Programmatic Analysis of Large-Scale Performance Data

Dragana Grbic

August 12, 2024

LLNL-PRES-868028

Lawrence Livermore National Laboratory

Measuring and Analyzing Applications with HPCToolkit

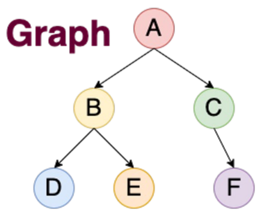
- HPCToolkit collects **fine-grained measurement data**
  - typically entire program executions
  - CPU and GPU performance
  - CCTs contain detailed information about program's execution
- Measuring does not require a lot of manual work
  - users don't have to annotate code or specify regions to measure
- Analyzing can be difficult and time consuming
  - generated databases can be huge: long executions, large-scale parallelism
  - manual inspection using GUI tool can be tedious because of the overwhelming detail
  - users need support for **automated and programmatic analysis**

Approaches for Programmatic Analysis

- Using existing tools for automated analysis
  - **Hatchet** for analyzing single application runs
  - **Thicket** for analyzing multiple application runs
  - techniques for automatically reducing large HPCToolkit's calling context trees
- New API for analyzing large-scale HPCToolkit data
  - **selective read of slices of performance data** from persistent storage
  - users can use **query expressions** to extract slices of performance data
  - more scalable and efficient for analyzing large-scale executions

Using Hatchet and Thicket to Analyze HPCToolkit Data

Hatchet and Thicket Performance Profiles



DataFrame

node	profile	time	time (i)
A	1	3.56	12.06
A	2	4.11	12.97
B	1	1.38	7.71
B	2	2.67	5.32
C	1	2.12	4.35
C	2	1.98	3.54
D	1	4.78	4.78
E	1	1.55	1.55
E	2	2.65	2.65
F	2	1.56	1.56

exist only in one profile

Metadata

profile	user	date
1	Bob	May 5
2	Mike	June 1

Thicket profile

DataFrame

node	time	time (i)
A	3.56	12.06
B	1.38	7.71
C	2.12	4.35
D	4.78	4.78
E	1.55	1.55
F	2.23	2.23

Hatchet Profile

- Programmatic analysis of performance data generated by different tools
  - **Hatchet** profile modeled with a **Graph** object and **DataFrame** table
  - **Thicket** profile modeled with a **Graph** object, **DataFrame** table, and **Metadata** table

Large Calling Context Trees

- HPCToolkit's calling context trees can contain many nodes
  - nodes with little cost incurred
  - implementation details of library functions
  - nodes with no line mapping information (compiled without **-g** option)
- Hatchet and Thicket were not designed to handle data as large as HPCToolkit's
  - calling context trees are huge and difficult to interpret and visualize
  - importing multiple application runs into Thicket is slow as unifying calling context trees is costly for large trees

MPI_Finalize Subtree

```
0.166 MPI_Finalize
└─ 0.166 line [libmonitor.so.0.0.0]:0
   └─ 0.166 PMPI_Finalize
      ├─ 0.166 line [libmpi.so.12.1.1]:0
      │  ├─ 0.166 MPID_Finalize
      │  │  └─ 0.166 line [libmpi.so.12.1.1]:0
      │  │     ├─ 0.006 MPIDI_RMA_finalize
      │  │     │  └─ 0.006 line [libmpi.so.12.1.1]:0
      │  │     │     ├─ 0.006 __GI___munmap
      │  │     │     │  └─ 0.006 line [libc-2.28.so]:0
      │  │     │     └─ 0.006 __GI___munmap
      │  │     │        └─ 0.006 line [libc-2.28.so]:0
      │  │     ├─ 0.006 MPIR_Comm_release_always
      │  │     │  └─ 0.006 line [libmpi.so.12.1.1]:0
      │  │     │     └─ 0.006 MPIR_Comm_delete_internal
      │  │     │        └─ 0.006 line [libmpi.so.12.1.1]:0
      │  │     │           └─ 0.006 free_2level_comm
      │  │     │              └─ 0.006 line [libmpi.so.12.1.1]:0
      │  │     │                 └─ 0.006 mv2_shm_coll_cleanup
      │  │     │                    └─ 0.006 line [libmpi.so.12.1.1]:0
      │  │     │                       ├─ 0.006 __GI___close
      │  │     │                       │  └─ 0.006 line [libc-2.28.so]:0
      │  │     │                       ├─ 0.006 __GI___munmap
      │  │     │                       │  └─ 0.006 line [libc-2.28.so]:0
      │  │     │                       └─ 0.006 _dl_runtime_resolve_xsavec
      │  │     │                          └─ 0.006 line /usr/src/debug/glibc-2.28-236.el8_9.13.x86_64/elf/../sysdeps/x86_64/dl-trampoline.h:126
      │  │     │                             └─ 0.006 _dl_fixup
      │  │     │                                └─ 0.006 line /usr/src/debug/glibc-2.28-236.el8_9.13.x86_64/elf/dl-runtime.c:94
      │  │     │                                   └─ 0.006 _dl_lookup_symbol_x
      │  │     │                                      └─ 0.006 loop /usr/src/debug/glibc-2.28-236.el8_9.13.x86_64/elf/dl-lookup.c:871
      │  │     │                                         └─ 0.006 line /usr/src/debug/glibc-2.28-236.el8_9.13.x86_64/elf/dl-lookup.c:872
      │  │     │                                            └─ 0.006 do_lookup_x
      │  │     │                                               └─ 0.006 loop /usr/src/debug/glibc-2.28-236.el8_9.13.x86_64/elf/dl-lookup.c:451
      │  │     │                                                  └─ 0.006 line /usr/src/debug/glibc-2.28-236.el8_9.13.x86_64/elf/dl-lookup.c:373
```

no line mapping information (→ line [libmonitor.so.0.0.0]:0)

little cost incurred (→ 0.006 MPIR_Comm_release_always)

the entire subtree contains around 307 nodes

for each MPI routine call, HPCToolkit will record a subtree with implementation details

