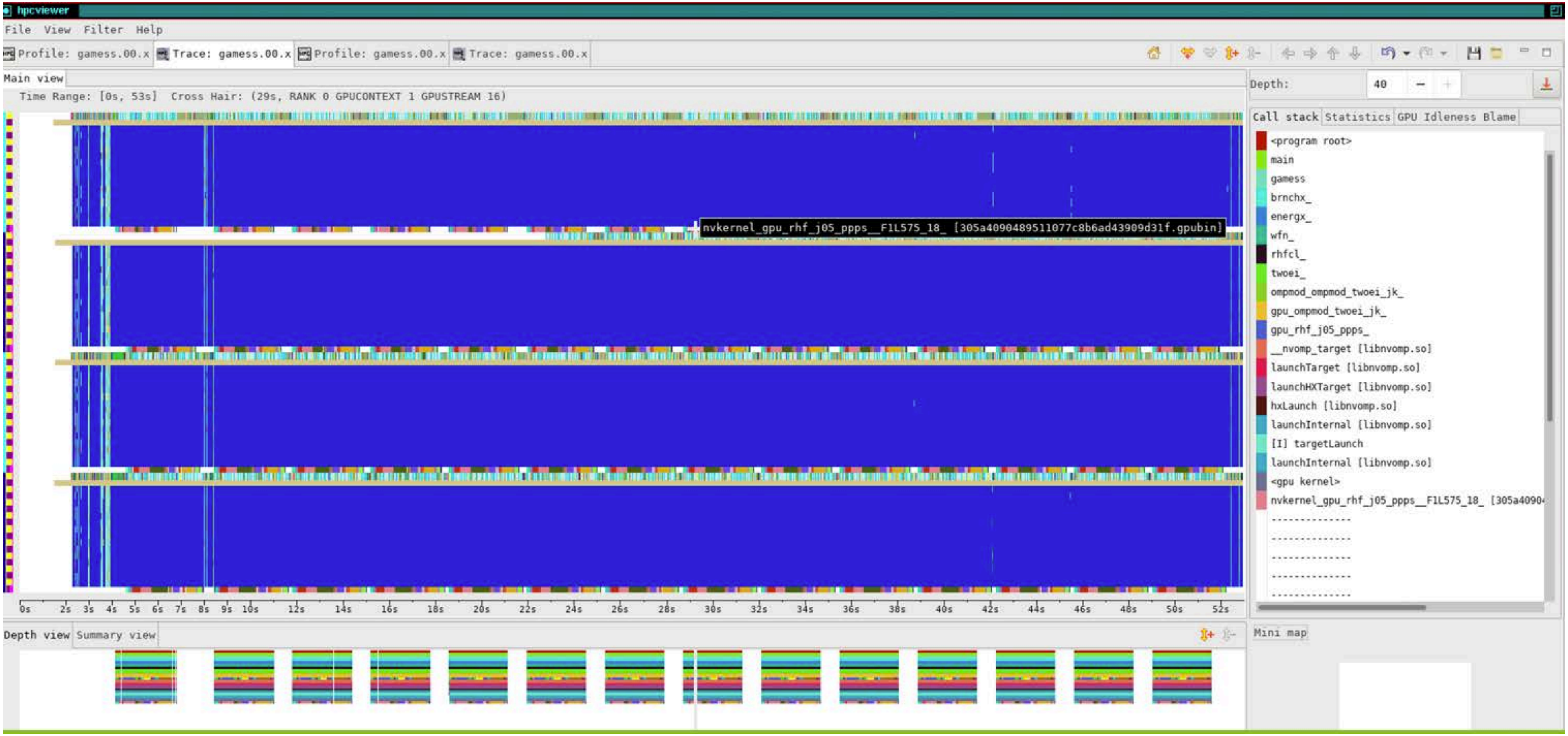
Minimap indicates part of execution trace shown

A multi-level call stack based view of execution over time

Case Studies

- **GAMESS: MPI, Fortran, OpenMP offloading**
- **Quicksilver: MPI, C++, CUDA**
- **LAMMPS: MPI, C++, Kokkos at exascale**

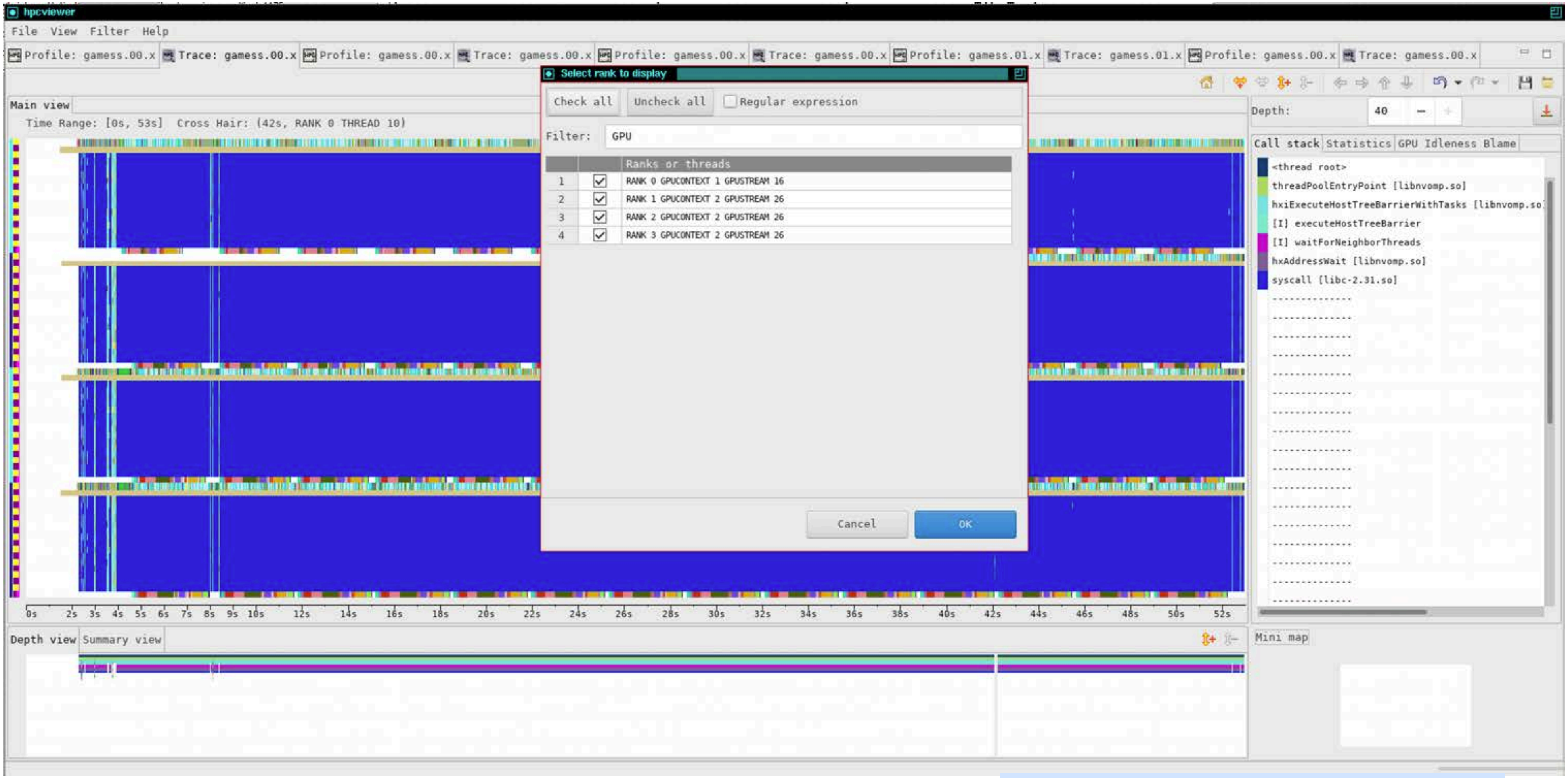
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



