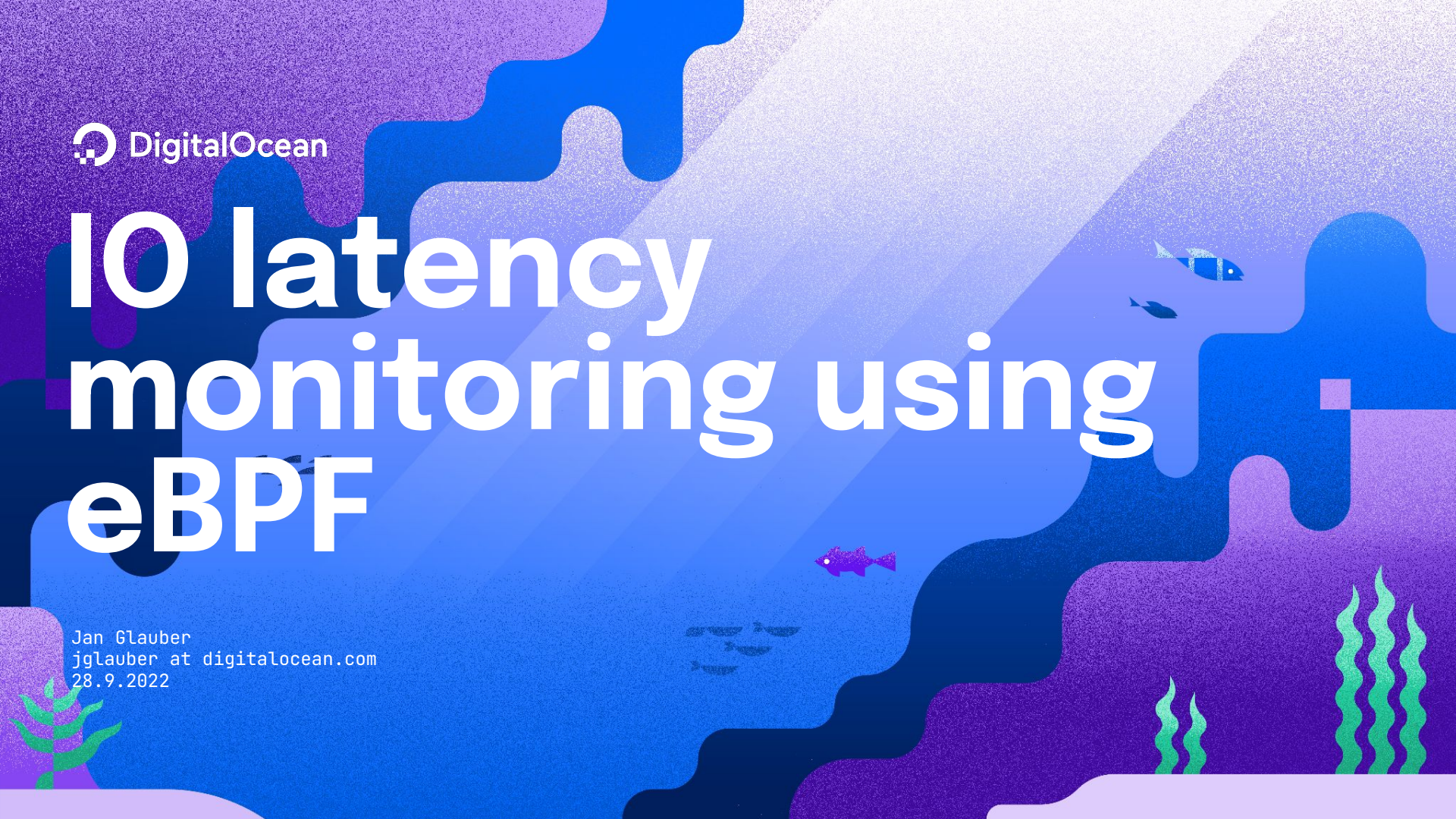

DigitalOcean
IO latency monitoring using eBPF
Jan Glauber
jglauber at digitalocean.com
28.9.2022

Problem statement

- Cloud provider, large fleet of servers running many VMs with qemu & kvm
- continuous monitoring for SLOs and SLAs
- per VM metrics for CPU, memory, IO, network, …
- Prometheus & Grafana
- local storage metrics wanted

Only per-disk latency data available, no per-VM data, no latency by blocksize

…but we want to track every IO request…

Enter eBPF

- Complex LVM/dm/RAID disk setup, which layer to trace?
- Kprobes or tracepoints or raw tracepoints?
- Userspace tooling & golang which library to use?
- Performance overhead? In-kernel aggregation or 1:1 monitoring?
- eBPF vs. proc stats?

