

Debugging Linux issues with eBPF

One incident from start to finish with dynamic tracing applied

Ivan Babrou
Performance @ Cloudflare

What does Cloudflare do



CDN

Moving content physically closer to visitors with our CDN.

Intelligent caching

Unlimited DDOS mitigation

Unlimited bandwidth at flat pricing with free plans

Edge access control

IPFS gateway

Onion service



Website Optimization

Making web fast and up to date for everyone.

TLS 1.3 (with 0-RTT)

HTTP/2 + QUIC

Server push

AMP

Origin load-balancing

Smart routing

Serverless / Edge Workers

Post quantum crypto



DNS

Cloudflare is the fastest managed DNS providers in the world.

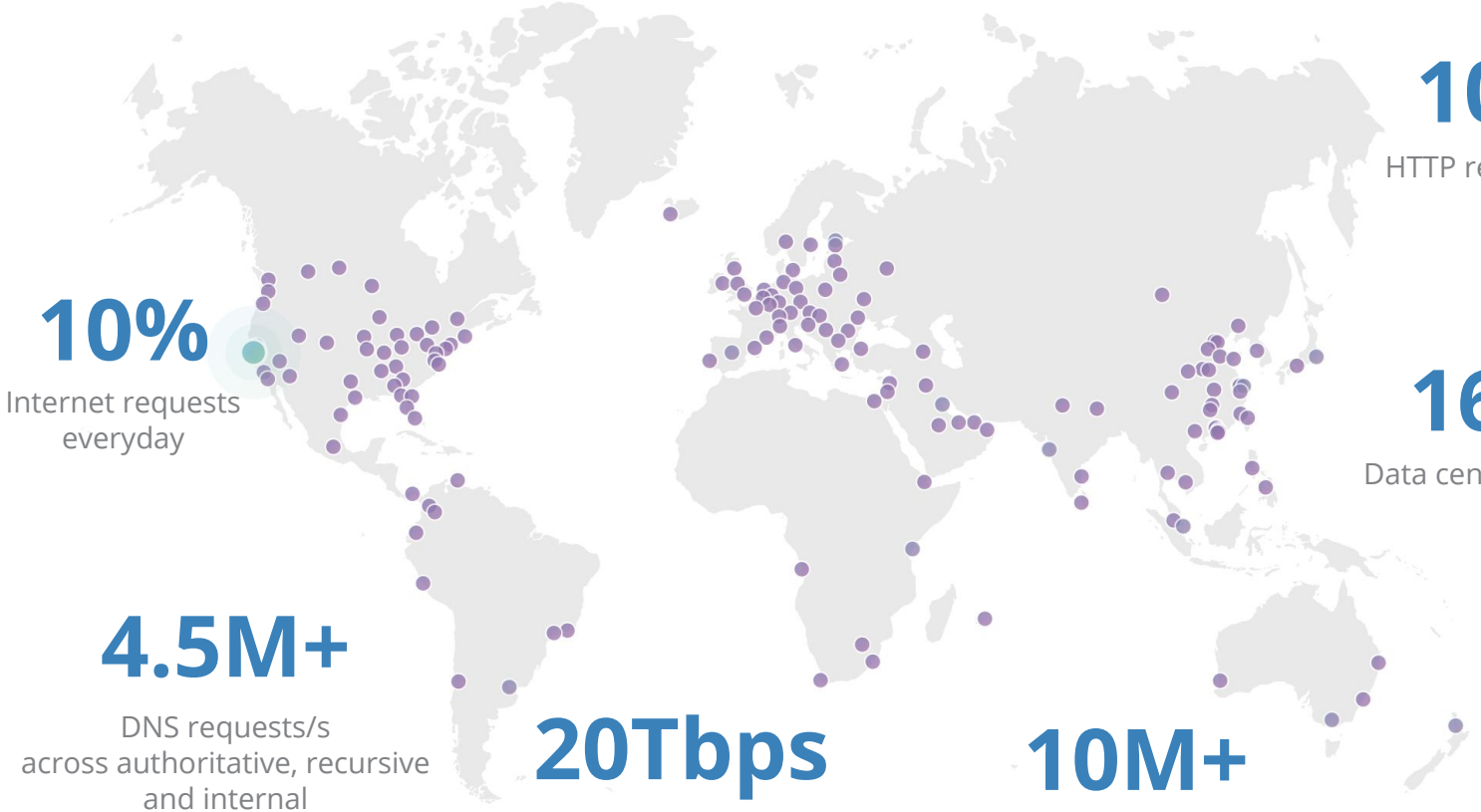
1.1.1.1

2606:4700:4700::1111

DNS over TLS

CLOUDFLARE

Cloudflare’s anycast network



10M+
HTTP requests/second

10%
Internet requests everyday

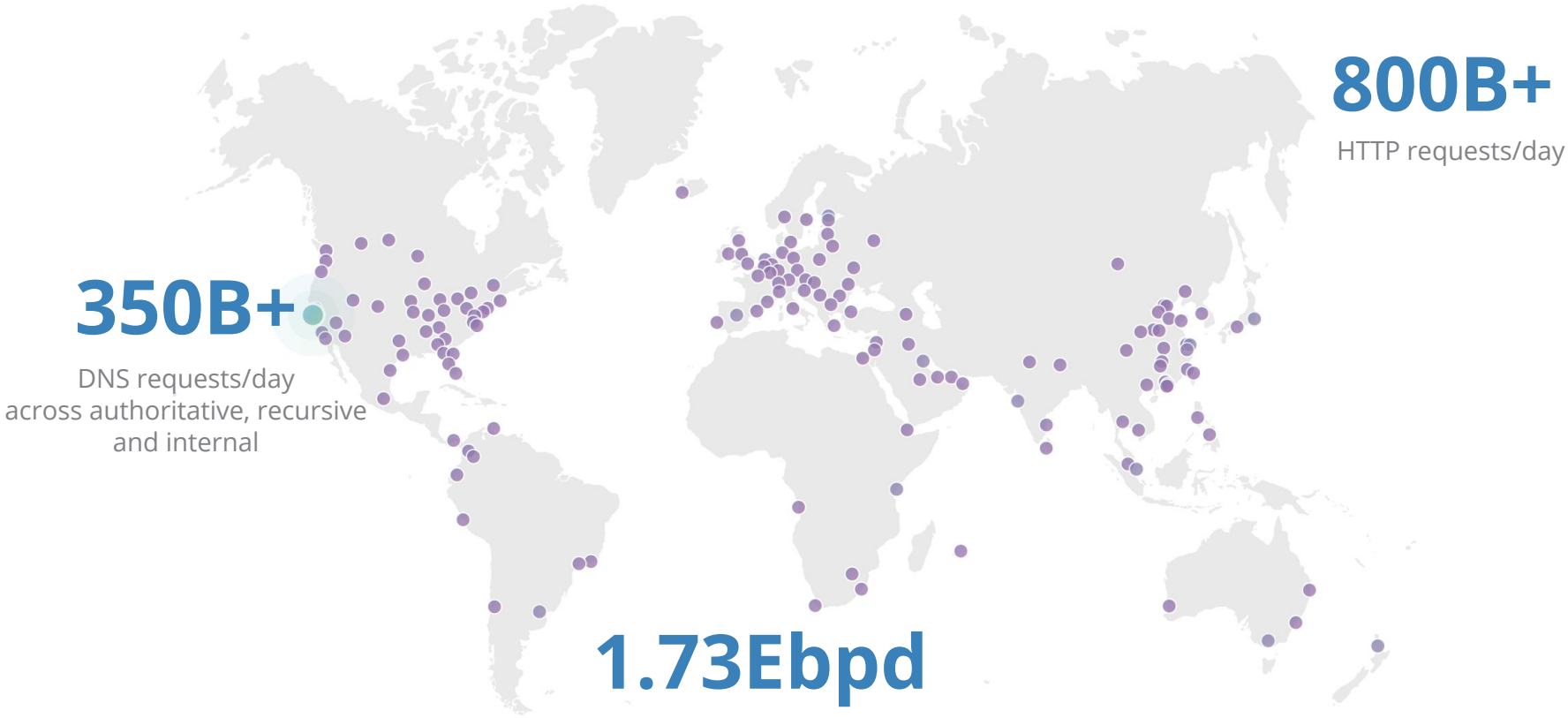
160+
Data centers globally

4.5M+
DNS requests/s across authoritative, recursive and internal

20Tbps
Network capacity

10M+
Websites, apps & APIs in 150 countries

Cloudflare’s anycast network (daily ironic numbers)



800B+

HTTP requests/day

350B+

DNS requests/day
across authoritative, recursive
and internal

1.73Ebpd

Network capacity

Link to slides with speaker notes

Slideshare doesn’t allow links on the first 3 slides

Cloudflare is a Debian shop

- All machines were running Debian Jessie on bare metal
- OS boots over PXE into memory, packages and configs are ephemeral
- Kernel can be swapped as easy as OS
- New Stable (stretch) came out, we wanted to keep up
- Very easy to upgrade:
  - Build all packages for both distributions
  - Upgrade machines in groups, look at metrics, fix issues, repeat
  - Gradually phase out Jessie
  - Pop a bottle of champagne and celebrate

Cloudflare core Kafka platform at the time

- Kafka is a distributed log with multiple producers and consumers
- 3 clusters: 2 small (dns + logs) with 9 nodes, 1 big (http) with 106 nodes
- 2 x 10C Intel Xeon E5-2630 v4 @ 2.2GHz (40 logical CPUs), 128GB RAM
- 12 x 800GB SSD in RAID0
- 2 x 10G bonded NIC
- Mostly network bound at ~100Gbps ingress and ~700Gbps egress
- Check out our blog post on Kafka compression
- We also blogged about our Gen 9 edge machines recently

Small clusters went ok, big one did not



Perf to the rescue: “perf top -F 99”

```
Samples: 5K of event 'cycles', Event count (approx.): 105435492870
  Children      Self  Shared Object       Symbol
+   70.36%     0.02%  [kernel]            [k] __splice_from_pipe
+   70.24%     0.06%  [kernel]            [k] tcp_sendpage
+   70.08%     0.27%  [kernel]            [k] tcp_sendmsg
+   58.87%     1.93%  [kernel]            [k] mod_timer
+   57.31%     0.05%  [kernel]            [k] sk_reset_timer
+   56.09%     0.03%  [kernel]            [k] do_syscall_64
+   54.57%     0.13%  [kernel]            [k] tcp_write_xmit