MPI_Waitall Subtree

```
0.024 PMPI_Waitall
└─ 0.024 line [libmpi.so.12.1.1]:0
   └─ 0.024 MPIR_Waitall_impl
      ├─ 0.006 line [libmpi.so.12.1.1]:0
      │  └─ 0.006 MPIDI_CH3_Progress_start
      │     └─ 0.006 line [libmpi.so.12.1.1]:0
      │        └─ 0.006 psm2_poll
      │           └─ 0.006 loop [libpsm2.so.2.2]:0
      │              └─ 0.006 line [libpsm2.so.2.2]:0
      │                 ├─ 0.006 /usr/lib64/libpsm2.so.2.2:160903
      │                 │  └─ 0.006 /usr/lib64/libpsm2.so.2.2:173073
      │                 │     └─ 0.006 /usr/lib64/libpsm2.so.2.2:262100
      │                 │        └─ 0.006 /usr/lib64/libpsm2.so.2.2:196624
      │                 └─ 0.006 /usr/lib64/libpsm2.so.2.2:48659
      │                    └─ 0.006 /usr/lib64/libpsm2.so.2.2:24223
      │                       ├─ 0.006 /usr/lib64/libpsm2.so.2.2:52176
      │                       └─ 0.006 /usr/lib64/libpsm2.so.2.2:52638
      │                          └─ 0.006 /usr/lib64/libpsm2.so.2.2:134776
      │                             └─ 0.006 /usr/lib64/libc-2.28.so:847475
      └─ 0.024 loop [libmpi.so.12.1.1]:0
         └─ 0.024 loop [libmpi.so.12.1.1]:0
            └─ 0.024 line [libmpi.so.12.1.1]:0
               └─ 0.024 psm_progress_wait
                  ├─ 0.006 line [libmpi.so.12.1.1]:0
                  │  ├─ 0.006 psm2_mq_test2
                  │  │  └─ 0.006 loop [libpsm2.so.2.2]:0
                  │  │     └─ 0.006 line [libpsm2.so.2.2]:0
                  │  └─ 0.006 psm_process_completion
                  │     └─ 0.006 line [libmpi.so.12.1.1]:0
                  │        └─ 0.006 MPID_Request_release
                  │           └─ 0.006 line [libmpi.so.12.1.1]:0
                  └─ 0.024 loop [libmpi.so.12.1.1]:0
                     ├─ 0.024 line [libmpi.so.12.1.1]:0
                     │  └─ 0.024 psm2_mq_ipeek2
                     │     ├─ 0.024 line [libpsm2.so.2.2]:0
                     │     │  ├─ 0.006 line [libpsm2.so.2.2]:0
                     │     │  └─ 0.018 loop [libpsm2.so.2.2]:0
                     │     │     └─ 0.018 line [libpsm2.so.2.2]:0
                     │     │        ├─ 0.012 /usr/lib64/libpsm2.so.2.2:160778
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:160798
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:160943
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:160949
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:161010
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:48516
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:48634
                     │     │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:48659
                     │     │        │  └─ 0.006 /usr/lib64/libpsm2.so.2.2:24223
                     │     │        │     └─ 0.006 /usr/lib64/libpsm2.so.2.2:52638
                     │     │        │        ├─ 0.006 /usr/lib64/libpsm2.so.2.2:134776
                     │     │        │        │  ├─ 0.006 /usr/lib64/libc-2.28.so:847475
                     │     │        │        │  └─ 0.006 __mempcpy_chk_avx_unaligned
                     │     │        │        │     ├─ 0.006 line [libc-2.28.so]:0
                     │     │        │        │     └─ 0.006 loop [libc-2.28.so]:0
                     │     │        │        │        └─ 0.006 line [libc-2.28.so]:0
                     │     │        │        └─ 0.006 /usr/lib64/libpsm2.so.2.2:134893
                     │     │        │           └─ 0.006 /usr/lib64/libpsm2.so.2.2:131372
                     │     │        │              └─ 0.006 /usr/lib64/libc-2.28.so:847475
                     │     │        └─ 0.006 /usr/lib64/libpsm2.so.2.2:48708
                     │     └─ 0.012 loop [libpsm2.so.2.2]:0
                     │        └─ 0.012 line [libpsm2.so.2.2]:0
                     └─ 0.006 loop [libmpi.so.12.1.1]:0
                        └─ 0.006 line [libmpi.so.12.1.1]:0
```

great fraction of nodes are missing line mapping information (system library compiled without -g option)

OpenMP Subtrees

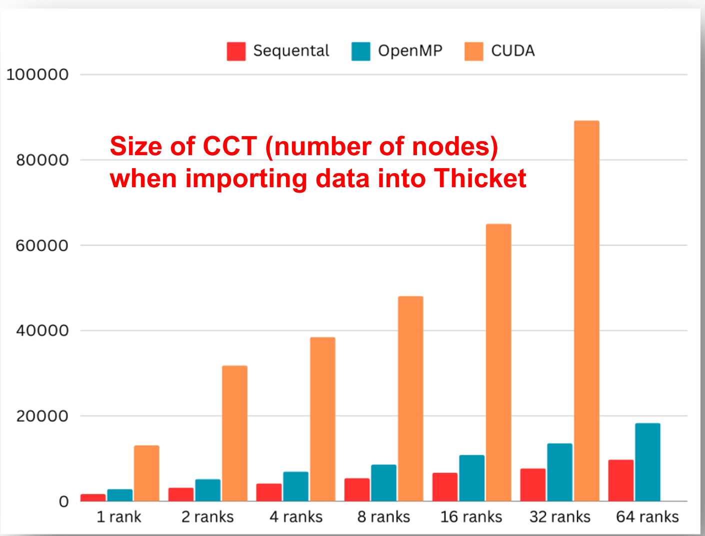
```
├─ 0.006 <omp barrier wait>
├─ 0.006 __kmpc_fork_call
│  ├─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_csupport.cpp:358
│  │  └─ 0.006 __kmp_fork_call
│  │     ├─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:2144
│  │     │  └─ 0.006 ompt_parallel_begin
│  │     │     └─ 0.006 line [libhpcrun.so]:0
│  │     │        └─ 0.006 hpcrun_get_thread_data_specific_avail
│  │     │           └─ 0.006 line [libhpcrun.so]:0
│  │     └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:2443
│  │        └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:2450
│  └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_csupport.cpp:371
│     └─ 0.006 __kmp_join_call
│        └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:2681
│           └─ 0.006 __kmp_internal_join
│              └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:8231
│                 └─ 0.006 __kmp_join_barrier
│                    └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:2332
│                       └─ 0.006 _ZN17_INTERNAL92a63c0c26__kmp_hyper_barrier_gatherE12barrier_typeP8kmp_infoiiPFvPvS3_ES3_
│                          └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1064
│                             └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1064
│                                └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1064
│                                   └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1110
│                                      └─ 0.006 kmp_flag_64
│                                         └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:890
│                                            └─ 0.006 kmp_flag_native
│                                               └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:179
│                                                  └─ 0.006 kmp_flag_native
│                                                     └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:179
│                                                        └─ 0.006 kmp_flag
│                                                           └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:139
│                                                              └─ 0.006 kmp_flag
│                                                                 └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:553
│                                                                    ├─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:553
│                                                                    │  └─ 0.006 kmp_flag_native
│                                                                    │     └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:205
│                                                                    └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:596
```

