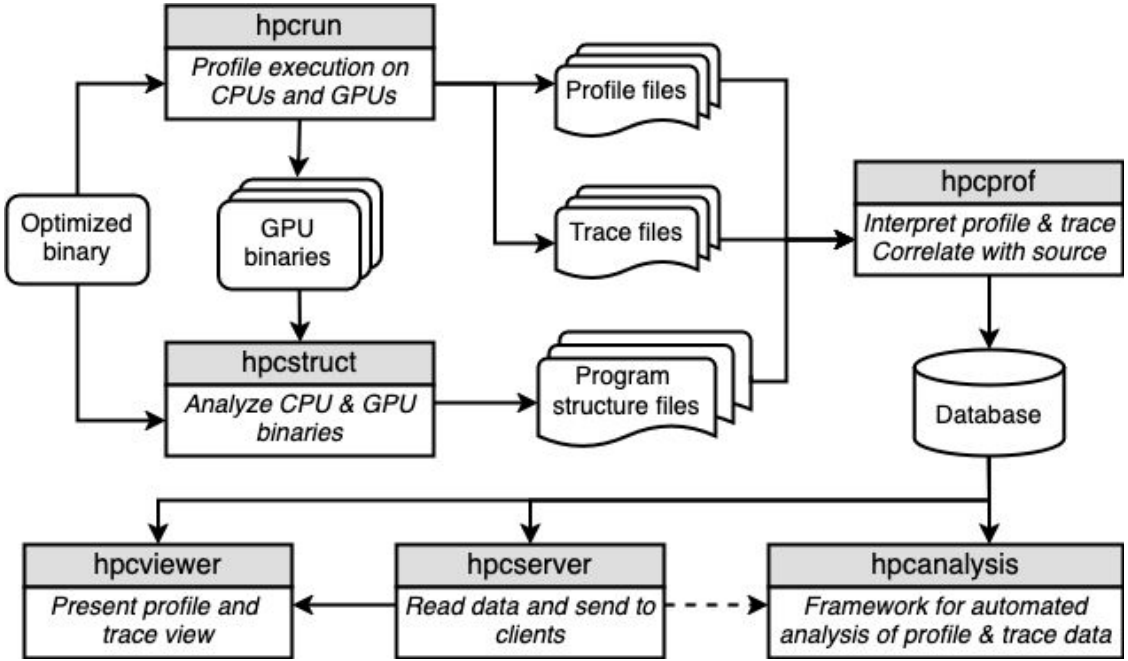
Enhancements to HPCToolkit for Analysis of CPU and GPU-accelerated Applications

