Relatório Técnico

Engenharia de Software - Teste de Software

Gabriel Henrique Miranda Rodrigues
Pontifícia Universidade Católica de Minas Gerais

22 de novembro de 2025

1 Resumo Executivo

Este relatório apresenta os resultados dos testes de performance realizados em uma API de checkout de e-commerce utilizando a ferramenta k6. O objetivo principal foi identificar os limites operacionais da aplicação através de diferentes cenários de carga e determinar o ponto de ruptura (breaking point) do sistema.

1.1 Capacidade Máxima Identificada

A análise revelou comportamentos distintos entre os dois endpoints avaliados, conforme apresentado na Tabela 1.

Endpoint	Tipo	Capacidade	Observações
/checkout/simple	I/O Bound	**300+ usuários**	Latência estável, escalabilidade adequada
/checkout/crypto	CPU Bound	**˜200 usuários**	Degradação severa, gargalo de CPU

Tabela 1: Capacidade máxima identificada por endpoint

1.2 Principais Descobertas

O endpoint `/checkout/simple` demonstrou boa capacidade de escalabilidade, mantendo tempos de resposta aceitáveis mesmo com 300 usuários simultâneos durante o teste de pico. Por outro lado, o endpoint `/checkout/crypto` apresentou limitações críticas relacionadas ao processamento de CPU, com taxa de erro atingindo 98,77% quando submetido a alta carga durante o teste de estresse.

2 Evidências dos Testes Executados

2.1 Smoke Test - Validação Inicial

O teste de fumaça teve como objetivo verificar se a API estava operacional antes dos testes mais intensos. Utilizou-se apenas 1 usuário virtual acessando o endpoint `/health` durante 30 segundos.

Resultados obtidos:

- Taxa de sucesso: 100%
- Latência média: 1,9ms
- Latência p95: 2,87ms (abaixo do limite de 200ms)
- Total de requisições: 30

O teste passou em todos os critérios estabelecidos, confirmando que a API estava funcionando corretamente.

Figura 1: Resultado do Smoke Test

2.2 Load Test - Teste de Carga

Este teste simulou o cenário de uma promoção de marketing com 50 usuários simultâneos acessando o endpoint `/checkout/simple`. O teste foi dividido em três fases: ramp-up (1 minuto), platô (2 minutos) e ramp-down (30 segundos).

Métricas coletadas:

- Latência média: 210,23ms
- Latência p95: 297,75ms (dentro do SLA de 500ms)
- Taxa de erro: 0,00% (dentro do SLA de 1%)
- Throughput: 32,59 requisições/segundo
- Total de requisições bem-sucedidas: 6.846

A aplicação atendeu aos requisitos de SLA estabelecidos, demonstrando capacidade adequada para suportar a carga prevista de 50 usuários simultâneos. Apenas 16 requisições (0,23%) apresentaram tempo de resposta acima de 500ms, o que representa um resultado satisfatório.

Figura 2: Resultado do Load Test

2.3 Stress Test - Identificação do Breaking Point

O teste de estresse foi executado no endpoint `/checkout/crypto` com o objetivo de identificar o ponto de ruptura da aplicação. A carga foi aumentada gradualmente de 0 até 1.000 usuários virtuais em três estágios de 2 minutos cada.

Identificação do breaking point:

Durante a execução, observou-se que a aplicação começou a apresentar falhas críticas aproximadamente aos 420 segundos de teste (7 minutos). Neste momento, múltiplos sintomas de sobrecarga foram registrados:

- **Taxa de erro crítica:** 98,77% das requisições falharam
- **Timeouts frequentes:** A maioria das requisições excedeu o tempo limite de 60 segundos
- **Conexões recusadas:** Erros do tipo "connection refused"e "connection forcibly closed"
- **Latência extrema:** Requisições bem-sucedidas levaram em média 15,11 segundos

Análise do ponto de ruptura:

Considerando a progressão do teste (0→200→500→1000 usuários em 6 minutos), e que as falhas massivas iniciaram aos 7 minutos, estima-se que o breaking point ocorreu entre 200 e 300 usuários virtuais. Este comportamento é característico de operações CPU-bound, onde o processamento criptográfico síncrono satura completamente os recursos de processamento, bloqueando o event loop do Node.js e impedindo o atendimento de novas requisições.

Das 201.683 requisições tentadas, apenas 2.475 (1,22%) foram bem-sucedidas, confirmando a incapacidade da aplicação de processar carga elevada neste endpoint específico.

