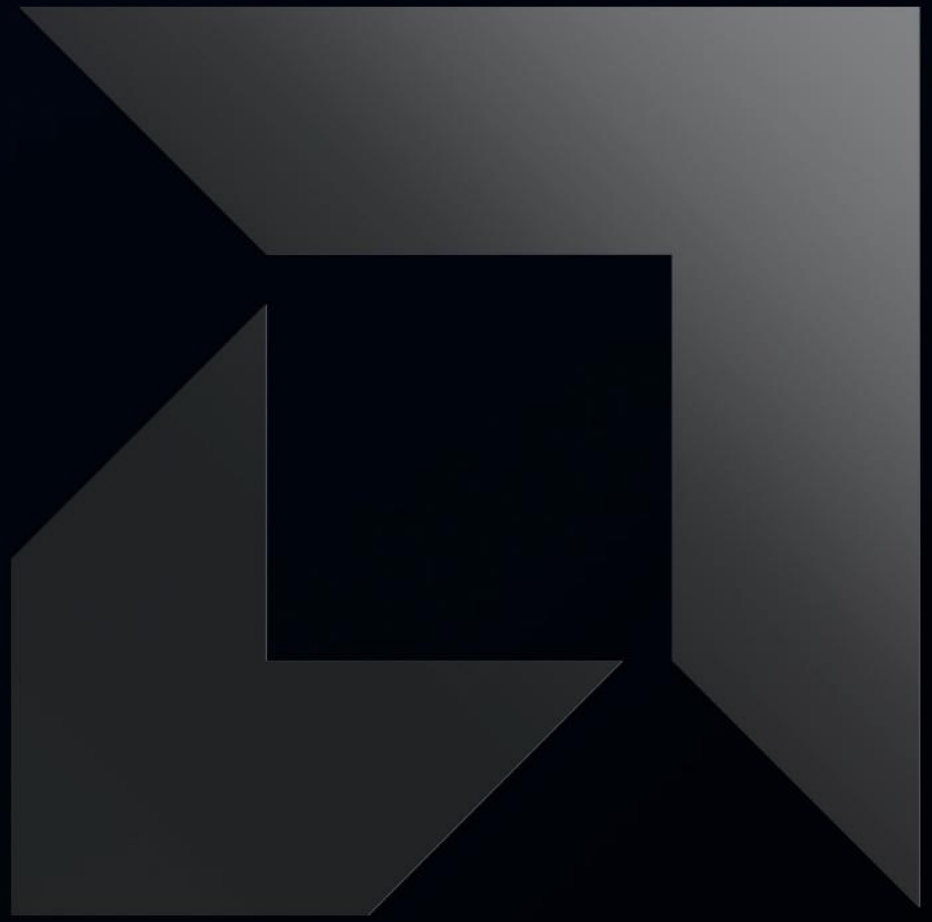


HPC Community Tools

Presenter: Giacomo Capodaglio
Oct 21-23, 2025
AMD @ Tsukuba University

AMD
together we advance_

HPC Community Tools

- TAU
  - https://www.cs.uoregon.edu/research/tau/home.php
- HPCToolkit
  - https://hpctoolkit.gitlab.io/
- Score-P
  - https://www.vi-hps.org/projects/score-p/overview/overview.html
- Likwid
  - https://docs.hpc.gwdg.de/software_stacks/applications/likwid/index.html
- Scalasca
  - https://scalasca.org/
- Grafana
  - https://grafana.com/
- Perfetto
  - https://ui.perfetto.dev/

TAU

TAU

- Tuning and Analysis Utilities, developed at University of Oregon
- Scalable and flexible performance analysis toolkit
- Automatic instrumentation through Program Database Toolkit (PDT) for routines, loops, I/O, memory, phases, etc.

Installing TAU

- Installed from the public mirror on GitHub
  - (currently using GIT_COMMIT="23a56e2a1a728e99ff03341c30f9d24892c5952b"):

```
git clone https://github.com/UO-OACISS/tau2.git
cd tau2
git checkout $GIT_COMMIT
wget http://tau.uoregon.edu/ext.tgz
tar zxf ext.tgz

 # in the latest version of TAU -roctrace and -rocprofiler are mutually exclusive and -roprofsdk should be used
ROCM_FLAGS="-rocm=${ROCM_PATH} -hip=${ROCM_PATH} -rocmsmi=${ROCM_PATH} -roctracer=${ROCM_PATH} -
rocprofiler=${ROCM_PATH}"
      result=`echo $ROCM_VERSION | awk '$1>6.1.2'` && echo $result
      if [[ "${result}" ]]; then # ROCM_VERSION >= 6.2
         ROCM_FLAGS="-rocm=${ROCM_PATH} -hip=${ROCM_PATH} -rocmsmi=${ROCM_PATH} -rocprofsdk=${ROCM_PATH}"
      fi
     # configure with: MPI OMPT OPENMP PDT ROCM
    ./configure -c++=amdclang++ -fortran=gfortran -cc=amdclang -prefix=${TAU_PATH} -zlib=download -otf=download
                -unwind=download -bfd=download  ${ROCM_FLAGS} -mpi -ompt -openmp -pdt=${PDT_PATH} -iowrapper
```

TAU creates a library for every configuration

How to use TAU

- There are mainly two approaches to use an application with TAU and GPUs
  - Use TAU Wrappers
    - For C: replace the compiler with tau_cc.sh
    - For C++: replace the compiler with tau_cxx.sh
    - For Fortran: replace the compiler with tau_f90.sh
  - Dynamic instrumentation, for example:
    - srun -n 2 -gpus 2 tau_exec –T rocm,rocprofiler -rocm ./test

