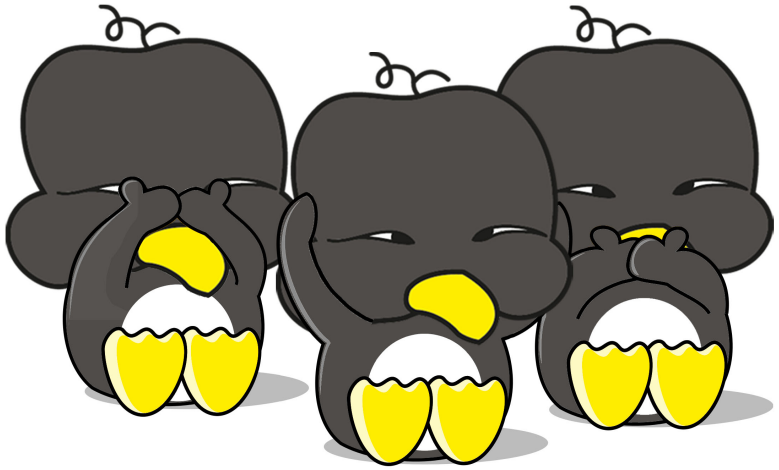
BPF Observability
NetConf 2018



Brendan Gregg

NETFLIX

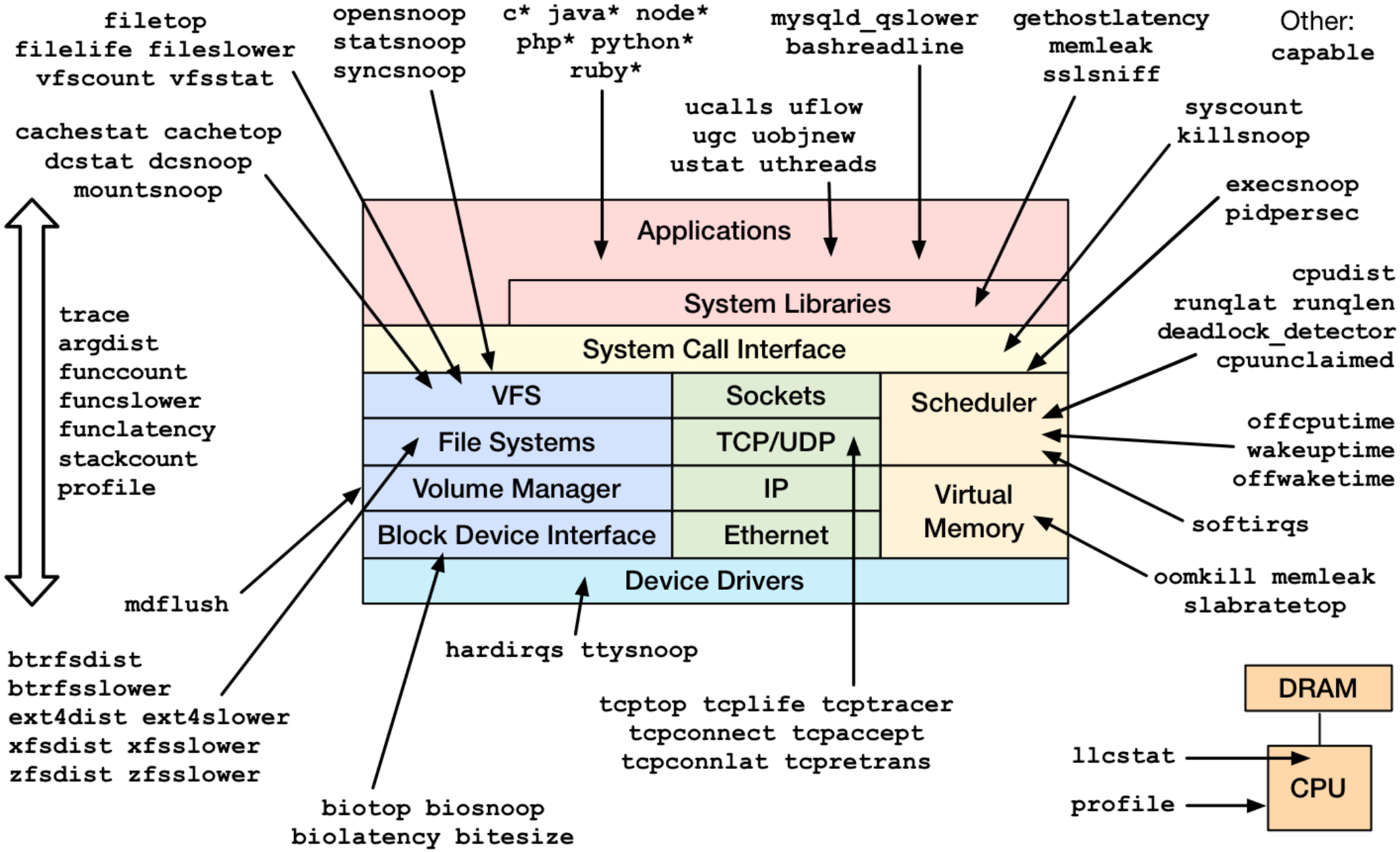
May 2018

Netflix & BPF

NETFLIX

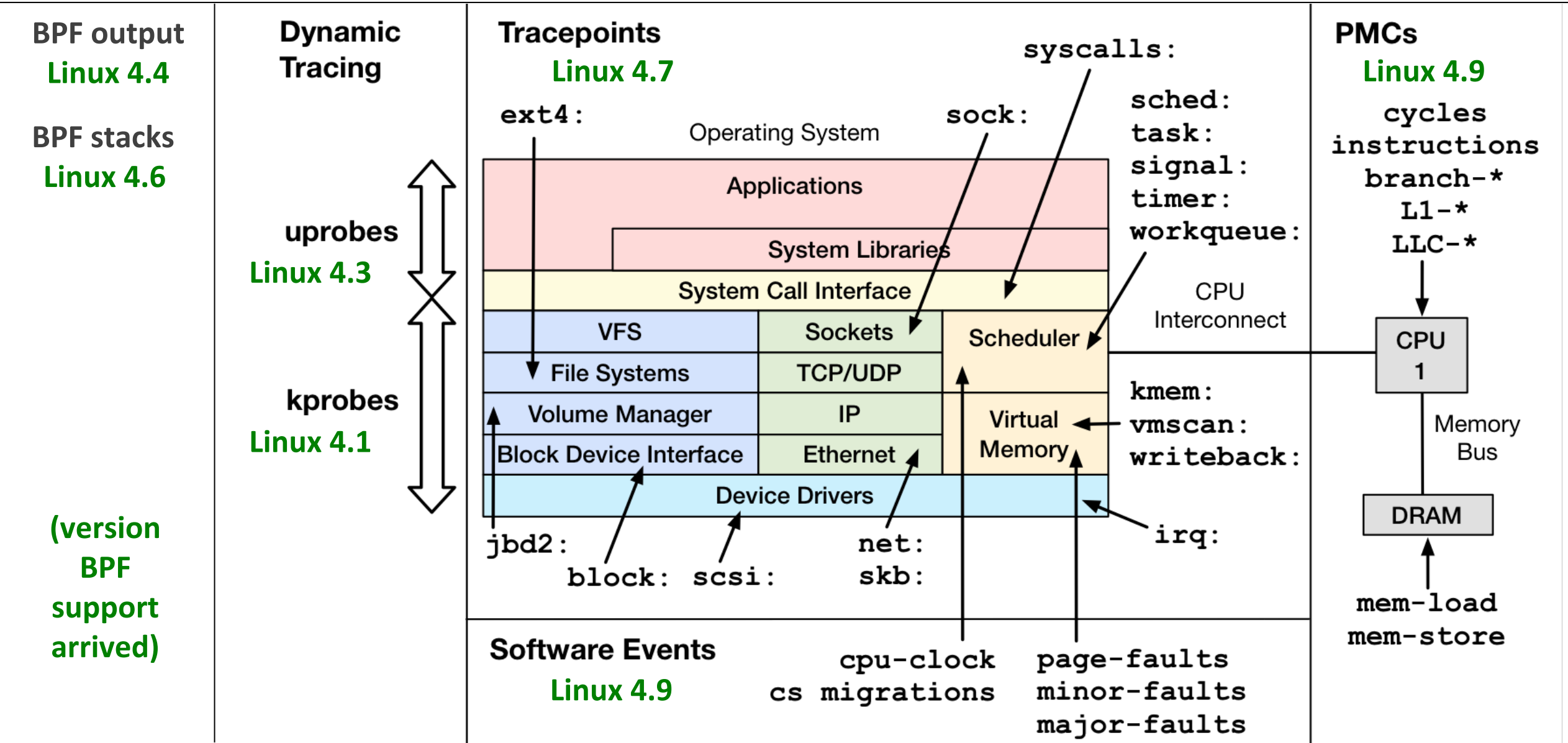
- **Performance Engineering**: observability
- **Network Engineering**: flow tracing, XDP
- **Security Team**: intrusion detection, whitelist generation
- **Container Team**: networking, security