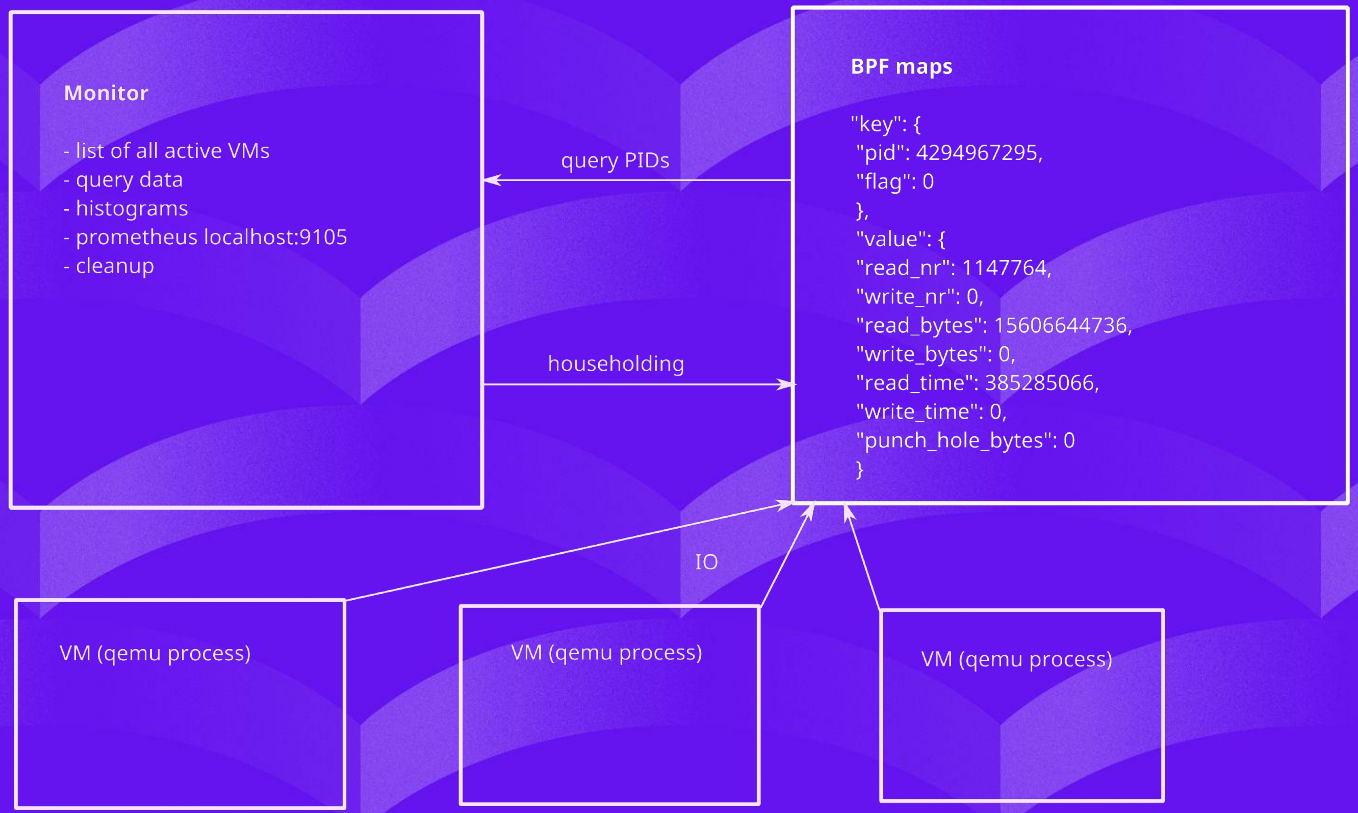
eBPF hooks

```
SEC("raw_tracepoint/block_bio_queue")
int BPF_PROG(trace_bio_start, struct request_queue *q, struct bio *bio)
{
        …
        if (!disk_traced(bio))
                return 0;
        bpf_map_update_elem(&bio_start, &bio, &ts, 0);
        …
}

SEC("raw_tracepoint/block_bio_complete")
int BPF_PROG(trace_bio_done, struct request_queue *q, struct bio *bio)
{
        …
        tsp = bpf_map_lookup_elem(&bio_start, &bio);
        if (!tsp) {
                return 0;   // missed issue
        }

        u64 sector = BPF_CORE_READ(bio, bi_iter.bi_sector);
        // ignore if sector is 0
        if (!sector) {
                goto cleanup;
        }
        …
}
```

eBPF monitoring



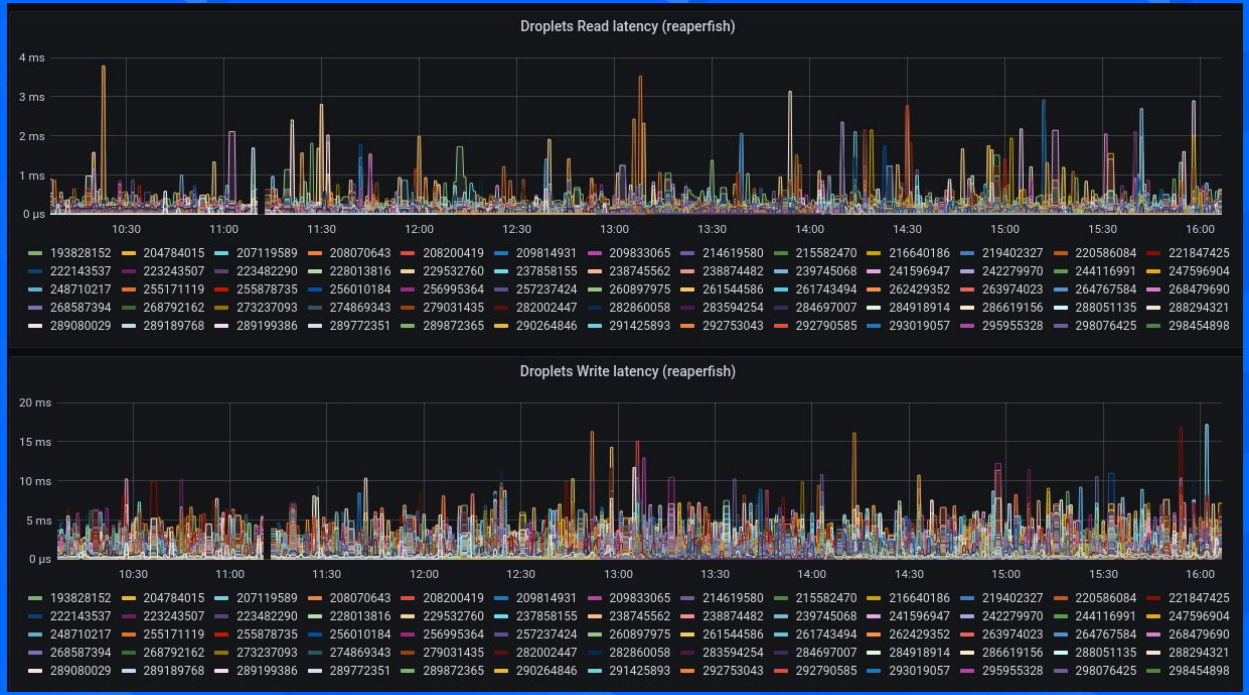
Results

- bio layer tracing
- raw tracepoints
- cilium ebpf library works great (mostly)
- Low performance overhead (bpf stats)
- Custom filtering possible, easy to extend
- Histograms, P50/90/99, etc.
- Faster than proc stats parsing

Issues & future work

- Get kernel versioning support
  - kernel version
  - Kconfig symbols
  - struct flavours
  - Tracepoint signature changes
- BPF map versioning (extend only)
- Map utilization observance
- Package with BPF programs
- Dynamic map sizing

Thank you!



Stand-alone CL monitoring tool:
https://github.com/el8/reaperfish