

پروژه‌ی چهارم درس مهندسی نرم افزار ۲  
اعضای تیم -> عطیه براتی نیا، هانیه زرنندی، هانیه محمدی اقدام، فاطمه منصوری پور

سوال اول -> فاطمه منصوری پور

تست بار (Load Test) : ما می دانیم که در شرایط مختلف تقریباً چند کاربر همزمان از سیستم ما استفاده خواهند کرد. پس با کمک ابزارها و روش‌های مختلف، این تعداد کاربر همزمان را برای سیستم خود شبیه سازی خواهیم کرد و نحوه عملکرد و کارکرد سیستم را ارزیابی می نماییم. در این تست زمان پاسخگویی سیستم و تعداد کاربر همزمان تست می‌شود.  
تست فشار (Stress Test) : در این تست ما بیشترین تعداد کاربر همزمانی که می دانیم قرار است به سیستم ما وصل شود را شبیه سازی می نماییم و کارایی سیستم را ارزیابی می نماییم.  
تست استرس یا تست فشار، برای تعیین کارایی سیستم در حالتی که بار سیستم در حالت بیشینه باشد، مورد استفاده قرار می گیرد. در واقع برنامه در مقابل بار سنگینی مانند مقادیر عددی پیچیده ، مقادیر زیاد ورودی و مقادیر زیاد پرس و جو امتحان میشود. تا با مواجه ساختن برنامه با موقعیت های غیر معمول و تزریق بار سنگین، میزان تحمل برنامه بررسی و تست شود، ولی تست بار برای تعیین کارایی سیستم در حالتی که بار سیستم به صورت طبیعی باشد، مورد استفاده قرار می گیرد.  
مرجع ->

<https://devtube.ir/article/1064/%D8%AA%D8%B3%D8%AA-%D8%A7%D8%B3%D8%AA%D8%B1%D8%B3-%D9%88-%DA%A9%D8%A7%D8%B1%D8%A7%DB%8C%DB%8C-%D8%A8%D8%B1-%D8%B1%D9%88%DB%8C-%D9%88%D8%A8-%D8%B3%D8%A7%DB%8C%D8%AA-%D9%87%D8%A7>

سوال دوم -> عطیه براتی نیا

checkها مانند assertionها هستند، اما از این جهت متفاوت هستند که اجرا را متوقف نمی کنند. در عوض، آنها نتیجه بررسی، پاس یا شکست را ذخیره می کنند و اجازه می دهند اجرای اسکریپت ادامه یابد.

threshold -> معیارهایی هستند که برای تعیین انتظارات عملکرد سیستم مورد آزمایش، استفاده می‌شوند. مثلاً سیستم نباید بیش‌تر از یک درصد ارور داشته باشد یا اینکه زمان پاسخگویی 95 درصد تست‌ها باید کمتر از 200ms باشد.

تست بار واحد -> عطیه براتی نیا

در این تست چک میکنیم که آیا اندپوینت مدنظر توانایی پاسخگویی به 100 یوزر را دارد یا نه. یک check تعریف می‌کنیم که در آن بررسی می‌کند که آیا status=200 یعنی موفقیت آمیز می‌شود یا نه. همانطور که در خروجی مشخص است 100 درصد درخواست‌ها status code=200 برگرداندند. میانگین مدت زمان پاسخگویی به هر درخواست، میانگین مدت زمان منتظر شدن هر درخواست و ... در خروجی مشخص است.

```
File Edit Selection View Go Run Terminal Help
unit-test.js - Project - Visual Studio Code

JS unit-test.js X
unit-test.js
1 stages: {
2   { duration: '5s', target: 100 }, // simulate ramp-up of traffic from 1 to 100 users over 5 minutes.
3   { duration: '10s', target: 100 }, // stay at 100 users for 10 minutes
4   { duration: '5s', target: 0 }, // ramp-down to 0 users
5 }
6
7
8
9
10
11
12 const BASE_URL = 'https://test-api.k6.io';
13
14
15 export default function() {
16
17   const myObjects = http.get(`${BASE_URL}/public/crocodiles/`);
18   check(myObjects, { 'is status 200': (obj) => obj.status === 200, });
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
* default: Up to 100 looping VUs for 20s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

running (28.2s), 000/100 VUs, 2889 complete and 0 interrupted iterations
default ✓ [=====] 000/100 VUs 20s

✓ is status 200

checks.....: 100.00% ✓ 2889 X 0
data received.....: 3.0 MB 105 kB/s
data sent.....: 488 KB 15 kB/s
avg=53.02ms min=0s med=0s max=14.76s p(90)=0s p(95)=0s
http_req_blocked.....: avg=10.66ms min=0s med=0s max=2.23s p(90)=0s p(95)=0s
http_req_connecting.....: avg=483.65ms min=190.16ms med=364.94ms max=15.35s p(90)=861.19ms p(95)=1.06s
{ expected_response:true }...: avg=483.65ms min=190.16ms med=364.94ms max=15.35s p(90)=861.19ms p(95)=1.06s
http_req_failed.....: 0.00% ✓ 0 X 2889
http_req_receiving.....: avg=140.28µs min=0s med=0s max=1.99ms p(90)=674.44µs p(95)=342.03µs
http_req_sending.....: avg=11.99µs min=0s med=0s max=1.33ms p(90)=0s p(95)=0s
http_req_tls_handshaking.....: avg=42.35ms min=0s med=0s max=14.49s p(90)=0s p(95)=0s
http_req_waiting.....: avg=483.56ms min=190.16ms med=364.9ms max=15.35s p(90)=861.19ms p(95)=1.06s
http_req.....: 2889 102.438066/s
iteration duration.....: avg=536.98ms min=190.16ms med=376.32ms max=15.35s p(90)=935.13ms p(95)=1.25s
iterations.....: 2889 102.438066/s
vus.....: 1 min=1 max=100
vus_max.....: 100 min=100 max=100

Ln 11, Col 1 Spaces: 4 UTF-8 CRLF JavaScript Go Live
```

## تست بار سناریو

### سناریوی اول -> عطیه براتی نیا

در این سناریو پس از گرفتن لیست کروکودیل‌های موجود، یک آی دی به صورت رندم انتخاب شده و اندپوینت آن کروکودیل گرفته می‌شود. در انتها یک چک نوشته شده که آیا اندپوینت آخر status code=200 برگردانده یا نه که نشان دهنده‌ی موفقیت تست است.

سناریوی دوم -> عطیه براتی نیا انجام داد و هانیه محمدی اقدام و هانیه حاجی رجب زرندی کمک کردند.

در این سناریو ابتدا توکن ایجاد شده در مرحله ی login گرفته شده و در مرحله ی مشاهده‌ی کروکودیل‌های خصوصی از آن استفاده می‌شود. در مرحله‌ی بعدی چک، بررسی می‌شود آیا کد 200 برگردانده می‌شود یا خیر که مشاهده می‌شود با موفقیت کد 200 برگردانده شده است. در مرحله‌ی بعدی باید تعداد کروکودیل‌ها رو به 10 برسانیم. با گذاشتن یک if چک می‌کنیم اگر تعداد کروکودیل‌ها کمتر از 10 است کروکودیل اضافه شود.

اضافه کردن threshold به دو سناریو -> عطیه براتی نیا انجام داد و هانیه زرندی کمک کرد.

در این قسمت به دو سناریوی قبلی threshold اضافه کرده و نتیجه را بررسی می‌کنیم. دو threshold اضافه شده مربوط به http\_req\_failed و http\_req\_duration است که اولی بررسی می‌کند که تعداد درخواست‌های خطا نباید از 1 درصد بیشتر باشد و دومی بررسی می‌کند که 90 درصد درخواست‌ها باید زیر 800ms پاسخ داده شوند. ابتدا در سناریوی اول، با 10 کاربر در هر دو سناریو با لود کم برنامه را تست می‌کنیم. همانطور که در تصویر زیر مشخص است، چک‌ها با موفقیت ران شد و همچنین thresholdهای تعیین شده رعایت شده‌اند.

```
scenario2.js
1
2
3
4
5
6
7 let session = new Httpx({baseUrl: 'https://test-api.k6.io'});
8
9 const USERNAME = 'hanieeeeeh';
10 const PASSWORD = 'hanieh';

scenarios: (100.00%) 2 scenarios, 20 max VUs, 40s max duration (incl. graceful stop):
  * scenario1: 10 iterations shared among 10 VUs (maxDuration: 10s, exec: func1, gracefulStop: 30s)
  * scenario2: 10 iterations shared among 10 VUs (maxDuration: 10s, exec: func2, gracefulStop: 30s)

running (01.8s), 00/20 VUs, 20 complete and 0 interrupted iterations
scenario1 ✓ [=====] 10 VUs 01.6s/10s 10/10 shared iters
scenario2 ✓ [=====] 10 VUs 01.8s/10s 10/10 shared iters

✓ is status 200
✓ logged in successfully
✓ retrieved crocodiles

checks.....: 100.00% ✓ 30 X 0
data_received.....: 165 kB 91 kB/s
data_sent.....: 17 kB 9.2 kB/s
http_req_blocked.....: avg=326.57ms min=0s med=271.2ms max=1.12s p(90)=684.92ms p(95)=701.82ms
http_req_connecting.....: avg=97.51ms min=0s med=89.12ms max=213.31ms p(90)=202.59ms p(95)=206.88ms
http_req_duration.....: avg=326.62ms min=188.97ms med=223.2ms max=952.62ms p(90)=644.39ms p(95)=875.11ms
  { expected_response:true }...: avg=326.62ms min=188.97ms med=223.2ms max=952.62ms p(90)=644.39ms p(95)=875.11ms
http_req_failed.....: 0.00% ✓ 0 X 40
http_req_receiving.....: avg=168.52µs min=0s med=0s max=620.9µs p(90)=75.04µs p(95)=232.8µs
http_req_sending.....: avg=48.97µs min=0s med=0s max=620.9µs p(90)=75.04µs p(95)=232.8µs
http_req_tls_handshaking.....: avg=228.85ms min=0s med=181.96ms max=928.95ms p(90)=480.14ms p(95)=401.58ms
http_req_waiting.....: avg=326.41ms min=188.97ms med=222.9ms max=952.62ms p(90)=644.39ms p(95)=874.28ms
http_reqs.....: 40 22.127486/s
iteration_duration.....: avg=1.3s min=970.21ms med=1.21s max=1.8s p(90)=1.73s p(95)=1.78s
iterations.....: 20 11.063743/s
vus.....: 18 min=18 max=18
vus_max.....: 20 min=20 max=20

PS J:\Atieh\University\software engineering 2\Homeworks\HW4>
```

