Relatorio Tecnico - Testes de Desempenho (1 pagina)

Projeto: ecommerce-checkout-api
Autor: Victor R. (preencher)
Data: 2025-11-24

1) Resumo executivo

- I/O (/checkout/simple): No cenario de Load (ramp-up 0->50 VUs, plato 50 VUs por 2min) a API sustentou 50 VUs com p95 ~ 296 ms. Conclusao: para operacoes I/O-bound simples a aplicacao responde dentro do SLA (p95 < 500ms) sob a carga esperada de 50 VUs.
- CPU (/checkout/crypto): o teste de stress completo nao gerou summary (thresholds violadas / conexoes recusadas durante a execucao). Portanto nao ha um numero exato final. Com base nos testes de spike e comportamento observado, estimo que o ponto de ruptura para operacoes CPU-bound esteja bem abaixo de 1000 VUs - provavelmente na faixa < 200 VUs (ver Analise abaixo). Recomenda-se execucao do stress-probe para diagnostico fino.

2) Evidencias (resumos k6 exportados)

- Smoke (tests/smoke.js) - results/smoke-summary.json
  - http_req_duration p(95) = 0.6046 s (~ 605 ms)
  - checks (status is 200): 142265 passes, 0 fails
- Load (tests/load.js) - results/load-summary.json
  - http_req_duration p(95) = 296.581435 ms
  - iterations = 6,874
  - checks (status is 201): 6,874 passes, 0 fails
- Spike (tests/spike.js) - results/spike-summary.json
  - http_req_duration (expected_response) p(95) ~ 50294.80 ms (~ 50.3 s)
  - http_req_duration avg (expected_response) ~ 14213 ms
  - http_req_failed rate ~ 0.979987 (~ 97.999% requests failed)
  - checks: 5,668 passes / 277,548 fails (~ 2% success)
- Stress: nao disponivel - k6 encerrou/violou thresholds e nao produziu results/stress-summary.json durante a execucao. Tambem houve conexoes recusadas em tentativas anteriores.

Arquivos exportados: results/smoke-summary.json, results/load-summary.json, results/spike-summary.json

3) Analise de estresse e ponto de ruptura (observacoes)

- I/O-bound (/checkout/simple)
  - Resultado claro: com 50 VUs o sistema apresentou p95 ~ 296 ms. Esse cenario atende o SLA definido (p95 < 500 ms).
- Spike (/checkout/simple)

  - Ao submeter um salto brusco (spike) a latencia explodiu: p95s muito grandes (ordem de dezenas de segundos) e alta taxa de falhas (~98%). Isso mostra que a aplicacao lida bem com cargas sustentadas moderadas, mas nao tolera bem picos subitos muito altos sem mitigacao (fila/caching/limitacao de taxa).
- CPU-bound (/checkout/crypto)
  - Nao foi possivel obter o summary do stress final. Com base no spike (altas latencias e falhas sob picos) e no fato de que /checkout/crypto roda bcrypt sincronico (bloqueante), e esperado que o throughput caia e o event loop seja bloqueado conforme a concorrencia aumenta. Estimativa conservadora: o ponto de ruptura para carga CPU-heavy deve estar substancialmente abaixo de 1000 VUs e provavelmente proximo ou abaixo de 200 VUs. Para determinar o breaking point com precisao, executar um stress-probe escalando progressivamente (por exemplo: 50 -> 100 -> 200 -> 400) e observar p95 e taxa de erro.

4) Ponto de ruptura estimado (resumo)

- I/O (/checkout/simple): suporta 50 VUs (testado) com p95 ~ 296 ms.
- CPU (/checkout/crypto): sem summary definitivo - estimativa de ruptura: $<$ 200 VUs (recomenda-se probe incremental para validar).

5) Recomendações e mitigacao

- Para I/O-bound:
  - Implementar cache (onde aplicavel) e otimizar latencia de I/O externo.
  - Usar filas/worker para suavizar picos (decoupling).
  - Adicionar protecao contra bursts (rate limiting / circuit breaker).
- Para CPU-bound:
  - Evitar trabalho sincronico no request handler: mover hashing/carga pesada para workers/filas/processos separados (child processes, worker pool, ou microservice).
  - Reduzir custo do algoritmo quando aceitavel (diminuir rounds do bcrypt apenas se seguro).
  - Escalonamento horizontal: aumentar instancias e balancear; usar mais cores por instancia.
- Operacional:
  - Executar stress-probe incremental (ate 200 VUs) para identificar breaking point com seguranca.
  - Configurar monitoramento de CPU, latencia e fila de requisicoes durante os testes.
  - Em producao: proteger contra picos subitos (queue, throttling).

6) Comandos usados e como reproduzir (execucao local)

- Rodar a API: npm install npm start # executa src/server.js na porta 3000
- Executar testes k6 (geram os JSONs em ./results): k6 run –summary-export=results/smoke-summary.json tests/smoke.js k6 run –summary-

export=results/load-summary.json tests/load.js k6 run –summary-export=results/spike-summary.json tests/spike.js # stress nao produziu summary no tempo da entrega; para probe seguro: k6 run –summary-export=results/stress-probe-summary.json tests/stress-probe.js

7) Observacoes finais sobre a entrega

- Entreguei o relatorio com base nos summaries disponiveis (smoke, load, spike). O summary do stress nao foi gerado por violacao de thresholds/conexoes recusadas durante a execucao - nao houve tempo suficiente para re-executar um stress confiavel antes do deadline.
- Se desejar, posso gerar o stress-probe (200 VUs) agora e atualizar o relatorio com o valor real do CPU-breaking point - ou, se preferir, voce pode rodar o stress-probe rapidamente e me enviar results/stress-probe-summary.json que eu atualizo o relatorio com os numeros exatos.