Speed-up perf tools using multi-thread

For faster perf report

Namhyung Kim

LinuxCon Japan 2015

Who am I

- Namhyung Kim
- work for LG Electronics
- contributing to Linux since 2010
- perf developer since 2012

perf tools

- performance analysis tools for Linux
- support hardware (PMU) and software (kernel) events
- very actively developed
- git-like architecture
  - record + report
  - stat, top, script, diff, timechart, probe ...

basic usage

- perf record ./a.out
- perf record -ag – sleep 1
- perf record -ag make -j 20
- perf report

perf record

- build perf_event_attr based on -e/–event option
  - default to cpu-cycles (or cpu-clock if not available)
- open perf event fd using the attr
  - man perf_event_open(2)
- mmap each event
  - same cpu/thread events share mappings
- kernel writes event contents to the mapping
- perf reads out events in a loop and writes to file
- write file header at last
  - with additional feature/info

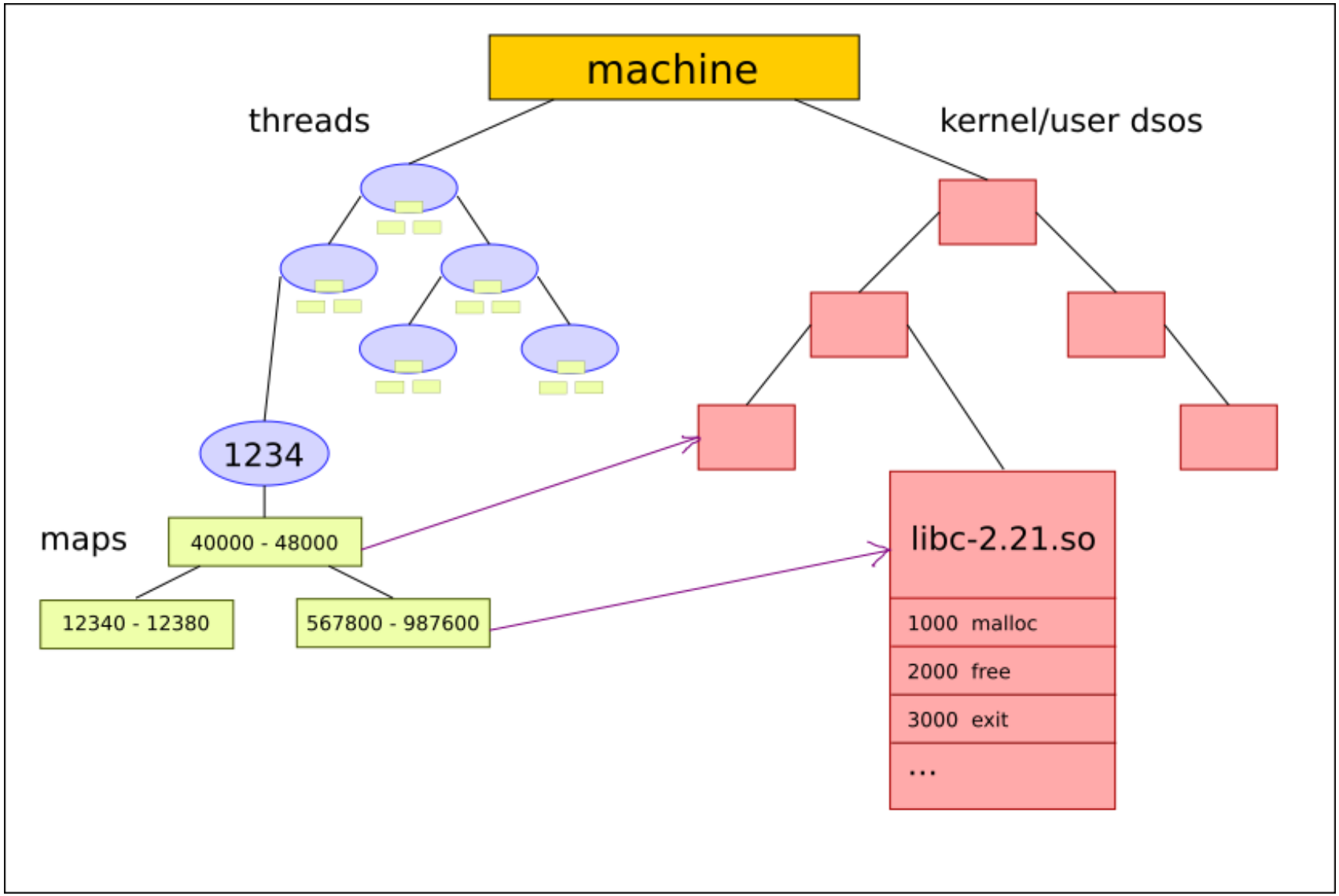
perf report

- parse file header and feature/info
- process meta/sample events
- build histogram entries
  - based on sort keys given
  - default to comm,dso,symbol
- collapse similar entries (optional)
- sort entries for output
- display (stdio/tui/gtk)

building histogram

- process events
  - meta events reconstruct machine state
  - sample events find matching symbols from ip
  - using a queue for event ordering
- add to histogram
  - thread -> map -> dso -> symbol
  - resolve callchains (if any)
  - merge same entries

machine state



but . . .

