Parallel Programming and Heterogeneous Computing

Shared-Memory: Profiling

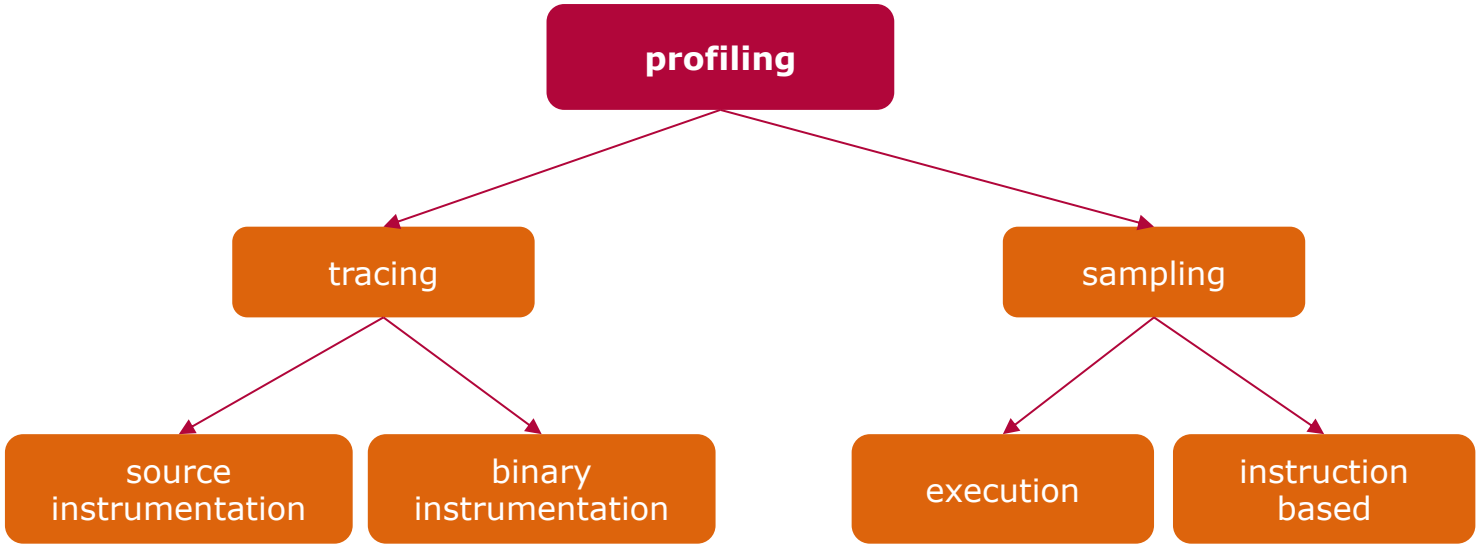
Max Plauth, *Sven Köhler*, Felix Eberhardt, Lukas Wenzel, and Andreas Polze

Operating Systems and Middleware Group

Profiling

- Many available profiling tools for shared-memory parallelism
- **Sampling profiler**
  - Sporadic recording of application state
  - **Time-driven**: Uniform time period between samples
  - **Event-driven**: Uniform event number between samples
  - Original code remains unchanged
  - Small impact on execution behavior, can uncover race conditions
  - Low overhead (hardware-based ~ 2%, software-based ~ 5%)
- **Instrumenting profiler (tracing)**
  - Modification of original application with measurement code
  - Allows gathering of all possible events
  - Higher accuracy than sampling, but also higher overhead
- Data gathering is one part, visualization a different one