Laksono Adhianto and John Mellor-Crummey
Rice University

Scalable Tools Workshop 2025

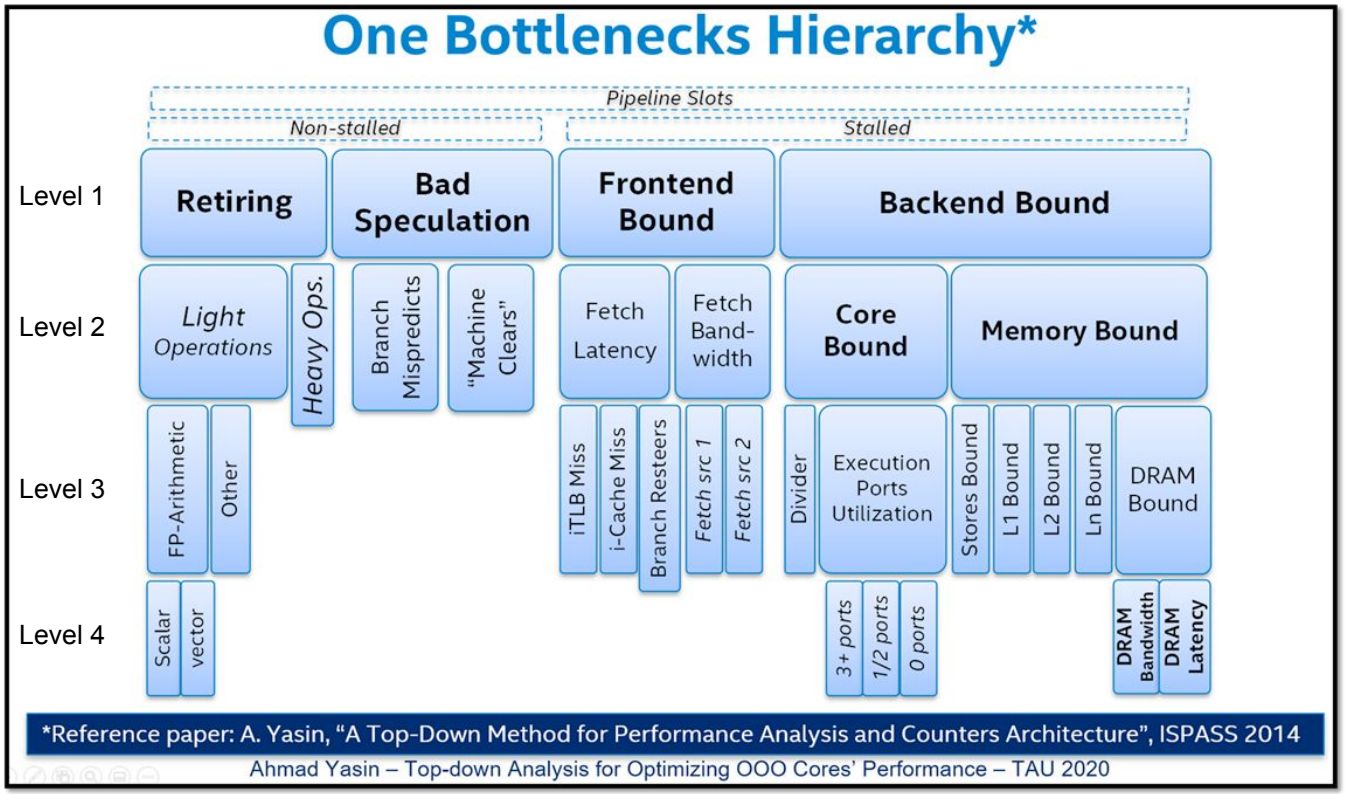
HPCToolkit Workflow



HPCToolkit New Features and Enhancements

- **CPU profiling**
  - Support for Intel Top-down analysis
- **GPU profiling**
  - Support for AMD, Intel and NVIDIA GPUs
- Access to remote databases
  - Have been used in several hackathon
  - Ongoing work to improve the performance
- Presentation
  - Updated Hpcviewer GUI with new look

Support for Top-Down Model Analysis (TMA)



PERF_METRICS MSR

- Intel PERF_METRICS MSR: a special counter to provide percentages of slots for four TMA level 1 and four TMA level 2 metrics
- Four additional TMA level 2 metrics (Core Bound, Fetch Bandwidth, Machine Clears and Light Operations) can be derived from the metrics
- **Pros**: No need to configure many counters
- **Cons**: Limited granularity, not a precise event, update rate ~1-10ms (observed, not official rate), **not be suitable for fine-grain measurement**

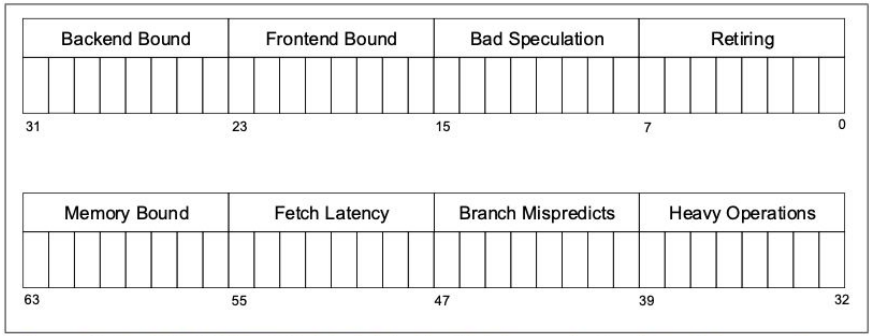


Figure 21-40. PERF_METRICS MSR Definition for 12th Generation Intel® Core™ Processor P-core

