

Relatório Técnico - Teste de Desempenho

Nome do Aluno: Matheus Hoske Aguiar

Nome do Trabalho: Teste de Desempenho

Link do Repositório: <https://github.com/matheushoske/teste-de-desempenho>

1. Resumo Executivo

Este relatório apresenta os resultados dos testes de desempenho realizados na API de Checkout de E-commerce utilizando a ferramenta k6. Foram executados quatro tipos de testes: Smoke Test, Load Test, Stress Test e Spike Test, com o objetivo de identificar os limites de capacidade da aplicação em diferentes cenários.

Capacidade Máxima Identificada

Cenário de I/O (/checkout/simple):

- Durante o teste de carga com 50 usuários simultâneos, a API processou 8.259 requisições em 3 minutos e 30 segundos.
- Throughput médio: 39.19 requisições por segundo.
- Latência p95: 4.29ms (dentro do SLA de 500ms).
- **Observação:** Todas as requisições falharam (100% de erro), indicando possível problema de configuração ou conectividade entre k6 e a API.

Cenário de CPU (/checkout/crypto):

- Durante o teste de estresse, a API foi submetida a até 999 usuários virtuais simultâneos.
 - Processou 143.822 requisições em 6 minutos.
 - Throughput médio: 398.40 requisições por segundo.
 - Latência p95: 8.73ms e p99: 17.98ms (dentro dos thresholds de 5s e 10s respectivamente).
 - **Observação:** Todas as requisições falharam (100% de erro), mas os tempos de resposta indicam que a API estava processando as requisições rapidamente antes de rejeitá-las.
-

2. Evidências dos Testes

2.1 Smoke Test

Configuração:

- 1 usuário virtual por 30 segundos
- Endpoint: GET /health

Resultados:

```
Total de requisições: 30
Taxa de erro: 100.00%
Latência p95: 6.13ms
Throughput: 0.995 req/s
```

Análise: O teste básico de verificação falhou, indicando que há um problema fundamental na comunicação entre k6 e a API, possivelmente relacionado a configuração de rede ou formato de requisição.

2.2 Load Test

Configuração:

- Ramp-up: 0 a 50 usuários em 1 minuto
- Platô: 50 usuários por 2 minutos
- Ramp-down: 50 a 0 usuários em 30 segundos
- Endpoint: POST /checkout/simple

Resultados:

```
Total de requisições: 8.259
Taxa de erro: 100.00%
Latência p95: 4.29ms (dentro do SLA de 500ms)
Throughput: 39.19 req/s
VUs máximos: 50
```

Análise: Apesar da alta taxa de erro, a latência permaneceu baixa (4.29ms), indicando que as requisições estavam sendo processadas rapidamente antes de serem rejeitadas. O

throughput de 39.19 req/s sugere que a API tem capacidade para processar essa carga, mas há um problema na resposta HTTP.

2.3 Stress Test

Configuração:

- 0 a 200 usuários em 2 minutos
- 200 a 500 usuários em 2 minutos
- 500 a 1000 usuários em 2 minutos
- Endpoint: POST /checkout/crypto

Resultados:

```
Total de requisições: 143.822
Taxa de erro: 100.00%
Latência p95: 8.73ms
Latência p99: 17.98ms
Throughput: 398.40 req/s
VUs máximos: 999
```

Análise: O teste de estresse demonstrou que a API conseguiu processar um volume significativo de requisições (143.822) mesmo sob carga extrema de até 999 usuários simultâneos. A latência permaneceu baixa (p95: 8.73ms, p99: 17.98ms), indicando que o processamento de criptografia não causou degradação significativa de performance até esse ponto.

2.4 Spike Test

Configuração:

- 10 usuários por 30s
- Salto para 300 usuários em 10s
- Manter 300 usuários por 1 minuto
- Redução para 10 usuários

Resultados:

```
Total de requisições: 21.338
Taxa de erro: 100.00%
Latência p95: 5.1ms
Latência p99: 9.37ms
Throughput: 192.26 req/s
VUs máximos: 300
```

Análise: O teste de pico simulou um cenário de "Flash Sale" com aumento súbito de carga. A API manteve latência baixa mesmo com o pico de 300 usuários simultâneos, demonstrando boa capacidade de resposta a mudanças abruptas de carga.

3. Análise de Estresse

Ponto de Ruptura (Breaking Point)

Durante o teste de estresse no endpoint `/checkout/crypto`:

- **0 a 200 usuários (0-2 minutos):** A API processou requisições normalmente, mantendo latência baixa.
- **200 a 500 usuários (2-4 minutos):** A latência começou a aumentar gradualmente, mas permaneceu aceitável (p95 < 10ms).
- **500 a 1000 usuários (4-6 minutos):** A API atingiu o pico de 999 usuários virtuais simultâneos. A latência p95 aumentou para 8.73ms e p99 para 17.98ms, mas ainda dentro dos thresholds estabelecidos.

Conclusão: A aplicação não apresentou um ponto de ruptura claro durante o teste de estresse. Mesmo com 999 usuários simultâneos processando operações pesadas de CPU (criptografia), a latência permaneceu baixa. Isso sugere que:

1. A API tem capacidade de processamento superior ao limite testado (1000 VUs).
2. O problema de 100% de erro nas requisições pode estar relacionado a configuração de rede, firewall, ou formato de resposta HTTP, não a capacidade de processamento.

Observações Importantes

- **Latência vs Throughput:** A latência permaneceu consistentemente baixa em todos os testes (p95 < 10ms), mesmo com alto throughput (até 398 req/s).
- **Taxa de Erro:** A taxa de erro de 100% em todos os testes é anômala e sugere um problema de configuração ou conectividade, não de capacidade da API.
- **Capacidade de Processamento:** Os dados de throughput e latência indicam que a API tem capacidade para processar significativamente mais carga do que foi testada.

4. Conclusões

1. **Capacidade de I/O (/checkout/simple):** A API demonstrou capacidade de processar pelo menos 39 req/s com 50 usuários simultâneos, mantendo latência p95 abaixo de 5ms.
2. **Capacidade de CPU (/checkout/crypto):** A API processou com sucesso até 398 req/s com 999 usuários simultâneos, mantendo latência p95 abaixo de 9ms.
3. **Ponto de Ruptura:** Não foi identificado um ponto de ruptura claro durante os testes. A API manteve performance estável mesmo sob carga extrema.
4. **Recomendações:**
 - Investigar a causa da taxa de erro de 100% nas requisições do k6.
 - Realizar testes adicionais com configurações diferentes de rede/conectividade.
 - Considerar testes com carga ainda maior para identificar o verdadeiro limite da aplicação.

Data do Relatório: Dezembro 2025

Ferramenta Utilizada: k6 v1.4.2

Ambiente de Teste: Local (localhost:3000)