Profiling Taxonomy

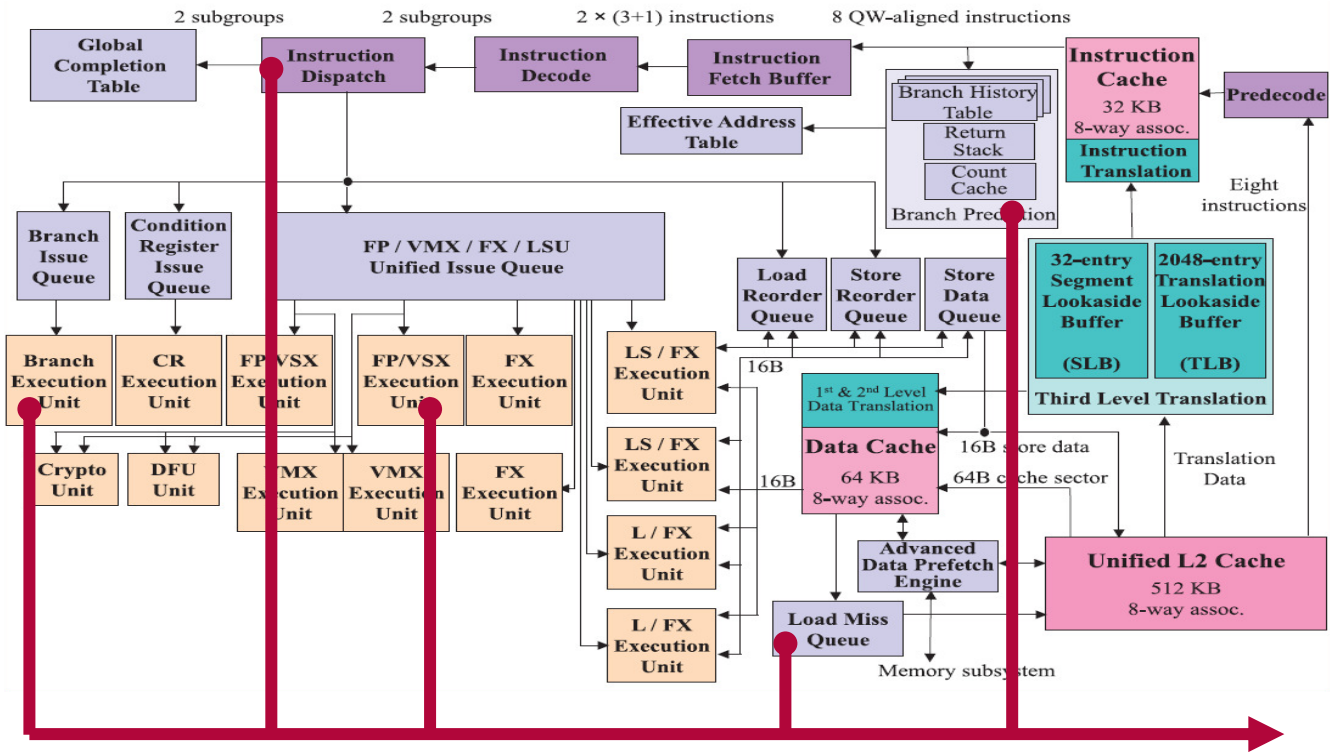


Hardware Performance Counters

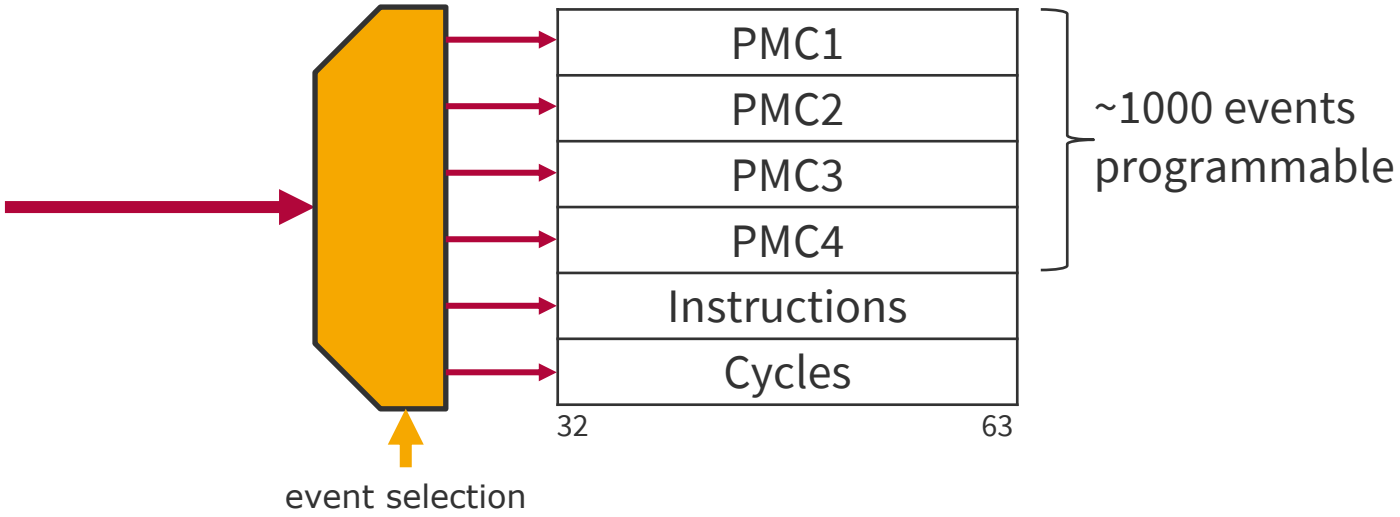
Hardware Performance Counters, part of

Performance Measurement Units [Intel]
Performance Monitoring Units [ARM]
Performance Monitor Counters [IBM]
Performance Counter Events [AMD]
…