TAU example in HPCTrainingExample repo

```
cd ${REPO_DIR}/HIP/jacobi


module load rocm
module load openmpi
module load tau

export TAU_PROFILE=${TAU_PROFILE}
export TAU_TRACE=${TAU_TRACE}

rm -rf profile.0*
rm -rf tautrace.0*
make clean
make

ROCM_VERSION=`cat ${ROCM_PATH}/.info/version | head -1 | cut -f1 -d'-' `
result=`echo ${ROCM_VERSION} | awk '$1>6.1.9'` && echo $result
if [[ "${result}" ]]; then
   mpirun -n 2 tau_exec -rocm -T rocm,rocprofsdk ./Jacobi_hip -g 2 1
else
   mpirun -n 2 tau_exec -T rocm,roctracer,rocprofiler ./Jacobi_hip -g 2 1
fi

ls
pprof
```

- Full code at: https://github.com/amd/HPCTrainingExamples/blob/main/tests/tau_exec_check.sh

Jacobi example (MPI+HIP) – Part 1/2

- Using tau_exec with ROCm options (MPI is included unless the *serial* is declared)

```
srun -n 2 --gpus 2 tau_exec  -T rocm,rocprofiler,roctracer -rocm ./Jacobi_hip -g 2 1
```

Partial output of *pprof* command:

```
NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0           7        1,942           1           1    1942701 .TAU application
 99.6          25        1,935           1       14020    1935195 taupreload_main
 32.5         630          630           1           1     630890 MPI_Init()
 31.6         613          613        1005           0        611 hipMemcpy
 18.8       0.059          365           1           3     365638 roctx: Jacobi_t::Jacobi_t::Top Level Init
 15.6       0.138          303           1           6     303459 roctx: CreateMesh::Init
 12.8         248          248           1           0     248057 hipMemset
  7.6         147          147           1           2     147517 MPI_Finalize()
  3.1          48           60           1          10      60800 roctx: InitializeData::Init
  2.8          54           54           2           0      27380 hipStreamCreate
  2.5          49           49        2000           0         25 hipStreamSynchronize
  2.1           1           40        1000        2000         40 roctx: Halo D2H::Halo Exchange
  1.7           2           33        1000        3000         34 roctx: MPI Exchange::Halo Exchange
  1.5          29           29        5002           0          6 hipLaunchKernel
  1.5          28           28        1000           0         29 MPI_Waitall()
  1.2           1           22        1000        2000         23 roctx: Halo H2D::Halo Exchange
  1.1           3           21        1003        3009         21 MPI_Allreduce()
  0.9          18           18        1005           0         18 MPI Collective Sync
  0.6          10           10        2000           0          5 hipEventRecord
  0.4           7            7        1000           0          7 hipMemcpyAsync
  0.2           4            4        1000           0          5 hipMemcpy2DAsync
  0.1           2            2        1000           0          3 MPI_Isend()
  0.1           2            2        1001           0          2 hipDeviceSynchronize
  0.1           1            1           7           0        217 hipMalloc
  0.1           1            1           1           3       1320 roctx: ApplyTopology::Init
  0.1        0.22            1           1           3       1182 MPI_Comm_dup()
  0.1           1            1        1000           0          1 hipEventElapsedTime
  0.1       0.983        0.983        2010           0          0 hipPointerGetAttributes
  0.0       0.637        0.637        1000           0          1 MPI_Irecv()
  0.0       0.525        0.525           3           0        175 hipHostMalloc
  0.0       0.131         0.31           1           3        310 MPI_Allgather()
  0.0       0.163        0.163           2           0         82 pthread_join
  0.0       0.012        0.012           2           0          6 hipGetDeviceCount
  0.0       0.007        0.007           2           0          4 hipEventCreate
  0.0       0.004        0.004           3           0          1 MPI_Barrier()
  0.0       0.004        0.004           1           0          4 hipSetDevice
  0.0       0.001        0.001           1           0          1 hipInit
  0.0           0            0           1           0          0 MPI_Comm_rank()
  0.0           0            0           1           0          0 MPI_Comm_size()
---------------------------------------------------------------------------------------
```

- On the top of the output you see Node 0 (means first MPI process) and the thread 0 is always on the CPU, the names are about TAU naming and not the real processes
- Then you can see the percentage of the time, the exclusive duration and other information for all the calls from the code, MPI, and HIP API

Jacobi example (MPI+HIP) – Part 2/2

- If you continue reading the output of the *pprof*, we can see the data transfers on thread 1 and 3, these are streams as the code uses stream
- Thread 2 is the actual process on the GPU, we can see the names of the kernels called, duration and other information

```
NODE 0;CONTEXT 0;THREAD 1:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0         877         890           1        1005    890197 .TAU application
  1.1           9           9           4           0      2454 CopyHostToDevice
  0.3           2           2        1001           0         3 CopyDeviceToHost

NODE 0;CONTEXT 0;THREAD 2:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0          94         773           1        5002    773552 .TAU application
 48.2         372         372        1001           0       372 KernelExecution NormKernel1(int, double, double, double const*, double*)
 21.8         168         168        1000           0       168 KernelExecution JacobiIterationKernel(int, double, double, double const*, double const*, double*, double*)
 16.5         127         127        1000           0       128 KernelExecution LocalLaplacianKernel(int, int, int, double, double, double const*, double*)
  1.0           8           8        1000           0         8 KernelExecution HaloLaplacianKernel(int, int, int, double, double, double const*, double const*, double*)
  0.3           2           2        1001           0         2 KernelExecution NormKernel2(int, double const*, double*)

NODE 0;CONTEXT 0;THREAD 3:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0         744         771           1        2000    771690 .TAU application
  3.1          23          23        1000           0        24 CopyDeviceToDevice2D
  0.5           3           3        1000           0         4 CopyHostToDevice
```