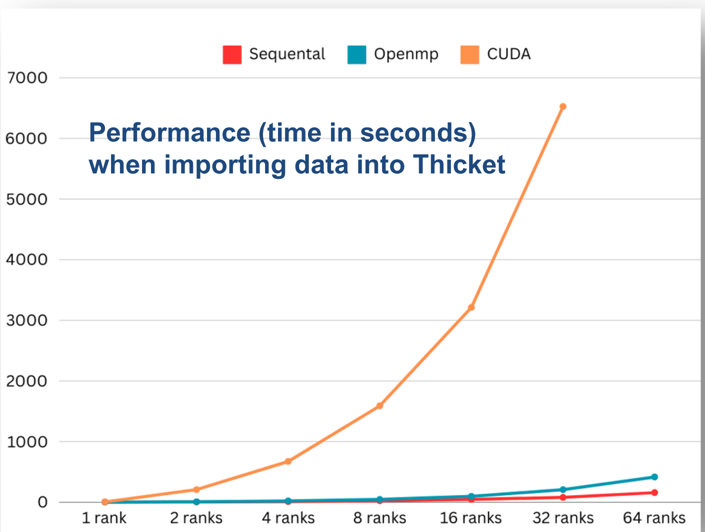
```
├─ 0.006 <omp barrier wait>
├─ 0.006 __kmpc_fork_call
│  ├─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_csupport.cpp:358
│  │  └─ 0.006 __kmp_fork_call
│  │     └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:2226
│  │        └─ 0.006 __kmp_serial_fork_call
│  │           └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:1802
│  │              └─ 0.006 __kmpc_serialized_parallel
│  │                 └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_csupport.cpp:582
│  │                    └─ 0.006 __kmp_serialized_parallel
│  │                       └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:1422
│  │                          └─ 0.006 ___kmp_allocate
│  │                             └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_alloc.cpp:2054
│  │                                └─ 0.006 scalable_aligned_malloc
│  │                                   └─ 0.006 line /nfs/site/proj/openmp/promo/tbb/oneTBB-20210907/src/tbbmalloc/frontend.cpp:3109
│  │                                      └─ 0.006 allocateAligned
│  │                                         └─ 0.006 line /nfs/site/proj/openmp/promo/tbb/oneTBB-20210907/src/tbbmalloc/frontend.cpp:2374
│  └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_csupport.cpp:371
│     └─ 0.006 __kmp_join_call
│        └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:2681
│           └─ 0.006 __kmp_internal_join
│              └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_runtime.cpp:8231
│                 └─ 0.006 __kmp_join_barrier
│                    └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:2332
│                       └─ 0.006 _ZN17_INTERNAL92a63c0c26__kmp_hyper_barrier_gatherE12barrier_typeP8kmp_infoiiPFvPvS3_ES3_
│                          └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1064
│                             └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1064
│                                └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1064
│                                   └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_barrier.cpp:1110
│                                      └─ 0.006 kmp_flag_64
│                                         └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:890
│                                            └─ 0.006 kmp_flag_native
│                                               └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:179
│                                                  └─ 0.006 kmp_flag_native
│                                                     └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:179
│                                                        └─ 0.006 kmp_flag
│                                                           └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:139
│                                                              └─ 0.006 kmp_flag
│                                                                 └─ 0.006 loop /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:553
│                                                                    └─ 0.006 line /nfs/site/proj/openmp/promo/20220128/tmp/lin_32e-rtl_int_5_nor_dyn.rel.c0.s0.t1..h1.w1-fxilab153/../../src/kmp_wait_release.h:596
```

cudaDeviceSynchronize Subtree

Testing Thicket with AMG Benchmark

	1 rank	2 ranks	4 ranks	8 ranks	16 ranks	32 ranks	64 ranks
Sequential	1689nodes 0.71s	3170nodes 3.99s	4194nodes 9.61s	5392nodes 24.58s	6663nodes 47.21s	7713nodes 79.25s	9749nodes 158.08s
OpenMP	2849nodes 1.19s	5195nodes 8.72s	6961nodes 22.65s	8577nodes 46.14s	10829nodes 97.80s	13553nodes 208.04s	18336nodes 416.86s
CUDA	13063nodes 5.61s	31828nodes 207.85s	38453nodes 672.13s	48045nodes 1589.72s	65030nodes 3213.55s	89239nodes 6525.85s	





Almost two hours

Data Reduction

- Heuristic for automatically reducing the size of large calling context trees before importing into analysis model
  - automatically detect and remove specific nodes from the tree and optionally their entire subtree
  - users can choose which heuristics they want to enable when reading the data
- Two performance improvements
  - the reader does not have to parse subtree of a node declared uninteresting by a specific heuristic
  - performing union operation of Hatchet profiles inside Thicket is faster for smaller trees

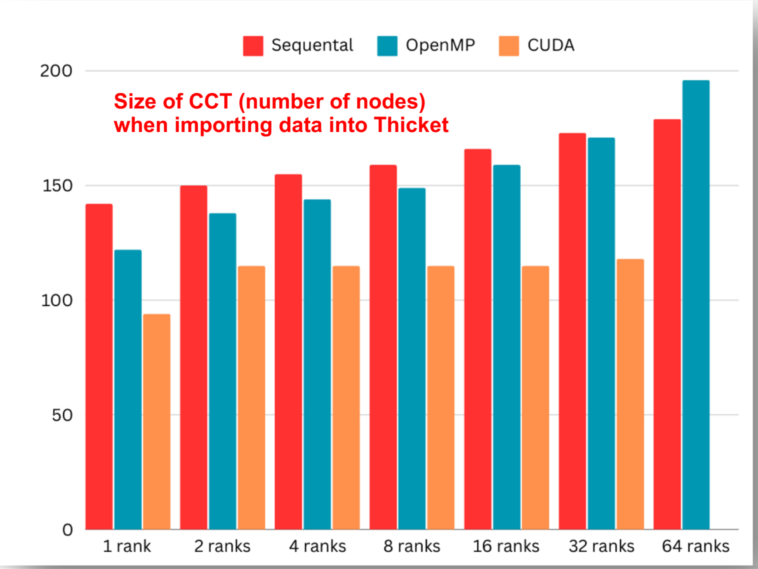
Reduction Heuristics

- removing nodes with inclusive time less than 1% of application time
- removing implementation details of library functions (MPI, OpenMP, CUDA)
- removing nodes with no line mapping information (system library routines)
- removing function call line nodes (each function call is recorded with a source line node that represents the place of the call and function itself)

Improvement: Number of CCT nodes

	1 rank	2 ranks	4 ranks	8 ranks	16 ranks	32 ranks	64 ranks
Sequential	142 out of 1689 (8%)	150 out of 3170 (5%)	155 out of 4194 (4%)	159 out of 5392 (3%)	166 out of 6663 (2%)	173 out of 7713 (2%)	179 out of 9749 (2%)
OpenMP	122 out of 2849 (4%)	138 out of 5195 (3%)	144 out of 6961 (2%)	149 out of 8577 (2%)	159 out of 10829 (1%)	171 out of 13553 (1%)	196 out of 18336 (1%)
CUDA	94 out of 13063 (0.7%)	115 out of 31828 (0.4%)	115 out of 38453 (0.3%)	115 out of 48045 (0.2%)	115 out of 65030 (0.2%)	118 out of 89239 (0.1%)	

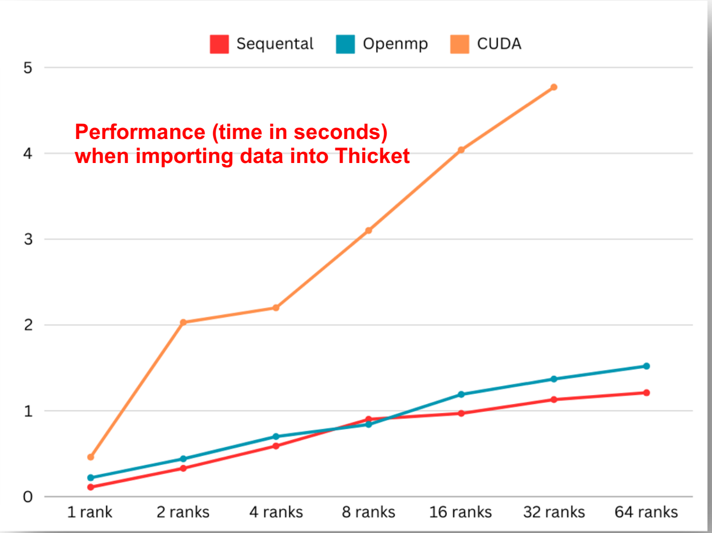
More than 95% of the database consists of regions where little cost was incurred, library implementation details, etc.