Hardware Performance Counters (Power8)

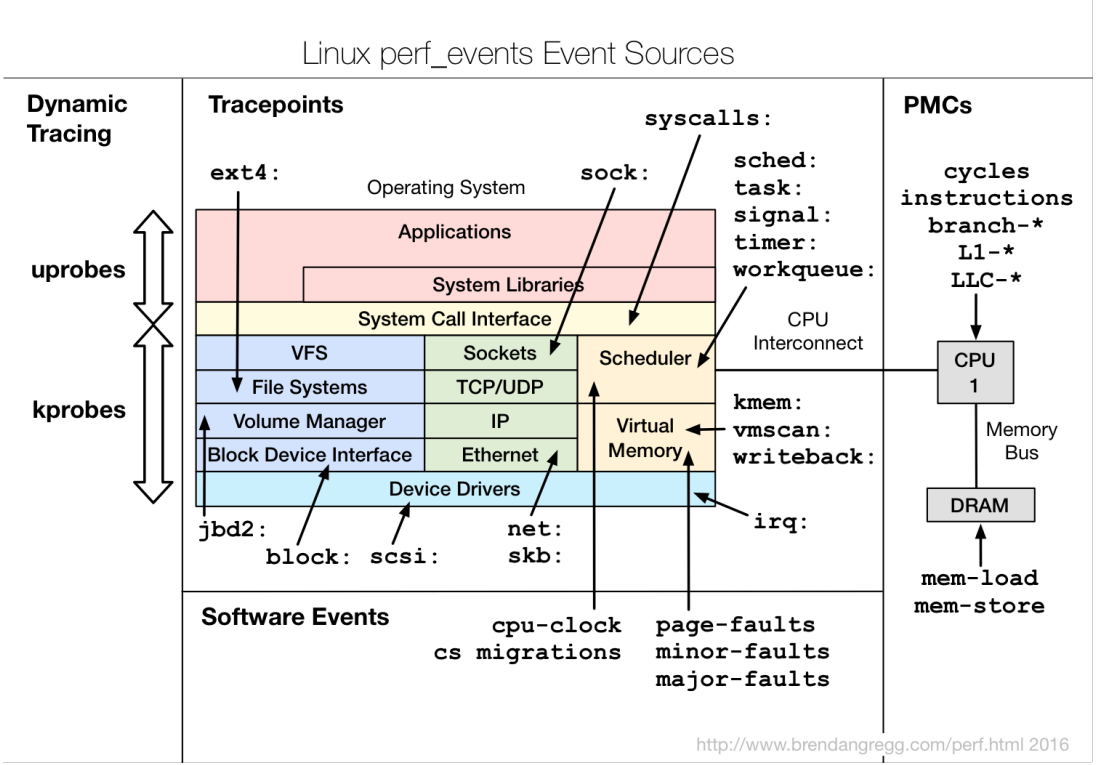


Hardware Performance Counters (Power8)