Jacobi visualization with Paraprof

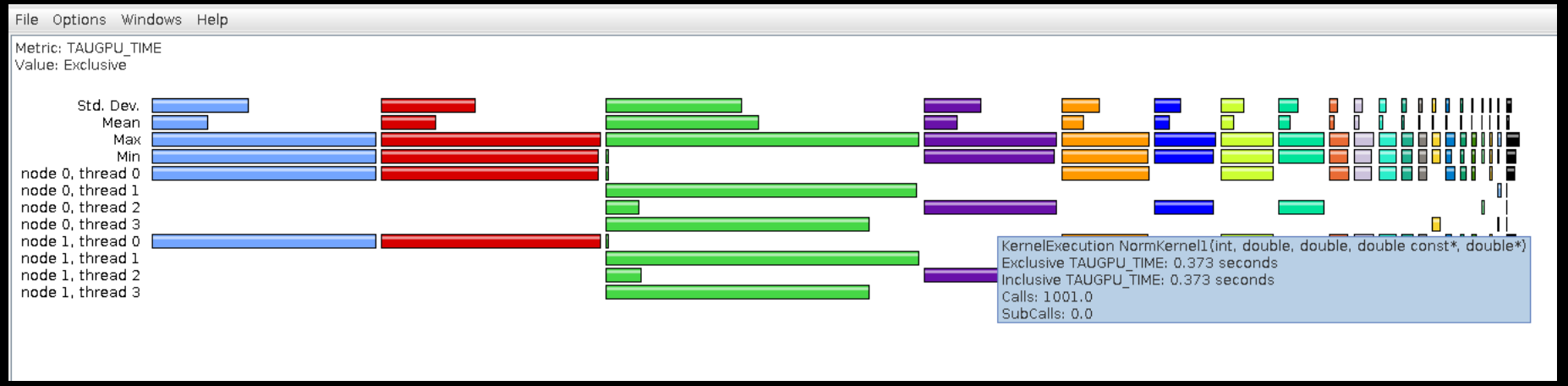
- Pack the profiling data to a file:

  paraprof --pack jacobi.ppk

- Then execute the paraprof tool

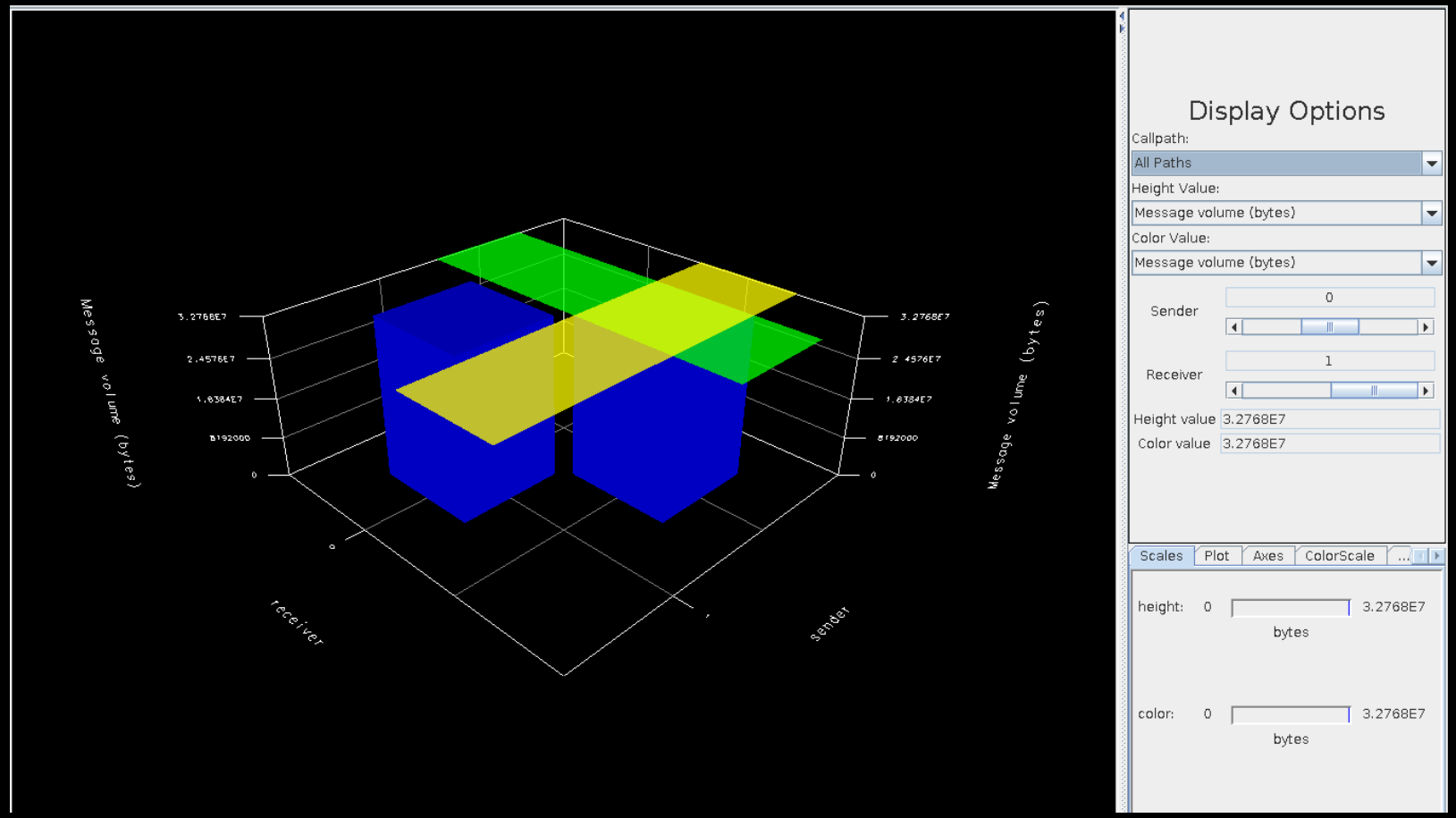
  paraprof jacobi.ppk

- Each color represents a different routine/API call, each row maps to the information on the left, mean/min/max/stdev values across all the threads, when you go the mouse over a color, it pop ups the name of the event as it is demonstrated.