Core_Bound = Backend_Bound - Memory_Bound
Fetch_Bandwidth = Frontend_Bound - Fetch_Latency
Machine_Clears = Bad_Speculation - Branch-Mispredict
Light_Operations = Retiring - Heavy_Operation

Tools Supporting Top-Down Analysis

Tool	Mode	Built-in top-down levels supported
Caliper	Counting & Sampling	Level 1-3
HPCToolkit	Sampling	Level 1-4 (mostly)
Likwid	Counting	Level 1 (at least)
Linux Perf stat	Counting	All levels, default level 1-2
PAPI	Counting	Level 1-2
Score-P	Counting	Level 1-2
VTune	Sampling	Level 1-4 and some level 5 & 6

Issues Top-Down Analysis in Sampling Mode

- Issues with libpfm4
  - Bug in translating from top-down events to perf_event configuration
  - Fixed in the main branch, but not in the release
    - New `TOPDOWN_M` pseudo event to use PERF_METRICS MSR
- Issue with Intel perfmon JSON file
  - Incorrect specification of some top-down events: `Serializing_Operation`, `AMX_Busy`, and `Nop_Instructions`
  - Fixed in the main branch
- Issue with Linux v5
  - Unable to group top-down events in sampling mode with perf_events
    - Grouping top-down events works fine in counting mode
  - Fixed in Linux v6

Issue with Linux v5 (case with perf tool)

```
$ perf stat -e '{slots,topdown-bad-spec}' /bin/ls
... (success)
```

```
$ perf record -e '{slots,topdown-bad-spec}' /bin/ls
Error: The sys_perf_event_open() syscall returned with 22 (Invalid argument) for event (topdown-bad-spec).
```

```
$ perf record -e '{slots,topdown-bad-spec}:S' /bin/ls
Error: The sys_perf_event_open() syscall returned with 22 (Invalid argument) for event (topdown-bad-spec).
```

```
$ perf record -e '{slots,cycles,topdown-bad-spec}:S' /bin/ls
...(success)
```

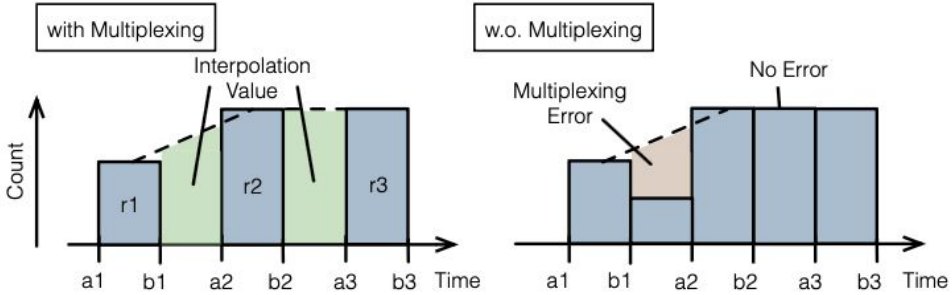
Top-Down Analysis Implementation

- Measurements
  - TMA level 1 and 2: use PERF_METRICS MSR
  - TMA level 3 and 4: use groups of hardware counters as specified in Intel JSON file
    - Avoid using all TMA metrics in JSON file, select only critical ones
    - Test usability of hardware counters before using them

      `CPU_CLK_UNHALTED.THREAD` **VS** `CPU_CLK_UNHALTED.THREAD_P`
    - Avoid using deprecated counters

      `OFFCORE_REQUESTS_OUTSTANDING.ALL_DATA_RD` **VS** `OFFCORE_REQUESTS_OUTSTANDING.DATA_RD`
  - Profiled with frequency-based sampling + multiplexing (time sharing)

Implementation Challenges



- So many events, so few registers
  - Even worse if SMT is enabled → even less registers
  - Multiplexing (time sharing) interpolates the samples → inaccuracy → uncertainty
- Accuracy issues
  - Some counters may overcount: FP scalars & FP vectors due to FMA instruction
  - Some counters may overlap: Branch resteers with MS switches, L3 hit latency with Contested accesses, …
- Precision issues
  - PERF_METRICS MSR is not designed for fine-grain measurement, can we mix with precise events?
- Presentation: how to present effectively to users?
  - Calling-context tree + Top-down Metrics + source codes

