

PUC Minas - Teste de Software

Relatório - Engenharia de Desempenho (Performance Testing)

Aluno(a): Ana Luiza Machado Alves

1. Resumo Executivo

Este relatório apresenta a avaliação de desempenho de uma API de checkout simulada, considerando dois cenários principais: I/O bound (/checkout/simple) e CPU bound (/checkout/crypto). No cenário de I/O, a aplicação foi submetida a testes de carga e de pico, mantendo latência p95 em torno de 295 ms e 0% de erros com até 300 usuários virtuais (VUs), o que indica que a API suporta, com folga, esse volume de acessos dentro do SLA adotado (p95 < 500 ms e taxa de erro < 1%).

Já no cenário de CPU, o teste de estresse com estágios de 200 até 1000 VUs provocou forte saturação: cerca de 95% das requisições falharam e as poucas bem-sucedidas apresentaram p95 próximo de 43 s. Assim, conclui-se que a capacidade prática do endpoint CPU-bound é significativamente inferior a 200 VUs, caracterizando ruptura sob cargas elevadas de processamento criptográfico.

Repositório do projeto: <https://github.com/analuizaam/ecommerce-checkout-api>

2. Evidências (k6)

Foram modelados quatro cenários de teste com k6: (i) Smoke Test em GET /health, com 1 VU por 30 s, para verificar disponibilidade; (ii) Teste de Carga em POST /checkout/simple, com ramp-up até 50 VUs, platô de 2 minutos e ramp-down, avaliando latência e taxa de erro sob carga sustentada; (iii) Teste de Pico (Spike) em POST /checkout/simple, partindo de 10 VUs e subindo rapidamente até 300 VUs, para observar o comportamento frente a picos abruptos; e (iv) Teste de Estresse em POST /checkout/crypto, com estágios de 0 → 200 → 500 → 1000 VUs, buscando identificar o ponto de ruptura no cenário CPU-bound. Abaixo, tem-se os resultados das execuções:

- **Smoke Test - /health:** 128.256 requisições em 30 s, 0% de erros, p95 ≈ 1 ms.

```

execution: local
  script: .\smoke.js
  output: -

scenarios: (100.00%) 1 scenario, 1 max VUs, 1m0s max duration (incl. graceful stop):
  * default: 1 looping VUs for 30s (gracefulStop: 30s)

■ THRESHOLDS

checks
✓ 'rate=1' rate=100.00%

http_req_failed
✓ 'rate=0' rate=0.00%

■ TOTAL RESULTS

checks_total.....: 128256 4275.112816/s
checks_succeeded...: 100.00% 128256 out of 128256
checks_failed.....: 0.00% 0 out of 128256

✓ status é 200

HTTP
http_req_duration.....: avg=195.85µs min=0s med=0s max=12.51ms p(90)=678.7µs p(95)=1ms
{ expected_response:true }...: avg=195.85µs min=0s med=0s max=12.51ms p(90)=678.7µs p(95)=1ms
http_req_failed.....: 0.00% 0 out of 128256
http_reqs.....: 128256 4275.112816/s

EXECUTION
iteration_duration.....: avg=229.78µs min=0s med=0s max=12.51ms p(90)=999.1µs p(95)=1ms
iterations.....: 128256 4275.112816/s
vus.....: 1 min=1 max=1
vus_max.....: 1 min=1 max=1

NETWORK
data_received.....: 37 MB 1.2 MB/s
data_sent.....: 9.7 MB 325 kB/s

running (0m30.0s), 0/1 VUs, 128256 complete and 0 interrupted iterations
default ✓ [=====] 1 VUs 30s

```

- **Carga - /checkout/simple:** ~6.877 requisições, 0% de erros, p95 ≈ 295 ms, cerca de 32 req/s com 50 VUs.

```

execution: local
  script: .\load.js
  output: -

scenarios: (100.00%) 1 scenario, 50 max VUs, 4m0s max duration (incl. graceful stop):
  * default: Up to 50 looping VUs for 3m30s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

■ THRESHOLDS

http_req_duration
✓ 'p(95)<500' p(95)=294.76ms

http_req_failed
✓ 'rate<0.01' rate=0.00%

■ TOTAL RESULTS

checks_total.....: 6877 32.679927/s
checks_succeeded...: 100.00% 6877 out of 6877
checks_failed.....: 0.00% 0 out of 6877

✓ status é 201

HTTP
http_req_duration.....: avg=206.57ms min=99.78ms med=207.6ms max=313.7ms p(90)=287.06ms p(95)=294.76ms
{ expected_response:true }...: avg=206.57ms min=99.78ms med=207.6ms max=313.7ms p(90)=287.06ms p(95)=294.76ms
http_req_failed.....: 0.00% 0 out of 6877
http_reqs.....: 6877 32.679927/s

EXECUTION
iteration_duration.....: avg=1.2s min=1.1s med=1.2s max=1.31s p(90)=1.28s p(95)=1.29s
iterations.....: 6877 32.679927/s
vus.....: 2 min=1 max=50
vus_max.....: 50 min=50 max=50

NETWORK
data_received.....: 2.0 MB 9.7 kB/s
data_sent.....: 2.0 MB 9.3 kB/s

running (3m30.4s), 00/50 VUs, 6877 complete and 0 interrupted iterations
default ✓ [=====] 00/50 VUs 3m30s