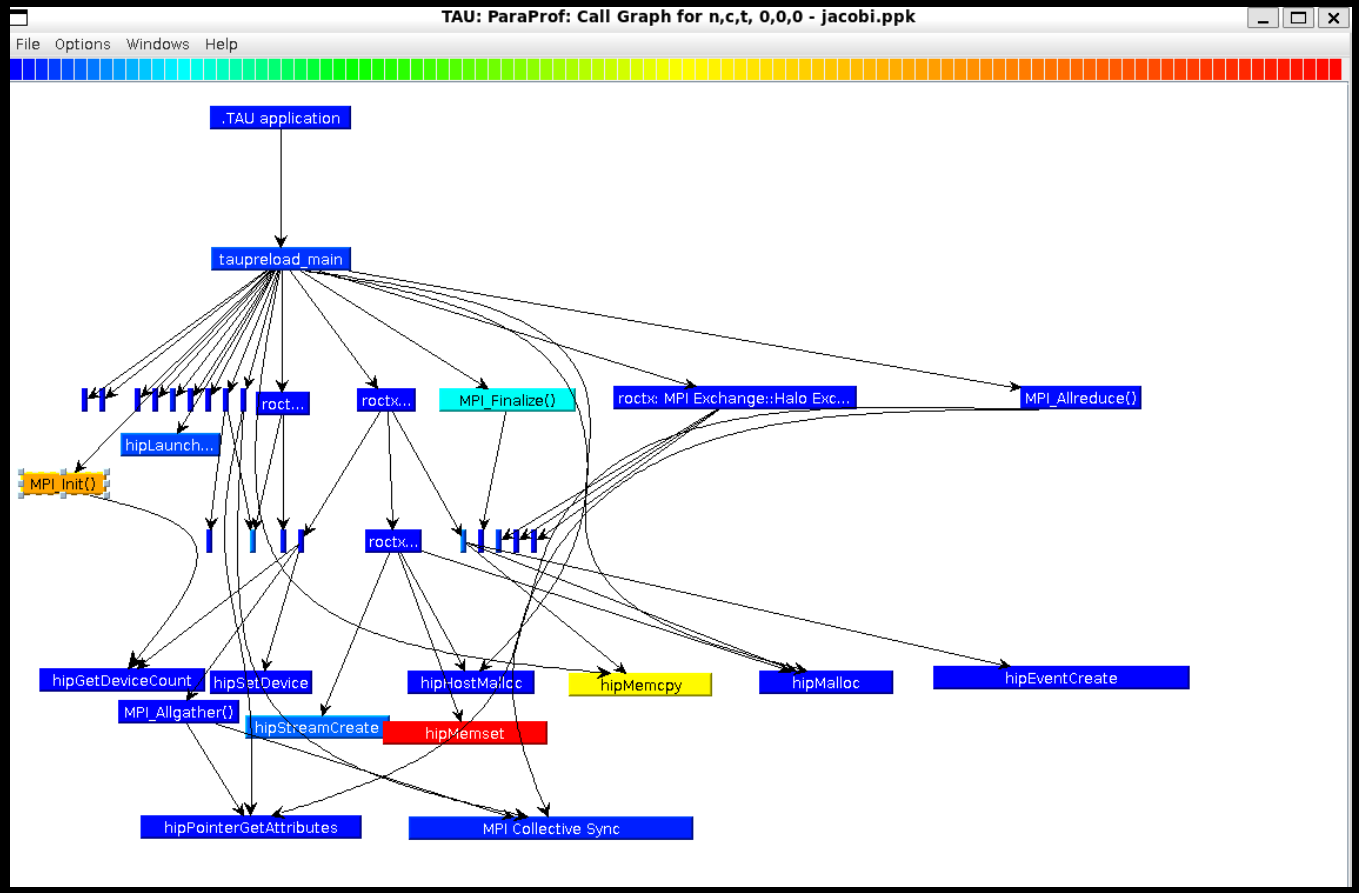
Jacobi - Communication Matrix

- Windows -> 3D Communication Matrix



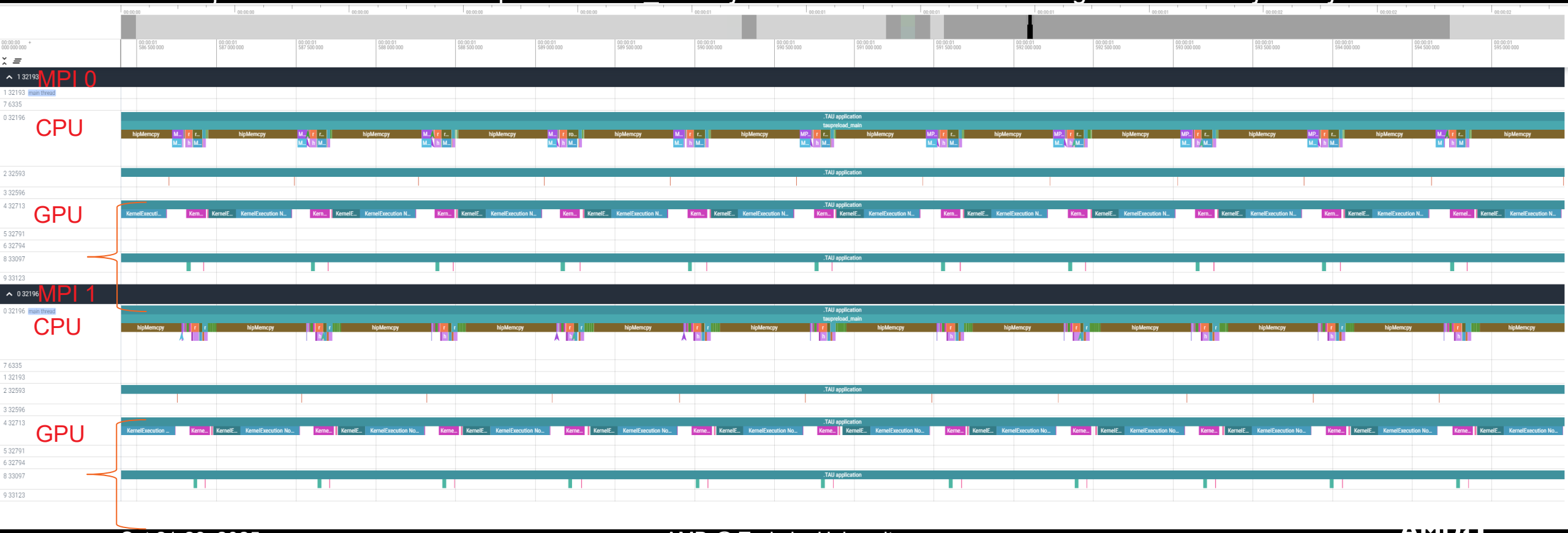
Jacobi - Callgraph

- Windows -> Thread -> Callgraph and select a thread, we chose the CPU thread (0)
  - The color indicates which calls take more time than others (blue color) as also you can see the call dependencies



Jacobi - Tracing

- Activate tracing: *export TAU_TRACE=1*
- Now we have one TRC file per thread and one EDF file per MPI process
- Merge all the files with the command: tau_treemerge.pl now there are two files named tau.trc and tau.edf
- Create a json file to visualize with perfetto: *tau_trace2json tau.trc tau.edf -chrome -ignoreatomic -o jacobi.json*



