FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO

Testes e Qualidade de Software (DevOps)
Relatório de Teste de Carga

SÃO PAULO
2024

FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO

Gustavo Henrique Santos Araujo
Emilly Mickeli Depine Da Silva
Fernando Jose Dos Santos
Renan Teixeira Pinheiro

Testes e Qualidade de Software (DevOps)
Relatório de Teste de Carga

SÃO PAULO
2024

1. Introdução

Este relatório documenta o teste de carga realizado para avaliar a performance do servidor de um sistema de doação de alimentos. O sistema é composto por uma API hospedada no Azure, um banco de dados MySQL gerenciado via MySQL Workbench, e um aplicativo mobile desenvolvido em Android Studio com o uso da biblioteca Retrofit para realizar requisições HTTP.

O objetivo deste teste é garantir que o sistema consiga lidar com um número significativo de requisições simultâneas, sem degradação na performance e sem falhas de serviço, de modo que o sistema possa ser utilizado por um grande número de usuários.

2. Objetivo do Teste

O principal objetivo do teste de carga é medir o comportamento do servidor quando submetido a uma carga crescente de requisições simultâneas. O foco está em verificar:

- A capacidade do servidor de suportar um volume crescente de requisições sem falhas.
- A performance da API em termos de tempo de resposta.
- O impacto no banco de dados MySQL quando múltiplas requisições simultâneas são realizadas.
- A estabilidade e escalabilidade do sistema ao longo do tempo.

3. Metodologia

3.1 - Ambiente de Teste

Servidor: Hospedagem na Azure, com API e banco de dados MySQL.
Ferramentas de Teste de Carga: Para simular a carga no servidor, utilizamos a ferramenta Apache JMeter.

Estrutura do Sistema:
Banco de dados: MySQL, hospedado no Azure.
API: Desenvolvida em Visual Studio, implementada com RESTful endpoints.
Aplicativo: Desenvolvido em Android Studio, utilizando Retrofit para realizar requisições HTTP.

3.2 - Planejamento de Teste

O teste de carga será executado em três fases principais:

Teste Inicial (Cenário Normal):
Realizar requisições para emular o uso cotidiano do sistema com um número moderado de usuários.
Número de requisições simuladas: 100 usuários simultâneos.

Teste de Carga Alta:
Submeter o servidor a um número elevado de requisições simultâneas para verificar o limite de performance do sistema.
Número de requisições simuladas: 1000 usuários simultâneos.

Teste de Estabilidade:
Realizar um teste prolongado para verificar como o sistema se comporta com a carga ao longo do tempo (stress test).
Número de requisições: 100 a 500 usuários simultâneos durante um período de 1 hora.

3.3 - Cenários de Teste

O teste irá simular os seguintes cenários de uso:

Cadastro de usuário: Requisição POST para registrar um novo usuário.
Login de usuário: Requisição POST para realizar login e autenticação.
Consulta de alimentos disponíveis: Requisição GET para listar alimentos disponíveis para doação.
Registro de alimentos para doação: Requisição POST para registrar alimentos no banco de dados.

3.4 - Métricas de Desempenho

As seguintes métricas serão analisadas:

Tempo de resposta: O tempo médio e máximo de resposta para as requisições.
Taxa de erro: Percentual de requisições que falham (ex: erros 500, 404).
Throughput: A quantidade de requisições processadas por unidade de tempo.
Uso de recursos do servidor: Utilização de CPU, memória e rede.
Latência: O tempo que leva para uma requisição sair do cliente até o servidor e retornar.

4. Resultados Esperados

A partir dos testes, espera-se que o sistema:

- Suporte pelo menos 1000 usuários simultâneos durante o Teste de Carga Alta, com tempo de resposta abaixo de 2 segundos para 95% das requisições.
- Não apresente falhas críticas ou quedas de desempenho durante o Teste de Estabilidade.
- Mantenha o tempo de resposta inferior a 3 segundos mesmo durante picos de carga.
- Tenha uma taxa de erro inferior a 1% durante os testes, indicando estabilidade e capacidade de recuperação em caso de falha.

5. Execução do Teste

5.1 - Teste Inicial (Cenário Normal)

Foram simuladas 100 requisições simultâneas, com o objetivo de emular um cenário de uso normal para o aplicativo. Durante essa fase, o desempenho do sistema foi monitorado quanto ao tempo de resposta, taxa de erro e recursos do servidor.
Tempo de Resposta Médio: 1.2 segundos.
Taxa de Erro: 0%.
Throughput: 50 requisições por segundo.
Uso de CPU e Memória: Utilização de CPU foi de 35%, e memória de 60% durante o pico.

5.2 - Teste de Carga Alta

Para este teste, simulamos 1000 requisições simultâneas, buscando identificar o ponto de quebra do servidor.
Tempo de Resposta Médio: 3.4 segundos.
Taxa de Erro: 2% (erros 500).
Throughput: 100 requisições por segundo.
Uso de CPU e Memória: Uso de CPU atingiu 80%, e memória chegou a 85% durante picos.

5.3 - Teste de Estabilidade

O teste de estabilidade durou 1 hora, com 200 requisições simultâneas sendo realizadas ao longo do tempo.
Tempo de Resposta Médio: 2.5 segundos.
Taxa de Erro: 1%.
Throughput: 150 requisições por segundo.
Uso de CPU e Memória: Utilização de CPU foi de 70%, e memória de 80% ao longo do teste.

6. Análise de Resultados

Desempenho Geral: O sistema apresentou um bom desempenho durante o Teste Inicial, com tempos de resposta médios abaixo de 2 segundos. No Teste de Carga Alta, o tempo de resposta subiu para 3.4 segundos, mas ainda foi aceitável. A taxa de erro aumentou devido ao pico de carga no servidor, indicando que o sistema precisará de ajustes para lidar com mais de 1000 usuários simultâneos.

Limitações do Sistema: O uso elevado de CPU e memória durante o Teste de Carga Alta sugere que o servidor precisa ser escalado para lidar com mais requisições simultâneas. Recomenda-se avaliar a utilização de escala automática na Azure para aumentar a capacidade conforme a demanda.

Estabilidade a Longo Prazo: Durante o Teste de Estabilidade, o sistema se comportou bem, mantendo o tempo de resposta dentro de um limite aceitável,

embora a memória estivesse próxima do limite máximo. A escalabilidade do banco de dados e a otimização das consultas podem ajudar a melhorar a performance nesse aspecto.

7. Conclusões e Recomendações

Com base nos testes realizados, concluímos que o sistema está funcional para um número razoável de usuários simultâneos, mas existem áreas de melhoria, principalmente em relação à escalabilidade do servidor e otimização do banco de dados.

Recomendações:

Escalabilidade automática: Habilitar a escalabilidade automática no Azure para garantir que o servidor consiga lidar com picos de tráfego.
Otimização do banco de dados: Realizar ajustes no banco de dados, como a indexação de consultas e o uso de cache, para reduzir a carga no MySQL durante picos de requisições.
Aumento de recursos: Aumentar a capacidade de CPU e memória para o servidor de API durante os períodos de maior demanda.