Oct, 2019

BPF Performance Tools Workshop

Brendan Gregg

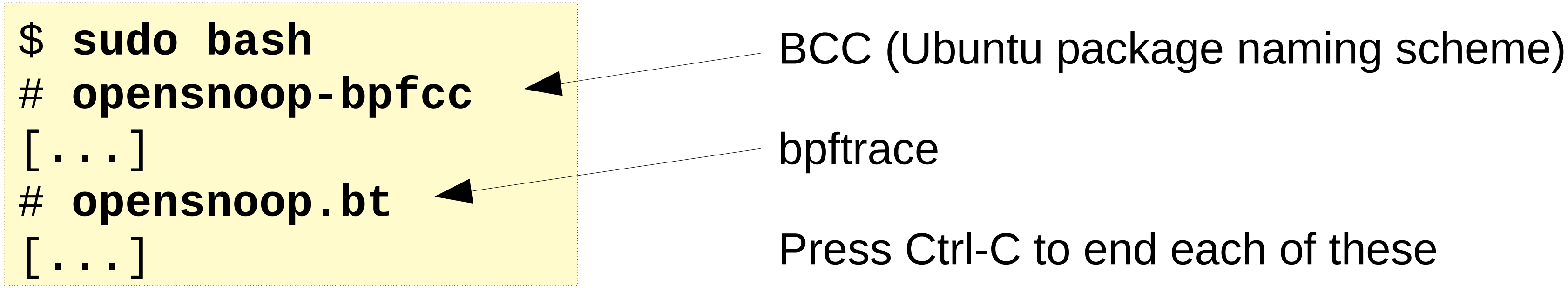
Senior Performance Engineer

NETFLIX

USENIX LISA 2019, Portland, Oct 28-30

Welcome to BPF Performance Tools

1. You can either:
   - A) SSH to a lab instance (see bit of paper)
   - B) setup your own system. Install BCC & bpftrace, the labs from https://github.com/brendangregg/bpf-perf-workshop, and extra tools from https://github.com/brendangregg/bpf-perf-tools-book
2. Check BCC and bpftrace tools work. On the lab instance:



```
$ sudo bash
# opensnoop-bpfcc
[...]
# opensnoop.bt
[...]
```

BCC (Ubuntu package naming scheme)

bpftrace

Press Ctrl-C to end each of these

3. Begin lab 1. You have 15 minutes.

Learning Objectives

1. Understand BPF, BCC, and bpftrace
2. Follow different analysis methodologies
3. Use BCC tools to analyze disk I/O issues
4. “ “ short-lived process issues
5. “ “ runq latency issues
6. Develop at least one new bpftrace tool

This was developed as a 90-minute workshop for USENIX LISA 2019

URLs

- https://github.com/brendangregg/bpf-perf-workshop
- https://github.com/iovisor/bcc Labs 1-3
  - https://github.com/iovisor/bcc/blob/master/docs/tutorial.md
- https://github.com/iovisor/bpftrace Labs 4-5
  - https://github.com/iovisor/bpftrace/blob/master/docs/tutorial_one_liners.md
  - https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md
- https://github.com/brendangregg/bpf-perf-tools-book

Lab 1 Discussion

BPF

BPF 1992: Berkeley Packet Filter

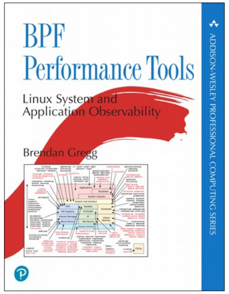
```
# tcpdump -d host 127.0.0.1 and port 80
(000) ldh      [12]
(001) jeq      #0x800           jt 2   jf 18
(002) ld       [26]
(003) jeq      #0x7f000001      jt 6   jf 4
(004) ld       [30]
(005) jeq      #0x7f000001      jt 6   jf 18
(006) ldb      [23]
(007) jeq      #0x84            jt 10  jf 8
(008) jeq      #0x6             jt 10  jf 9
(009) jeq      #0x11            jt 10  jf 18
(010) ldh      [20]
(011) jset     #0x1fff          jt 18  jf 12
(012) ldxb     4*([14]&0xf)
(013) ldh      [x + 14]
(014) jeq      #0x50            jt 17  jf 15
(015) ldh      [x + 16]
(016) jeq      #0x50            jt 17  jf 18
(017) ret      #262144
(018) ret      #0
```

A limited
virtual machine for
efficient packet filters

BPF 2019: aka extended BPF

bpftrace

XDP


BPF
Performance Tools
Linux System and Application Observability
Brendan Gregg
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

cilium

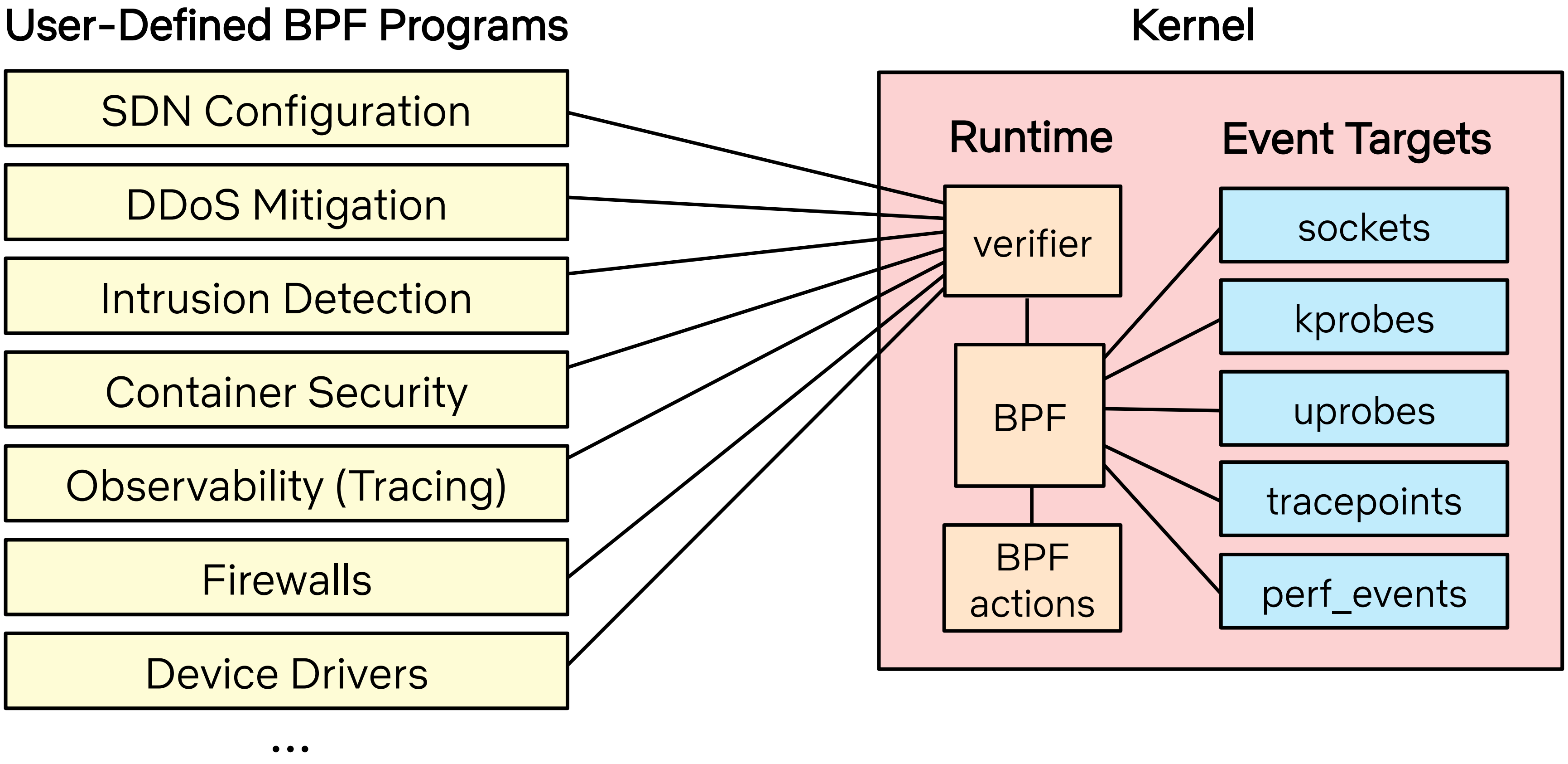
bpfconf

PLUMBERS CONFERENCE

BPF microconference

& Facebook Katran, Google KRSI, Netflix flowsrus, and many more

BPF 2019


User-Defined BPF Programs
SDN Configuration
DDoS Mitigation
Intrusion Detection
Container Security
Observability (Tracing)
Firewalls
Device Drivers
...
Kernel
Runtime
verifier
BPF
BPF actions
Event Targets
sockets
kprobes
uprobes
tracepoints
perf_events

