

# The Next Linux Superpower: eBPF Primer

Sasha Goldshtein  
CTO, Sela Group  
@goldshn

# Agenda

Modern Linux tracing landscape

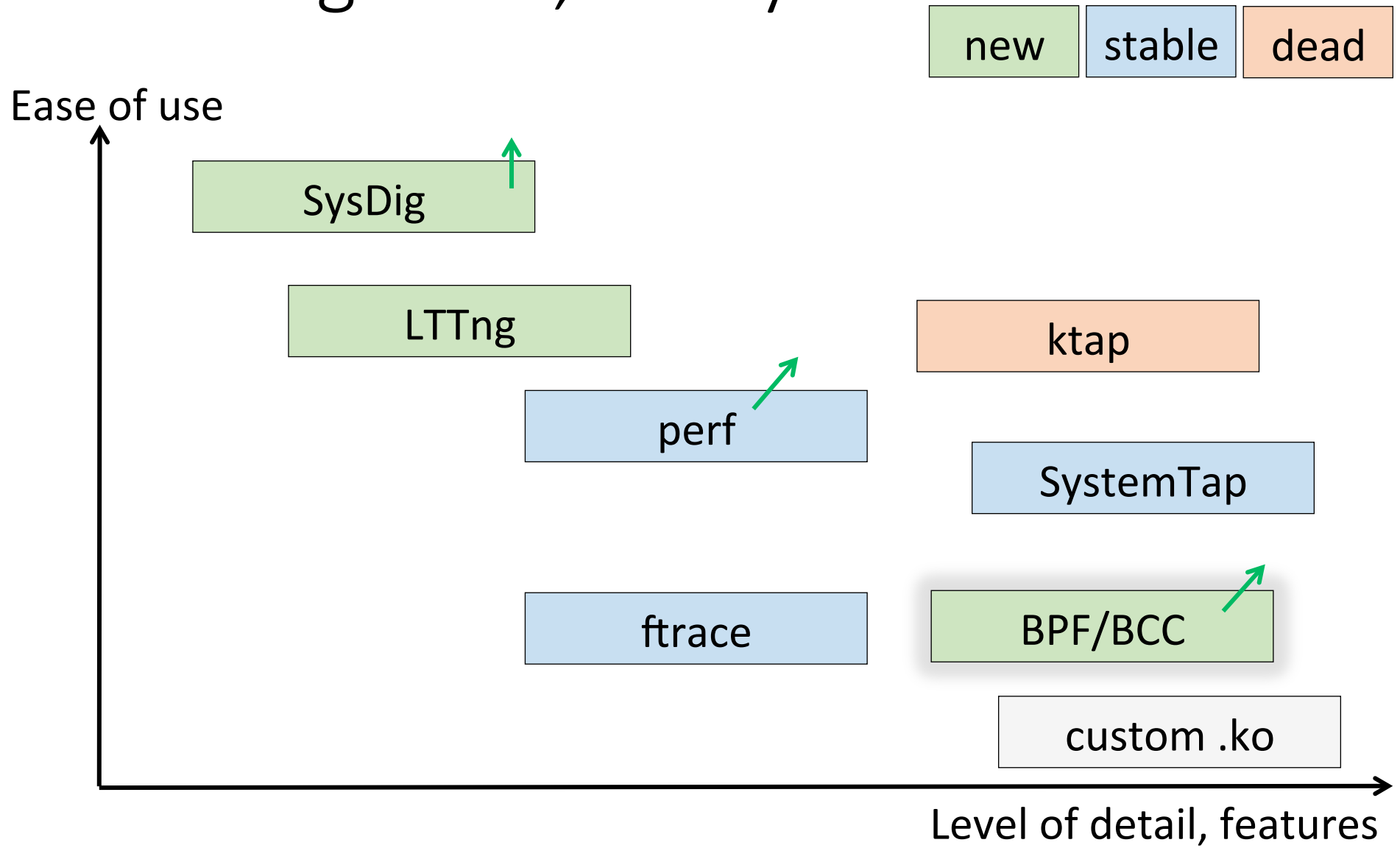
BPF

BCC – BPF Compiler Collection

Using BCC tools

Authoring BCC tools

# Linux Tracing Tools, Today



# Berkeley Packet Filters (BPF)

Originally designed for, well, packet filtering:

```
dst port 80 and len >= 100
```