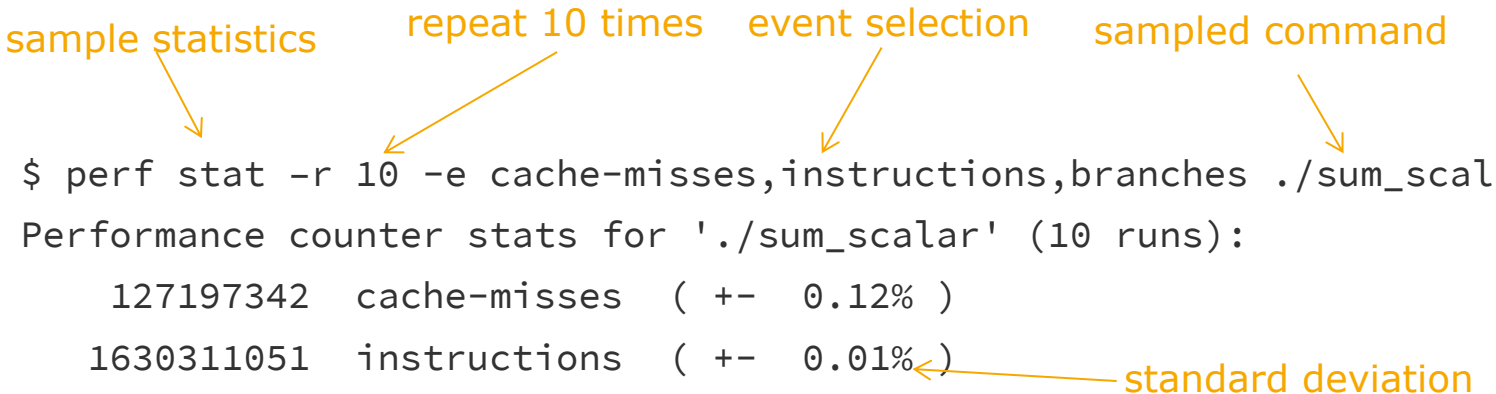
- Low-overhead measurements, non-overhead to program execution
- Usually requires kernel privileges to be read
- Libraries and tools abstracting different vendors: perf, hwlock, libpapi
- Event cycling (round-robin) when measured more than present registers

Linux: perf



Alternative for instrumentation: see `man 2 perf_event_open`

Perf Event Counting (scalar vs vector example)




```
$ perf stat -r 10 -e cache-misses,instructions,branches ./sum_scalar
Performance counter stats for './sum_scalar' (10 runs):
     127197342  cache-misses   ( +-  0.12% )
    1630311051  instructions   ( +-  0.01% )
     271443407  branches       ( +-  0.01% )

  0.512781785 seconds time elapsed  ( +-  0.18% )
```

Perf Event Counting (scalar vs vector example)


```
$ perf stat -r 10 -e cache-misses,instructions,branches ./sum_vector4
Performance counter stats for './sum_vector4' (10 runs):
    144145872  cache-misses   ( +-  0.03% )
    422847600  instructions   ( +-  0.00% )
    226690657  branches       ( +-  0.02% )

  0.429783986 seconds time elapsed  ( +-  0.12% )
```

standard deviation

Determine available events

```
$ perf list
List of pre-defined events (to be used in -e):
  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  cache-misses                                       [Hardware event]
  cache-references                                   [Hardware event]
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  # ...
```

Raw Events

- Many vendors provide special events, that not map to symbolic names

skid (see next section)

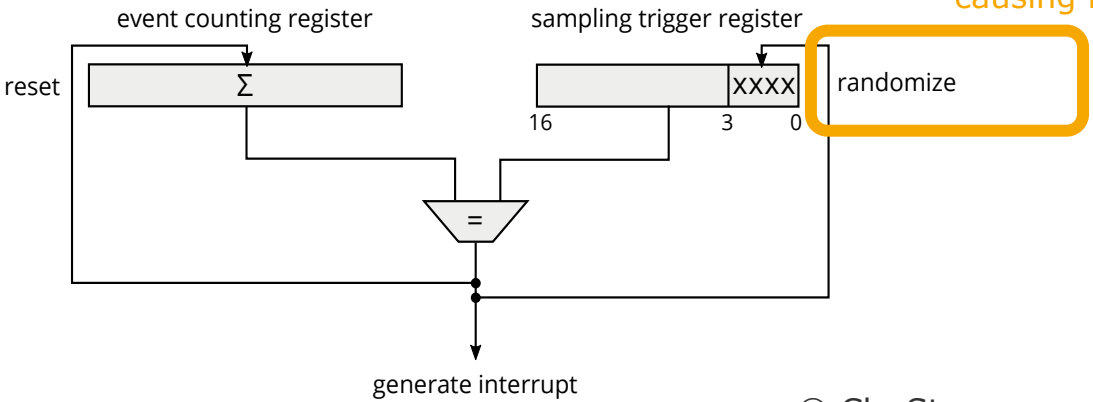
```
$ perf stat -e ra09c:ppp,ra09e:ppp ./sum_scalar
         0   ra09c:ppp
         0   ra09e:ppp

$ perf stat -e ra09c:ppp,ra09e:ppp ./sum_vector4
  33719464   ra09c:ppp
  33392615   ra09e:ppp
```

