```
@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

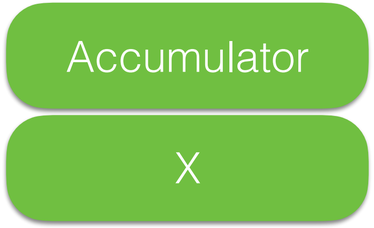
Alastair Robertson
G-Research

# Berkeley Packet Filter

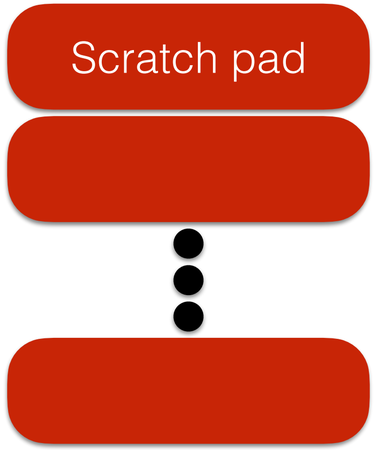
```
# tcpdump -d port 80
(000) ldh      [12]
(001) jeq      #0x86dd          jt 2 jf 10
(002) ldb      [20]
(003) jeq      #0x84            jt 6 jf 4
(004) jeq      #0x6             jt 6 jf 5
(005) jeq      #0x11            jt 6 jf 23
(006) ldh      [54]
(007) jeq      #0x50            jt 22  jf 8
(008) ldh      [56]
(009) jeq      #0x50            jt 22  jf 23
(010) jeq      #0x800           jt 11  jf 23
...
```

# BPF

32-bit registers

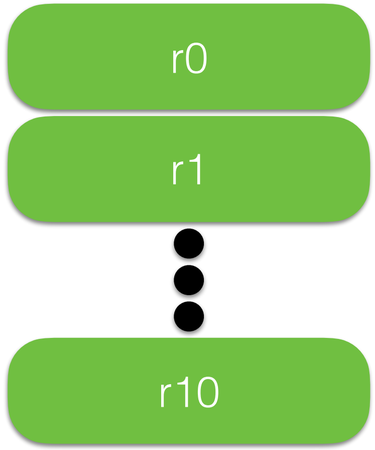


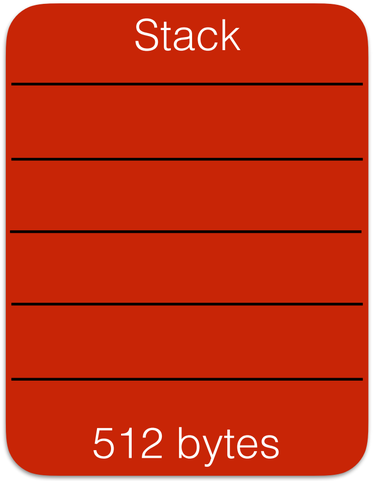

16 x 32-bit memory



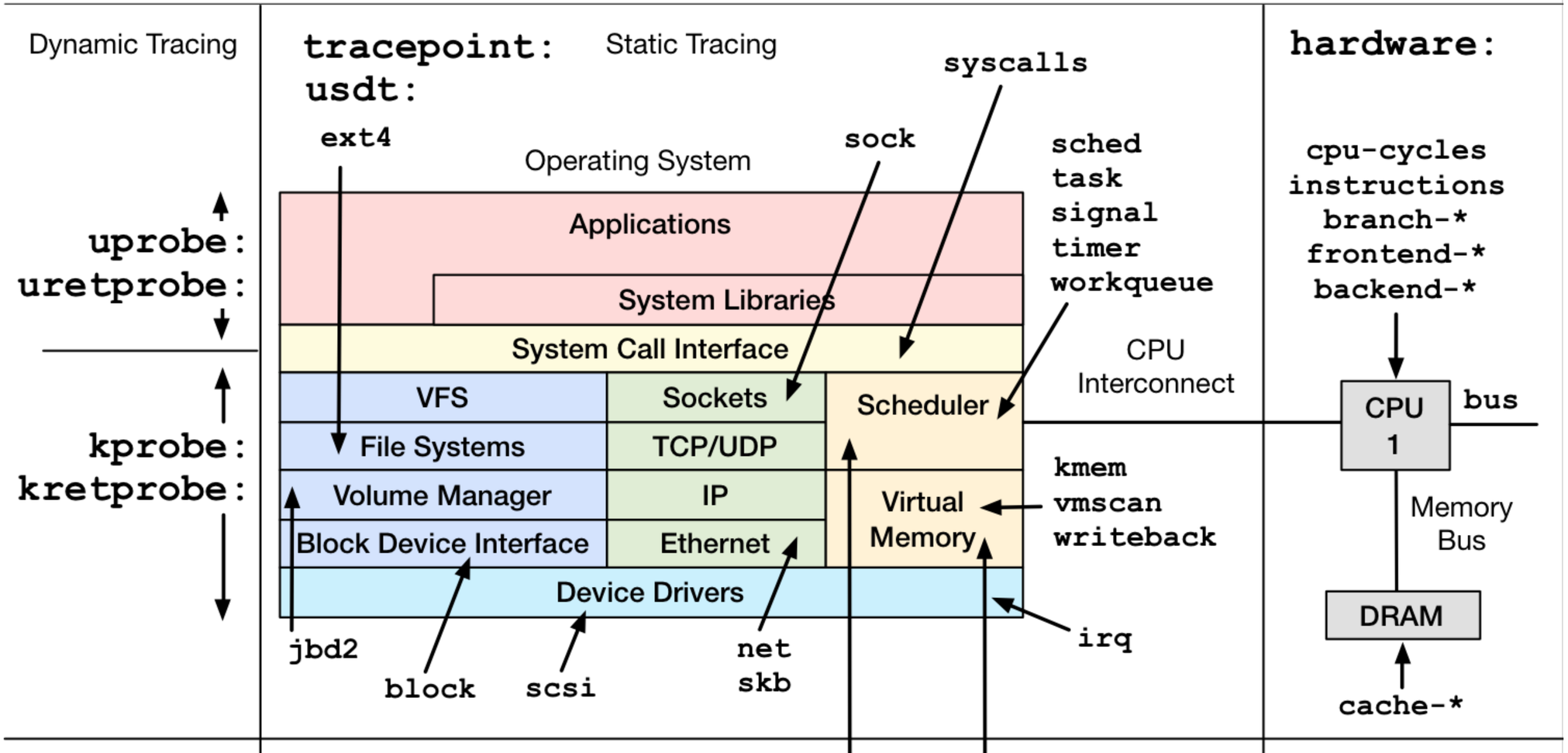

# eBPF

64-bit registers




# bpftrace Probe Types




**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

# Linux Requirements

| | |
|---|---|
| Kprobes (4.1) | `kprobe:vfs_read { ... }` |
| Uprobes (4.3) | `uprobe:/bin/bash:readline { ... }` |
| USDT (4.3) | |
| Stack traces, per-cpu maps (4.6) | |
| Tracepoints (4.7) | `tracepoint:sched:sched_switch { ... }` |
| Timers (4.9) | `profile:hz:99 { ... }` |
| Software events (4.9) | `software:faults: { ... }` |
| Hardware events (4.9) | `hardware:cache-references: { ... }` |

```
tracepoint:syscalls:sys_enter_read
{
  @mymap = count();
}
```

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

```
profile:hz:99
{
  @[stack] = count();
}
```

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

kprobe:blk_account_io_completion /@start[arg0]/
{
  $diff = (nsecs - @start[arg0]) / 1000;
  @usecs = hist($diff);
  delete(@start[arg0]);
}
```

```
struct Conference
{
  int rating;
  char name[32];
}

uprobe:./a.out:get_rating
{
  $foo = (Conference*)arg0;
  printf("%s gets %d out of 10\n",
         $foo->name, $foo->rating);
}
```

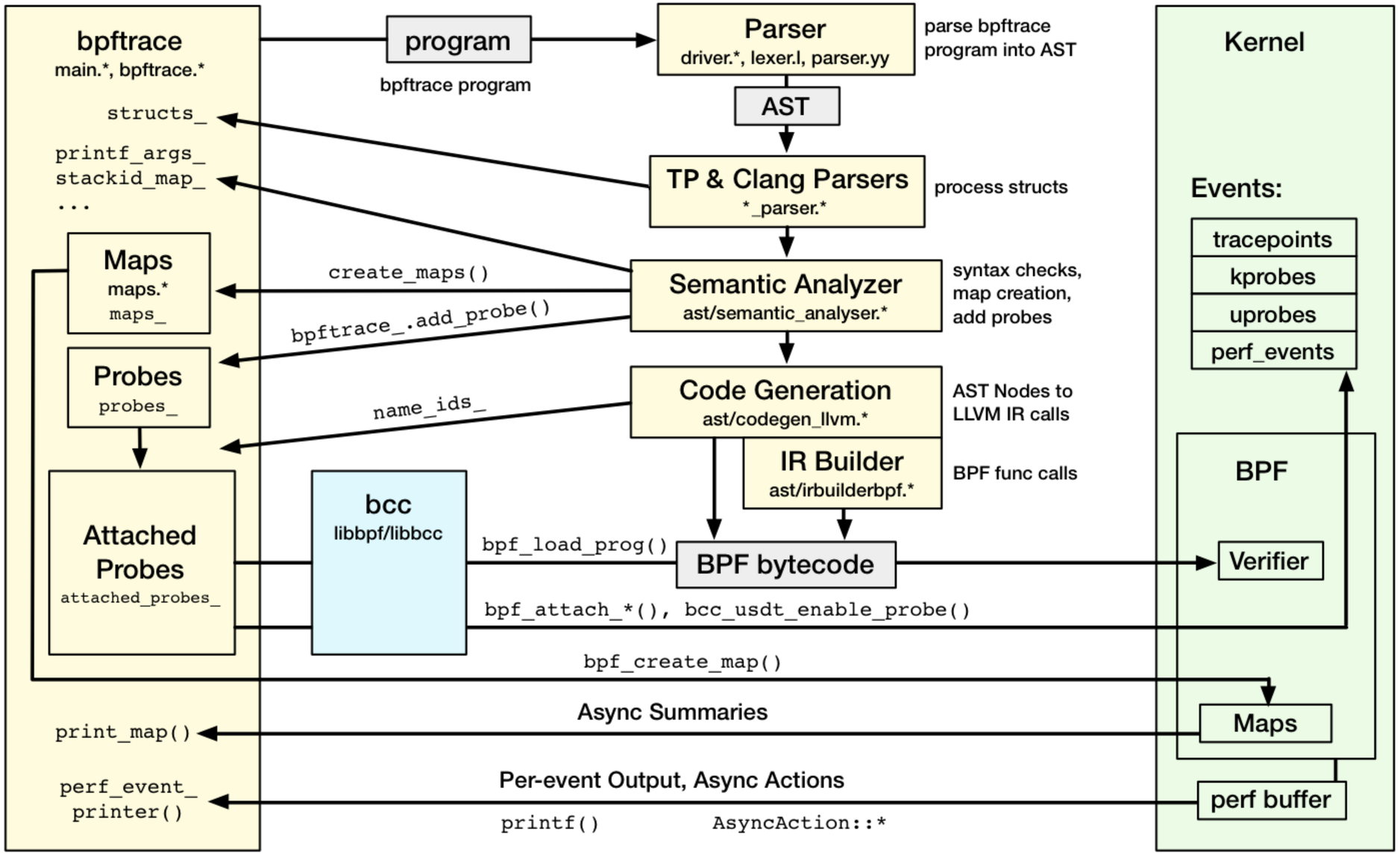
```
tracepoint:syscalls:sys_enter_open
{
  printf("%s %s\n", comm, str(args->filename));
}
```

# Builtins

- 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

# bpftrace Internals




https://github.com/iovisor/bpftrace 2018

# BCC

```
from __future__ import print_function
from bcc import BPF
from time import strftime
import ctypes as ct

# load BPF program
bpf_text = """
#include <uapi/linux/ptrace.h>
struct str_t {
    u64 pid;
    char str[80];
};
BPF_PERF_OUTPUT(events);
int printret(struct pt_regs *ctx) {
    struct str_t data  = {};
    u32 pid;
    if (!PT_REGS_RC(ctx))
        return 0;
    pid = bpf_get_current_pid_tgid();
    data.pid = pid;
    bpf_probe_read(&data.str, sizeof(data.str), (void *)PT_REGS_RC(ctx));
    events.perf_submit(ctx,&data,sizeof(data));
    return 0;
};
"""
STR_DATA = 80

class Data(ct.Structure):
    _fields_ = [
        ("pid", ct.c_ulonglong),
        ("str", ct.c_char * STR_DATA)
    ]

b = BPF(text=bpf_text)
b.attach_uretprobe(name="/bin/bash", sym="readline", fn_name="printret")

# header
print("%-9s %-6s %s" % ("TIME", "PID", "COMMAND"))

def print_event(cpu, data, size):
    event = ct.cast(data, ct.POINTER(Data)).contents
    print("%-9s %-6d %s" % (strftime("%H:%M:%S"), event.pid,
                            event.str.decode('utf-8', 'replace')))

b["events"].open_perf_buffer(print_event)
while 1:
b.perf_buffer_poll()
```

# bpftrace

```
BEGIN
{
    printf("Tracing bash commands... Hit Ctrl-C to end.\n");
    printf("%-9s %-6s %s\n", "TIME", "PID", "COMMAND");
}

uretprobe:/bin/bash:readline
{
    time("%H:%M:%S  ");
    printf("%-6d %s\n", pid, str(retval));
}
```

# Future Work

65 Open ✓ 50 Closed Author Labels Projects Milestones Assignee Sort

- Building With Multiple Versions Of Clang Installed
  #211 opened 2 hours ago by atti1a
- `sym()` resolves symbols in aggregations, but not in printf bug 1
  #209 opened a day ago by Birch-san
- cannot run opensnoop.bt 6
  #207 opened 3 days ago by yujinqiu
- Error loading program, with no details (kern version) bug 6
  #206 opened 4 days ago by brendangregg
- bpftrace crashes during finalization with specific one-liner (probably related to lhist function)
  #201 opened 5 days ago by caringi
- str() to accept optional length
  #199 opened 5 days ago by brendangregg
- strings in predicates failures bug
  #198 opened 5 days ago by brendangregg
- bash command completions 1
  #197 opened 6 days ago by brendangregg
- Use after free from BPFtrace::get_arg_values bug 2
  #194 opened 7 days ago by ajor
- if statement semicolon 1
  #192 opened 7 days ago by brendangregg
- Tidy USAGE message good first issue
  #191 opened 7 days ago by brendangregg
- Fix codegen tests

https://github.com/iovisor/bpftrace