27 Feb 2017

BPF TOOLS

Brendan Gregg
Senior Performance Architect
NETFLIX

image: http://makeitstranger.com

Observability

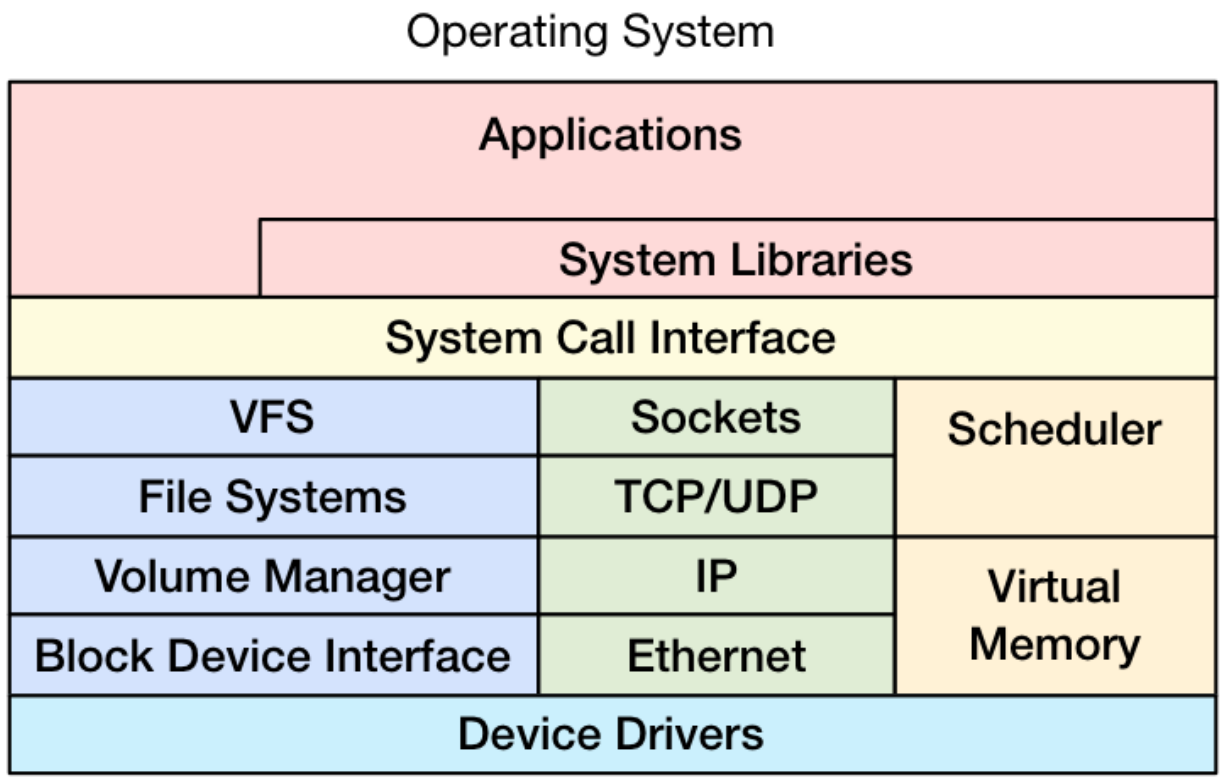
Best Possible Performance
&
Root Cause Analysis

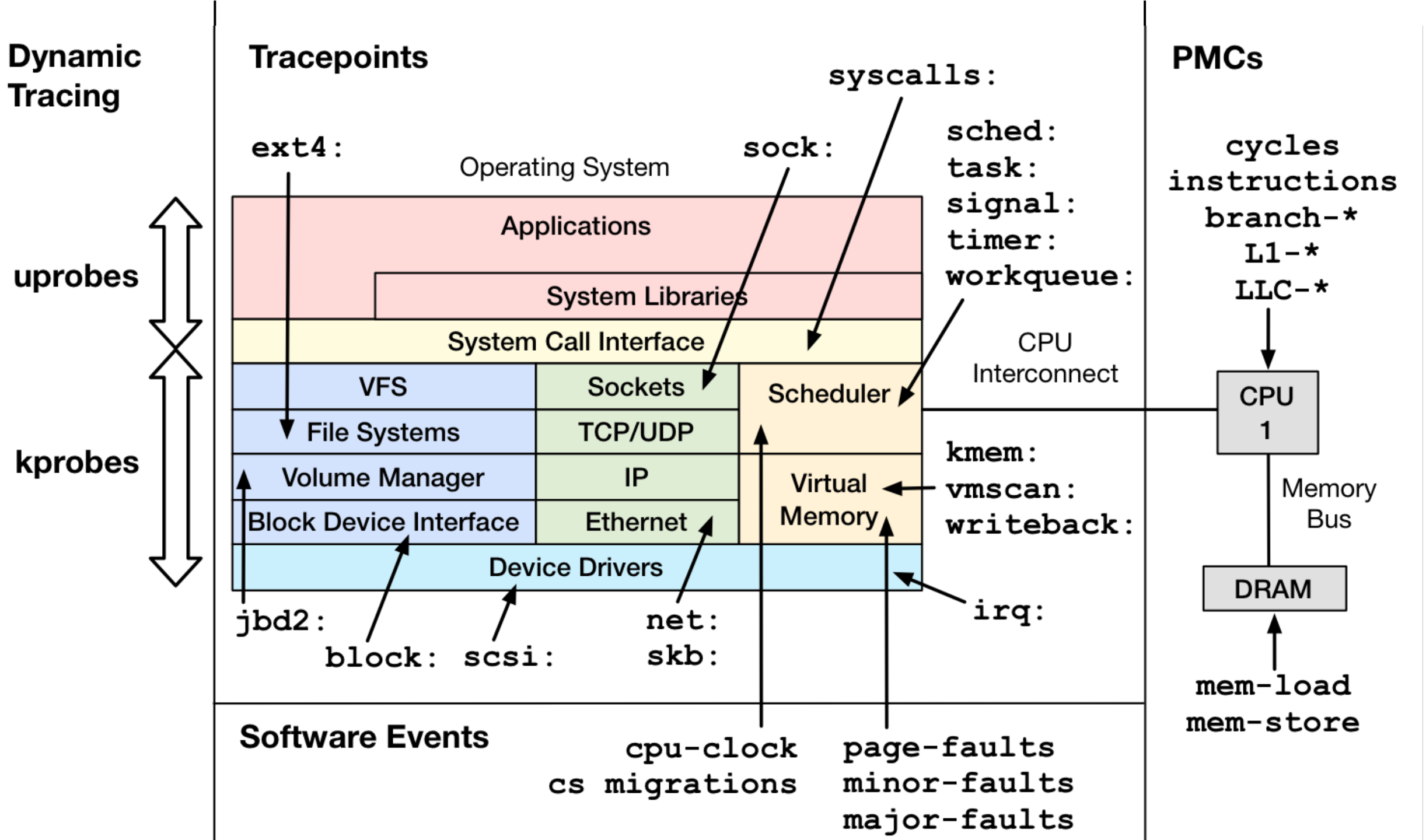
Needed:

Observe Everything

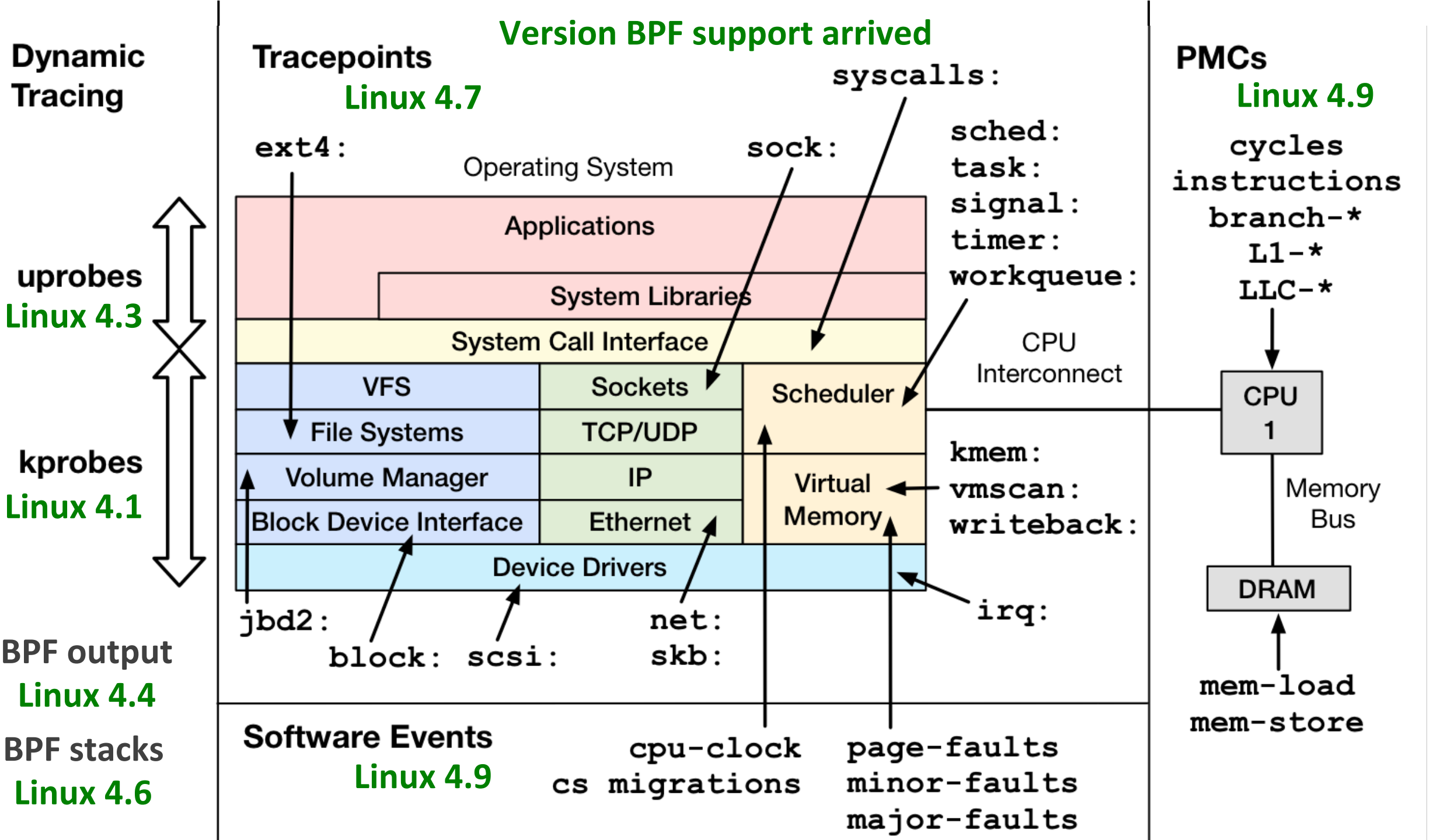
In Production

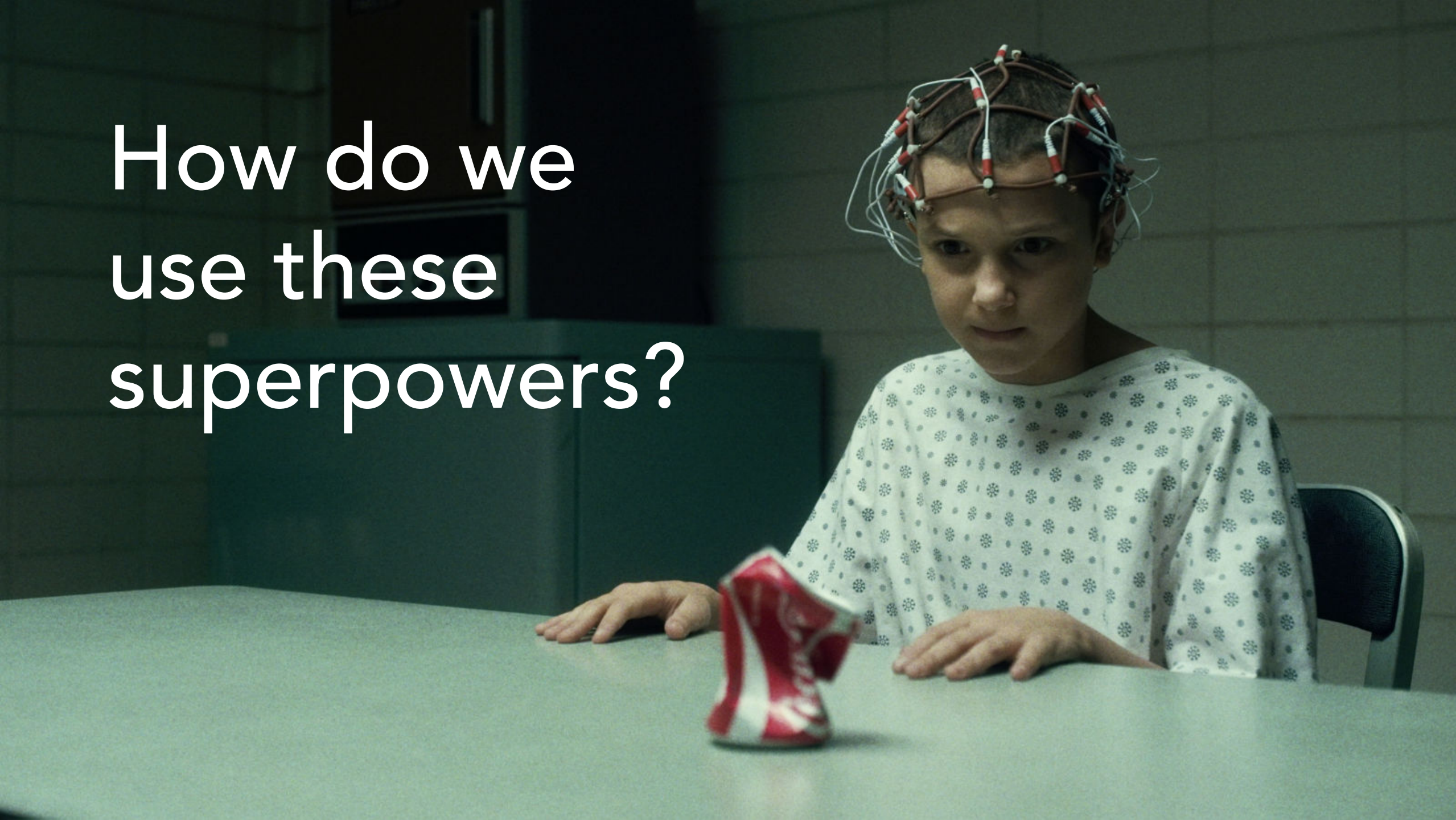
Quickly



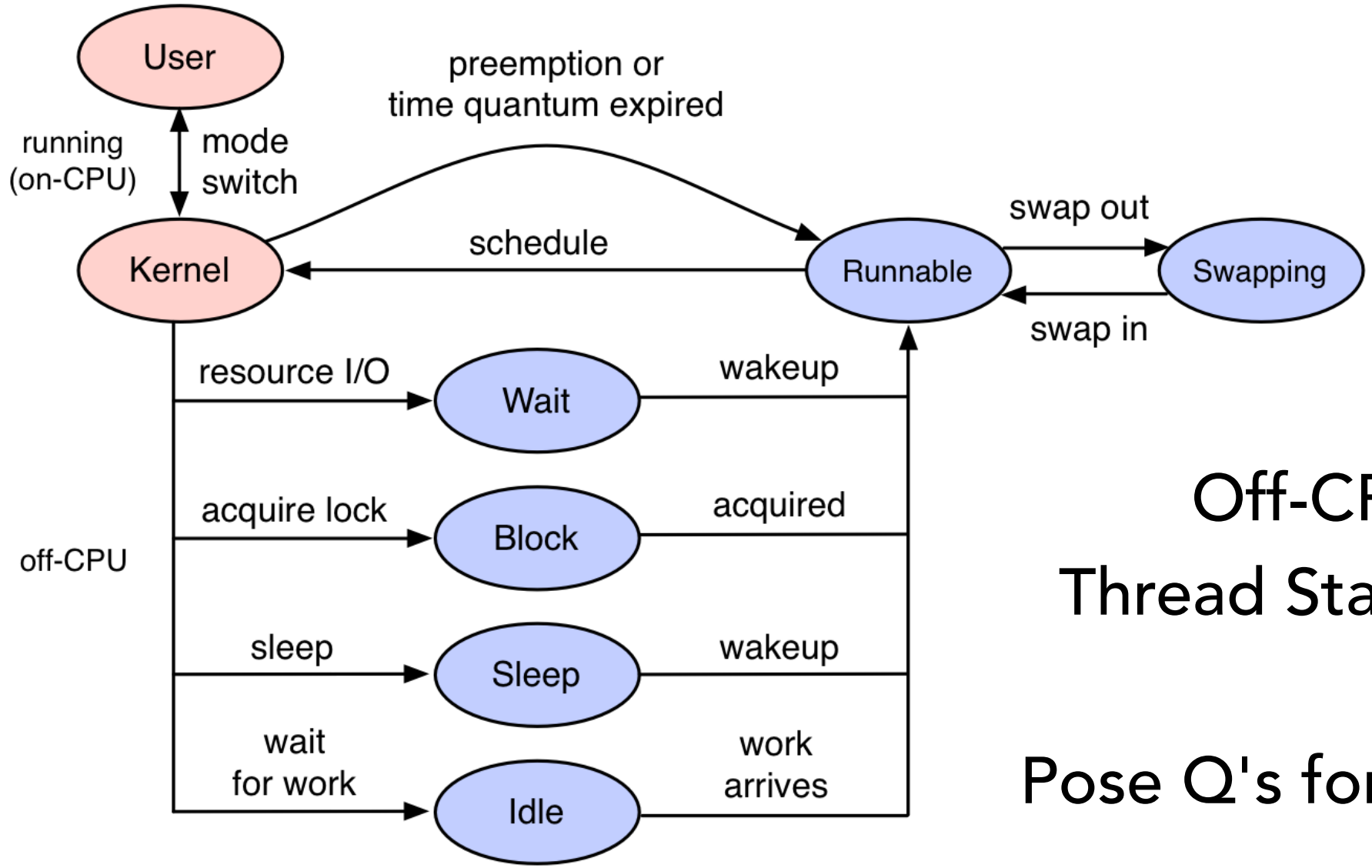

Dynamic Tracing
Tracepoints
PMCs
uprobes
kprobes
Operating System
Applications
System Libraries
System Call Interface
VFS
Sockets
File Systems
TCP/UDP
Volume Manager
IP
Block Device Interface
Ethernet
Device Drivers
Scheduler
Virtual Memory
CPU Interconnect
CPU 1
Memory Bus
DRAM
ext4:
jbd2:
block:
scsi:
sock:
net:
skb:
syscalls:
sched:
task:
signal:
timer:
workqueue:
kmem:
vmscan:
writeback:
irq:
cycles
instructions
branch-*
L1-*
LLC-*
mem-load
mem-store
Software Events
cpu-clock
cs
migrations
page-faults
minor-faults
major-faults

Enhanced BPF is in Linux


Version BPF support arrived
Dynamic Tracing
Tracepoints
Linux 4.7
PMCs
Linux 4.9
ext4:
Operating System
sock:
syscalls:
sched:
task:
signal:
timer:
workqueue:
cycles
instructions
branch-*
L1-*
LLC-*
uprobes
Linux 4.3
Applications
System Libraries
System Call Interface
CPU Interconnect
VFS
Sockets
Scheduler
CPU 1
kprobes
Linux 4.1
File Systems
TCP/UDP
Volume Manager
IP
Virtual Memory
kmem:
vmscan:
writeback:
Memory Bus
Block Device Interface
Ethernet
Device Drivers
DRAM
jbd2:
block:
scsi:
net:
skb:
irq:
mem-load
mem-store
BPF output
Linux 4.4
BPF stacks
Linux 4.6
Software Events
Linux 4.9
cpu-clock
cs
migrations
page-faults
minor-faults
major-faults


How do we
use these
superpowers?