Custom instruction set, interpreted/JIT compiled

Verified to be safe: no unsafe memory accesses, no backward jumps

```
0: (bf) r6 = r1
1: (85) call 14
2: (67) r0 <<= 32
3: (77) r0 >>= 32
4: (15) if r0 == 0x49f goto pc+40
```

# Extended BPF (3.18 and ongoing)

Attach BPF programs to kprobes/uprobes (4.1) and tracepoints (4.7)

Data structures: array, hash (expandable), stack map (4.6)

Output to trace buffer (4.3) and perf cyclic buffer (4.4)

Helper functions: get time, get current comm, get current CPU, etc.

# BCC: BPF Compiler Collection

Library and Python/Lua module for compiling, loading, and executing BPF programs

- Compile BPF program from C source
- Attach BPF program to kprobe/uprobe/tracepoint/USDT/socket
- Poll data from BPF program using Python/Lua

**Can do in-kernel aggregation and filtering**

Growing collection of tracing and performance tools

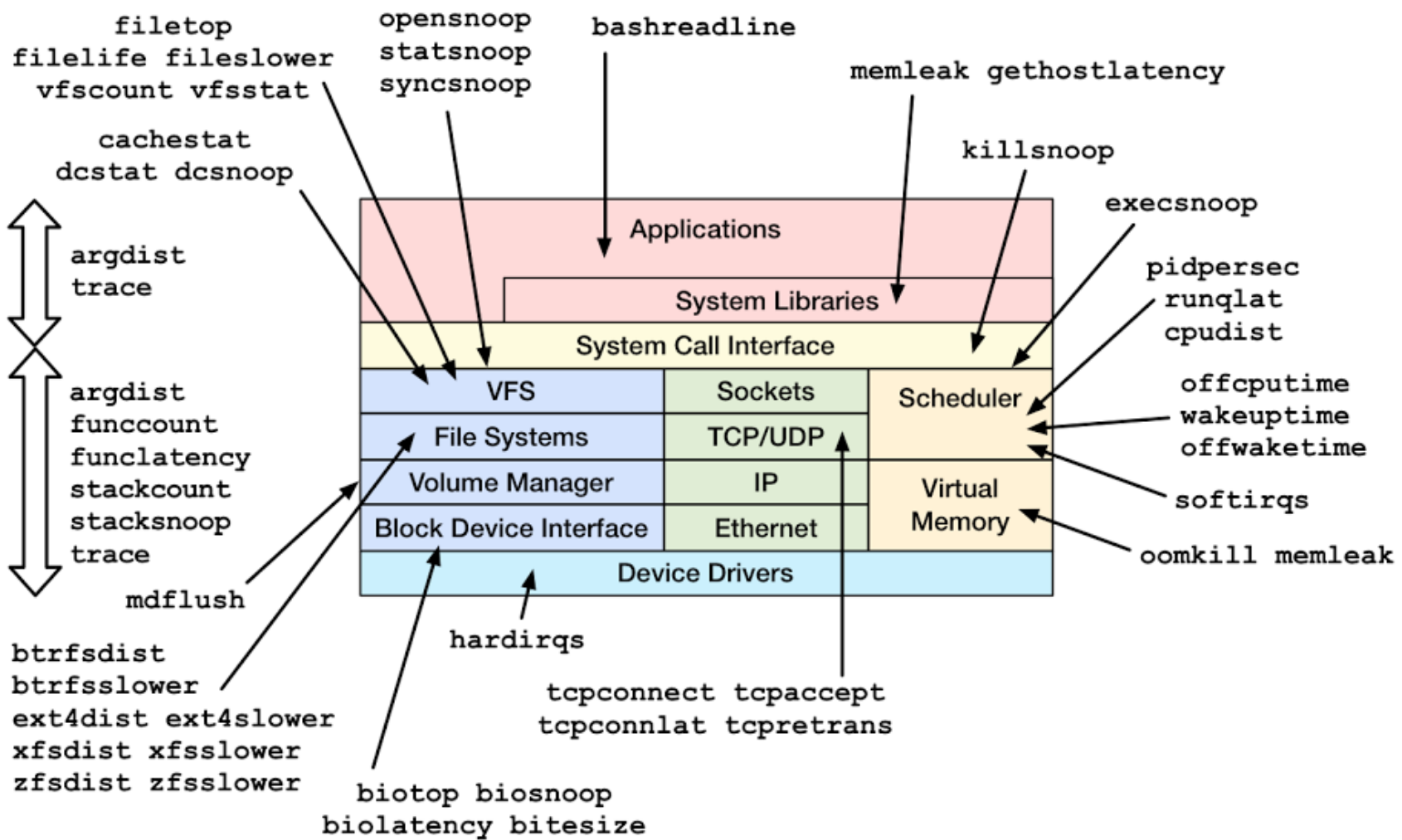
# BCC

The BCC repository contains a variety of existing scripts and tools to get you started

The BPF module (Python/Lua) can be used to build new tools or one-off scripts

```
ls.py  
rgdist.py  
ashreadline.py  
iolatency.py  
iosnoop.py  
iotop.py  
itesize.py  
trfsdist.py  
trfsslower.py  
achestat.py  
pudist.py  
csnoop.py  
cstat.py  
  
execsnoop.py  
ext4dist.py  
ext4slower.py  
filelife.py  
fileslower.py  
filetop.py  
funccount.py  
funclatency.py  
gethostlatency.py  
hardirqs.py  
killsnoop.py  
mdflush.py  
memleak.py  
  
offcputime.py  
offwaketime.py  
oomkill.py  
opensnoop.py  
pidpersec.py  
runqlat.py  
softirqs.py  
solisten.py  
stackcount.py  
stacksnoop.py  
statsnoop.py  
syncsnoop.py  
tcpaccept.py  
  
tcpconnect.py  
tcpconnlat.py  
tcpretrans.py  
tplist.py  
trace.py  
vfscount.py  
vfsstat.py  
wakeuptime.py  
xfsdist.py  
xfsslower.py  
zfsdist.py  