GAMESS original

All CPU threads and GPU streams

Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



GAMESS original

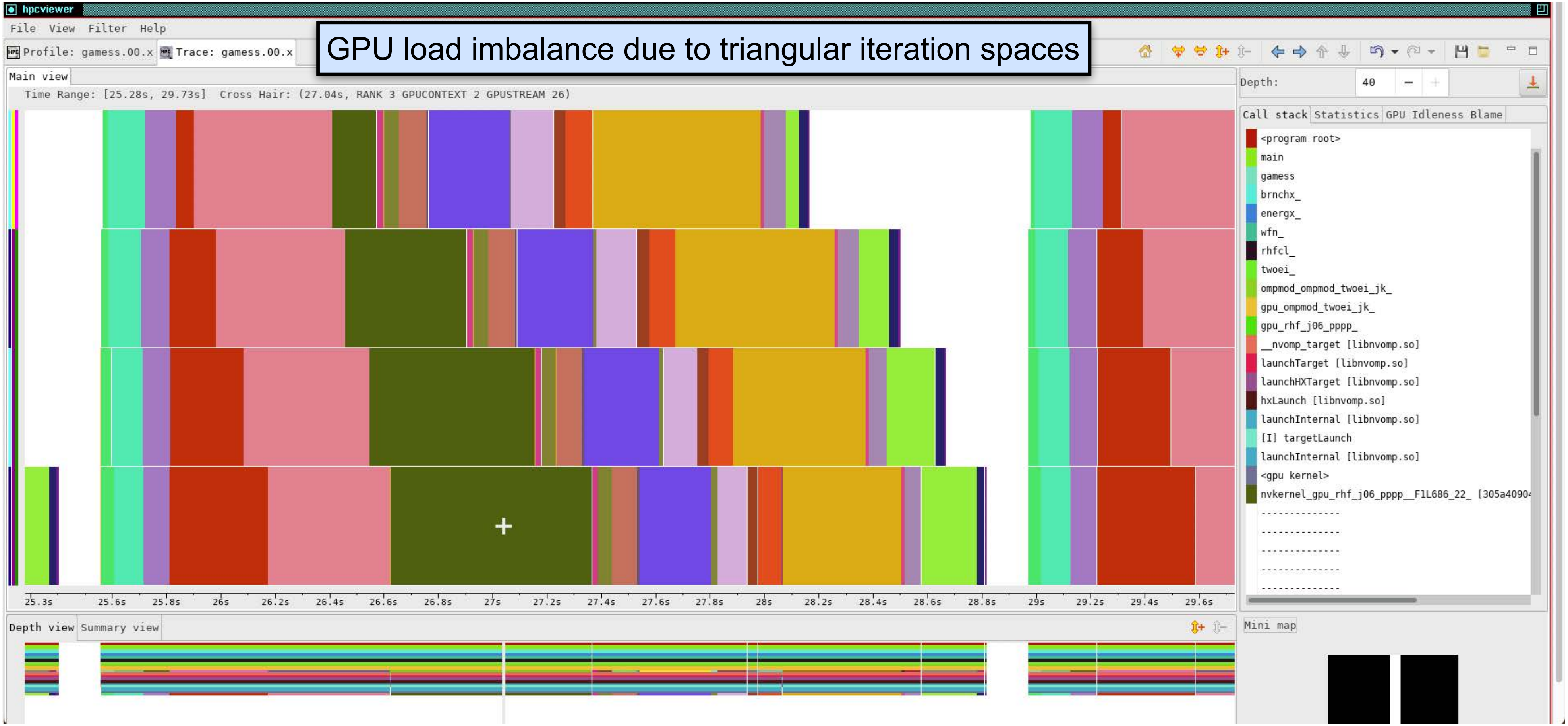
All CPU threads and GPU streams

Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter

GAMESS original

All GPU streams, whole execution

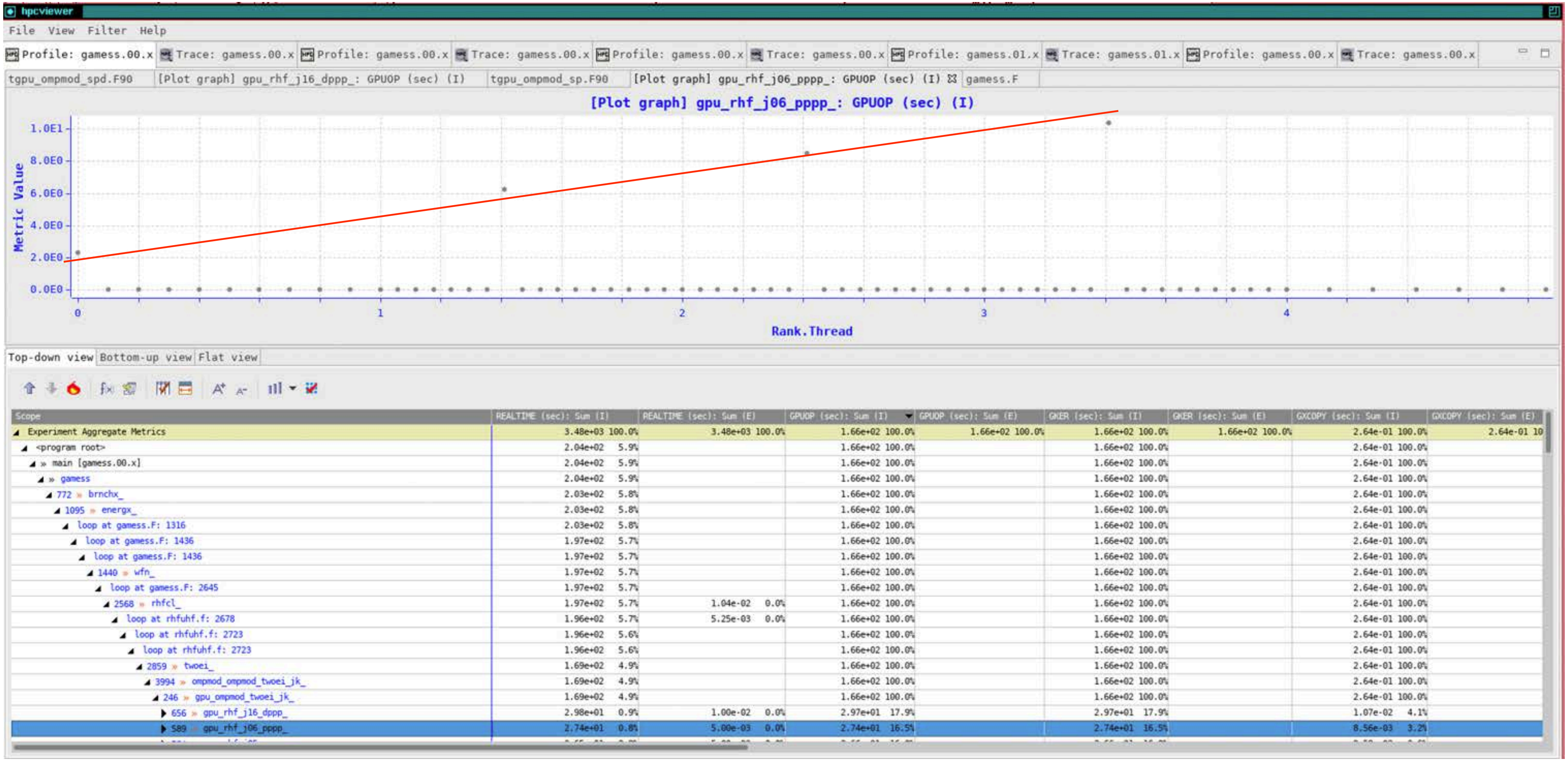
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