+   51.38%     0.02%  [kernel]            [k] sock_no_sendpage
+   47.52%     0.61%  [kernel]            [k] lock_timer_base
+   47.02%    47.02%  [kernel]            [k] queued_spin_lock_slowpath
+   47.00%     1.13%  [kernel]            [k] _raw_spin_lock_irqsave
+   44.02%     0.02%  [kernel]            [k] tcp_push_one
+   30.98%     0.05%  [kernel]            [k] tcp_schedule_loss_probe
+   28.09%     0.02%  [kernel]            [k] tcp_rearm_rto
+   24.92%     0.00%  [kernel]            [k] return_from_SYSCALL_64
+   23.92%     0.00%  libc-2.24.so        [.] sendfile
+   23.92%     0.00%  perf-30038.map      [.] 0x00007f649d861662
+   23.92%     0.00%  perf-30038.map      [.] 0x00007f649e15c510
+   23.92%     0.00%  [unknown]           [k] 0x00000001f8014d87
```

RCU stalls in dmesg

```
[ 4923.462841] INFO: rcu_sched self-detected stall on CPU
[ 4923.462843]  13-...: (2 GPs behind) idle=ea7/140000000000001/0 softirq=1/2 fqs=4198
[ 4923.462845]   (t=8403 jiffies g=110722 c=110721 q=6440)
```

CLOUDFLARE

Error logging issues

```
Aug 15 21:51:35 myhost kernel: INFO: rcu_sched detected stalls on CPUs/tasks:
Aug 15 21:51:35 myhost kernel:         26-...: (1881 ticks this GP) idle=76f/140000000000000/0
softirq=8/8 fqs=365
Aug 15 21:51:35 myhost kernel:         (detected by 0, t=2102 jiffies, g=1837293, c=1837292, q=262)
Aug 15 21:51:35 myhost kernel: Task dump for CPU 26:
Aug 15 21:51:35 myhost kernel: java            R  running task    13488  1714   1513 0x00080188
Aug 15 21:51:35 myhost kernel:  ffffc9000d1f7898 ffffffff814ee977 ffff88103f410400 000000000000000a
Aug 15 21:51:35 myhost kernel:  0000000000000041 ffffffff82203142 ffffc9000d1f78c0 ffffffff814eea10
Aug 15 21:51:35 myhost kernel:  0000000000000041 ffffffff82203142 ffff88103f410400 ffffc9000d1f7920
Aug 15 21:51:35 myhost kernel: Call Trace:
Aug 15 21:51:35 myhost kernel:  [<ffffffff814ee977>] ? scrup+0x147/0x160
Aug 15 21:51:35 myhost kernel:  [<ffffffff814eea10>] ? lf+0x80/0x90
Aug 15 21:51:35 myhost kernel:  [<ffffffff814eecb5>] ? vt_console_print+0x295/0x3c0
```

CLOUDFLARE

Page allocation failures

```
Aug 16 01:14:51 myhost systemd-journald[13812]: Missed 17171 kernel messages
Aug 16 01:14:51 myhost kernel:  [<ffffffff81171754>] shrink_inactive_list+0x1f4/0x4f0
Aug 16 01:14:51 myhost kernel:  [<ffffffff8117234b>] shrink_node_memcg+0x5bb/0x780
Aug 16 01:14:51 myhost kernel:  [<ffffffff811725e2>] shrink_node+0xd2/0x2f0
Aug 16 01:14:51 myhost kernel:  [<ffffffff811728ef>] do_try_to_free_pages+0xef/0x310
Aug 16 01:14:51 myhost kernel:  [<ffffffff81172be5>] try_to_free_pages+0xd5/0x180
Aug 16 01:14:51 myhost kernel:  [<ffffffff811632db>] __alloc_pages_slowpath+0x31b/0xb80
```

...

```
[78991.546088] systemd-network: page allocation stalls for 287000ms, order:0,
mode:0x24200ca(GFP_HIGHUSER_MOVABLE)
```