Image taken from: Sudvarg, Li, et al. “Tintin: PMU Scheduling to Minimize Uncertainty.” 19th annual workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT), July 2025

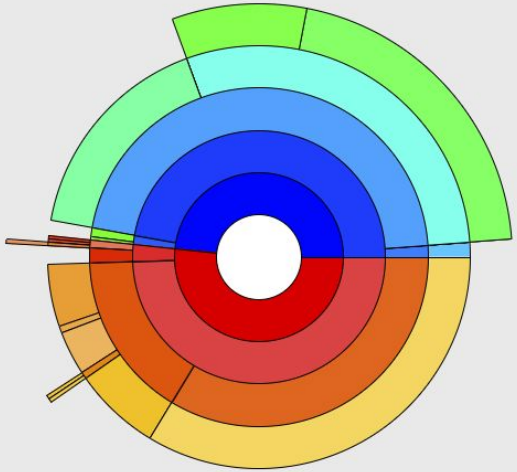
hpcviewer

Profile: single.nooffload.gcc.cpu | Profile: lulesh2.0

Topdown metrics

- 48.5% Active (I)
  - 47.6% Retiring: Sum (I)
    - 1.3% Heavy-Operations: Sum (I)
      - 1.3% Uops Sequencer (I)
      - 0.0% Few_Uops_Instructions (I)
    - 46.3% Light-Operations (I)
      - 29.8% FP-Arithmetic (I)
        - 0.0% X87-Use (I)
        - 21.3% Instr-Scalar (I)
        - 8.5% Instr-Vector (I)
      - 0.0% Int-Operations (I)
      - 16.5% Mem-operations (I)
  - 0.9% Bad-Speculation: Sum (I)
    - 0.7% Mispredict: Sum (I)
    - 0.2% Machine-Clears (I)
- 51.5% Stall (I)
  - 2.5% Frontend-Bound: Sum (I)
    - 1.0% Fetch-Latency: Sum (I)
      - 0.3% ICache-Miss (I)
      - 0.1% ITLB-Miss (I)
      - 0.7% Branch-Resteer (I)
        - 0.0% unknown_branch (I)
      - 1.4% Fetch-Bandwidth (I)
  - 49.1% Backend-Bound: Sum (I)
    - 16.0% Memory-Bound: Sum (I)
      - 5.0% L1-Bound (I)
      - 0.4% L2-Bound (I)
      - 3.3% L3-Bound (I)
      - 0.5% DRAM-Bound (I)
        - 0.2% DRAM-Bandwidth (I)
        - 0.3% DRAM-Latency (I)
      - 6.7% Store-Bound (I)
    - 33.1% Core-Bound (I)
      - 33.1% Arith-Divider (I)
      - 0.0% Serialization (I)
      - 0.0% AMX-Busy (I)

Topdown metrics of [I] CalcElemFBHourglassForce



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

Scope	Total-Slots (I)		Total-Slots (E)	
Experiment Aggregate Metrics	3.00e+11	100.0%	3.00e+11	100.0%
<thread root>	1.59e+11	53.1%		
» gomp_thread_start [libgomp.so.1.0.0]	1.59e+11	53.1%	9.18e+08	0.3%
loop at team.c: 139	1.59e+11	53.1%	9.18e+08	0.3%
130 » gomp_team_barrier_wait_end [libgomp.so.1.0.0]	2.61e+10	8.7%		
129 » CalcFBHourglassForceForElems(Domain&, double*, double*, double*, double*, double*, double*, double*, double, int, i...	2.51e+10	8.4%	8.03e+09	2.7%
loop at lulesh.cc: 824	2.51e+10	8.4%	2.30e+09	0.8%
923 » [I] CalcElemFBHourglassForce	1.11e+10	3.7%	1.11e+10	3.7%
lulesh.cc: 726	1.95e+09	0.6%	1.95e+09	0.6%
lulesh.cc: 724	1.29e+09	0.4%	1.29e+09	0.4%
lulesh.cc: 702	1.26e+09	0.4%	1.26e+09	0.4%
lulesh.cc: 720	9.09e+08	0.3%	9.09e+08	0.3%