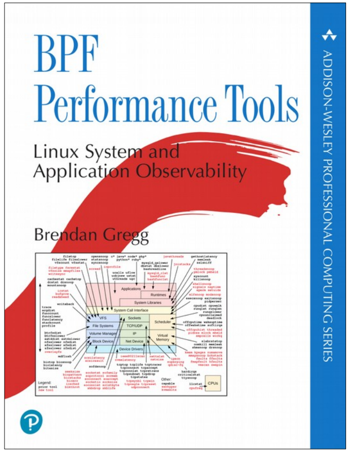
BPF Recommended Tracing Front Ends

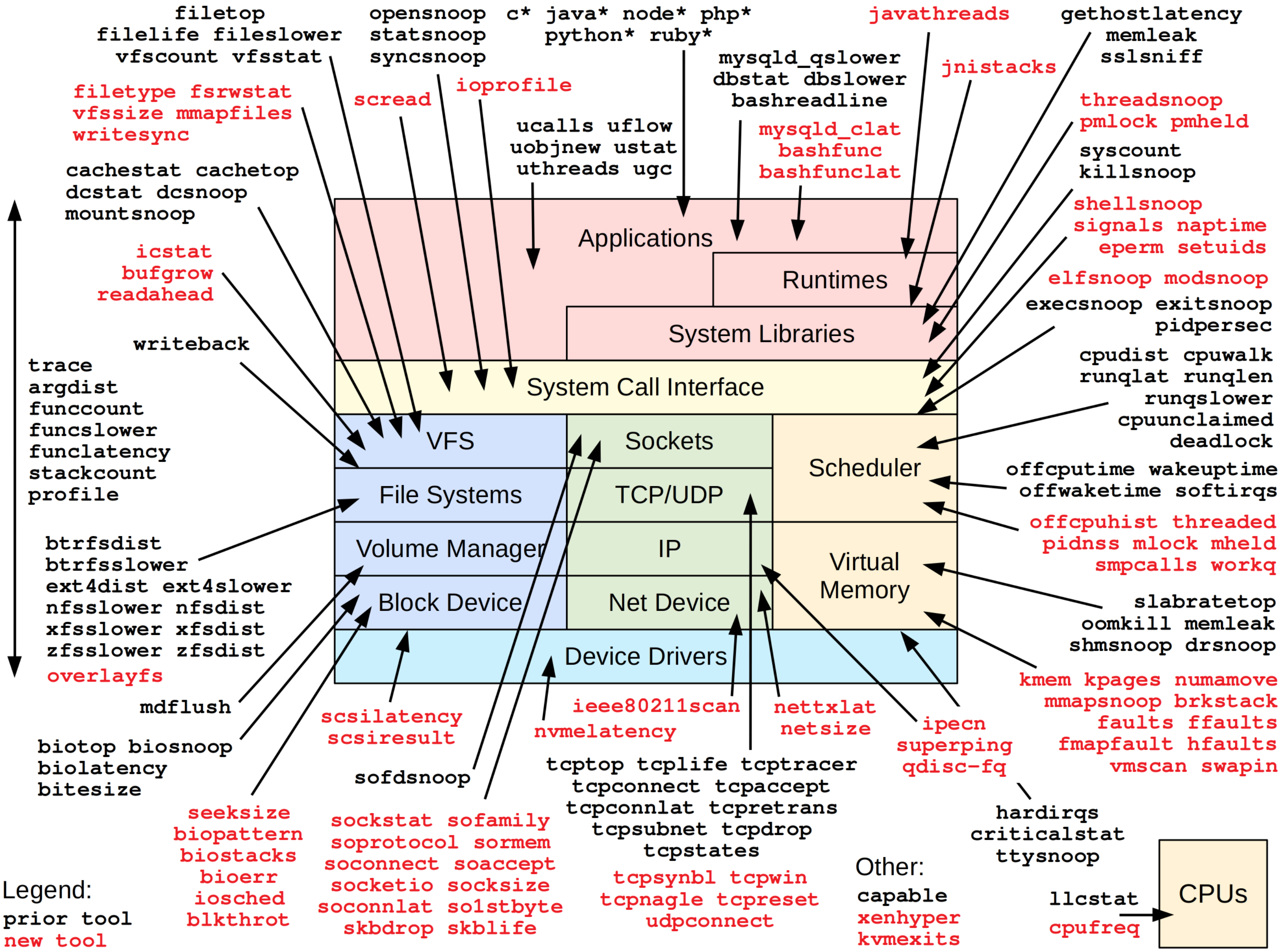
- BCC
  - Great for canned tools & daemons
  - https://github.com/iovisor/bcc
- bpftrace
  - Great for developing new short tools & one-liners
  - https://github.com/iovisor/bpftrace

The difference is like C vs shell
Polished products are better in C (BCC)
Custom one-offs are better in shell (bpftrace)

BPF Perf Tools (2019)

BCC & bpftrace repos contain those in black; the book repo has extras in red


BPF Performance Tools
Linux System and Application Observability
Brendan Gregg
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES



Applications
Runtimes
System Libraries
System Call Interface
VFS
Sockets
Scheduler
File Systems
TCP/UDP
Volume Manager
IP
Virtual Memory
Block Device
Net Device
Device Drivers
CPUs

filetop
filelife fileslower
vfscount vfsstat

filetype fsrwstat
vfssize mmapfiles
writesync

opensnoop
statsnoop
syncsnoop

scread

ioprofile

c* java* node* php*
python* ruby*

ucalls uflow
uobjnew ustat
uthreads ugc

mysqld_qslower
dbstat dbslower
bashreadline

mysqld_clat
bashfunc
bashfunclat

javathreads

jnistacks

gethostlatency
memleak
sslsniff

threadsnoop
pmlock pmheld

syscount
killsnoop

shellsnoop
signals naptime
eperm setuids

elfsnoop modsnoop

execsnoop exitsnoop
pidpersec

cachestat cachetop
dcstat dcsnoop
mountsnoop

icstat
bufgrow
readahead

writeback

trace
argdist
funccount
funcslower
funclatency
stackcount
profile

btrfsdist
btrfsslower
ext4dist ext4slower
nfsslower nfsdist
xfsslower xfsdist
zfsslower zfsdist
overlayfs

mdflush

biotop biosnoop
biolatency
bitesize

seeksize
biopattern
biostacks
bioerr
iosched
blkthrot

scsilatency
scsiresult

sofdsnoop

sockstat sofamily
soprotocol sormem
soconnect soaccept
socketio socksize
soconnlat so1stbyte
skbdrop skblife

ieee80211scan
nvmelatency

nettxlat
netsize

tcptop tcplife tcptracer
tcpconnect tcpaccept
tcpconnlat tcpretrans
tcpsubnet tcpdrop
tcpstates

tcpsynbl tcpwin
tcpnagle tcpreset
udpconnect

ipecn
superping
qdisc-fq

cpudist cpuwalk
runqlat runqlen
runqslower
cpuunclaimed
deadlock

offcputime wakeuptime
offwaketime softirqs

offcpuhist threaded
pidnss mlock mheld
smpcalls workq

slabratetop
oomkill memleak
shmsnoop drsnoop

kmem kpages numamove
mmapsnoop brkstack
faults ffaults
fmapfault hfaults
vmscan swapin

hardirqs
criticalstat
ttysnoop

Other:
capable
xenhyper
kvmexits

llcstat
cpufreq

Legend:
prior tool
new tool

BPF Tool Examples

CPUs: execsnoop

New process trace

```
# execsnoop.py -T
TIME(s) PCOMM            PID    PPID   RET ARGS
0.506   run              8745   1828     0 ./run
0.507   bash             8745   1828     0 /bin/bash
0.511   svstat           8747   8746     0 /command/svstat /service/nflx-httpd
0.511   perl             8748   8746     0 /usr/bin/perl -e $l=<>;$l=~/(\d+) sec/;p...
0.514   ps               8750   8749     0 /bin/ps --ppid 1 -o pid,cmd,args
0.514   grep             8751   8749     0 /bin/grep org.apache.catalina
0.514   sed              8752   8749     0 /bin/sed s/^ *//;
0.515   xargs            8754   8749     0 /usr/bin/xargs
0.515   cut              8753   8749     0 /usr/bin/cut -d  -f 1
0.523   echo             8755   8754     0 /bin/echo
0.524   mkdir            8756   8745     0 /bin/mkdir -v -p /data/tomcat
[...]
1.528   run              8785   1828     0 ./run
1.529   bash             8785   1828     0 /bin/bash
1.533   svstat           8787   8786     0 /command/svstat /service/nflx-httpd
1.533   perl             8788   8786     0 /usr/bin/perl -e $l=<>;$l=~/(\d+) sec/;p...
[...]
```