CLOUDFLARE

Downgrade and investigate

- System CPU was up, so it must be the kernel upgrade
- Downgrade Stretch to Jessie
- Downgrade Linux 4.9 to 4.4 (known good, but no allocation stall logging)
- Investigate without affecting customers
- Bisection pointed at OS upgrade, kernel was not responsible

Make a flamegraph with perf

```
#!/bin/sh -e
# flamegraph-perf [perf args here] > flamegraph.svg

# Explicitly setting output and input to perf.data is needed to make perf work over ssh without TTY.
perf record -o perf.data "$@"

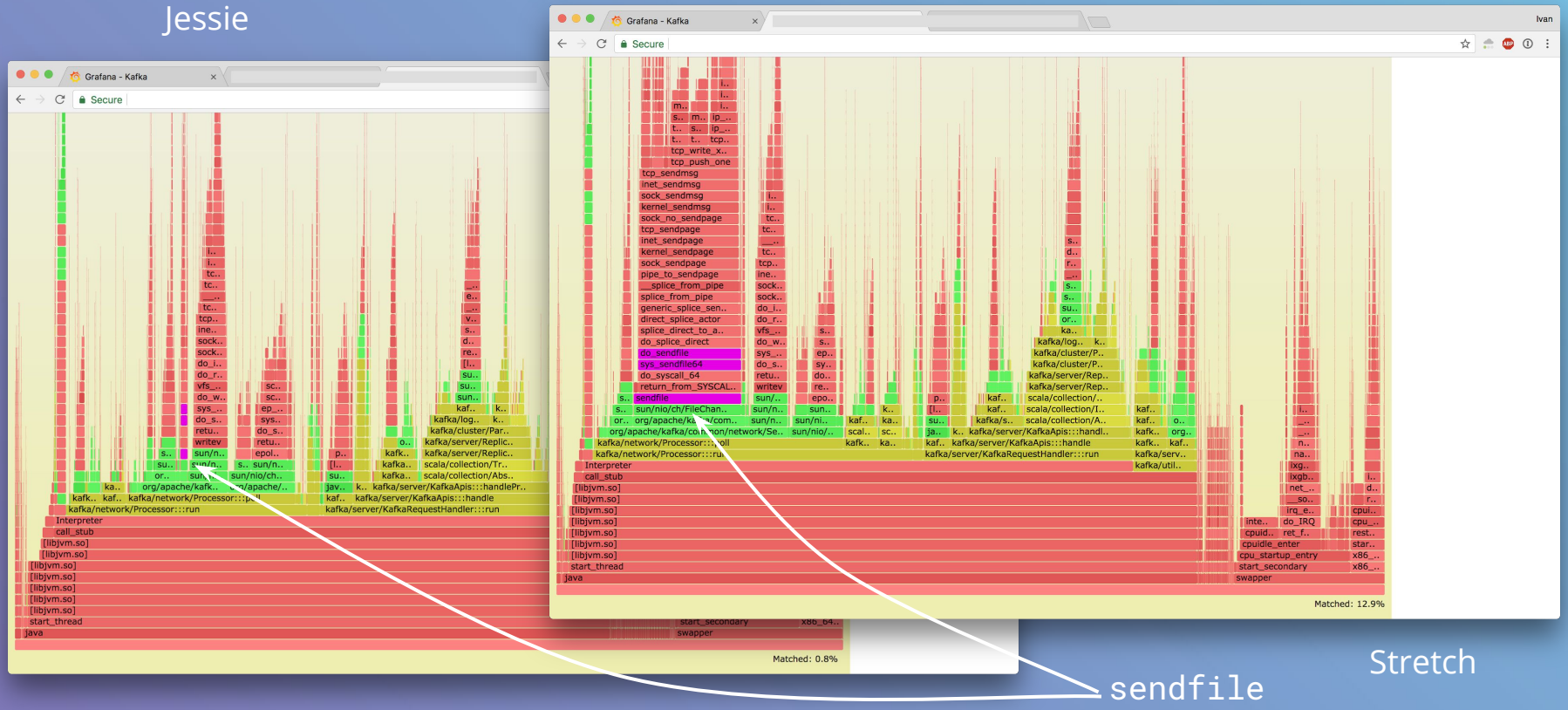
# Fetch JVM stack maps if possible, this requires -XX:+PreserveFramePointer
export JAVA_HOME=/usr/lib/jvm/oracle-java8-jdk-amd64 AGENT_HOME=/usr/local/perf-map-agent
/usr/local/flamegraph/jmaps 1>&2

IDLE_REGEXPS="^swapper;.*(cpuidle|cpu_idle|cpu_bringup_and_idle|native_safe_halt|xen_hypercall_sched_op|xen_hypercall_vcpu_op)"

perf script -i perf.data | /usr/local/flamegraph/stackcollapse-perf.pl --all grep -E -v "$IDLE_REGEXPS" | /usr/local/flamegraph/flamegraph.pl --colors=java --hash --title=$(hostname)
```