```

- **Pico (Spike) - /checkout/simple:** ~18.062 requisições, 0% de erros, p95 ≈ 293 ms, aproximadamente 128 req/s no pico com 300 VUs.

```

execution: local
  script: .\spike.js
  output: -

scenarios: (100.00%) 1 scenario, 300 max VUs, 2m50s max duration (incl. graceful stop):
  * default: Up to 300 looping VUs for 2m20s over 5 stages (gracefulRampDown: 30s, gracefulStop: 30s)

THRESHOLDS

http_req_failed
  ✓ 'rate<0.05' rate=0.00%

TOTAL RESULTS

checks_total.....: 18048 127.944469/s
checks_succeeded...: 100.00% 18048 out of 18048
checks_failed.....: 0.00% 0 out of 18048

✓ status é 201

HTTP
http_req_duration.....: avg=203.32ms min=99.27ms med=202.26ms max=313.8ms p(90)=283.52ms p(95)=293.52ms
  { expected_response:true }...: avg=203.32ms min=99.27ms med=202.26ms max=313.8ms p(90)=283.52ms p(95)=293.52ms
http_req_failed.....: 0.00% 0 out of 18048
http_reqs.....: 18048 127.944469/s

EXECUTION
iteration_duration.....: avg=1.2s min=1.09s med=1.2s max=1.31s p(90)=1.28s p(95)=1.29s
iterations.....: 18048 127.944469/s
vus.....: 1 min=1 max=300
vus_max.....: 300 min=300 max=300

NETWORK
data_received.....: 5.3 MB 38 kB/s
data_sent.....: 4.2 MB 30 kB/s

running (2m21.1s), 000/300 VUs, 18048 complete and 0 interrupted iterations
default ✓ [=====] 000/300 VUs 2m20s

```

- **Estresse - /checkout/crypto:** ~151.791 requisições totais, ~95% de falhas.

```

THRESHOLDS

http_req_failed
  ✗ 'rate<0.2' rate=95.27%

TOTAL RESULTS

checks_total.....: 154759 359.318059/s
checks_succeeded...: 4.72% 7311 out of 154759
checks_failed.....: 95.27% 147448 out of 154759

✗ status é 201
  ↳ 4% - ✓ 7311 / ✗ 147448

HTTP
http_req_duration.....: avg=655.63ms min=0s med=0s max=49.12s p(90)=0s p(95)=4.48s
  { expected_response:true }...: avg=12.52s min=56.16ms med=4.53s max=49.12s p(90)=41.85s p(95)=43.43s
http_req_failed.....: 95.27% 147448 out of 154759
http_reqs.....: 154759 359.318059/s

EXECUTION
iteration_duration.....: avg=1.15s min=500ms med=500.52ms max=49.62s p(90)=501.48ms p(95)=4.98s
iterations.....: 154754 359.30645/s
vus.....: 13 min=2 max=999
vus_max.....: 1000 min=1000 max=1000

NETWORK
data_received.....: 2.3 MB 5.3 kB/s
data_sent.....: 1.9 MB 4.4 kB/s

running (7m10.7s), 0000/1000 VUs, 154754 complete and 69 interrupted iterations
default ✓ [=====] 0000/1000 VUs 7m0s
ERRO[0431] thresholds on metrics 'http_req_failed' have been crossed

```

Entre as requisições bem-sucedidas, observou-se $p95 \approx 43$ s e tempos máximos próximos de 49 s.

3. Análise de Estresse

No teste de estresse, o endpoint CPU-bound /checkout/crypto foi submetido a um aumento progressivo de carga até 1000 VUs. Por se tratar de uma operação criptográfica (bcrypt) que consome intensamente CPU e bloqueia o event loop do Node.js, o servidor passou a acumular requisições em fila e a responder cada vez mais lentamente à medida que a carga aumentava. O k6 registrou aproximadamente 95% de falhas e cruzou o threshold configurado para http_req_failed, evidenciando que o sistema não consegue operar de forma estável nesse patamar.

Entre as requisições que ainda foram concluídas, a latência média ficou na ordem de segundos, com p95 em torno de 43 s, o que é incompatível com um cenário real de produção. Dessa forma, considera-se que o ponto de ruptura da aplicação no cenário CPU-bound é atingido antes da faixa de 200–1000 VUs utilizada no experimento, e que a capacidade máxima sustentável para esse endpoint deve ser estimada em valores bem inferiores a 200 usuários virtuais simultâneos.