Methodologies



Off-CPU Analysis

Thread State Analysis

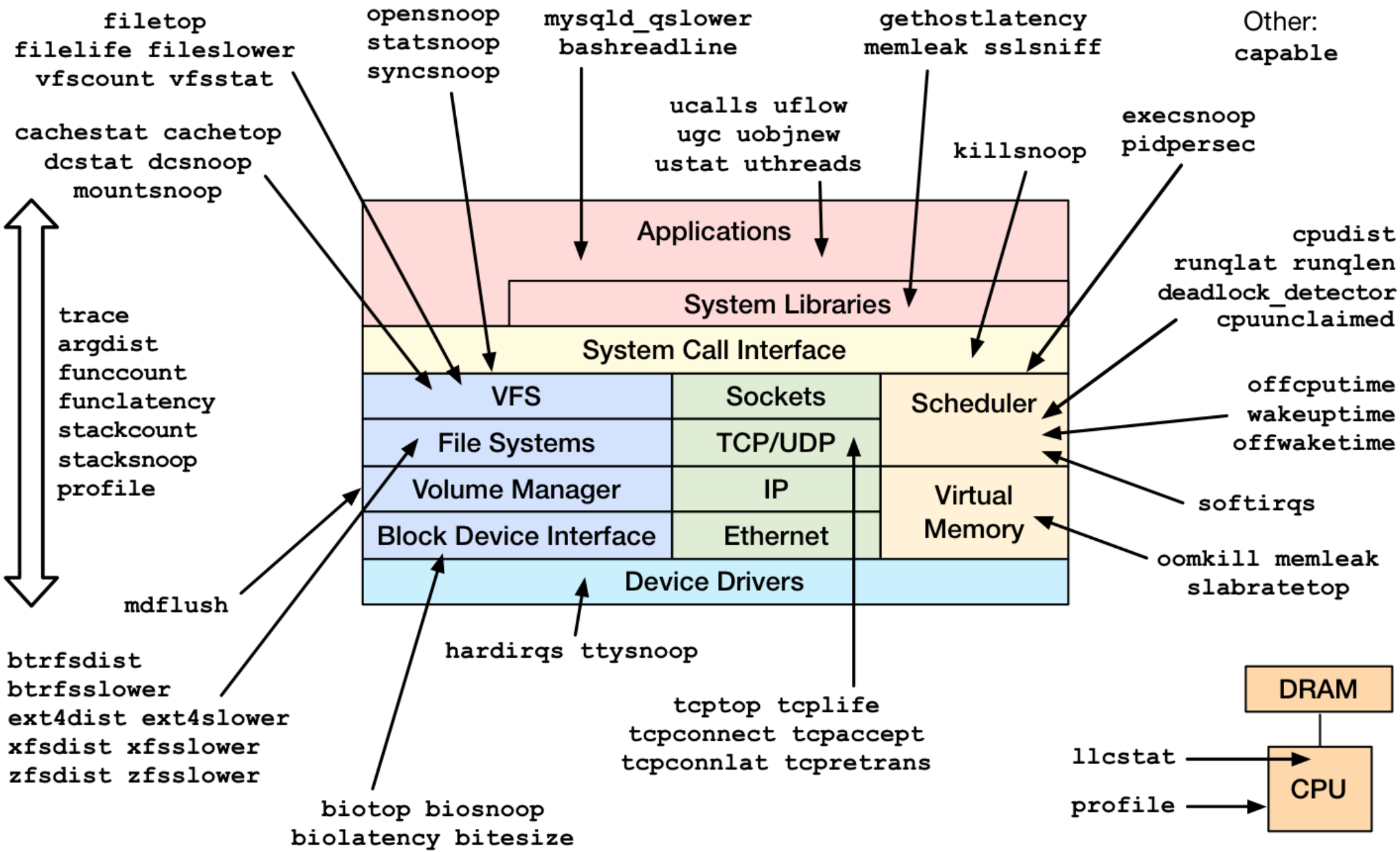
e.t.c.

Pose Q's for tools to A

Current Tools

bcc: BPF Compiler Collection

https://github.com/iovisor/bcc



https://github.com/iovisor/bcc#tools 2017

Single Purpose Tools

Multi-Tools

Single purpose vs Multi-tools

```
# opensnoop
PID    COMM               FD ERR PATH
10085  sshd                3   0 /lib/x86_64-linux-gnu/libresolv.so.2
10085  sshd                3   0 /lib/x86_64-linux-gnu/libgpg-error.so.0
10085  sshd                3   0 /dev/urandom
10085  sshd               -1   2 /lib/x86_64-linux-gnu/.libcrypto.so.1.0.0.hmac
10085  sshd               -1   2 /proc/sys/crypto/fips_enabled
```

```
# trace 'do_sys_open "%s", arg2' 'r::do_sys_open "ret:%d", retval'
PID    TID    COMM          FUNC             -
1651   1651   redis-server do_sys_open      /proc/1651/stat
1968   1968   redis-server do_sys_open      /proc/1968/stat
1651   1651   redis-server do_sys_open      ret:5
1968   1968   redis-server do_sys_open      ret:5
2218   2218   snmp-pass    do_sys_open      /proc/cpuinfo
2218   2218   snmp-pass    do_sys_open      ret:4
2218   2218   snmp-pass    do_sys_open      /proc/stat
2218   2218   snmp-pass    do_sys_open      ret:4
```

Single purpose tool usage

```
# biolatency -h
usage: biolatency [-h] [-T] [-Q] [-m] [-D] [interval] [count]

Summarize block device I/O latency as a histogram

positional arguments:
  interval              output interval, in seconds
  count                 number of outputs

optional arguments:
  -h, --help            show this help message and exit
  -T, --timestamp       include timestamp on output
  -Q, --queued          include OS queued time in I/O time
  -m, --milliseconds    millisecond histogram
  -D, --disks           print a histogram per disk device

examples:
    ./biolatency            # summarize block I/O latency as a histogram
[...]
```

CLI

Tool Design

Template 1: Per Event Output

```
# opensnoop
PID    COMM               FD ERR PATH
10085  sshd                3   0 /lib/x86_64-linux-gnu/libkeyutils.so.1
10085  sshd                3   0 /lib/x86_64-linux-gnu/libresolv.so.2
10085  sshd                3   0 /lib/x86_64-linux-gnu/libgpg-error.so.0
10085  sshd                3   0 /dev/urandom
10085  sshd               -1   2 /lib/x86_64-linux-gnu/.libcrypto.so.1.0.0.hmac
10085  sshd               -1   2 /proc/sys/crypto/fips_enabled
10085  sshd                3   0 /proc/filesystems
10085  sshd                3   0 /dev/null
10085  sshd                3   0 /proc/10085/fd
10085  sshd                3   0 /usr/lib/ssl/openssl.cnf
10085  sshd                3   0 /etc/gai.conf
10085  sshd                3   0 /etc/nsswitch.conf
10085  sshd                3   0 /etc/ld.so.cache
10085  sshd                3   0 /lib/x86_64-linux-gnu/libnss_compat.so.2
10085  sshd                3   0 /etc/ld.so.cache
10085  sshd                3   0 /lib/x86_64-linux-gnu/libnss_nis.so.2
[…]
```