GAMESS original

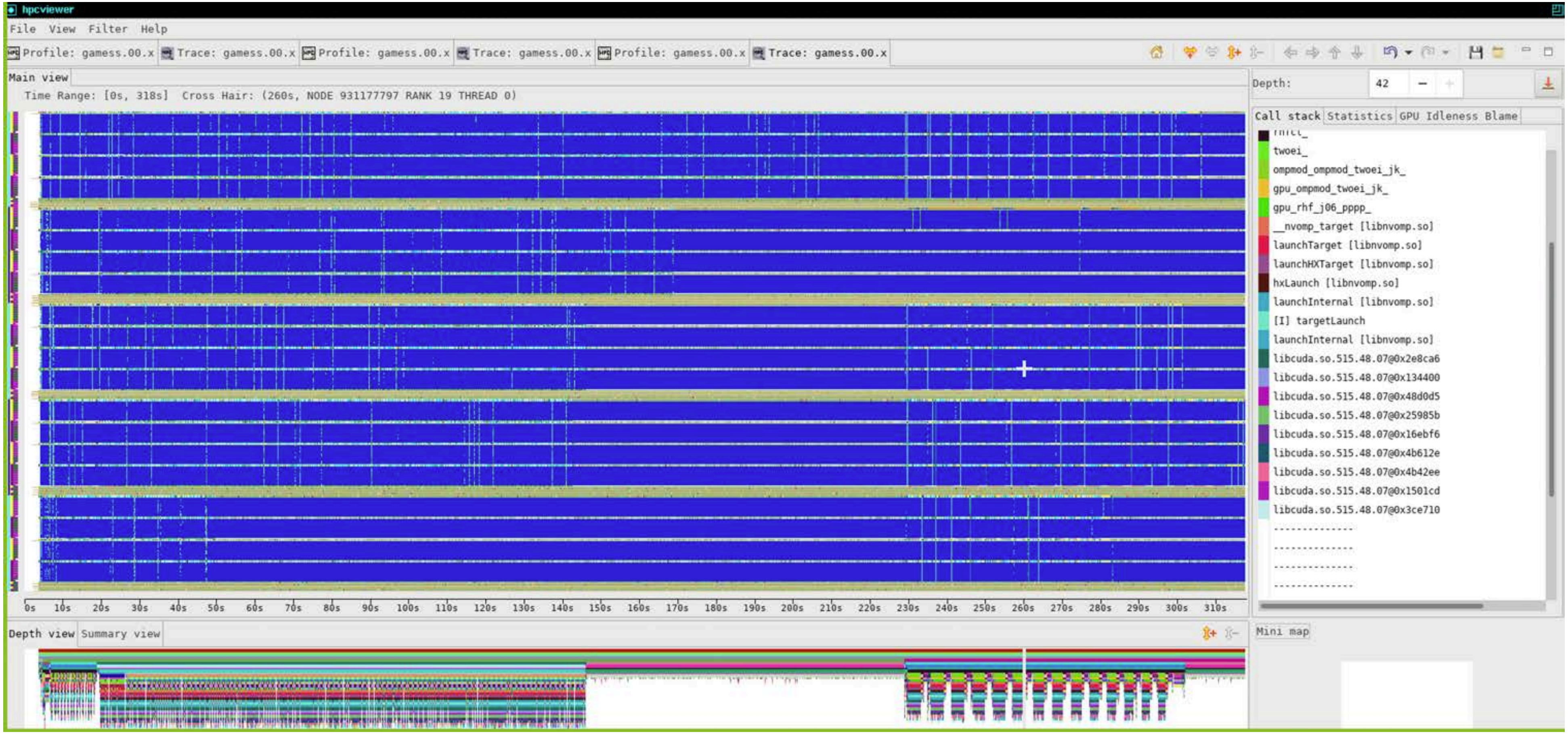
GPU streams: 1 iteration

Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



GAMESS original

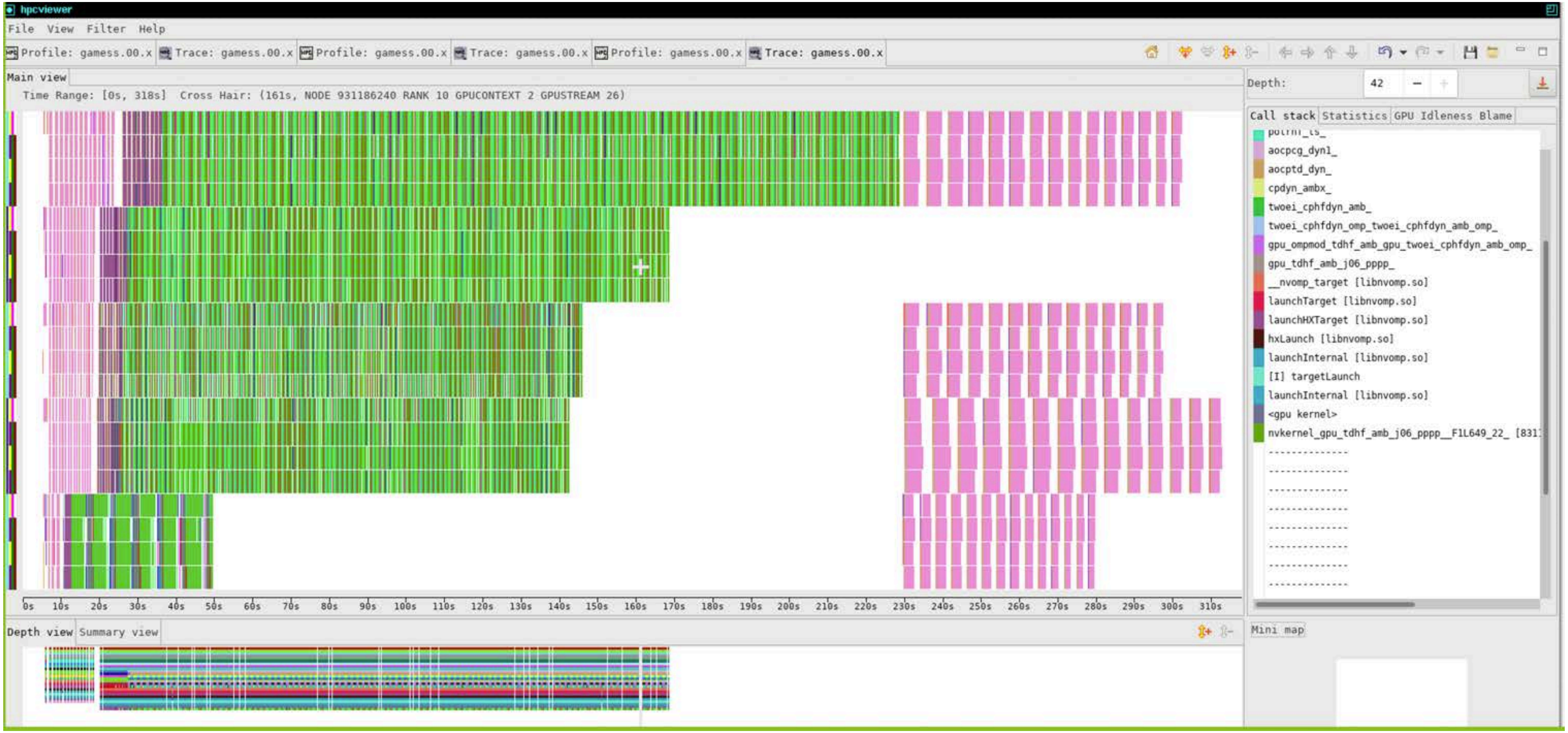
Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



