MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - PICOS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO
DISCIPLINA: REDES DE COMPUTADORES II
PROFESSOR(A): RAYNER GOMES SOUSA
ALUNOS: MARCOS ANTONIO E VANDIRLEYA BARBOSA

CÓDIGO SERVIDOR

Importações e Configuração do Logging
O código começa com as importações necessárias para a implementação do servidor web: socket, threading, mimetypes, os e logging. O módulo logging é configurado para registrar mensagens de informação com um formato que inclui a data, o nível de log e a mensagem.

Função handle_client_connection
A função handle_client_connection é responsável por processar a requisição HTTP recebida de um cliente. A requisição é lida e decodificada, e a linha de requisição é dividida em partes. Se a requisição não for válida, é enviada uma resposta de erro "400 Bad Request". Se o método da requisição não for GET, é enviada uma resposta "405 Method Not Allowed". O caminho padrão é definido como /index.html se o caminho solicitado for /. Se o arquivo solicitado não existir ou não for acessível, são enviadas respostas "404 Not Found" ou "403 Forbidden", respectivamente. Caso contrário, o arquivo é enviado utilizando a função send_file.

Função send_response
A função send_response cria e envia uma resposta HTTP com o código de status e a mensagem fornecida. Esta função é utilizada para enviar respostas de erro quando a requisição não pode ser processada corretamente.

Função send_file
A função send_file determina o tipo MIME do arquivo solicitado utilizando o módulo mimetypes. O conteúdo do arquivo é lido e enviado junto com os cabeçalhos HTTP apropriados.

Função run_server
A função run_server inicializa o servidor, cria um socket de servidor e começa a escutar por conexões de clientes. Quando uma nova conexão é aceita, uma nova thread é criada para processar a requisição do cliente utilizando a função handle_client_connection.

A execução do servidor é iniciada chamando a função run_server no endereço 0.0.0.0 e na porta 8080.

CÓDIGO SERVIDOR PARALELO

Importações e Configuração do Logging
O código começa com as importações necessárias para a implementação do servidor web: socket, threading, mimetypes, os e logging. O módulo logging é configurado para registrar mensagens de informação com um formato que inclui a data, o nível de log e a mensagem.

Função process_request
A função process_request é responsável por processar a requisição HTTP recebida de um cliente. A requisição é lida e decodificada, e a linha de requisição é dividida em partes. Se a requisição não for válida, é enviada uma resposta de erro "400 Bad Request". Se o método da requisição não for GET, é enviada uma resposta "405 Method Not Allowed". O caminho padrão é definido como /index.html se o caminho solicitado for /. Se o arquivo solicitado não existir ou não for acessível, são enviadas respostas "404 Not Found" ou "403 Forbidden", respectivamente. Caso contrário, o arquivo é enviado utilizando a função send_file_content.

Função send_http_response
A função send_http_response cria e envia uma resposta HTTP com o código de status e a mensagem fornecida. Esta função é utilizada para enviar respostas de erro quando a requisição não pode ser processada corretamente.

Função send_file_content
A função send_file_content determina o tipo MIME do arquivo solicitado utilizando o módulo mimetypes. O conteúdo do arquivo é lido e enviado junto com os cabeçalhos HTTP apropriados.

Função run_parallel_server
A função run_parallel_server inicializa o servidor, cria um socket de servidor e começa a escutar por conexões de clientes. Quando uma nova conexão é aceita, uma nova thread é criada para processar a requisição do cliente utilizando a função process_request.

A execução do servidor é iniciada chamando a função run_parallel_server no endereço 0.0.0.0 e na porta 8081.

CÓDIGO HTML SIMPLES

O arquivo HTML index.html foi utilizado para testar a funcionalidade do servidor, exibindo uma mensagem simples "Olá, mundo!".

CÓDIGO CLIENTE

Importações e Configuração do Logging
O código começa com as importações necessárias: socket para comunicação de rede, time para medição de tempo, matplotlib.pyplot para geração de gráficos, numpy para operações

matemáticas e os para manipulação de diretórios. O módulo logging é configurado para registrar mensagens informativas.

Variáveis Globais
São definidas variáveis globais para armazenar os diferentes números de requisições a serem testados, o caminho do arquivo a ser requisitado, o diretório de saída para os gráficos e os endereços e portas dos servidores sequencial e paralelo.

Função make_request
A função make_request realiza uma requisição HTTP GET para o host e porta especificados. A função cria um socket, conecta-se ao servidor, envia a requisição e recebe a resposta.

Função measure_response_times
A função measure_response_times mede os tempos de resposta para um número especificado de requisições. Para cada requisição, o tempo de início e fim são registrados, e o tempo de resposta é calculado e armazenado em uma lista.

Função save_plot
A função save_plot gera e salva um gráfico de linhas dos tempos de resposta. O gráfico é salvo no diretório de saída especificado.

Função save_comparison_plot
A função save_comparison_plot gera e salva um gráfico comparativo de barras, mostrando os tempos médios de resposta dos servidores sequencial e paralelo para diferentes números de requisições.

Função run_tests
A função run_tests executa os testes de desempenho, medindo os tempos de resposta dos servidores sequencial e paralelo para diferentes números de requisições. Os tempos médios são calculados e armazenados, e os gráficos são gerados e salvos.

Execução dos Testes
A execução dos testes é iniciada chamando a função run_tests no bloco principal do script.

CÓDIGO DOCKER

Dockerfile para Servidor Sequencial
O Dockerfile para o servidor sequencial configura um ambiente de execução baseado na imagem python:3.9-slim. O script servidor.py e o arquivo index.html são copiados para o contêiner, e o servidor é iniciado com o comando python servidor.py.