CLOUDFLARE

Full system flamegraphs point at sendfile

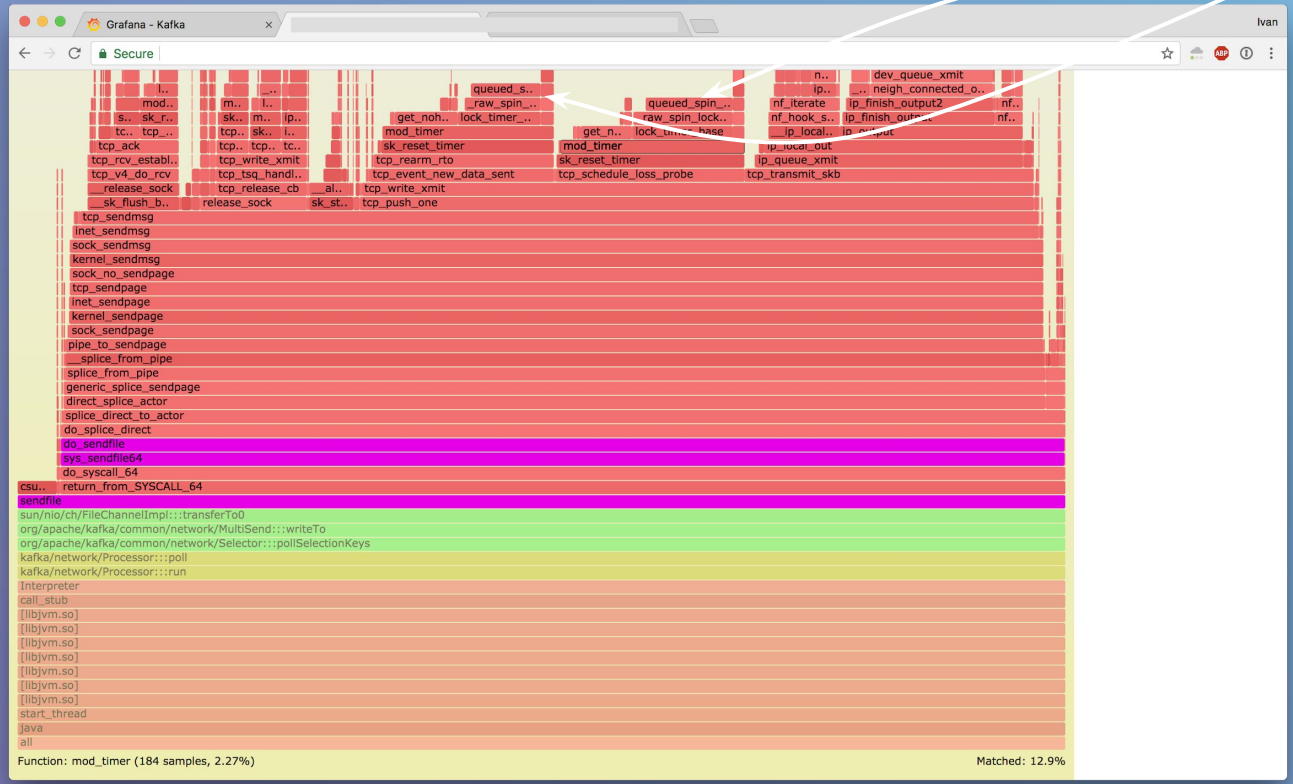


Enhance

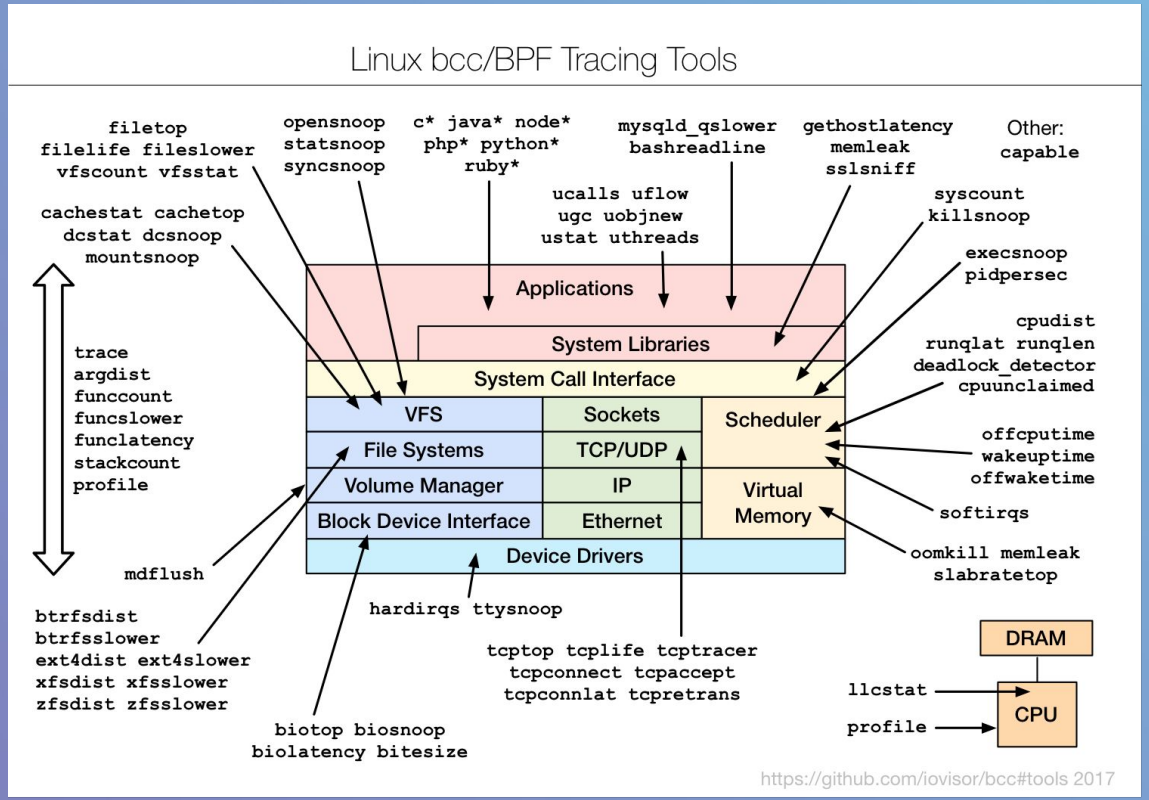


Stretch sendfile flamegraph

spinlocks



eBPF and BCC tools



Latency of sendfile on Jessie: < 31us

```
$ sudo /usr/share/bcc/tools/funclatency -uTi 1 do_sendfile
Tracing 1 functions for "do_sendfile"... Hit Ctrl-C to end.
23:27:25
     usecs               : count     distribution
         0 -> 1          : 9        |                                        |
         2 -> 3          : 47       |****                                    |
         4 -> 7          : 53       |*****                                   |
         8 -> 15         : 379      |****************************************|
        16 -> 31         : 329      |**********************************      |
        32 -> 63         : 101      |**********                              |
        64 -> 127        : 23       |**                                      |
       128 -> 255        : 50       |*****                                   |
       256 -> 511        : 7        |                                        |
```

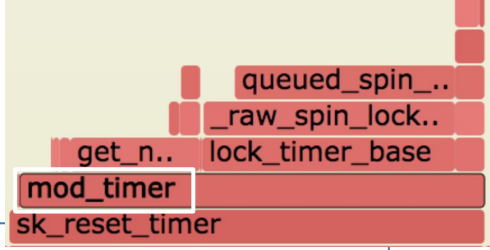
CLOUDFLARE

Latency of sendfile on Stretch: < 511us

```
     usecs               : count     distribution
         0 -> 1          : 1        |                                        |
         2 -> 3          : 20       |***                                     |
         4 -> 7          : 46       |*******                                 |
         8 -> 15         : 56       |********                                |
        16 -> 31         : 65       |**********                              |
        32 -> 63         : 75       |***********                             |
        64 -> 127        : 75       |***********                             |
       128 -> 255        : 258      |****************************************|
       256 -> 511        : 144      |**********************                  |
       512 -> 1023       : 24       |***                                     |
      1024 -> 2047       : 27       |****                                    |
      2048 -> 4095       : 28       |****                                    |
      4096 -> 8191       : 35       |*****                                   |
```

CLOUDFLARE

Number of mod_timer runs

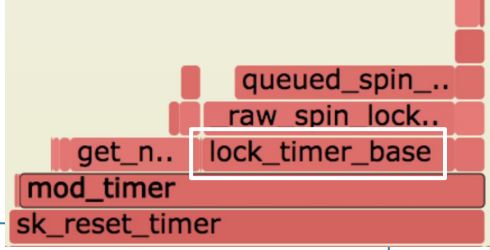



```
# Jessie
$ sudo /usr/share/bcc/tools/funccount -T -i 1 mod_timer
Tracing 1 functions for "mod_timer"... Hit Ctrl-C to end.
00:33:36
FUNC                                    COUNT
mod_timer                               60482
00:33:37
FUNC                                    COUNT
mod_timer                               58263
00:33:38
FUNC                                    COUNT
mod_timer                               54626
```

```
# Stretch
$ sudo /usr/share/bcc/tools/funccount -T -i 1 mod_timer
Tracing 1 functions for "mod_timer"... Hit Ctrl-C to end.
00:33:28
FUNC                                    COUNT
mod_timer                              149068
00:33:29
FUNC                                    COUNT
mod_timer                              155994
00:33:30
FUNC                                    COUNT
mod_timer                              160688
```

