



SC21

St. Louis, MO | science & beyond.

Analyzing GPU-accelerated Applications with HPCToolkit

John Mellor-Crummey

Department of Computer Science
Rice University

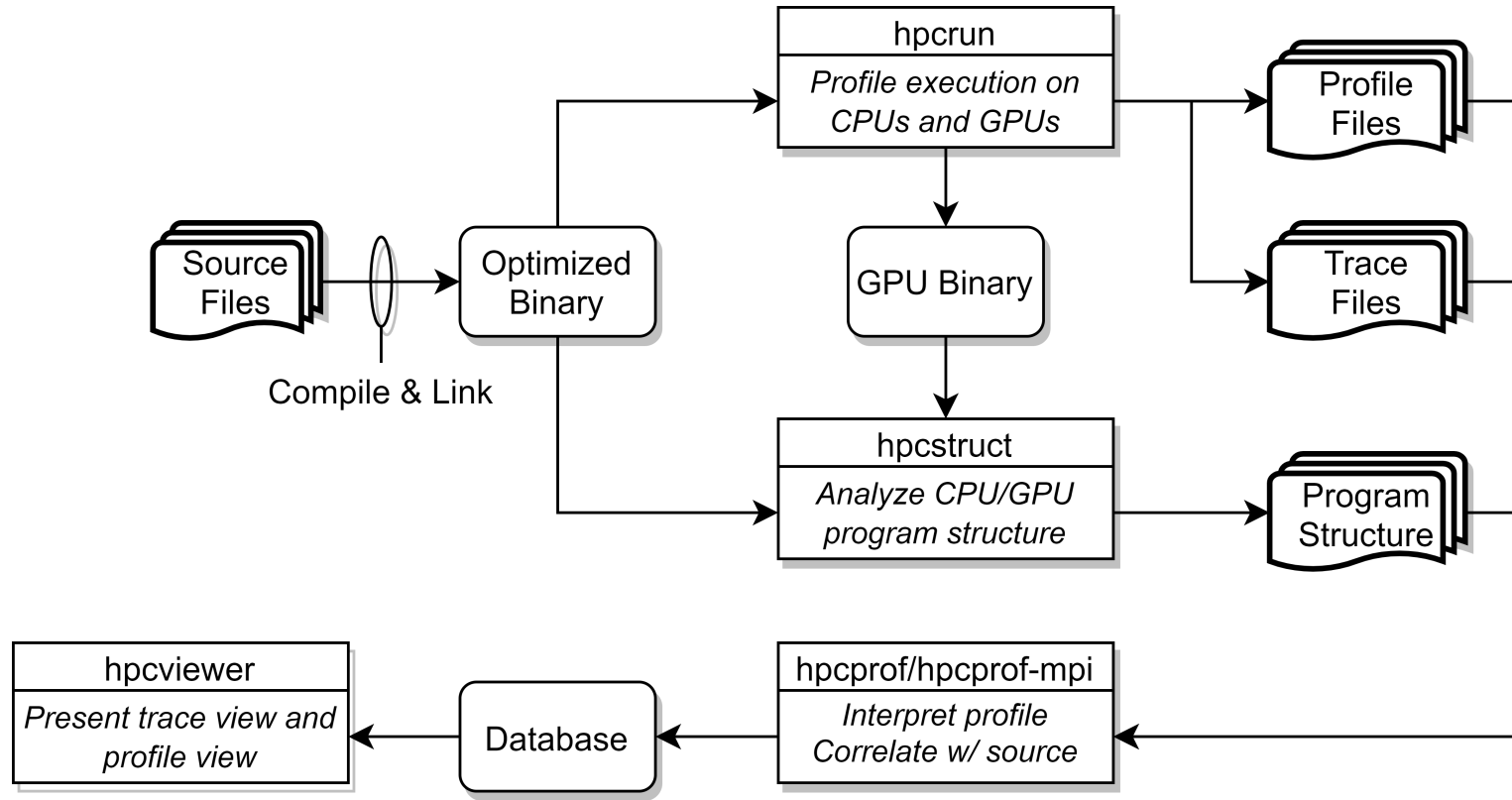
15 November 2021



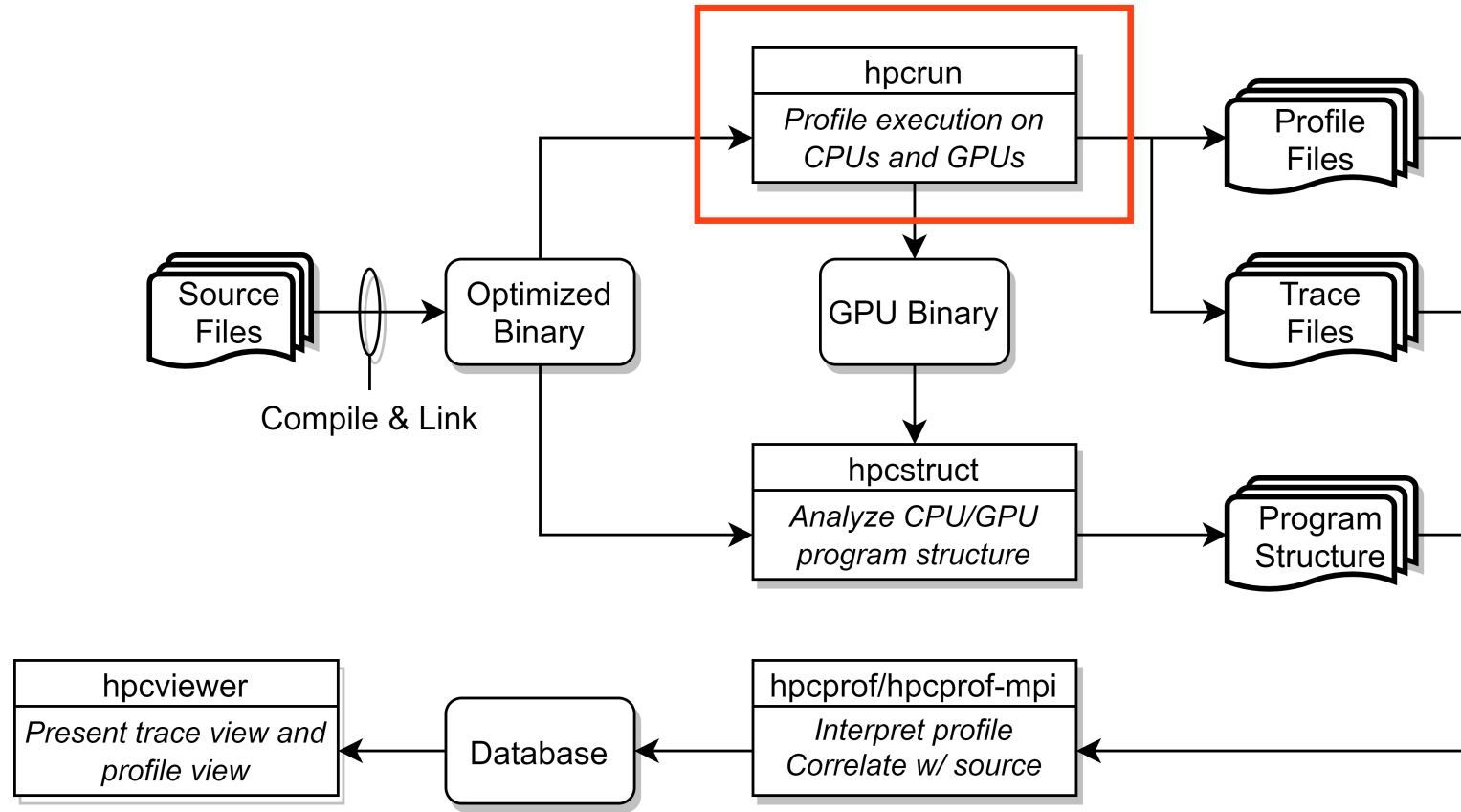
Outline

- **Describe use of HPCToolkit for GPU-accelerated applications**
- **Status for GPU vendors**
- **Demo on NVIDIA GPU**
 - operation-level monitoring
 - PC sampling
- **Ongoing work**

HPCToolkit's Workflow for GPU-accelerated Applications



HPCToolkit's Workflow for GPU-accelerated Applications



hpcrun - Measure CPU and GPU execution

- GPU profiling

- hpcrun -e gpu=**xxx** <app>

// **xxx** ∈ {nvidia,amd,opencl,level0}

- GPU tracing (-t)

- hpcrun -e gpu=**yyy** -t <app>

// **yyy** ∈ {nvidia,amd,opencl}

- GPU PC sampling (NVIDIA only)