- perf report can take minutes (hours?) on large (GB) data file
  - mostly to insert/merge hist entries and callchains
- recent version will be slower than before
  - due to changes like atomic refcounting and locks
- currently Ctrl+C doesn't work on TUI
  - patch posted/merged recently
  - `https://lkml.org/lkml/2015/5/28/770`

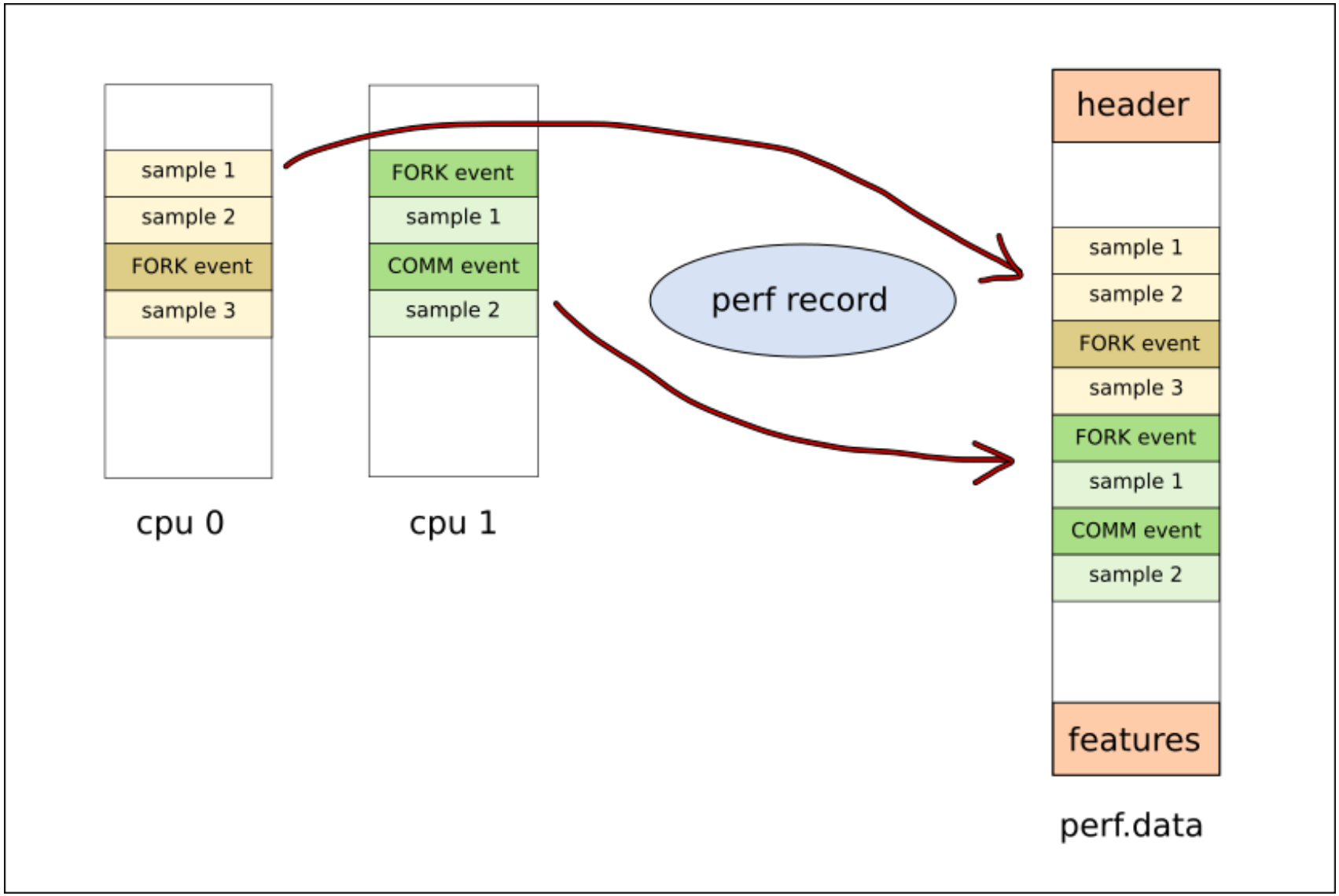
speed-up

- insert/merge hist entry + callchains is cpu-intensive work
  - this can be partitioned well as they came from different source
- separate data into pieces and process with multi-thread
  - index data file when recording
  - construct and keep full machine states over time
  - each thread parses/processes sample events

perf data file format

- header
- event description
  - attr + id
  - ‘perf evlist -v‘
- events
  - meta + sample events
- feature
  - ‘perf report –header(-only)‘

