```
@times:
[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)            25 |                                                    |
[512, 1k)            578 |@@@@@@@@@@@@@@@                                     |
[1k, 2k)             826 |@@@@@@@@@@@@@@@@@@@@@                               |
[2k, 4k)            1969 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
[4k, 8k)            1335 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@                 |
[8k, 16k)            512 |@@@@@@@@@@@@@                                       |
[16k, 32k)           240 |@@@@@@                                              |
[32k, 64k)            46 |@                                                   |
[64k, 128k)            4 |                                                    |
[128k, 256k)           1 |                                                    |
[256k, 512k)           4 |                                                    |
[512k, 1M)             0 |                                                    |
[1M, 2M)               0 |                                                    |
[2M, 4M)               0 |                                                    |
[4M, 8M)               1 |                                                    |
```

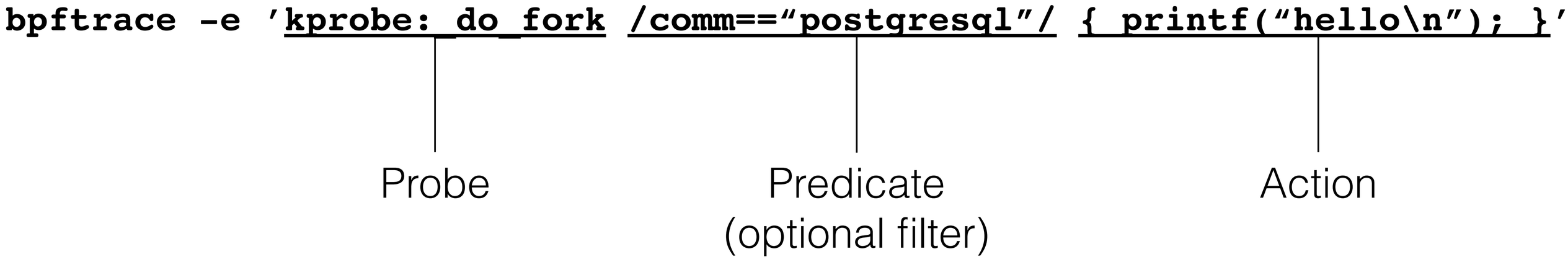
bpftrace

Yellowbrick

Alastair Robertson

bpftrace overview

Syntax


bpftrace -e 'kprobe:_do_fork /comm==“postgresql”/ { printf(“hello\n”); }'
Probe
Predicate
(optional filter)
Action

Builtin variables & functions

- pid - Process ID (kernel tgid)
- tid - Thread ID (kernel pid)
- cgroup - Cgroup ID of the current process
- uid - User ID
- gid - Group ID
- nsecs - Nanosecond timestamp
- cpu - Processor ID
- comm - Process name
- stack - Kernel stack trace
- ustack - User stack trace
- arg0, arg1, ... etc. - Arguments to the function being traced
- retval - Return value from function being traced
- func - Name of the function currently being traced
- probe - Full name of the probe
- curtask - Current task_struct as a u64
- rand - Random number of type u32

- hist(int n) - Produce a log2 histogram of values of n
- lhist(int n, int min, int max, int step) - Produce a linear histogram of values of n
- count() - Count the number of times this function is called
- sum(int n) - Sum this value
- min(int n) - Record the minimum value seen
- max(int n) - Record the maximum value seen
- avg(int n) - Average this value
- stats(int n) - Return the count, average, and total for this value
- delete(@x) - Delete the map element passed in as an argument
- str(char *s) - Returns the string pointed to by s
- printf(char *fmt, ...) - Print formatted to stdout
- print(@x[, int top [, int div]]) - Print a map, with optional top entry count and divisor
- clear(@x) - Delete all key/values from a map
- sym(void *p) - Resolve kernel address
- usym(void *p) - Resolve user space address
- kaddr(char *name) - Resolve kernel symbol name
- uaddr(char *name) - Resolve user space symbol name
- reg(char *name) - Returns the value stored in the named register
- join(char *arr[]) - Prints the string array
- time(char *fmt) - Print the current time
- system(char *fmt) - Execute shell command
- exit() - Quit bpftrace

Probe Types

Kprobes (4.1)	`kprobe:vfs_read`
Uprobes (4.3)	`uprobe:/bin/bash:readline`
USDT (4.3)	`usdt:/usr/bin/postgres:query_plan_start`
Tracepoints (4.7)	`tracepoint:sched:sched_switch`
Timers (4.9)	`profile:hz:99`
Software events (4.9)	`software:cpu-migrations:`
Hardware events (4.9)	`hardware:cache-misses:`
Watchpoints (4.9)	`watchpoint::0x10000000:8:rw`

```
tracepoint:syscalls:sys_enter_*
{
   @mycount[probe] = count();
}
```