eBPF bcc



https://github.com/iovisor/bcc

Linux Events & BPF Support



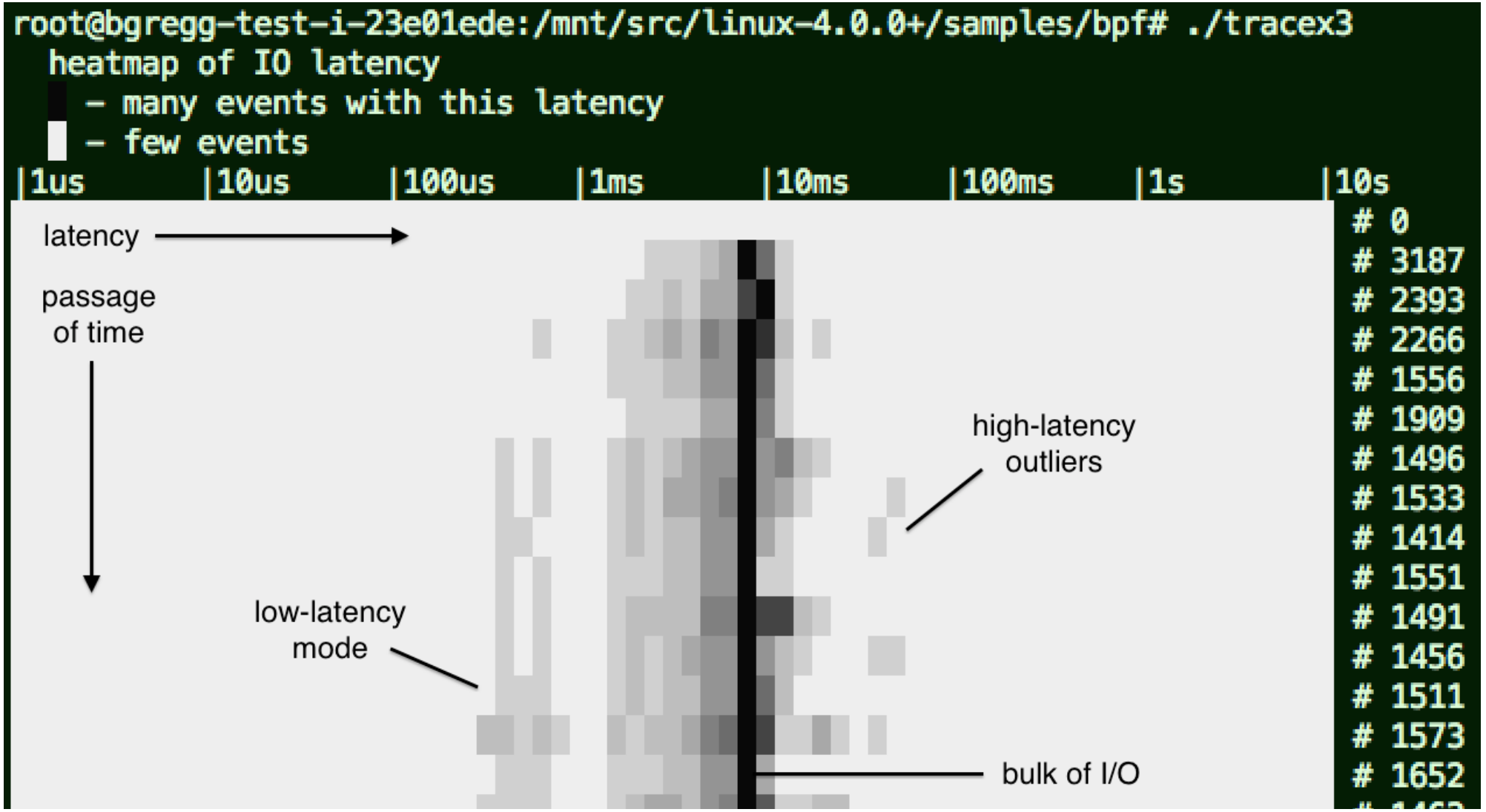
Basics like disk I/O latency histograms

```
# biolatency -mT 10
Tracing block device I/O... Hit Ctrl-C to end.

19:19:04
     msecs               : count     distribution
         0 -> 1          : 238      |*********                               |
         2 -> 3          : 424      |*****************                       |
         4 -> 7          : 834      |********************************        |
         8 -> 15         : 506      |*******************                     |
        16 -> 31         : 986      |**************************************|
        32 -> 63         : 97       |***                                     |
        64 -> 127        : 7        |                                        |
       128 -> 255        : 27       |*                                       |

19:19:14
     msecs               : count     distribution
         0 -> 1          : 427      |*****************                       |
         2 -> 3          : 424      |*****************                       |
[…]
```