CLOUDFLARE

Number of lock_timer_base runs




```
# Jessie
$ sudo /usr/share/bcc/tools/funccount -T -i 1 lock_timer_base
Tracing 1 functions for "lock_timer_base"... Hit Ctrl-C to end.
00:32:36
FUNC                                    COUNT
lock_timer_base                         15962
00:32:37
FUNC                                    COUNT
lock_timer_base                         16261
00:32:38
FUNC                                    COUNT
lock_timer_base                         15806
```

```
# Stretch
$ sudo /usr/share/bcc/tools/funccount -T -i 1 lock_timer_base
Tracing 1 functions for "lock_timer_base"... Hit Ctrl-C to end.
00:32:32
FUNC                                    COUNT
lock_timer_base                        119189
00:32:33
FUNC                                    COUNT
lock_timer_base                        196895
00:32:34
FUNC                                    COUNT
lock_timer_base                        140085
```

CLOUDFLARE

We can trace timer tracepoints with perf

```
$ sudo perf list | fgrep timer:
  timer:hrtimer_cancel                               [Tracepoint event]
  timer:hrtimer_expire_entry                         [Tracepoint event]
  timer:hrtimer_expire_exit                          [Tracepoint event]
  timer:hrtimer_init                                 [Tracepoint event]
  timer:hrtimer_start                                [Tracepoint event]
  timer:itimer_expire                                [Tracepoint event]
  timer:itimer_state                                 [Tracepoint event]
  timer:tick_stop                                    [Tracepoint event]
  timer:timer_cancel                                 [Tracepoint event]
  timer:timer_expire_entry                           [Tracepoint event]
  timer:timer_expire_exit                            [Tracepoint event]
  timer:timer_init                                   [Tracepoint event]
  timer:timer_start                                  [Tracepoint event]
```

CLOUDFLARE

Number of timers per function

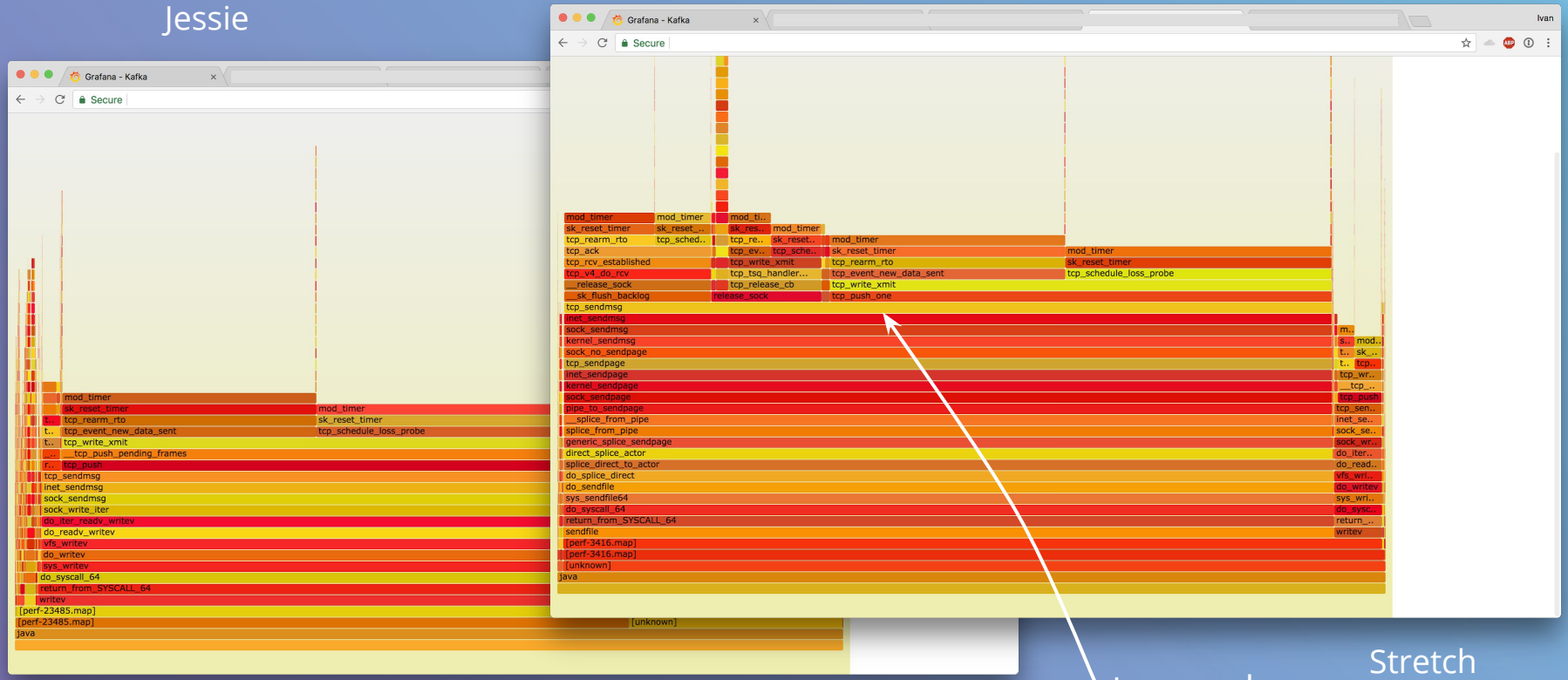
```
# Jessie
$ sudo perf record -e timer:timer_start -p 23485 -- sleep 10 && sudo perf script | sed 's/.* function=//g' | awk '{ print $1 }' | sort | uniq -c
[ perf record: Woken up 54 times to write data ]
[ perf record: Captured and wrote 17.778 MB perf.data (173520 samples) ]
      2 clocksource_watchdog
      5 cursor_timer_handler
      2 dev_watchdog
     10 garp_join_timer
      2 ixgbe_service_timer
   4769 tcp_delack_timer
    171 tcp_keepalive_timer
 168512 tcp_write_timer
```

```
# Stretch
$ sudo perf record -e timer:timer_start -p 3416 -- sleep 10 && sudo perf script | sed 's/.* function=//g' | awk '{ print $1 }' | sort | uniq -c
[ perf record: Woken up 671 times to write data ]
[ perf record: Captured and wrote 198.273 MB perf.data (1988650 samples) ]
      6 clocksource_watchdog
     12 cursor_timer_handler
      2 dev_watchdog
     18 garp_join_timer
      4 ixgbe_service_timer
   4622 tcp_delack_timer
      1 tcp_keepalive_timer
1983978 tcp_write_timer
```

CLOUDFLARE

Timer flamegraphs comparison

Jessie



tcp_push_one

Stretch

Number of calls for hot functions

```
# Jessie
$ sudo /usr/share/bcc/tools/funccount -T -i 1 tcp_sendmsg
Tracing 1 functions for "tcp_sendmsg"... Hit Ctrl-C to end.
03:33:33
FUNC                                    COUNT
tcp_sendmsg                             21166
$ sudo /usr/share/bcc/tools/funccount -T -i 1 tcp_push_one
Tracing 1 functions for "tcp_push_one"... Hit Ctrl-C to end.
03:37:14
FUNC                                    COUNT
tcp_push_one                              496
```

```
# Stretch
$ sudo /usr/share/bcc/tools/funccount -T -i 1 tcp_sendmsg
Tracing 1 functions for "tcp_sendmsg"... Hit Ctrl-C to end.
03:33:30
FUNC                                    COUNT
tcp_sendmsg                             53834
$ sudo /usr/share/bcc/tools/funccount -T -i 1 tcp_push_one
Tracing 1 functions for "tcp_push_one"... Hit Ctrl-C to end.
03:37:10
FUNC                                    COUNT
tcp_push_one                            64483
```