There are two symmetric Vector-and-Scalar Units (VSU) pipelines on POWER

EventCode: 0xa09c
EventName: PM_VSU0_4FLOP
BriefDescription: SP vector versions of single flop instructions

https://github.com/open-power/power-pmu-events/blob/master/events/power8.json
cross-platform: libpapi's src/libpfm4/examples/showevtinfo

Instruction Based Sampling

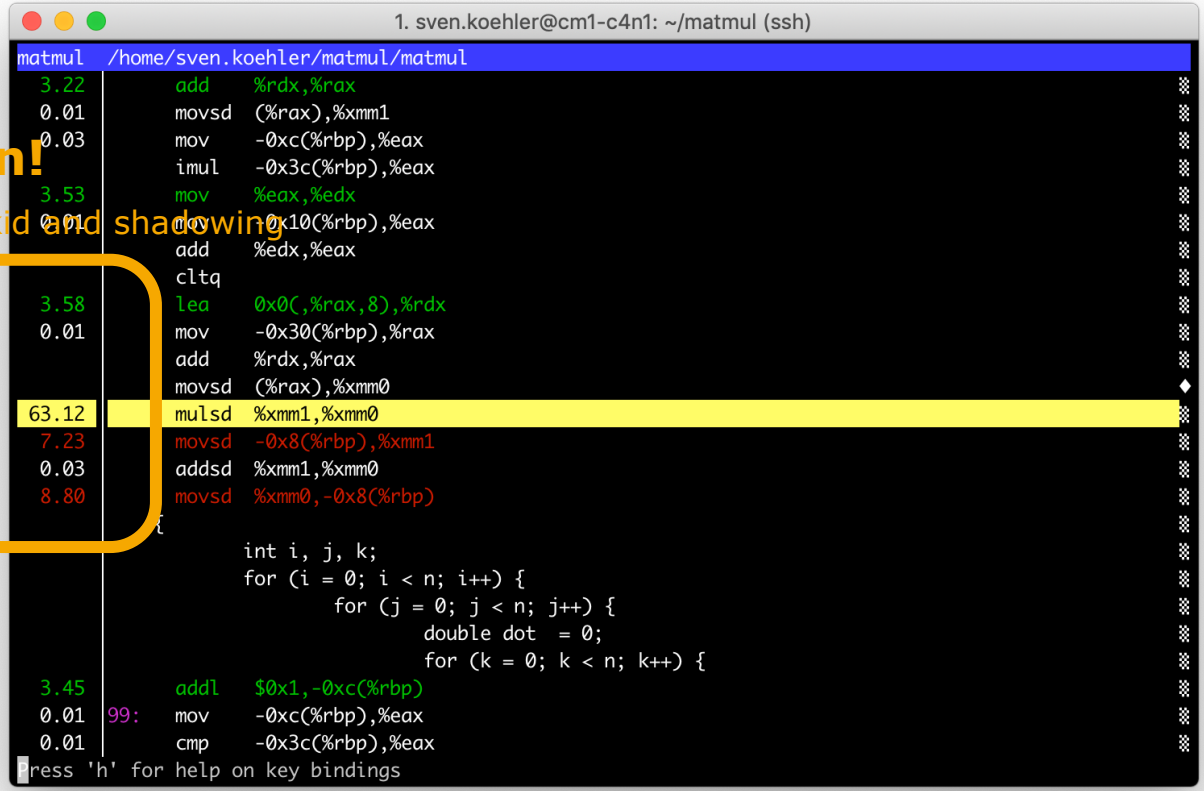


To overcome sampling errors (tight loops, etc), as you cannot increase the sampling rate without causing much interrupt overhead

perf record -> perf.data -> perf annotate

Caution!

prone to skid and shadowing



Problems with ISB

Skid ::

Small delay of unknown duration, which is the time between issuing an instruction and detecting the PMU event [1].

Causes the instruction pointer to spread around the actual address of the event triggering instruction.

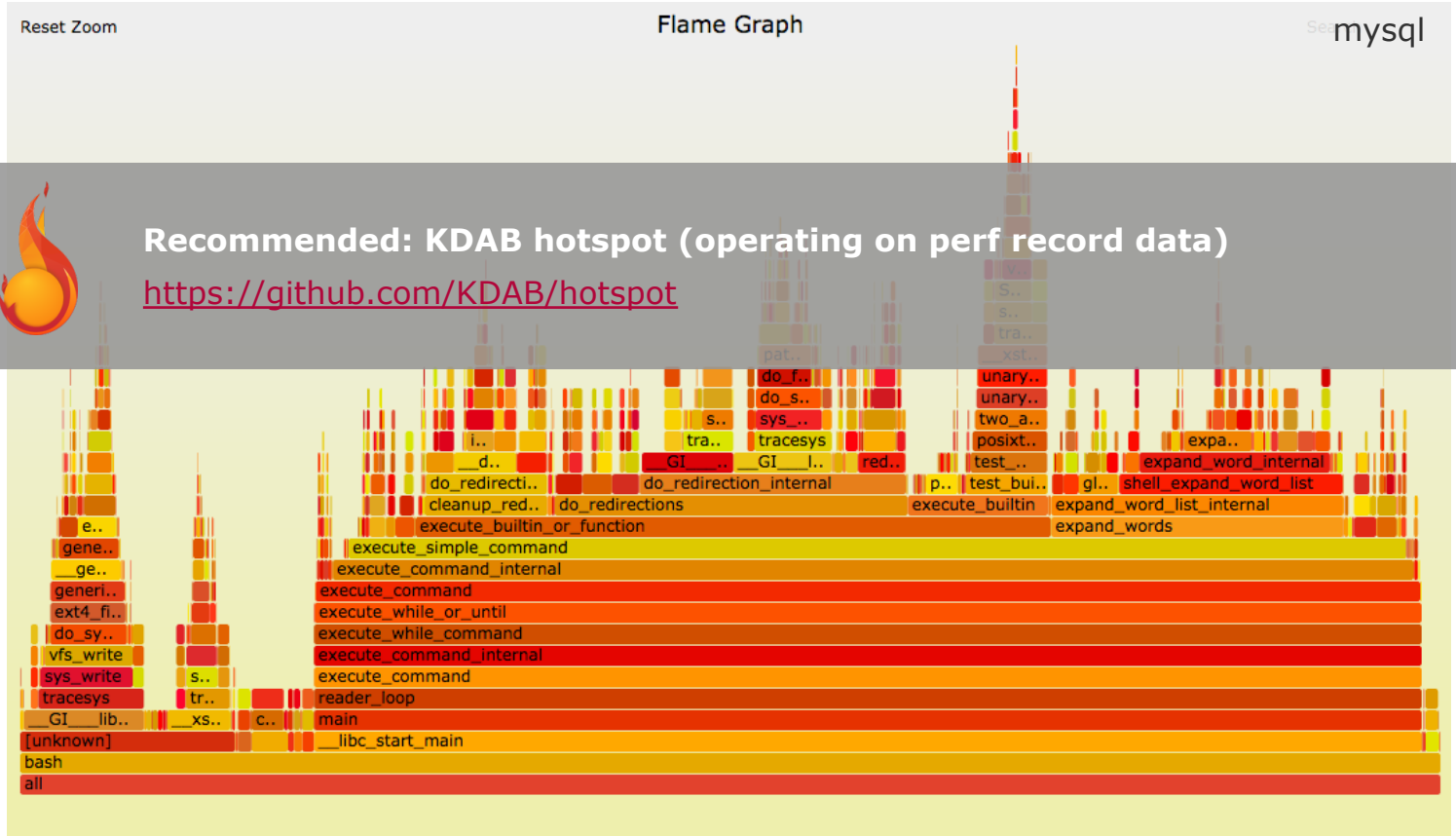
Use eventname:ppp in perf to reduce the sampling trigger.

Shadowing ::

Instructions causing long stalls (e.g. TLB misses or NUMA remote accesses) get sampled disproportionally more often than shorter instructions and appear to cause a higher number of events than actually happened [2].

[1] Paul Drongowski et al. "Incorporating instruction-based sampling into AMD CodeAnalyst". In: Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on. IEEE. 2010, pp. 119–120
[2] Dehao Chen et al. "Taming hardware event samples for FDO compilation". In: Proceedings of the 8th annual IEEE/ACM international symposium on Code generation and optimization. ACM. 2010, pp. 42–52.

Event Flame Graph



Recommended: KDAB hotspot (operating on perf record data)

https://github.com/KDAB/hotspot

ParProg20 B5 Profiling

Sven Köhler

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

And now for a break and
a cup of cappuccino*.

*or beverage of your choice