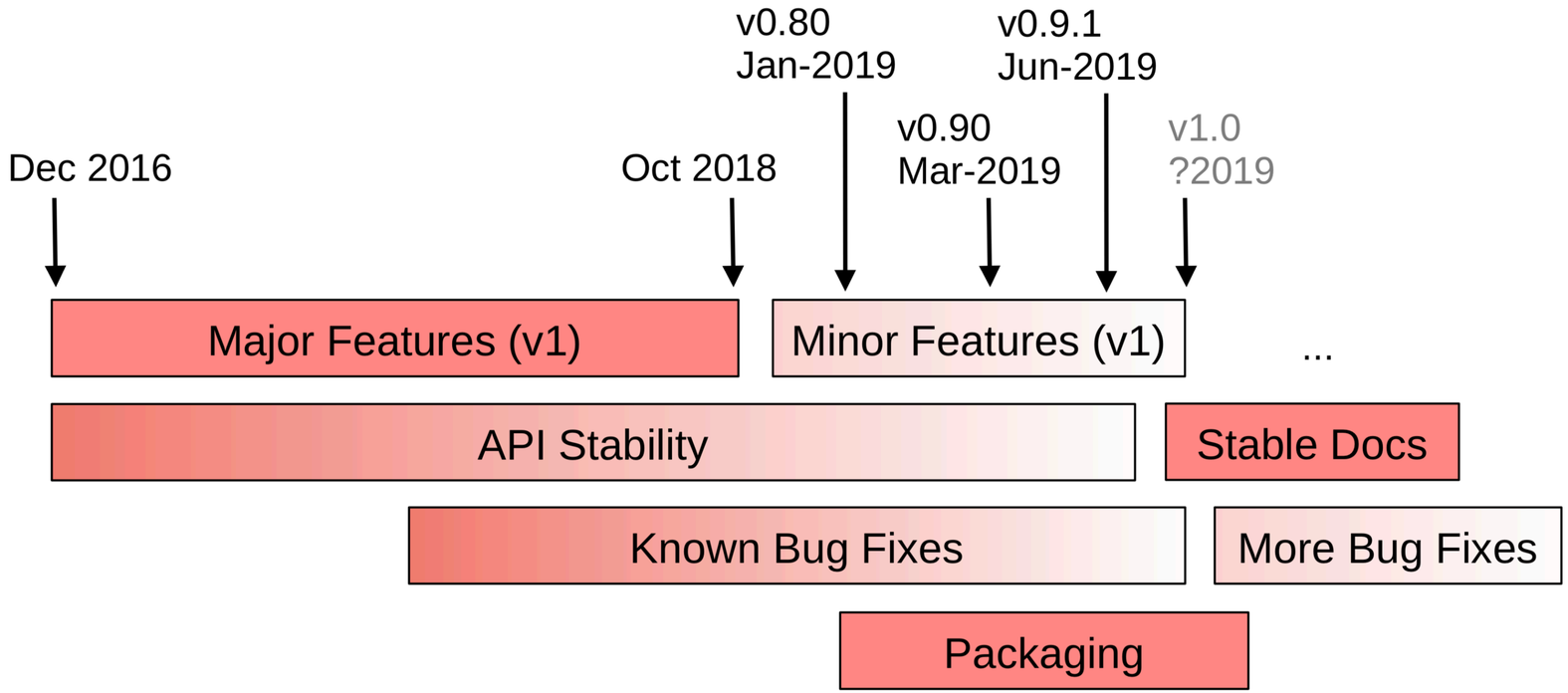
```
# bpftrace syscall_count.bt
Attaching 320 probes...
^C

@mycount[tracepoint:syscalls:sys_enter_geteuid]: 1
@mycount[tracepoint:syscalls:sys_enter_getsockopt]: 1
…
@mycount[tracepoint:syscalls:sys_enter_ioctl]: 572
@mycount[tracepoint:syscalls:sys_enter_dup2]: 580
@mycount[tracepoint:syscalls:sys_enter_read]: 881
@mycount[tracepoint:syscalls:sys_enter_close]: 988
@mycount[tracepoint:syscalls:sys_enter_writev]: 1012
@mycount[tracepoint:syscalls:sys_enter_poll]: 1323
@mycount[tracepoint:syscalls:sys_enter_recvmsg]: 3164
```