- hpcrun -e gpu=**nvidia**,pc <app>

- CPU and GPU profiling and tracing

- hpcrun -e REALTIME -e gpu=**yyy** -t <app>

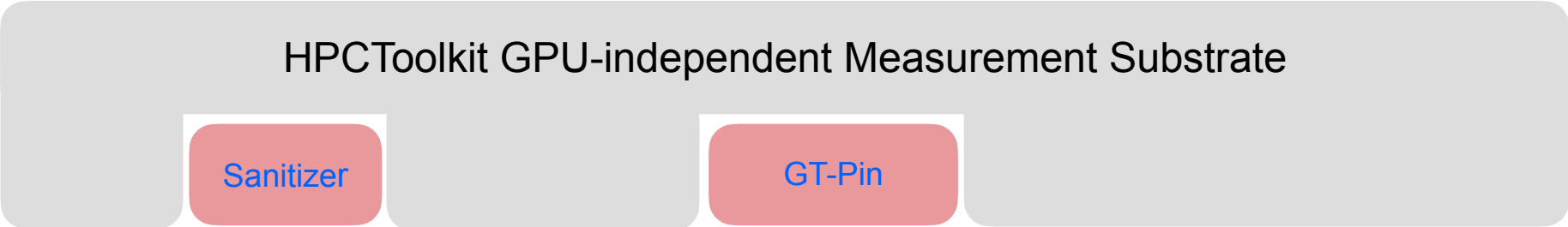
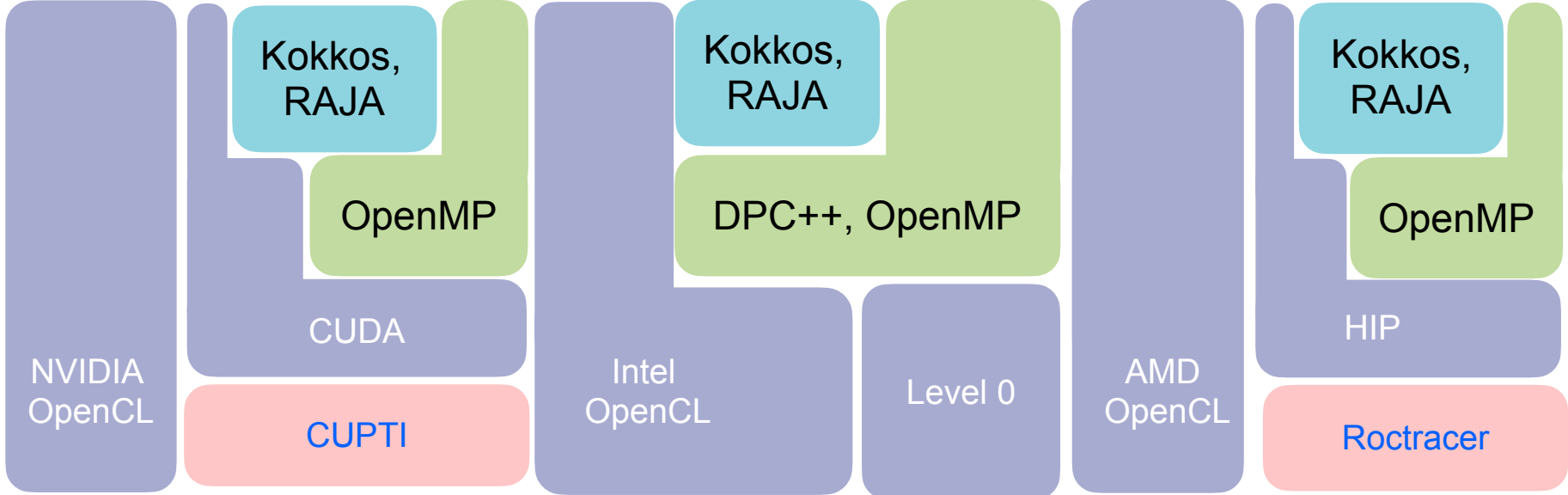
- Use hpcrun with job launchers

- jsrun -n 32 -g 1 -a 1 hpcrun -e gpu=**xxx** <app>

- srun -n 1 -G 1 hpcrun -e gpu=**xxx** <app>

- aprun -n 16 -N 8 -d 8 hpcrun -e gpu=**xxx** <app>

HPCToolkit and GPU Software Stacks



Measurement for GPU-accelerated Supercomputers

- **Measurement interfaces**

- Hardware
 - CPU hardware performance monitoring unit
 - GPU hardware counters and PC sampling
- Software
 - Glibc LD_AUDIT for tracking dynamic loading of shared libraries
 - Linux perf_events for kernel measurement
 - GPU monitoring and instrumentation libraries from vendors