… over time as a latency heat map



Basics like function counts

```
# ./funccount.py 'tcp*'
Tracing 334 functions for "tcp*"... Hit Ctrl-C to end.
^C
FUNC                                    COUNT
tcp_init_cwnd                               1
tcp_fastopen_active_disable_ofo_check       1
tcp_finish_connect                          1
tcp_write_queue_purge                       1
[...]
tcp_gro_receive                           155
tcp4_gro_receive                          155
tcp_chrono_stop                           197
tcp_release_cb                            209
tcp_schedule_loss_probe                   212
tcp_established_options                   269
tcp_v4_md5_lookup                         271
tcp_md5_do_lookup                         417
tcp_poll                                  572
tcp_stream_memory_free                    700
Detaching...
```

… I know

```
# echo 'tcp*' > /sys/kernel/debug/tracing/set_ftrace_filter
# echo 1 > /sys/kernel/debug/tracing/function_profile_enabled
# cat /sys/kernel/debug/tracing/trace_stat/function0
  Function                               Hit    Time            Avg             s^2
  --------                               ---    ----            ---             ---
  tcp_v4_rcv                             256    2218.794 us     8.667 us        21.572 us
  tcp_v4_do_rcv                          242    1557.531 us     6.436 us        14.847 us
  tcp_sendmsg                             87    1357.115 us     15.599 us       30.813 us
  tcp_rcv_established                    234    1309.926 us     5.597 us        7.151 us
  tcp_sendmsg_locked                      87    1273.319 us     14.635 us       27.509 us
  tcp_push                                87    845.834 us      9.722 us        14.658 us
  tcp_write_xmit                          88    808.439 us      9.186 us        13.235 us
  tcp_ack                                182    603.849 us      3.317 us        3.252 us
  tcp_transmit_skb                       112    513.866 us      4.588 us        4.871 us
  tcp_clean_rtx_queue                    146    298.300 us      2.043 us        0.480 us
  tcp_recvmsg                            109    295.443 us      2.710 us        1.768 us
  tcp4_gro_receive                       265    194.116 us      0.732 us        0.147 us
  tcp_v4_inbound_md5_hash                243    188.469 us      0.775 us        0.065 us
  tcp_data_queue                          50    186.623 us      3.732 us        13.787 us
  tcp_send_mss                            87    158.441 us      1.821 us        0.568 us
  tcp_established_options                197    142.160 us      0.721 us        0.046 us
[...]
```

… I know…

```
# cat /sys/kernel/debug/tracing/events/raw_syscalls/sys_enter/hist
    # trigger info: hist:keys=id.syscall:vals=hitcount:sort=hitcount:size=2048
[active]
[…]

    { id: sys_rt_sigprocmask                [ 14] } hitcount:        952
    { id: sys_futex                         [202] } hitcount:       1534
    { id: sys_write                         [  1] } hitcount:       2689
    { id: sys_setitimer                     [ 38] } hitcount:       2797
    { id: sys_read                          [  0] } hitcount:       3202
    { id: sys_select                        [ 23] } hitcount:       3773
    { id: sys_writev                        [ 20] } hitcount:       4531
    { id: sys_poll                          [  7] } hitcount:       8314
    { id: sys_recvmsg                       [ 47] } hitcount:      13738
    { id: sys_ioctl                         [ 16] } hitcount:      21843

Totals:
        Hits: 67612
        Entries: 72
        Dropped: 0
```

More advanced counts!

```
# tcptop
Tracing... Output every 1 secs. Hit Ctrl-C to end
<screen clears>
19:46:24 loadavg: 1.86 2.67 2.91 3/362 16681

PID    COMM         LADDR                 RADDR                  RX_KB  TX_KB
16648  16648        100.66.3.172:22       100.127.69.165:6684        1      0
16647  sshd         100.66.3.172:22       100.127.69.165:6684        0   2149
14374  sshd         100.66.3.172:22       100.127.69.165:25219       0      0
14458  sshd         100.66.3.172:22       100.127.69.165:7165        0      0
```

TCP session logging, via ~~tcp_set_state()~~ sock:inet_sock_set_state

```
# ./tcplife
PID   COMM         LADDR           LPORT RADDR           RPORT TX_KB RX_KB MS
22597 recordProg   127.0.0.1       46644 127.0.0.1       28527     0     0 0.23
3277  redis-serv   127.0.0.1       28527 127.0.0.1       46644     0     0 0.28
22598 curl         100.66.3.172    61620 52.205.89.26    80        0     1 91.79
22604 curl         100.66.3.172    44400 52.204.43.121   80        0     1 121.38
22624 recordProg   127.0.0.1       46648 127.0.0.1       28527     0     0 0.22
3277  redis-serv   127.0.0.1       28527 127.0.0.1       46648     0     0 0.27
22647 recordProg   127.0.0.1       46650 127.0.0.1       28527     0     0 0.21
3277  redis-serv   127.0.0.1       28527 127.0.0.1       46650     0     0 0.26
[…]
```