Recent work

A year of bpftrace

Since Linux Plumbers 2018:

- 298 pull request from 48 individuals
- 20292 lines of code changed
- Packaged on distros:
  - Ubuntu
  - Fedora
  - Gentoo
  - Debian
  - openSUSE
  - NixOS
  - …



Compiler error context

```
# bpftrace -e 'i:s:1 { 0 < "abc" }’

stdin:1:9-18: ERROR: Type mismatch for '<': comparing
                     ‘integer' with 'string'
i:s:1 { 0 < "abc" }
        ^~~~~~~~~
```

Tests

```
NAME printf
RUN bpftrace -v -e 'i:ms:1 { printf("hi!\n"); exit();}'
EXPECT hi!
TIMEOUT 5
```

Now have 173 end to end tests and growing

IP addresses

```
#include <net/sock.h>

kprobe:tcp_connect
{
  $sk = ((struct sock *) arg0);
  printf(“%s\n”, ntop(AF_INET, $sk->__sk_common.skc_daddr));
}
```

New tools:

- tcpaccept.bt
- tcpconnect.bt
- tcpdrop.bt
- tcpretrans.bt

Watchpoints

```
watchpoint::0x100000000:8:rw
{
  printf(“hit!”\n”);
}
```

Programmatic bpftrace

```
# bpftrace -f json -e 'tracepoint:syscalls:sys_enter_getuid { printf("getuid:
%d\n", uid); @count = count(); }'
{"type": "attached_probes", "data": {"probes": 1}}
{"type": "printf", "data": "getuid: 1000\n"}
{"type": "printf", "data": "getuid: 1000\n"}
{"type": "printf", "data": "getuid: 1000\n"}
{"type": "printf", "data": "getuid: 1000\n"}
^C

{"type": "map", "data": {"@count": 4}}
```

More

- Run and trace another program
  - `bpftrace -c /usr/bin/program_to_trace myscript.bt`
  - `-o` Redirect output to file

- List available probes/tracepoint arguments
  - `bpftrace -l kprobe:*fork*`
  - `bpftrace -lv tracepoint:syscalls:sys_enter_read`

- Go calling conventions (stack arguments)
  - `sarg0, sarg1, etc.`

- Multiple tracepoint args
  - `tracepoint:syscalls:sys_enter_open,tracepoint:syscalls:sys_enter_openat { @[str(args->filename)] = count(); }`

Coming Improvements

Bounded loops

Asynchronous builtin functions **`clear`**, **`zero`** and **`print`** can be confusing.

BPF loops will allow them to be synchronous.

```
# bpftrace -e 'BEGIN { @x = 5; zero(@x); printf(“x value: %d\n”, @x); }'
Attaching 1 probe...
x value: 5
^C

@x: 0
```

BPF Type Format (BTF)

Information about functions and structs without kernel headers

```
# bpftrace -e 'kprobe:do_dentry_open { printf("%s %d - %s %d - %s %d\n", \
                    curtask->comm, curtask->pid, \
                    curtask->parent->comm, curtask->parent->pid, \
                    curtask->parent->parent->comm, curtask->parent->parent->pid); }'
Attaching 1 probe...
chronyd 1102 - systemd 1 - swapper/0 0
bash 14311 - bash 14207 - su 14203
bash 14311 - bash 14207 - su 14203
ls 14311 - bash 14207 - su 14203
ls 14311 - bash 14207 - su 14203
systemd-journal 861 - systemd 1 - swapper/0 0
NetworkManager 1098 - systemd 1 - swapper/0 0
sssd 1099 - systemd 1 - swapper/0 0
NetworkManager 1098 - systemd 1 - swapper/0 0
NetworkManager 14312 - NetworkManager 1098 - systemd 1
```

More

- New CPU architectures
- New builtins
- New LLVM versions
- Custom code generator
- …?

Thanks

- bpftrace
  - Netflix: Brendan Gregg, Matheus Marchini
  - Sthima: Willian Gaspar
  - Facebook: Jon Haslam, Dan Xu
  - Red Hat: Augusto Mecking Caringi
  - Shopify: Dale Hamel
- eBPF & BCC
  - Facebook: Alexei Starovoitov, Teng Qin, Yonghong Song, Martin Lau, Mark Drayton …
  - Netflix: Brendan Gregg
  - VMware: Brenden Blanco
  - Daniel Borkmann, David S. Miller, Sasha Goldsthein, Paul Chaignon, …
- Some content in these slides from Brendan Gregg, used with permission