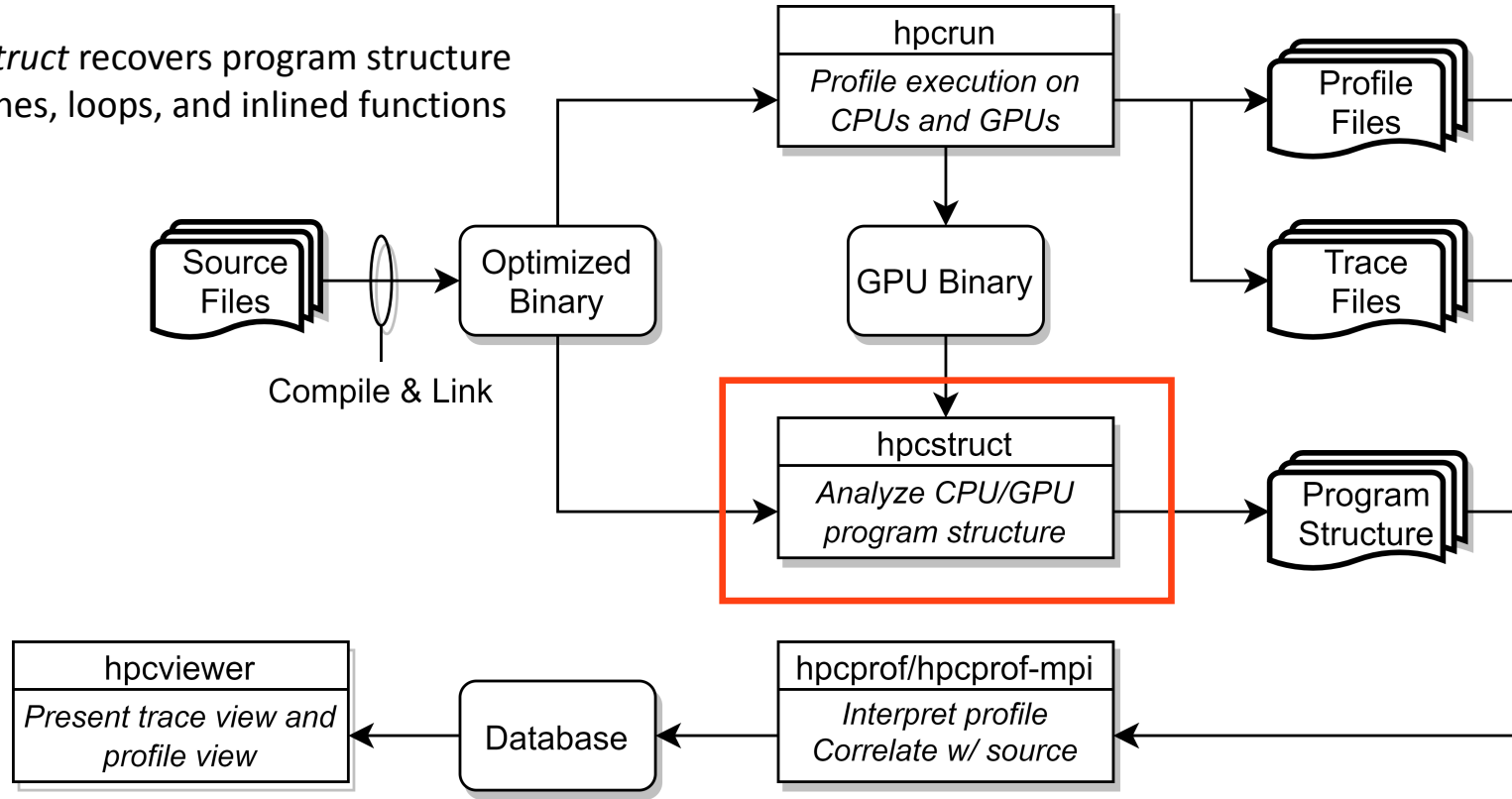
- **Multiple measurement modalities and interfaces**

- Sampling on the CPU
- Callbacks when GPU operations are launched and (sometimes) completed
- GPU event stream, including PC sampling measurements

HPCToolkit's Workflow for GPU-accelerated Applications

Step 3:

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



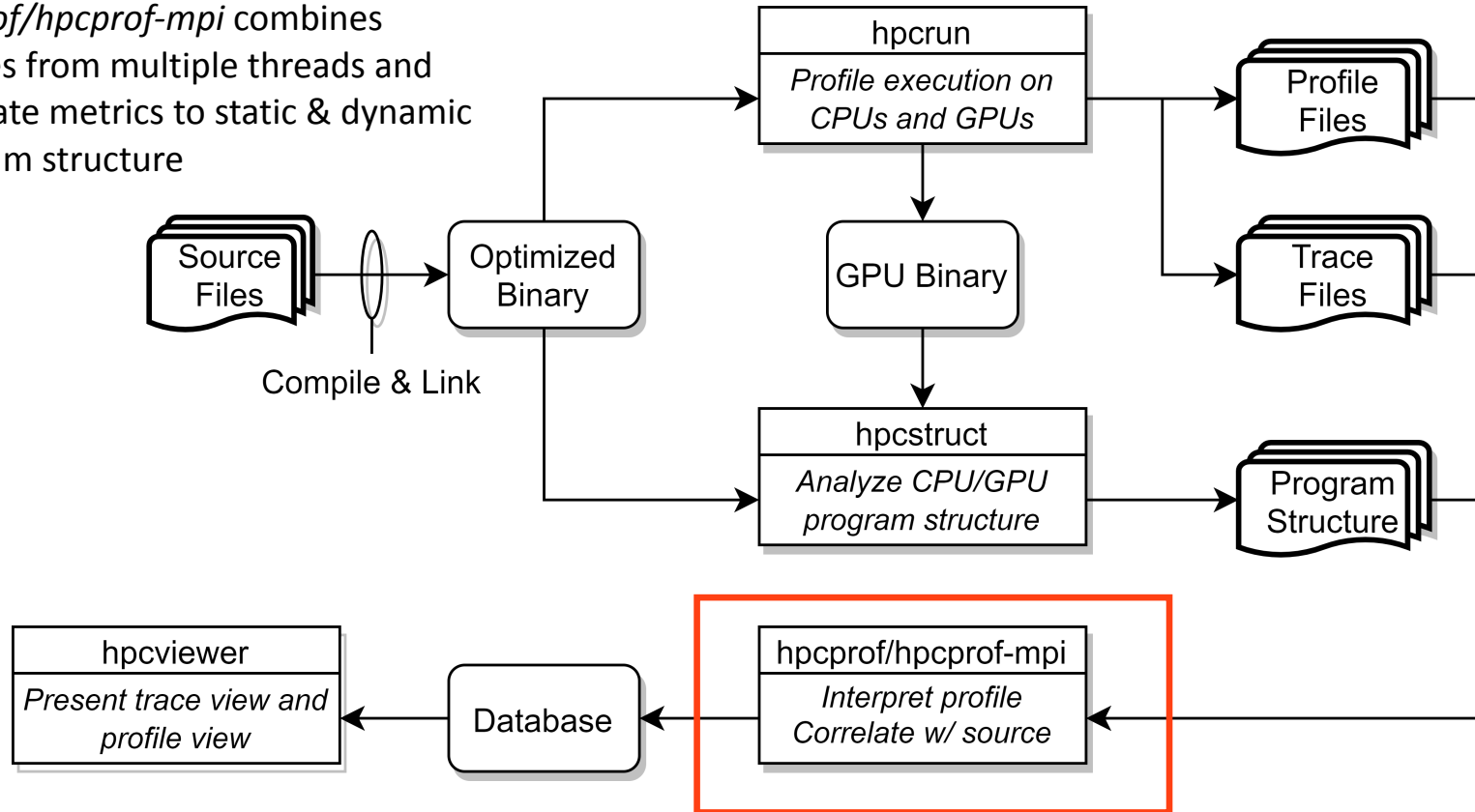
hpcstruct - Recover Structure for GPU-accelerated Programs

- Analyze all GPU binaries in <measurements-dir>
 - hpcstruct [--gpucfg yes] <measurements-dir>
 - “gpucfg yes” means recover GPU loop nests, calling context information
 - only useful on NVIDIA platforms at present: no fine-grain measurement from other vendors
 - use with care on large binaries: very costly because of NVIDIA’s lack of necessary APIs
 - adds a program structure file to the measurement directory for each GPU binary