CPUs: runqlat

Scheduler latency (run queue latency)

```
# runqlat.py 10 1
Tracing run queue latency... Hit Ctrl-C to end.

     usecs               : count     distribution
         0 -> 1          : 1906     |***                                     |
         2 -> 3          : 22087    |****************************************|
         4 -> 7          : 21245    |**************************************  |
         8 -> 15         : 7333     |*************                           |
        16 -> 31         : 4902     |********                                |
        32 -> 63         : 6002     |**********                              |
        64 -> 127        : 7370     |*************                           |
       128 -> 255        : 13001    |***********************                 |
       256 -> 511        : 4823     |********                                |
       512 -> 1023       : 1519     |**                                      |
      1024 -> 2047       : 3682     |******                                  |
      2048 -> 4095       : 3170     |*****                                   |
      4096 -> 8191       : 5759     |**********                              |
      8192 -> 16383      : 14549    |**************************              |
     16384 -> 32767      : 5589     |**********                              |
```

CPUs: runqlen

Run queue length

```
# runqlen.py 10 1
Sampling run queue length... Hit Ctrl-C to end.

     runqlen       : count     distribution
        0          : 47284    |****************************************|
        1          : 211      |                                        |
        2          : 28       |                                        |
        3          : 6        |                                        |
        4          : 4        |                                        |
        5          : 1        |                                        |
        6          : 1        |                                        |
```

Memory: ffaults (book)

Page faults by filename

```
# ffaults.bt
Attaching 1 probe...
^C

[...]
@[dpkg]: 18
@[sudoers.so]: 19
@[ld.so.cache]: 27
@[libpthread-2.27.so]: 29
@[ld-2.27.so]: 32
@[locale-archive]: 34
@[system.journal]: 39
@[libstdc++.so.6.0.25]: 43
@[libapt-pkg.so.5.0.2]: 47
@[BrowserMetrics-5D8A6422-77F1.pma]: 86
@[libc-2.27.so]: 168
@[i915]: 409
@[pkgcache.bin]: 860
@[]: 25038
```

Disks: biolatency

Disk I/O latency histograms, per second

```
# biolatency.py -mT 1 5
Tracing block device I/O... Hit Ctrl-C to end.

06:20:16
     msecs               : count     distribution
       0 -> 1          : 36       |**************************************  |
       2 -> 3          : 1        |*                                       |
       4 -> 7          : 3        |***                                     |
       8 -> 15         : 17       |*****************                       |
      16 -> 31         : 33       |**********************************      |
      32 -> 63         : 7        |*******                                 |
      64 -> 127        : 6        |******                                  |

06:20:17
     msecs               : count     distribution
       0 -> 1          : 96       |************************************    |
       2 -> 3          : 25       |*********                               |
       4 -> 7          : 29       |***********                             |
[...]
```

File Systems: xfsslower

XFS I/O slower than a threshold (variants for ext4, btrfs, zfs)

```
# xfsslower.py 50
Tracing XFS operations slower than 50 ms
TIME     COMM           PID    T BYTES   OFF_KB   LAT(ms) FILENAME
21:20:46 java           112789 R 8012    13925      60.16 file.out
21:20:47 java           112789 R 3571    4268      136.60 file.out
21:20:49 java           112789 R 5152    1780       63.88 file.out
21:20:52 java           112789 R 5214    12434     108.47 file.out
21:20:52 java           112789 R 7465    19379      58.09 file.out
21:20:54 java           112789 R 5326    12311      89.14 file.out
21:20:55 java           112789 R 4336    3051       67.89 file.out
[...]
22:02:39 java           112789 R 65536   1486748   182.10 shuffle_6_646_0.data
22:02:39 java           112789 R 65536   872492     30.10 shuffle_6_646_0.data
22:02:39 java           112789 R 65536   1113896   309.52 shuffle_6_646_0.data
22:02:39 java           112789 R 65536   1481020   400.31 shuffle_6_646_0.data
22:02:39 java           112789 R 65536   1415232   324.92 shuffle_6_646_0.data
22:02:39 java           112789 R 65536   1147912   119.37 shuffle_6_646_0.data
[...]
```

File Systems: xfsdist

XFS I/O latency histograms by operation

```
# xfsdist.py 60
Tracing XFS operation latency... Hit Ctrl-C to end.

22:41:24:

operation = 'read'
     usecs               : count     distribution
         0 -> 1          : 382130   |****************************************|
         2 -> 3          : 85717    |*********                               |
         4 -> 7          : 23639    |**                                      |
         8 -> 15         : 5668     |                                        |
        16 -> 31         : 3594     |                                        |
        32 -> 63         : 21387    |**                                      |
[...]
operation = 'write'
     usecs               : count     distribution
         0 -> 1          : 12925    |*****                                   |
         2 -> 3          : 83375    |*************************************   |
[...]
```

Networking: tcplife

TCP session lifespans with connection details

```
# tcplife.py
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
[...]
```

Networking: tcpsynbl (book)

TCP SYN backlogs as histograms

```
# tcpsynbl.bt
Attaching 4 probes...
Tracing SYN backlog size. Ctrl-C to end.
^C
@backlog[backlog limit]: histogram of backlog size

@backlog[128]:
[0]                    2 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|

@backlog[500]:
[0]                 2783 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[1]                    9 |                                                    |
[2, 4)                 4 |                                                    |
[4, 8)                 1 |                                                    |
```

Languages: funccount

Count native function calls (C, C++, Go, etc)

```
# funccount.py 'tcp_s*'
Tracing 50 functions for "tcp_s*"... Hit Ctrl-C to end.
^C
FUNC                                    COUNT
[...]
tcp_setsockopt                           1839
tcp_shutdown                             2690
tcp_sndbuf_expand                        2862
tcp_send_delayed_ack                     9457
tcp_set_state                           10425
tcp_sync_mss                            12529
tcp_sendmsg_locked                      41012
tcp_sendmsg                             41236
tcp_send_mss                            42686
tcp_small_queue_check.isra.29           45724
tcp_schedule_loss_probe                 64067
tcp_send_ack                            66945
tcp_stream_memory_free                 178616
Detaching...
```

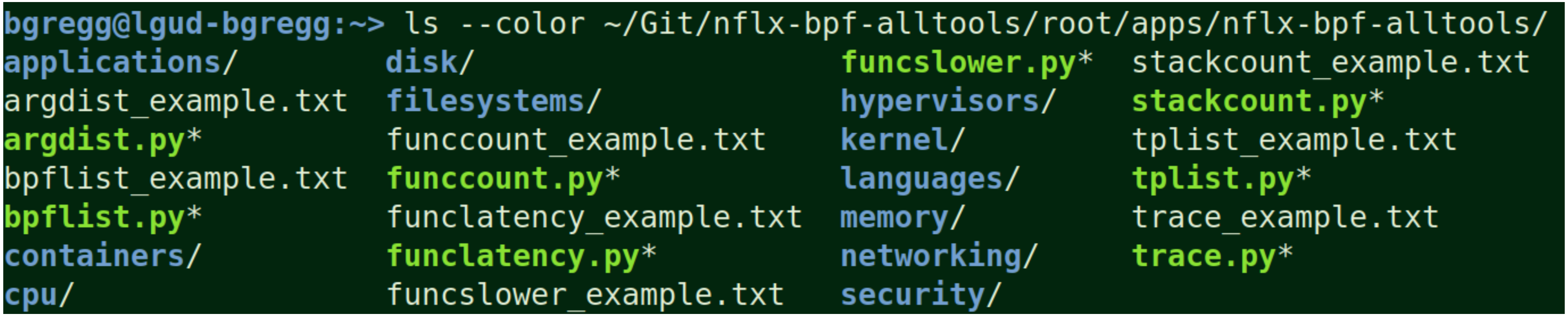
Applications: mysqld_qslower

MySQL queries slower than a threshold

```
# mysqld_qslower.py $(pgrep mysqld)
Tracing MySQL server queries for PID 9908 slower than 1 ms...
TIME(s)    PID        MS QUERY
0.000000   9962  169.032 SELECT * FROM words WHERE word REGEXP '^bre.*n$'
1.962227   9962  205.787 SELECT * FROM words WHERE word REGEXP '^bpf.tools$'
9.043242   9962   95.276 SELECT COUNT(*) FROM words
23.723025  9962  186.680 SELECT count(*) AS count FROM words WHERE word REGEXP 
'^bre.*n$'
30.343233  9962  181.494 SELECT * FROM words WHERE word REGEXP '^bre.*n$' ORDER BY 
word
[...]
```

