Brendan Gregg's Blog   home

Linux perf_events Off-CPU Time Flame Graph

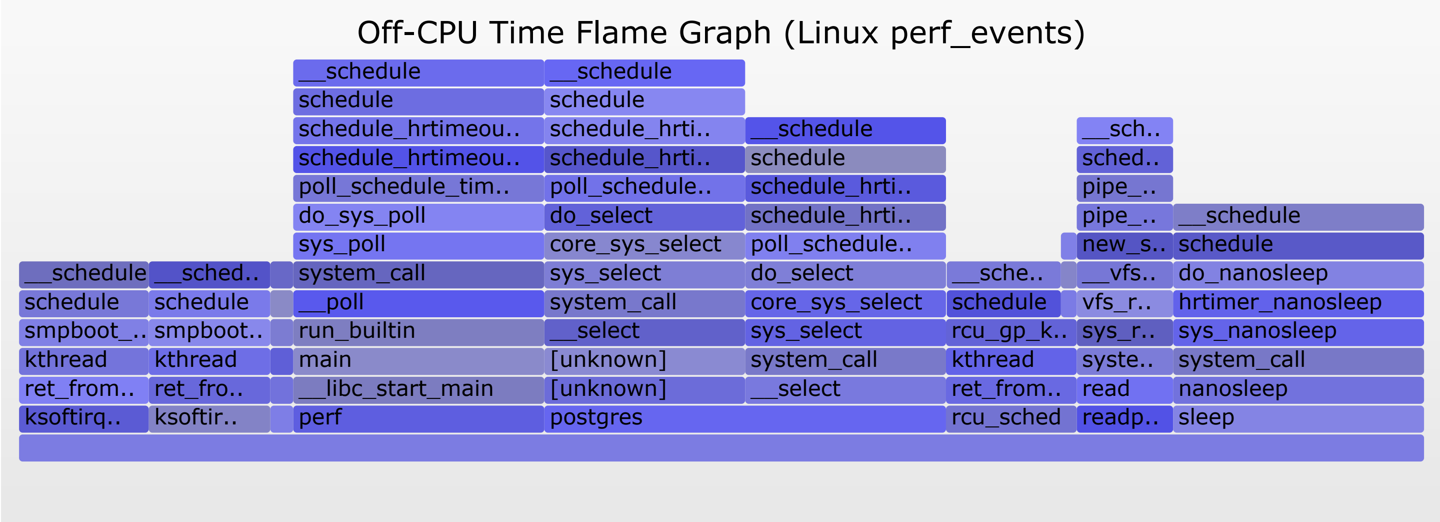
26 Feb 2015

I've been asked how to do these several times, so here's a quick blog post.

CPU Flame Graphs are great for understanding CPU usage, but what about performance issues of latency, when programs are blocked and not running on-CPU? There's a generic methodology to attack these, which I've called Off-CPU Analysis. This involves instrumenting call stacks and timestamps when the scheduler blocks a thread, and when it later wakes it up. There are even Off-CPU Time Flame Graphs to visualize this blocked time, which are the counterpart to on-CPU flame graphs.

Off-CPU time flame graphs may solve (say) 60% of the issues, with the remainder requiring walking the thread wakeups to find root cause. I explained off-CPU time flame graphs, this wakeup issue, and additional work, in my LISA13 talk on flame graphs (slides, youtube).

Here I'll show one way to do off-CPU time flame graphs using Linux perf_events. Example (click to zoom):



Unlike the CPU flame graph, in this graph the width spans the total duration that a code path was sleeping. A "sleep 1" command was caught, shown on the far right as having slept for 1,000 ms.

I was discussing how to do these with Nicolas on github, as he had found that perf_events could almost do this using the `perf inject -s` feature, when I noticed perf_events has added "-f period" to perf script (added in about 3.17). That simplifies things a bit, so the steps can be:

```
# perf record -e sched:sched_stat_sleep -e sched:sched_switch \
    -e sched:sched_process_exit -a -g -o perf.data.raw sleep 1
# perf inject -v -s -i perf.data.raw -o perf.data
# perf script -f comm,pid,tid,cpu,time,period,event,ip,sym,dso,trace | awk '
    NF > 4 { exec = $1; period_ms = int($5 / 1000000) }
    NF > 1 && NF <= 4 && period_ms > 0 { print $2 }
    NF < 2 && period_ms > 0 { printf "%s\n%d\n\n", exec, period_ms }' | \
    ./stackcollapse.pl | \
    ./flamegraph.pl --countname=ms --title="Off-CPU Time Flame Graph" --colors=io > offcpu.svg
```

Update: on more recent kernels, use "perf script -F ..." instead of "perf script -f ...". Your kernel will also need CONFIG_SCHEDSTATS for the tracepoints to all be present, which may be missing (eg, RHEL7).

The awk I'm using merely turns the perf script output into something stackcollapse.pl understands. This whole perf inject workflow strikes me as a bit weird, and this step could be skipped by doing your own processing of the perf script output (more awk!), to stitch together events. I'd do this if I were on older kernels, that lacked the perf script -f period.

Warning: scheduler events can be very frequent, and the overhead of dumping them to the file system (perf.data) may be prohibitive in production environments. Test carefully. This is also why I put a "sleep 1" in the perf record (the dummy command that sets the duration), to start with a small amount of trace data. If I had to do this in production, I'd consider other tools that could summarize data in-kernel to reduce overhead, including perf_events once it supports more in-kernel programming (eBPF).

There may be some scenarios where the current perf_events overhead is acceptable, especially if you match on a single PID of interest (in perf record, using "-p PID" instead of "-a").

Update 1: Since Linux 4.5 you may need the following for this to work:

```
# echo 1 > /proc/sys/kernel/sched_schedstats
```

Update 2: Since Linux 4.6 you can use BPF to do this *much* more efficiently: aggregating the stacks in-kernel context (using the BPF stack trace feature in 4.6), and only passing the summary to user-level. I developed the offcputime tool bcc to do this. I also wrote a post about it, Off-CPU eBPF Flame Graph, although that was written before Linux 4.6's stack trace support, so I was unwinding stacks the hard way.

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).

About this blog

Perf Methods
USE Method
TSA Method
Off-CPU Analysis
Active Bench.
Flame Graphs
Heat Maps
Frequency Trails
Colony Graphs
perf Examples
eBPF Tools
DTrace Tools
DTraceToolkit
DtkshDemos
Guessing Game
Specials
Books
Other Sites