zfsslower.py
```

# Linux bcc/BPF Tracing Tools



<https://github.com/iovisor/bcc#tools> 2016



# Specialized Tools

**./hardirqs.py**  
Tracing hard irq event time... Hit Ctrl-C to end.

| ARDIRQ            | TOTAL_usecs |
|-------------------|-------------|
| virtio0-input.0   | 959         |
| nci[0000:00:1f.2] | 1290        |

**./biolatency.py**  
Tracing block device I/O... Hit Ctrl-C to end.

| usecs        | : count | distribution |
|--------------|---------|--------------|
| 64 -> 127    | : 7     | *****        |
| 128 -> 255   | : 14    | *****        |
| 256 -> 511   | : 5     | *****        |
| 512 -> 1023  | : 30    | *****        |
| 1024 -> 2047 | : 1     | *            |

# Specialized Tools

## ./filetop.py

01:35:51 loadavg: 0.01 0.04 0.03 2/139 3611

| ID  | COMM       | READS | WRITES | R_Kb | W_Kb | T | FILE            |
|-----|------------|-------|--------|------|------|---|-----------------|
| 496 | sshd       | 3     | 1      | 48   | 0    | 0 | ptmx            |
| 939 | screen     | 4     | 1      | 16   | 0    | 0 | ptmx            |
| 496 | sshd       | 1     | 3      | 16   | 0    | S | TCP             |
| 511 | clear      | 2     | 0      | 8    | 0    | R | screen          |
| 939 | screen     | 1     | 3      | 4    | 0    | 0 | 0               |
| 589 | filetop.py | 2     | 0      | 2    | 0    | R | loadavg         |
| 511 | clear      | 1     | 0      | 0    | 0    | R | libtinfo.so.5.9 |
| 511 | clear      | 1     | 0      | 0    | 0    | R | libc-2.21.so    |
| 511 | filetop.py | 3     | 0      | 0    | 0    | R | clear           |
| 511 | filetop.py | 2     | 0      | 0    | 0    | R | ld-2.21.so      |
| 511 | clear      | 0     | 1      | 0    | 0    | 0 | 2               |
| 589 | filetop.py | 0     | 3      | 0    | 0    | 0 | 2               |