Coping with so many BPF tools

- On all Netflix servers, **/apps/nflx-bpf-alltools** has tools from:
  - BCC, bpftrace, my book, Netflix internal
- Latest tools are fetched & installed in a hierarchy: cpu, disk, …



```
bgregg@lgud-bgregg:~> ls --color ~/Git/nflx-bpf-alltools/root/apps/nflx-bpf-alltools/
applications/        disk/                    funcslower.py*   stackcount_example.txt
argdist_example.txt  filesystems/             hypervisors/     stackcount.py*
argdist.py*          funccount_example.txt    kernel/          tplist_example.txt
bpflist_example.txt  funccount.py*            languages/       tplist.py*
bpflist.py*          funclatency_example.txt  memory/          trace_example.txt
containers/          funclatency.py*          networking/      trace.py*
cpu/                 funcslower_example.txt   security/
```

  - An employee can look in “disk” for all disk tools.
- We are also building **GUIs** to front these tools

We might open source the nflx-bpf-alltools framework at some point

Performance Analysis with BPF

Start With Basics

- From my BCC tutorial: https://github.com/iovisor/bcc/blob/master/docs/tutorial.md

1. uptime
2. dmesg | tail
3. vmstat 1
4. mpstat -P ALL 1
5. pidstat 1
6. iostat -xz 1
7. free -m
8. sar -n DEV 1
9. sar -n TCP,ETCP 1
10. top

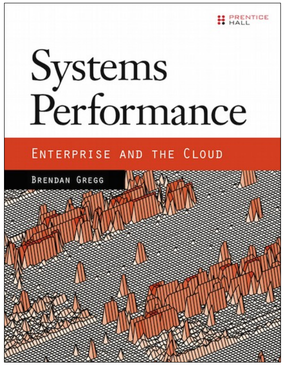
BCC Checklist

- ...continuing the BCC tutorial:

1. execsnoop
2. opensnoop
3. ext4slower
   (or btrfs*, xfs*, zfs*)
4. biolatency
5. biosnoop
6. cachestat
7. tcpconnect
8. tcpaccept
9. tcpretrans
10. runqlat
11. profile

The **most important** skill to learn

- It’s not the tools or the languages (bpftrace, BCC)
  - Analogy: good golf clubs don’t make you a great golfer.
- It’s the **know how**. It’s years of experience and wisdom gained from practicing performance engineering.
- I’ve taught tracing to >1000 students as a professional instructor. What I found most effective was develop and teach performance analysis **methodologies**.
  - I documented many in my last book. Look out for 2nd ed.

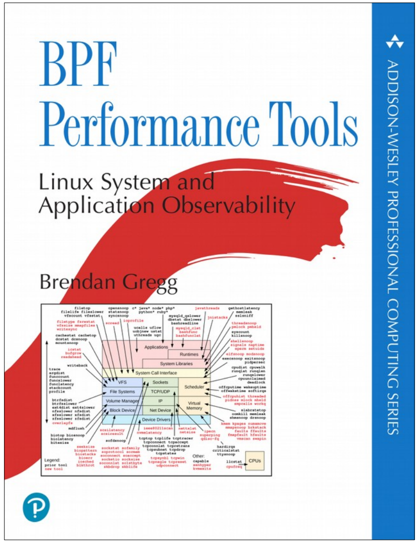


Methodologies

- Checklists
- Thread state analysis
- Workload characterization
- Reverse diagnosis
- Drill-down analysis
- Process of elimination
- 5 Whys
- (etc)

Checklists

BPF Perf Tools: my diagram can be a checklist

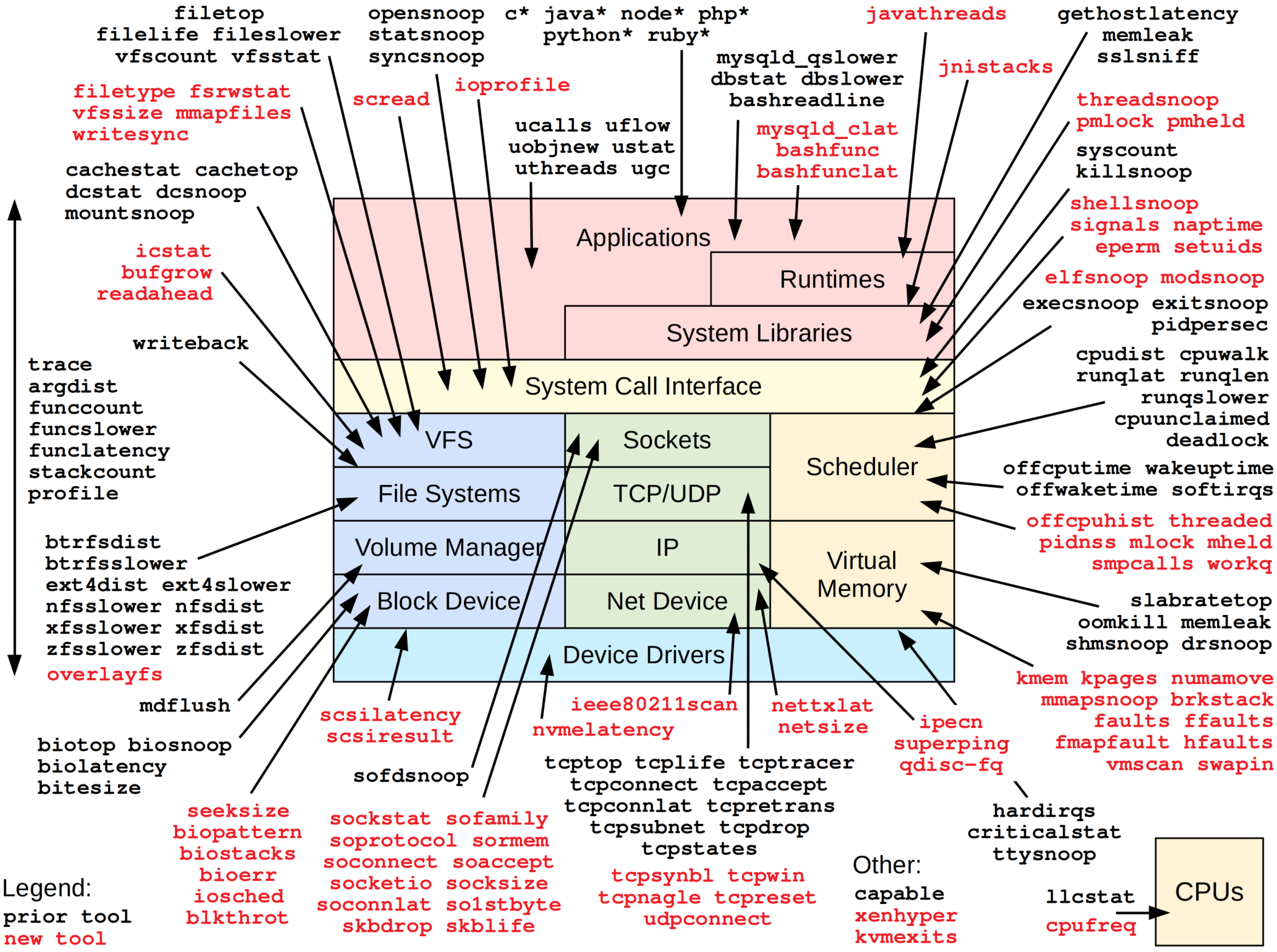


BPF Performance Tools

Linux System and Application Observability

Brendan Gregg

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES



Applications
Runtimes
System Libraries
System Call Interface
VFS
Sockets
Scheduler
File Systems
TCP/UDP
Volume Manager
IP
Virtual Memory
Block Device
Net Device
Device Drivers
CPUs

filetop
filelife fileslower
vfscount vfsstat

filetype fsrwstat
vfssize mmapfiles
writesync

opensnoop
statsnoop
syncsnoop

scread

ioprofile

c* java* node* php*
python* ruby*

ucalls uflow
uobjnew ustat
uthreads ugc

mysqld_qslower
dbstat dbslower
bashreadline

mysqld_clat
bashfunc
bashfunclat

javathreads

jnistacks

gethostlatency
memleak
sslsniff

threadsnoop
pmlock pmheld

syscount
killsnoop

shellsnoop
signals naptime
eperm setuids

elfsnoop modsnoop

execsnoop exitsnoop
pidpersec

cachestat cachetop
dcstat dcsnoop
mountsnoop

icstat
bufgrow
readahead

writeback

trace
argdist
funccount
funcslower
funclatency
stackcount
profile

btrfsdist
btrfsslower
ext4dist ext4slower
nfsslower nfsdist
xfsslower xfsdist
zfsslower zfsdist
overlayfs

mdflush

biotop biosnoop
biolatency
bitesize

seeksize
biopattern
biostacks
bioerr
iosched
blkthrot

scsilatency
scsiresult

sofdsnoop

sockstat sofamily
soprotocol sormem
soconnect soaccept
socketio socksize
soconnlat so1stbyte
skbdrop skblife

nvmelatency

ieee80211scan

tcptop tcplife tcptracer
tcpconnect tcpaccept
tcpconnlat tcpretrans
tcpsubnet tcpdrop
tcpstates

tcpsynbl tcpwin
tcpnagle tcpreset
udpconnect

nettxlat
netsize

ipecn
superping
qdisc-fq

cpudist cpuwalk
runqlat runqlen
runqslower
cpuunclaimed
deadlock

offcputime wakeuptime
offwaketime softirqs

offcpuhist threaded
pidnss mlock mheld
smpcalls workq

slabratetop
oomkill memleak
shmsnoop drsnoop

kmem kpages numamove
mmapsnoop brkstack
faults ffaults
fmapfault hfaults
vmscan swapin