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



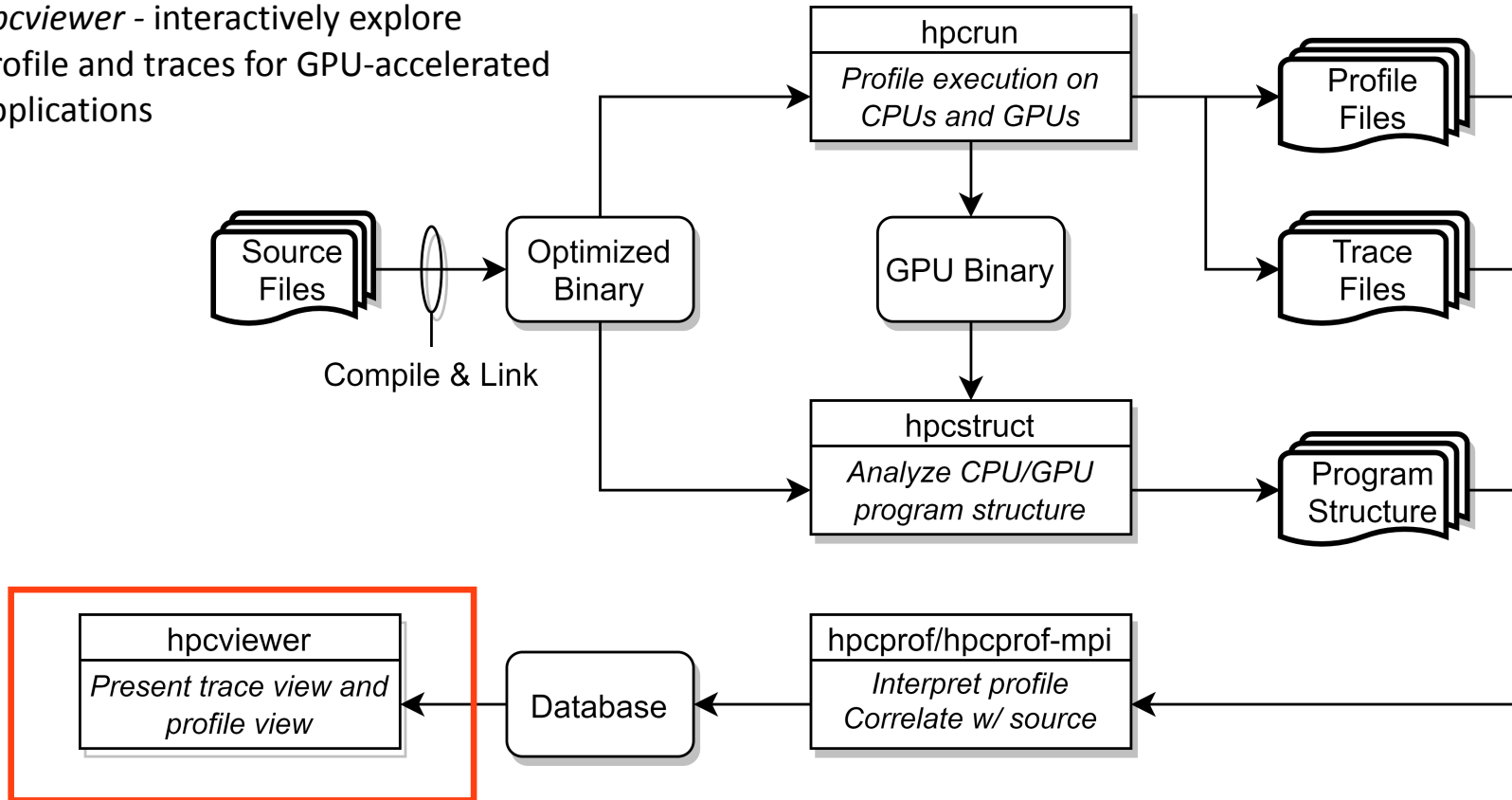
hpcprof/hpcprof-mpi - Correlate Measurements with Code

- Use a single process to combine performance data
 - `hpcprof <measurements-dir>`
- Use multiple processes to combine large-scale performance data
 - `jsrun -n <np> hpcprof-mpi <measurements-dir>`
 - `srun -n <np> hpcprof-mpi <measurements-dir>`

HPCToolkit's Workflow for GPU-accelerated Applications

Step 4:

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



GPU Monitoring Capabilities of HPCToolkit

Measurement Capability	NVIDIA	AMD	Intel
kernel launches, explicit memory copies, synchronization	callbacks + activity API	callbacks + Activity API	callbacks
instruction-level measurement and analysis	PC sampling of GPU code	Future*: PC sampling (as seen on Github) of GPU code	GTPin; Future*: instruction-level measurement of GPU code
kernel characteristics	Activity API	(available statically)	(unknown)

