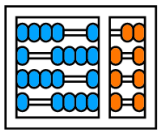



UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

MC714 — Sistemas Distribuídos
2° Semestre de 2024

Projeto 1

Balanceador de Carga para um Sistema Distribuído

Carlos Alberto Astudillo Trujillo (Professor)
Silvio Eduardo Sales de Britto Ribeiro (Estudante de Mestrado - PED)
Luis Fernando Solis Navarro (Estudante de Mestrado - PED)

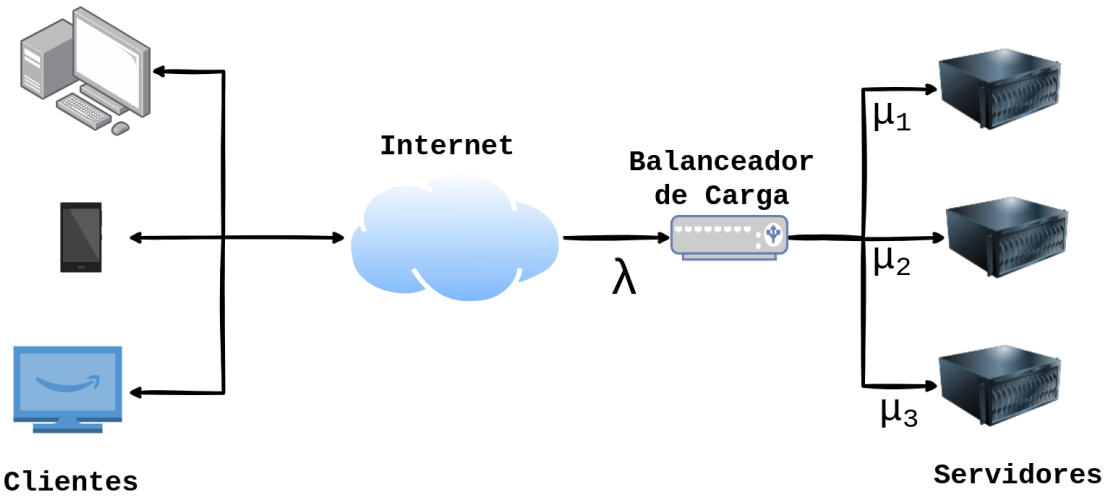
Campinas, Agosto de 2024

1 Instruções

O balanceamento de carga em um sistema distribuído é uma técnica fundamental para otimizar o uso de recursos e garantir eficiência. O objetivo é distribuir a carga entre pontos do sistema, evitando que alguns fiquem sobrecarregados enquanto outros permanecem subutilizados, trazendo mais escalabilidade e uma melhor alocação dos recursos. Este projeto visa a implementação de um balanceador de carga que receba requisições e seja capaz de alocar tais requisições em sistemas computacionais distribuídos, levando em consideração algumas políticas. Simule o tráfego do sistema e implemente as diferentes políticas para o balanceamento da carga. Pense porque e quando cada política é mais adequada do que a outra, e em quais contextos se aplicam. Entregue um relatório de até 4 páginas descrevendo o processo, além do código fonte. Este projeto deverá ser desenvolvido em dupla. A avaliação levará em consideração o relatório e o código fonte da solução.

1.1 Trabalho a fazer

- Linguagem de programação e plataforma:
  - Escolha qualquer linguagem de programação (por exemplo, Python, Java, C++, Go).
- Políticas fundamentais de balanceamento de carga: implemente pelo menos as seguintes três políticas.
  - *Escolha Aleatória*: o balanceador de carga seleciona aleatoriamente um servidor do conjunto para cada pedido recebido.
  - *Round Robin*: O balanceador de carga distribui igualmente as requisições entre os servidores, sem levar em consideração o tamanho da requisição nem os recursos disponíveis do servidor.
  - *Fila Mais Curta*: O balanceador de carga agora deverá monitorar a fila de execução de todos os servidores, e atribuir a requisição atual ao servidor com a menor fila.
- Simulação de tráfego
  - Simule o tráfego de entrada gerando uma sequência de solicitações.
  - Varie as requisições – algumas consumirão mais tempo de CPU, outras menos, outras mais tempo de I/O.
  - Os pedidos devem chegar em intervalos aleatórios, imitando cenários do mundo real.
  - Certifique-se de que o balanceador de carga pode lidar com uma variedade de padrões de tráfego, incluindo explosões de pedidos e fluxos constantes.
- Simulação de servidores
  - Simule 3 servidores que possam lidar com as requisições. Defina uma fila para armazenar as requisições e um tempo para o processamento de cada servidor. A capacidade de processamento e os recursos dos servidores podem ser iguais.
  - Deve ser adicionado um ligeiro atraso para simular o tempo real de processamento. Esse atraso pode ser variável de acordo com a requisição, isto é, se a requisição faz uso intensivo de CPU o atraso é um, se faz mais I/O o atraso pode ser maior.
  - O estado do servidor (ocupado, inativo) e o comprimento da fila devem ser monitorados.
- Detalhes da implementação:
  - O balanceador de carga deve ter uma opção de configuração para alternar entre as três políticas.
  - A implementação dever ser capaz de mostrar como a carga foi distribuída para cada política utilizada, através de um log ou apresentando as informações das alocações na saída padrão.
  - Registre o desempenho do balanceador de carga em diferentes condições de tráfego e nas três diferentes políticas através de, pelo menos, duas métricas principais: **vazão do sistema (*throughput*), tempo médio de resposta (*average response time*).** Outras métricas poderão ser utilizadas, por exemplo, utilização do sistema.



