perf Linux profiler

CONCEPTS

perf stat counts events, perf record samples them: count first to see what is off, then sample to find where.

perf record samples events
↓ writes
perf.data sample file
↓ read by
report / script annotate, diff
↓ shows
Hot spots functions, stacks

Call graphs need frame pointers by default (-g). If stacks look bogus, use --call-graph dwarf.

On permission errors or missing kernel data, check kernel.perf_event_paranoid and try sudo.

Without a command perf stat and record measure all CPUs; end with Ctrl+C or append -- sleep <secs>.

COUNTING

`perf stat <cmd>`
Count default events for a command

`perf stat -r <n> <cmd>`
Repeat, show mean and stddev

`perf stat -e <ev>,<ev> <cmd>`
Count chosen events

`perf stat -p <pid> -- sleep <secs>`
Count a running process for a while

`perf stat -a -- sleep <secs>`
Count system-wide

`perf stat -d <cmd>`
Add cache and more detailed events

SAMPLING

`perf record -- <cmd>`
Sample a command into perf.data

SAMPLING (cont.)

`perf record -g -- <cmd>`
Sample with call stacks

`perf record -F <hz> -- <cmd>`
Set the sampling frequency

`perf record -a -g -- sleep <secs>`
Sample all CPUs for a while

`perf record -g -p <pid>`
Attach to a running process (Ctrl+C ends)

`perf record --call-graph dwarf -- <cmd>`
Stacks without frame pointers

`perf record -e <ev> -- <cmd>`
Sample on another event

REPORTING

`perf report`
Browse samples (TUI)

`perf report --stdio`
Plain text output

`perf report --no-children`
Self cost only, not inclusive

`perf report -s <keys>`
Group by comm, dso, symbol, cpu

`perf annotate <symbol>`
Per-instruction samples

LIVE VIEW

`perf top`
Live hottest functions

`perf top -p <pid>`
Only one process

EVENTS

`perf list`
List available events

`sudo sysctl kernel.perf_event_paranoid=-1`
Allow unprivileged events