Universidade Católica de Brasília

Pró-Reitoria Acadêmica
Curso de Ciência da Computação
Programação Concorrente e Distribuída

Sistema de Busca Distribuído com Sockets em Java

Autores: Breno Silva
Célio Silva
Kelvin Rodrigues
Lara Dayrell
Orientador: João Robson

Brasília - DF
2025

Sistema de Busca Distribuído com Sockets em Java

1. Introdução e Fundamentação Teórica

Este projeto tem como objetivo o desenvolvimento de um sistema de busca distribuído para artigos científicos, com base no conjunto de dados do arXiv, utilizando a linguagem Java 17 e comunicação via sockets TCP. O sistema é composto por três servidores (A, B e C), sendo dois deles responsáveis pela realização de buscas locais e um pela coordenação das requisições.

- **Computação Distribuída:**

  Trata-se de um paradigma onde componentes de software em diferentes computadores se comunicam e coordenam suas ações por meio de uma rede. Segundo Tanenbaum e Van Steen (2007), sistemas distribuídos são caracterizados por funcionarem como um sistema único coeso, mesmo sendo compostos por diversos componentes autônomos conectados por uma rede.

- **Escalabilidade e Tolerância a Falhas:**

  A arquitetura distribuída permite que o sistema escale horizontalmente, dividindo a carga entre múltiplos servidores. A tolerância a falhas pode ser aprimorada com redundância e monitoramento, mas não foi implementada diretamente neste projeto.

  A aplicação no projeto se dá por meio de divisão base de dados entre dois servidores (B e C). Cada servidor realiza buscas em paralelo e de forma independente. Essa abordagem permite que, se os dados crescerem, mais servidores possam ser adicionados para manter o desempenho — ou seja, escala horizontal.

- **Vantagens:**

  - Melhor desempenho por distribuição da carga.

  - Possibilidade de paralelismo.

  - Maior modularidade.

- **Desvantagens:**

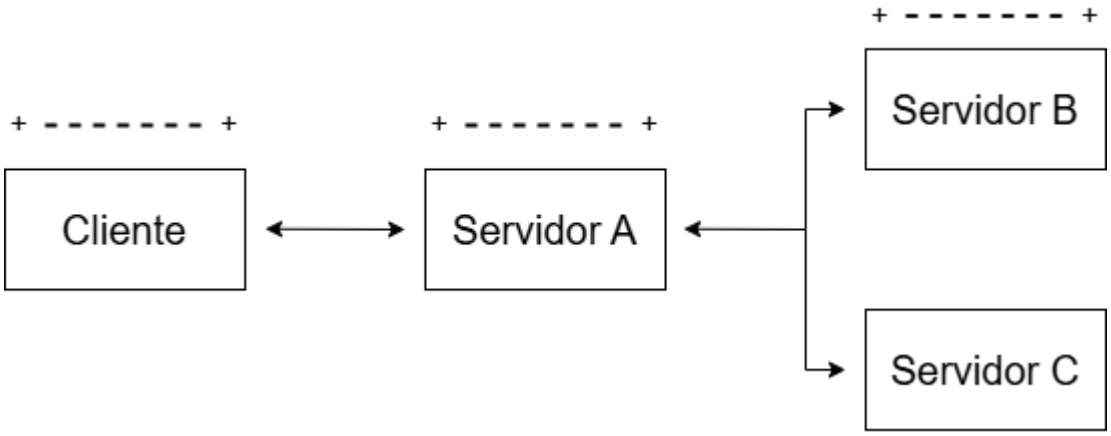
  - Complexidade de implementação.

  - Dificuldade de depuração.

  - Necessidade de sincronização entre componentes.

2. Arquitetura da Solução

2.1 Diagrama de Comunicação



2.2 Formato dos Dados

- **Requisição do Cliente:**

{ "query": "termo de busca" }

- Resposta dos Servidores B/C para A:

[{ "onde": "TÍTULO", "texto": "..." }, { "onde": "INTRODUÇÃO", "texto": "..." }]

- Resposta de A para o Cliente: concatenação das respostas dos servidores B e C.

2.3 Justificativa do Algoritmo de Busca

O algoritmo adotado foi a busca por substring com destaque, utilizando String.contains() em combinação com replaceAll e expressões regulares ANSI para destacar visualmente a string no terminal. Justifica-se por:

- Ser eficiente para volumes pequenos/médios.
- Não requer bibliotecas externas.
- Fácil manutenção e legível.

3. Implementação e Estrutura do Projeto

- Cliente: conecta-se ao servidor A, envia a query, exibe resultados formatados.
- Servidor A: repassa a query para B e C, agrega as respostas, e retorna ao cliente.
- Servidor B e C: realizam buscas locais em seus arquivos JSON.

