Universidade Católica de Brasília

Pró-Reitoria Acadêmica
Curso de Ciência da Computação

AT2/N1 - Relatório

Autores: Matheus Anderson de Sousa Gomes,
Rafael Alves de Araujo,
Scarllet Gomes Rodrigues,
Vinícius Baltazar Barros Santos,
Wallacy William dos Santos Oliveira

Orientador: João Robson

Brasília – DF
2024

Sumário

○ O que é computação distribuída:

A computação distribuída envolve execução de tarefas pela conexão, em rede, de dois ou mais computadores independentes trabalhando em conjunto, como se fossem um único sistema mais poderoso. Essa colaboração permite resolver problemas complexos que demandariam muito tempo ou recursos de um único computador, pois os computadores são capazes de compartilhar recursos, como poder de processamento e armazenamento.

○ Os conceitos de escalabilidade e tolerância a falhas e como eles se aplicam ao projeto:

Escalabilidade é a propriedade de um sistema que lhe permite aumentar seu tamanho ou volume de processamento sem comprometer sua performance. A capacidade de adicionar mais recursos (nós) ao sistema permite lidar com cargas de trabalho crescentes de forma eficiente. Enquanto tolerância a falhas é a habilidade de um sistema continuar funcionando mesmo quando um ou mais dos seus componentes falham, garantindo a continuidade do serviço.

Relação entre a escalabilidade e tolerância a falhas:

- Complementaridade**:** Ambos os conceitos são complementares e trabalham juntos para garantir a alta disponibilidade e o desempenho de um sistema.
- Escalabilidade e replicação: Para alcançar a escalabilidade, muitas vezes é necessário replicar componentes do sistema em diferentes servidores. Essa replicação também contribui para a tolerância a falhas, pois se um componente falhar, outros podem assumir sua função.
- Tolerância a falhas e escalabilidade: Sistemas tolerantes a falhas são mais fáceis de escalar, pois a redundância de componentes permite adicionar novos recursos sem interromper o serviço.

No projeto deste relatório a escalabilidade é usada através da divisão de tarefas entre os servidores A e B e o sistema é capaz de fazer as duas buscas simultaneamente. Caso essa busca fosse realizada por apenas um servidor, ele teria muitos dados para analisar sozinho, o que aumentaria o tempo de execução e demandaria mais processamento de apenas uma máquina. Já a tolerância a falhas se dá na hipótese de o servidor B falhar. Nesse caso o servidor A ainda é capaz de realizar a sua busca e fornecer a sua resposta ao cliente. O que garante que este receba uma resposta, mesmo que parcial, ao invés de uma falha no sistema.

○ Vantagens e desvantagens de utilizar essa arquitetura:

Referente as vantagens de utilizar esta arquitetura além das já mencionadas no tópico anterior que são a escalabilidade e tolerância a falhas temos também a alta disponibilidade onde a distribuição de dados e processos em múltiplos nós torna o sistema mais resistente a falhas, pois a perda de um único nó geralmente não compromete todo o sistema, melhor desempenho pois a distribuição de tarefas entre múltiplos processadores pode acelerar o processamento de grandes volumes de dados, a flexibilidade que permite a utilização de diferentes tipos de hardware e software, adaptando-se a diversas necessidades e a expansão facilitada, pois é possível adicionar novos servidores para lidar com mais dados, ou para serem cópias de segurança caso algum falhe.

Desvantagens:

- Complexidade: O design, implementação e gerenciamento de sistemas distribuídos são mais complexos do que sistemas centralizados, exigindo um maior conhecimento técnico.
- Gerenciamento: A coordenação de múltiplos nós e a garantia da consistência dos dados podem ser desafiadoras.
- Comunicação: A comunicação entre os nós pode ser lenta e sujeita a falhas, impactando o desempenho do sistema.

