Universidade Católica de Brasília

Curso de Ciência da Computação

TRABALHO DE SOCKETS

Autor: Júlia Corrêa de Souza Oliveira – UC22200753
Professor: João Robson Santos Martins

Computação distribuída refere-se ao campo da ciência da computação que estuda sistemas compostos por múltiplos computadores interconectados, que colaboram para resolver problemas ou processar tarefas. Esses sistemas permitem que as operações sejam realizadas de maneira paralela, e buscam otimizar o desempenho em problemas de larga escala por meio do compartilhamento de recursos, permitindo maior eficiência na resolução de tarefas complexas (HAJIBABA; GORGIN, 2014). Isso faz com que múltiplas máquinas, que podem estar localizadas em diferentes pontos geográficos, compartilhem recursos computacionais e trabalhem juntas como um sistema único.

Em sistemas distribuídos, os diferentes componentes geralmente se comunicam através de redes de computadores, utilizando protocolos como o TCP/IP. A computação distribuída é amplamente utilizada em sistemas que exigem alta disponibilidade, como bancos de dados distribuídos, serviços de nuvem e também em processos que envolvem a busca de grandes volumes de dados.

Escalabilidade é a capacidade de um sistema de se adaptar ao aumento de carga, seja através da adição de mais recursos (escala vertical) ou da adição de mais unidades de processamento (escala horizontal). Uma noção intuitiva de escalabilidade é que ela implica uma comparação favorável entre uma versão maior de algum sistema paralelo com uma versão sequencial desse mesmo sistema ou uma máquina paralela teórica. No contexto do projeto de busca distribuída, a escalabilidade é crucial, pois a separação dos dados entre dois servidores (A e B) permite processar as buscas de forma mais eficiente, já que cada servidor lida com uma parte do banco de dados, reduzindo o tempo de resposta.

A tolerância a falhas é essencial para o funcionamento eficaz de aplicações críticas na Internet, em que interrupções podem resultar em prejuízos significativos. Sistemas com essa característica conseguem continuar operando mesmo na presença de falhas de componentes, o que os torna indispensáveis em ambientes de alta criticidade (GORENDER; MACÊDO, 2002). Em sistemas distribuídos, isso é alcançado por meio de redundância, replicação de dados e protocolos de recuperação que asseguram que, mesmo em caso de falhas de servidores ou de comunicação, o sistema como um todo permaneça operacional.

No projeto de busca distribuída, a tolerância a falhas e a escalabilidade são importantes para garantir que, mesmo se um servidor falhar, o sistema continue funcionando, e que o aumento no número de clientes ou dados não prejudique o desempenho do sistema.

Vantagens:

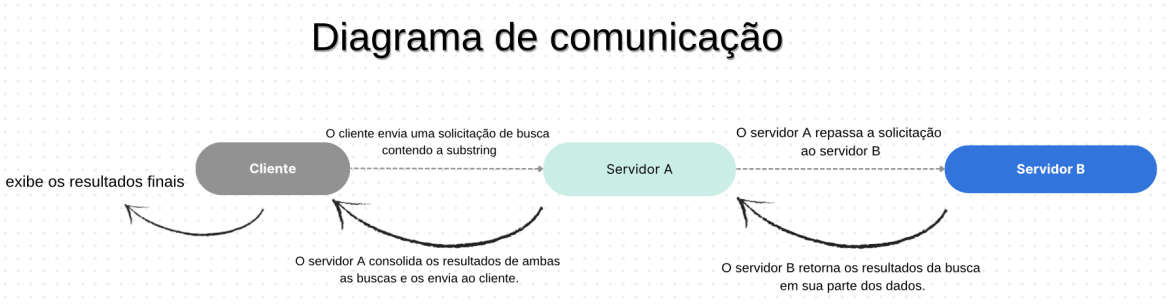
- Alto desempenho: Ao distribuir a carga de trabalho entre múltiplos servidores, o sistema pode reduzir o tempo total necessário para a execução das operações,

especialmente em aplicações que envolvem grandes volumes de dados ou alta demanda por recursos computacionais.