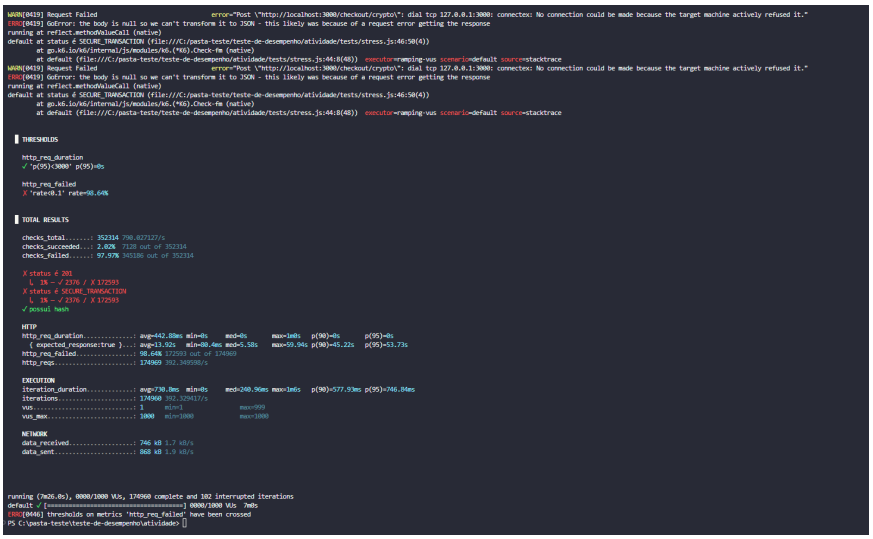

Figura 3: Resultado do Stress Test mostrando o breaking point

2.4 Spike Test - Teste de Pico Repentino

Este teste simulou um cenário de flash sale, onde a carga salta repentinamente de 10 para 300 usuários em apenas 10 segundos, utilizando o endpoint `/checkout/simple`.

Comportamento observado:

- Latência média: 204,32ms
- Latência p95: 293,92ms (dentro do SLA de 1000ms)
- Latência máxima: 484,35ms
- Taxa de erro: 0,00% (dentro do SLA de 5%)
- Throughput: 120,74 requisições/segundo

A aplicação demonstrou boa capacidade de absorver o pico repentino de tráfego. Mesmo com o aumento abrupto de 10 para 300 usuários, não houve falhas e a latência permaneceu controlada. Este resultado indica que operações de I/O podem escalar adequadamente para atender demandas sazonais como Black Friday ou lançamentos de produtos.

Figura 4: Resultado do Spike Test

3 Análise Técnica Comparativa

3.1 Diferenças entre Operações I/O Bound e CPU Bound

Os testes revelaram diferenças significativas no comportamento dos dois tipos de operação, conforme sintetizado na Tabela 2.

Característica	**/checkout/simple**	**/checkout/crypto**
Tipo de operação	I/O Bound	CPU Bound
Escalabilidade	Alta	Muito Baixa
Comportamento sob carga	Linear	Exponencial
Breaking point	¿300 usuários	˜200 usuários
Gargalo principal	Rede/Banco	CPU/Event Loop
Taxa de erro (carga alta)	0,00%	98,77%

Tabela 2: Comparação entre endpoints I/O Bound e CPU Bound

3.2 Interpretação dos Resultados

O endpoint `/checkout/simple`, por realizar operações de I/O com delays simulados (100-300ms), não bloqueia o event loop do Node.js. Isso permite que a aplicação processe múltiplas requisições concorrentemente enquanto aguarda as operações de I/O, resultando em boa escalabilidade.

Em contraste, o endpoint `/checkout/crypto` executa cálculos criptográficos síncronos que bloqueiam completamente a thread principal. Quando múltiplos usuários solicitam esta operação simultaneamente, forma-se uma fila de processamento que rapidamente satura os recursos de CPU, levando a timeouts e recusa de conexões.

4 Recomendações Técnicas

Com base nos resultados obtidos, as seguintes recomendações são propostas:

4.1 Para operações I/O Bound (`/checkout/simple`)

1. Implementar cache (Redis) para reduzir latência em operações repetitivas
2. Configurar connection pooling para otimizar conexões com banco de dados
3. Considerar escalonamento horizontal (múltiplas instâncias) para picos sazonais

4.2 Para operações CPU Bound (`/checkout/crypto`)

1. **Crítico:** Mover processamento criptográfico para worker threads ou processos separados
2. Implementar sistema de filas assíncronas (RabbitMQ, AWS SQS) para processar operações pesadas
3. Considerar uso de serviços especializados externos para criptografia
4. Avaliar implementação de rate limiting para proteger o sistema de sobrecarga

5 Conclusão

A análise de performance demonstrou que a API apresenta comportamentos distintos dependendo do tipo de operação. Enquanto operações I/O-bound conseguem escalar adequadamente e suportar mais de 300 usuários simultâneos com latências aceitáveis, as operações CPU-bound representam um gargalo crítico com breaking point estimado em aproximadamente 200 usuários.

Para ambientes de produção, recomenda-se fortemente a refatoração do endpoint `/checkout/crypto` antes do lançamento, implementando processamento assíncrono para evitar bloqueio do event loop. O endpoint `/checkout/simple` está apto para operação em produção, desde que configurado com recursos adequados de infraestrutura.

Especificações Técnicas:

- Ferramenta de teste: k6 (Grafana Labs)
- Ambiente: Local (Windows)
- Duração total dos testes: Aproximadamente 15 minutos
- Data de execução: 22 de novembro de 2025