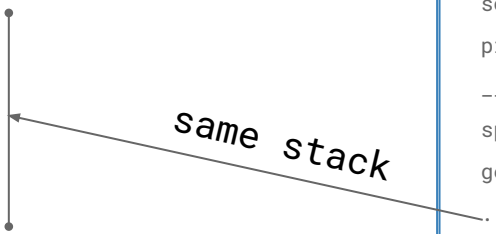
CLOUDFLARE

Count stacks leading to tcp_push_one

```
$ sudo stackcount -i 10 tcp_push_one
```

Stacks for tcp_push_one (stackcount)

```
tcp_push_one
inet_sendpage
kernel_sendpage
sock_sendpage
pipe_to_sendpage
__splice_from_pipe
splice_from_pipe
generic_splice_sendpage
direct_splice_actor
splice_direct_to_actor
do_splice_direct
do_sendfile
sys_sendfile64
do_syscall_64
return_from_SYSCALL_64
  4950
```

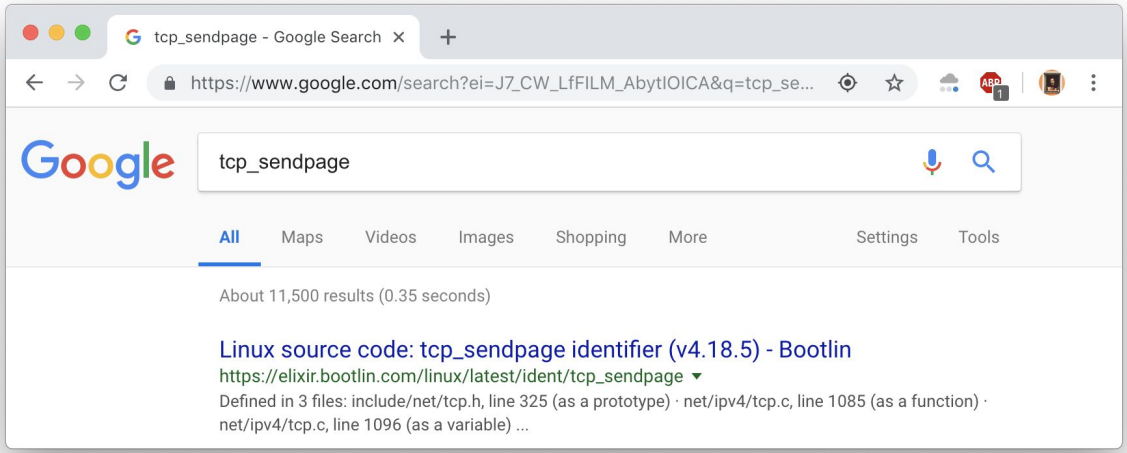



```
tcp_push_one
inet_sendmsg
sock_sendmsg
kernel_sendmsg
sock_no_sendpage
tcp_sendpage
inet_sendpage
kernel_sendpage
sock_sendpage
pipe_to_sendpage
__splice_from_pipe
splice_from_pipe
generic_splice_sendpage
...
return_from_SYSCALL_64
  735110
```

CLOUDFLARE

Diff of the most popular stack

```
--- jessie.txt  2017-08-16 21:14:13.000000000 -0700
+++ stretch.txt 2017-08-16 21:14:20.000000000 -0700
@@ -1,4 +1,9 @@
 tcp_push_one
+inet_sendmsg
+sock_sendmsg
+kernel_sendmsg
+sock_no_sendpage
+tcp_sendpage
 inet_sendpage
 kernel_sendpage
 sock_sendpage
```



CLOUDFLARE

Let's look at tcp_sendpage

```
int tcp_sendpage(struct sock *sk, struct page *page, int offset, size_t size, int flags) {
    ssize_t res;
    if (!(sk->sk_route_caps & NETIF_F_SG) ||
        !sk_check_csum_caps(sk))
        return sock_no_sendpage(sk->sk_socket, page, offset, size,
                    flags);

    lock_sock(sk);

    tcp_rate_check_app_limited(sk);  /* is sending application-limited? */
    res = do_tcp_sendpages(sk, page, offset, size, flags);
    release_sock(sk);
    return res;
}
```

segmentation offload

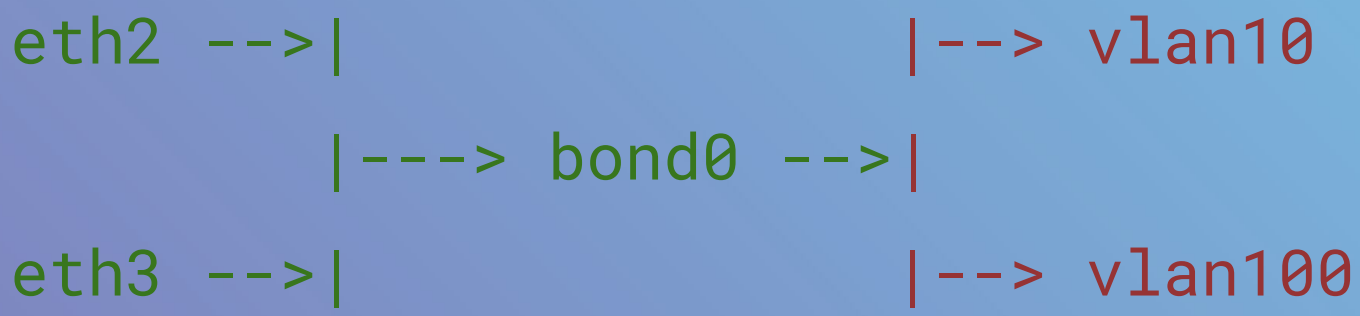
what we see on the stack

CLOUDFLARE

Cloudflare network setup



Missing offload settings



Compare ethtool -k settings on vlan10

```
-tx-checksumming: off
+tx-checksumming: on
-       tx-checksum-ip-generic: off
+       tx-checksum-ip-generic: on
-scatter-gather: off
-       tx-scatter-gather: off
+scatter-gather: on
+       tx-scatter-gather: on
```

```
-tcp-segmentation-offload: off
-       tx-tcp-segmentation: off [requested on]
-       tx-tcp-ecn-segmentation: off [requested on]
-       tx-tcp-mangleid-segmentation: off [requested on]
-       tx-tcp6-segmentation: off [requested on]
-udp-fragmentation-offload: off [requested on]
-generic-segmentation-offload: off [requested on]
+tcp-segmentation-offload: on
+       tx-tcp-segmentation: on
+       tx-tcp-ecn-segmentation: on
+       tx-tcp-mangleid-segmentation: on
+       tx-tcp6-segmentation: on
+udp-fragmentation-offload: on
+generic-segmentation-offload: on
```

