# bpftrace

high-level tracing language powered by BPF

**Mary Marchini** 



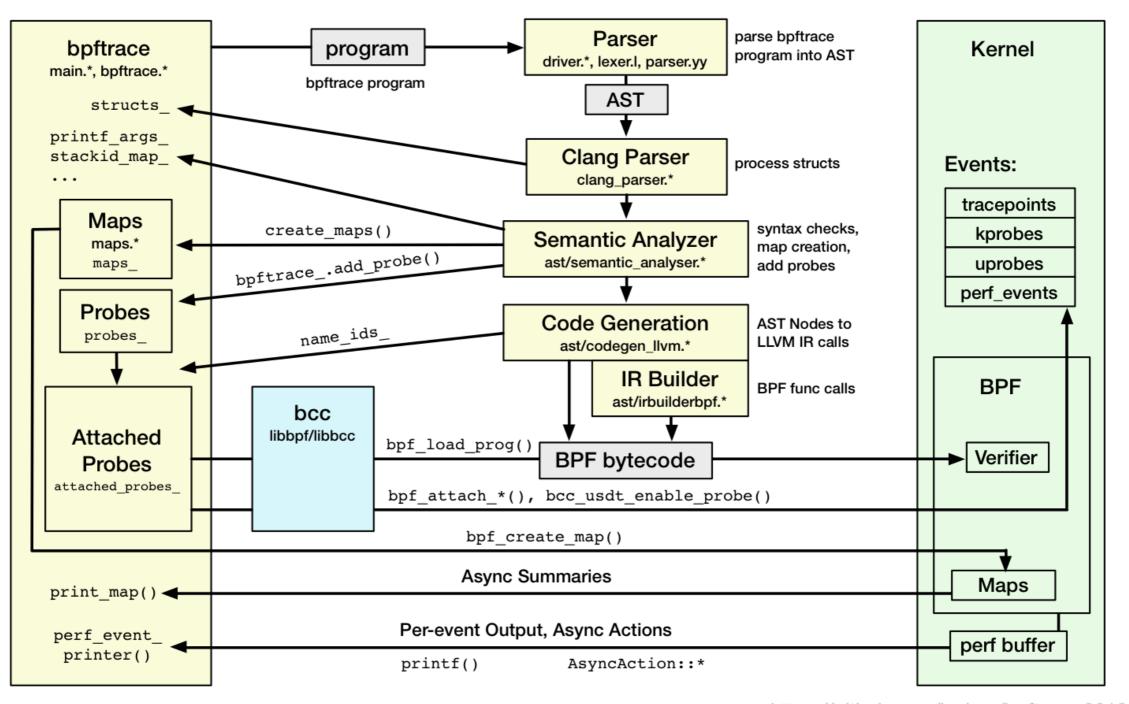


### Demo

#### bpftrace Probe Types hardware: **Dynamic Tracing** tracepoint: Static Tracing syscalls ext4 sock sched cpu-cycles **Operating System** task instructions signal branch-\* uprobe: **Applications** timer frontend-\* uretprobe: workqueue backend-\* System Libraries usdt: **CPU** System Call Interface Interconnect **VFS** Sockets bus Scheduler / **CPU** kprobe: File Systems TCP/UDP kmem kretprobe: IP Volume Manager Virtual 4 vmscan Memory writeback Memory A Block Device Interface Ethernet <sub>4</sub> Bus **Device Drivers** DRAM irq jbd2 net skb block scsi cache-\* profile: BEGIN software: page-faults cpu-clock interval: minor-faults cs migrations **END** major-faults **Timed Events** Special Events https://github.com/iovisor/bpftrace 2018

https://github.com/iovisor/bpftrace

### bpftrace Internals



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

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

## **Current Challenges**

- Keeping up with latest LLVM (API changes)
- Stack Walking with ORC
- uprobe-compatibility with other calling conventions
  - Go, Rust, Node.js, etc.
- New BPF helpers
  - BPF\_FUNC\_kill(int)
  - BPF\_FUNC\_path(struct file \*file)
  - BPF\_FUNC\_get\_current\_ppid()