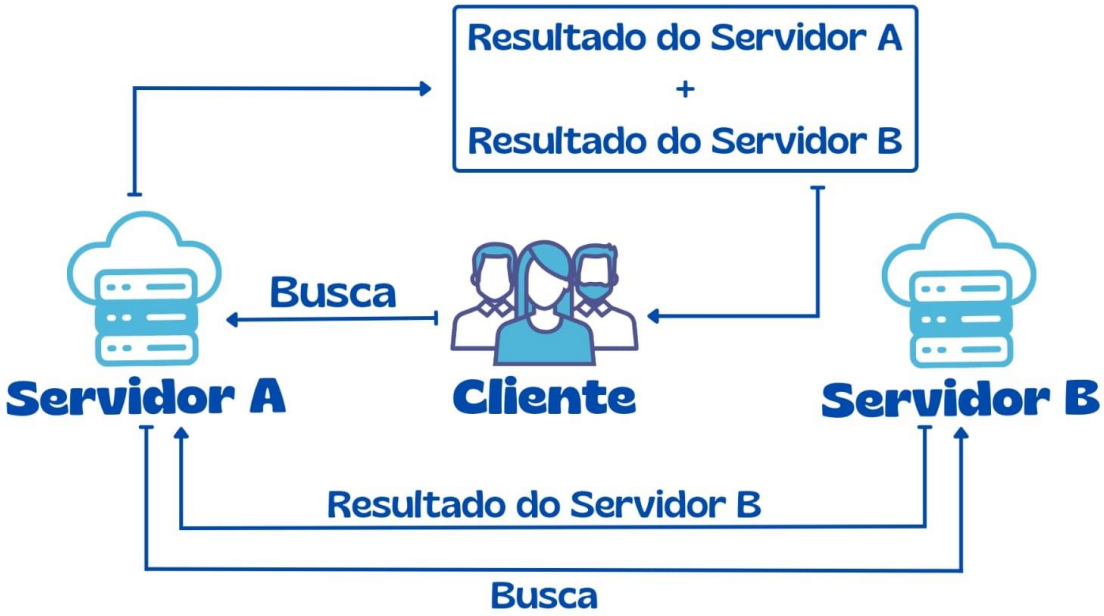
- Segurança: Aumentar a superfície de ataque com múltiplos nós pode comprometer a segurança do sistema.
- Consistência de dados: Garantir a consistência dos dados em um ambiente distribuído pode ser complexo, especialmente em caso de falhas ou atualizações simultâneas.
- Dependência de rede: O desempenho do sistema está diretamente ligado à qualidade da rede de comunicação

○ Arquitetura da solução:

A solução é composta por um cliente e dois servidores. O cliente faz buscas por termos específicos, e os servidores processam essas buscas e retornam os resultados.
Comunicação:

- O cliente envia o termo de busca para o servidor A;
- O servidor A processa a busca nos próprios dados e, simultaneamente, envia a busca para o servidor B;
- O servidor B processa a busca nos seus dados e devolve os resultados ao servidor A;
- O servidor A combina os resultados locais e remotos e os envia ao cliente.

○ Diagrama explicando comunicação durante uma busca entre cliente e servidores:



○ Formato do dado (escolhido pelo grupo) trafegado entre servidor A e B e servidor A e cliente:

No projeto o formato de dados trafegados são *strings* simples, que correspondem ao termo de buscas e os resultados formatados. A decisão de utilizar *strings* simples como formato de dados trafegados entre os servidores e entre o servidor A e o cliente foi baseada em diversos fatores relacionados à simplicidade, eficiência e clareza. Elas são fáceis de implementar e manipular, não é necessário criar um método complexo para empacotar e desempacotar os dados e podem ser enviadas com a sua formatação. *Strings* também são fáceis de serem

processadas em comparação com JSON e XML por exemplo, são mais leves para trafegar em rede, além de ser mais fácil perceber erros no desenvolvimento com o uso delas.

○ A**lgoritmo de busca utilizado:**

O algoritmo utilizado no sistema é o *Naive String Matching*. Ele é simples e direto, adequado para sistemas com uma quantidade moderada de dados.

Como Funciona:

- O algoritmo percorre cada posição do texto (como os títulos dos artigos) e verifica se o termo de busca está presente.
- Se encontrar uma sequência correspondente, considera que o termo foi encontrado.

Justificativa**:**

- Facilidade de Implementação: É fácil de entender e programar.
- Bom para Pequenos Dados: Embora tenha um custo maior em dados muito grandes, funciona bem para o volume de dados deste projeto.
- Flexível: Não precisa de configurações ou pré-processamento dos dados.

○ **Referências**:

- IBM. Computação distribuída. 2024. Disponível em: https://www.ibm.com/docs/pt-br/cics-ts/6.x?topic=interfaces-distributed-computing . Acesso em: 19 nov. 2024.
- AWS. O que é computação distribuída? 2024. Disponível em: https://aws.amazon.com/pt/what-is/distributed-computing/ . Acesso em: 19 nov. 2024.
- FORD, Neal. *Arquitetura de Software: as partes difíceis: análises modernas de trade-off para arquiteturas distribuídas*. Rio de Janeiro: Alta Books, 2024. Disponível em: https://ucb.pergamum.com.br/acervo/5073161. Acesso em: 20 de nov. 2024.
- TANENBAUM, Andrew S. S*istemas Distribuídos Princípios e Paradigmas*. 2. ed. São Paulo: Pearson Education, 2007. Acesso em: 22 nov. 2024.