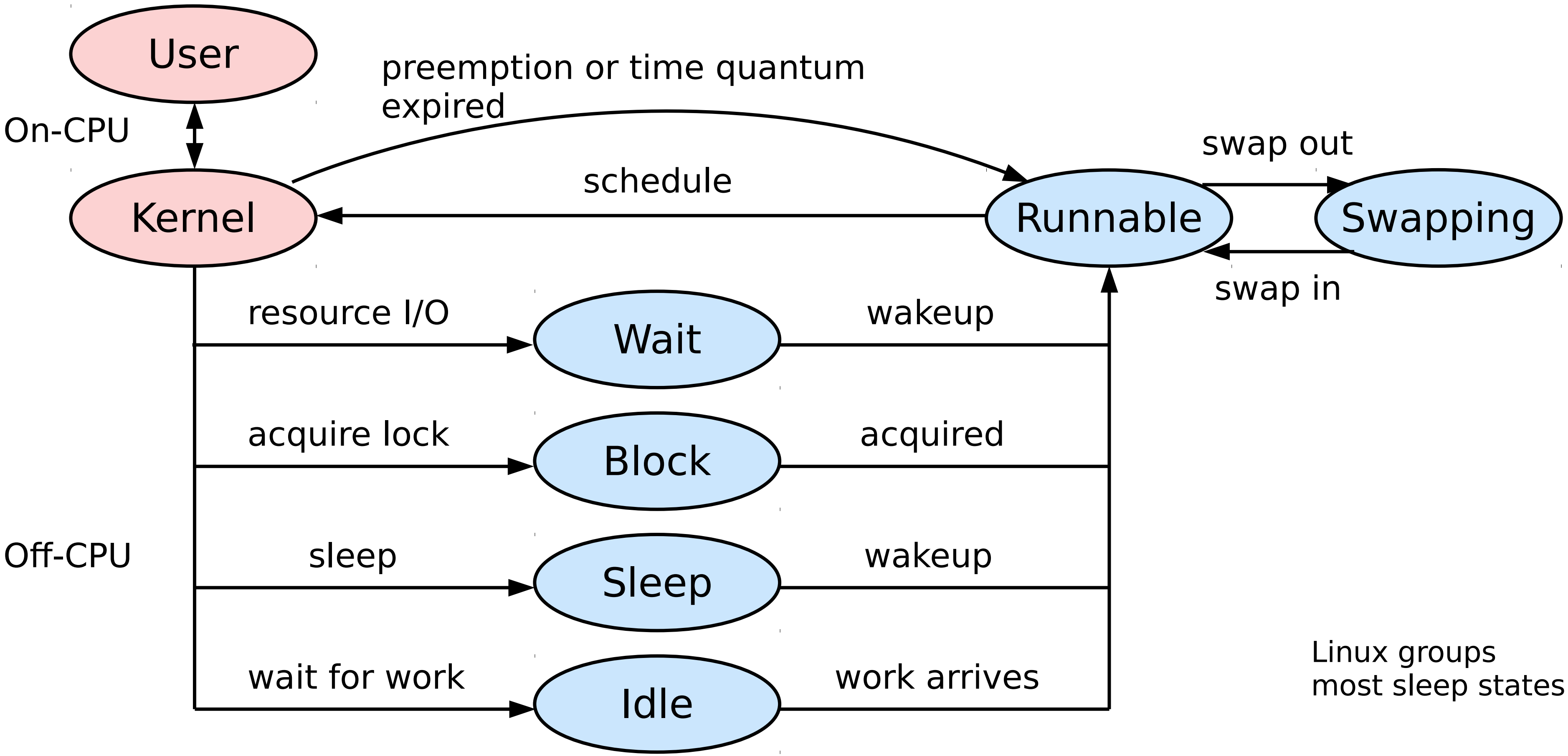
hardirqs
criticalstat
ttysnoop

Other:
capable
xenhyper
kvmexits

llcstat
cpufreq

Legend:
prior tool
new tool

Thread State Analysis



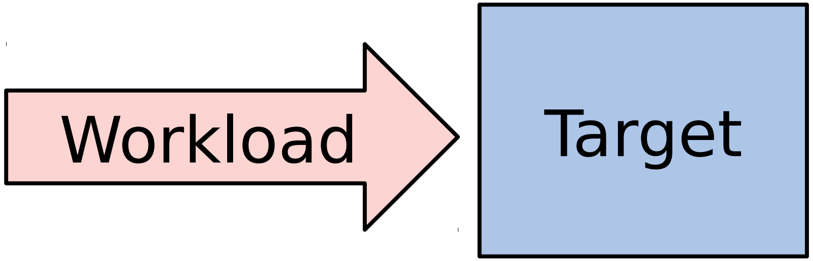
Linux groups
most sleep states

Workload Characterization

Analyze workload characteristics, not resulting performance

For example, CPUs:

1. **Who**: which PIDs, programs, users
2. **Why**: code paths, context
3. **What**: CPU instructions, cycles
4. **How**: changing over time



Reverse Diagnosis

- Performance problems can have multiple causes and solutions. List them then find the metrics.
- Example: disk I/O bounded workload:
  - A) buy faster disks
  - B) change filesystem tuning to reduce I/O
  - C) change application config to improve I/O (e.g., more I/O threads)
  - D) change application to do I/O later (e.g., sync→async)
  - E) change application to cache I/O
  - F) change application to eliminate I/O (e.g., logic change)
- What metrics would identify each of these solutions?

Other Methodologies

- Drill down analysis
- Process of elimination
- 5 Whys

Lab 2

- bpf-perf-workshop/lab2.md

Lab 2 Discussion

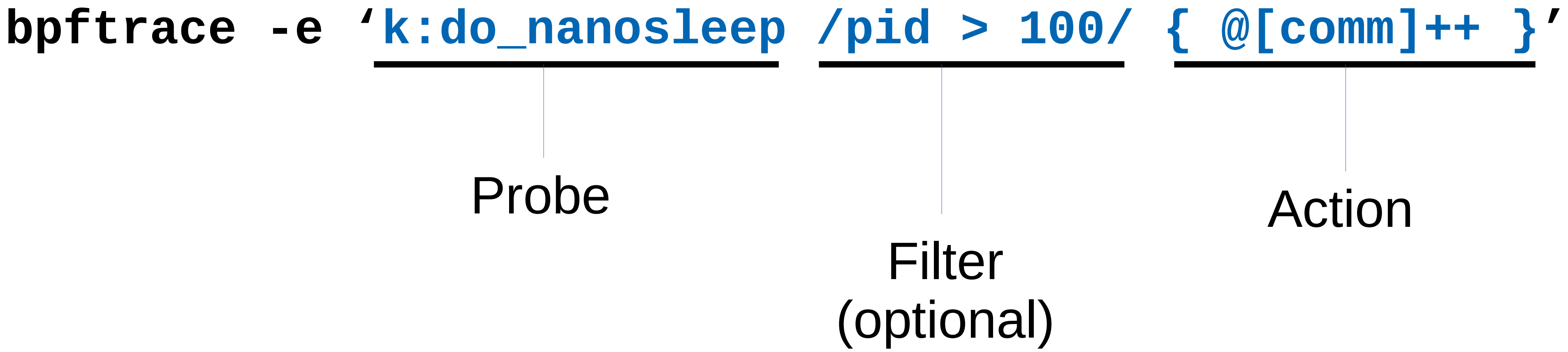
Lab 3

- bpf-perf-workshop/lab3.md

Lab 3 Discussion

bpftrace

bpftrace Syntax



```
bpftrace -e ‘k:do_nanosleep /pid > 100/ { @[comm]++ }’
```

Probe

Filter
(optional)

Action

Probe Type Shortcuts

`tracepoint`	`t`	Kernel static tracepoints
`usdt`	`U`	User-level statically defined tracing
`kprobe`	`k`	Kernel function tracing
`kretprobe`	`kr`	Kernel function returns
`uprobe`	`u`	User-level function tracing
`uretprobe`	`ur`	User-level function returns
`profile`	`p`	Timed sampling across all CPUs
`interval`	`i`	Interval output
`software`	`s`	Kernel software events
`hardware`	`h`	Processor hardware events