CLOUDFLARE

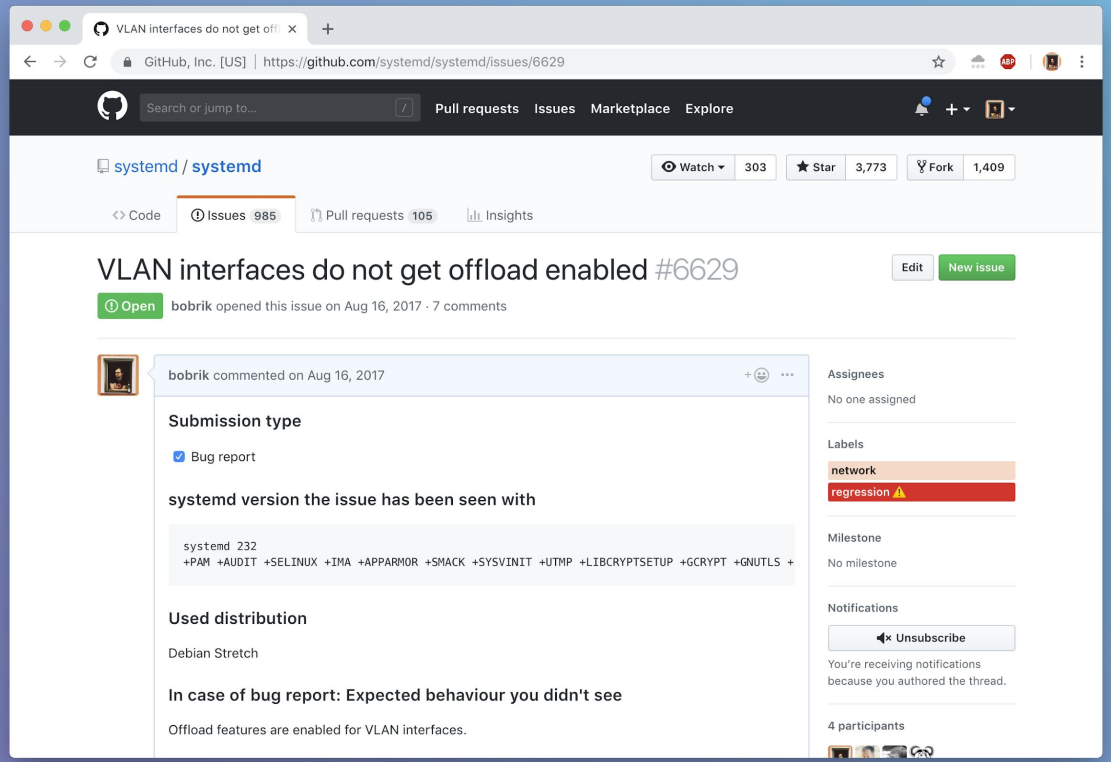
Ha! Easy fix, let's just enable it:

```
$ sudo ethtool -K vlan10 sg on
Actual changes:
tx-checksumming: on
        tx-checksum-ip-generic: on
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp-ecn-segmentation: on
        tx-tcp-mangleid-segmentation: on
        tx-tcp6-segmentation: on
udp-fragmentation-offload: on
```