Kernel drop tracing via tcp_drop() … (written yesterday in the YOTEL Boston)

```
# ./tcpdrop.py
TIME     PID    IP SADDR:SPORT           > DADDR:DPORT           STATE (FLAGS)
20:49:06 0      4  10.32.119.56:443      > 10.66.65.252:22912    CLOSE (ACK)
        tcp_drop+0x1
        tcp_v4_do_rcv+0x135
        tcp_v4_rcv+0x9c7
        ip_local_deliver_finish+0x62
        ip_local_deliver+0x6f
        ip_rcv_finish+0x129
        ip_rcv+0x28f
        __netif_receive_skb_core+0x432
        __netif_receive_skb+0x18
        netif_receive_skb_internal+0x37
        napi_gro_receive+0xc5
        ena_clean_rx_irq+0x3c3
        ena_io_poll+0x33f
        net_rx_action+0x140
        __softirqentry_text_start+0xdf
        irq_exit+0xb6
        do_IRQ+0x82
[…]
```

tcp_drop

Was:

```
discard:
        __kfree_skb(skb);
}
```

Now:

```
discard:
        tcp_drop(sk, skb);
}
```

Future?:

```
discard:
        tcp_drop(sk, skb, reason);
}
```

tcp_stats (future?)

Now:

```
__NET_INC_STATS(sock_net(sk), LINUX_MIB_LISTENDROPS);
__TCP_INC_STATS(net, TCP_MIB_CSUMERRORS);
```

Future?:

```
tcp_stats(sk, LINUX_MIB_LISTENDROPS);
[…]
```

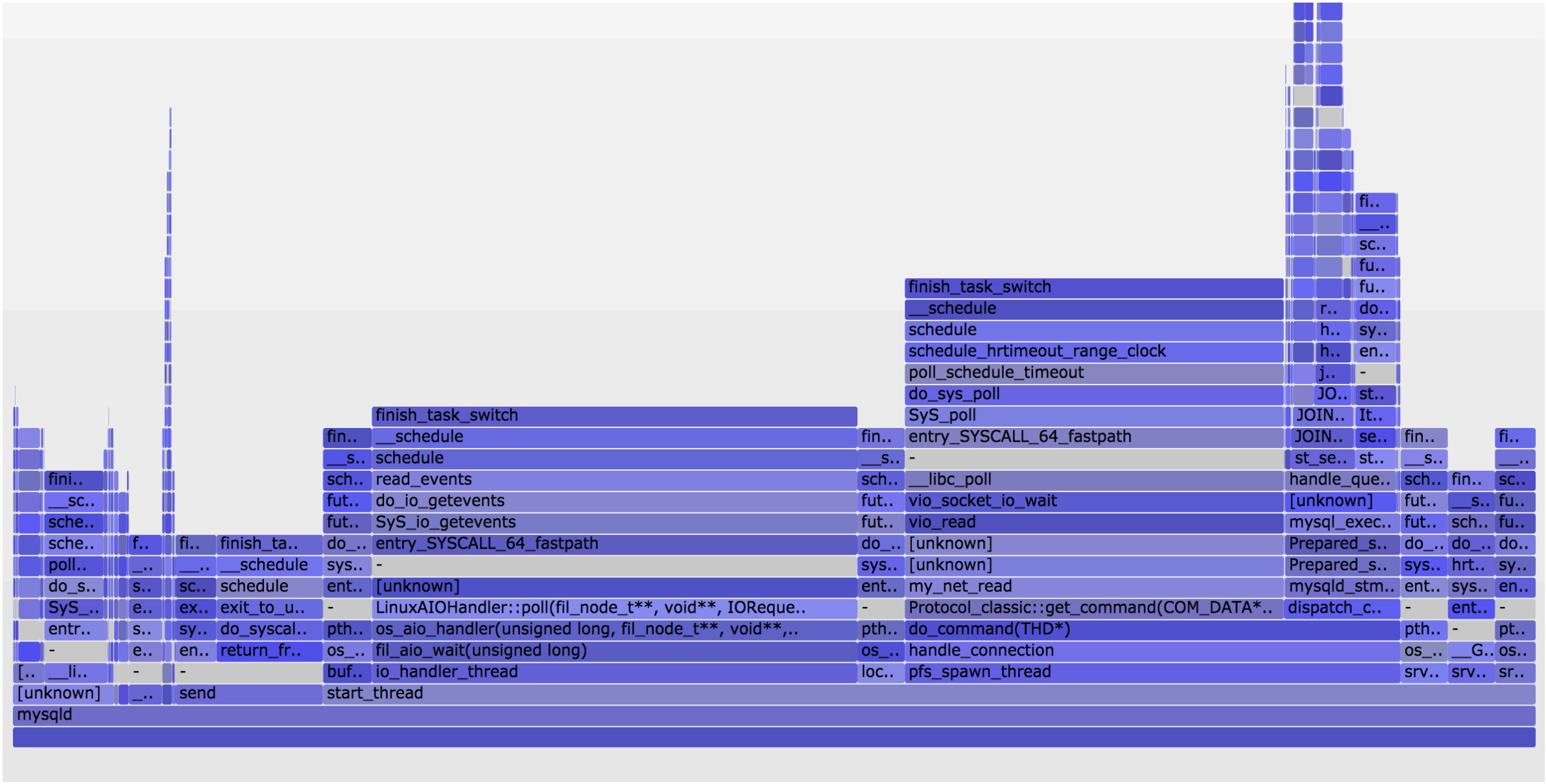
- for custom per-session stats