Porting an MPI CPU application

- Identify hot loops for CPU code using TAU with PDT

Name	Exclusive TIME ▽	Inclusive TIME	Calls	Child Calls
Loop: SETUPMOD::FELLI_MESHREF [{setupmod.f90} {2392,5}-{2534,10}]	6.072	8.979	896	395,065
MATHMOD::HYBRD [{mathmod.f90} {2707,12}] [THROTTLED]	2.045	2.547	254,745	1,366,010
Loop: MOMMOD::VIS3D [{mommod.f90} {1842,1}-{2654,16}]	0.983	1.02	10	1,330
Loop: MGCCMOD::SMOOTH [{mgccmod.f90} {2923,7}-{2943,12}]	0.561	0.561	5,771	0
MATHMOD::HYBRD1 [{mathmod.f90} {3255,12}] [THROTTLED]	0.349	2.897	289,683	254,745
MGCCMOD::RESTRICT [{mgccmod.f90} {4706,12}]	0.27	0.393	4,179	16,716
Loop: GEOMETRYMOD::GRDF3D [{geometrymod.f90} {966,5}-{1291,10}]	0.254	0.384	54	676,999
Loop: GODSPLIT3D::Z_VEL_CALC_CONS [{godmod.f90} {1625,1}-{1653,6}]	0.232	0.241	10	100,001
Loop: GODSPLIT3D::Y_VEL_CALC_CONS [{godmod.f90} {1162,1}-{1189,6}]	0.222	0.222	10	0
Loop: GODSPLIT3D::X_VEL_CALC_CONS [{godmod.f90} {712,1}-{740,6}]	0.222	0.222	10	0
Loop: MGCCMOD::SMOOTH [{mgccmod.f90} {2395,7}-{2429,12}]	0.18	0.18	199	0
Loop: MGCCMOD::PROLONGATE [{mgccmod.f90} {6393,3}-{6408,8}]	0.131	0.131	4,179	0
Loop: MGCCMOD::RESTRICT [{mgccmod.f90} {5380,9}-{5421,14}]	0.122	0.122	4,179	0
Loop: GODMOD::GODSPLIT3D [{godmod.f90} {349,1}-{632,13}]	0.102	1.36	10	120
MPI_File_write_at_all()	0.1	0.1	48	145
Loop: GODSPLIT3D::Z_VEL_CALC_CONS [{godmod.f90} {1678,1}-{1693,6}]	0.077	0.077	10	0
Loop: GODSPLIT3D::Z_VEL_CALC_CONS [{godmod.f90} {1658,1}-{1673,6}]	0.076	0.076	10	0
Loop: GODSPLIT3D::Y_VEL_CALC_CONS [{godmod.f90} {1213,1}-{1227,6}]	0.075	0.075	10	0
Loop: GODSPLIT3D::X_VEL_CALC_CONS [{godmod.f90} {764,1}-{778,6}]	0.075	0.075	10	0
Loop: GODSPLIT3D::Y_VEL_CALC_CONS [{godmod.f90} {1194,1}-{1208,6}]	0.075	0.075	10	0
Loop: GODSPLIT3D::X_VEL_CALC_CONS [{godmod.f90} {745,1}-{759,6}]	0.075	0.075	10	0
Loop: SETUPMOD::SETUP3D [{setupmod.f90} {407,3}-{865,8}]	0.074	9.116	1	109,377
Loop: SETUPMOD::SETUP3D [{setupmod.f90} {914,5}-{922,10}]	0.072	0.117	1	144,209
MATHMOD::R1UPDT [{mathmod.f90} {3797,12}] [THROTTLED]	0.057	0.105	100,001	400,004
BCMOD::BC_VECTOR [{bcmod.f90} {1813,12}]	0.054	0.065	1,294	9,078
GEOMETRYMOD::GRDF3D [{geometrymod.f90} {155,12}]	0.049	0.437	54	486
MATHMOD::QFORM [{mathmod.f90} {3391,12}] [THROTTLED]	0.045	0.08	100,001	300,003
FS3DMOD::FS3D_INIT [{fs3d.f90} {1684,12}]	0.043	9.519	1	40
Loop: MGCCMOD::DISCRETIZE [{mgccmod.f90} {7158,3}-{7178,8}]	0.042	0.042	21	0
MPI_File_write_at()	0.037	0.037	138	0
FS3DMOD::EULER_EXPL [{fs3d.f90} {8154,12}]	0.034	4.547	10	170
MATHMOD::QRFAC [{mathmod.f90} {3492,12}] [THROTTLED]	0.034	0.083	100,001	200,002
MATHMOD::DOGLEG [{mathmod.f90} {2267,12}] [THROTTLED]	0.034	0.051	100,001	164,917
MATHMOD::R1MPYQ [{mathmod.f90} {3686,12}] [THROTTLED]	0.033	0.053	100,001	200,002