Template 2: Filtered Event Output

```
# ext4slower 1
Tracing ext4 operations slower than 1 ms
TIME     COMM             PID    T BYTES   OFF_KB   LAT(ms) FILENAME
06:49:17 bash             3616   R 128     0           7.75 cksum
06:49:17 cksum            3616   R 39552   0           1.34 [
06:49:17 cksum            3616   R 96      0           5.36 2to3-2.7
06:49:17 cksum            3616   R 96      0          14.94 2to3-3.4
06:49:17 cksum            3616   R 10320   0           6.82 411toppm
06:49:17 cksum            3616   R 65536   0           4.01 a2p
06:49:17 cksum            3616   R 55400   0           8.77 ab
06:49:17 cksum            3616   R 36792   0          16.34 aclocal-1.14
06:49:17 cksum            3616   R 15008   0          19.31 acpi_listen
06:49:17 cksum            3616   R 6123    0          17.23 add-apt-repository
06:49:17 cksum            3616   R 6280    0          18.40 addpart
06:49:17 cksum            3616   R 27696   0           2.16 addr2line
06:49:17 cksum            3616   R 58080   0          10.11 ag
06:49:17 cksum            3616   R 906     0           6.30 ec2-meta-data
06:49:17 cksum            3616   R 6320    0          10.00 animate.im6
[…]
```

Template 3: Interval Summary

```
# dcstat
TIME         REFS/s   SLOW/s   MISS/s     HIT%
08:11:47:      2059      141       97    95.29
08:11:48:     79974      151      106    99.87
08:11:49:    192874      146      102    99.95
08:11:50:      2051      144      100    95.12
08:11:51:     73373    17239    17194    76.57
08:11:52:     54685    25431    25387    53.58
08:11:53:     18127     8182     8137    55.12
08:11:54:     22517    10345    10301    54.25
08:11:55:      7524     2881     2836    62.31
08:11:56:      2067      141       97    95.31
08:11:57:      2115      145      101    95.22
[…]
```

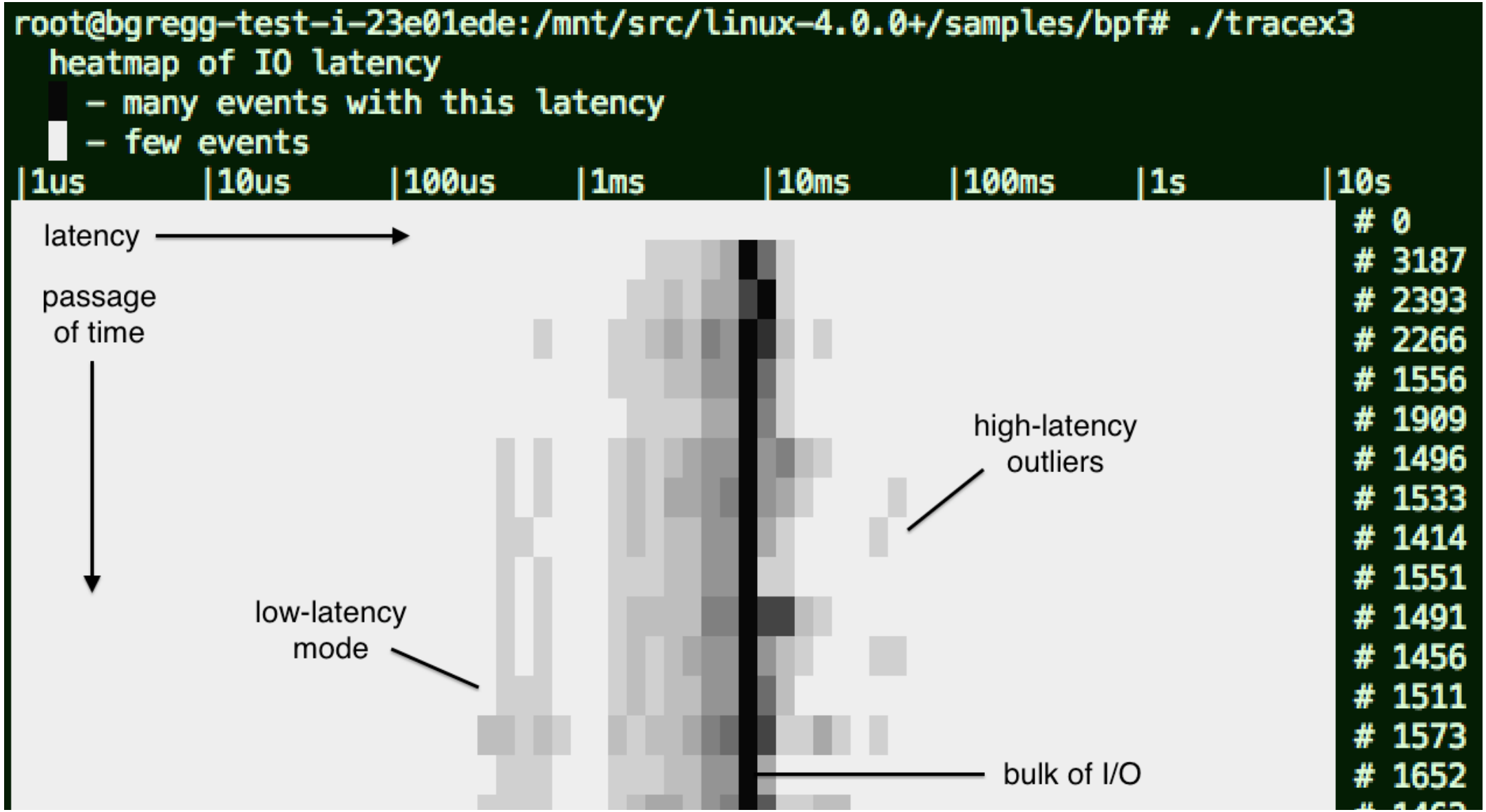
Template 4: Count Summary

```
# funccount 'vfs_*'
Tracing... Ctrl-C to end.
^C
ADDR             FUNC                          COUNT
ffffffff811efe81 vfs_create                        1
ffffffff811f24a1 vfs_rename                        1
ffffffff81215191 vfs_fsync_range                   2
ffffffff81231df1 vfs_lock_file                    30
ffffffff811e8dd1 vfs_fstatat                     152
ffffffff811e8d71 vfs_fstat                       154
ffffffff811e4381 vfs_write                       166
ffffffff811e8c71 vfs_getattr_nosec               262
ffffffff811e8d41 vfs_getattr                     262
ffffffff811e3221 vfs_open                        264
ffffffff811e4251 vfs_read                        470
Detaching...
```