GAMESS improved

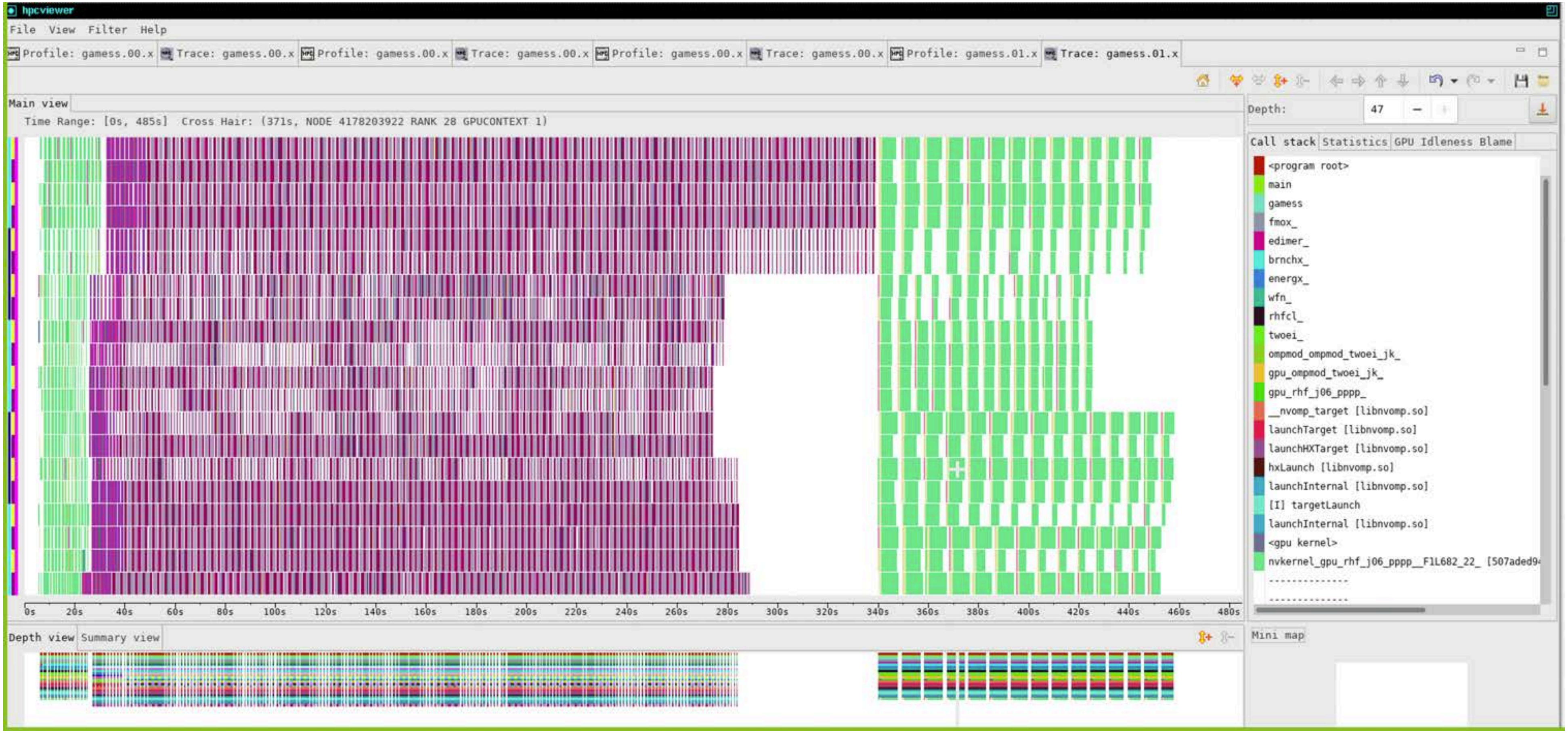
CPU Threads and GPU Streams

Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



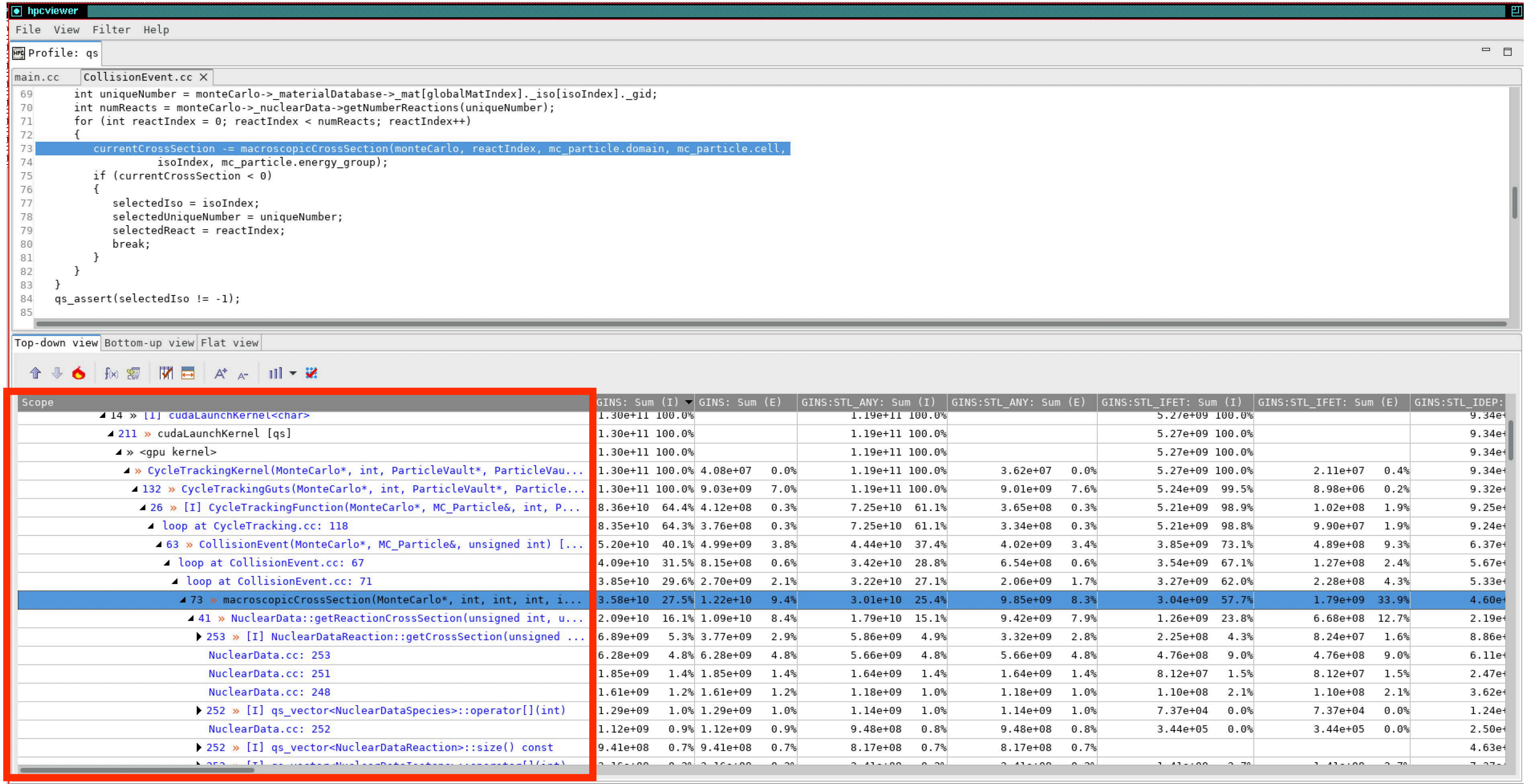
GAMESS improved

Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter

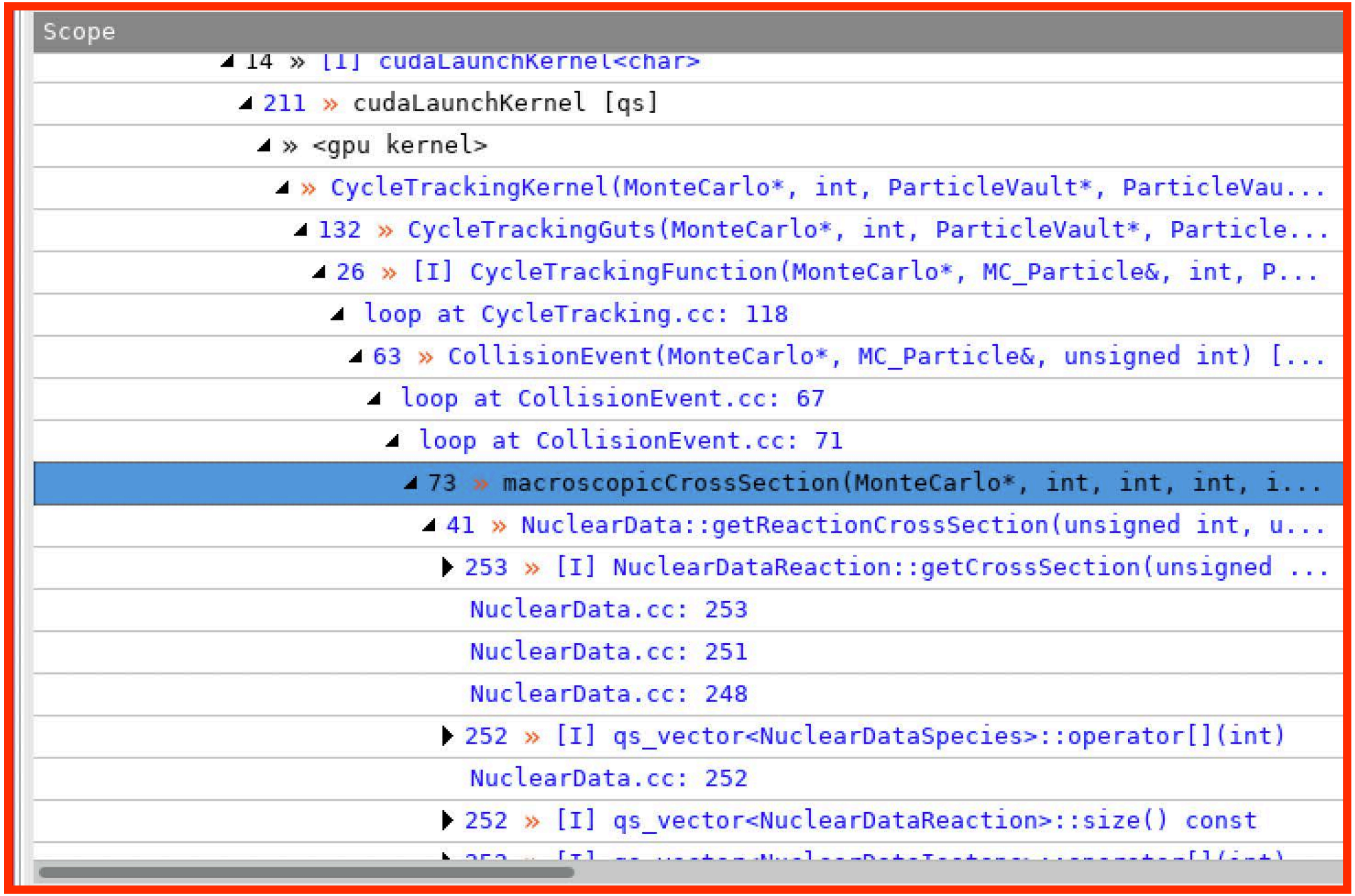