GEOMETRYMOD::INTERFACE_HEIGHT_TOP [{geometrymod.f90} {4730,12}] [THROTTLED]	0.033	0.056	100,001	200,002
SETUPMOD::FELLI_MESHREF [{setupmod.f90} {2252,12}] [THROTTLED]	0.029	0.038	100,001	100,001
Loop: MATHMOD::FDJAC1 [{mathmod.f90} {2639,6}-{2658,11}] [THROTTLED]	0.028	0.032	100,001	41,460
Loop: SURFACETENSIONMOD::STEN3D [{surfacetensionmod.f90} {581,1}-{665,6}]	0.027	0.027	11	0
Loop: MATHMOD::QRFAC [{mathmod.f90} {3611,3}-{3682,8}] [THROTTLED]	0.025	0.026	100,001	11,124
.TAU application	0.024	14.322	1	1
Loop: MGCCMOD::SMOOTH [{mgccmod.f90} {2904,7}-{2919,12}]	0.024	0.024	5,771	0
Loop: MGCCMOD::SMOOTH [{mgccmod.f90} {3011,7}-{3027,12}]	0.023	0.023	5,771	0
SETUPMOD::VORBELEGUNGSFUNKTIONEN [{setupmod.f90} {5382,12}] [THROTTLED]	0.022	0.061	100,001	100,001
MATHMOD::FDJAC1 [{mathmod.f90} {2528,12}] [THROTTLED]	0.022	0.054	100,001	100,001
MATHMOD::RGAUSS [{mathmod.f90} {106,30}] [THROTTLED]	0.021	0.042	100,001	100,001
Loop: MATHMOD::QRFAC [{mathmod.f90} {3594,3}-{3596,8}] [THROTTLED]	0.021	0.022	100,001	11,124
Loop: MGCCMOD::RAP [{mgccmod.f90} {6609,3}-{6650,8}]	0.021	0.021	126	0
MGCCMOD::BC_ALL [{mgccmod.f90} {4443,12}]	0.021	0.021	199	199
MPI_File_open()	0.02	0.02	10	0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Fortran+OMP offloading (Nekbone)

- Execute the application and declare to tau_exec to profile ROCm events (MPI is activated by default, otherwise define SERIAL)
- srun –n1 tau_exec -T rocm,rocprofiler,roctracer -rocm ./nekbone data

File Options Windows Help

Metric: TAUGPU_TIME
Sorted By: Exclusive
Units: seconds

%Total Time	Exclusive	Inclusive	#Calls	#Child Calls	Inclusive/Call	Name
100.0	0.687	0.993	1	2614	0.993	.TAU application
7.7	0.076	0.076	200	0	3.8E-4	__omp_offloading_25c66b0a_c601f135_ax_acc__l461.kd
7.2	0.072	0.072	602	0	1.2E-4	__omp_offloading_25c66b0a_c601f1c1_glsc3_acc__l1447.kd
7.1	0.071	0.071	200	0	3.5E-4	__omp_offloading_25c66b0a_c601f135_ax_acc__l496.kd
3.8	0.038	0.038	600	0	6.3E-5	__omp_offloading_25c66b0a_c601f1c1_add2s2_acc__l1483.kd
1.3	0.012	0.012	200	0	6.2E-5	__omp_offloading_25c66b0a_c601f1c1_add2s1_acc__l1467.kd
1.0	0.01	0.01	200	0	4.9E-5	__omp_offloading_25c66b0a_c601f1cd_dssum_acc__l458.kd
0.9	0.009	0.009	200	0	4.7E-5	__omp_offloading_25c66b0a_c601f1c1_copy_acc__l1425.kd
0.9	0.009	0.009	200	0	4.5E-5	__omp_offloading_25c66b0a_c601f1cd_dssum_acc__l486.kd
0.8	0.008	0.008	200	0	3.8E-5	__omp_offloading_25c66b0a_c601f135_maskit_acc__l655.kd
0.0	1.6E-4	1.6E-4	4	0	3.9E-5	__omp_offloading_25c66b0a_c601f17d_gather_double_add_l148.kd
0.0	1.3E-4	1.3E-4	4	0	3.1E-5	__omp_offloading_25c66b0a_c601f17d_scatter_double_l148.kd
0.0	1.0E-4	1.0E-4	2	0	5.0E-5	__omp_offloading_25c66b0a_c601f1c1_rzero_acc__l1393.kd
0.0	6.0E-6	6.0E-6	2	0	3.0E-6	__omp_offloading_25c66b0a_c601f17d_init_double_l148.kd