- Escalabilidade: Sistemas distribuídos podem ser facilmente expandidos, adicionando novos nós para lidar com maiores volumes de dados ou para aumentar a capacidade de processamento. Essa característica torna o sistema adaptável a cenários de crescimento de demanda, garantindo desempenho consistente mesmo em condições adversas.
- Confiabilidade: Um sistema distribuído pode continuar operando, mesmo que um ou mais nós falhem. Essa propriedade é especialmente útil em cenários críticos, onde a interrupção total do sistema deve ser evitada.

Desvantagens:

- Complexidade de implementação: O desenvolvimento, a configuração e a manutenção de sistemas distribuídos são significativamente mais desafiadores em comparação a sistemas centralizados. Isso ocorre devido à necessidade de lidar com aspectos como sincronização, comunicação entre nós, e gerenciamento de falhas.
- Latência de Rede: Como a comunicação entre os nós ocorre por meio de redes, podem haver atrasos significativos na troca de mensagens, especialmente em redes de baixa qualidade ou com alta latência. Esse fator pode impactar o desempenho geral do sistema, dependendo do volume de dados trafegados e da frequência das comunicações.



Descrição: O cliente se conecta ao servidor A para enviar a substring a ser buscada. O servidor A processa sua parte dos dados e repassa a solicitação para o servidor B. Ambos realizam a busca localmente e o servidor A consolida os resultados, enviando-os de volta ao cliente. Por fim, o cliente exibe os resultados finais.

O formato de dado escolhido para o tráfego de informações entre os servidores e o cliente é o JSON (JavaScript Object Notation). Esse formato foi selecionado devido à sua simplicidade, leveza e fácil manipulação em Java. O uso do JSON facilita a comunicação

estruturada e eficiente entre os componentes do sistema, garantindo que as informações sejam transmitidas de forma clara e facilmente processada. A biblioteca JSON possibilita a serialização de estruturas JavaScript, facilitando seu armazenamento e troca de dados, sendo amplamente utilizada em aplicações web devido à sua eficiência e simplicidade (SMITH, 2020).

O algoritmo de busca escolhido para o projeto é baseado na busca linear via (‘contains’), uma técnica simples e direta de pesquisa em listas de dados. A busca linear foi escolhida por conta da sua simplicidade de implementação e adequação para o volume de dados do projeto, considerando que os dados estão estruturados em um formato de lista (títulos de artigos) e a busca será realizada em strings (substrings nos títulos ou introduções dos artigos).

O algoritmo de busca linear percorre cada elemento de uma lista ou coleção de dados, comparando-o com a substring fornecida. Caso o item da lista contenha a substring, ele é adicionado aos resultados da busca. O processo continua até que todos os itens tenham sido verificados.

No contexto desse projeto, o algoritmo realiza a busca nos títulos ou nas introduções dos artigos, procurando pela substring fornecida pelo Cliente. Essa busca acontece de forma sequencial, comparando cada título com a substring, até que todos os dados sejam verificados.

REFERÊNCIAS BIBLIOGRÁFICAS:

1. TANENBAUM, Andrew S.; VAN STEEN, Maarten. *Distributed Systems: Principles and Paradigms*. 2. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.
2. SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. *Operating System Concepts*. 9. ed. Hoboken, NJ: John Wiley & Sons, 2012.
3. HAJIBABA, Majid; GORGIN, Saeid. A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing. *Journal of Computing and Information Technology*, v. 22, n. 2, p. 69-84, 2014.
4. GORENDER, Sérgio; MACÊDO, Ricardo José de Araújo. Um modelo para tolerância a falhas em sistemas distribuídos com QoS. In: *Anais do Simpósio Brasileiro de Redes de Computadores – SBRC 2002*. [S.l.]: Sociedade Brasileira de Computação, 2002. p. 277-292.
5. SMITH, Ben. *JSON básico: conheça o formato de dados preferido da web*. São Paulo: Novatec Editora, 2020.