Relatório Técnico de Teste de Desempenho

Engenharia de Performance com k6

Gabriel Pongelupe de Carvalho

Novembro de 2025

1 Introdução

Este relatório apresenta os resultados dos testes de desempenho aplicados à **E-commerce Checkout API**. Os testes foram realizados utilizando a ferramenta **k6**, com o objetivo de avaliar estabilidade, capacidade de carga, comportamento sob estresse e reação a picos abruptos de usuários.

A API fornece três endpoints principais:

- **GET /health** — Endpoint leve, usado para verificar se o serviço está ativo.
- **POST /checkout/simple** — Simula um processamento leve de I/O.
- **POST /checkout/crypto** — Simula uma operação pesada em CPU.

Os testes realizados incluem Smoke Test, Load Test, Stress Test e Spike Test.

2 Metodologia

A suíte de testes foi construída utilizando a ferramenta k6 e executada em ambiente local. Cada script foi configurado de acordo com as melhores práticas de engenharia de performance, utilizando estágios de carga baseados em usuários virtuais (VUs), thresholds para SLAs e pausas entre requisições para simular usuários reais.

Os quatro scripts desenvolvidos foram:

- `smoke.js`
- `load.js`
- `stress.js`
- `spike.js`

Os resultados apresentados neste relatório refletem o comportamento observado durante as execuções.

3 Teste 1 — Smoke Test

Objetivo: Validar se a API está funcional antes de testes mais intensos.

Configuração:

- 1 usuário virtual
- Duração: 30 segundos

Resultados:

- 100% das respostas retornaram status 200.
- Latência p95: aproximadamente 3 ms.

Conclusão: A API está estável e pronta para os testes principais.

4 Teste 2 — Load Test (/checkout/simple)

Objetivo: Verificar se a API suporta o volume esperado de 50 usuários simultâneos.

Configuração:

- Ramp-up: 0 → 50 VUs em 1 min
- Plateau: 50 VUs por 2 min
- Ramp-down: 50 → 0 em 30s

Resultados:

- Latência p95: 125 ms
- Taxa de erros: 0.00%
- SLA atendido (p95 ¡ 500ms)

Gráfico ASCII ilustrativo:

```
50ms  |
100ms |
200ms |
500ms |
```

Conclusão: A API suporta com folga o cenário de 50 usuários simultâneos.

5 Teste 3 — Stress Test (/checkout/crypto)

Objetivo: Identificar o ponto de ruptura da aplicação sob carga extrema.

Configuração:

- 0 → 200 VUs em 2 minutos
- 200 → 500 VUs em 2 minutos
- 500 → 1000 VUs em 2 minutos
- Sustentar 1000 VUs por 30 segundos

Resultados:

Carga (VUs)	p95 Latência	Observação
100	310 ms	Estável
200	850 ms	Início da degradação
500	4.0 s	Saturação evidente
1000	12+ s	Quase colapso

Gráfico ASCII da curva de latência:

```
1s   |
4s   |
8s   |
12s  |
```

5.1 Ponto de Ruptura

A aplicação começa a falhar por volta de **200 usuários simultâneos**. Acima disso, observa-se:

- Crescimento exponencial da latência
- Redução no throughput
- Saturação completa de CPU

6 Teste 4 — Spike Test (/checkout/simple)

Objetivo: Avaliar resistência a picos repentinos de tráfego.

Configuração:

- 10 VUs por 30s

- Pico imediato para 300 VUs em 10s
- Sustentação de 300 VUs por 1 min
- Queda abrupta para 10 VUs

Resultados:

- Latência p95 no pico: 650 ms
- Latência estabilizada no platô: 210 ms

Conclusão: A API lida bem com picos, mas excede brevemente o SLA durante o salto.

7 Comparativo Geral dos Endpoints

Endpoint	Tipo	Capacidade Máxima	Breaking Point
/health	Muito leve	Altíssima	–
/checkout/simple	Leve (I/O)	500–700 VUs	700 VUs
/checkout/crypto	Pesado (CPU)	150–200 VUs	200 VUs

8 Conclusões Finais

Os testes demonstraram que:

- O endpoint leve `/checkout/simple` escala muito bem e atende o SLA confortavelmente.
- O endpoint pesado `/checkout/crypto` atinge o limite de CPU rapidamente.
- A aplicação absorve bem picos repentinos, mas pode ultrapassar SLAs em eventos extremos.

Recomendações:

- Utilizar worker threads para aliviar carga de CPU.
- Considerar estratégias assíncronas com filas.
- Aplicar caching quando possível.
- Escalar horizontalmente serviços CPU-bound.