Significant support in master branch

Prototype support in master branch

Prototype support in master branch

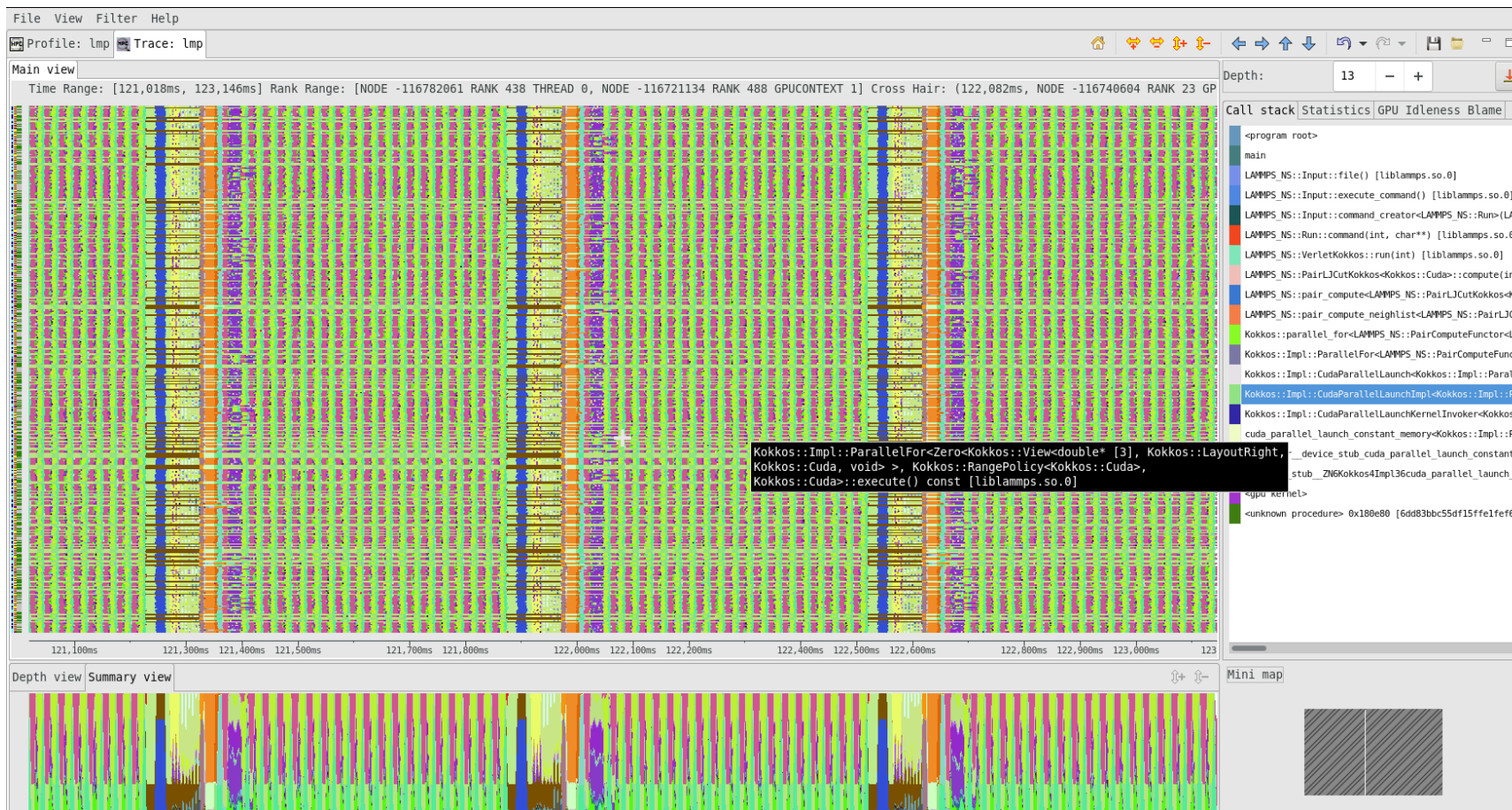
Demo: Measure and Analyze LLNL's Quicksilver (Video)

The screenshot shows a Linux desktop environment with a terminal window displaying the source code of a program named 'quicksilver'. The code is written in C++ and includes comments and function calls. The terminal window is titled 'hpcviewer (on ufront.cs.rice.edu)'. Below the code editor, there is a performance analysis tool interface showing a flame graph and a table of performance metrics.

The performance analysis tool displays a table of performance metrics for various scopes. The table has columns for Scope, GINS:Sum, GINS:Sum [E], GINS:STL, ANV:S, GINS:STL ANV:S, GINS:STL IFET:S, GINS:STL IFET:S, GINS:STL IDEP:S, and G1. The table shows performance data for various scopes, including 'Experiment Aggregate Metrics', 'program root', 'main', 'cycleTracking(MonteCarlo)', 'device stub_Z19CycleTrackingKernel', 'cudaLaunchKernel [qs]', 'gpu kernel', 'CycleTrackingKernel(MonteCarlo)', 'CycleTrackingGuts(MonteCarlo)', 'loop at CycleTracking.cc: 110', 'CollisionEvent(MonteCarlo)', 'loop at CollisionEvent.cc: 67', 'loop at CollisionEvent.cc: 71', 'NuclearData::getR...', 'NuclearData.cc: 253', 'NuclearData.cc: 251', 'NuclearData.cc: 252', 'NuclearData.cc: 248', 'MacroscopicCrossSection...', and 'loop at CollisionEvent.cc: 71'.