Filters

- **`/pid == 181/`**
- **`/comm != “sshd”/`**
- **`/@ts[tid]/`**

Actions

- Per-event output
  - **`printf()`**
  - **`system()`**
  - **`join()`**
  - **`time()`**
- Map Summaries
  - **`@ = count()`** `or` **`@++`**
  - **`@ = hist()`**
  - …

The following is in the https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md

Functions

`hist(n)`	Log2 histogram
`lhist(n, min, max, step)`	Linear hist.
`count()`	Count events
`sum(n)`	Sum value
`min(n)`	Minimum value
`max(n)`	Maximum value
`avg(n)`	Average value
`stats(n)`	Statistics
`str(s)`	String
`ksym(p)`	Resolve kernel addr
`usym(p)`	Resolve user addr
`kaddr(n)`	Resolve kernel symbol
`uaddr(n)`	Resolve user symbol

`printf(fmt, ...)`	Print formatted
`print(@x[, top[, div]])`	Print map
`delete(@x)`	Delete map element
`clear(@x)`	Delete all keys/values
`reg(n)`	Register lookup
`join(a)`	Join string array
`time(fmt)`	Print formatted time
`system(fmt)`	Run shell command
`cat(file)`	Print file contents
`exit()`	Quit bpftrace

Variable Types

- Basic Variables
  - **`@global`**
  - **`@thread_local[tid]`**
  - **`$scratch`**
- Associative Arrays
  - **`@array[key] = value`**
- Buitins
  - **`pid`**
  - **`...`**

Builtin Variables

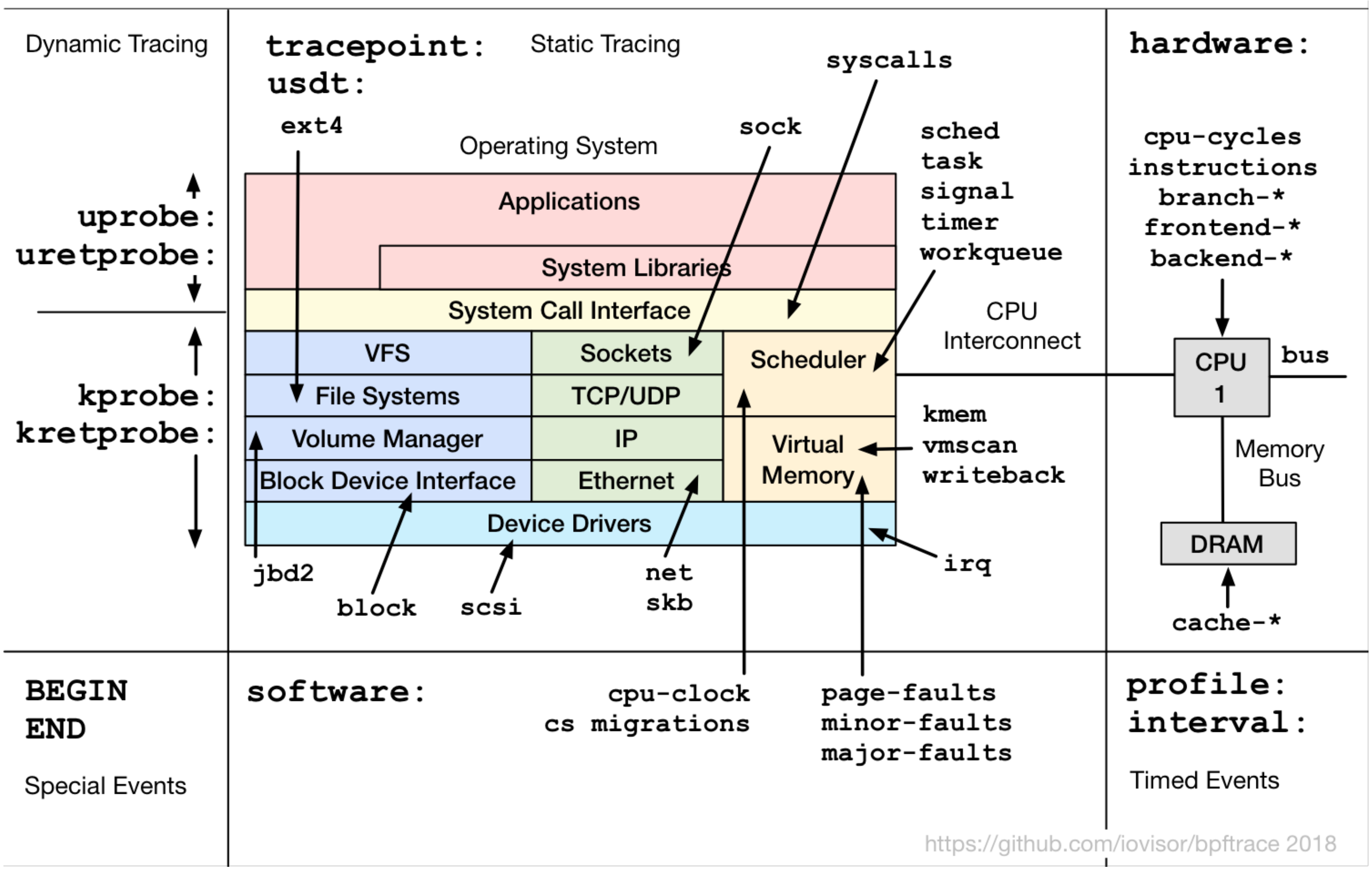
pid	Process ID (kernel tgid)
tid	Thread ID (kernel pid)
cgroup	Current Cgroup ID
uid	User ID
gid	Group ID
nsecs	Nanosecond timestamp
cpu	Processor ID
comm	Process name
kstack	Kernel stack trace
ustack	User stack trace

arg0, **arg1**, …	Function args
retval	Return value
args	Tracepoint args
func	Function name
probe	Full probe name
curtask	Curr task_struct (u64)
rand	Random number (u32)

bpftrace: BPF observability front-end

```
# Files opened by process
bpftrace -e 't:syscalls:sys_enter_open { printf("%s %s\n", comm,
    str(args->filename)) }'

# Read size distribution by process
bpftrace -e 't:syscalls:sys_exit_read { @[comm] = hist(args->ret) }'

# Count VFS calls
bpftrace -e 'kprobe:vfs_* { @[func]++ }'

# Show vfs_read latency as a histogram
bpftrace -e 'k:vfs_read { @[tid] = nsecs }
    kr:vfs_read /@[tid]/ { @ns = hist(nsecs - @[tid]); delete(@tid) }’

# Trace user-level function
bpftrace -e 'uretprobe:bash:readline { printf(“%s\n”, str(retval)) }’
...
```

Probes


Dynamic Tracing
uprobe:
uretprobe:
kprobe:
kretprobe:
tracepoint:
usdt:
Static Tracing
syscalls
ext4
sock
Operating System
sched
task
signal
timer
workqueue
Applications
System Libraries
System Call Interface
VFS
Sockets
Scheduler
File Systems
TCP/UDP
Volume Manager
IP
Virtual Memory
Block Device Interface
Ethernet
Device Drivers
CPU Interconnect
kmem
vmscan
writeback
irq
jbd2
block
scsi
net
skb
hardware:
cpu-cycles
instructions
branch-*
frontend-*
backend-*
CPU 1
bus
Memory Bus
DRAM
cache-*
BEGIN
END
Special Events
software:
cpu-clock
cs migrations
page-faults
minor-faults
major-faults
profile:
interval:
Timed Events
https://github.com/iovisor/bpftrace 2018

biolatency

```
#!/usr/local/bin/bpftrace

BEGIN
{
    printf("Tracing block device I/O... Hit Ctrl-C to end.\n");
}

kprobe:blk_account_io_start
{
    @start[arg0] = nsecs;
}

kprobe:blk_account_io_completion
/@start[arg0]/

{
    @usecs = hist((nsecs - @start[arg0]) / 1000);
    delete(@start[arg0]);
}
```