GPU Support in HPCToolkit

Unreleased

	ROCm	Level0	CUDA	OpenCL
Profiling GPU Operations	✓	✓	✓	✓
Tracing of GPU operations	✓	✓	✓	✓
PC sampling	✓	✓	✓	
HW counters for kernel launches	✓			
Page migration; scratch space mgmt	✓			
Binary instrumentation		✓		

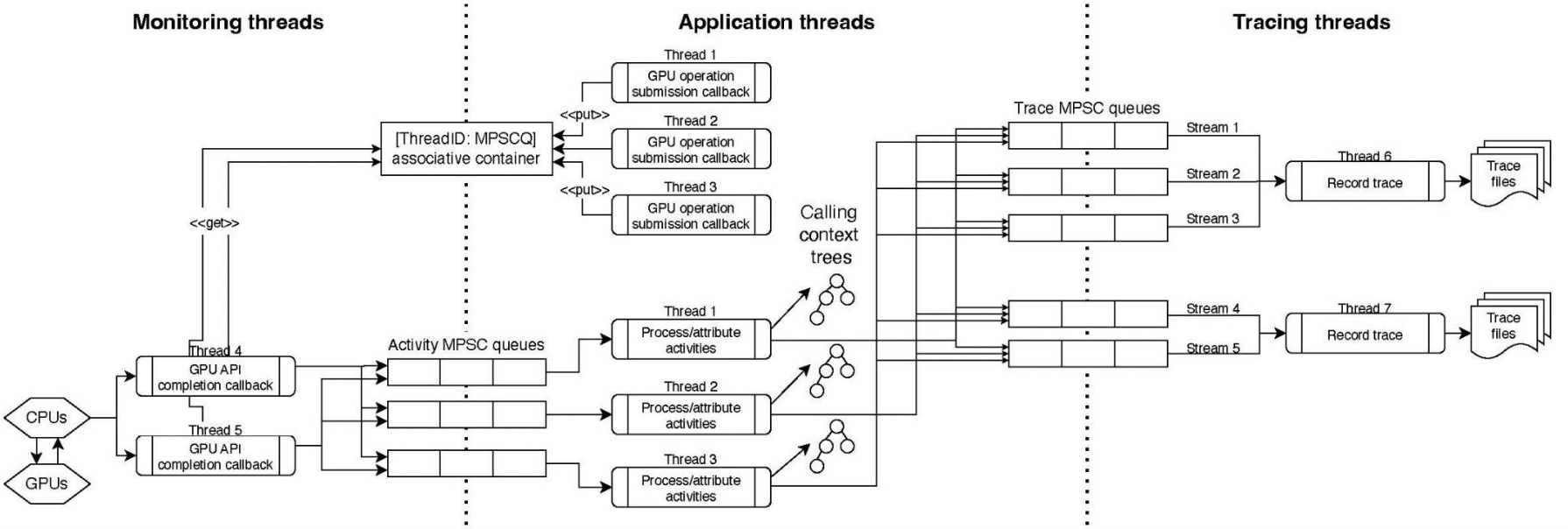
Key Additions to AMD Rocprofiler-sdk API This Year

- Initialization: rocprofiler_configure/rocprofiler_force_configure
  - Integration with HPCToolkit was surprisingly subtle
    - Either HPCToolkit initialization or rocprofiler_configure callback may occur first
      - HPCToolkit initialization of rocprofiler-sdk triggers rocprofiler_force_configure
      - rocprofiler_configure triggers HPCToolkit initialization
    - Requires a “rendevous” so that all HPCToolkit initialization occurs within prepare_measurement_subsystem and all rocprofiler-sdk initialization occurs in the scope of a callback
- PC sampling configuration
  - Informs a tool which GPUs will be used by a process: enables selective configuration

New Design for HPCToolkit's GPU Monitoring Substrate

- New design supports multiple monitoring threads
- GPU monitoring
  - CUDA: single monitoring thread
  - Level 0 and OpenCL: unspecified threads
  - AMD rocprofiler-sdk supports multiple monitoring threads
    - HW counter reporting
    - PC sample reporting
    - Activity API for reporting GPU operations

Runtime Processing of GPU Measurement Data



Vendor GPU Monitoring Concerns

- Intel Level0 reports distinct GPU binaries per MPI rank!
- Intel's PTI View API is only half complete
  - Provides a completion callback that delivers a sequence of "activity records" for GPU ops
  - Lacking several key capabilities
    - initialization
    - intercept launch of GPU operations (for correlating them with their invocation context)
- AMD GPU OpenMP support is awkward
  - Had to use special ROCm API for monitoring rather than OMPT interface
- NVIDIA Activity API record for a kernel provides only string name
  - Lacks the precise attribution to function objects present in PC samples
  - Requires awkward recording of strings rather than addresses!

