Universidade Católica de Brasília

Luiz Henrique Alves Rodrigues (UC23101594)

Relatório: Conceitos e aspectos relacionados à Computação Distribuída.

Brasília, 15 de Junho de 2025

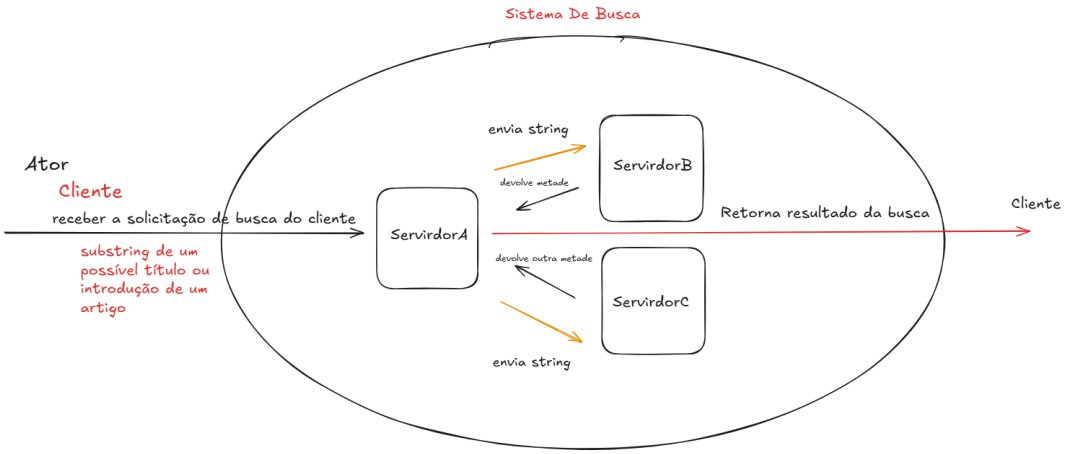
O conceito de Computação distribuída envolve a execução de tarefas em diferentes processos para atingir um objetivo em comum. Nesse sentido, por "diferentes processos" pode-se mencionar o cenário onde tarefas são executadas em máquinas dispersas geograficamente, o que representa um exemplo de computação em grid, onde um objetivo final é estabelecido. Geralmente, trata-se de tarefas cujo resultado final seria quase impossível de alcançar por meio de apenas um processo, então, ele é distribuído em partes para que seja atingido. Diante desse contexto, a computação distribuída tornou-se tornou viável graças à evolução das redes, sendo de extrema importância a existência de redes com baixa latência e que suportam múltiplos nós. Assim, esse mecanismo é fundamental para a computação científica, onde muitas vezes o objetivo final envolve problemas altamente complexos.

A partir do entendimento de que um sistema distribuído tende a ter mais requisitos ao decorrer de sua vida, faz-se necessário que seja escalável. Na prática, o conceito de escalabilidade se aplica ao presente projeto ao idealizar um cenário em que seria adicionado outro servidor para receber mais dados no formato JSON. Sobre essa ótica, o servidor principal, responsável pela comunicação com o cliente e com os demais servidores, é o Servidor A. O que o torna escalável é o fato de serem necessárias apenas pequenas alterações para que ele também receba dados do novo servidor, considerando que o algoritmo desenvolvido é compatível com qualquer JSON no mesmo formato.

Quanto à tolerância a falhas, trata-se da capacidade do sistema continuar oferecendo sua funcionalidade principal mesmo que alguma parte do mesmo apresente falhas ou pare de funcionar. Pode-se observar, a partir do presente projeto autoral, um exemplo desse aspecto ao tentar conectar-se a uma porta inexistente. Nesse caso, o sistema não quebra: ele simplesmente retorna os dados do servidor que foi encontrado com sucesso.

A vantagem de adotar essa arquitetura é que pode-se dividir a busca entre diferentes repositórios de artigos e executá-la simultaneamente, com o objetivo final de reunir os artigos que contém a substring enviada. A desvantagem, nesse caso, está na complexidade de estruturar um esquema em que o cliente envie a substring para o Servidor A, que, por sua vez, repasse essa informação para os demais servidores realizarem suas buscas, e então, possa receber os resultados e os retornar ao cliente. Dessa forma, essa cascata de comunicação pode gerar confusão para quem analisa o código, especialmente se ele não estiver bem documentado ou modularizado.

Imagem 1: Diagrama Sistema de Busca



Elaboração autoral

Para este cenário, os dados são trafegados da seguinte maneira: o cliente insere uma substring (ou palavra-chave) que deseja buscar, e essa informação é enviada para o Servidor A em formato String. O Servidor A, então, encaminha essa mesma String para os outros servidores, como o ServidorB e o ServidorC. Cada um desses servidores recebe a substring e a utiliza em um método interno chamado montarLista, que também recebe o caminho de um arquivo JSON como parâmetro. Dentro desse método, o conteúdo do arquivo JSON é lido e convertido para uma lista de objetos Java do tipo ArtigoServidorDTO, utilizando a biblioteca Jackson com o método mapper.readValue. Ademais, a classe ArtigoServidorDTO possui os atributos title, abstractText e label e cada entrada do JSON se transforma em um objeto dessa classe.

Depois que os objetos são criados, eles podem ser filtrados conforme a substring recebida, verificando se essa substring aparece, por exemplo, no título ou no resumo do artigo. Em seguida, os resultados encontrados são convertidos novamente para String por meio do método toString, que foi sobrescrito na própria classe ArtigoServidorDTO. Essas Strings são, então, devolvidas ao ServidorA, que as repassa de volta ao cliente como resposta final. Portanto, esse processo ilustra bem a troca de dados em um sistema distribuído que utiliza objetos DTO e conversão JSON para tratar informações de forma estruturada.

Diante tais perspectivas, o algoritmo adotado no projeto foi o de busca linear. Ao obter a lista gerada a partir dos arquivos JSON, foi utilizado uma verificação simples com o método contains, aplicada aos campos title, abstract e label. Caso a substring seja encontrada em qualquer um desses campos, o item correspondente é adicionado à lista listaDeVerificados, que é retornada ao final do processo. O método é considerado linear

porque percorre item por item da lista fornecida, verificando cada um deles. Isso caracteriza uma complexidade de tempo O(n), ou seja, se a lista tiver 60 mil itens, serão feitas 60 mil verificações.

Por fim, esse algoritmo foi adotado por ser o mais simples de implementar e explicar dentro do contexto do projeto. Logo, tendo em vista que a quantidade de dados não é excessivamente grande, a escolha da busca linear não representa um problema de desempenho relevante.

REFERÊNCIAS BIBLIOGRÁFICAS

AMAZON WEB SERVICES**. O que é computação distribuída?** Disponível em:< https://aws.amazon.com/pt/what-is/distributed-computing/>. Acesso em: 15 jun. 2025.

KAHANWAL, Brijender. SINGH, T. P. **The Distributed Computing Paradigms**: P2P, Grid, Cluster, Cloud, and Jungle. International Journal of Latest Research in Science and Technology, v. 1, n. 2, p. 183–187, 2012. Disponível em:<https://doi.org/10.48550/arXiv.1311.3070>. Acesso em: 15 jun. 2025.

BHARGAVA, Aditya Y. **Entendendo algoritmos**: um guia ilustrado para programadores e outros curiosos. São Paulo: Novatec, 2017.