>120 code examples

```
bpftrace/tools> ls *.bt
bashreadline.bt  dcsnoop.bt        oomkill.bt       swapin.bt        tcpretrans.bt
biolatency.bt    execsnoop.bt      opensnoop.bt     syncsnoop.bt     tcpsynbl.bt
biosnoop.bt      gethostlatency.bt pidpersec.bt     syscount.bt      threadsnoop.bt
biostacks.bt     killsnoop.bt      runqlat.bt       tcpaccept.bt     vfscount.bt
bitesize.bt      loads.bt          runqlen.bt       tcpconnect.bt    vfsstat.bt
capable.bt       mdflush.bt        setuids.bt       tcpdrop.bt       writeback.bt
cpuwalk.bt       naptime.bt        statsnoop.bt     tcplife.bt       xfsdist.bt
bpf-perf-tools-book/originals> ls */*.bt
Ch06_CPUs/cpufreq.bt          Ch10_Networking/skblife.bt
Ch06_CPUs/execsnoop.bt        Ch10_Networking/so1stbyte.bt
Ch06_CPUs/offcputime.bt       Ch10_Networking/soaccept.bt
Ch06_CPUs/runqlat.bt          Ch10_Networking/socketio.bt
Ch06_CPUs/runqlen.bt          Ch10_Networking/socksize.bt
Ch06_CPUs/smpcalls.bt         Ch10_Networking/sockstat.bt
Ch07_Memory/brkstack.bt       Ch10_Networking/soconnect.bt
Ch07_Memory/faults.bt         Ch10_Networking/soconnlat.bt
Ch07_Memory/ffaults.bt        Ch10_Networking/sofamily.bt
Ch07_Memory/hfaults.bt        Ch10_Networking/soprotocol.bt
[...]
```

Lab 4

- bpf-perf-workshop/lab4.md

Lab 5 is optional advanced exercises

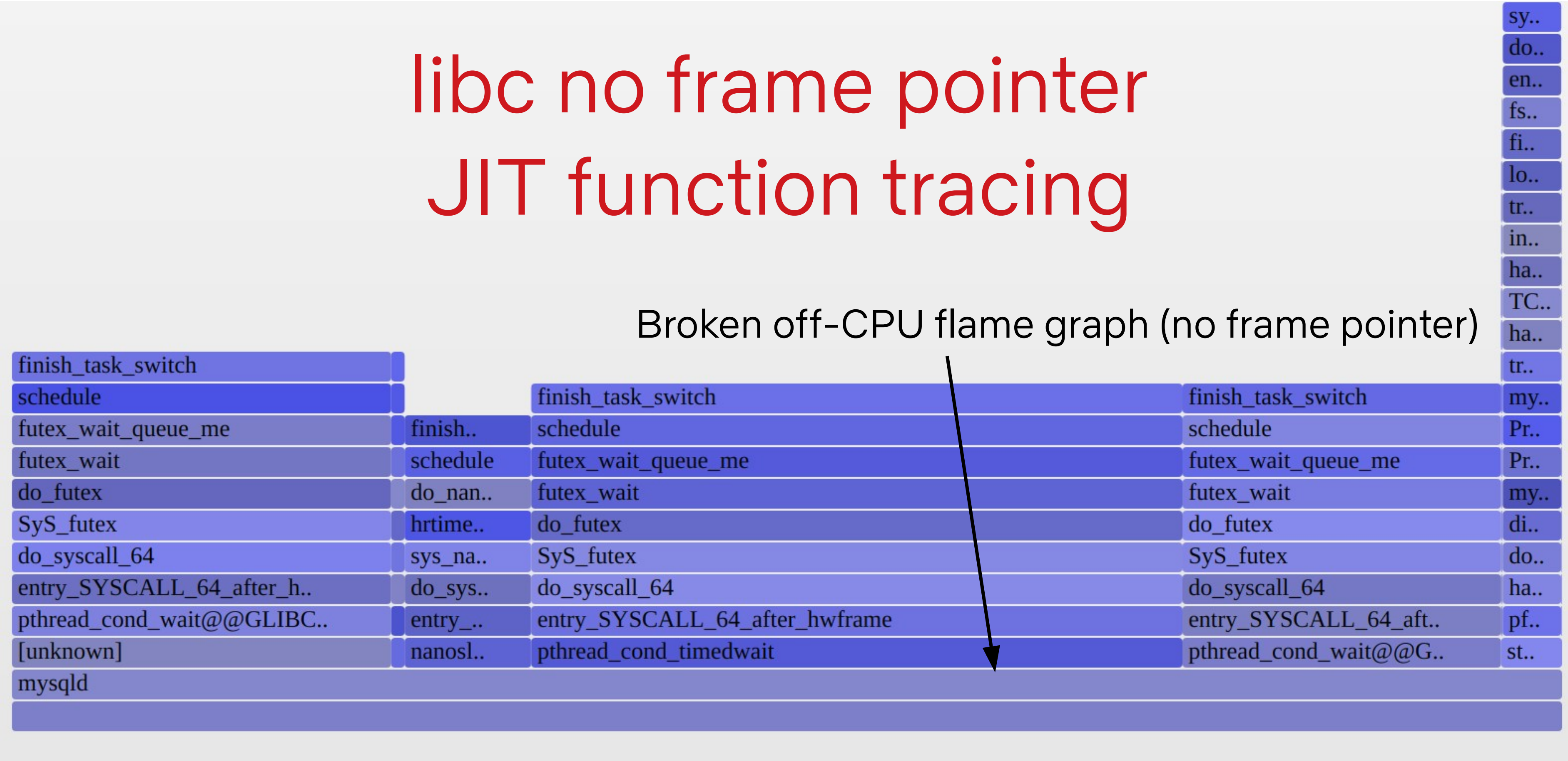
Lab 4 & 5 Discussion

Challenges

Observability Challenges

libc no frame pointer

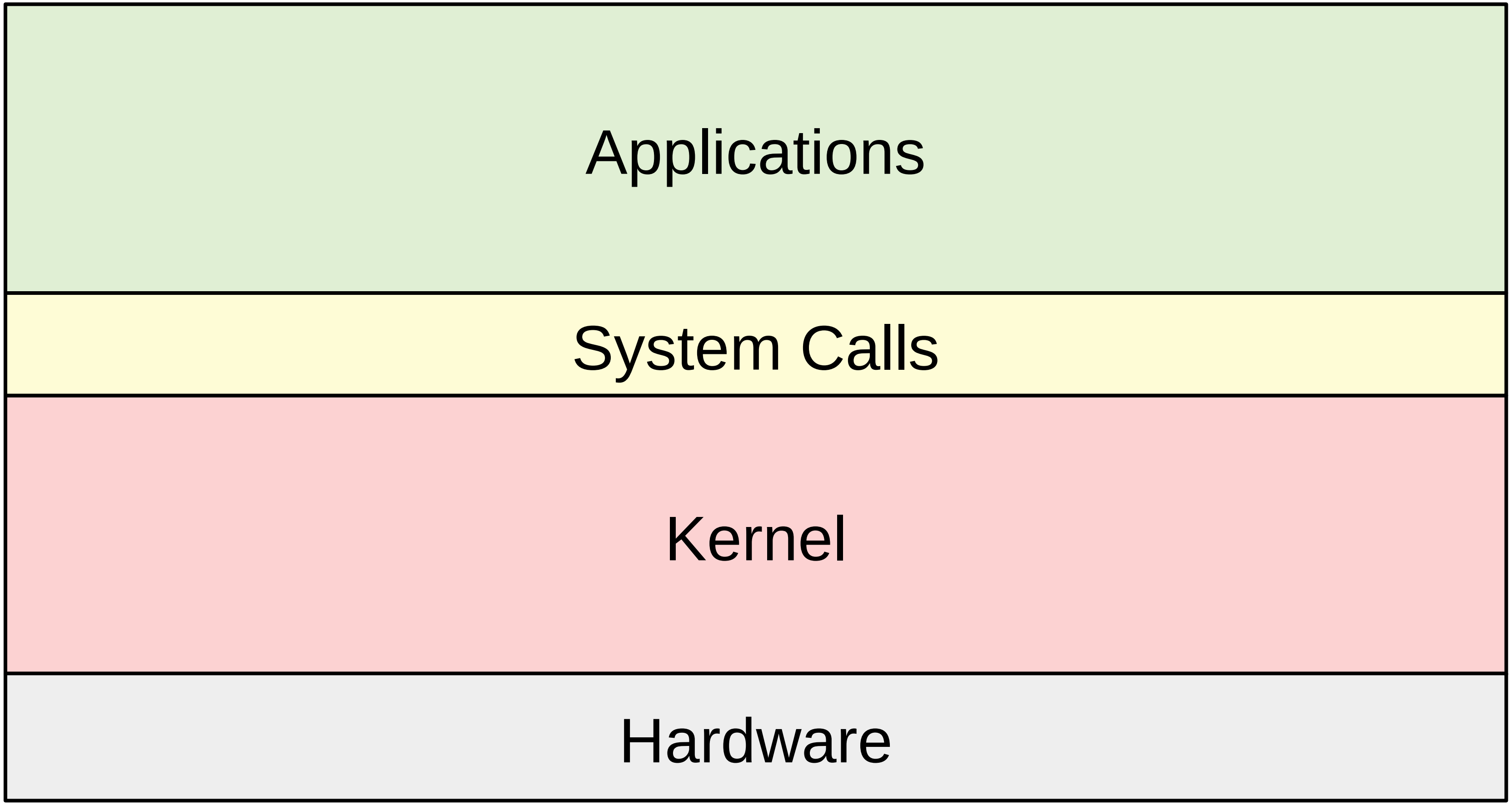
JIT function tracing



Broken off-CPU flame graph (no frame pointer)