GAMESS improved with better manual distribution of work in input

Quicksilver: Detailed analysis within a Kernel using PC Sampling

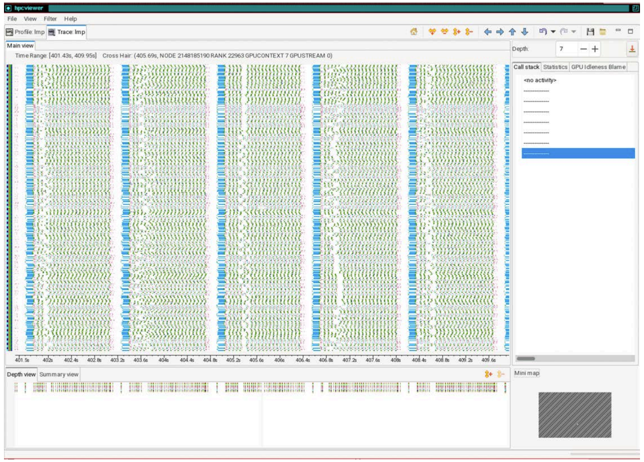


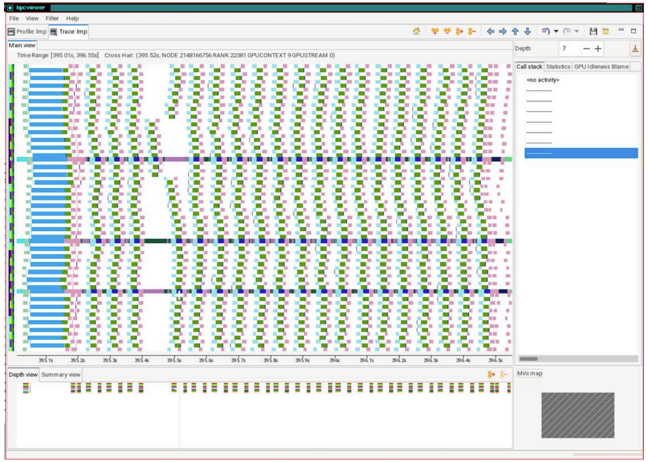
Quicksilver: Detailed analysis within a Kernel using PC Sampling