Improvement: Performance of Importing Data

	1 rank	2 ranks	4 ranks	8 ranks	16 ranks	32 ranks	64 ranks
Sequential	**0.71s** **0.11s**	**3.99s** **0.33s**	**9.61s** **0.59s**	**24.58s** **0.90s**	**47.21s** **0.97s**	**79.25s** **1.13s**	**158.08s** **1.21s**
OpenMP	**1.19s** **0.22s**	**8.72s** **0.44s**	**22.65s** **0.70s**	**46.14s** **0.84s**	**97.80s** **1.19s**	**208.04s** **1.37s**	**416.86s** **1.52s**
CUDA	**5.61s** **0.46s**	**207.85s** **2.03s**	**672.13s** **2.20s**	**1589.72s** **3.10s**	**3213.55s** **4.04s**	**6525.85s** **4.77s**	

Several hours vs. several seconds



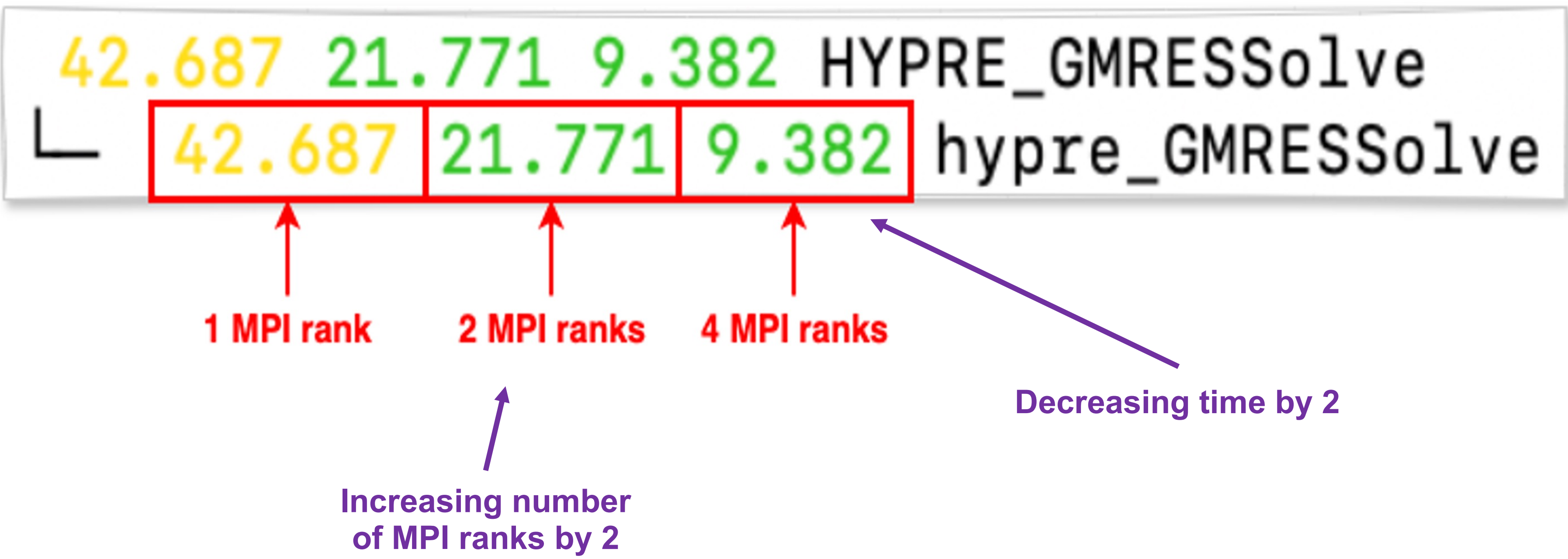
Transforming the Original Tree

Large Calling
Context Tree
13063 CCT nodes

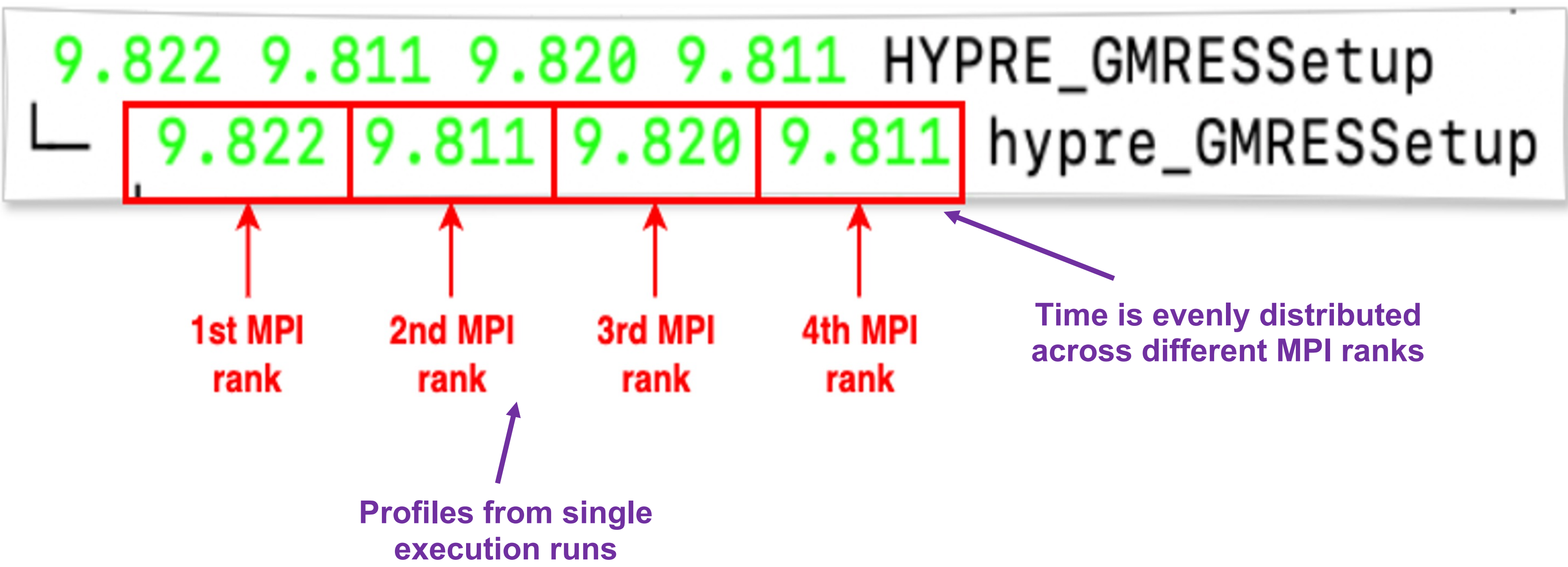
```
4.278 entry
└─ 4.278 main
   ├─ 1.025 BuildIJLaplacian27pt
   │  ├─ 0.349 HYPRE_IJMatrixAssemble
   │  │  └─ 0.349 hypre_IJMatrixAssembleParCSRDevice
   │  │     └─ 0.268 hypre_IJMatrixAssembleSortAndReduce1
   │  │        └─ 0.229 hypreDevice_StableSortTupleByTupleKey
   │  │           └─ 0.229 thrust::stable_sort_by_key
   │  ├─ 0.058 HYPRE_IJMatrixSetValues
   │  │  └─ 0.058 HYPRE_IJMatrixSetValues2
   │  │     └─ 0.058 hypre_IJMatrixSetAddValuesParCSRDevice
   │  ├─ 0.070 hypre_Memcpy
   │  │  └─ 0.070 hypre_Memcpy_core
   │  │     └─ 0.070 cudaMemcpy
   │  ├─ 0.289 loop AMG_DIR/amg.c:1037
   │  │  └─ 0.289 loop AMG_DIR/amg.c:1039
   │  │     └─ 0.289 loop AMG_DIR/amg.c:1041
   │  │        ├─ 0.060 line AMG_DIR/amg.c:1419
   │  │        └─ 0.053 line AMG_DIR/amg.c:1493
   │  └─ 0.190 loop AMG_DIR/amg.c:1707
   │     └─ 0.190 loop AMG_DIR/amg.c:1709
   │        └─ 0.190 loop AMG_DIR/amg.c:1711
   ├─ 0.610 HYPRE_GMRESSetup
   │  └─ 0.610 hypre_GMRESSetup
   │     ├─ 0.556 HYPRE_BoomerAMGSetup
   │     │  └─ 0.556 hypre_BoomerAMGSetup
   │     │     └─ 0.546 loop HYPRE_DIR/parcsr_ls/par_amg_setup.c:981
   │     │        ├─ 0.214 hypre_BoomerAMGBuildModExtPIInterp
   │     │        │  └─ 0.214 hypre_BoomerAMGBuildExtPIInterpDevice
   │     │        │     └─ 0.097 hypre_ParCSRMatrixGenerateFFFCDevice
   │     │        │        └─ 0.097 hypre_ParCSRMatrixGenerateFFFCDevice_core
   │     │        ├─ 0.092 hypre_BoomerAMGCoarsenPMIS
   │     │        │  └─ 0.092 hypre_BoomerAMGCoarsenPMISDevice
   │     │        └─ 0.191 hypre_ParCSRMatrixRAPKT
   │     │           └─ 0.191 hypre_ParCSRMatrixRAPKTDevice
   │     │              └─ 0.141 hypre_CSRMatrixTripleMultiplyDevice
   │     │                 ├─ 0.050 hypre_CSRMatrixMultiplyDevice