HPCToolkit

HPCToolkit: Overview

- Not an AMD profiler - developed for 20+ years, mainly at Rice University
- HPCToolkit - suite of tools for tracing, profiling and analyzing parallel programs

- Combine sampling data with a static analysis of the program structure for loops and inline functions
- Present top-down, bottom-up and flat views of calling context tree and time-sequence trace view

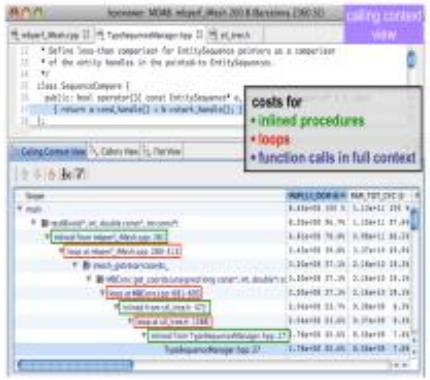
- Low overhead, easy to use, interactive analysis with hpcviewer
- No instrumentation or recompilation needed, as long as “-g” compiler flag is used
- Supports threads (pthreads, OpenMP®), MPI, hybrid (MPI+threads), and GPUs (AMD, Intel®, NVIDIA)

Installing HPCToolkit

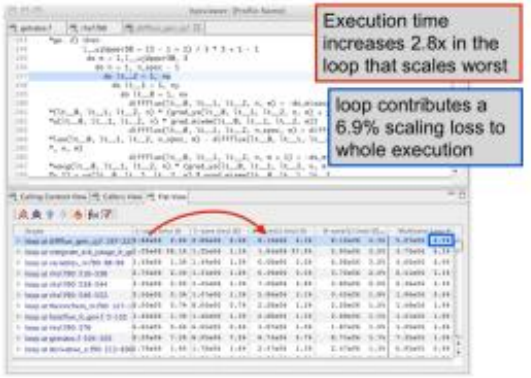
- **HPCToolkit** – integrated suite of tools for measurement and analysis of program performance on computers ranging from multicore desktop systems to GPU-accelerated supercomputers (Rice University)
- Installation from Gitlab repo using **Meson**:

```
pipx install 'meson>=1.3.2'
export PATH=$HOME/.local/bin:$PATH
git clone https://gitlab.com/hpctoolkit/hpctoolkit.git
git checkout 0bfc9cd58da73e3e5b973d754677ebfe2b0da55d
cd hpctoolkit
export CMAKE_PREFIX_PATH=$ROCM_PATH:$CMAKE_PREFIX_PATH
meson setup -Drocm=enabled --prefix=${HPCTOOLKIT_PATH}
            --libdir=${HPCTOOLKIT_PATH}/lib build
cd  build
meson compile
meson install
```