User-level instrumentation as well, eg, DNS lookups:

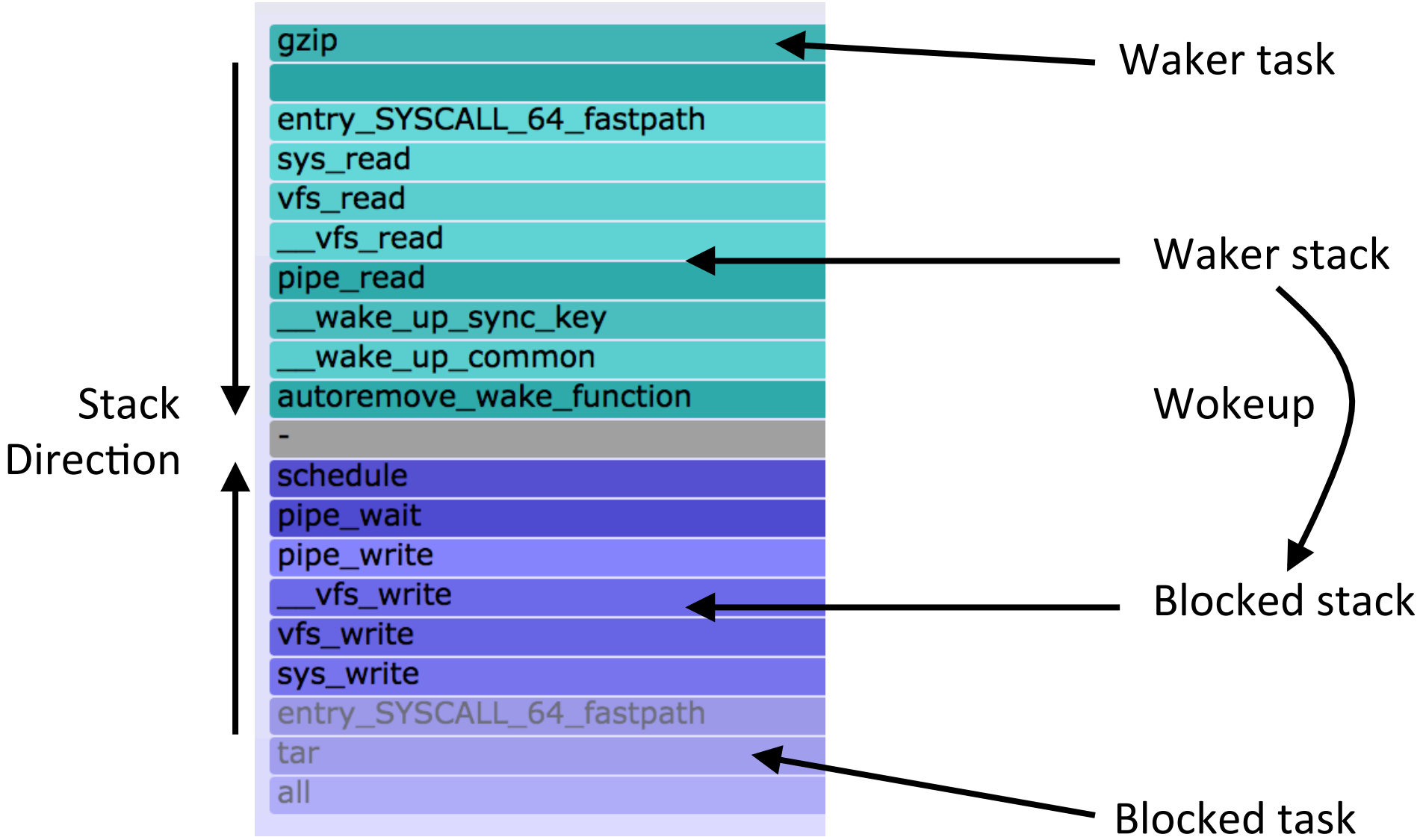
```
# /usr/share/bcc/tools/gethostlatency
TIME      PID    COMM                  LATms HOST
18:56:36  5055   mesos-slave            0.01 100.82.166.217
18:56:40  5590   java                   3.53 ec2-…-79.compute-1.amazonaws.com
18:56:51  5055   mesos-slave            0.01 100.82.166.217
18:56:53  30166  ncat                   0.21 localhost
18:56:56  6661   java                   2.19 atlas-alert-….prod.netflix.net
18:56:59  5589   java                   1.50 ec2-…-207.compute-1.amazonaws.com
18:57:03  5370   java                   0.04 localhost
18:57:03  30259  sudo                   0.07 titusagent-mainvpc-m…3465
18:57:06  5055   mesos-slave            0.01 100.82.166.217
18:57:10  5590   java                   3.10 ec2-…-79.compute-1.amazonaws.com
18:57:21  5055   mesos-slave            0.01 100.82.166.217
18:57:29  5589   java                  52.36 ec2-…-207.compute-1.amazonaws.com
18:57:36  5055   mesos-slave            0.01 100.82.166.217
18:57:40  5590   java                   1.83 ec2-…-79.compute-1.amazonaws.com
18:57:51  5055   mesos-slave            0.01 100.82.166.217
[…]
```