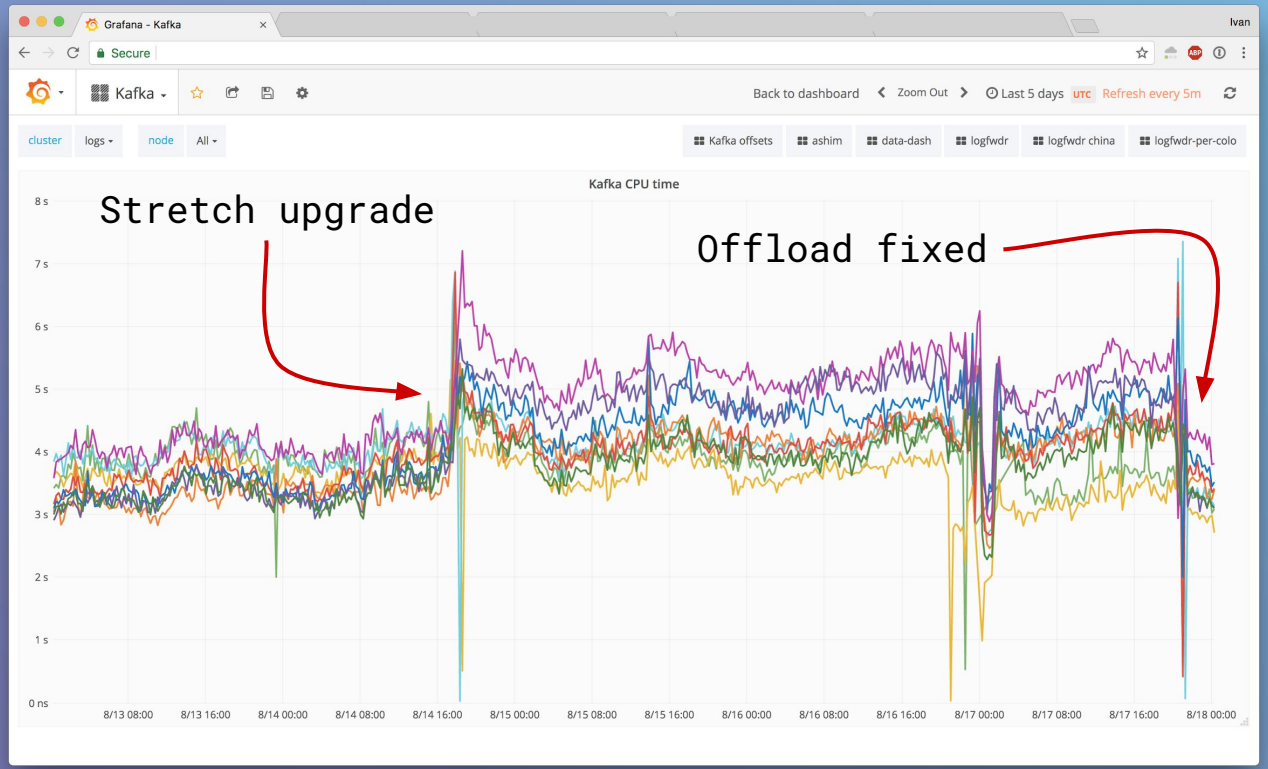
R in SRE stands for Reboot



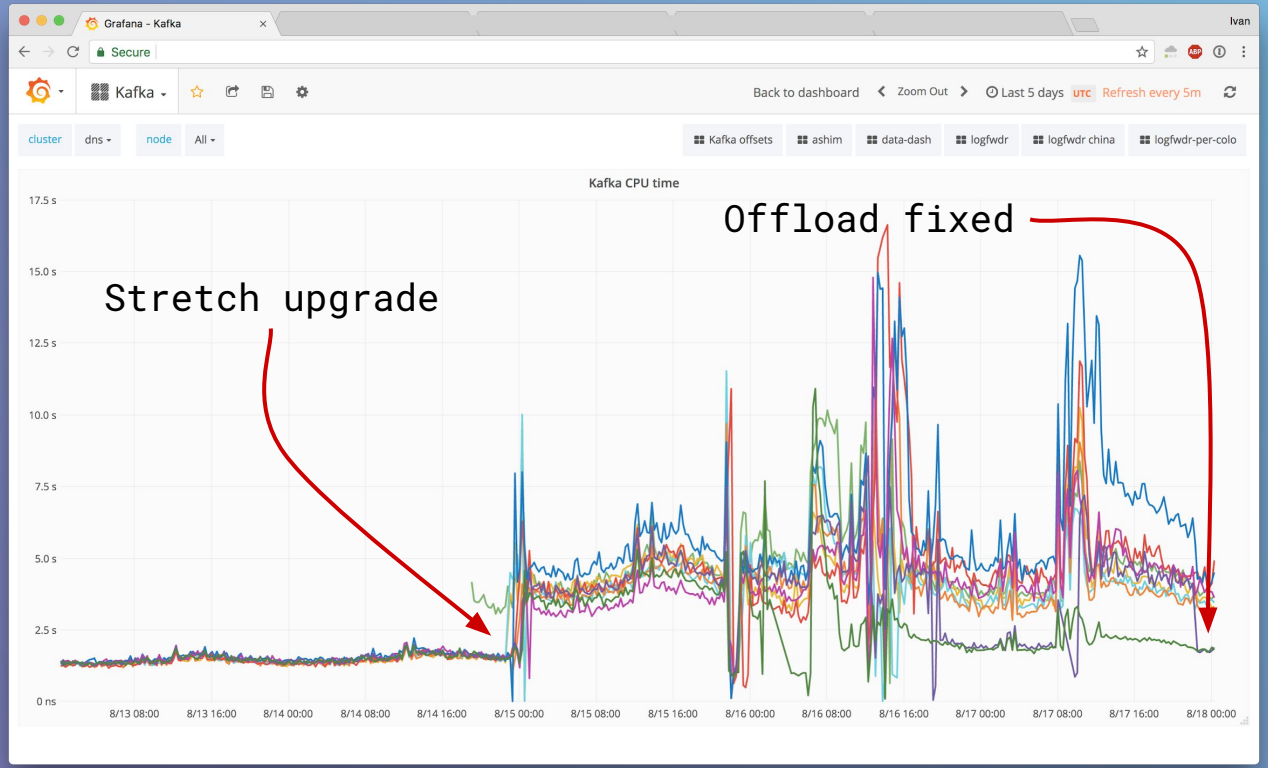
It was a bug in systemd all along



Logs cluster effect



DNS cluster effect



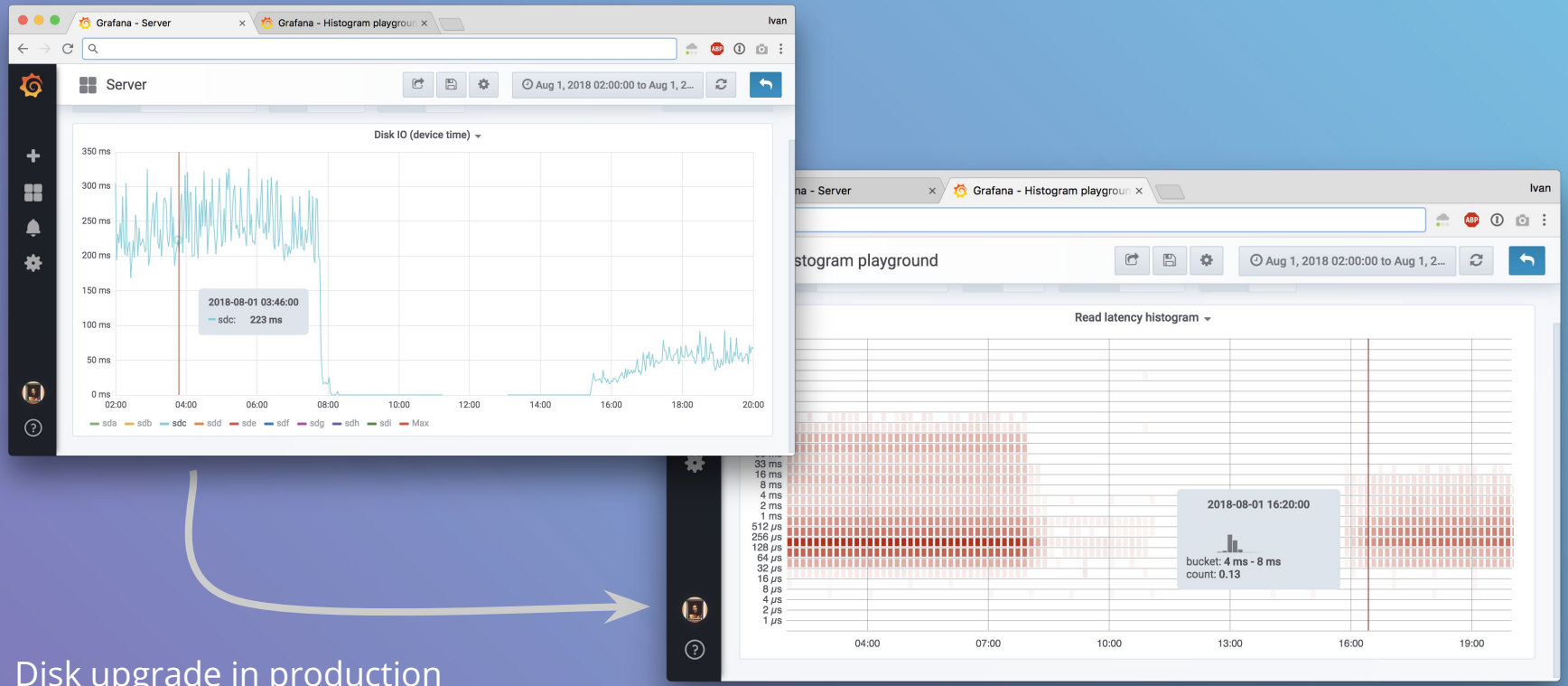
Lessons learned

- It’s important to pay closer attention and seemingly unrelated metrics
- Linux kernel can be easily traced with perf and bcc tools
  - Tools work out of the box
  - You don’t have to be a developer
- TCP offload is incredibly important and applies to vlan interfaces
- Switching OS on reboot proved to be useful

But really it was just an excuse

- Internal blog post about this is from Aug 2017
- External blog post in Cloudflare blog is from May 2018
- All to show where ebpf_exporter can be useful
  - Our tool to export hidden kernel metrics with eBPF
  - Can trace any kernel function and hardware counters
  - IO latency histograms, timer counters, TCP retransmits, etc.
  - Exports data in Prometheus (OpenMetrics) format

Can be nicely visualized with new Grafana



Disk upgrade in production

Thank you

- Blog post this talk is based on
- Github for ebpf_exporter: https://github.com/cloudflare/ebpf_exporter
- Slides for ebpf_exporter talk with presenter notes (and a blog post)
  - Disclaimer: contains statistical dinosaur gifs
- Training on ebpf_exporter with Alexander Huynh
  - Look for “Hidden Linux Metrics with Prometheus eBPF Exporter”
  - Wednesday, Oct 31st, 11:45 - 12:30, Cumberland room 3-4
- We’re hiring

Ivan on twitter: @ibobrik