Template 5: Histogram Summary

```
# biolatency
Tracing block device I/O... Hit Ctrl-C to end.
^C
     usecs               : count     distribution
       4 -> 7            : 0        |                                        |
       8 -> 15           : 0        |                                        |
      16 -> 31           : 0        |                                        |
      32 -> 63           : 0        |                                        |
      64 -> 127          : 1        |                                        |
     128 -> 255          : 12       |********                                |
     256 -> 511          : 15       |**********                              |
     512 -> 1023         : 43       |******************************          |
    1024 -> 2047         : 52       |************************************    |
    2048 -> 4095         : 47       |********************************        |
    4096 -> 8191         : 52       |************************************    |
    8192 -> 16383        : 36       |*************************               |
   16384 -> 32767        : 15       |**********                              |
   32768 -> 65535        : 2        |*                                       |
   65536 -> 131071       : 2        |*                                       |
```

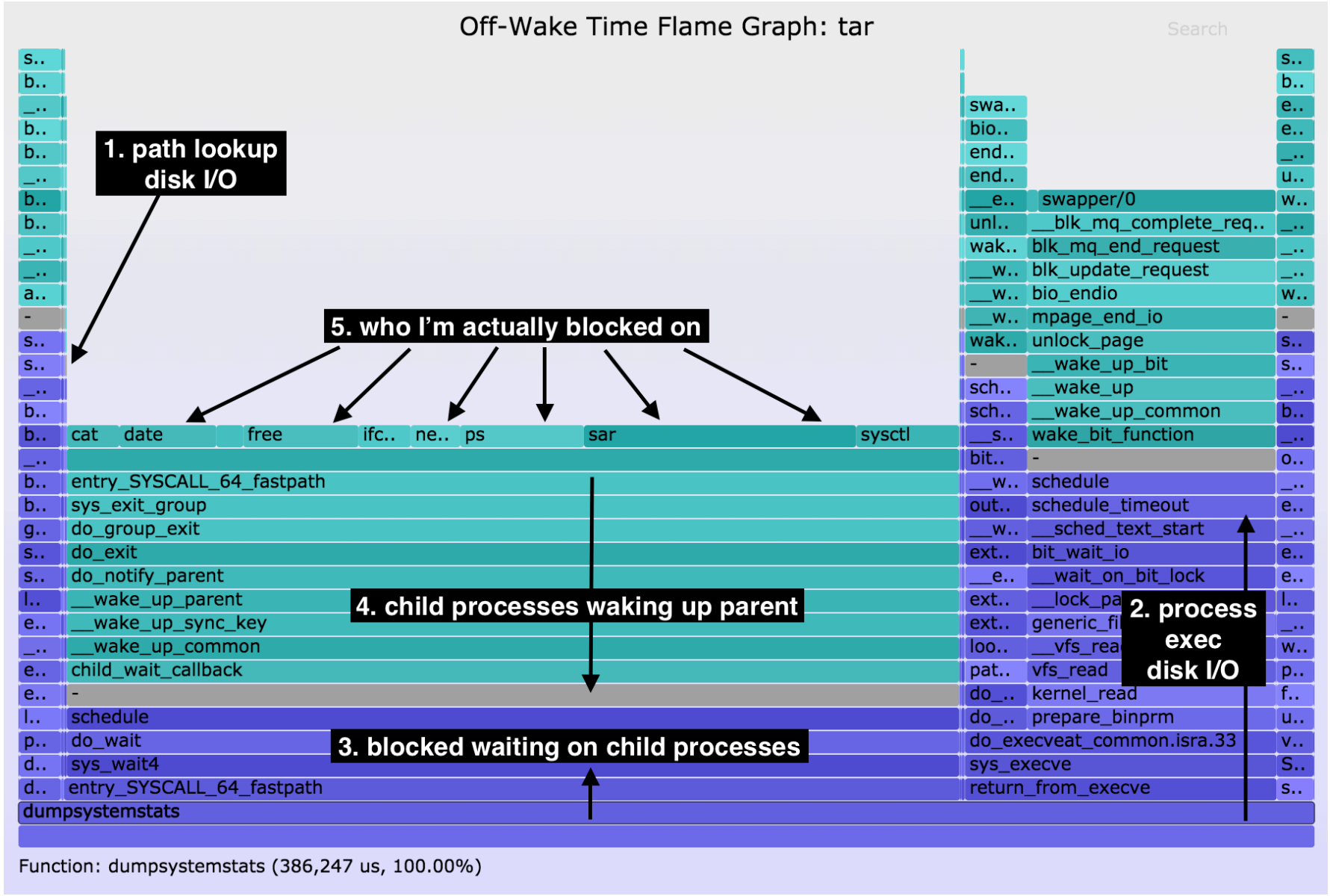
Template 6: Heatmap Summary

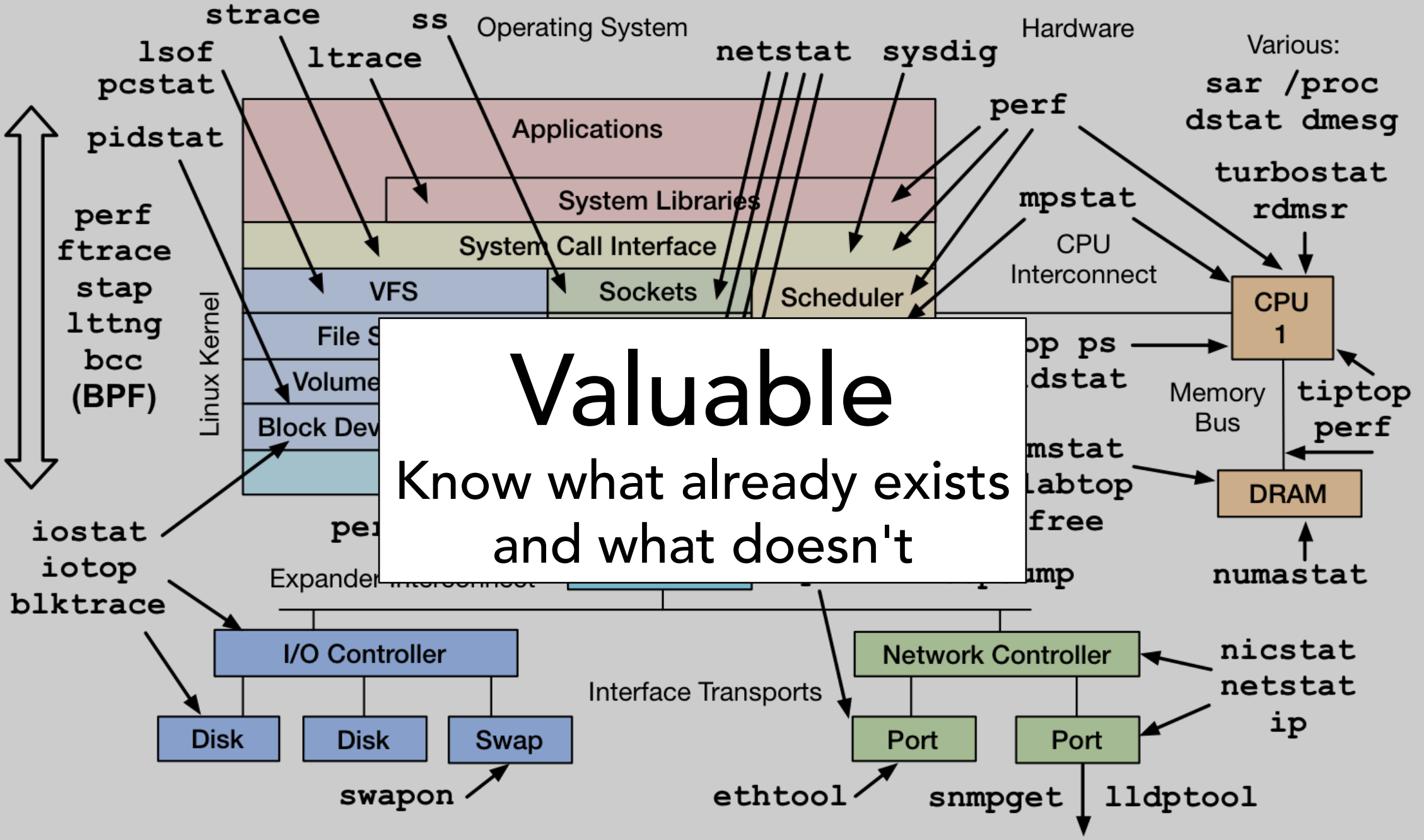


Template 7: Folded stack output for flame graphs

```
offcputime -f
offwaketime -f
wakeuptime -f
profile -f

| flamegraph.pl
> out.svg
```





Valuable

Know what already exists and what doesn't

Documented

code comments

man pages

example files

Concise, intuitive self-explanatory

```
# iolatency
Tracing block I/O. Output every 1 seconds. Ctrl-C to end.

  >=(ms) .. <(ms)   : I/O        |Distribution                           |
      0 -> 1        : 4381       |######################################|
      1 -> 2        : 9          |#                                     |
      2 -> 4        : 5          |#                                     |
      4 -> 8        : 0          |                                      |
      8 -> 16       : 1          |#                                     |
[…]
```

```
# ./biolatency -h
usage: biolatency [-h] [-T] [-Q] [-m] [-D] [interval] [count]

Summarize block device I/O latency as a his

positional arguments:
  interval              output interval, in s
  count                 number of outputs

optional arguments:
  -h, --help            show this help message and exit
  -T, --timestamp       include timestamp on output
  -Q, --queued          include OS queued time in I/O time
  -m, --milliseconds    millisecond histogram
  -D, --disks           print a histogram per disk device

examples:
    ./biolatency            # summarize block I/O latency as a histogram
    ./biolatency 1 10       # print 1 second summaries, 10 times
    ./biolatency -mT 1      # 1s summaries, milliseconds, and timestamps
    ./biolatency -Q         # include OS queued time in I/O time
    ./biolatency -D         # show each disk device separately
```

POSIX-style arguments

Option	Alternate	Expectation
-a	--all	all events
-c CMD	--cmd …	run this command
-d SECONDS	--duration …	duration of tool execution
-h	--help	help
-i FILE	--input …	input file
-i SECONDS	--interval …	summary interval
-n name	--name …	this process name only
-o FILE	--output …	output file
-p PID	--pid …	this process ID only
-P	--by-process	per-process ID breakdown
-P PORT	--port …	this TCP port only
-t or -T	--[no]timestamp	include or exclude timestamps
-v	--verbose	verbose output
-x	--extended, --errors	extended output, or only failures
[interval [count]]	-	summary interval, and # of outputs

Tested

If you can't write the workload,

you can't write the tool

Future Challenges

State of BPF, Feb 2017

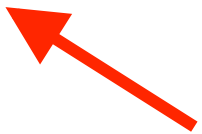
1. Dynamic tracing, kernel-level (BPF support for kprobes)
2. Dynamic tracing, user-level (BPF support for uprobes)
3. Static tracing, kernel-level (BPF support for tracepoints)
4. Timed sampling events (BPF with perf_event_open)
5. PMC events (BPF with perf_event_open)
6. Filtering (via BPF programs)
7. Debug output (bpf_trace_printk())
8. Per-event output (bpf_perf_event_output())
9. Basic variables (global & per-thread variables, via BPF maps)
10. Associative arrays (via BPF maps)
11. Frequency counting (via BPF maps)
12. Histograms (power-of-2, linear, and custom, via BPF maps)
13. Timestamps and time deltas (bpf_ktime_get_() and BPF)
14. Stack traces, kernel (BPF stackmap)
15. Stack traces, user (BPF stackmap)
16. Overwrite ring buffers
17. String factory (stringmap)
18. Optional: bounded loops, < and <=, …

done

not yet

State of bcc, Feb 2017

1. Static tracing, user-level (USDT probes via uprobes)
2. Static tracing, dynamic USDT (needs library support)
3. Debug output (Python with BPF.trace_pipe() and BPF.trace_fields())
4. Per-event output (BPF_PERF_OUTPUT macro and BPF.open_perf_buffer())
5. Interval output (BPF.get_table() and table.clear())
6. Histogram printing (table.print_log2_hist())
7. C struct navigation, kernel-level (maps to bpf_probe_read())
8. Symbol resolution, kernel-level (ksym(), ksymaddr())
9. Symbol resolution, user-level (usymaddr())
10. BPF tracepoint support (via TRACEPOINT_PROBE)
11. BPF stack trace support (incl. walk method for stack frames)
12. Examples (under /examples)
13. Many tools (/tools)
14. Tutorials (/docs/tutorial*.md)
15. Reference guide (/docs/reference_guide.md)
16. **Open issues**: (https://github.com/iovisor/bcc/issues)