   │     │                 │  └─ 0.050 hypreDevice_CSRSpGemm
   │     │                 └─ 0.080 hypre_CSRMatrixMultiplyDevice
   │     │                    └─ 0.080 hypreDevice_CSRSpGemm
   │     └─ 0.054 hypre_ParKrylovCreateVectorArray
   │        └─ 0.054 hypre_CAlloc
   │           └─ 0.054 hypre_MAlloc_core
   │              └─ 0.054 hypre_DeviceMalloc
   │                 └─ 0.054 cudaMalloc
   ├─ 1.864 HYPRE_GMRESSolve
   │  └─ 1.864 hypre_GMRESSolve
   │     ├─ 0.838 hypre_ParKrylovInnerProd
   │     │  └─ 0.838 hypre_ParVectorInnerProd
   │     │     └─ 0.838 hypre_SeqVectorInnerProd
   │     │        └─ 0.838 hypre_SeqVectorInnerProdDevice
   │     │           └─ 0.838 hypre_DeviceDataCublasHandle
   │     │              └─ 0.838 cublasCreate_v2
   │     ├─ 0.704 hypre_ParKrylovMatvec
   │     │  └─ 0.704 hypre_ParCSRMatrixMatvec
   │     │     └─ 0.704 hypre_ParCSRMatrixMatvecOutOfPlace
   │     │        └─ 0.704 hypre_ParCSRMatrixMatvecOutOfPlaceDevice
   │     │           └─ 0.704 hypre_CSRMatrixMatvecOutOfPlace
   │     │              └─ 0.704 hypre_CSRMatrixMatvecDevice
   │     │                 └─ 0.704 hypre_CSRMatrixMatvecDevice2
   │     │                    └─ 0.704 hypre_CSRMatrixMatvecCusparse
   │     │                       └─ 0.704 hypre_CSRMatrixMatvecCusparseNewAPI
   │     │                          └─ 0.704 hypre_DeviceDataCusparseHandle
   │     │                             └─ 0.704 cusparseGetMatFillMode
   │     └─ 0.322 loop HYPRE_DIR/krylov/gmres.c:481
   │        └─ 0.312 loop HYPRE_DIR/krylov/gmres.c:534
   │           ├─ 0.164 HYPRE_BoomerAMGSolve
   │           │  └─ 0.164 hypre_BoomerAMGSolve
   │           │     └─ 0.164 loop HYPRE_DIR/parcsr_ls/par_amg_solve.c:257
   │           │        └─ 0.164 hypre_BoomerAMGCycle
   │           │           └─ 0.164 loop HYPRE_DIR/parcsr_ls/par_cycle.c:286
   │           │              └─ 0.126 loop HYPRE_DIR/parcsr_ls/par_cycle.c:388
   │           │                 └─ 0.126 loop HYPRE_DIR/parcsr_ls/par_cycle.c:395
   │           │                    └─ 0.126 hypre_BoomerAMGRelaxIF
   │           │                       └─ 0.126 hypre_BoomerAMGRelax
   │           │                          └─ 0.126 hypre_BoomerAMGRelax18WeightedL1Jacobi
   │           │                             └─ 0.126 hypre_BoomerAMGRelax7Jacobi
   │           │                                └─ 0.096 hypre_ParVectorInnerProd
   │           │                                   └─ 0.096 hypre_SeqVectorInnerProd
```