- **Organização do Código:**
  - Divisão clara em pacotes: cliente, servidorA, servidorB, servidorC.
  - Código modularizado com métodos reutilizáveis.
  - Nomes de variáveis autoexplicativos e indentado.

4. Referências Bibliográficas

- TANENBAUM, A. S.; VAN STEEN, M. Distributed Systems: Principles and Paradigms. Prentice Hall, 2007.
- https://www.geeksforgeeks.org/pattern-searching/
- https://www.devmedia.com.br/trabalhando-com-json-em-java-o-pacote-org-json/25480
- https://arxiv.org/

5. Anexo

Aqui listamos os códigos referente ao cliente e aos servidores.

```
package cliente;

import java.io.*;
import java.net.*;
import org.json.JSONObject;
import org.json.JSONArray;
public class Cliente {
    public static void main(String[] args) {
        // Endereço do servidor e porta
        String host = "localhost";
        int porta = 5000;
        try (
            // Criação do socket para se conectar ao servidor
            Socket socket = new Socket(host, porta);

            // Lê dados recebidos do servidor
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            // Envia dados para o servidor
            BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

            // Lê dados do teclado (entrada do usuário)
            BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in));
        ) {
            String busca = "";

            // Validação da entrada do usuário
            while (true) {
                System.out.print("Digite a substring de busca: ");
                busca = teclado.readLine();

                // Valida se a busca está vazia ou tem menos de 3
caracteres
                if (busca == null || busca.trim().isEmpty()) {
                    System.out.println("A busca não pode estar vazia.
Tente novamente.");
                } else if (busca.trim().length() < 3) {
                    System.out.println("Digite pelo menos 3 caracteres
para uma busca significativa.");
```

```
                } else {
                    break; // Entrada válida, sai do loop
                }
            }

            // Cria um objeto JSON com a substring de busca
            JSONObject json = new JSONObject();
            json.put("query", busca);

            // Envia o JSON ao servidor
            out.write(json.toString());
            out.newLine(); // Necessário para indicar fim da linha
            out.flush();   // Força o envio dos dados

            // Lê a resposta do servidor
            String resposta = in.readLine();
            System.out.println("\nResultados encontrados:\n");

            // Converte a resposta em um array JSON
            JSONArray resultadoJson = new JSONArray(resposta);

            // Verifica se há resultados
            if (resultadoJson.length() == 0) {
                System.out.println("Nenhum resultado encontrado para: \""
+ busca + "\"\n");
            } else {
                // Itera sobre os resultados recebidos
                for (int i = 0; i < resultadoJson.length(); i++) {
                    JSONObject obj = resultadoJson.getJSONObject(i);

                    // Obtém os campos "onde" e "texto", com valores
padrão caso não existam
                    String onde = obj.optString("onde", "DESCONHECIDO");
                    String texto = obj.optString("texto", "");

                    // Exibe os resultados no console
                    System.out.println("Local encontrado: " + onde);
                    System.out.println("Trecho: " + texto);
                    System.out.println("----------------------------------
------------------\n");
                }
            }

        } catch (IOException e) {
            // Trata exceções de entrada/saída
```

```
            e.printStackTrace();
        }
    }
}
```

```
package servidorA;

import java.io.*;
import java.net.*;
import org.json.JSONArray;
import org.json.JSONObject;

public class ServidorA {
    public static void main(String[] args) throws IOException {
        int porta = 5000;
        // Cria um servidor socket que escuta na porta 5000
        ServerSocket serverSocket = new ServerSocket(porta);
        System.out.println("Servidor A aguardando conexões...");

        // Loop infinito para aceitar múltiplas conexões de clientes
        while (true) {
            Socket clienteSocket = serverSocket.accept(); // Aceita
conexão de um cliente
            // Cria uma nova thread para tratar a requisição do cliente
            new Thread(() -> tratarCliente(clienteSocket)).start();
        }
    }

    // Método para tratar a comunicação com o cliente
    public static void tratarCliente(Socket clienteSocket) {
        try (
            // Prepara para receber e enviar dados ao cliente
            BufferedReader in = new BufferedReader(new
InputStreamReader(clienteSocket.getInputStream()));
            BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(clienteSocket.getOutputStream()));
        ) {
            // Lê a entrada enviada pelo cliente (em formato JSON)
            String input = in.readLine();
            JSONObject json = new JSONObject(input);
            String query = json.getString("query"); // Extrai a string de
busca

            // Realiza buscas nos servidores B e C e combina os resultados
```

```
            JSONArray resultados = combinarResultados(
                buscarEmServidor("localhost", 6000, query),  // busca no
servidor B
                buscarEmServidor("localhost", 7000, query)   // busca no
servidor C
            );

            // Envia os resultados combinados de volta ao cliente
            out.write(resultados.toString());
            out.newLine();
            out.flush();
        } catch (IOException e) {
            e.printStackTrace(); // Em caso de erro de comunicação
        }
    }

    // Método que envia uma consulta para um servidor (B ou C) e recebe os
resultados
    public static JSONArray buscarEmServidor(String host, int porta,
String query) {
        JSONArray resultados = new JSONArray(); // Inicializa array de
resultados
        try (
            // Abre conexão com o servidor B ou C
            Socket socket = new Socket(host, porta);
            BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        ) {
            // Envia o JSON com a busca
            JSONObject json = new JSONObject();
            json.put("query", query);
            out.write(json.toString());
            out.newLine();
            out.flush();

            // Recebe a resposta do servidor (também em formato JSON)
            String resposta = in.readLine();
            resultados = new JSONArray(resposta); // Converte resposta em
JSONArray
        } catch (IOException e) {
            e.printStackTrace(); // Em caso de falha na conexão com B ou C
        }
        return resultados; // Retorna os resultados obtidos
```

```
    }

    // Junta os resultados dos servidores B e C em um único JSONArray
    public static JSONArray combinarResultados(JSONArray a, JSONArray b) {
        JSONArray combinados = new JSONArray(); // Novo array para
armazenar todos os resultados

        // Adiciona os elementos do primeiro array
        for (int i = 0; i < a.length(); i++) {
            combinados.put(a.get(i));
        }

        // Adiciona os elementos do segundo array
        for (int i = 0; i < b.length(); i++) {
            combinados.put(b.get(i));
        }

        return combinados; // Retorna todos os resultados combinados
    }
}
```

```
package servidorB;

import java.io.*;
import java.net.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import org.json.JSONObject;
import org.json.JSONArray;
import java.util.regex.Pattern;

public class ServidorB {
    public static void main(String[] args) throws IOException {
        int porta = 6000;
        // Cria o servidor socket que escuta na porta 6000
        ServerSocket serverSocket = new ServerSocket(porta);
        System.out.println("Servidor B aguardando...");

        // Loop infinito para aceitar múltiplas conexões
        while (true) {
            Socket socket = serverSocket.accept(); // Aceita conexão de um
cliente (Servidor A)
            // Cria uma thread para tratar a busca de forma paralela
            new Thread(() -> tratarBusca(socket)).start();
```

```
        }
    }

    // Método para tratar a busca recebida
    public static void tratarBusca(Socket socket) {
        try (
            // Fluxo de entrada e saída de dados com o cliente
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
        ) {
            // Lê a entrada do cliente (Servidor A)
            String entrada = in.readLine();
            JSONObject req = new JSONObject(entrada);
            String query = req.getString("query").toLowerCase(); //
Converte a busca para minúsculo

            // Carrega os dados locais do servidor B a partir de um
arquivo JSON
            JSONArray dados = new
JSONArray(Files.readString(Paths.get("dados/dados_servidor_b.json")));
            JSONArray resultados = new JSONArray(); // Armazena os
resultados encontrados

            // Percorre cada artigo do arquivo JSON
            for (int i = 0; i < dados.length(); i++) {
                JSONObject artigo = dados.getJSONObject(i);
                String titulo = artigo.optString("title",
"").toLowerCase();       // Título do artigo
                String introducao = artigo.optString("abstract",
"").toLowerCase(); // Introdução/resumo do artigo

                // Verifica se a query está presente no título
                if (titulo.contains(query)) {
                    JSONObject resultado = new JSONObject();
                    resultado.put("onde", "TÍTULO");
                    resultado.put("texto", destaque(titulo, query)); //
Destaca a parte correspondente
                    resultados.put(resultado);
                }

                // Verifica se a query está presente na introdução
                if (introducao.contains(query)) {
                    JSONObject resultado = new JSONObject();
```

```
                    resultado.put("onde", "INTRODUÇÃO");
                    resultado.put("texto", destaque(introducao, query));
// Destaca a parte correspondente
                    resultados.put(resultado);
                }
            }

            // Envia os resultados encontrados para o cliente (Servidor A)
            out.write(resultados.toString());
            out.newLine();
            out.flush();
        } catch (Exception e) {
            // Exibe erro no console em caso de falha
            e.printStackTrace();
        }
    }

    // Função que destaca a palavra encontrada usando ANSI (cor amarela no
terminal)
    public static String destaque(String texto, String query) {
        String ANSI_YELLOW = "\u001B[33m";
        String ANSI_RESET = "\u001B[0m";
        // Substitui ocorrências da query pela mesma palavra destacada em
amarelo
        return texto.replaceAll("(?i)(" + Pattern.quote(query) + ")",
ANSI_YELLOW + "$1" + ANSI_RESET);
    }

    /* Alternativa para destacar a palavra com markdown, útil se for
exibido em um ambiente web:
    public static String destaque(String texto, String query) {
        return texto.replaceAll("(?i)(" + Pattern.quote(query) + ")",
"**$1**");
    }
    */
}
```