Tool Challenges and Approaches

- Tool code in the application namespace: unwanted interactions
  - Dangerous to load a tool's C++ library in application namespace
    - Application and tool may be linked with different C++ libraries
  - Application symbols interfere with tools
    - Some applications use libunwind that conflict with our tools, others define mmap
    - Some versions of bash (e.g. RHEL 8) define getenv, which interferes with tool startup
      - HPCToolkit, rocprofiler-sdk, PAPI, and libpfm all expect libc getenv
- Namespaces in Linux: dlmopen (glibc 2.3.4), LD_AUDIT (glibc 2.4)
- HPCToolkit
  - Uses multiple namespaces to avoid conflicts
  - Uses LD_AUDIT to monitor dynamic library loading and symbol binding

Software Infrastructure Woes

- Thread-local variables aren’t not async signal safe
  - https://sourceware.org/glibc/wiki/TLSandSignals
- Pthread keys don’t support multiple namespaces!
  - https://sourceware.org/bugzilla/show_bug.cgi?id=24776

- Worse: dynamic linker prior to glibc 2.34 uses pthread key
  - Support for multiple namespaces is broken on Aurora (glibc 2.31)

Pthread Keys don’t Support Multiple Namespaces!

```
typedef int (*pthread_key_create_t)(pthread_key_t *, void (*)(void*));
int main() {
  pthread_key_t k1, k2;

  int rc = pthread_key_create(&k1, NULL);                    // create key in the default namespace

  void *h = dlmopen(LM_ID_NEWLM, "libpthread.so.0", RTLD_LAZY); // open libpthread in a new namespace
  assert(h != NULL);
  // find pthread_key_create in the new namespace
  pthread_key_create_t fn = (pthread_key_create_t)dlsym(h, "pthread_key_create");
  assert(fn != NULL);

  rc = fn(&k2, NULL);                                        // create a key in the new namespace
  assert(rc == 0);

  assert(k2 != k1);                                          // the key is same in both namespaces

  return 0;
}
```