Dockerfile para Servidor Paralelo
O Dockerfile para o servidor paralelo é semelhante ao do servidor sequencial, mas copia o script servidor_paralelo.py e inicia o servidor paralelo com o comando python servidor_paralelo.py.

Dockerfile para o Cliente

O Dockerfile para o cliente configura um ambiente de execução baseado na imagem python:3.9-slim. O script cliente.py é copiado para o contêiner, e a biblioteca matplotlib é instalada para a geração de gráficos. O cliente é iniciado com o comando python cliente.py.

Orquestração com Docker Compose

O arquivo docker-compose.yml define os serviços para os servidores sequencial e paralelo, e para o cliente. Cada servidor é construído a partir de seu respectivo Dockerfile e expõe a porta apropriada. O serviço do cliente monta um volume para armazenar os gráficos de saída e depende da inicialização dos servidores.

EXECUÇÃO DOS TESTES

Configuração e Inicialização dos Contêineres

Os contêineres foram inicializados usando o comando docker-compose up. Este comando constrói as imagens dos Dockerfiles e inicia os contêineres de acordo com as especificações no arquivo docker-compose.yml.

Sequencia de comandos docker

Buildar imagem após dockerfile ser criado
docker build -t nome_imagem -f nome_arquivo_dockerfile.dockerfile . (faça isso para cada servidor e o cliente)

Verificar imagens criadas
docker images

Executar imagem
docker run -d nome_da_imagem (aqui executa o docker compose no lugar)

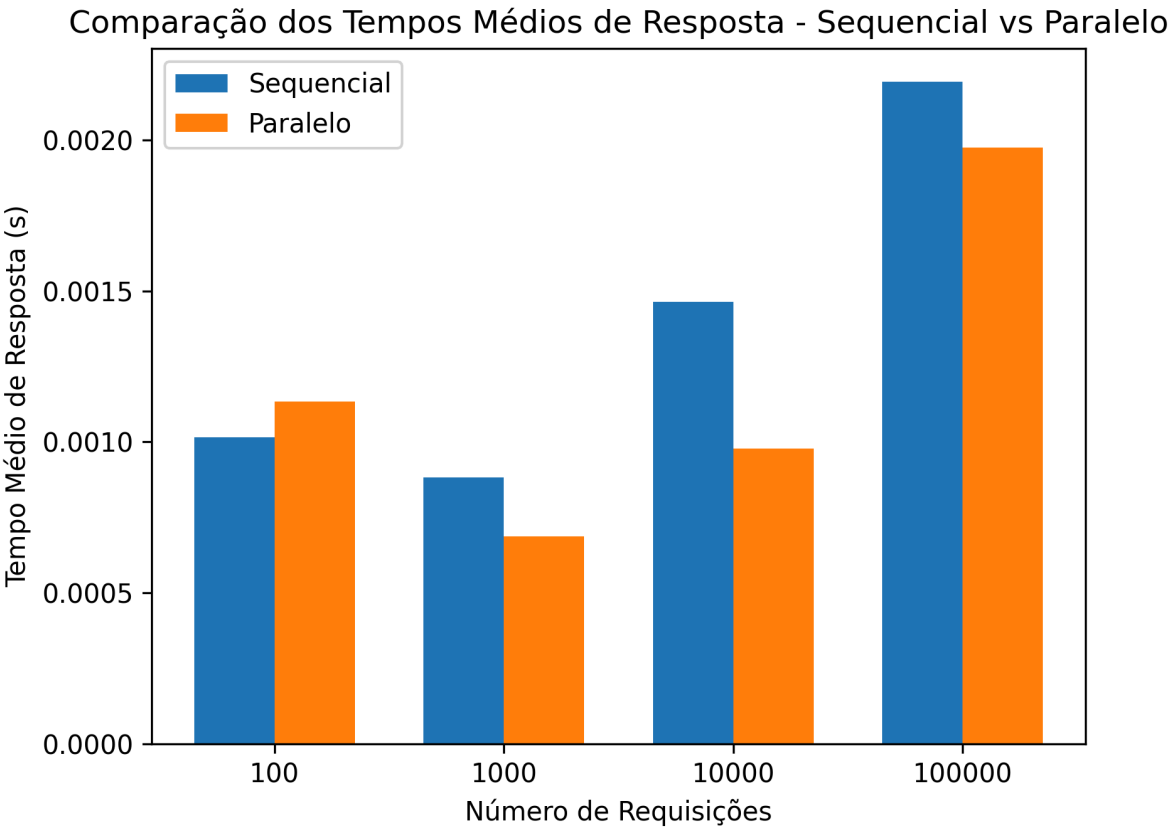
Remover imagem
docker rmi -f nome_imagem

Medição dos Tempos de Resposta

O cliente implementado em Python faz requisições HTTP para os servidores sequencial e paralelo, medindo os tempos de resposta para diferentes números de requisições (100, 1000, 10000 e 100000). Os tempos médios de resposta são calculados e gráficos comparativos são gerados.

RESULTADOS

Na figura abaixo podemos observar o resultado dos tempos de execução no servidor sequencial e paralelo e seus respectivos tempos de resposta com uma determinada quantidade N de requisições.


Comparação dos Tempos Médios de Resposta - Sequencial vs Paralelo
Sequencial
Paralelo
Tempo Médio de Resposta (s)
0.0020
0.0015
0.0010
0.0005
0.0000
100
1000
10000
100000
Número de Requisições