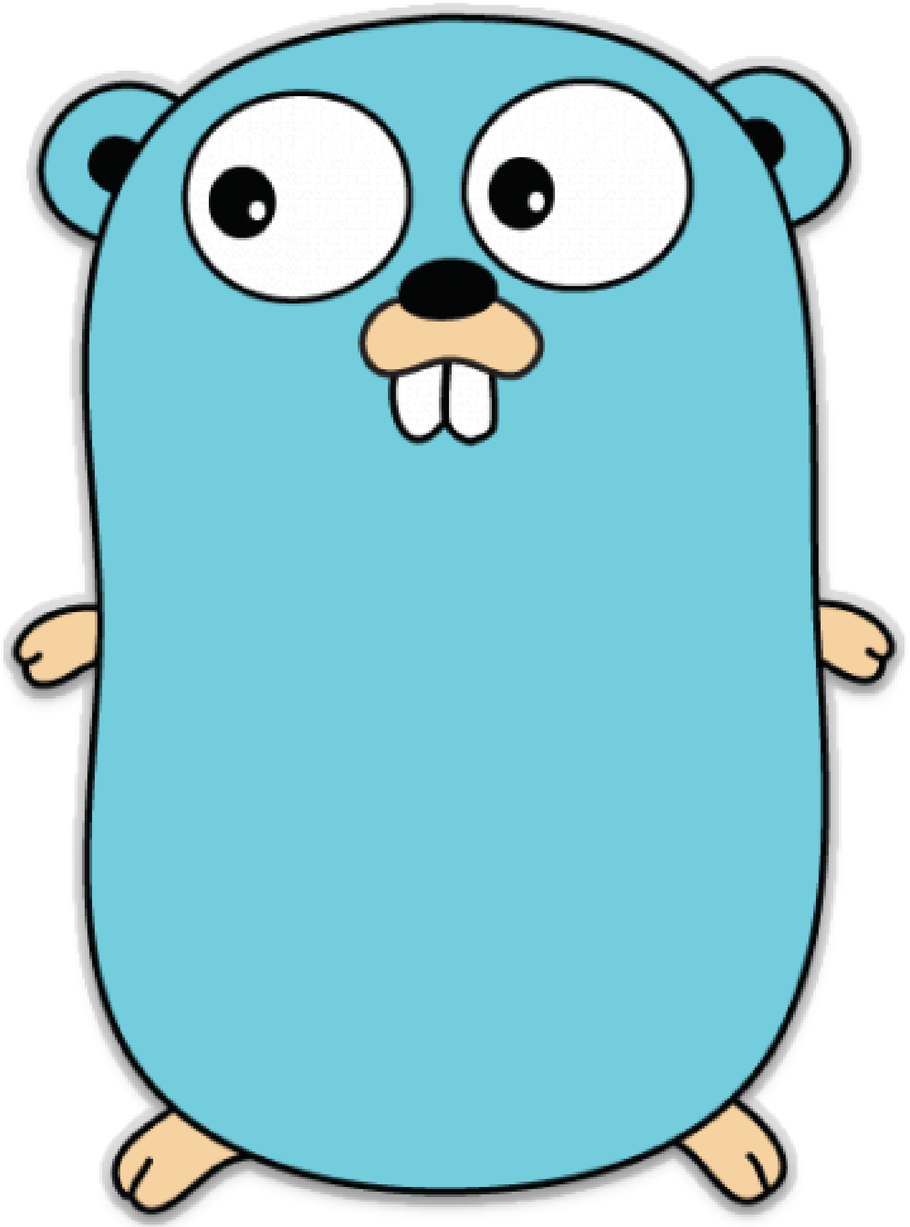


API LOAD TEST PRESENTATION

นำเสนอโดย

ศรราม เจริญมา

พสิษฐ์ คำเสนา

HTTP2

เทียบกันโดยใช้

- http
- fasthttp

HTTP2(fasthttp) 700VUs 10s			
ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs
1	3.33s	26.06%	2594
2	3.52s	31.38%	2399
3	3.41s	34.33%	2467
4	3.33s	32.77%	2450
5	3.36s	28.36%	2612
avg	3.39s	30.58%	2504.4

HTTP2(http) 700vus 10s			
ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs
1	3.85s	33.23%	2422
2	3.31s	26.58%	2595
3	3.57s	32.20%	2428
4	3.5s	34.61%	2470
5	3.57s	34.18%	2460
avg	3.56s	32.178%	2475