94 CCT nodes

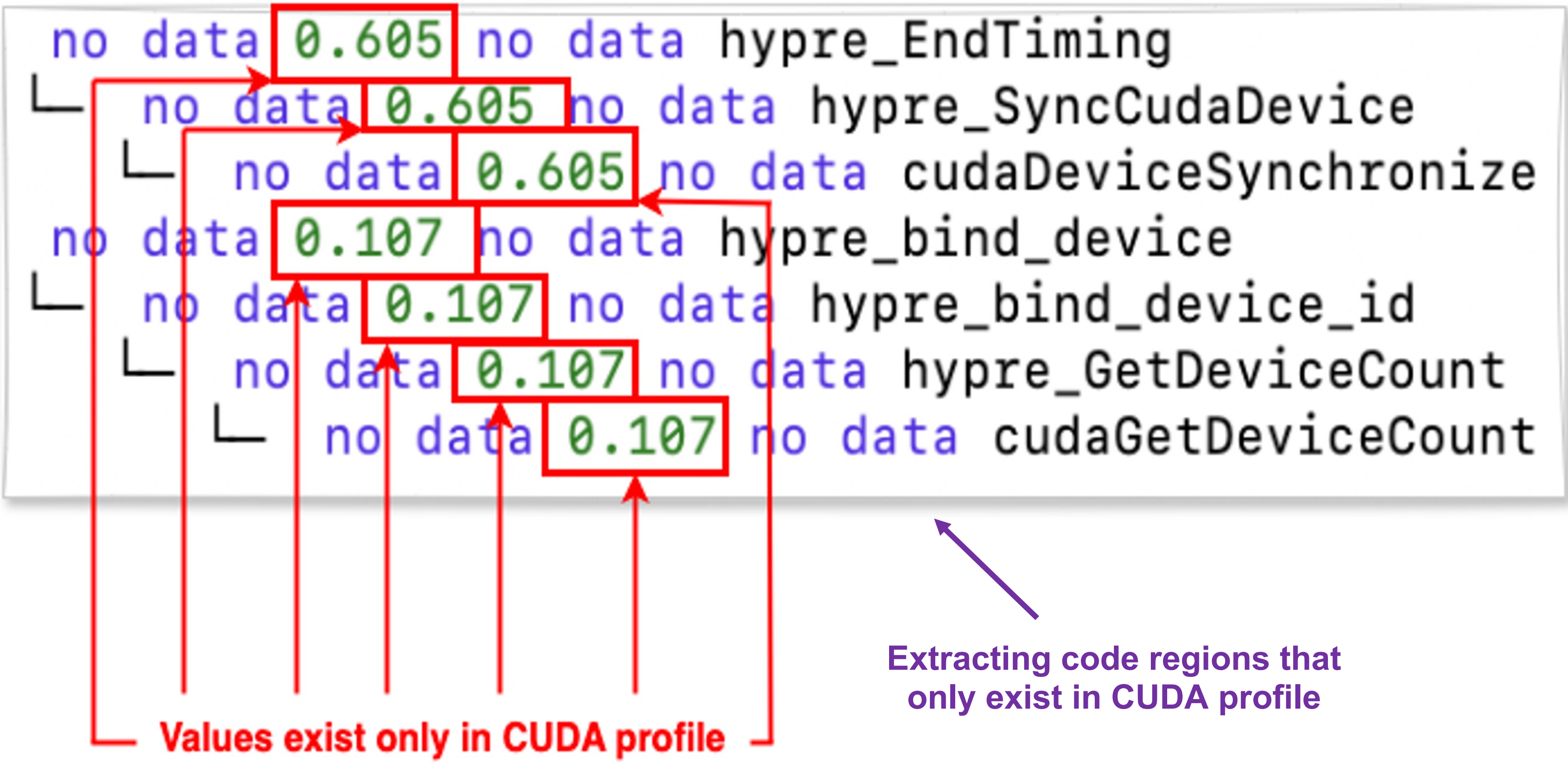
Experiments with Thicket



Detecting Load Imbalance



Comparing Different Parallelization Strategies



New API for Analyzing HPCToolkit Data

New API: Selective Read of Slices of Data

- When analyzing large-scale executions users might want to **selectively read slices of performance data**
  - performance profiles for specific execution contexts
  - performance profiles for specific calling contexts
  - performance profiles for specific metrics
  - trace lines for specific execution contexts
  - trace lines for specific time intervals

Hatchet and Thicket don't support trace analysis

What is a Slice of Profile?

- Slicing can be performed in three dimensions
  - slicing by execution context (**“rank(0).thread(1,5-7)”**)
  - slicing by calling context (**“function(MPI_*)”**)
  - slicing by metrics (**“time (i)”**)

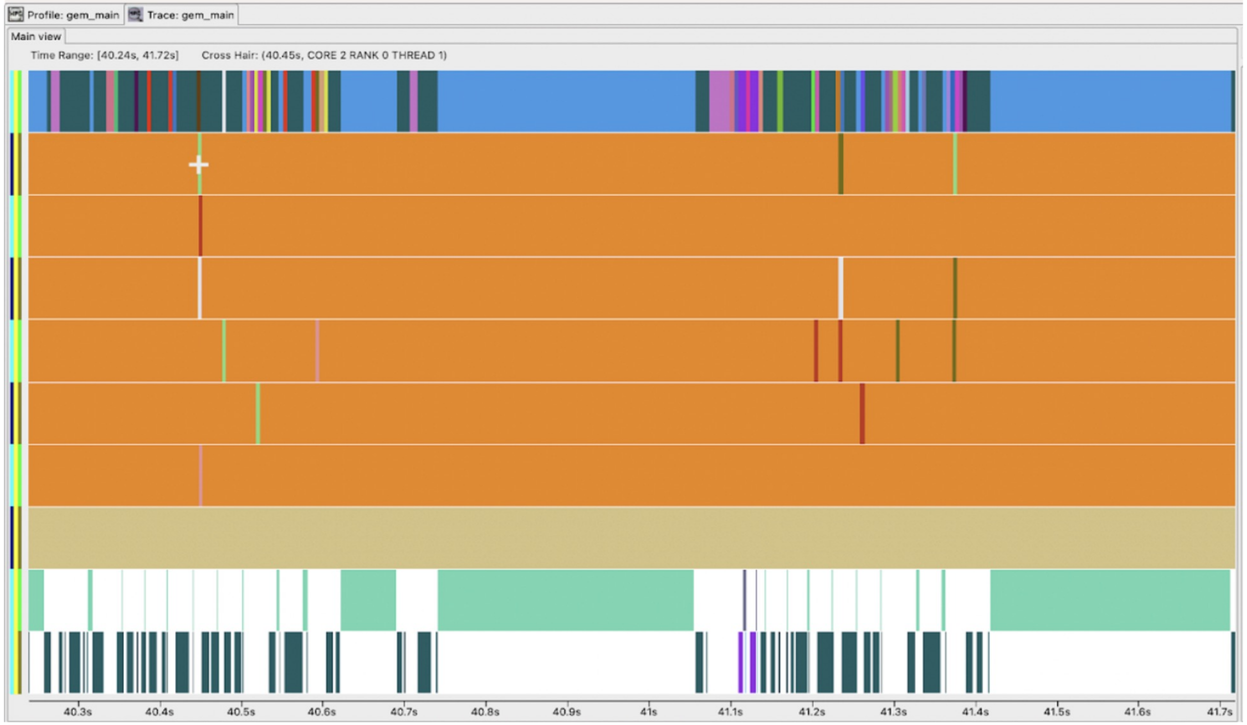
```
import hpcanalysis  # library for extracting and analyzing large-scale performance data

query_api = hpcanalysis.open_db(DATA_DIR)._query_api  # DATA_DIR is the location of the performance database


query_api.query_profile_slices(
    "rank(0).thread(1,5-7)",      # extract profiles with rank ID 0 and thread IDs 1,5,6,7
    "function(MPI_*)",            # extract MPI functions
    "time (i)"                    # extract time metric with inclusive scope
)
```

slicing by execution context

slicing by metric

slicing by calling context

Example of Reading Slices of Profiles



Trace view of GEM execution showcasing CPU underutilization

- **Slicing execution context:** instead of reading all parallel profiles, **extract only CPU profiles**
- **Slicing CCT:** instead of reading the entire CCT, **extract only OpenMP idle nodes**
- **Slicing metrics:** instead of reading all metrics, **extract only time metric**

How is Slicing Performed?

- Query API maps query expressions into positions inside file
  - "rank(0).thread(1,5-7)" -> **profile IDs**
  - "function(MPI_*)" -> **CCT IDs**
  - "time (i)" -> **metric IDs**

profiles are organized into an array where index is equal to profile ID

CCT nodes are sorted by ID inside a profile

metrics are sorted by ID inside a profile

- The API uses special metadata tables to map queries to logical IDs of data slices within the file

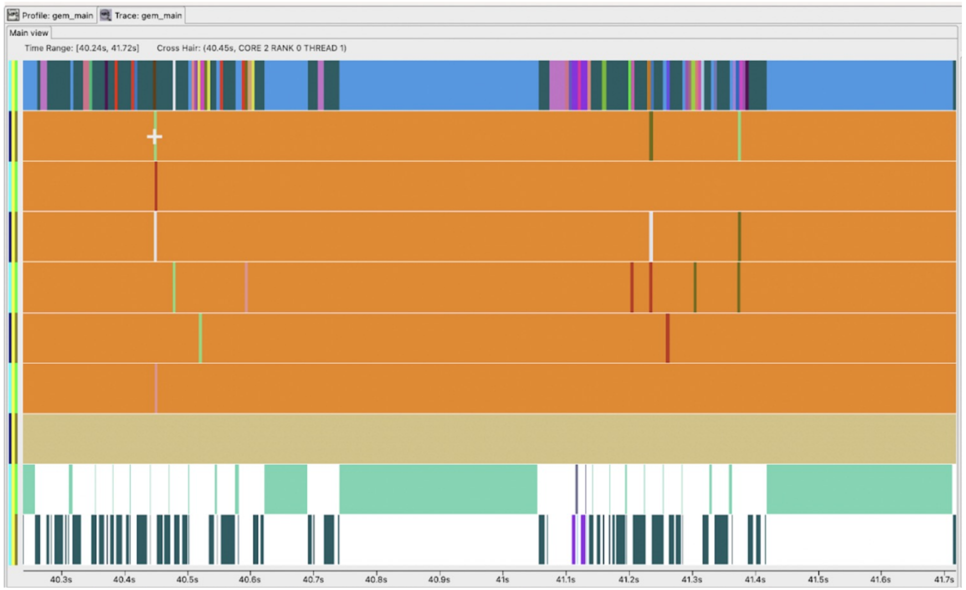
Extracting and Storing Slices of Profiles

