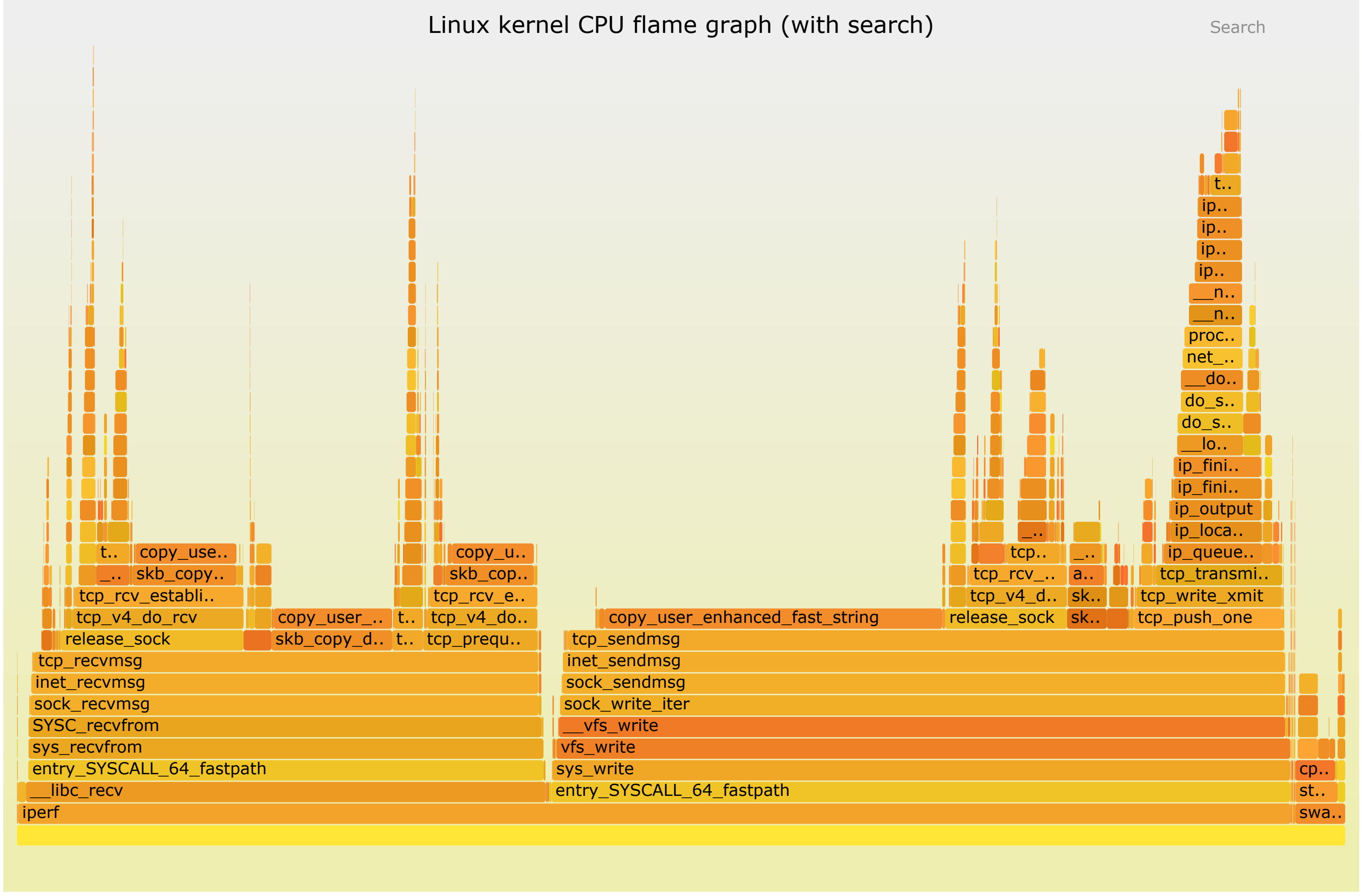
Flame Graph Search

11 Aug 2015

Click the "Search" button on the top right and enter "tcp_send":



You can also type Ctrl-F to search after selecting this SVG, or when on the [SVG link](#).

[Flame graphs](#) now have search! Under the large tcp_send() frame on the right we can see a write code-path. But what's those small tcp_send()s on the left, above a tcp_*recv*msg()? Think about why this could happen, then click the highlighted frames to zoom and see why.

Other interesting search terms are "re?cv", "spin", "copy", and "xen". This is a great way to learn Linux kernel internals, or the internals of any software.

The number of samples matching the search is shown on the bottom right as a percentage (this excludes vertical overlap). "tcp_send" was present in 59.3% of samples. I'm reminded of the following quote from [TCP-IP Digest, Vol 1 #6](#) (Nov 1981), by Bill Joy:

> As an experiment to investigate the performance of the resulting TCP/IP implementation ... the 11/750 is CPU saturated, but the 11/780 has about 30% idle time. The time spent in the system processing the data is spread out among handling for the Ethernet (20%), IP packet processing (10%), TCP processing (30%), checksumming (25%), and user system call handling (15%), with no single part of the handling dominating the time in the system.

Such breakdowns are incredibly useful, as they quantify performance and can be used to prioritize work, and estimate speed up from changes. But they can also be laborious to calculate from traditional profiles even today, over 30 years after Bill Joy's email. Flame graph search makes this much easier.

I had done rxNetty vs Tomcat analysis earlier this year, and needed to quantify functions in the CPU profiles. Flame graphs helped a lot, but it was still a bit time consuming to find and add up disparate frames. A number of us wanted a way to search for terms and get matched percentages, leading to this feature. Thorsten Lorenz had already implemented search last year in his [web flamegraph interface](#), as it is incredibly useful.

This actual flame graph here is from an iperf local benchmark, which was generated using:

```
$ perf record -F 99 -a -g -- sleep 30
$ perf script | gzip > out.perf01.gz
[... copy to local machine ...]
$ gzcat out.perf01.gz | ./stackcollapse-perf.pl | \
    sed '/cpu_idle/d;s/\[unknown\];//g' | \
    ./flamegraph.pl --width 1000 --hash \
    --title "Linux kernel CPU flame graph (with search)" > flamegraph_search.svg
```

The flame graph software is on github. I'm using sed(1) to remove CPU idle stacks, and to remove the "[unknown]" frames, which exist as I'm profiling a stripped and optimized version of iperf. If I want to see user-level stacks as well, I either need to rebuild iperf without -fno-omit-frame-pointer, or use perf's other stack walkers (libunwind) with iperf debuginfo. I didn't here, as I was just interested in the kernel CPU profile.

I also wrote about mixed-mode Java flame graphs recently on the Netflix Tech Blog.

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).