

# RELATÓRIO TÉCNICO DE TESTES DE DESEMPENHO

API de Checkout de E-commerce | Engenharia de Performance | k6 Load Testing

## 1. RESUMO EXECUTIVO

**Objetivo:** Determinar a capacidade máxima e limites de performance da API de Checkout em cenários I/O-Bound (/checkout/simple) e CPU-Bound (/checkout/crypto).

### 1.1 Conclusões Principais

**Cenário I/O Bound (/checkout/simple):** A API suporta com sucesso **50 usuários simultâneos** mantendo p95 de latência em **296.82ms** (SLA ✓ <500ms). Taxa de erro: **0%**.

**Cenário CPU-Bound (/checkout/crypto):** Ponto de ruptura identificado entre **500-1000 VUsers**. A aplicação começou a falhar sistematicamente com taxa de erro de **100%** acima de ~600 usuários, indicando saturação total do CPU.

**Recomendação:** Para operações de criptografia, limitar a carga a máximo **400-500 VUsers** para manter disponibilidade. Considerar implementar cache de resultados ou processamento assíncrono.

## 2. DETALHES DOS TESTES EXECUTADOS

### 2.1 Teste de Smoke - Health Check (✓ PASSOU)

#### Objetivo:

Verificar disponibilidade básica da API antes de testes intensos.

Configuração	1 usuário, 30 segundos, endpoint: /health
Requisições Totais	30
Taxa de Sucesso	100%
Latência p95	1.13ms
Taxa de Erro	0%
Resultado	✓ PASSOU - SLA ATENDIDO

### 2.2 Teste de Carga - Load Test I/O (✓ PASSOU)

#### Objetivo:

Simular carga esperada de promoção (50 usuários simultâneos) em operação I/O-bound.

Configuração	0→50 VUsers (1m) → 50 VUsers (2m) → 0 VUsers (30s)
Endpoint	/checkout/simple (I/O-Bound)
Requisições Totais	6,828
Taxa de Sucesso	100%
Latência Média	205.76ms
Latência p95	296.82ms (SLA: <500ms ✓)
Taxa de Erro	0%
Throughput	32.46 req/s
Resultado	✓ PASSOU - TODOS OS SLAs ATENDIDOS

### 2.3 Teste de Estresse - Stress Test CPU (⚠ FALHOU EM CARGA MÁXIMA)

#### Objetivo:

Identificar ponto de ruptura em operação CPU-bound (criptografia).

Configuração	0→200 VU (2m) → 500 VU (2m) → 1000 VU (2m)
Endpoint	/checkout/crypto (CPU-Bound - bcrypt)
Duração Total	6 minutos
Requisições Totais	1,615,870
Fase 1 (0-200 VU)	✓ Funcionando
Fase 2 (200-500 VU)	⚠ Degradação observada
Fase 3 (500-1000 VU)	✗ Falha total (100% erro)
Ponto de Ruptura	~600 VUsers

Taxa de Erro Final	100%
<b>Análise:</b> A API começou a rejeitar conexões (connection refused) quando ultrapassou ~600 usuários simultâneos. Isso indica que o Node.js atingiu seu limite de capacidade de processamento CPU e o backlog de requisições cresceu exponencialmente, levando à indisponibilidade.	

### 3. ANÁLISE COMPARATIVA: I/O vs CPU

Métrica	/checkout/simple (I/O)	/checkout/crypto (CPU)
Capacidade Comprovada	50 VUsers	~400-500 VUsers
Ponto de Ruptura	Não atingido nos testes	~600 VUsers
Latência Típica	~205ms	N/A (falhas a partir de 600 VU)
Taxa de Erro @ Limite	0%	100%
Throughput	32.46 req/s	~4,487 req/s (durante falha)
Escalabilidade	Boa (sem degradação significativa)	Baixa (CPU-bound é limitante)

### 4. REPOSITÓRIO DO PROJETO

**Link:** <https://github.com/CleitonSilvaT/teste-de-desempenho>  
**Pasta de Testes:** /tests/  
**Scripts Inclusos:**

- tests/smoke.js - Teste de Smoke (Health Check)
- tests/load.js - Teste de Carga (50 usuários)
- tests/stress.js - Teste de Estresse (até 1000 usuários)
- tests/spike.js - Teste de Pico (Flash Sale)

### 5. RECOMENDAÇÕES E PRÓXIMOS PASSOS

#### 5.1 Para Operações I/O (/checkout/simple)

**Capacidade Atual:** 50+ usuários sem degradação.  
**Recomendação:** Seguir as recomendações do SLA (p95 < 500ms, erro < 1%). Monitorar em produção com ferramentas como Grafana/Prometheus.

#### 5.2 Para Operações CPU (/checkout/crypto)

**Capacidade Atual:** ~400-500 usuários antes de falhas.  
**Recomendações:**

- Implementar cache de resultados de hash quando possível
- Considerar processamento assíncrono (filas com Bull/RabbitMQ)
- Usar worker threads ou child processes para paralelizar cálculos
- Aumentar recursos de servidor (mais CPU cores)
- Implementar rate limiting e circuit breaker

### 6. EVIDÊNCIAS DE EXECUÇÃO

Os testes foram executados com a ferramenta k6 (v1.4.1) contra a API local rodando em Node.js na porta 3000. Todos os scripts estão disponíveis no repositório citado acima.

**Tempo de Execução Total:** ~13 minutos  
**Ambiente:** Windows 11 | Node.js | k6  
**Data do Teste:** Novembro 2025