- Documentação e Relatório:
  - Forneça instruções claras sobre como compilar e executar o seu código.
  - Inclua um relatório curto explicando a estrutura do seu código, as escolhas de projeto que você fez e como testar o balanceador de carga.
  - Inclua uma descrição de como foi feita a distribuição do trabalho, especificando a tarefa desenvolvias por cada membro da dupla. Isto será verificado e contrastado com o histórico do repositório do projeto.
  - Faça uma análise dos resultados das diferentes políticas considerando as métricas avaliadas.
  - Comente o seu código adequadamente para explicar o objetivo das seções principais.
- Instruções de Submissão:
  - Submeta o seu código-fonte num único ficheiro comprimido (zip ou tar.gz).
  - Certifique-se de que o seu código é executável num ambiente Windows ou Linux.
- Ponto extra (opcional): Implemente políticas adicionais de balanceamento de carga, como:
  - Variar o poder de processamento dos servidores, permitindo que alguns lidem com mais solicitações do que outros e fazendo com que o balanceador de carga utilize essa informação para alocar as requisições.
  - Atribuir um servidor de backup que deverá ser acionado quando o buffer de algum dos servidores ficar cheio.

1.2 Entregáveis

- `relatorio_pojeto1_nome_de_um_integrante_do_grupo.pdf`
- `codigo_pojeto1_nome_de_um_integrante_do_grupo.zip`

1.3 Data de entrega

A data de entrega é no dia **24 de setembro de 2024**. Deverão subir os arquivos no Classroom dentro da tarefa do projeto. Não precisam duplicar as submissões, apenas uma pessoa por grupo deverá submeter o projeto. Recomenda-se submeter os trabalhos com antecedência e não esperar aos últimos minutos para a submissão.

2 Relatório

- Prepare um relatório de **máximo 4 páginas** de conteúdo no formato IEEE (coluna dupla).
- O relatório deve conter, pelo menos:
  - Resumo.
  - Descrição da criação do balanceador de carga, políticas adotadas e decisões de implementação.
  - Descrição da criação dos servidores e seu funcionamento.
  - Descrição do tráfego gerado.
  - Descrição das métricas de avaliação utilizadas.
  - Conclusão (explicar os pontos principais do trabalho)

3 Dicas

- **Não comece o projeto dias antes da submissão**. Planejem sua entrega e façam avanços oportunamente.
- Provavelmente na linguagem que você escolher, existirão bibliotecas que facilitam simulação de tráfego, servidores, etc. **Se desejar**, é permitido usá-las. No Python, por exemplo, existe a Simpy[1] que pode lhe ser útil. Em Java, tem a JSL [2].

4 Avaliação

O projeto abrange a avaliação do relatório (conteúdo, estruturação, redação e organização) e o código-fonte. Serão avaliados o relatório entregue, assim como o trabalho realizado no repositório (através do histórico no repositório). O detalhe da avaliação é apresentado na Tabela 1.

- É proibido copiar o trabalho de outra dupla. Se detectado, todos os envolvidos serão penalizados.

Critérios de avaliação:

- Correção: O balanceador de carga implementa corretamente as três políticas?
- Eficiência: O balanceador de carga lida bem com diferentes padrões de tráfego?
- Qualidade do código: O código é bem estruturado, modular e fácil de entender?
- Documentação: O relatório é claro e o código está devidamente comentado?

Tabela 1: Avaliação do projeto.

Item	**%**
Relatório	30
Geração de Tráfego/Requisições (código e funcionalidade)	15
Implementação dos Servidores (código e funcionalidade)	15
Implementação do Mecanismo de Balanceamento de Carga (código e funcionalidade)	20
Implementação das Políticas de Balancemaento de Carga (código e funcionalidade)	20

[1] https://simpy.readthedocs.io/en/latest/
[2] https://rossetti.github.io/JSLBook/