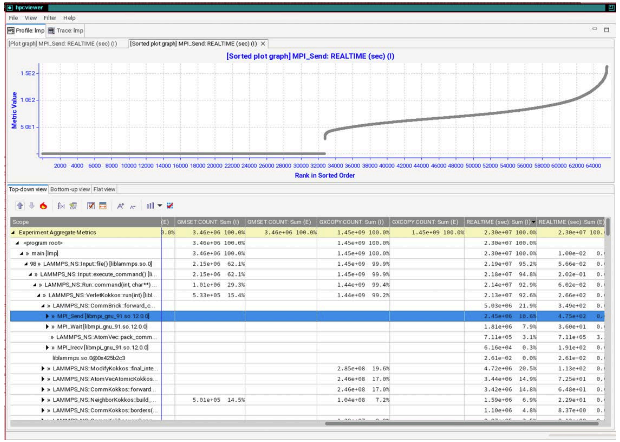


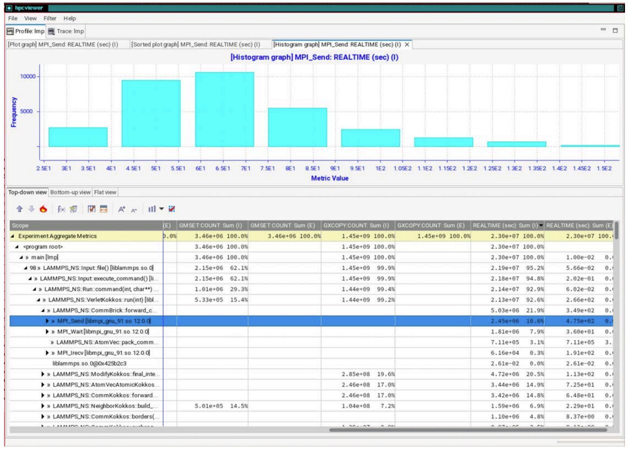
Argonne NATIONAL LABORATORY

LAMMPS on Frontier: Executions with Kernel Duration of Milliseconds

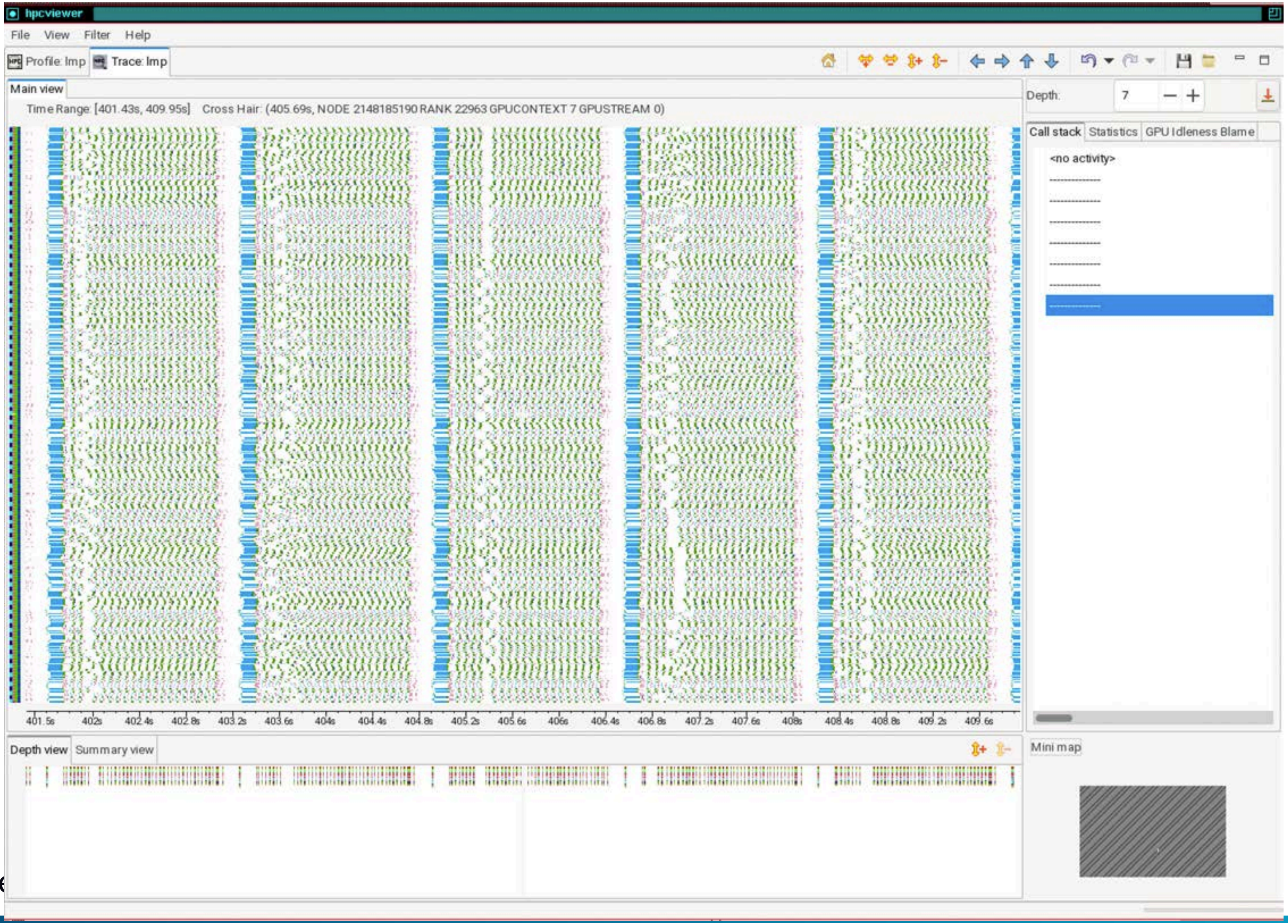






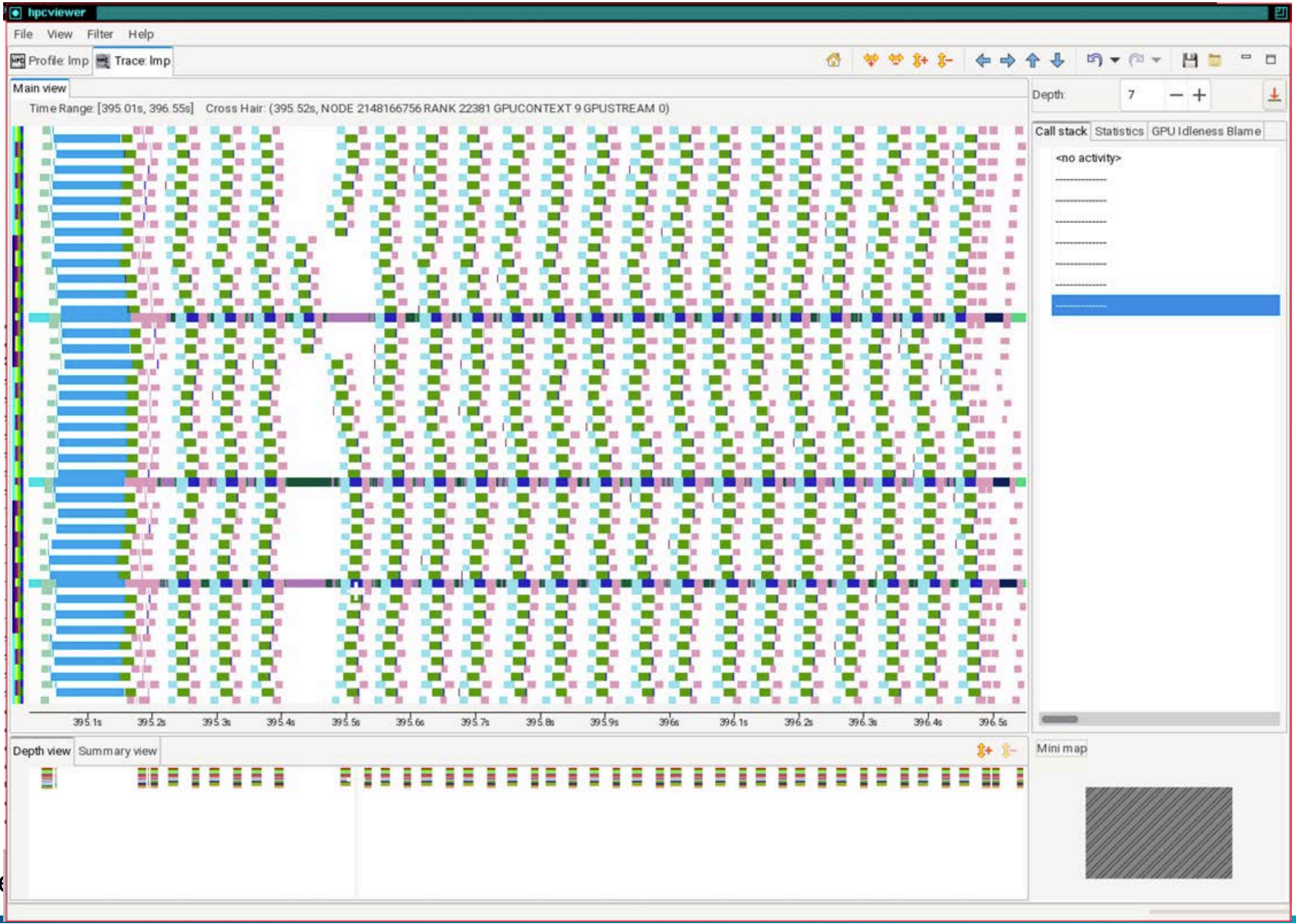


LAMMPS on Frontier: Executions with Kernel Duration of Milliseconds



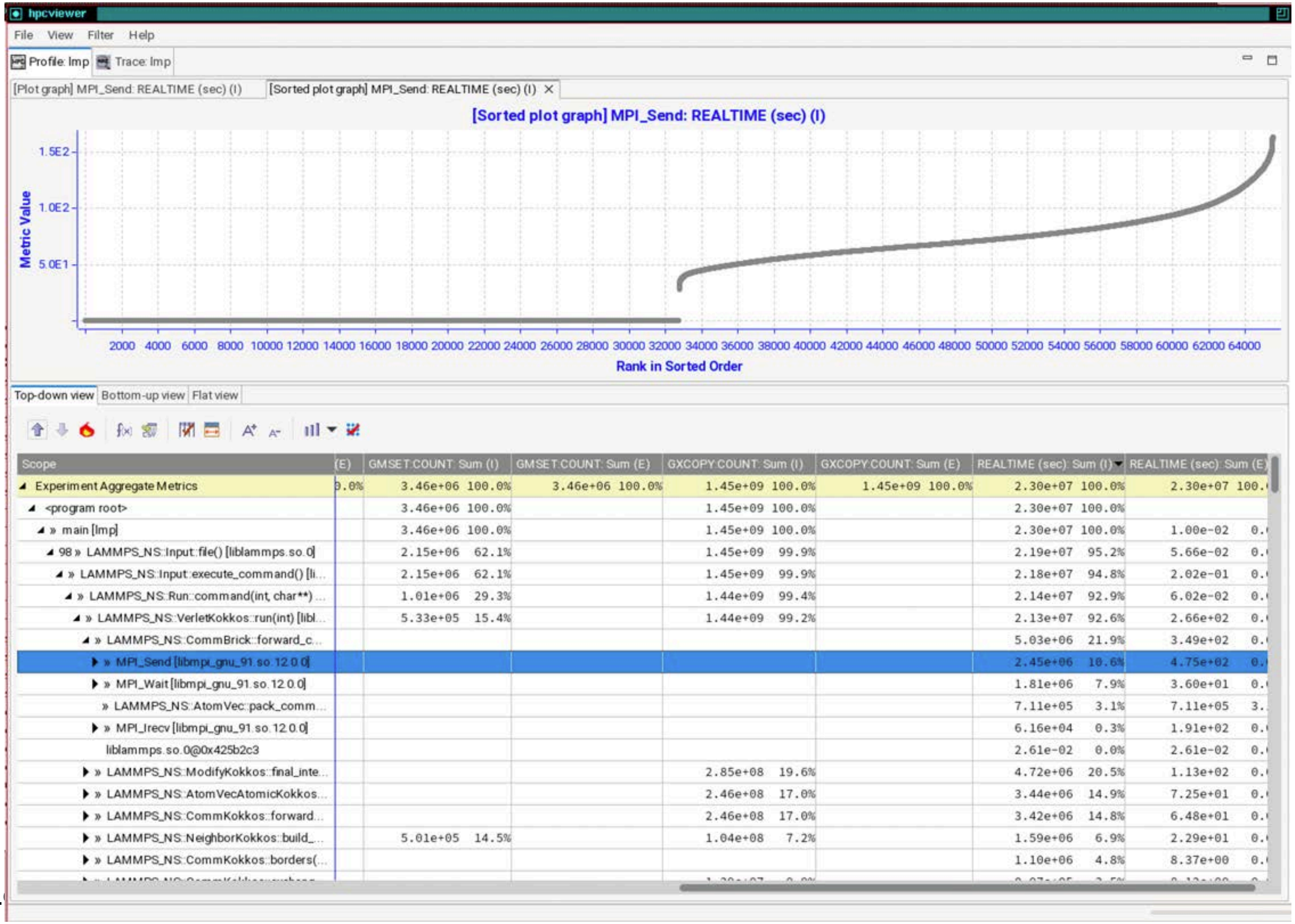
Argonne NATIONAL LABORATORY

LAMMPS on Frontier: Executions with Kernel Duration of Milliseconds