assert fails on ALL Linux systems

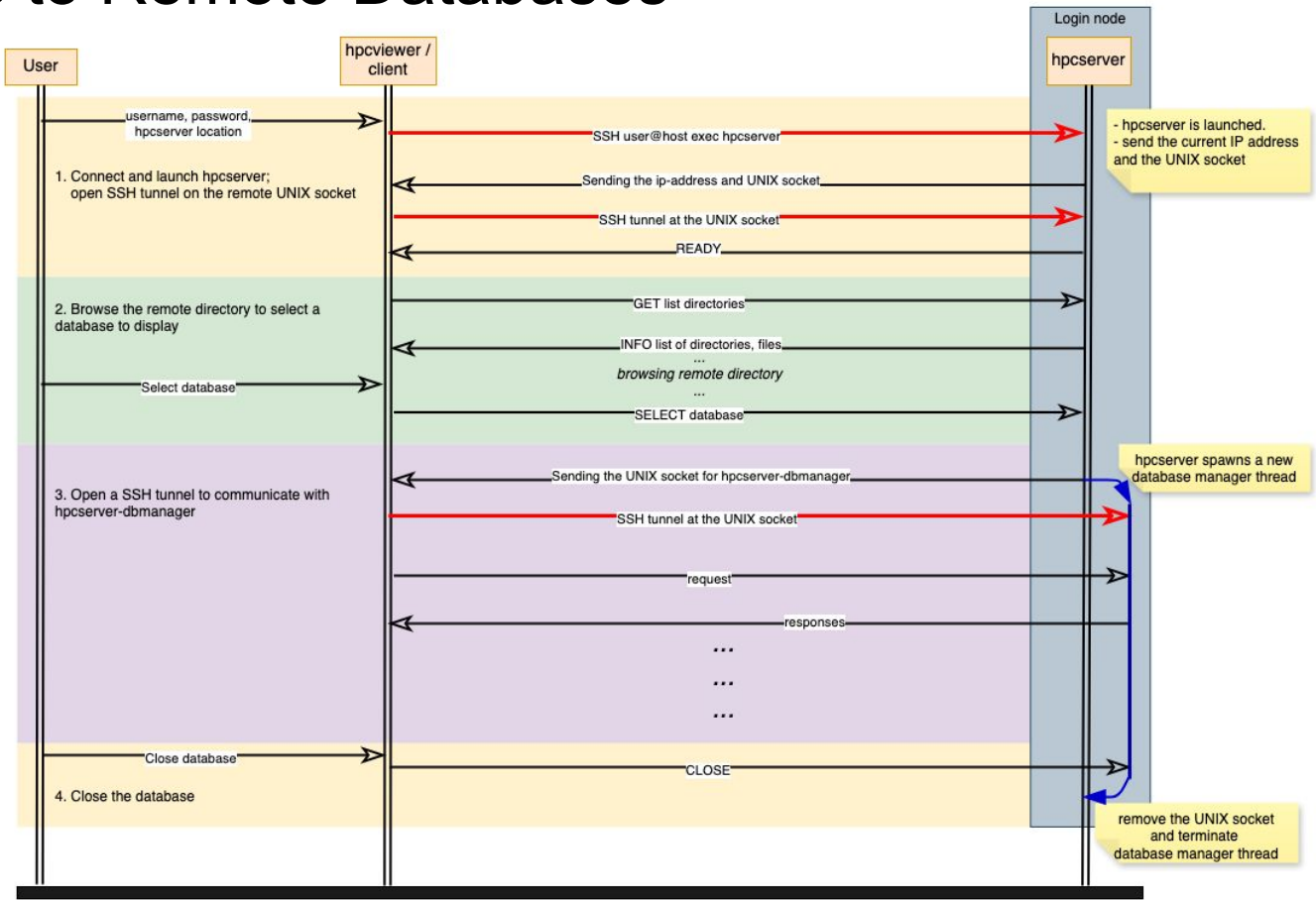
Paul Pluzhnikov 2021-11-20 https://sourceware.org/bugzilla/show_bug.cgi?id=24776

Summary

- Support for Top-down analysis
  - HPCToolkit provides metrics in different views and different scopes for procedures, loops or lines
  - Current support: Intel Sapphire Rapids or newer
    - Not supported: E-Core in hybrid architectures
    - Can be extended to other platforms: AMD and ARM and GPUs
  - Ongoing work: handle uncertainty
- Support for new GPU monitoring substrate, rocprofiler-sdk will soon be released
- Need continued engagement with Intel to get a usable PTI-View interface
- Re-engage Linux glibc team about improving support for namespaces, dynamic linking, and LD_AUDIT

Backup Slides

Access to Remote Databases



Multi-producer, Single-consumer, Wait-free Queue

```
1   struct Element
2       Element *next, Record r
3
4   struct Queue
5       Element *head, Element *tail
6
7   procedure init(Queue q)
8       q.head = NULL; q.tail = NULL
9
10  procedure enqueue(Queue q, Element e)
11      atomic_store(&e.next, NULL)
12      previous_tail = atomic_exchange(&q.tail, e)
13      if previous_tail is NULL then
14          atomic_store(&q.head, e)
15      else
16          atomic_store(&previous_tail.next, e)
17
```

```
18  function dequeue(Queue q)
19      first = atomic_load(&q.head)
20      if first is NULL then return NULL
21      successor = atomic_load(&first->next)
22      if successor is not NULL then
23          atomic_store(&q->head, successor)
24          return first
25      else if atomic_compare_exchange(
26                          &q.tail, &first, NULL)
27          expected_head = first
28          atomic_compare_exchange(&Q.head,
29                  &expected_head, NULL)
30          return first
31      else
32          return NULL
```

Note: sacrifice linearizability for wait-freedom