Future

50 Years, one (dominant) OS model



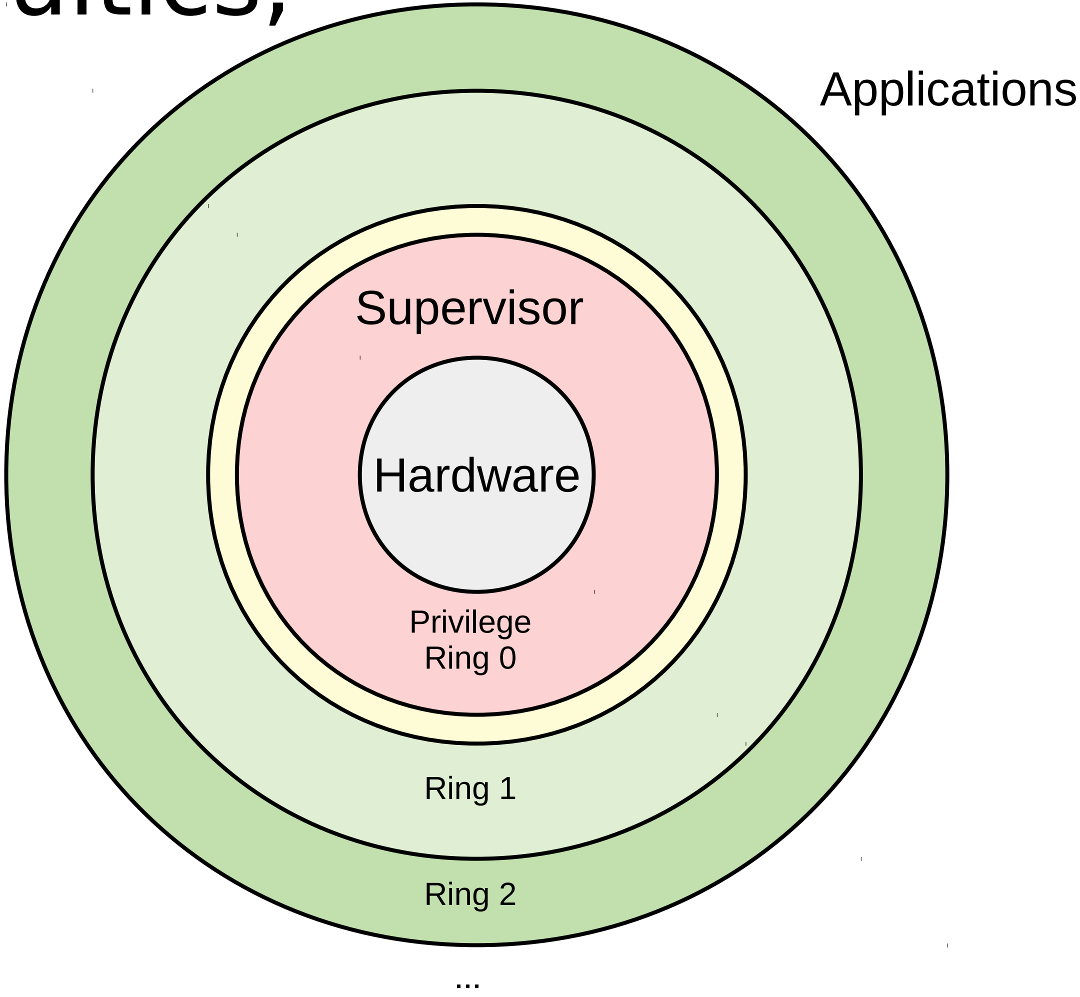
Applications

System Calls

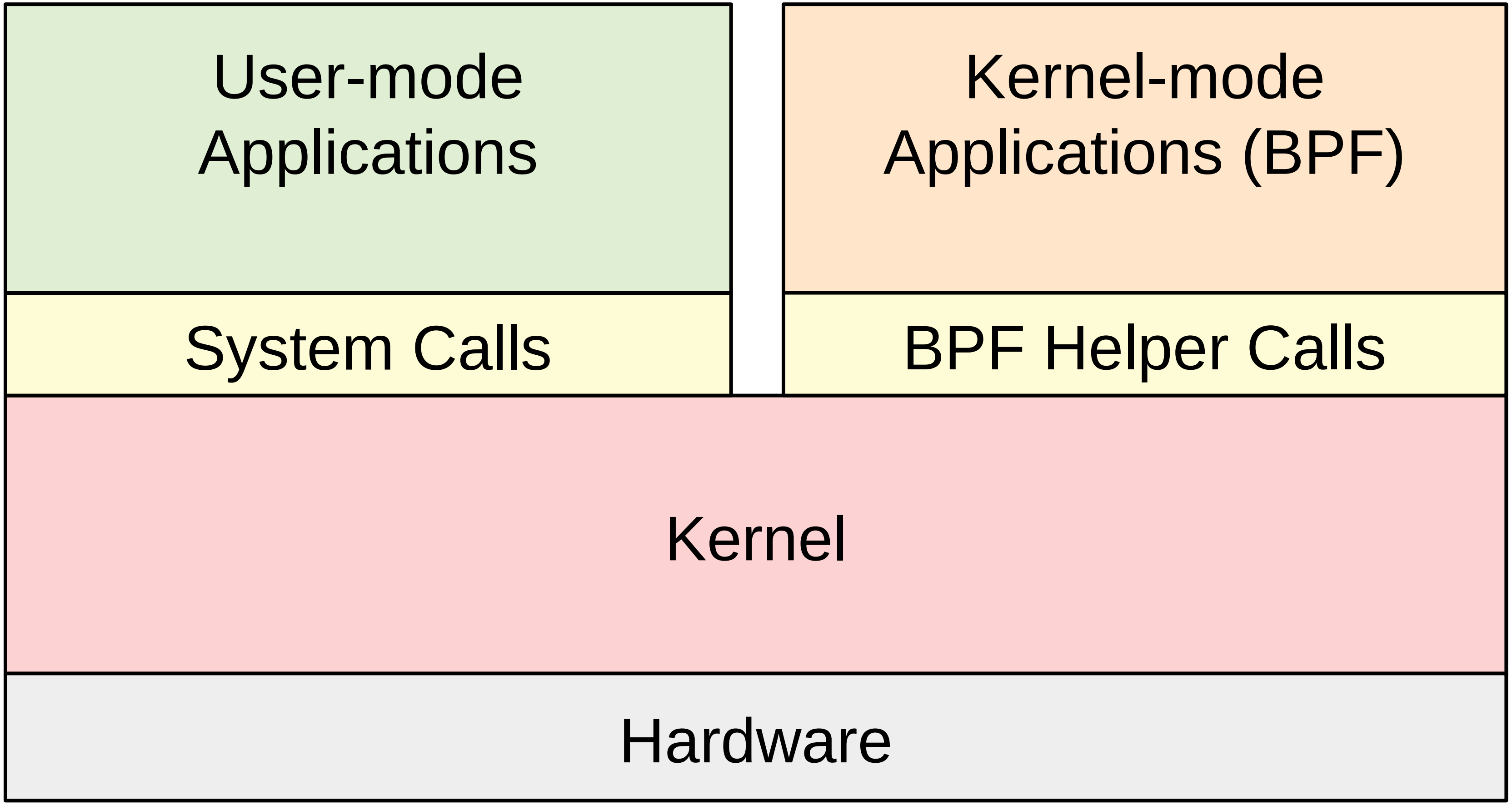
Kernel

Hardware

Origins: Multics, 1960s



Modern Linux: A new OS model



Predictions

Recap

Learning Objectives

1. slides: Understand BPF, BCC, and bpftrace
2. slides & discussion: Follow different analysis methodologies
3. lab1: Use BCC tools to analyze disk I/O issues
4. lab2: “ “ short-lived process issues
5. lab3: “ “ runq latency issues
6. lab4-5: Develop at least one new bpftrace tool

Thanks

NETFLIX

BPF: Alexei Starovoitov, Daniel Borkmann, David S. Miller, Linus Torvalds, BPF community

BCC: Brenden Blanco, Yonghong Song, Sasha Goldsthein (who also does great workshops), BCC community

bpftrace: Alastair Robertson, Matheus Marchini, Dan Xu, bpftrace community

And thanks to Jérôme Petazzoni for tips on tutorials:
http://jpetazzo.github.io/2015/09/10/how-to-deliver-great-tech-tutorials/