## ./cachestat.py

| HITS | MISSES | DIRTIES | READ_HIT% | WRITE_HIT% | BUFFERS_MB | CACHED_MB |
|------|--------|---------|-----------|------------|------------|-----------|
| 0    | 0      | 0       | 0.0%      | 0.0%       | 54         | 482       |
| 842  | 0      | 0       | 100.0%    | 0.0%       | 54         | 482       |
| 889  | 128    | 0       | 87.4%     | 12.6%      | 54         | 482       |

# Specialized Tools

```
./stackcount.py __kmalloc  
Tracing 1 functions for "__kmalloc"... Hit Ctrl-C to end.
```

```
__kmalloc  
alloc_fdttable  
dup_fd  
copy_process.part.31  
do_fork  
sys_clone  
do_syscall_64  
return_from_SYSCALL_64  
4
```

```
__kmalloc  
create_pipe_files  
__do_pipe_flags  
sys_pipe  
entry_SYSCALL_64_fastpath  
6
```

```
__kmalloc  
htree_dirblock_to_tree  
ext4_htree_fill_tree  
ext4_readdir  
iterate_dir  
Sys_getdents  
entry_SYSCALL_64_fastpath  
14
```

# BPF Tracing Targets

| Target             | Support                             | Overhead                            |
|--------------------|-------------------------------------|-------------------------------------|
| kprobes            | Native                              | Low                                 |
| uprobes            | Native                              | Medium<br><i>handler runs in KM</i> |
| Kernel tracepoints | Native <b>NEW</b>                   | Low                                 |
| USDT tracepoints   | Temporary<br><i>through uprobes</i> | Medium<br><i>handler runs in KM</i> |

# Multi-Tools: argdist

```
./argdist.py -C 'p:c:write(int fd, const void *buf,  
size_t count):size_t:count:fd==1'  
01:49:00]  
:c:write(int fd, const void *buf, size_t count):size_t:count:fd==1  
COUNT      EVENT  
1            count = 3134  
1            count = 170  
1            count = 181  
2            count = 18  
3            count = 30
```

# Multi-Tools: argdist

```
./argdist.py -i 5 -H 'r::__vfs_read(void *file, void *buf, size_t count):size_t:$entry(count):$latency > 1000000'
```

01:51:40]

| count        | : count | distribution |
|--------------|---------|--------------|
| 0 -> 1       | : 20    | *****        |
| 2 -> 3       | : 0     |              |
| 4 -> 7       | : 0     |              |
| 8 -> 15      | : 0     |              |
| 16 -> 31     | : 0     |              |
| 32 -> 63     | : 0     |              |
| 64 -> 127    | : 0     |              |
| 128 -> 255   | : 6     | *****        |
| 256 -> 511   | : 0     |              |
| 512 -> 1023  | : 0     |              |
| 1024 -> 2047 | : 1     | **           |

# Multi-Tools: trace

```
./trace.py 'r:/usr/bin/bash:readline "%s", retval'
```

| TIME     | PID  | COMM | FUNC     | -           |
|----------|------|------|----------|-------------|
| 02:02:26 | 3711 | bash | readline | ls -la      |
| 02:02:36 | 3711 | bash | readline | wc -l src.c |

```
./tplist.py -v block:block_rq_complete
```

```
block:block_rq_complete  
dev_t dev;  
sector_t sector;  
unsigned int nr_sector;  
int errors;  
char rwbs[8];
```

```
./trace.py 't:block:block_rq_complete "sectors=%d", tp.nr_sector'
```

| TIME     | PID | COMM      | FUNC              | -          |
|----------|-----|-----------|-------------------|------------|
| 02:03:56 | 0   | swapper/0 | block_rq_complete | sectors=16 |
| 02:03:56 | 0   | swapper/0 | block_rq_complete | sectors=8  |
| 02:03:58 | 0   | swapper/0 | block_rq_complete | sectors=24 |
| 02:04:00 | 0   | swapper/0 | block_rq_complete | sectors=0  |