Instruments using user-level dynamic tracing of getaddrinfo(), gethostbyname(), etc.

Advanced things, like Off-CPU time flame graphs



Even more advanced: off-wake flame graphs



Chain graphs: merge all wakeup stacks

More networking tools todo

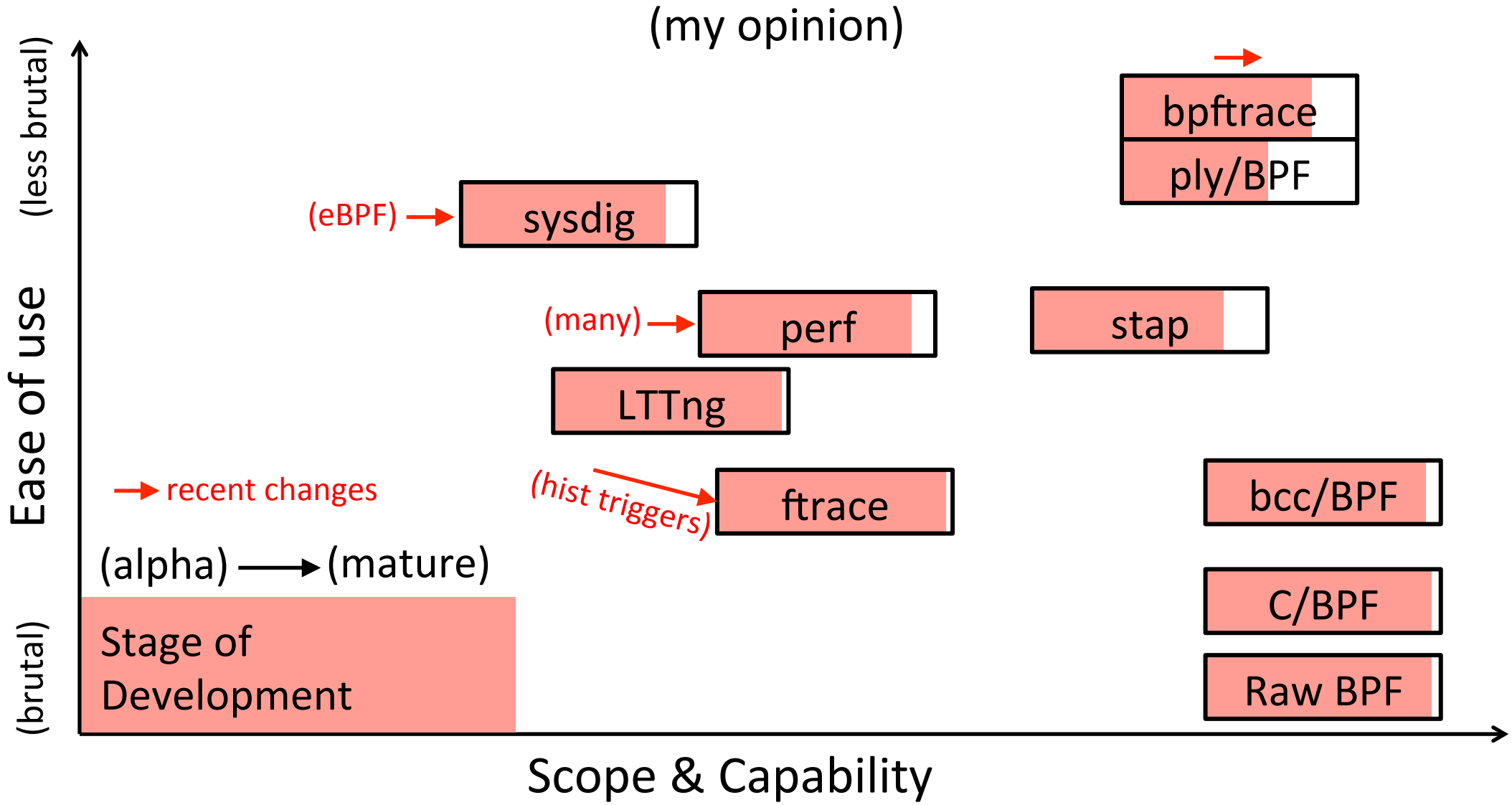
- TCP RTT histograms by host
- TCP buffer analysis
- TCP/IP inter-stack latency
- …