Dynamic tracing stability

need those smoke tests
switch tools to static tracepoints

Invalid Tools

Overhead

Especially current uprobes

Ease of Coding

bcc/BPF

```
# load BPF program
b = BPF(text="""
#include <uapi/linux/ptrace.h>
#include <linux/blkdev.h>
BPF_HISTOGRAM(dist);
int kprobe__blk_account_io_completion(struct pt_regs *ctx,
    struct request *req)
{
    dist.increment(bpf_log2l(req->__data_len / 1024));
    return 0;
}
""")
```

```
# header
print("Tracing... Hit Ctrl-C to end.")

# trace until Ctrl-C
try:
    sleep(99999999)
except KeyboardInterrupt:
    print

# output
b["dist"].print_log2_hist("kbytes")
```

bcc examples/tracing/bitehist.py
entire program

ply/BPF

```
#!/usr/bin/env ply

kprobe:SyS_read
{
    $sizes.quantize(arg(2))
}
```

https://github.com/wkz/ply/blob/master/README.md

entire program

Visualizations

Visualizations and GUIs

Vector

Eg, Netflix self-service UI:

Ancient Linux

Linux 3.18

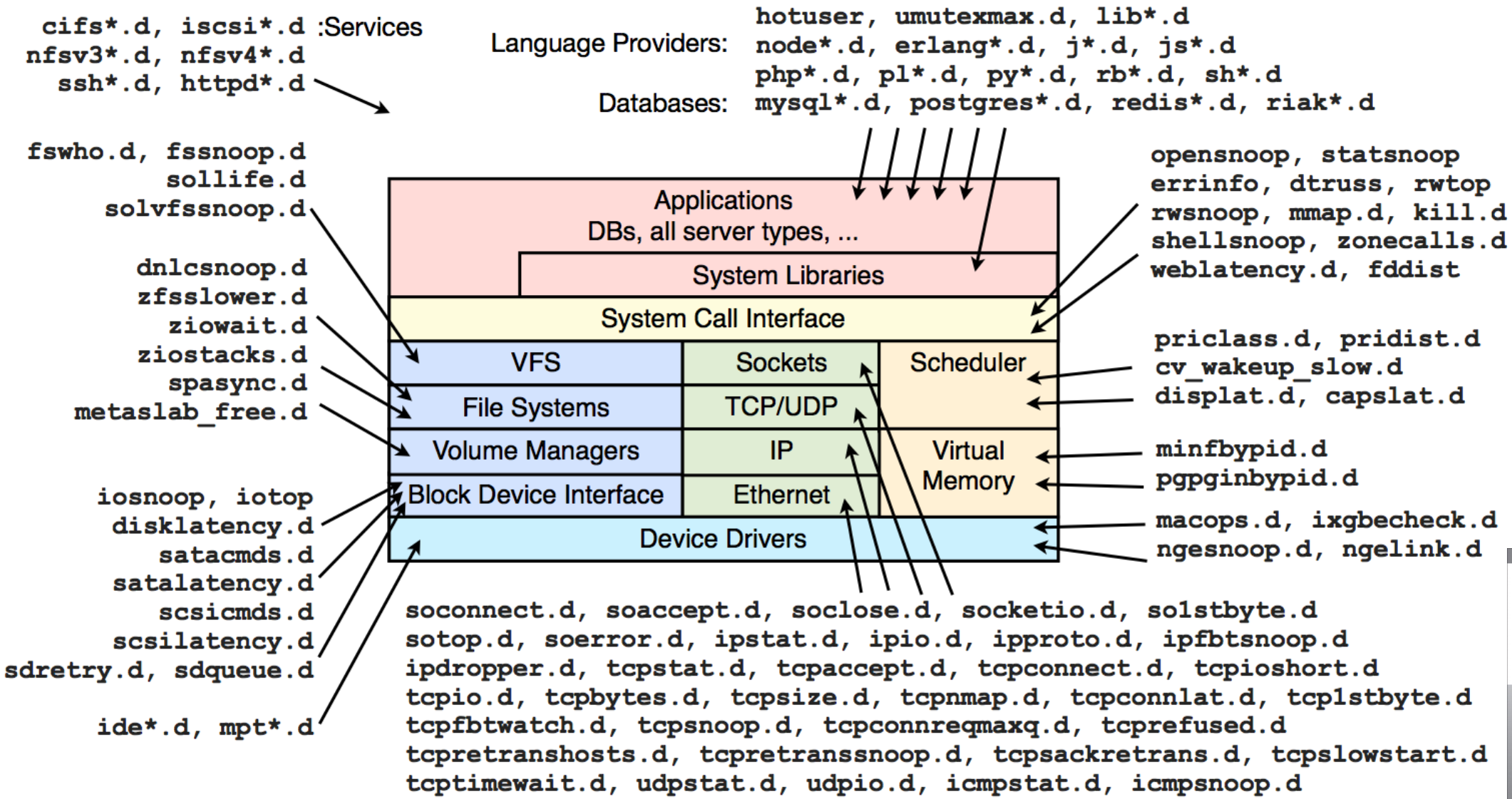
Linux 3.10

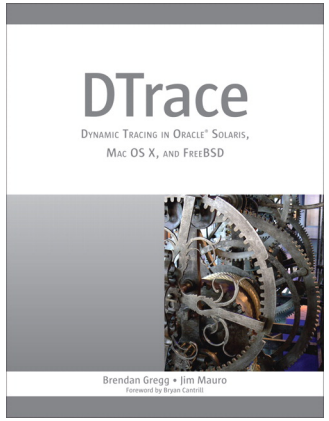
Linux 3.2

Linux 2.6.x

(Some) More Tools

Finish porting my old DTrace tools





Links & References

iovisor bcc:

- https://github.com/iovisor/bcc
- https://github.com/iovisor/bcc/tree/master/docs
- http://www.brendangregg.com/blog/ (search for "bcc")
- http://blogs.microsoft.co.il/sasha/2016/02/14/two-new-ebpf-tools-memleak-and-argdist/
- I'll change your view of Linux tracing: https://www.youtube.com/watch?v=GsMs3n8CB6g
- On designing tracing tools: https://www.youtube.com/watch?v=uibLwoVKjec

BPF:

- https://www.kernel.org/doc/Documentation/networking/filter.txt
- https://github.com/iovisor/bpf-docs
- https://suchakra.wordpress.com/tag/bpf/

Flame Graphs:

- http://www.brendangregg.com/flamegraphs.html
- http://www.brendangregg.com/blog/2016-01-20/ebpf-offcpu-flame-graph.html
- http://www.brendangregg.com/blog/2016-02-01/linux-wakeup-offwake-profiling.html

Linux Performance: http://www.brendangregg.com/linuxperf.html

Thanks

Discussion?

iovisor bcc: https://github.com/iovisor/bcc

http://www.brendangregg.com

http://slideshare.net/brendangregg

bgregg@netflix.com

@brendangregg

Thanks to Alexei Starovoitov (Facebook), Brenden Blanco (PLUMgrid/VMware), Sasha Goldshtein (Sela), Daniel Borkmann (Cisco), Wang Nan (Huawei), and other BPF and bcc contributors!