# Multi-Tools: trace

```
./tplist.py -l pthread -v libpthread:pthread_create  
usr/lib64/libpthread-2.21.so libpthread:pthread_create  
location 0x7c63
```

```
8 unsigned bytes @ register %rax  
8 unsigned bytes @ -192(%rbp)  
8 unsigned bytes @ -168(%rbp)  
8 unsigned bytes @ -176(%rbp)
```

```
./trace.py 'u:pthread:pthread_create "%11x", arg3'
```

| TIME    | PID  | COMM        | FUNC           | -      |
|---------|------|-------------|----------------|--------|
| 2:07:29 | 4051 | contentions | pthread_create | 400e00 |
| 2:07:29 | 4051 | contentions | pthread_create | 400e00 |
| 2:07:29 | 4051 | contentions | pthread_create | 400e00 |
| 2:07:29 | 4051 | contentions | pthread_create | 400e00 |

```
C
```



# Multi-Tools: trace

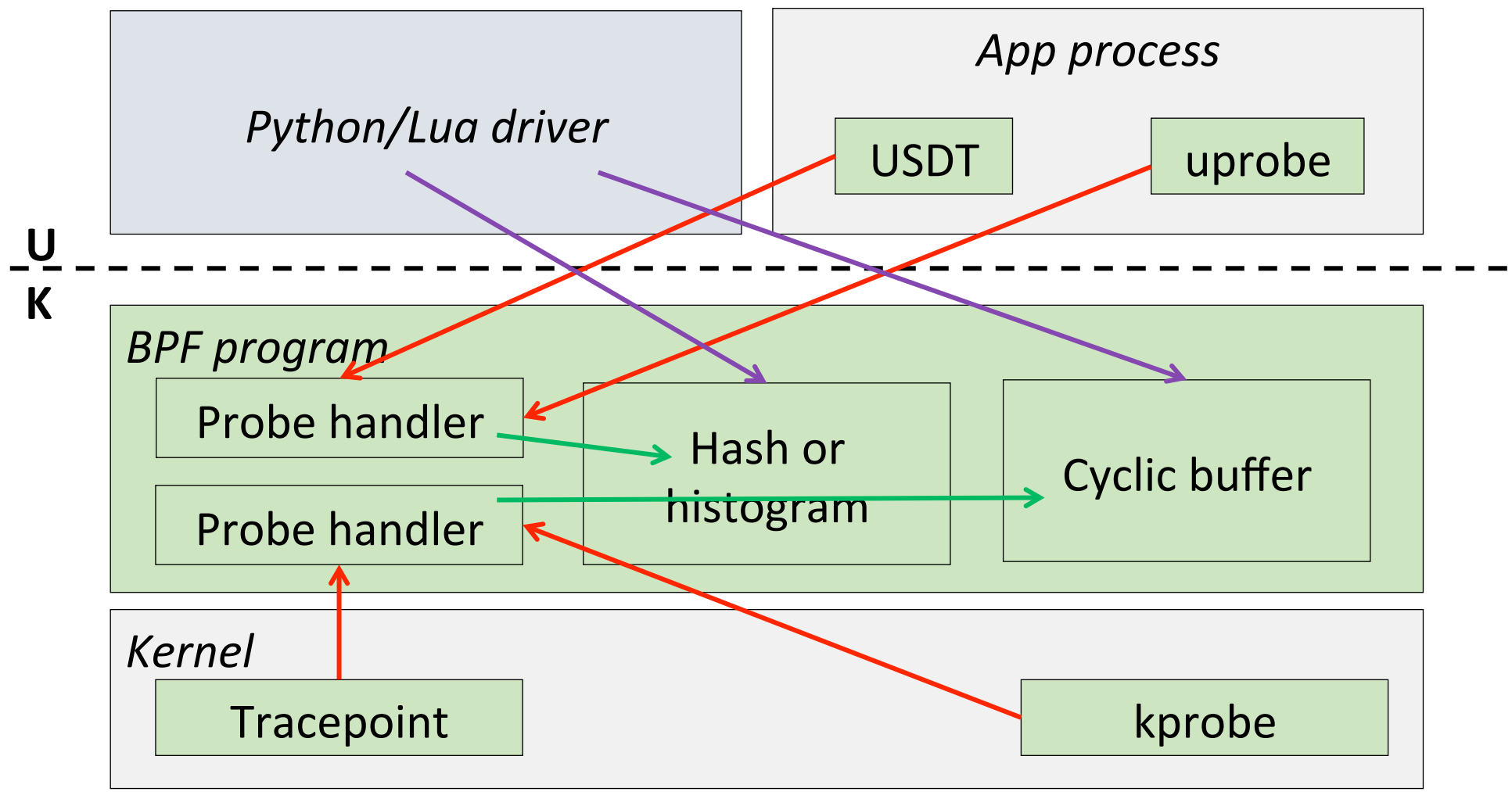
```
trace -p $(pidof node) 'u:node:http__server__request
"%s %s (from %s:%d)" arg5, arg6, arg3, arg4'
```