HPCToolkit Capabilities at a Glance

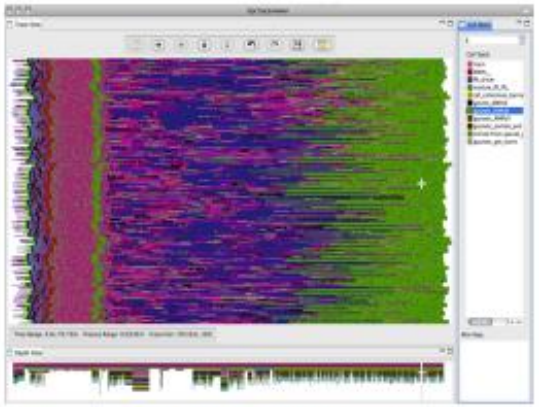


Attribute Costs to Code

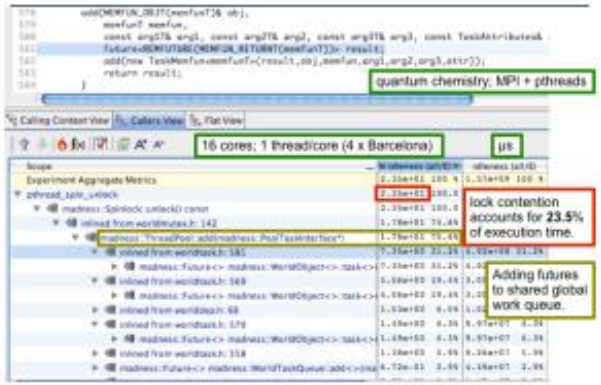


Pinpoint & Quantify Scaling Bottlenecks

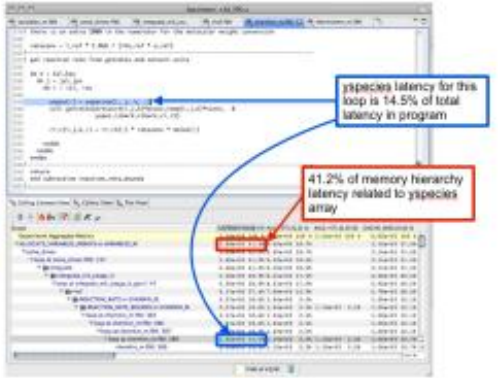
Assess Imbalance and Variability



Analyze Behavior over Time



Shift Blame from Symptoms to Causes

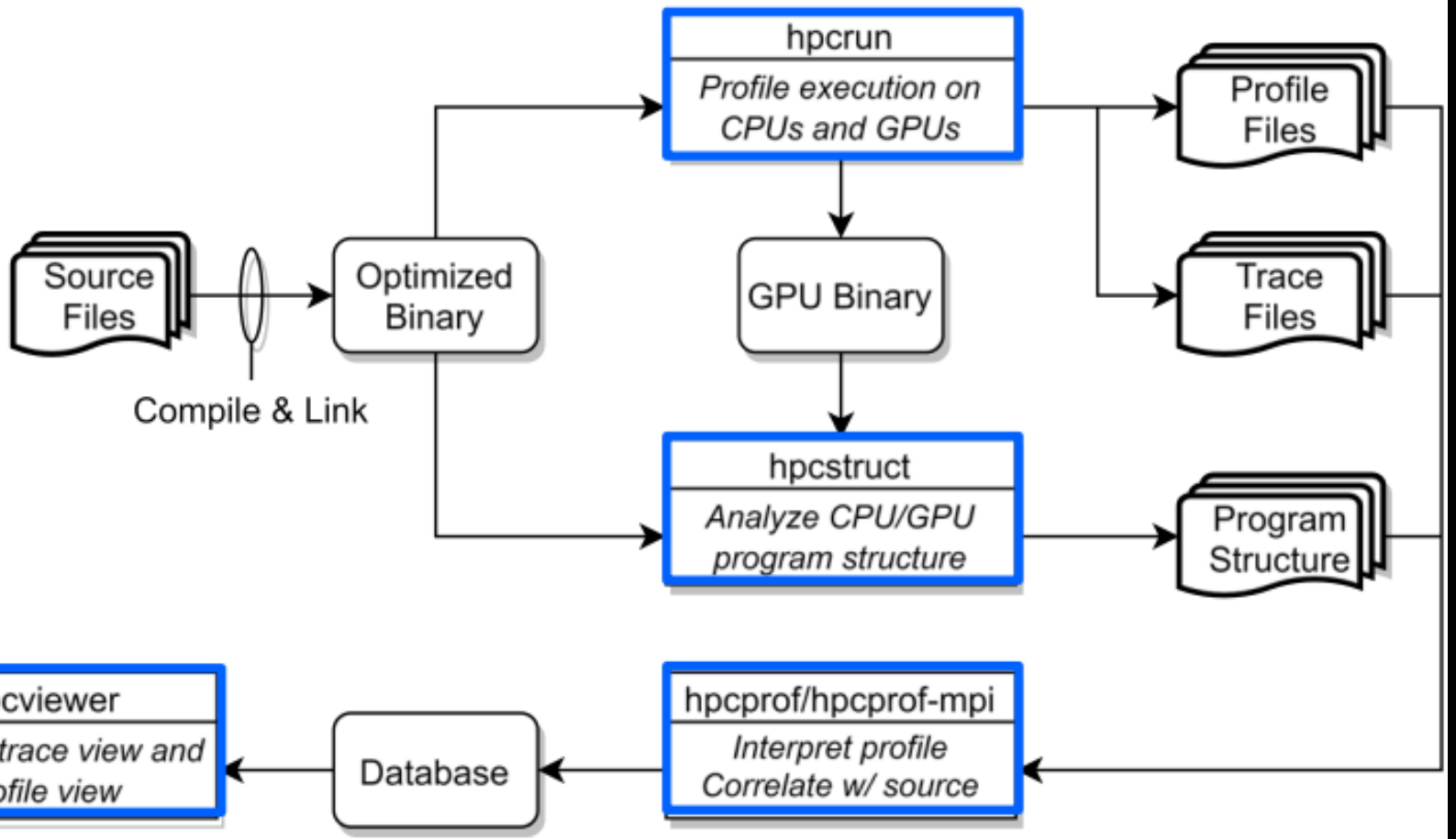


Associate Costs with Data

RICE

hpctoolkit.org

HPCToolkit's Workflow for GPU-accelerated Applications



HPCToolkit examples in HPCTrainingExample repo

hpcrun

```
# This test checks that hpcrun
# runs without errors

module load rocm

module load hpctoolkit
REPO_DIR="$(dirname "$(dirname "$(readlink -fm "$0")")")"
pushd ${REPO_DIR}/HIP/Stream_Overlap/0-Orig/
rm -rf build_for_test
mkdir build_for_test; cd build_for_test
cmake ../
make -j

hpcrun -e CPUTIME -e gpu=amd -t ./compute_comm_overlap 2
ls hpctoolkit-compute_comm_overlap-measurements*
```

hpcstruct

```
# This test checks that hpcstruct
# runs without errors

module load rocm

module load hpctoolkit
REPO_DIR="$(dirname "$(dirname "$(readlink -fm "$0")")")"
pushd ${REPO_DIR}/HIP/Stream_Overlap/0-Orig/
rm -rf build_for_test
mkdir build_for_test; cd build_for_test
cmake ../
make -j

hpcrun -e CPUTIME -e gpu=amd -t ./compute_comm_overlap 2
hpcstruct hpctoolkit-compute_comm_overlap-measurements*
ls hpctoolkit-compute_comm_overlap-measurements*
```

hpcprof

```
# This test checks that hpcprof
# runs without errors

module load rocm

module load hpctoolkit
REPO_DIR="$(dirname "$(dirname "$(readlink -fm "$0")")")"
pushd ${REPO_DIR}/HIP/Stream_Overlap/0-Orig/
rm -rf build_for_test
mkdir build_for_test; cd build_for_test
cmake ../
make -j

hpcrun -e CPUTIME -e gpu=amd -t ./compute_comm_overlap 2
hpcstruct hpctoolkit-compute_comm_overlap-measurements*
hpcprof hpctoolkit-compute_comm_overlap-measurements*
ls hpctoolkit-compute_comm_overlap-database*
```

- Full code at: https://github.com/amd/HPCTrainingExamples/blob/main/tests/

DISCLAIMERS AND ATTRIBUTIONS


AMD