RELATÓRIO TÉCNICO: SISTEMA DE BUSCA DISTRIBUÍDA UTILIZANDO SOCKETS EM JAVA

1. EXPLICAÇÃO TEÓRICA

O que é Computação Distribuída

A computação distribuída é um paradigma de sistemas computacionais que utiliza múltiplos computadores independentes, interconectados por uma rede, para colaborar na execução de tarefas. Em vez de depender de um único sistema centralizado, a computação distribuída busca distribuir a carga de trabalho, aumentando a eficiência, escalabilidade e tolerância a falhas. Cada máquina no sistema é chamada de nó, podendo atuar como servidor, cliente ou ambos, dependendo do contexto.

Segundo Tanenbaum (2007), "um sistema distribuído é uma coleção de computadores independentes que se apresentam ao usuário como um sistema único e coerente."

Escalabilidade e Tolerância a Falhas

Escalabilidade é a capacidade de um sistema continuar eficiente à medida que cresce em tamanho ou em volume de trabalho. No projeto desenvolvido, a arquitetura distribuída permite que, se o volume de dados crescer, novos servidores de busca possam ser adicionados com facilidade, dividindo a carga e mantendo a performance.

Tolerância a falhas é a capacidade de um sistema continuar funcionando, mesmo que partes dele falhem. No contexto deste projeto, caso um dos servidores de busca (B ou C) fique indisponível, o Servidor A ainda pode coletar resultados do outro servidor e retornar uma busca parcial ao cliente, mantendo certa funcionalidade.

Vantagens e Desvantagens da Arquitetura Distribuída

Vantagens:
- Melhor aproveitamento de recursos computacionais.
- Facilidade de expansão (escalabilidade).
- Tolerância a falhas localizada.
- Redução de gargalos computacionais.

Desvantagens:

- Maior complexidade de implementação e depuração.
- Necessidade de sincronização e comunicação eficiente entre os nós.
- Potencial para inconsistências em caso de falhas parciais ou latências de rede.

2. ARQUITETURA DA SOLUÇÃO

Diagrama de Comunicação

Cliente <--> Servidor A <--> Servidor B

\--> Servidor C

Cliente <--> Servidor A <--> Servidor B

\--> Servidor C

1. O Cliente envia uma string de busca ao Servidor A.
2. O Servidor A repassa a string para os Servidores B e C.
3. Cada um realiza a busca em sua própria base de dados (JSON).
4. Os resultados são enviados de volta ao Servidor A.
5. O Servidor A consolida e responde ao Cliente.

Formato dos Dados Trafegados

- Entre Cliente e Servidor A:
  - Texto (string) enviado via BufferedReader e PrintWriter.
  - Resposta é uma lista de objetos Artigo serializados via ObjectOutputStream.

- Entre Servidor A e B/C:
  - Solicitação: string simples.
  - Resposta: List<Artigo> via ObjectOutputStream.

Justificativa do Algoritmo de Busca

Foi utilizado o algoritmo de busca ingênua (Naive Search), baseado no método String.contains(). Ele foi escolhido por:

- Sua implementação simples e direta;
- Baixo custo computacional para volumes moderados de dados;
- Clareza para manutenção e leitura do código;
- Alinhamento com o escopo acadêmico do projeto.

Caso o sistema venha a escalar para grandes volumes, pode-se substituir por algoritmos mais eficientes como KMP ou Boyer-Moore.

3. REFERÊNCIAS BIBLIOGRÁFICAS

- Tanenbaum, A. S., & van Steen, M. (2007). Distributed Systems: Principles and Paradigms. Prentice-Hall.
- GeeksForGeeks. (s.d.). Pattern Searching Algorithms. https://www.geeksforgeeks.org/pattern-searching/