| TIME    | PID   | COMM | FUNC                  | -                            |
|---------|-------|------|-----------------------|------------------------------|
| 4:50:44 | 22185 | node | http__server__request | GET /foofoo (from ::1:51056) |
| 4:50:46 | 22185 | node | http__server__request | GET / (from ::1:51056)       |

```
./trace.py 'u:/tmp/libjvm.so:thread__start "%s [%d]", arg1, arg4' \
'u:/tmp/libjvm.so:thread__stop "%s [%d]", arg1, arg4'
```

| TIME    | PID   | COMM | FUNC          | -                          |
|---------|-------|------|---------------|----------------------------|
| 6:55:24 | 32157 | java | thread__start | Reference Handler [32157]  |
| 6:55:24 | 32158 | java | thread__start | Finalizer [32158]          |
| 6:55:24 | 32159 | java | thread__start | Signal Dispatcher [32159]  |
| 6:55:24 | 32160 | java | thread__start | C2 CompilerThread0 [32160] |
| 6:55:24 | 32161 | java | thread__start | C2 CompilerThread1 [32161] |
| 6:55:24 | 32162 | java | thread__start | C1 CompilerThread2 [32162] |
| 6:55:24 | 32163 | java | thread__start | Service Thread [32163]     |
| 6:55:28 | 32159 | java | thread__stop  | Signal Dispatcher [32159]  |

# Custom Tool Design



# BPF Program: Counting Allocations

```
include <linux/ptrace.h>

struct alloc_info_t {
    u64 count;
    u64 size;
};

PF_HASH(allocs, u32, struct alloc_info_t);

int handler(struct pt_regs *ctx, size_t size) {
    u32 pid = bpf_get_current_pid_tgid();
    struct alloc_info_t init = { 0 }, *info;

    info = allocs.lookup_or_init(&pid, &init);
    info->count += 1;
    info->size += size;

    return 0;
}
```

# BPF Driver

```
#!/usr/bin/env python
from bcc import BPF
from time import sleep

program = BPF(src_file="allocs.c")
program.attach_kprobe(event="__kmalloc", fn_name="handler")
allocs = program.get_table("allocs")

while True:
    sleep(5)
    print("\n%-8s %-8s %-10s" % ("PID", "COUNT", "SIZE"))
    for key, value in sorted(
        allocs.items(), key=lambda (k, v): k.value):
        print("%-8d %-8d %-8d" %
              (key.value, value.count, value.size))
```

# BPF Execution

`./allocs.py`

| ID   | COUNT | SIZE |
|------|-------|------|
| 8064 | 3     | 456  |
| 8157 | 10    | 76   |
| 8158 | 5     | 1116 |

| ID   | COUNT | SIZE  |
|------|-------|-------|
| 8001 | 113   | 1828  |
| 8064 | 8     | 1216  |
| 8110 | 38    | 683   |
| 8157 | 46    | 328   |
| 8158 | 5     | 1116  |
| 8159 | 41    | 12894 |

C

# Deployment

At development-time, BCC has a bunch of dependencies, which are not required for deployment

For Python tools, deploy Python + libbcc.so

For Lua tools, deploy only **bcc-lua**

- Statically links libbcc.a but allows plugging libbcc.so

# Summary

Tracing can identify bugs and performance issues that no debugger or profiler can catch

Tools make low-overhead, dynamic, production tracing possible

BPF is the next-generation backend for Linux tracing tools

# Thank You!

Sasha Goldshtein

@goldshn