Argonne NATIONAL LABORATORY

LAMMPS on Frontier: Executions with Kernel Duration of Milliseconds



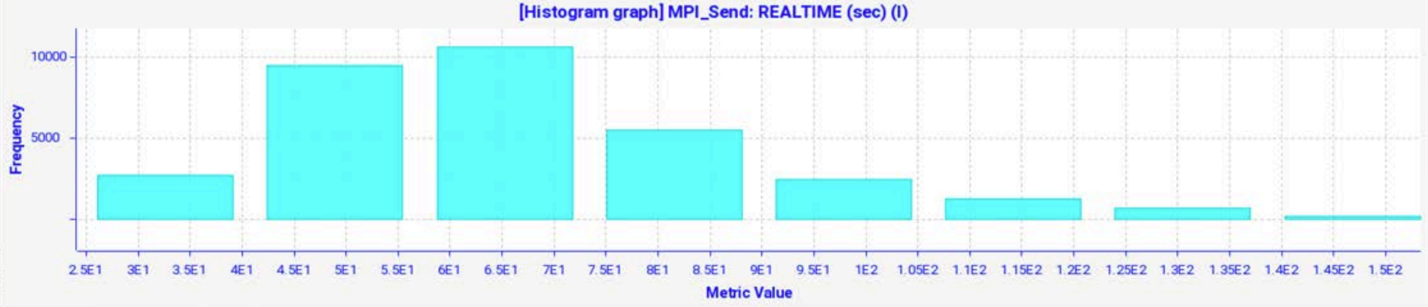
LAMMPS on Frontier: Executions with Kernel Duration of Milliseconds

hpcviewer

File View Filter Help

Profile: lmp | Trace: lmp

[Plot graph] MPI_Send: REALTIME (sec) (I) | [Sorted plot graph] MPI_Send: REALTIME (sec) (I) | [Histogram graph] MPI_Send: REALTIME (sec) (I)



