Grafana Labs

Lies, damned lies, and request times

Nikola Grcevski – Mario Macias

PromCon 2023 – Berlin, Germany

Lightning talk

Let’s build a simple web service

- A foo REST service that needs to respond **under 100ms**

Our SLO for response time

```
type FooServer struct {
    responseTime prometheus.Histogram
}

func (ps *FooServer) Foo(rw http.ResponseWriter, req *http.Request) {
    start := time.Now()
    fooHandler(rw, req)
    ps.responseTime.Observe(time.Now().Sub(start).Seconds() * 1000)
}
```

- Using Grafana’s K6 to add some load and measure it

Let’s visualize some prometheus metrics

Let's compare notes with what K6 saw

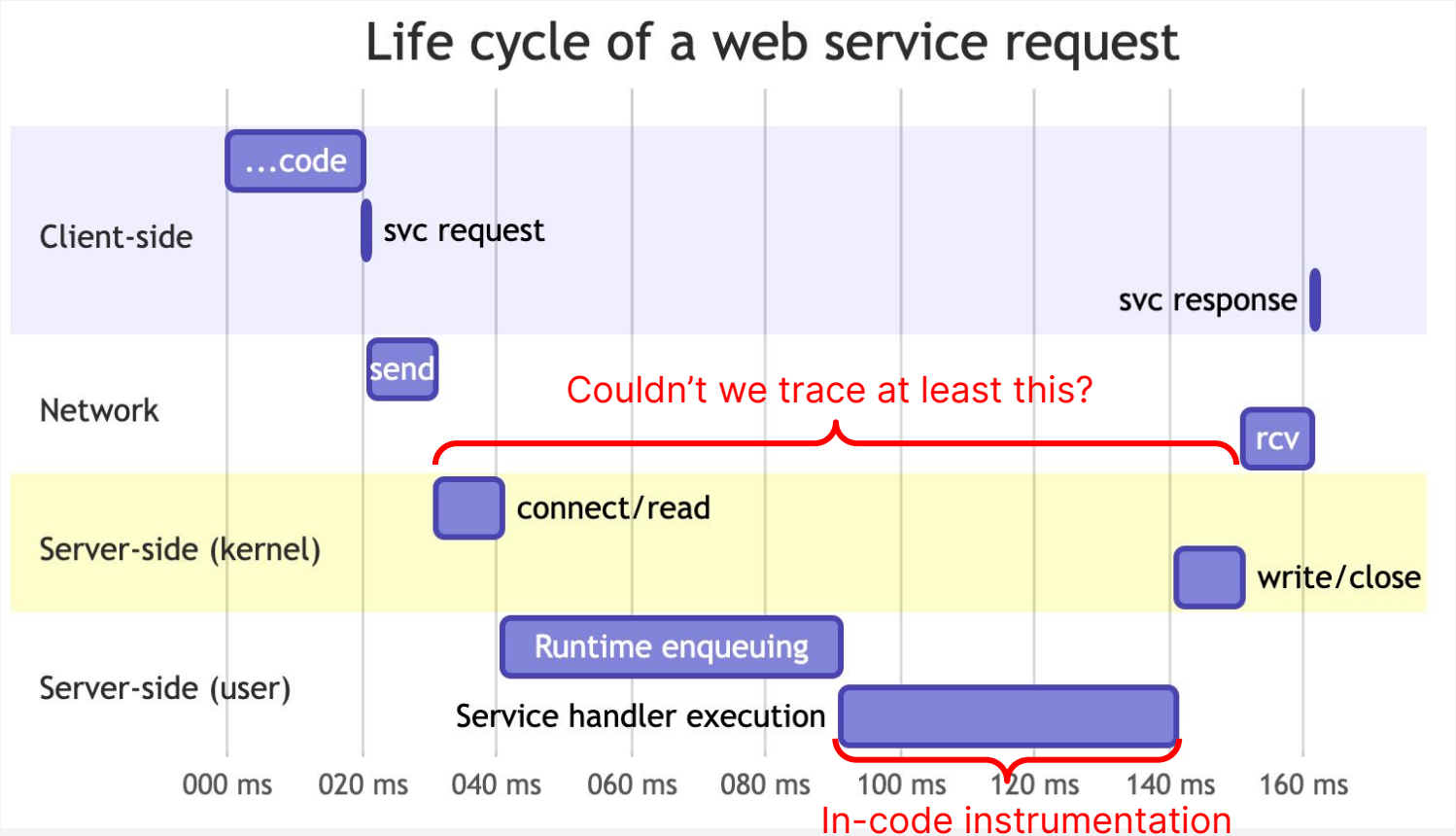
```
(...)
http_req_duration : avg=111.09ms  min=340µs   med=88.1ms
max=1.14s  p(90)=228.61ms p(95)=283.24ms
(...)
Http                    2188 503.622464
```

Prometheus measured AVG: ~20ms

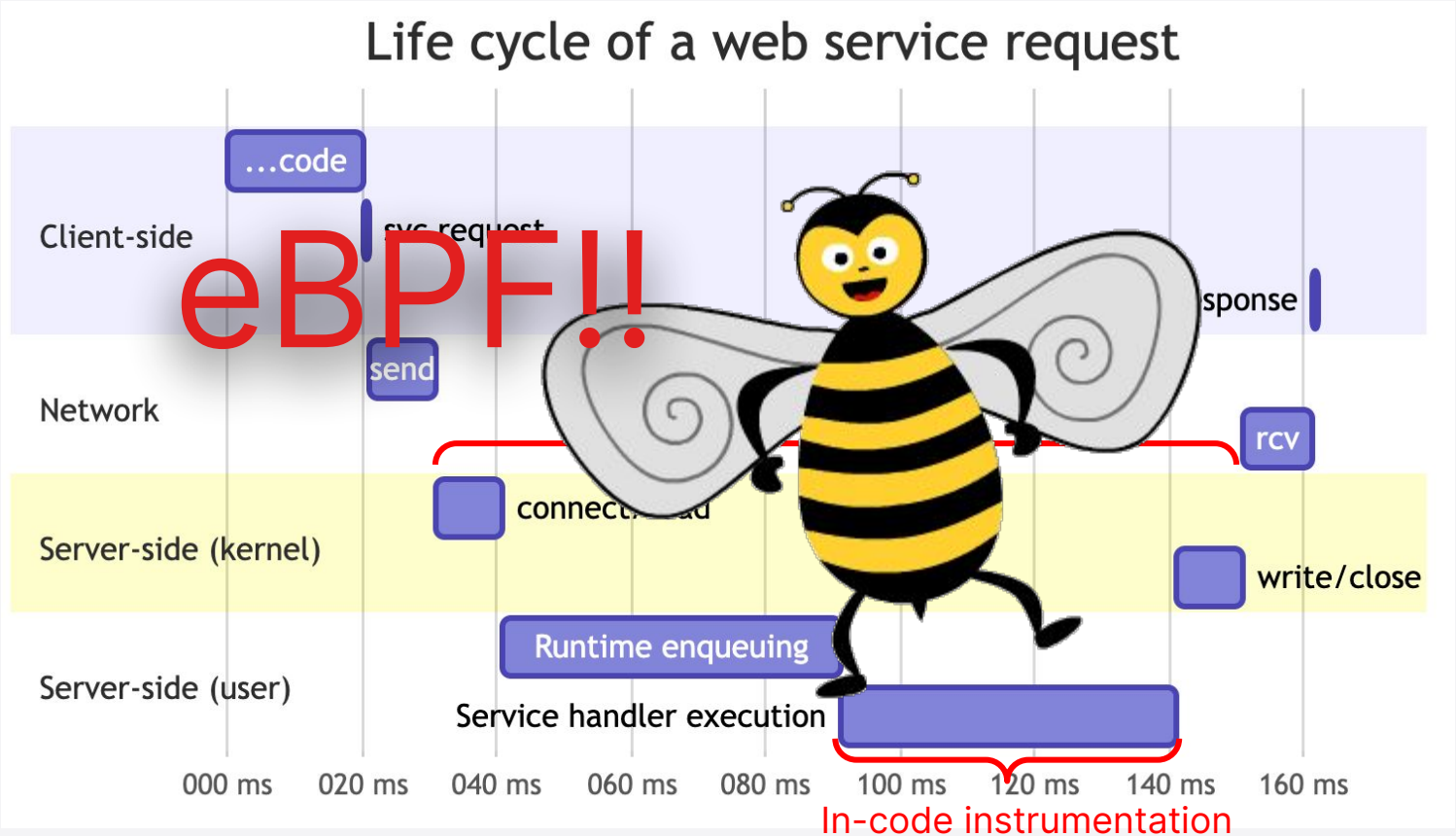
Prometheus measured P95: ~63ms

```
import http from 'k6/http';
import {sleep} from 'k6';

export const options = {
    vus: 80,
    duration: '600s',
    noConnectionReuse: true,
};

export default function () {
    http.get('http://localhost:8080/ping');
    sleep(0.01);
}
```

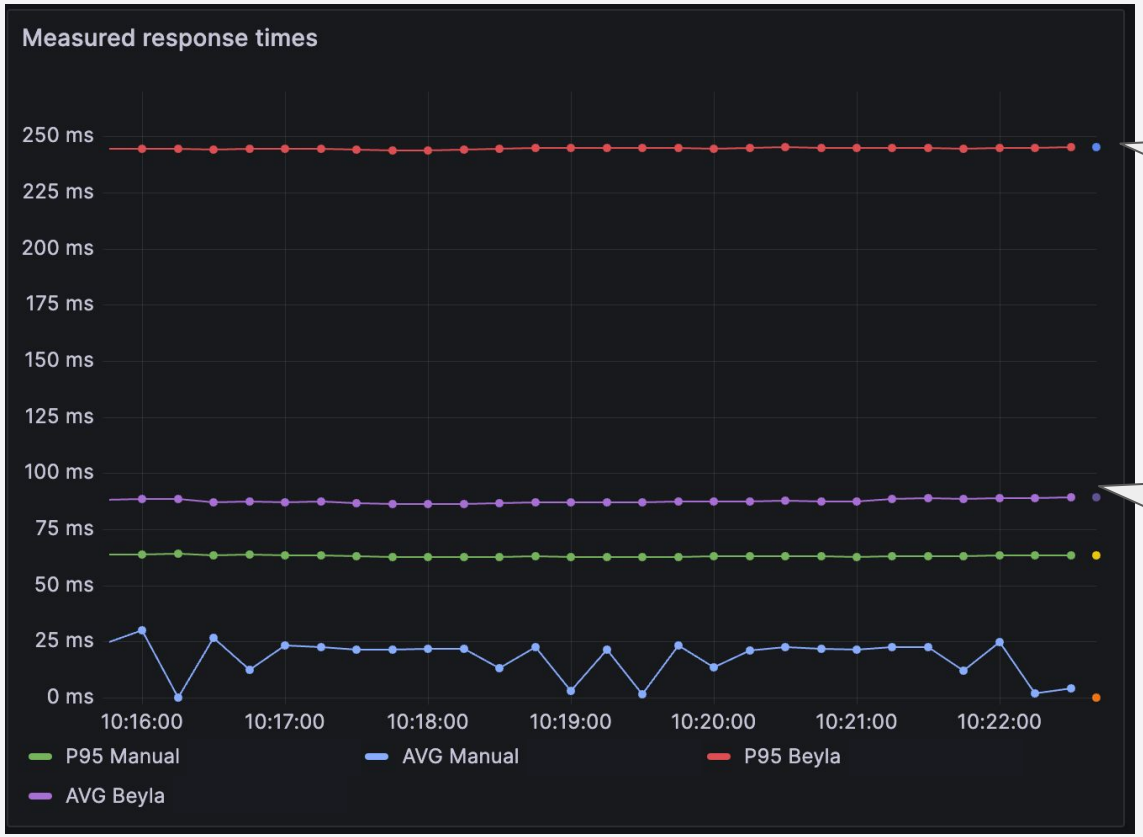
Is Prometheus Go library lying?



Is Prometheus Go library lying?



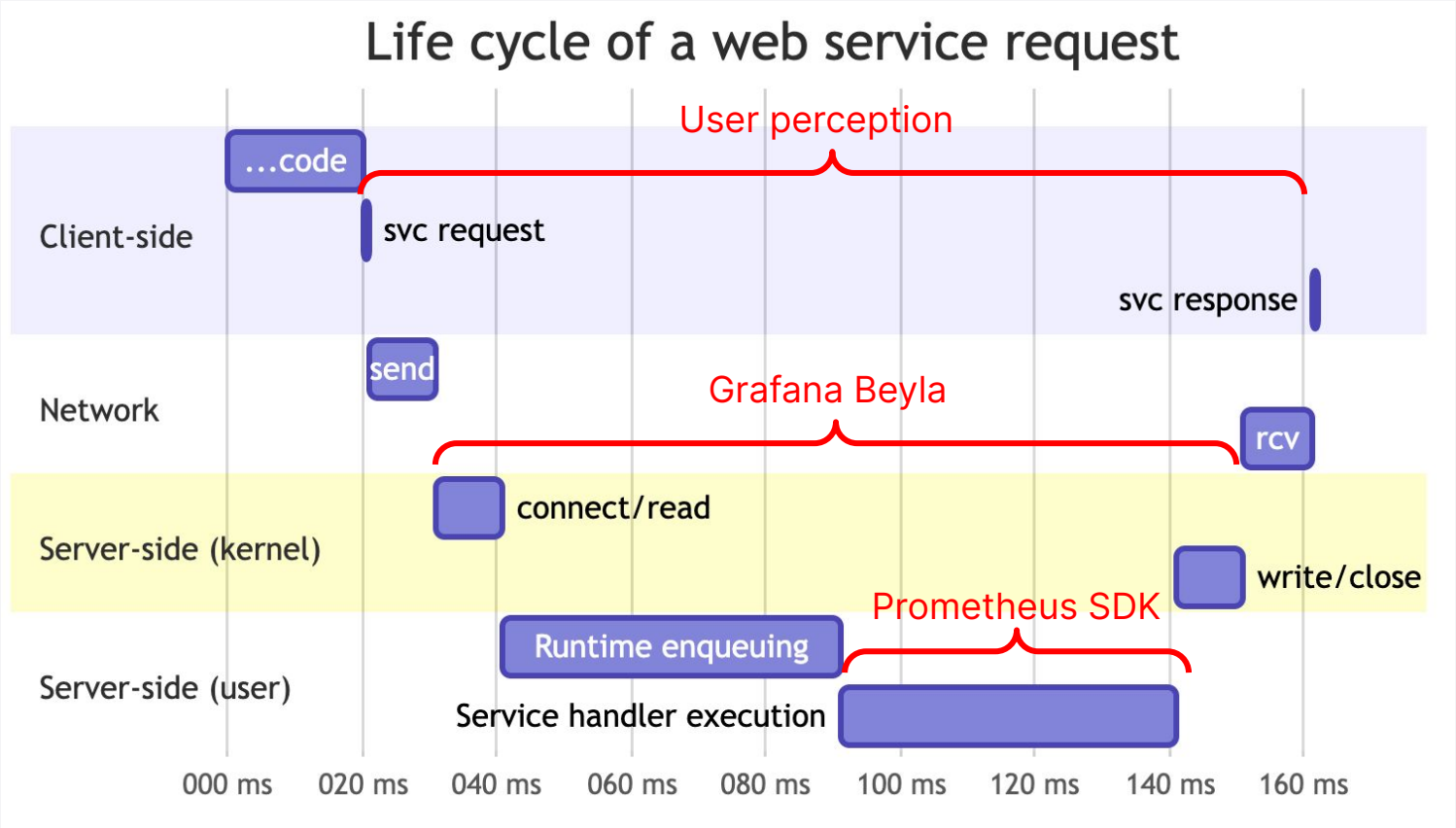
Grafana Beyla reported timings



K6: 283ms

K6: 111ms

Not yet perfect!



Grafana Labs

Thank you!

github.com/grafana/beyla