A Linux Tracing Timeline

- 1990's: Static tracers, prototype dynamic tracers
- 2000: LTT + DProbes (dynamic tracing; not integrated)
- 2004: kprobes (2.6.9)
- 2005: DTrace (not Linux), SystemTap (out-of-tree)
- 2008: ftrace (2.6.27)
- 2009: perf_events (2.6.31)
- 2009: tracepoints (2.6.32)
- 2010-2016: ftrace & perf_events enhancements
- 2012: uprobes (3.5)
- **2014-2018: enhanced BPF patches: supporting tracing events**
- 2016-2018: ftrace hist triggers
- **2018: ply or bpftrace?**

also: LTTng, ktap, sysdig, ...

The Tracing Landscape, May 2018



ply

https://github.com/iovisor/ply

```
# ply -A -c 'kprobe:SyS_read { @start[tid()] = nsecs(); }
    kretprobe:SyS_read /@start[tid()]/ { @ns.quantize(nsecs() - @start[tid()]);
        @start[tid()] = nil; }'
2 probes active
^Cde-activating probes
[...]
@ns:

     [ 512,   1k)           3 |########                                |
     [  1k,   2k)           7 |####################                    |
     [  2k,   4k)          12 |##################################      |
     [  4k,   8k)           3 |########                                |
     [  8k,  16k)           2 |#####                                   |
     [ 16k,  32k)           0 |                                        |
     [ 32k,  64k)           0 |                                        |
     [ 64k, 128k)           3 |########                                |
     [128k, 256k)           1 |###                                     |
     [256k, 512k)           1 |###                                     |
     [512k,   1M)           2 |#####                                   |
[...]
```

bpftrace

https://github.com/ajor/bpftrace

```
# bpftrace -e 'kprobe:SyS_read { @start[tid] = nsecs; } kretprobe:SyS_read /@start[tid]/
  { @ns = quantize(nsecs - @start[tid]); @start[tid] = delete(); }'
Attaching 2 probes...
^C

@ns:
[0, 1]                 0 |                                                    |
[2, 4)                 0 |                                                    |
[4, 8)                 0 |                                                    |
[8, 16)                0 |                                                    |
[16, 32)               0 |                                                    |
[32, 64)               0 |                                                    |
[64, 128)              0 |                                                    |
[128, 256)             0 |                                                    |
[256, 512)             0 |                                                    |
[512, 1k)              0 |                                                    |
[1k, 2k)               6 |@@@@@                                               |
[2k, 4k)              20 |@@@@@@@@@@@@@@@@@@@                                 |
[4k, 8k)               4 |@@@                                                 |
[8k, 16k)             14 |@@@@@@@@@@@@@                                       |
[16k, 32k)            53 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[32k, 64k)             2 |@                                                   |
```

BTF

- BPF Type Format
- Adding in Linux 4.18 (coming soon)
- Last bit needed for a bpftrace/ply high-level language that can walk structs like Kernel C

Resources

- https://github.com/iovisor/bcc
- http://www.brendangregg.com/ebpf.html
- http://www.brendangregg.com/blog/
- https://github.com/ajor/bpftrace
- https://github.com/iovisor/ply