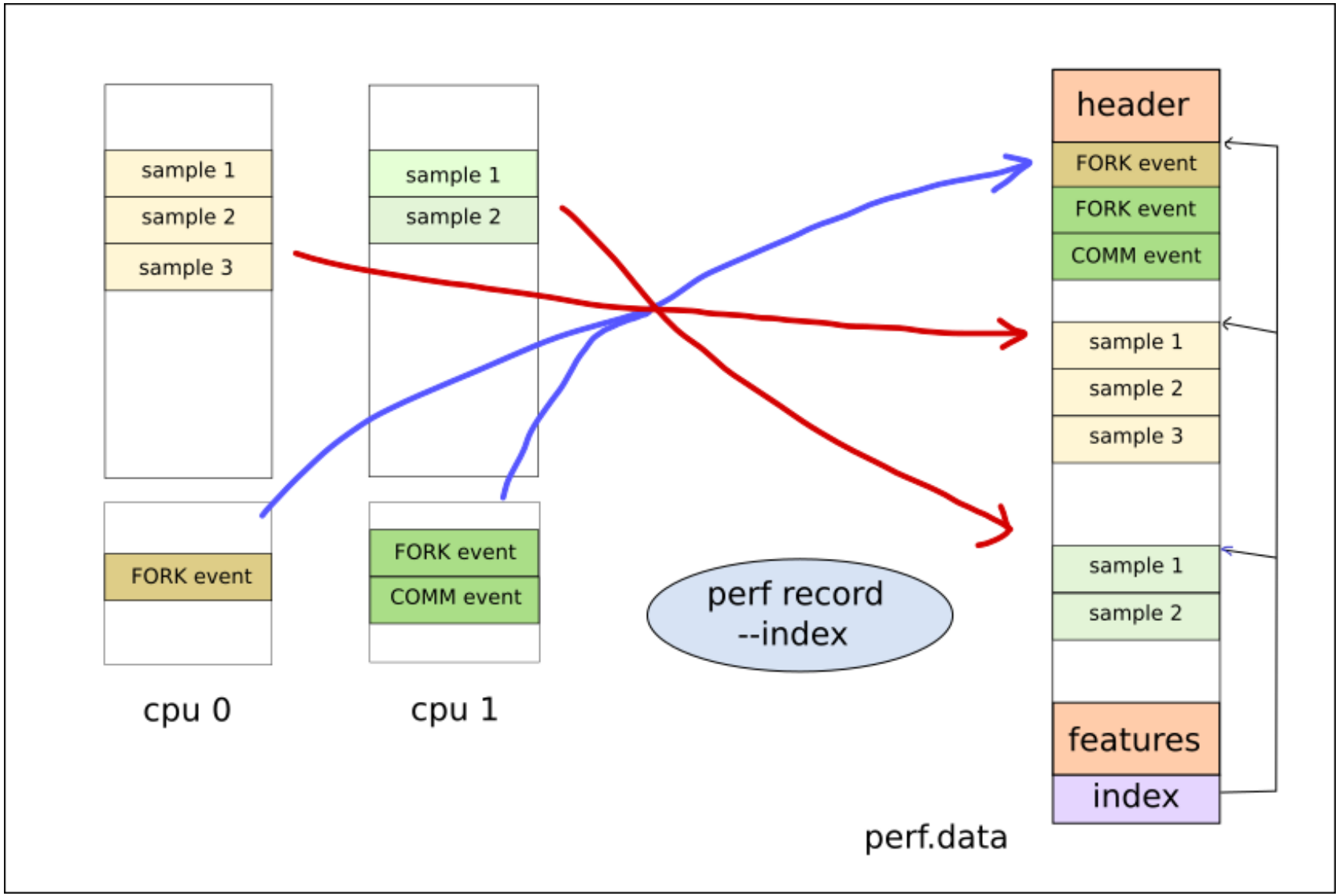
normal perf data file



indexing data file

- split meta and sample events
  - save events in separate (intermediate) files
  - and then merge them into a single file
  - index saved in the feature area
- Index 0: meta events
  - processed serially at first
- index N: sample events
  - processed parallelly at the same time

indexed perf data file



processing meta events

- construct machine state using timestamp
- original perf only keeps current state
  - overwrite old state as it never be accessed again
- keep old state as well
  - to be found by old samples later
  - pid can be reused for different thread
  - comm, map can be changed in a thread

processing samples

- find thread using pid/timestamp
  - tid/pid might be recycled
- find map group using timestamp
  - thread might call exec()
- find map using address
  - there might be overlaps
- find symbol using address
- create hist entry
- compare and insert/merge
- processing callchains if any

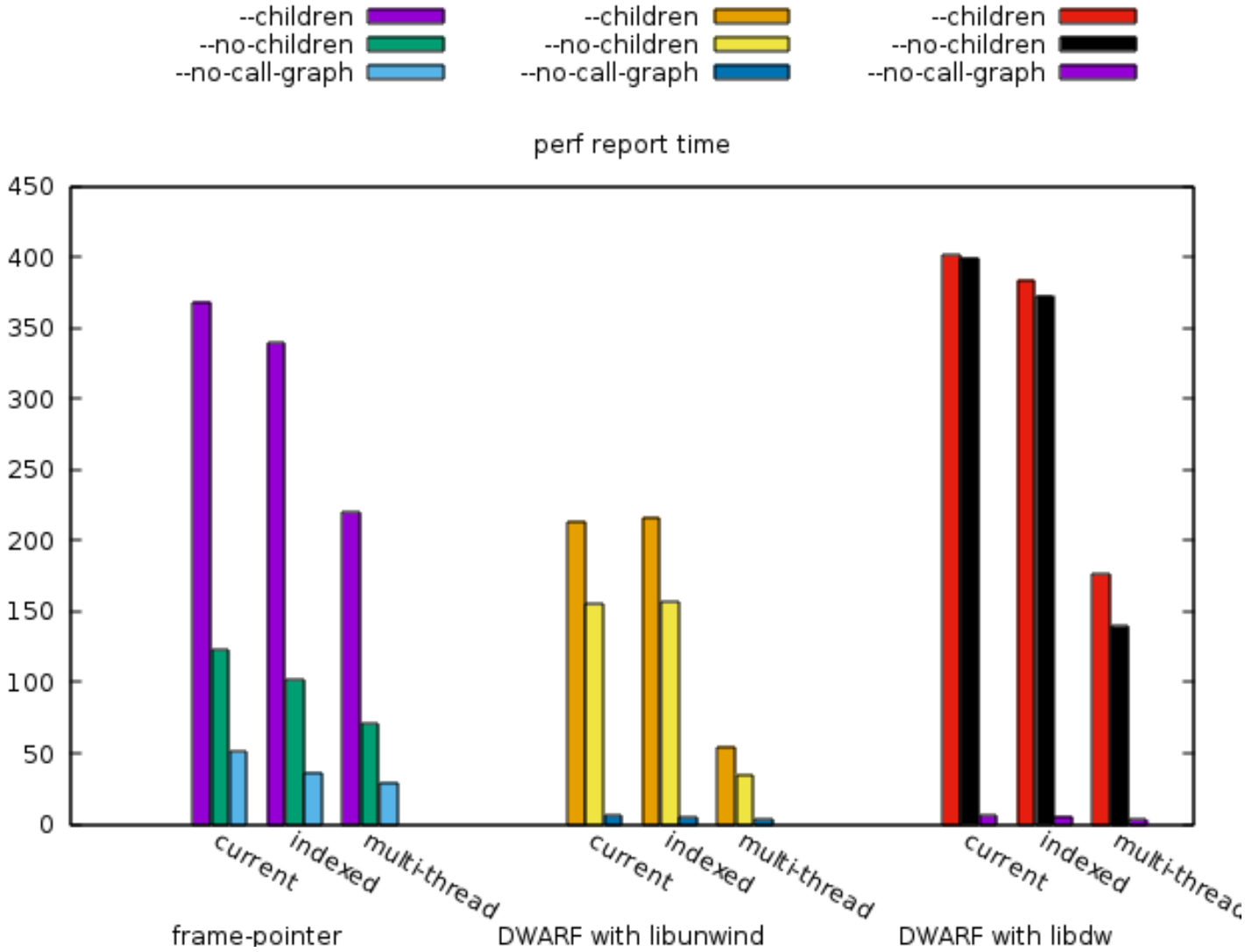
processing callchains

- for each callchain entry
  - locate (function) address from CFI (for DWARF callchains)
  - find map using address
  - find symbol using address
  - create callchain node
  - compare and insert/merge

collapsing per-thread entries

- samples are processed by multi-thread
- hist entries are kept per-thread for performance
- it merges hist entries after processing finished
- same as collapsing phase

performance result



Thanks