مرحله‌ی بعدی بار برنامه را زیاد کرده و برای هر سناریو 100 کاربر میگذاریم.

```
scenario2.js
16
17
18 scenarios: {
19   scenario1: {
20     executor: 'shared-iterations',
21     exec: 'func1'
22   }
23 },
24
25 at go.k6.io/js/common.bind.func1 (native)
26 at func2 (file:///J:/Atieh/University/software%20engineering%202/Homeworks/HW4/scenario-test2.js:75:378(18)) executor=shared-iterations scenario=scenario2 source=stacktrace

running (40.0s), 000/200 VUs, 199 complete and 1 interrupted iterations
scenario1 X [=====] 100 VUs 40.0s/10s 099/100 shared iters
scenario2 ✓ [=====] 100 VUs 35.9s/10s 100/100 shared iters

X logged in successfully
L 99% = ✓ 99 / X 1
✓ is status 200
✓ retrieved crocodiles

checks.....: 99.66% ✓ 297 X 1
data_received.....: 1.6 MB 41 kB/s
data_sent.....: 166 kB 4.2 kB/s
http_req_blocked.....: avg=1.61s min=0s med=773.49ms max=9.31s p(90)=4.91s p(95)=5.65s
http_req_connecting.....: avg=189.6ms min=0s med=189.6ms max=3.24s p(90)=3.2s p(95)=1.21s
http_req_duration.....: avg=2.8s min=188.33ms med=2.61s max=32.26s p(90)=4.71s p(95)=5.05s
  { expected_response:true }...: avg=2.73s min=188.33ms med=2.61s max=7.56s p(90)=4.7s p(95)=5.04s
http_req_failed.....: 0.25% ✓ 1 X 396
http_req_receiving.....: avg=137.07µs min=0s med=0s max=3.98ms p(90)=648.74µs p(95)=971.73µs
http_req_sending.....: avg=87.06µs min=0s med=0s max=6.61ms p(90)=48.66µs p(95)=506.9µs
http_req_tls_handshaking.....: avg=1.03s min=0s med=438.02ms max=9.05s p(90)=3.41s p(95)=4.85s
http_req_waiting.....: avg=2.8s min=188.33ms med=2.61s max=32.26s p(90)=4.71s p(95)=5.05s
http_reqs.....: 397 9.92379/s
iteration_duration.....: avg=8.81s min=1.65s med=8.71s max=35.89s p(90)=11.47s p(95)=11.59s
iterations.....: 199 4.974393/s
vus.....: 1 min=1 max=200
vus_max.....: 200 min=200 max=200

ERROR[0042] some thresholds have failed
PS J:\Atieh\University\software engineering 2\Homeworks\HW4>
```

همانطور که در تصویر مشخص است 99.66 درصد لاگین‌ها با موفقیت انجام شد و فقط یک مورد به خطا خورد. بقیه‌ی check‌ها به درستی ران شدند. همانطور که مشخص است یکی از threshold‌ها شکست خورد. وقتی بار سیستم زیاد شد http\_req\_duration سیستم افزایش یافت و از حد مجاز تجاوز کرد. در بار کم مدت زمان پاسخ دهی 90 درصد آنها 644ms بود که این زمان با تعداد کاربر زیاد به 4 ثانیه افزایش یافت.