สรุป

fasthttp มีความเร็วมากกว่า http
แต่ fasthttp ถ้ามี VUs มีค่ามากขึ้นจะทำให้ server หยุดทำงาน และ ตัวเลขจากการทดลองไม่มีความต่างอย่างเป็นนัยะสำคัญ ดังนั้น จึงสรุปได้ว่า http มีประสิทธิภาพในการทำ API Load Test มากกว่า

HTTP2 VS HTTP1.1

เทียบกันโดยใช้

- http

HTTP2 700VUs 10s

ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs
1	3.26s	27.25%	2594
2	3.36s	25.59%	2594
3	3.49s	30.51%	2327
4	3.75s	31.58%	2390
5	3.41s	32.89%	2429
avg	3.454s	29.564%	2466.8

HTTP1.1 700VUs 10s

ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs
1	3.58s	25.41%	3376
2	3.58s	17.63%	3192
3	3.54s	20.21%	3270
4	3.52s	23.96%	3347
5	3.59s	24.52	3351
avg	3.562s	22.166%	3307.2

สรุป

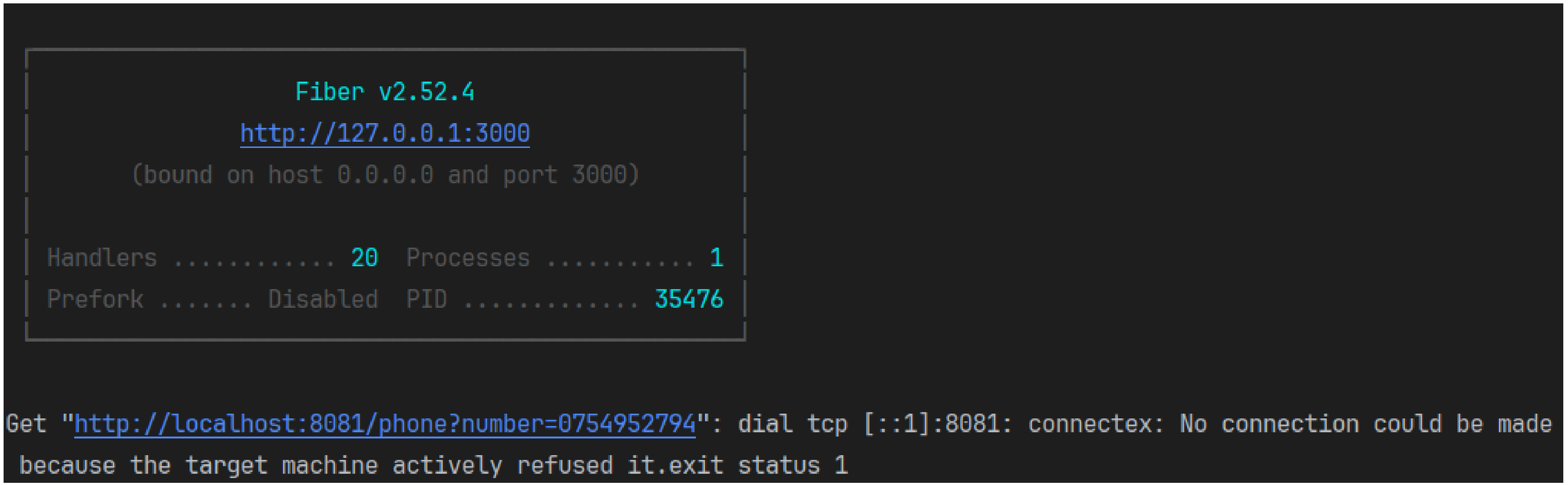
http1.1 มี http_req_failed น้อยกว่า http2 และมี http_reqs มากกว่า http2 ดังนั้นจึงสรุปได้ว่า http1.1 มีประสิทธิภาพมากกว่าในการทดลองครั้งนี้

ปัญหาคอขวด

ถ้าค่า VUs ในการทดสอบมีค่าสูงขึ้นจะทำให้ server ปิดตัวลง เนื่องจากไม่สามารถ call API ไปยัง mock-lookup ได้เพราะถูกปฏิเสธการเชื่อมต่อ