Scope	GINS:Sum	GINS:Sum [E]	GINS:STL	ANV:S	GINS:STL ANV:S	GINS:STL IFET:S	GINS:STL IFET:S	GINS:STL IDEP:S	G1
Experiment Aggregate Metrics	1.73e+11 100.0%	1.73e+11 100.0%	8.11e+10 100.0%	8.11e+10 100.0%	5.18e+09 100.0%	5.18e+09 100.0%	9.65e+09 100.0%		
program root	1.73e+11 100.0%		8.11e+10 100.0%		5.18e+09 100.0%		9.65e+09 100.0%		
main	1.73e+11 100.0%		8.11e+10 100.0%		5.18e+09 100.0%		9.65e+09 100.0%		
cycleTracking(MonteCarlo)	1.73e+11 100.0%		8.11e+10 100.0%		5.18e+09 100.0%		9.65e+09 100.0%		
device stub_Z19CycleTrackingKernel	1.73e+11 100.0%		8.11e+10 100.0%		5.18e+09 100.0%		9.65e+09 100.0%		
cudaLaunchKernel [qs]	1.73e+11 100.0%		8.11e+10 100.0%		5.18e+09 100.0%		9.65e+09 100.0%		
gpu kernel	1.73e+11 100.0%		8.11e+10 100.0%		5.18e+09 100.0%		9.65e+09 100.0%		
CycleTrackingKernel(MonteCarlo)	1.73e+11 100.0%	7.21e+07 0.0%	8.11e+10 100.0%	3.42e+07 0.0%	5.18e+09 100.0%	1.86e+07 0.4%	9.65e+09 100.0%		
CycleTrackingGuts(MonteCarlo)	1.73e+11 99.9%	1.40e+10 8.1%	8.11e+10 99.9%	6.95e+09 8.6%	5.16e+09 99.6%	1.15e+08 2.2%	9.64e+09 99.9%		
loop at CycleTracking.cc: 110	1.31e+11 75.6%	1.81e+09 1.0%	6.01e+10 74.1%	8.68e+08 1.1%	5.12e+09 98.0%	9.24e+07 1.0%	9.54e+09 98.9%		
CollisionEvent(MonteCarlo)	3.26e+10 18.8%	3.65e+10 21.0%	5.44e+10 67.1%	7.01e+09 87.7%	3.02e+09 58.3%	5.13e+09 100.0%	6.43e+09 66.8%		
loop at CollisionEvent.cc: 67	6.59e+10 37.5%	1.62e+09 0.9%	2.92e+10 36.0%	7.18e+08 8.9%	3.51e+09 67.7%	1.59e+08 3.1%	5.94e+09 61.5%		
loop at CollisionEvent.cc: 71	6.13e+10 35.4%	3.86e+09 2.2%	2.76e+10 34.0%	1.65e+09 2.0%	3.24e+09 62.4%	2.09e+08 4.0%	5.59e+09 57.9%		
NuclearData::getR...	3.41e+10 19.7%	3.41e+10 19.7%	1.56e+10 19.2%	1.56e+10 19.2%	1.25e+09 24.2%	1.25e+09 24.2%	2.26e+09 23.4%		
NuclearData.cc: 253	1.09e+10 6.3%	1.09e+10 6.3%	5.12e+09 6.2%	5.12e+09 6.2%	4.77e+08 9.2%	4.77e+08 9.2%	6.18e+07 0.6%		
NuclearData.cc: 251	6.61e+09 3.8%	6.61e+09 3.8%	3.08e+09 3.8%	3.08e+09 3.8%	8.25e+07 1.6%	8.25e+07 1.6%	3.32e+08 3.4%		
NuclearData.cc: 252	1.82e+09 1.0%	1.82e+09 1.0%	8.27e+08 1.0%	8.27e+08 1.0%	2.33e+05 0.0%	2.33e+05 0.0%	2.47e+08 2.6%		
NuclearData.cc: 248	7.08e+08 0.4%	7.08e+08 0.4%	2.98e+08 0.4%	2.98e+08 0.4%	1.10e+08 2.1%	1.10e+08 2.1%	1.49e+08 1.5%		
MacroscopicCrossSection...	1.01e+10 5.8%	1.01e+10 5.8%	4.40e+09 5.4%	4.40e+09 5.4%	1.39e+09 26.9%	1.39e+09 26.9%	1.36e+09 14.1%		
loop at CollisionEvent.cc: 71	4.32e+09 2.5%	4.32e+09 2.5%	1.98e+09 2.4%	1.98e+09 2.4%	1.19e+05 0.0%	1.19e+05 0.0%	3.38e+08 3.5%		
MacroscopicCrossSection...	2.05e+09 1.2%	2.05e+09 1.2%	9.12e+08 1.1%	9.12e+08 1.1%			1.97e+08 2.0%		

Emerging Work: Large-scale LAMMPS



Work in Progress

- **GPU Enhancements**
 - Intel GPUs
 - Measurement support for Intel GPUs using OpenCL and Level 0
 - Fine-grain measurement using GTPin
 - Fine-grain attribution using binary analysis
 - AMD GPUs
 - Binary analysis and instrumentation for fine-grain measurement and attribution
 - NVIDIA GPUs
 - New support for NVIDIA inlining info - distribute our patches to the community
 - Reduce fine-grain measurement overhead with low-overhead PC sampling (CUDA 11.3)
- **Scalability**
 - finalizing new version of hpcprof-mpi with massive threading and sparse formats
- **User interface**
 - overhaul metric view to enhance performance and scalability
 - associate trace lines with metadata (node, GPU, MPI rank, GPU stream ...)
 - improve presentation of the many GPU metrics
- **Reliability and Completeness**