- Profiles are stored in a Pandas DataFrame that is initially empty
  - on the first access on a specific slice of profile, **only that slice is fetched** and stored in memory
  - users extract slices using queries, and **Query API maps queries to logical positions inside the file**

```
import hpcanalysis  # library for extracting and analyzing large-scale performance data

query_api = hpcanalysis.open_db(DATA_DIR)._query_api  # DATA_DIR is the location of the performance database


query_api.query_profile_slices(
    "rank(0).thread(1,5-7)",        # extract profiles with rank ID 0 and thread IDs 1,5,6,7
    "function(MPI_*)",              # extract MPI functions
    "time (i)"                      # extract time metric with inclusive scope
)
```

only specific slices are fetched from file

Detecting CPU Underutilization



Trace view of GEM execution showcasing CPU underutilization

```
import hpcanalysis  # library for extracting and analyzing large-scale performance data

hpc_api = hpcanalysis.open_db(DATA_DIR)  # DATA_DIR is the location of the performance database


hpc_api.flat_profile(
    "function(<omp idle>)",
    "rank"
)  # internally it utilizes `query_profile_slices("rank", "function(<omp idle>)", "time (i)")`
```

extract only OpenMP idle nodes

group profiles by rank

rank	thread	<omp idle> (sec)	percentage (%)
0	1	183.652248	99.32
	2	183.666212	99.33
	3	183.528916	99.26
	4	183.470563	99.23
	5	183.565956	99.28
	6	183.680228	99.33
1	1	183.684245	99.34
	2	183.634589	99.32
	3	183.544208	99.26
	4	183.607711	99.30
	5	183.639453	99.31
	6	183.572144	99.27
2	1	183.609443	99.31
	2	183.524535	99.25
	3	183.607519	99.30
	4	183.556586	99.27
	5	183.507577	99.24
	6	183.665352	99.32

Detecting GPU Idleness

Trace view of GAMESS execution showcasing GPU idleness

extract profiles only for the root of application and time and GPU metric → `gpu_idleness = hpc_api.gpu_idleness(`

extract only MPI_Barrier nodes → `"function(MPI_Barrier)",`

extract only CPU profiles → `"rank"`

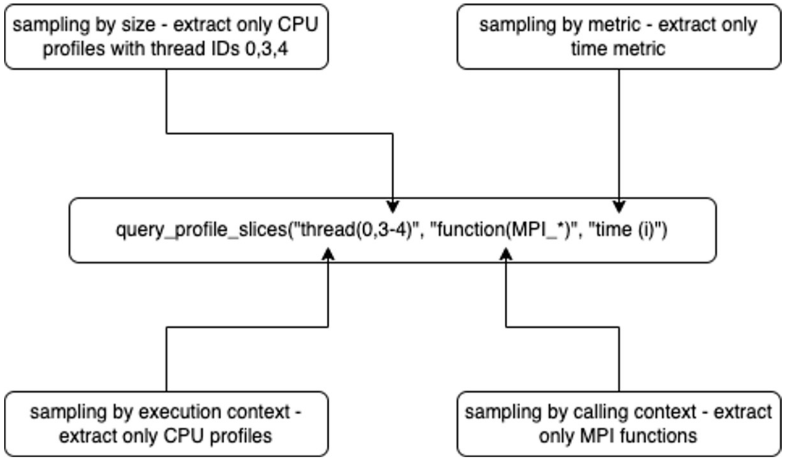
merge tables → `) # merges two tables by their index`

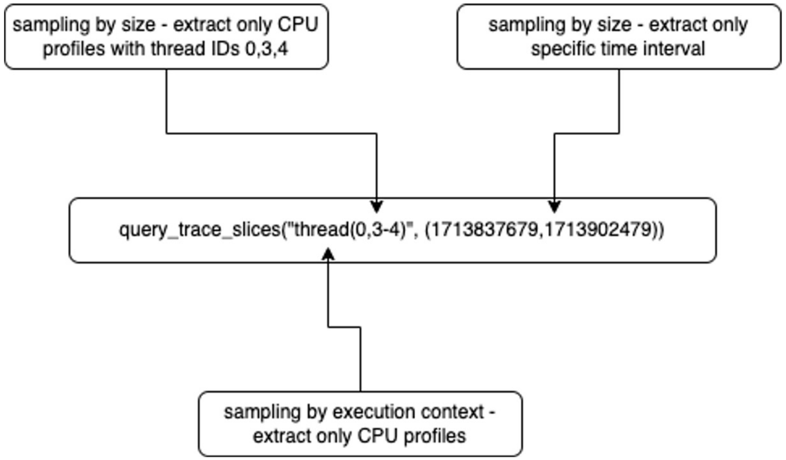
```
import hpcanalysis  # library for extracting and analyzing large-scale performance data

hpc_api = hpcanalysis.open_db(DATA_DIR)  # DATA_DIR is the location of the performance database



gpu_idleness = hpc_api.gpu_idleness(
)  # internally it utilizes `query_profile_slices("rank", "application", ["time (i)", "gpuop (i)"])`
mpi_barrier_flat_profile = hpc_api.flat_profile(
    "function(MPI_Barrier)",
    "rank"
)  # internally it utilizes `query_profile_slices("rank", "function(MPI_Barrier)", "time (i)")`
gpu_idleness.merge(
    mpi_barrier_flat_profile,
    how="left",
    left_index=True,
    right_index=True
)  # merges two tables by their index
```

rank	GPU total time (sec)	GPU idle time (sec)	MPI_Barrier (sec)
6	379.957775	79.858442	11.194035
2	378.830011	81.006407	34.237175
38	368.295603	91.608697	64.447354
0	365.880202	93.668491	34.131954
4	359.614108	100.009650	34.207448
34	346.879048	112.856035	69.843765
30	322.464969	137.355333	67.799732
36	320.288131	139.580611	69.862606
32	319.383060	140.467428	67.748216
26	314.848771	144.919576	87.029797
16	312.376347	147.438938	89.359397
24	310.638123	149.151571	86.975268
28	300.709540	158.974191	67.771281
20	298.809228	161.044371	96.421229
8	285.247963	174.591349	11.238722
12	267.222066	192.546391	89.464256
10	263.630137	196.264350	11.253637
22	251.233301	208.621200	119.560394
18	240.735282	218.906282	89.379163
14	237.997966	221.905124	89.423164

Sampling Strategies

- Query API enables users to sample performance data in two different ways
  - **sampling by context** - sampling by execution context, calling context, or sampling by specific metrics when extracting slices of profiles
  - **sampling by size** - sampling by fragments of data - range of values for the execution context or time intervals when extracting slices of traces





