Pró-Reitoria Acadêmica

Curso de Ciência da Computação

Trabalho de Programação Concorrente e distribuída

Algoritmo de Busca utilizando Sockets

Autores:

Breno Camarô Arraes

Caio Vinícius Rodrigues Moreira

Gabriel de Paula Alvarenga Adeodato

Lucas Sabino Assis

Orientador(a): Prof. João Robson Santos Martins

Brasília – DF

2024

1) **O que é computação distribuída:**

Segundo a AWS, a computação distribuída consiste no uso de vários computadores ou processadores trabalhando em conjunto para resolver um único problema computacional ou executar uma tarefa. Já Tanenbaum define a computação distribuída como a interação de múltiplos processadores autônomos, que não compartilham memória primária, mas cooperam entre si por meio do envio de mensagens através de uma rede de comunicação.

2) **Conceitos de escalabilidade e tolerância a falhas e como eles se aplicam ao projeto:**

Escalabilidade:

Consiste na capacidade do aplicativo de lidar e suportar o aumento da carga de trabalho sem sacrificar o desempenho. Um indicador que demonstra escalabilidade é a latência, que pode ser definida como o tempo que um sistema leva para responder uma solicitação do usuário. Se a carga de tráfego em um sistema aumentar e a latência permanecer a mesma, isso demonstra escalabilidade.

A escalabilidade está aplicada ao projeto na escolha do algoritmo de busca, haja vista que o algoritmo utilizado possui uma complexidade de tempo constante. Pois, um programa que possui complexidade $O(n^2)$ ou maior, onde n é o tamanho do conjunto de dados, possui uma dificuldade elevada para ser escalado. Uma vez que, à medida que a complexidade do algoritmo cresce, o desempenho se degrada exponencialmente.

Ademais, o uso de dois servidores simultaneamente, ao invés de um, é uma aplicação da escalabilidade horizontal, já que divide o trabalho para mais de um servidor.

Tolerância a falhas:

Consiste na capacidade do sistema fornecer seus serviços esperados mesmo na presença de falhas.

O projeto faz uso da tolerância, pois as instruções que estabelecem a comunicação entre cliente e servidor são inseridas em um bloco de código denominado try-catch, cuja finalidade é evitar com que falhas inesperadas sejam um impedimento para o funcionamento do sistema, além disso, o servidor A é capaz de buscar a substring escrita pelo cliente, mesmo se o servidor B estiver inoperante.

Porém, por conta da arquitetura utilizada, temos um problema que o servidor B não é funcional caso o servidor A esteja apresentando alguma falha, portanto, para uma implementação futura é interessante que o servidor B tenha um certo grau de independência.

3) **Vantagens e desvantagens de utilizar essa arquitetura:**

a) Vantagens:

i) - Distribuir os dados e o processamento em mais de um servidor, assim evitando a falha completa, caso um deles pare de funcionar e proporcione escalabilidade horizontal.

b) Desvantagem:

i) - O servidor B não funciona sem o servidor A, já que ele recebe a palavra a ser encontrada do servidor A e não do cliente.

4) **Arquitetura da solução:**

a) Diagrama:

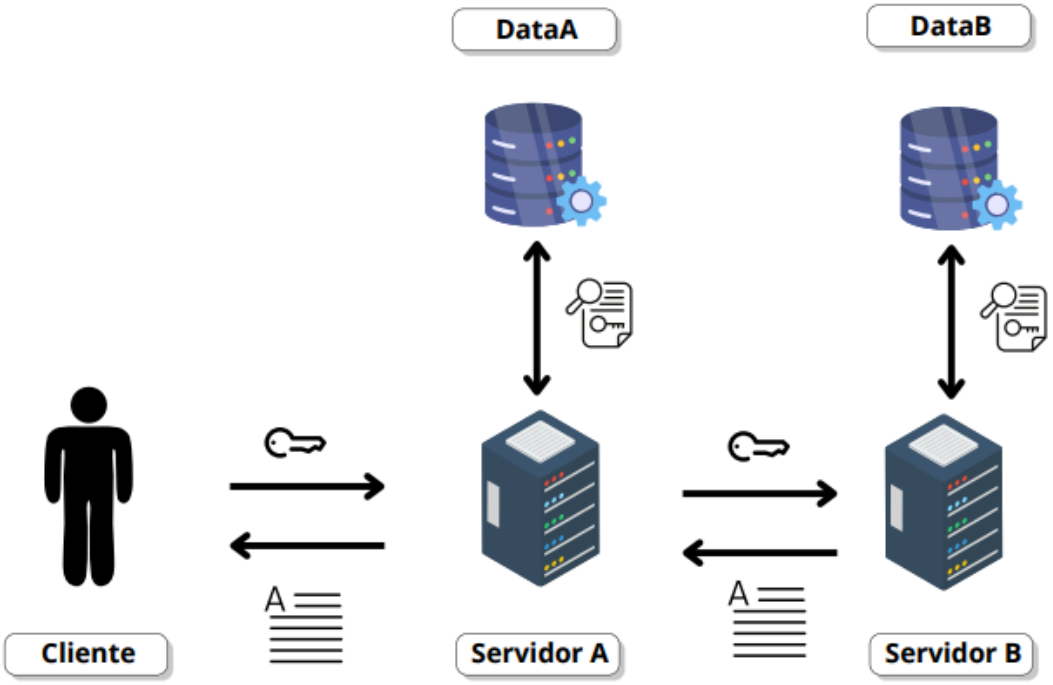

Diagrama de comunicação entre Cliente e Servidor

b) Formato do dado escolhido: O formato do dado trafegado entre o servidor A e B é por meio de String e do servidor B para o A e do A para o cliente é por meio de uma lista do tipo Linked List.

c) Explicação e justificativa do algoritmo de busca utilizado: O algoritmo de busca utilizado foi o KMP algorithm, pois o seu tempo de execução é de apenas O(m+n) e por que ele é um algoritmo utilizado para encontrar um padrão dentro de textos grandes de forma eficiente, pois evita realizar comparações redundantes.

Realizamos alguns testes comparando esse algoritmo com outros de tempo semelhante como o Z algorithm e a velocidade de processamento se mostrou 25% mais rápida, demonstrando assim a sua eficiência.

Referências bibliográficas:

BAL, Henri E.; STEINER, Jennifer G.; TANENBAUM, Andrew S. Programming languages for distributed computing systems. **ACM Computing Surveys (CSUR)**, v. 21, n. 3, p. 261-322, 1989.

AMAZON WEB SERVICES. *O que é computação distribuída?*. Disponível em: https://aws.amazon.com/pt/what-is/distributed-computing/ . Acesso em: 22 nov. 2024.

SIQUEIRA, Frank. *Fundamentos de sistemas distribuídos*. Disponível em: https://www.inf.ufsc.br/~frank.siqueira/INE5418/1.Fundamentos-Slides.pdf. Acesso em: 22 nov. 2024.

IBM. *What is distributed computing?*. Disponível em: https://www.ibm.com/docs/en/txseries/8.2?topic=overview-what-is-distributed-computing. Acesso em: 22 nov. 2024.

HILL, Mark D. What is scalability?. **ACM SIGARCH Computer Architecture News**, v. 18, n. 4, p. 18-21, 1990.

JALOTE, Pankaj. **Fault tolerance in distributed systems**. Prentice-Hall, Inc., 1994.

WEBER, Taisy Silva. Tolerância a falhas: conceitos e exemplos. **UFRGS,** 2018. Disponível em: https://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF. Acesso em: 24 nov. 2024.

STORM, Tania. Escalabilidade: Capítulo 3. **Medium,** 2022. Disponível em: https://medium.com/@tanstorm/cap%C3%ADtulo-3-escalabilidade-b2a86eb705b6. Acesso em: 23 nov. 2024.

GEEKSFORGEEKS. *KMP Algorithm for Pattern Searching*. Disponível em: https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/. Acesso em: 23 nov. 2024.