Top-down view | Bottom-up view | Flat view

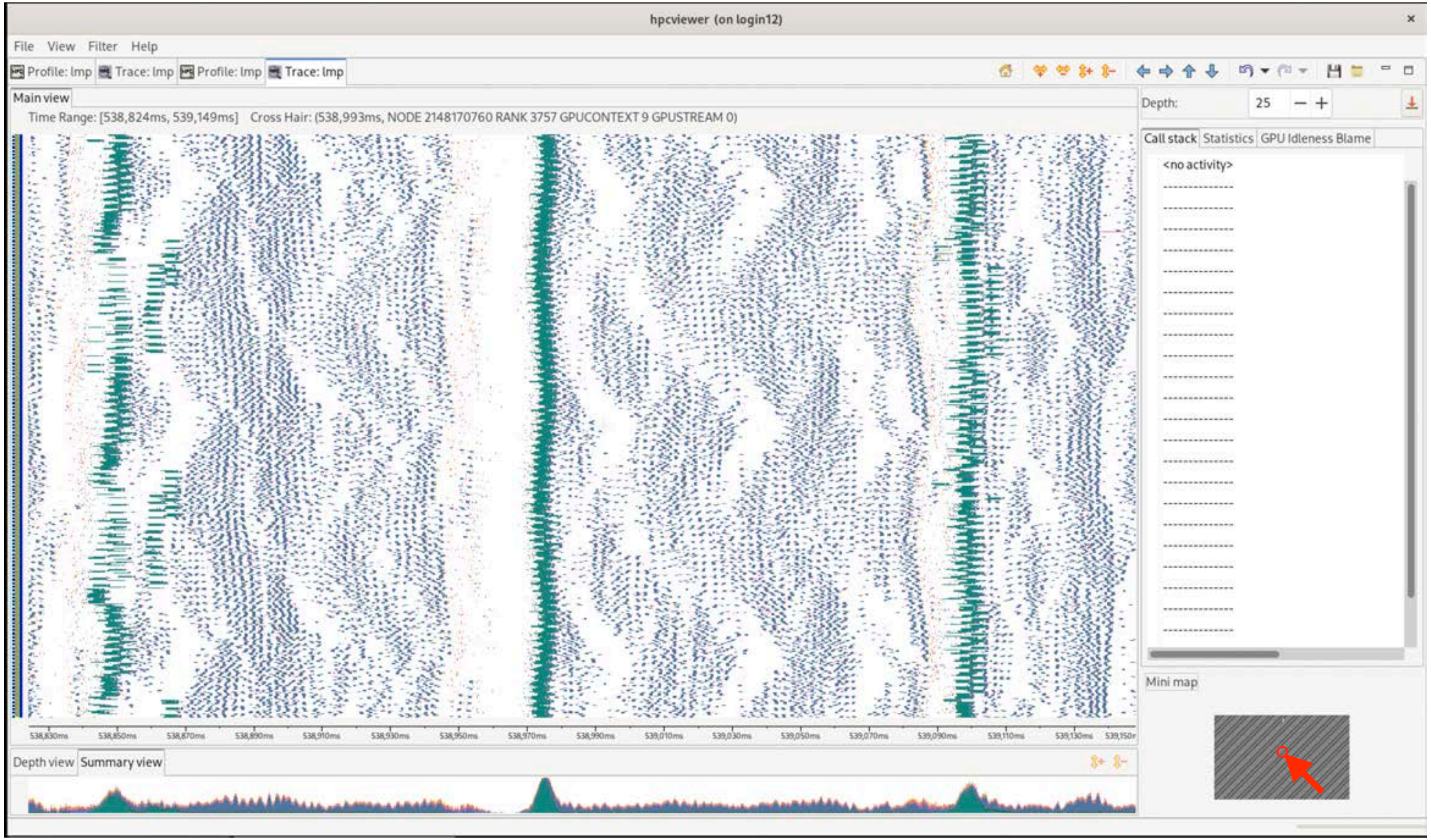
Scope	(E)	GMSET:COUNT: Sum (I)	GMSET:COUNT: Sum (E)	GXCOPY:COUNT: Sum (I)	GXCOPY:COUNT: Sum (E)	REALTIME (sec): Sum (I)	REALTIME (sec): Sum (E)
Experiment Aggregate Metrics	0.0%	3.46e+06 100.0%	3.46e+06 100.0%	1.45e+09 100.0%	1.45e+09 100.0%	2.30e+07 100.0%	2.30e+07 100.
<program root>		3.46e+06 100.0%		1.45e+09 100.0%		2.30e+07 100.0%	
» main [lmp]		3.46e+06 100.0%		1.45e+09 100.0%		2.30e+07 100.0%	1.00e-02 0.
98 » LAMMPS_NS::Input::file() [liblammps.so.0]		2.15e+06 62.1%		1.45e+09 99.9%		2.19e+07 95.2%	5.66e-02 0.
» LAMMPS_NS::Input::execute_command() [li...		2.15e+06 62.1%		1.45e+09 99.9%		2.18e+07 94.8%	2.02e-01 0.
» LAMMPS_NS::Run::command(int, char**) ...		1.01e+06 29.3%		1.44e+09 99.4%		2.14e+07 92.9%	6.02e-02 0.
» LAMMPS_NS::VerletKokkos::run(int) [libl...		5.33e+05 15.4%		1.44e+09 99.2%		2.13e+07 92.6%	2.66e+02 0.
» LAMMPS_NS::CommBrick::forward_c...						5.03e+06 21.9%	3.49e+02 0.
» MPI_Send [libmpi_gnu_91.so.12.0.0]						2.45e+06 10.6%	4.75e+02 0.
» MPI_Wait [libmpi_gnu_91.so.12.0.0]						1.81e+06 7.9%	3.60e+01 0.
» LAMMPS_NS::AtomVec::pack_comm...						7.11e+05 3.1%	7.11e+05 3.
» MPI_Irecv [libmpi_gnu_91.so.12.0.0]						6.16e+04 0.3%	1.91e+02 0.
liblammps.so.0@0x425b2c3						2.61e-02 0.0%	2.61e-02 0.
» LAMMPS_NS::ModifyKokkos::final_inte...				2.85e+08 19.6%		4.72e+06 20.5%	1.13e+02 0.
» LAMMPS_NS::AtomVecAtomicKokkos...				2.46e+08 17.0%		3.44e+06 14.9%	7.25e+01 0.
» LAMMPS_NS::CommKokkos::forward...				2.46e+08 17.0%		3.42e+06 14.8%	6.48e+01 0.
» LAMMPS_NS::NeighborKokkos::build_...		5.01e+05 14.5%		1.04e+08 7.2%		1.59e+06 6.9%	2.29e+01 0.
» LAMMPS_NS::CommKokkos::borders(...						1.10e+06 4.8%	8.37e+00 0.

Argonne NATIONAL LABORATORY

LAMMPS on Frontier: 8K nodes, 64K MPI ranks + GPU times

Kernel duration of microseconds

Trace: 4.3TB



Argonne NATIONAL LABORATORY

HPCToolkit Status on AMD, Intel, NVIDIA GPUs

- **Heterogeneous profiles**
- **GPU operation traces**
- **Hardware counters to measure GPU kernels**
- **Instruction-level measurement within GPU kernels**
  - NVIDIA: PC sampling
  - Intel: binary instrumentation
  - AMD: PC sampling coming soon!

HPCToolkit Documentation and Training

- **User Manual**
  - http://www.hpctoolkit.org/manual/HPCToolkit-users-manual.pdf
- **Installing HPCToolkit's hpcviewer on your Laptop**
  - http://www.hpctoolkit.org/download.html
- **Training slides and videos: http://www.hpctoolkit.org/training.html**
  - Introduction to HPCToolkit [Youtube (13:22)] [Slides]
  - Sampling-based Performance Analysis with HPCToolkit [Youtube (23:09)] [Slides]
  - Identifying Scalability Bottlenecks with HPCToolkit [Youtube (19:27)] [Slides]
  - Analyzing GPU-accelerated Applications with HPCToolkit [Youtube (23:59)] [Slides]
  - Using HPCToolkit to Analyze the Performance of GPU-accelerated Applications [Youtube (1:27:35)] [Slides]
  - Analyzing GPU-accelerated Applications Using HPCToolkit [Youtube (34:02)] [Slides]
  - HPCToolkit Graphical User Interface [Youtube (35:03)] [Slides]
  - Analyzing CPU Applications with HPCToolkit [Youtube (2:24:30)] [Slides]
- **Downloading and Installing HPCToolkit**
  - http://www.hpctoolkit.org/software-instructions.html

Want Some Help?

- **Join our hpctoolkit-ECP Slack workspace**
  - https://join.slack.com/t/hpctoolkit-ecp/shared_invite/zt-24rtkvwma-4HNYe~TiwFwEiJpH~RqUuw
- **Send email to our mailing list**
  - hpctoolkit-forum@rice.edu

Hands-on Directions

- **Log into Polaris with X11 forwarding for hpcviewer**
  - `ssh -Y <username>@polaris.alcf.anl.gov`
- **Download example programs to measure and analyze**
  - `git clone https://github.com/hpctoolkit-tutorial-examples`
- **Load HPCToolkit into your environment**
  - `module use /soft/perftools/hpctoolkit/polaris/modulefiles`
  - `module load hpctoolkit`
- **Set up environment variables for running HPCToolkit examples at the workshop**
  - `export HPCTOOLKIT_TUTORIAL_PROJECTID=fallwkshp23`
  - `export HPCTOOLKIT_TUTORIAL_RESERVATION=fallws23single`
  - `export HPCTOOLKIT_HPCSTRUCT_CACHE=$HOME/.hpctoolkit/hpcstruct-cache`

Working with HPCToolkit's Tutorial Examples

- **Examples in** `hpctoolkit-tutorial-examples/examples/gpu`
  - Quicksilver (highly recommended)
  - LAGHOS
  - MiniQMC
  - PeleC
  - LAMMPS
- **Working with an example**
  - `cd <example-name>`
  - `source setup-env/polaris.sh`
  - `make build`
  - `make` *# enumerates commands for running jobs and inspecting their results*
- **Pro tip**
  - `watch ls -l` # wait until a done file appears, e.g. `log.run-pc.done` to indicate your data is ready
  - also look at the `log.*.stderr` and `log*.out` logs

Argonne NATIONAL LABORATORY