Analyzing Large-Scale Executions

- Large-scale execution of LAMMPS - 4TB of data

```
import hpcanalysis  # library for extracting and analyzing large-scale performance daa

hpc_api = hpcanalysis.open_db(DIR_PATH)  # DIR_PATH is the location of the performance database


hpc_api.load_imbalance(
    "rank(0-100000:10000).thread(0)", "function(MPI_*)")  # extract 100000/10000=10 samples
hpc_api.load_imbalance(
    "rank(0-100000:1000).thread(0)", "function(MPI_*)")   # extract 100000/1000=100 samples
hpc_api.load_imbalance(
    "rank(0-100000:100).thread(0)", "function(MPI_*)")    # extract 100000/100=1000 samples
hpc_api.load_imbalance(
    "rank(0-100000:10).thread(0)", "function(MPI_*)")     # extract 100000/10=10000 samples
hpc_api.load_imbalance(
    "rank(0-100000).thread(0)", "function(MPI_*)")        # extract 100000/1=100000 samples
```

extract every 1000th rank

extract every 100th rank

extract every 10th rank

extract every rank

	samples count	time (sec)	memory (bytes)	MSE error
0	10	3.742822	10976	0.034474
1	100	3.641423	49096	0.017945
2	1000	4.270574	405595	0.007701
3	10000	15.736037	4006007	0.000425
4	100000	2197.340860	45430836	0.000000

function	load imbalance
MPI_Cart_rank	1.000000
MPI_Scan	1.000000
MPI_Finalize	0.842299
MPI_Reduce	0.600000
MPI_Bcast	0.336671
MPI_Cart_create	0.317299
MPI_Barrier	0.286229
MPI_Sendrecv	0.215673
MPI_Irecv	0.192319
MPI_Send	0.155680
MPI_Wait	0.135578
MPI_Allreduce	0.049684

10 samples

function	load imbalance
MPI_Cart_rank	0.971205
MPI_Scan	0.997816
MPI_Finalize	0.787869
MPI_Reduce	0.434601
MPI_Bcast	0.234888
MPI_Cart_create	0.092303
MPI_Barrier	0.280955
MPI_Sendrecv	0.209480
MPI_Irecv	0.153511
MPI_Send	0.143607
MPI_Wait	0.136728
MPI_Allreduce	0.044204

100 samples

function	load imbalance
MPI_Cart_rank	0.848439
MPI_Scan	0.976709
MPI_Finalize	0.716778
MPI_Reduce	0.327811
MPI_Bcast	0.216440
MPI_Cart_create	0.082632
MPI_Barrier	0.277750
MPI_Sendrecv	0.203069
MPI_Irecv	0.144470
MPI_Send	0.138225
MPI_Wait	0.132652
MPI_Allreduce	0.043685

1000 samples

function	load imbalance
MPI_Cart_rank	0.726031
MPI_Scan	0.769399
MPI_Finalize	0.614790
MPI_Reduce	0.298850
MPI_Bcast	0.211739
MPI_Cart_create	0.078125
MPI_Barrier	0.276795
MPI_Sendrecv	0.182100
MPI_Irecv	0.129364
MPI_Send	0.138305
MPI_Wait	0.132081
MPI_Allreduce	0.043631

10000 samples

function	load imbalance
MPI_Cart_rank	0.719402
MPI_Scan	0.726261
MPI_Finalize	0.558767
MPI_Reduce	0.285497
MPI_Bcast	0.212510
MPI_Cart_create	0.078036
MPI_Barrier	0.277058
MPI_Sendrecv	0.179116
MPI_Irecv	0.116940
MPI_Send	0.130573
MPI_Wait	0.118880
MPI_Allreduce	0.042803

100000 samples

Blaming GPU Idleness

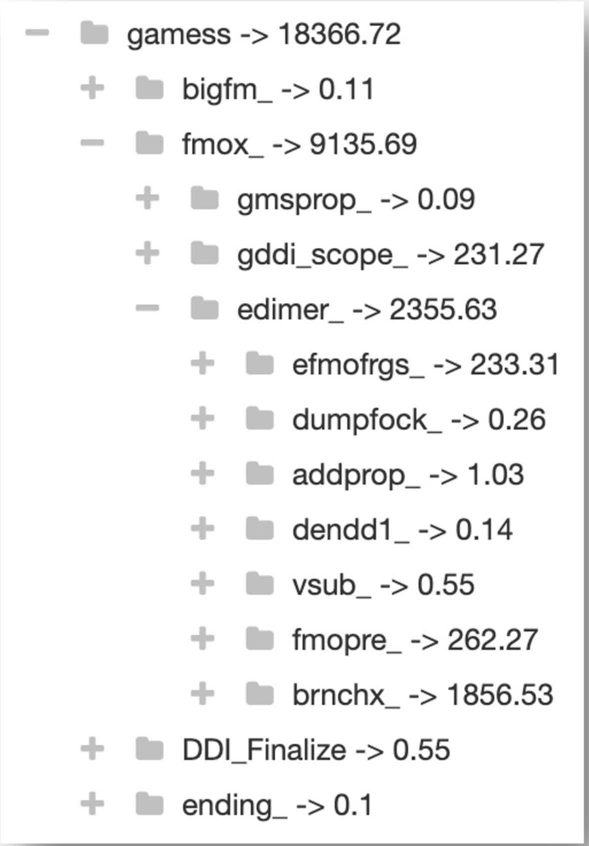
extract entire GPU trace line

extract slice of CPU trace line

reconstruct tree for CPU trace segment

```
import hpcanalysis  # library for extracting and analyzing large-scale performance data

hpc_api = hpcanalysis.open_db(DIR_PATH)  # DIR_PATH is the location of the performance database


gpu_trace = hpc_api._query_api.query_trace_slices(
    "rank(0).gpucontext(1)"  # extract GPU trace line
)

gpu_trace["duration"] = gpu_trace["end_timestamp"] - gpu_trace["start_timestamp"]  # calculate the duration
gpu_idle_events = gpu_trace[gpu_trace["ctx_id"] == 0]  # extract idle events
longest_idle_event = gpu_idle_events.loc[gpu_idle_events["duration"].idxmax()]  # determine the longest idle event

cpu_trace_segment = hpc_api._query_api.query_trace_slices(
    "rank(0).thread(0)",
    (longest_idle_event["start_timestamp"], longest_idle_event["end_timestamp"])
)  # extract CPU trace segment being executed while the GPU was idle

cpu_events = cpu_trace_segment["ctx_id"].unique().tolist()  # extract CPU events

hpc_api.visualize_cct(
    cct_indices=cpu_events,  # reconstruct CCT from CPU events
    show_only_functions=True,  # visualize only functions
    show_metrics=True  # show metric values recorded for each node in the reconstructed subtree
)
```

Detecting longest GPU idle event and blaming corresponding CPU code



Visualizing CPU activity while GPU was idle

Conclusion

- Users can analyze HPCToolkit data using Hatchet and Thicket
  - when reading the data, they can enable various data reduction heuristics
- New API for extracting slices of HPCToolkit data
  - users can extract slices of performance data from the persistent storage using queries
  - efficient solution when analyzing very large-scale executions
  - users can analyze both profiles and traces
  - ongoing work:
    - creating custom regression tests for validating the performance database
    - more work on trace analysis
    - extending the library to read the data from a remote server

Lawrence Livermore National Laboratory

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC