AWS re:Invent

OPN303

BPF performance analysis at Netflix

Brendan Gregg
Senior Performance Architect
Netflix

AWS re:Invent

aws

Superpowers Demo

AWS re:Invent

aws

Agenda

Why BPF is changing linux

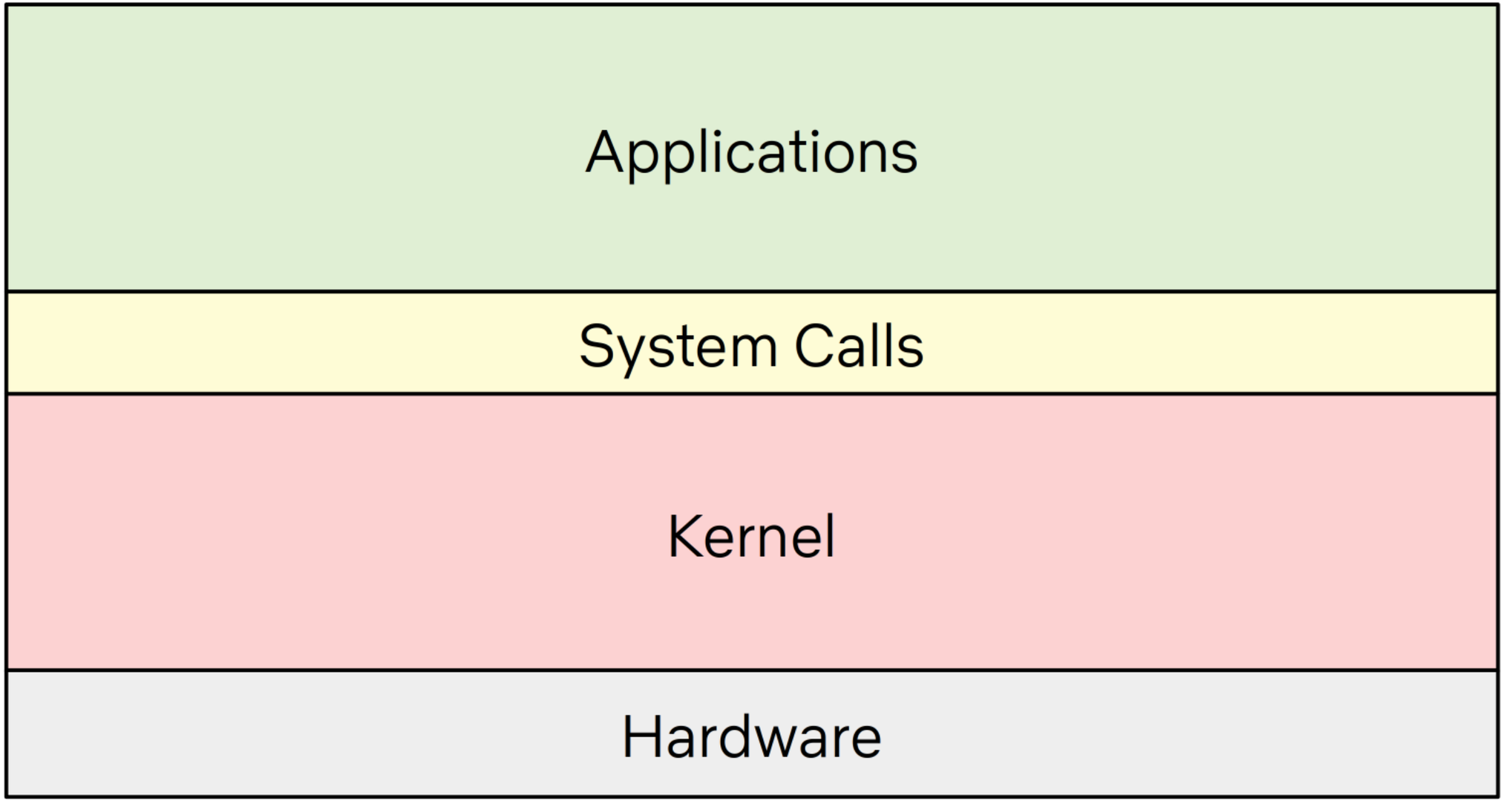
BPF internals

Performance analysis

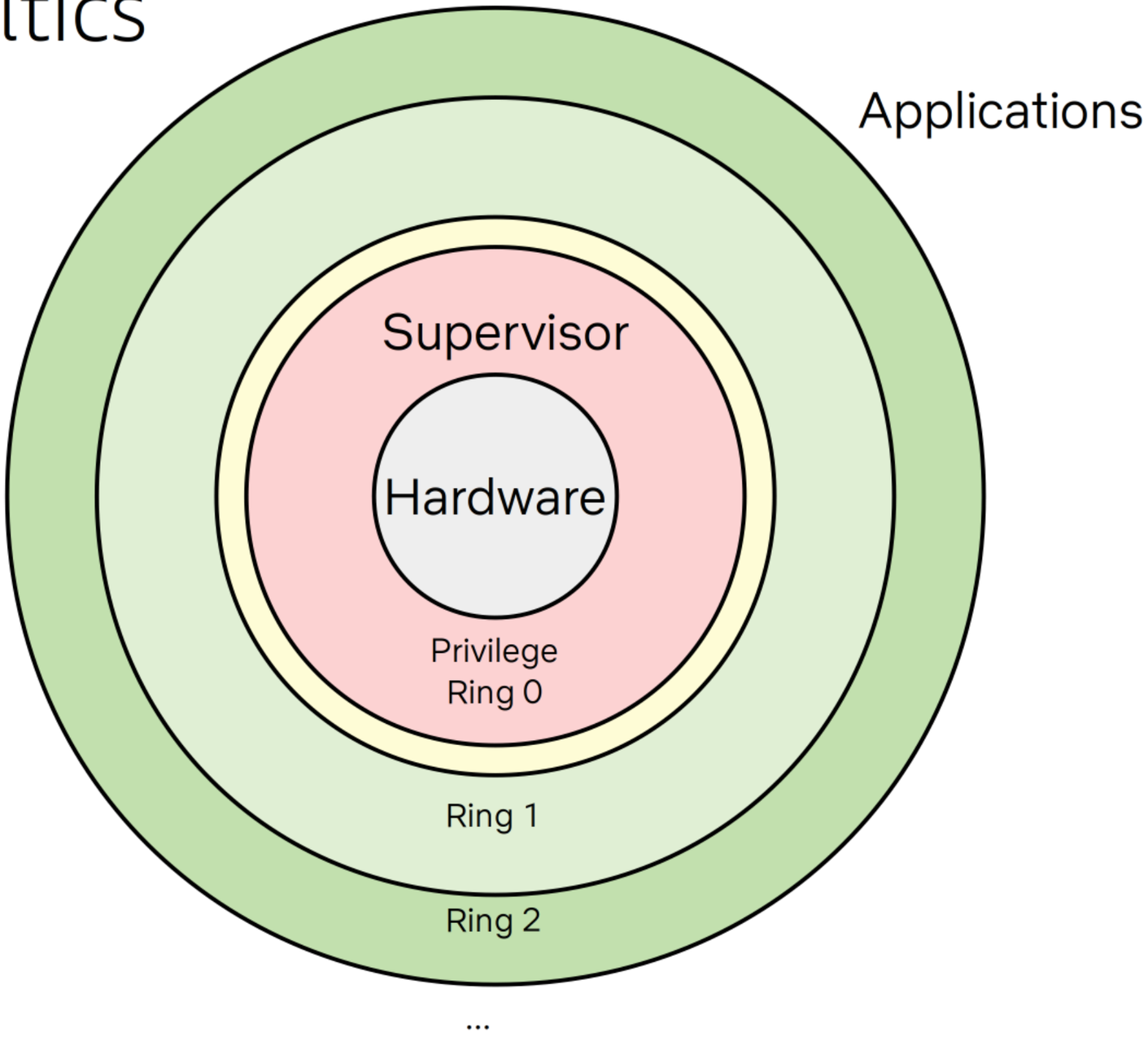
Tool development

Why BPF is changing linux

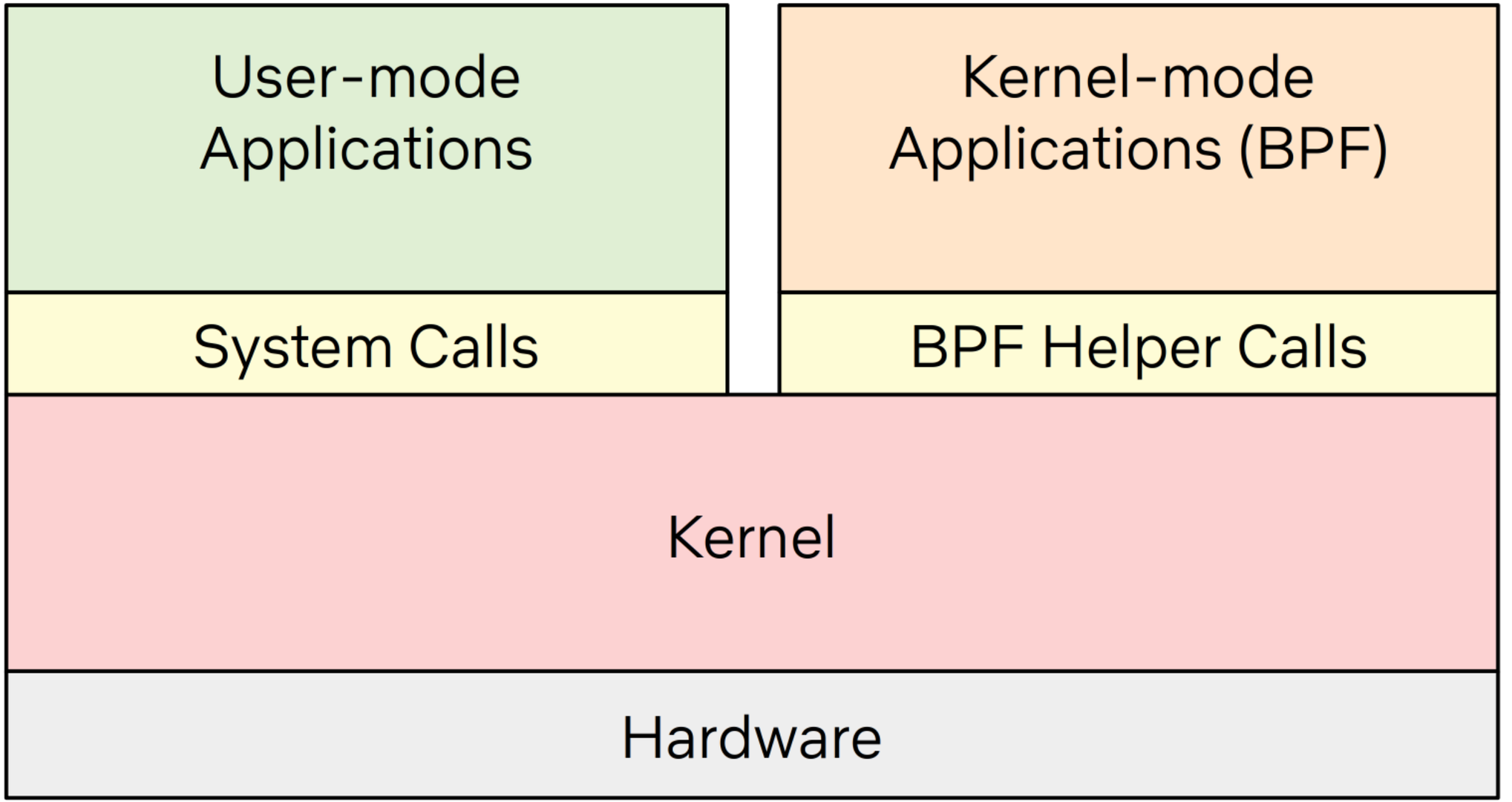
50 years, one (dominant) OS model



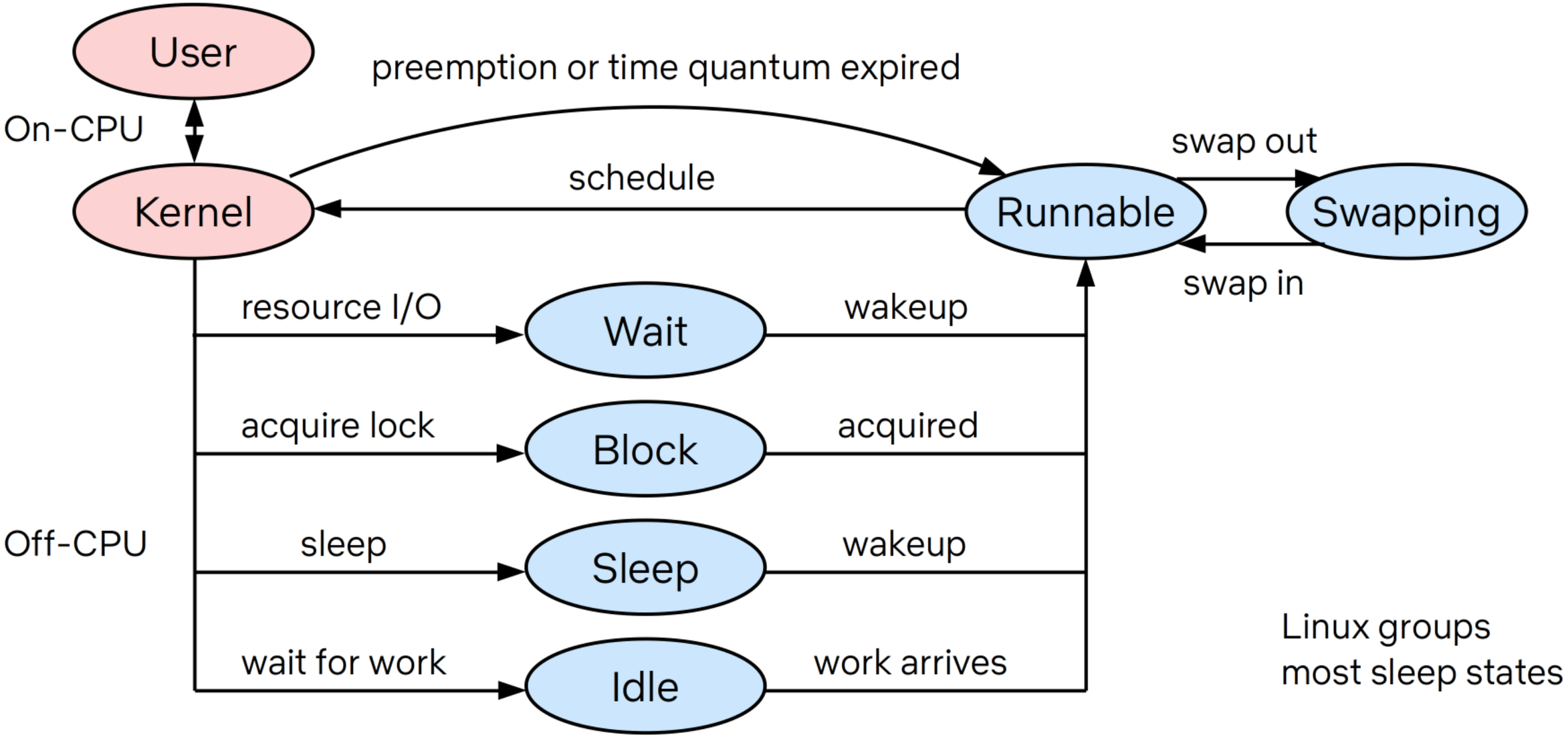
Origins: Multics 1960s



Modern Linux: a new OS model

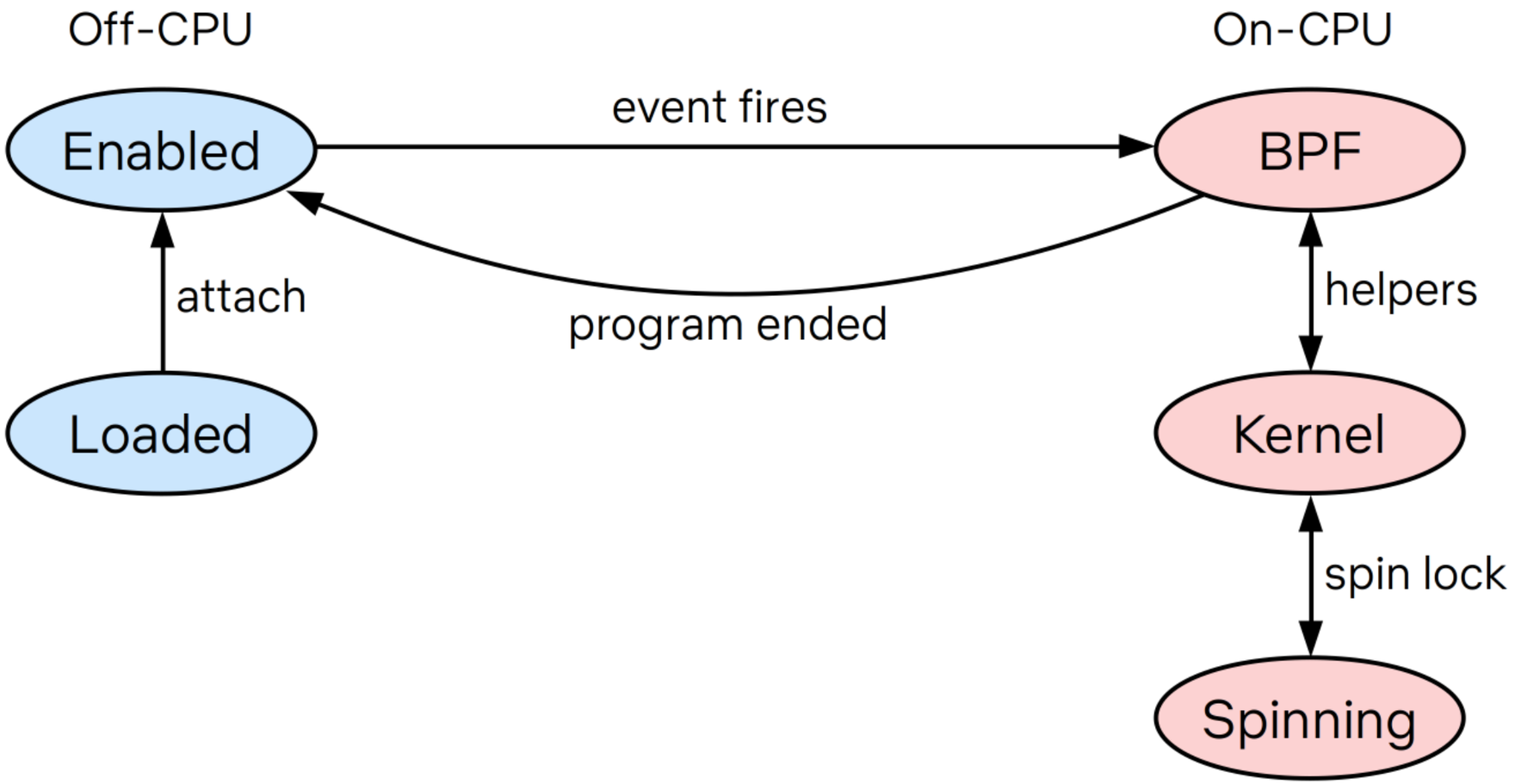


50 years, one process state model



Linux groups
most sleep states

BPF uses a new program state model



Netconf 2018
Alexei Starvoitov

BPF verifier in the future

- move away from existing brute force "walk all instructions" ap technology and static analysis
- remove #define BPF_COMPLEXITY_LIMIT 128k crutch
- remove #define BPF_MAXINSNS 4k
- support arbitrary large programs and libraries
  - 1 Million BPF instructions
- an algorithm to solve Rubik's cube will be expressible in BPF

Kernel Recipes 2019, Alexei Starovoitov

~40 active BPF programs on every Facebook server

NETFLIX

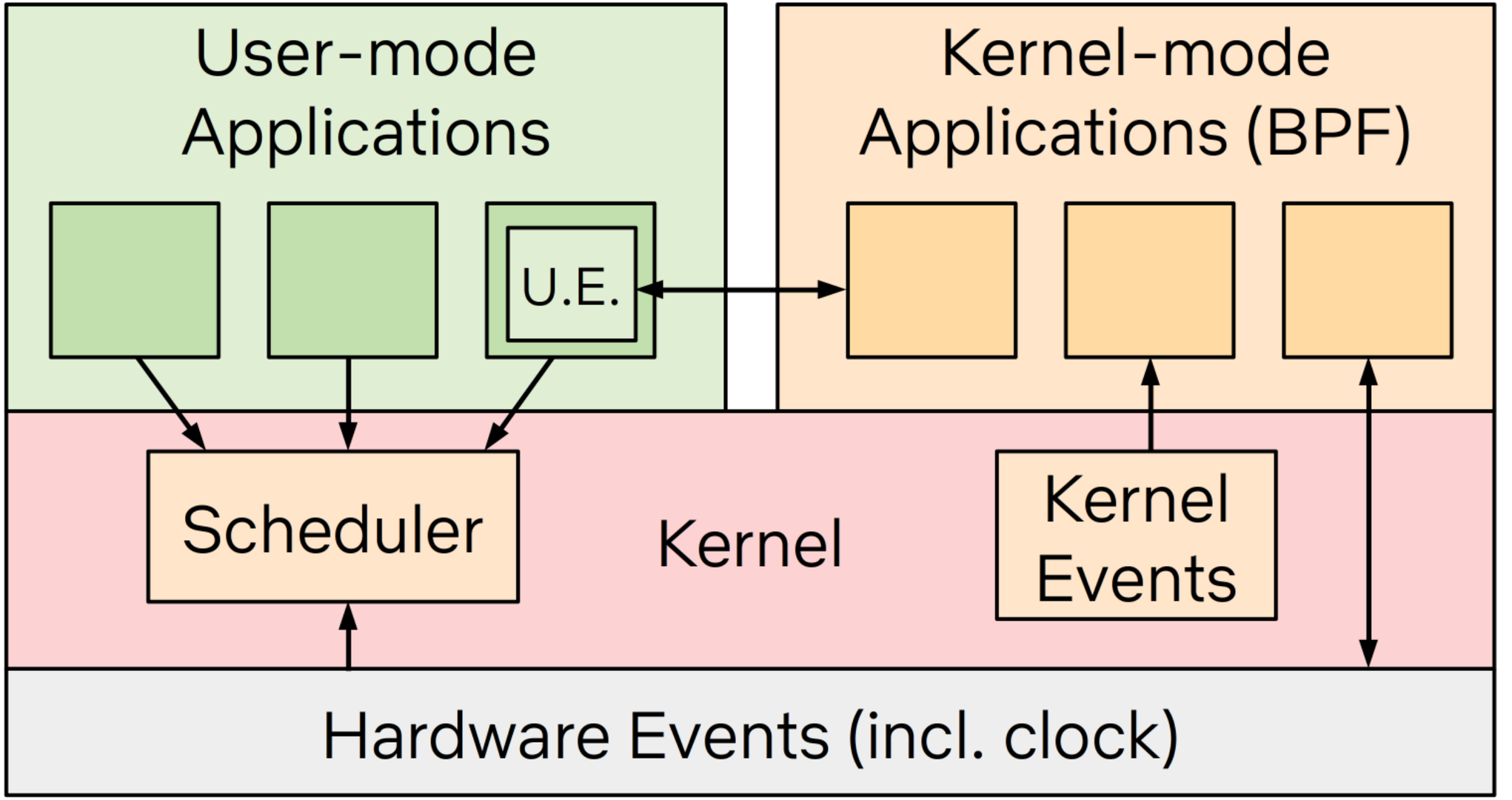
>150K Amazon EC2 server instances

~34% US Internet traffic at night

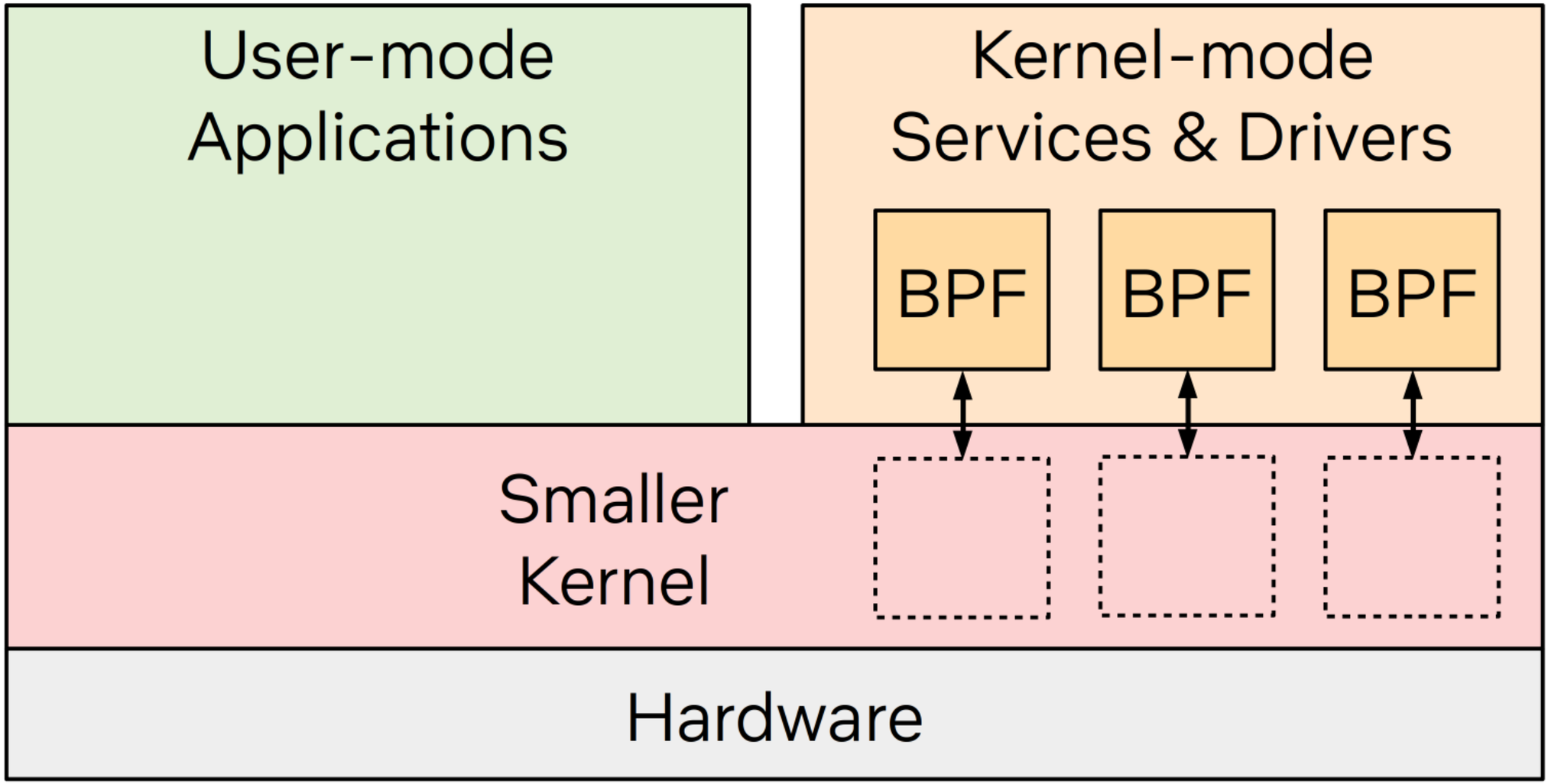
>130M subscribers

~14 active BPF programs on every instance (so far)

Modern Linux: Event-based Applications



Modern Linux is becoming microkernel-ish



The word “microkernel” has already been invoked by Jonathan Corbet, Thomas Graf, Greg Kroah-Hartman, ...

BPF internals

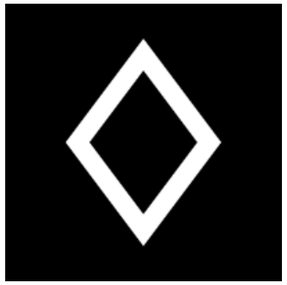
BPF 1992: Berkeley Packet Filter

```
# 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
```

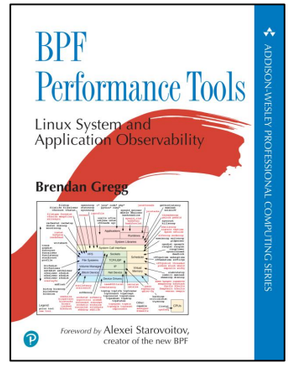
A limited
virtual machine for
efficient packet filters

BPF 2019: aka extended BPF

bpftrace



XDP



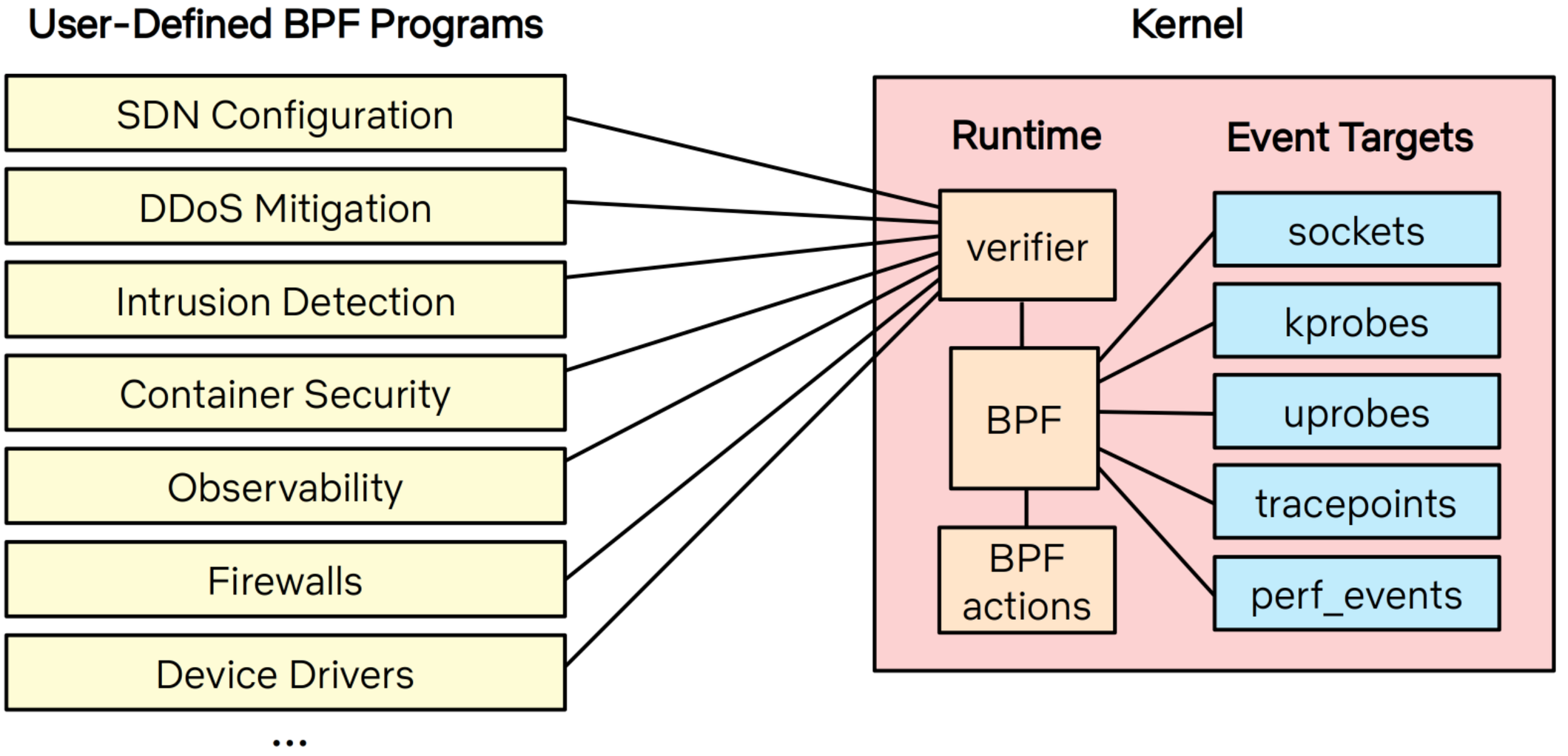
cilium

bpfconf

BPF Microconference

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

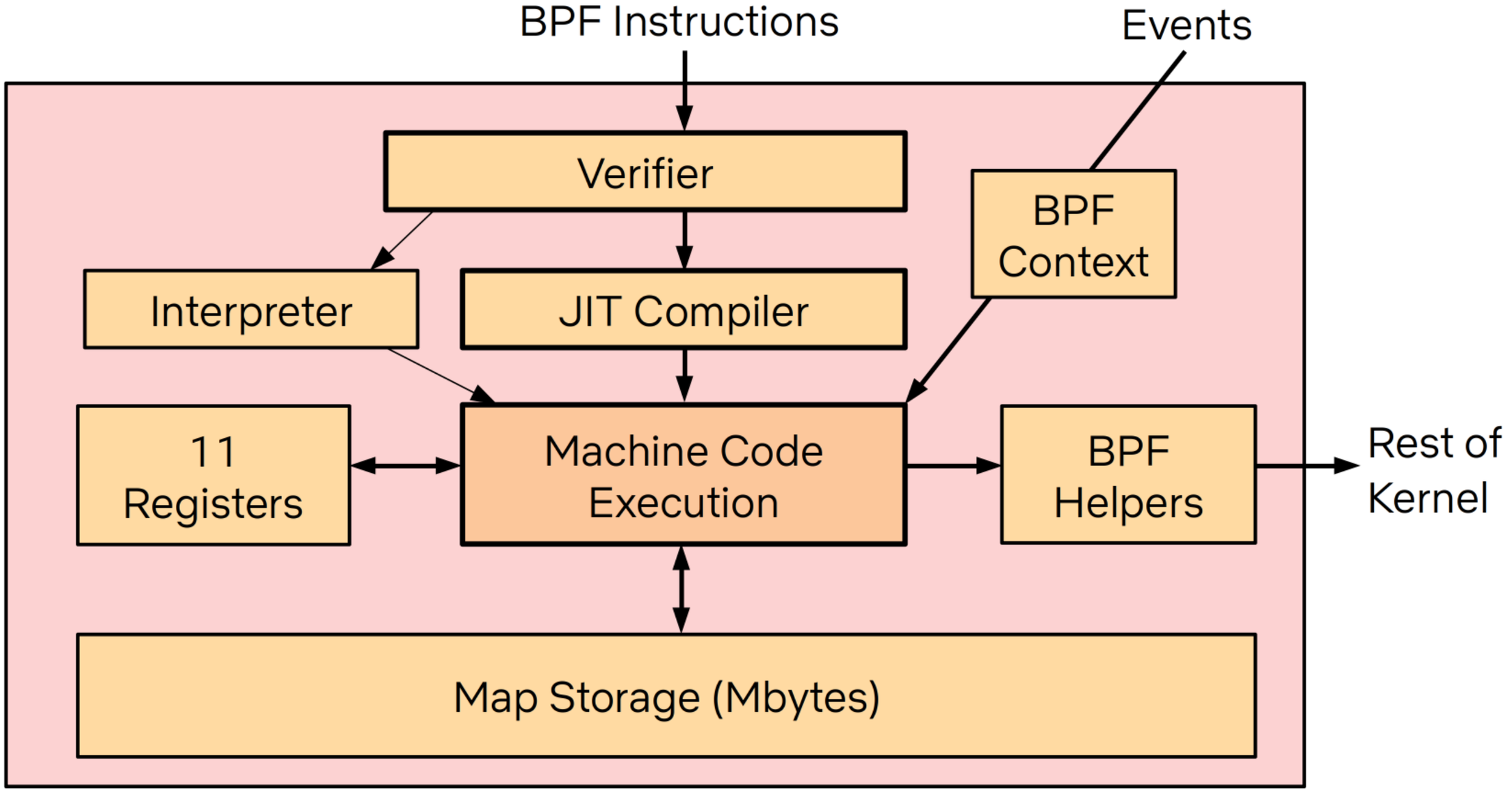
BPF 2019


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

BPF is open source and in the Linux kernel
(you're all getting it)

BPF is also now a technology name,
and no longer an acronym

BPF Internals




Is BPF Turing complete?

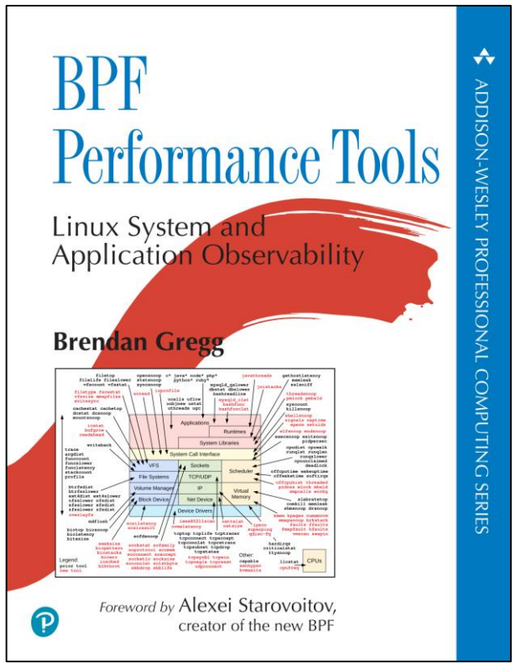
BPF: a new type of software

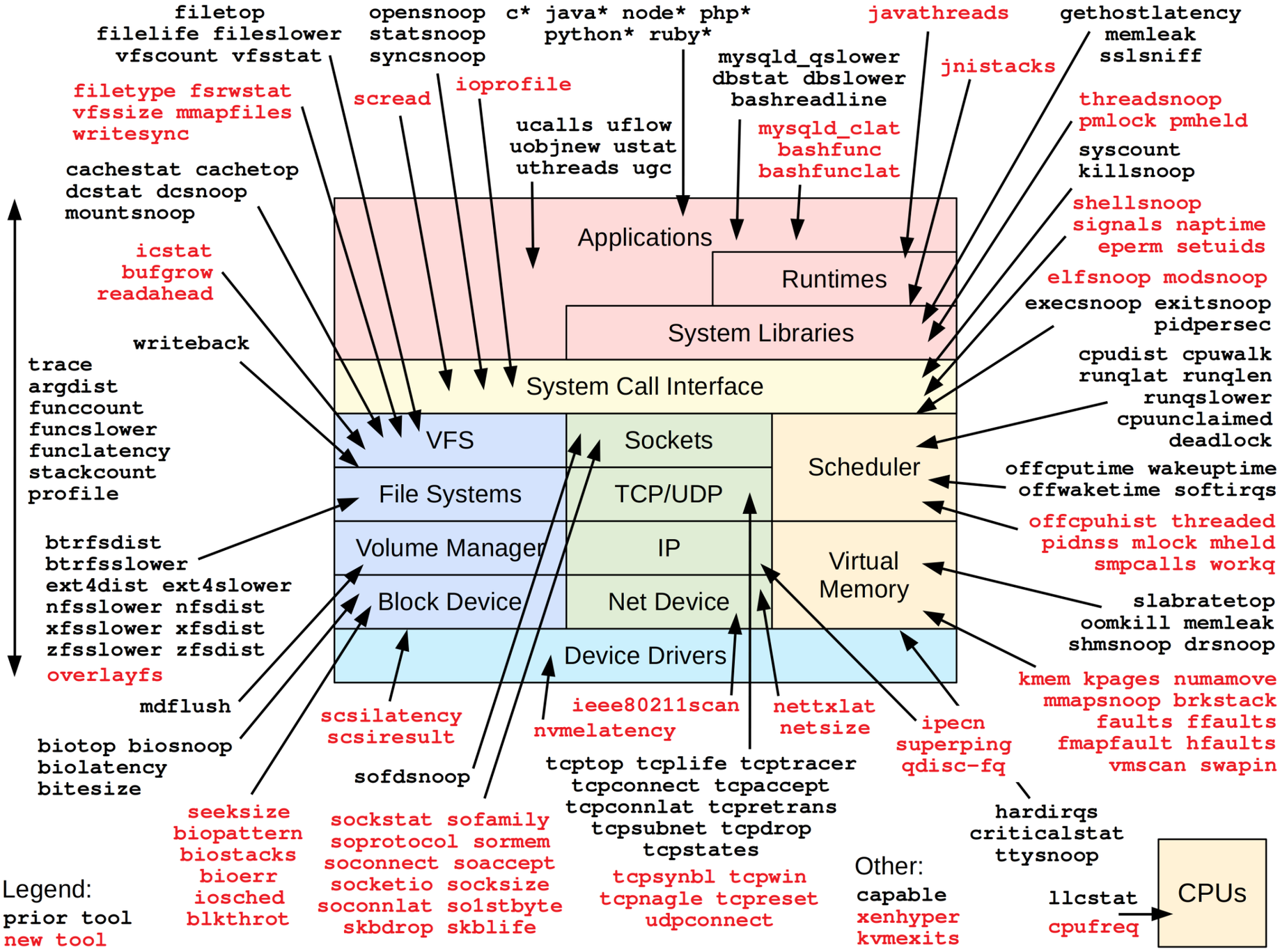
	Execution model	User-defined	Compile	Security	Failure mode	Resource access
User	task	yes	any	user-based	abort	syscall, fault
Kernel	task	no	static	none	panic	direct
BPF	event	yes	JIT, CO-RE	verified, JIT	error message	restricted helpers

Performance analysis

BPF enables a new class of
custom, efficient, and production-safe
performance analysis tools

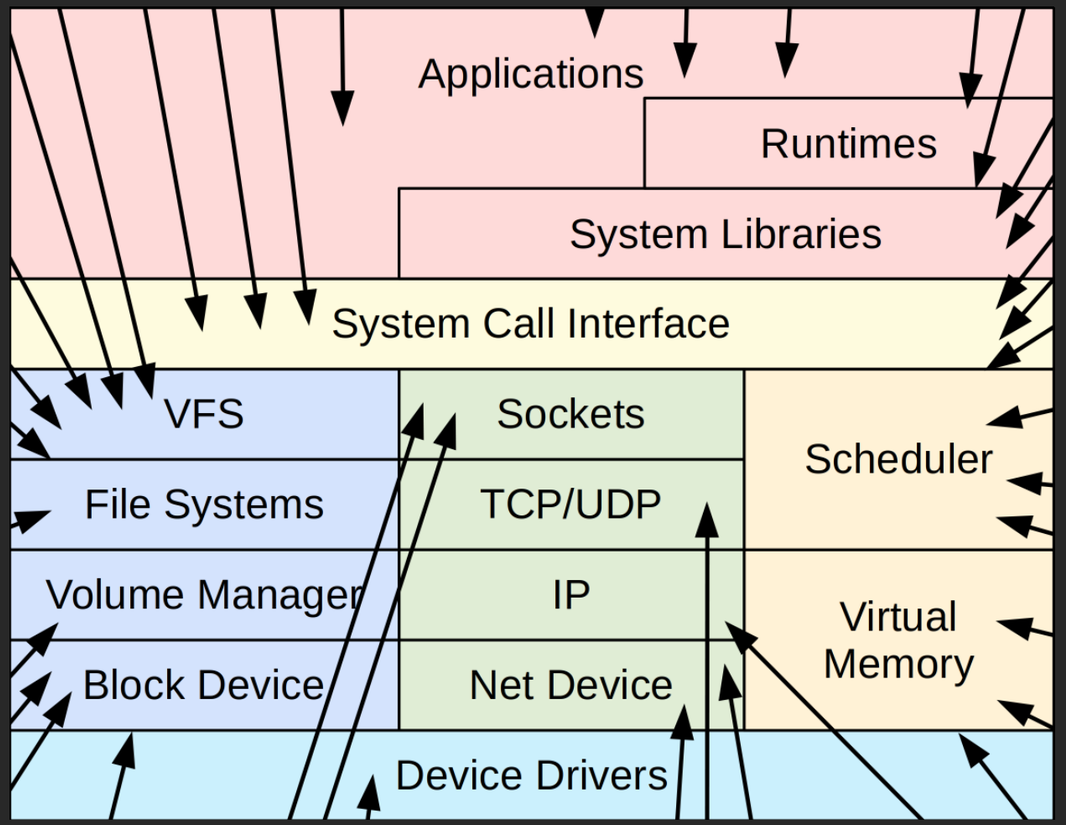
BPF Performance Tools





Tool examples by subsystem

1. CPUs (scheduling)
2. Memory
3. Disks
4. File systems
5. Networking
6. Languages
7. Applications
8. Kernel
9. Hypervisors
10. Containers



Tool extensions & sources

.py: BCC (Python)
.bt: bpftrace
(some tools exist for both)

https://github.com/iovisor/bcc
https://github.com/iovisor/bpftrace
https://github.com/brendangregg/bpf-perf-tools-book

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/;pr...
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/;pr...
[...]
```

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
[...]
```

Kernel: workq (book)

Work queue function execution times

```
# workq.bt
Attaching 4 probes...
Tracing workqueue request latencies. Ctrl-C to end.
^C

@us[blk_mq_timeout_work]:
[1]                    1 |@@                                                  |
[2, 4)                11 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[4, 8)                18 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|

@us[xfs_end_io]:
[1]                    2 |@@@@@@@@                                            |
[2, 4)                 6 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[4, 8)                 6 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[8, 16)               12 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[16, 32)              12 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[32, 64)               3 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|

[...]
```

Hypervisor: xenhyper (book)

Count hypercalls from Xen PV guests

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

@[mmu_update]: 44
@[update_va_mapping]: 78
@[mmuext_op]: 6473
@[stack_switch]: 23445
```

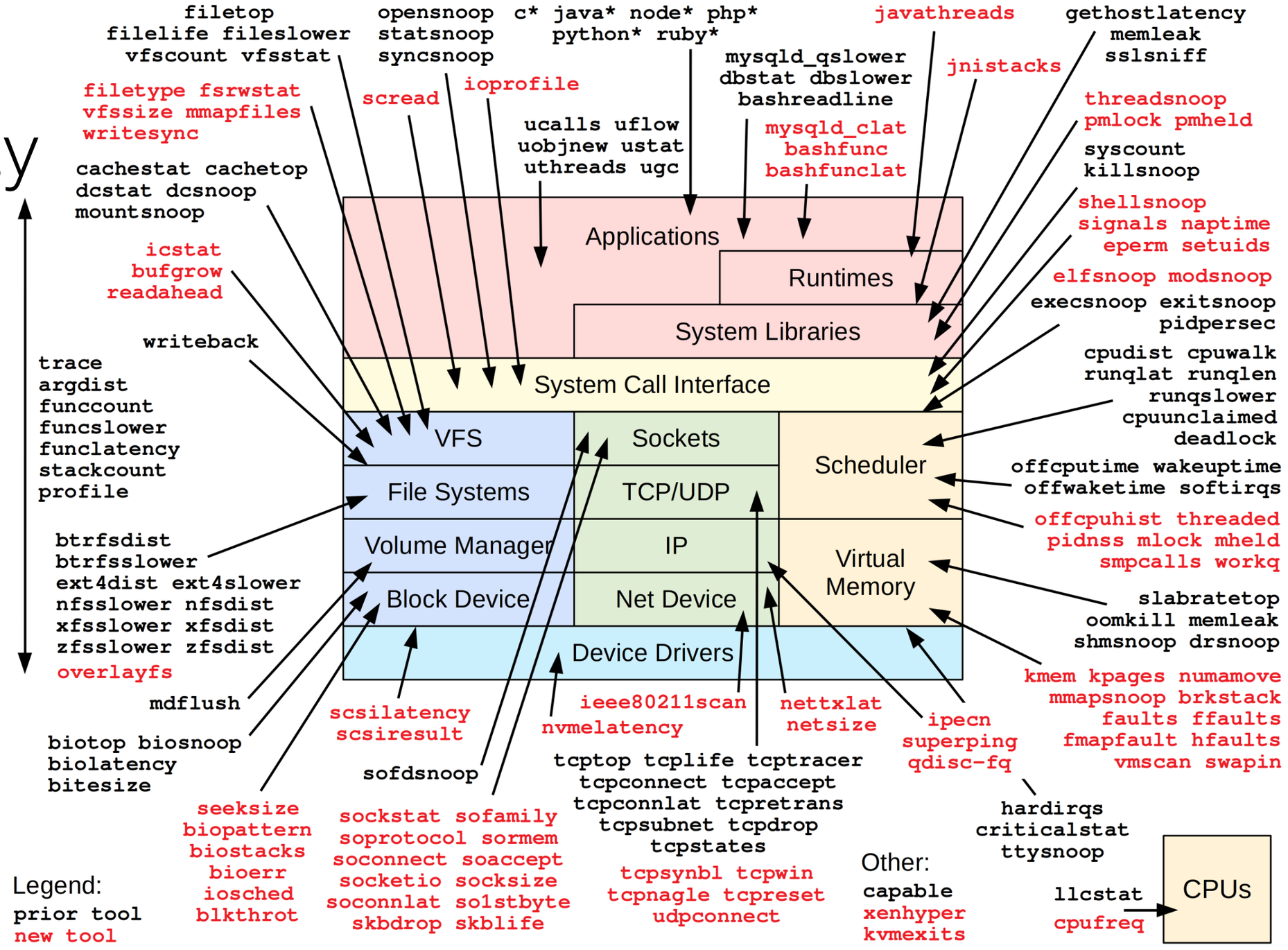
Containers: blkthrot (book)

Count block I/O throttles by blk cgroup

```
# blkthrot.bt
Attaching 3 probes...
Tracing block I/O throttles by cgroup. Ctrl-C to end
^C

@notthrottled[1]: 506

@throttled[1]: 31
```

That was only
14 out of
150+ tools

All are
open source



Not all 150+
tools shown here

Coping with so many BPF tools at Netflix

- On Netflix servers, **/apps/nflx-bpf-alltools** has all the tools
  - BCC, bpftrace, my book, Netflix internal
  - Open source at: https://github.com/Netflix-Skunkworks/**bpftoolkit**
- Latest tools are fetched & put 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/
```

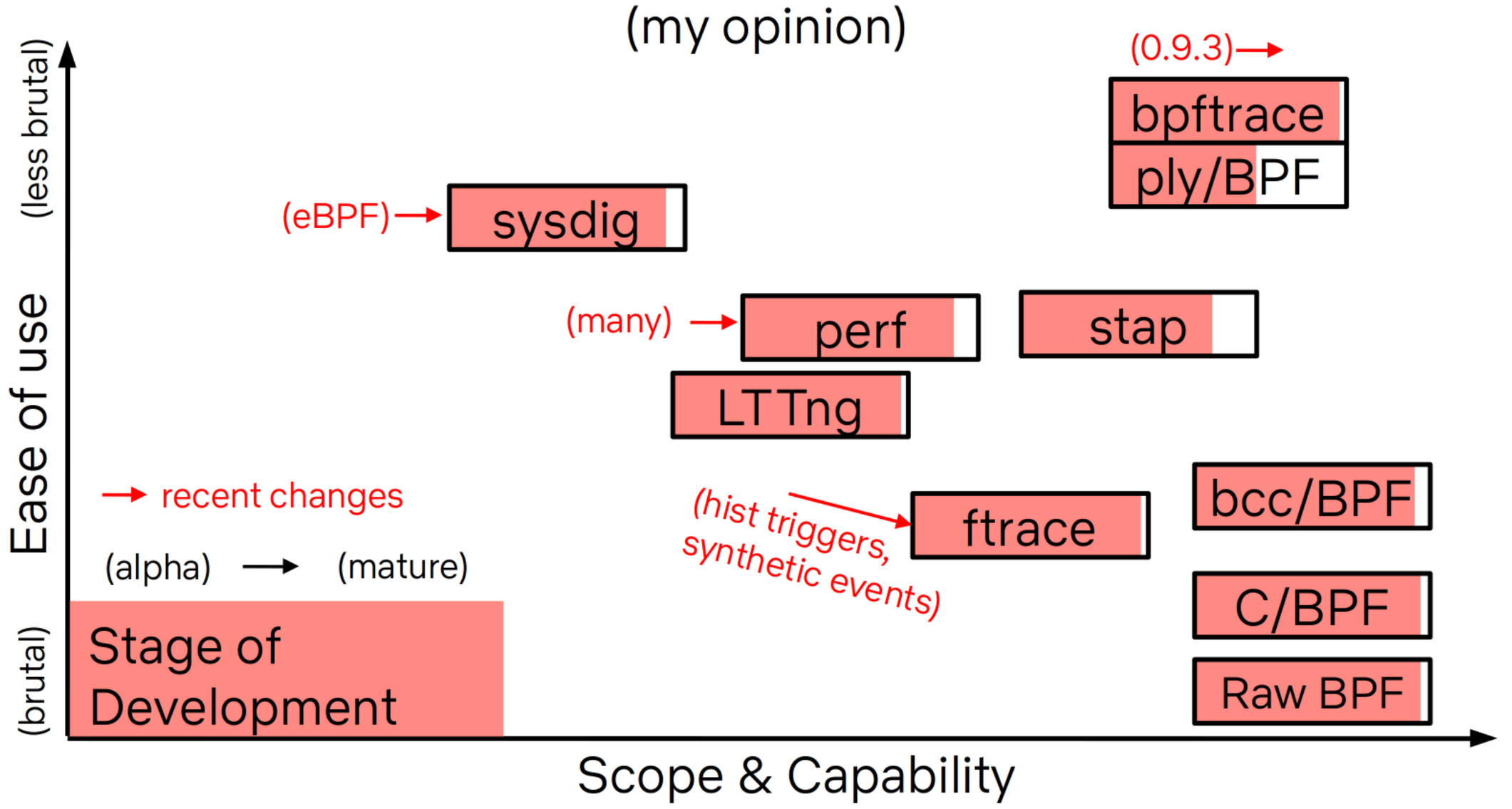
- We are also building **GUIs** to front these tools

Tool development

Only one engineer at your company needs to learn tool development

They can turn everyone’s ideas into tools

The Tracing Landscape, Dec 2019



bcc/BPF (C & Python)

```
# 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

bpftrace/BPF

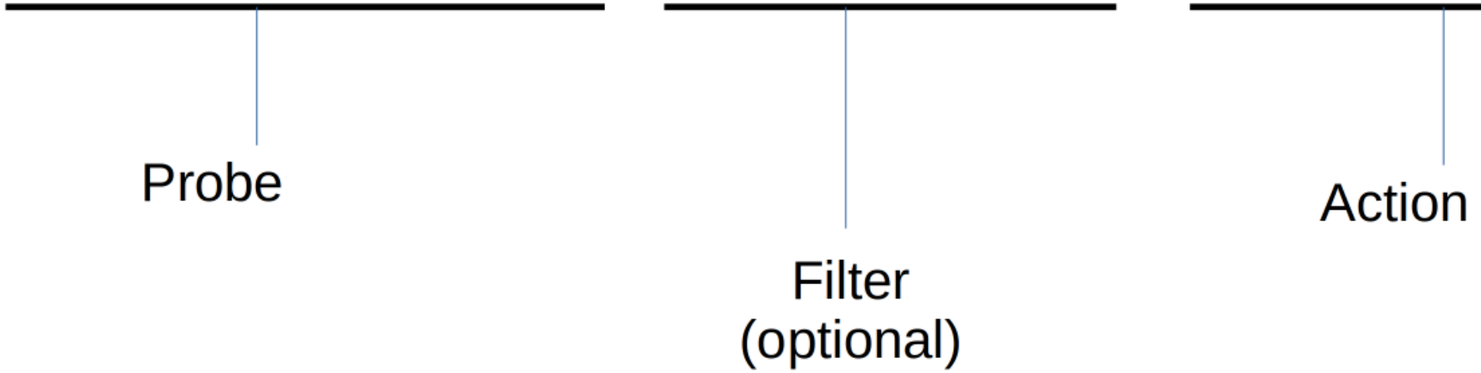
```
bpftrace -e 'kr:vfs_read { @ = hist(retval); }'
```

https://github.com/iovisor/bpftrace

entire program

bpftrace Syntax

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



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)) }’
...
```

Linux 4.9+, https://github.com/iovisor/bpftrace

Example: bpftrace biolatency

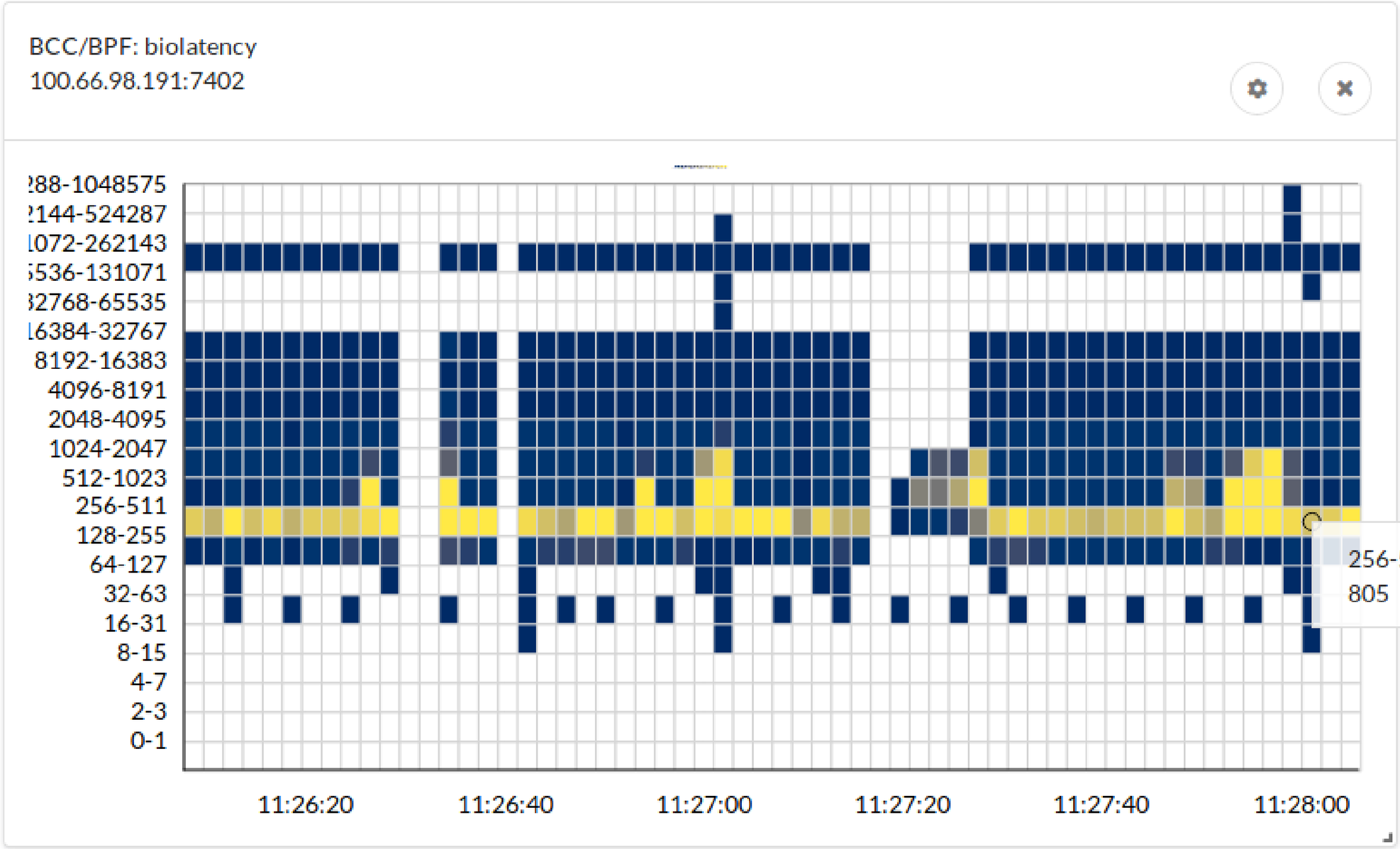
Disk I/O latency histograms, per second

```
# biolatency.bt
Attaching 3 probes...
Tracing block device I/O... Hit Ctrl-C to end.
^C

@usecs:
[256, 512)             2 |                                                    |
[512, 1K)             10 |@                                                   |
[1K, 2K)             426 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[2K, 4K)             230 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[4K, 8K)               9 |@                                                   |
[8K, 16K)            128 |@@@@@@@@@@@@@@@@                                    |
[16K, 32K)            68 |@@@@@@@@                                            |
[32K, 64K)             0 |                                                    |
[64K, 128K)            0 |                                                    |
[128K, 256K)          10 |@                                                   |
[...]
```

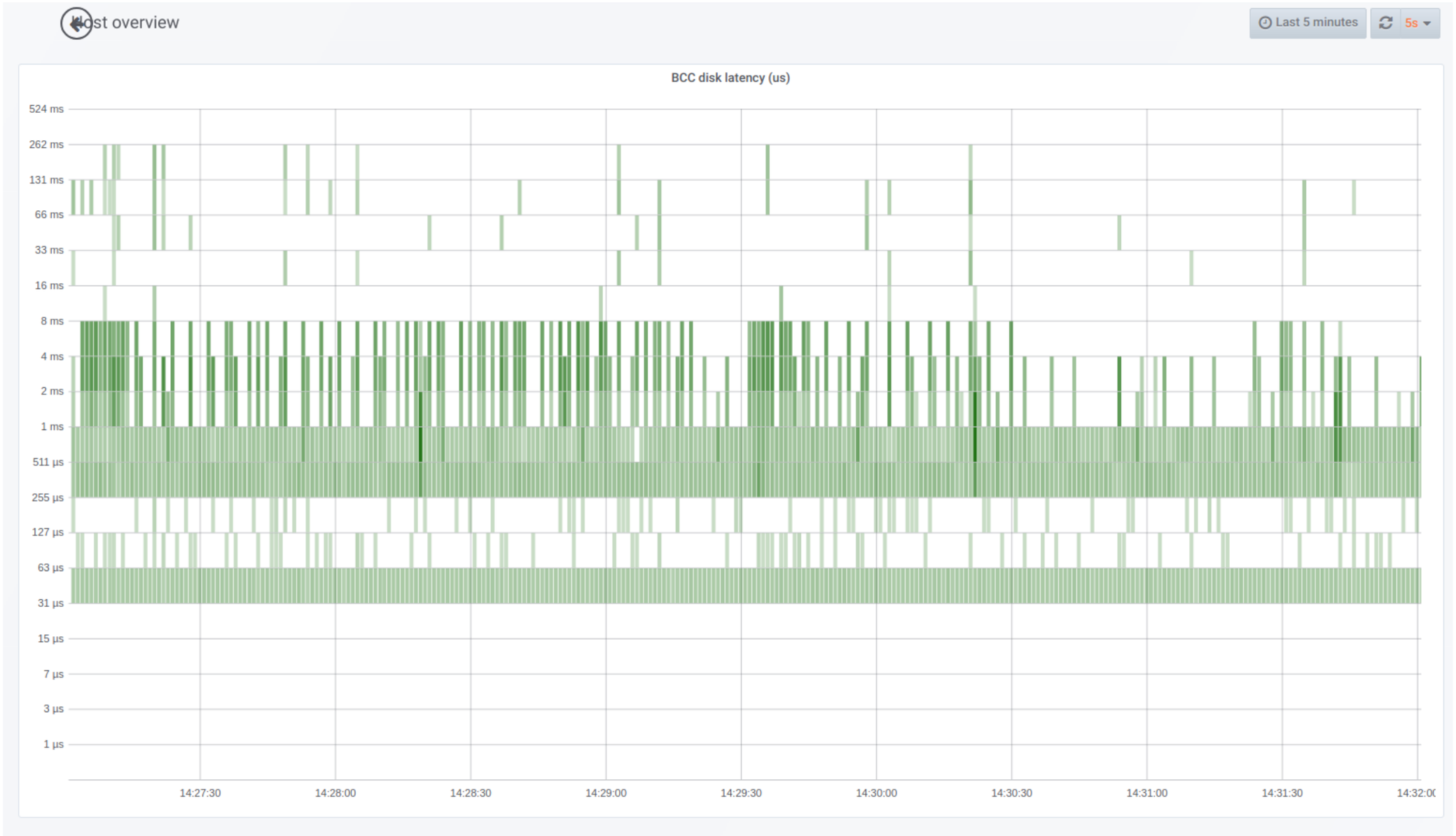
Example: bpftrace biolatency

Implemented in <20 lines of bpftrace

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



Netflix Vector
(old)



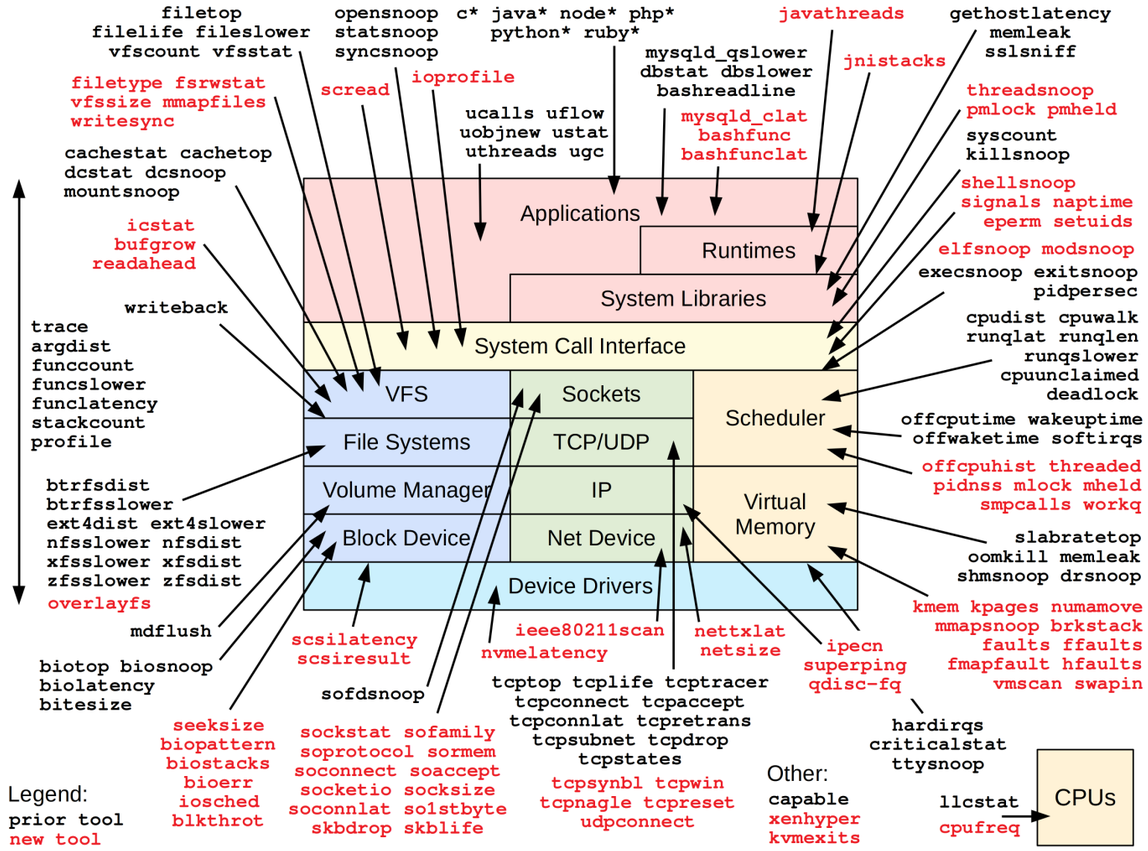
Grafana at Netflix

Takeaways

Add BCC & bpftrace packages to your servers

Start using BPF perf tools directly or via GUIs

Identify 1+ engineer at your company to develop tools & GUIs



From: BPF Performance Tools: Linux System and Application Observability, Brendan Gregg, Addison Wesley 2019

Thanks & URLs

NETFLIX

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

BCC: Brenden Blanco, Yonghong Song, Sasha Goldsthein, BCC community

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

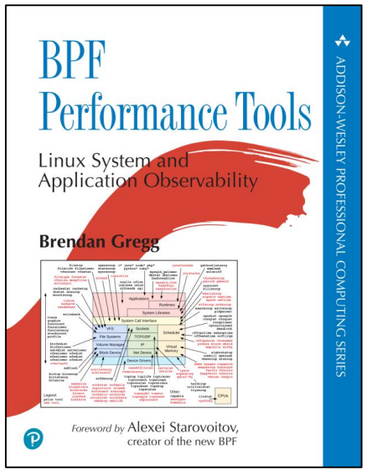
https://github.com/iovisor/bcc

https://github.com/iovisor/bpftrace

https://github.com/brendangregg/bpf-perf-tools-book

http://www.brendangregg.com/ebpf.html

http://www.brendangregg.com/bpf-performance-tools-book.html



All diagrams and photos (slides 11 & 22) are my own; slide 12 is from KernelRecipes: https://www.youtube.com/watch?v=bbHFg9IsTk8

Thank you!

Brendan Gregg

@brendangregg
bgregg@netflix.com

AWS re:Invent

aws

Please complete the session survey in the mobile app.