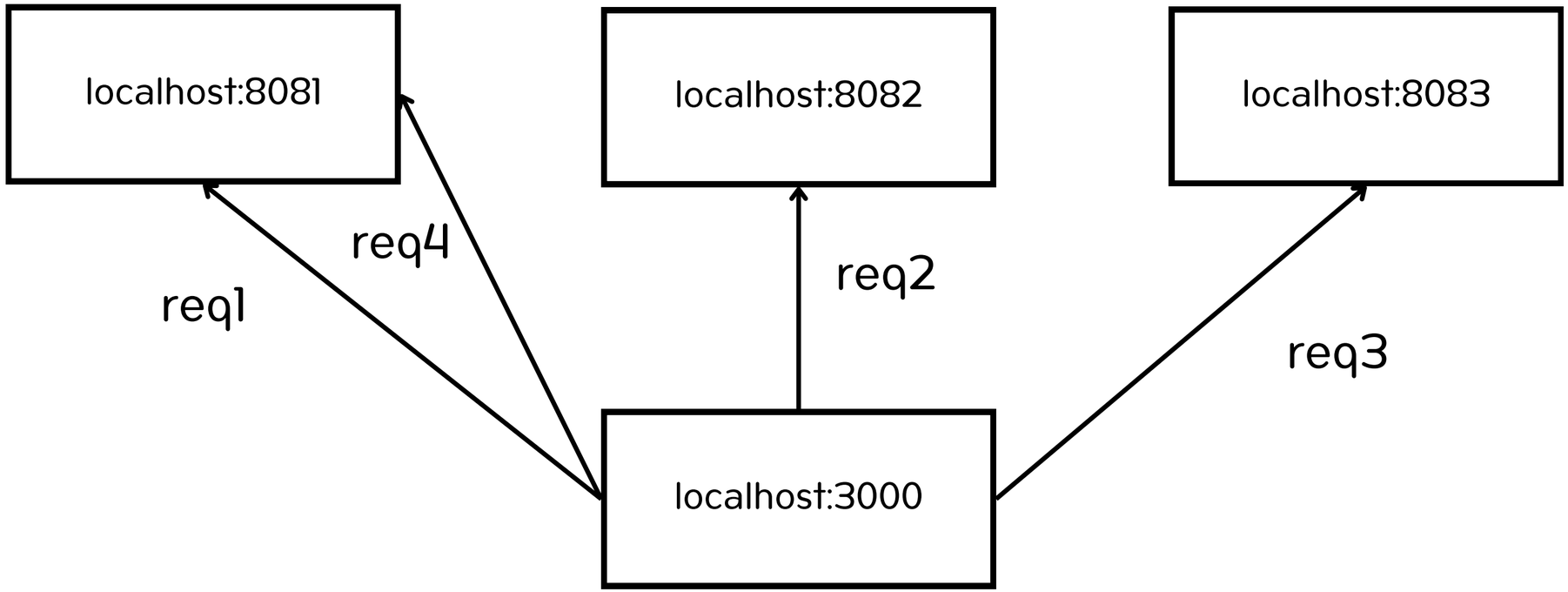
```
import http from "k6/http"
import {sleep} from "k6"

export const options : {duration: string, vus: number}  = {  no usages   night-sornram +1 *
    vus: 1000,
    duration: '15s',
}



export default function () : void  {  no usages   night-sornram +1
    const res = http.get("http://127.0.0.1:3000/http1-1/default/phone/0754952794", null, {
        headers: {"Content-Type" : "application/json"}
    })
    sleep(1);
}
```

```
data_received..................: 33 kB  2.1 kB/s
data_sent......................: 124 kB 7.8 kB/s
http_req_blocked...............: avg=7.56ms  min=0s med=0s    max=222.3ms  p(90)=0s       p(95)=63.49ms
http_req_connecting............: avg=7.55ms  min=0s med=0s    max=219.72ms p(90)=0s       p(95)=63.02ms
http_req_duration..............: avg=273.1ms min=0s med=0s    max=4.31s    p(90)=777.16ms p(95)=2.59s
  { expected_response:true }...: avg=3.39s   min=3s med=3.45s max=3.62s    p(90)=3.47s    p(95)=3.57s
http_req_failed................: 97.40% ✓ 8833       × 235
http_req_receiving.............: avg=1.44µs  min=0s med=0s    max=610.29µs p(90)=0s       p(95)=0s
http_req_sending...............: avg=97.36µs min=0s med=0s    max=59.44ms  p(90)=0s       p(95)=68.76µs
http_req_tls_handshaking.......: avg=0s      min=0s med=0s    max=0s       p(90)=0s       p(95)=0s
http_req_waiting...............: avg=273ms   min=0s med=0s    max=4.31s    p(90)=776.3ms  p(95)=2.59s
http_reqs......................: 9068   566.424564/s
iteration_duration.............: avg=1.68s   min=1s med=1.5s  max=5.34s    p(90)=1.99s    p(95)=3.61s
iterations.....................: 9068   566.424564/s
vus............................: 2      min=2        max=1000
vus_max........................: 1000   min=1000     max=1000
```


Fiber v2.52.4
http://127.0.0.1:3000
(bound on host 0.0.0.0 and port 3000)
Handlers ..... 20 Processes 1
Prefork ..... Disabled PID 35476
Get "http://localhost:8081/phone?number=0754952794": dial tcp [::1]:8081: connectex: No connection could be made because the target machine actively refused it.exit status 1

วิธีแก้ปัญหา

ทำ round-robin โดยการเพิ่ม Port ที่ตัว mock-lookup แล้วกระจายไปเรียก API แบบ วน loop



gRPC VS HTTP1.1

เทียบกันโดยใช้

- http แบบ round-robin

gRPC 1000VUs 10s			
ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs
1	3.35s	19.54%	3729
2	3.59s	38.27%	3809
3	3.78s	42.32%	3865
4	3.87s	46.35%	3922
5	3.65s	46.13%	3932
avg	3.648s	38.522%	3851.4

HTTP1.1 1000VUs 10s			
ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs
1	3.26s	35.12%	3906
2	3.46s	52.95%	3907
3	4.03s	57.29%	3948
4	3.91s	50.63%	3725
5	4.09s	54.07%	3739
avg	3.75s	50.012%	3845

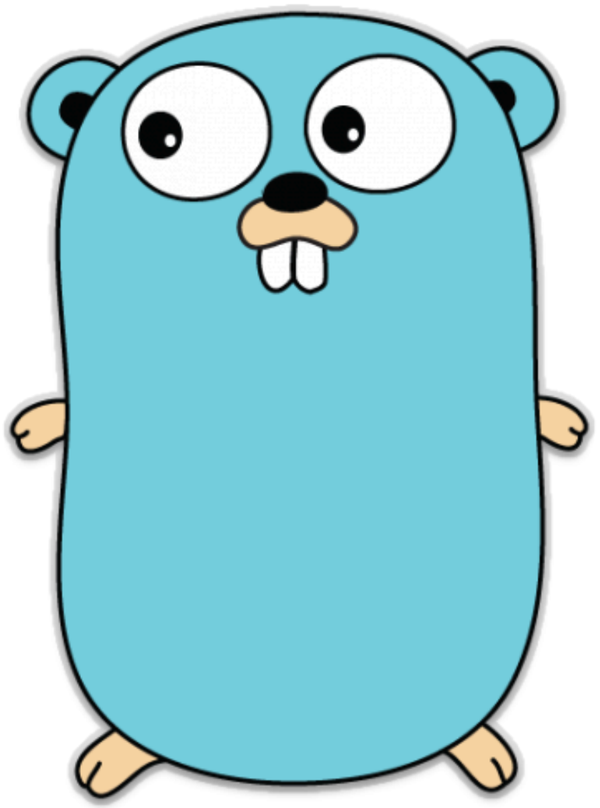
สรุป

gRPC มี http_req_failed และ http_req_duration น้อยกว่า http1.1 ดังนั้นจึงสรุปได้ว่า gRPC มีประสิทธิภาพมากกว่าในการทดลองครั้งนี้

HTTP2 VS HTTP1.1 VS gRPC

เทียบกันโดยใช้

- 3000VUs 10s
- Time Out 3.3s
- Round-Robin 3 Port



	gRPC 3000VUs 10s			HTTP1.1 3000VUs 10s			HTTP2 3000VUs 10s		
ครั้งที่	http_req_duration (p95)	http_req_failed	http_reqs	http_req_duration (p95)	http_req_failed	http_reqs	http_req_duration (p95)	http_req_failed	http_reqs
1	3.63s	47.85%	11948	4.56s	81.90%	12636	6.19s	85.55%	8669
2	3.9s	69.93%	13807	5.5s	87.73%	10044	6.36s	89.89%	8637
3	3.71s	56.93%	13394	6.96s	90.74%	8599	4.63s	83.15%	11090
4	3.6s	54.35%	11982	5.19s	86.24%	10525	6.41s	89.55%	8623
5	3.68s	56.90%	13281	5.46s	85.47%	10022	5.91s	87.81%	8647
avg	3.704s	57.192%	12882.4	5.534s	86.416%	10365	5.9s	87.19%	9133.2

*set time out ไว้ที่ 3.3s

สรุป

gRPC มีประสิทธิภาพมากที่สุด ในการทดลองด้วย k6 option 3000VUs 10s โดยที่ตัวเลขผลการทดลองมีความแตกต่างกันอย่างชัดเจน

Notion Link

https://warm-summer-ced.notion.site/6baf35f42d2143c49d25470ace4534c4?v=2db68ba2f0d74313baca7a62db85c573