UNIVERSIDADE CATÓLICA DE BRASÍLIA

CIÊNCIA DA COMPUTAÇÃO

Arthur Lopes Moura - UC20190001666

Pedro Xavier Lodonio - UC21200125

Ygor Machado Gonçalves de Oliveira - UC18200363

Relatório da Atividade Prática Coletiva N2

BRASÍLIA-DF

2024

Arthur Lopes Moura

Pedro Xavier Lodonio

Ygor Machado Gonçalves de Oliveira

Relatório da Atividade Prática Coletiva N2

Relatório apresentado à matéria de Programação Concorrente e Distribuída, como parte dos requisitos necessários à obtenção da N2.
Orientador: João Robson

BRASÍLIA-DF

2024

Lista de Ilustrações

Figura 1 – Diagrama explicando comunicação durante uma busca entre cliente e servidores 9

Sumário

1 INTRODUÇÃO 5
2 INTRODUÇÃO À COMPUTAÇÃO DISTRIBUÍDA 6
2.1 Características fundamentais 6
2.2 Cliente-Servidor 6
3 ESCALABILIDADE 8
4 VANTAGENS DA ARQUITETURA PROPOSTA 9
5 DESVANTAGENS E DESAFIOS 10
6 DIAGRAMA 11
7 ALGORITMO KMP - ANÁLISE E JUSTIFICATIVA 12
7.1 Justificativa 12
7.1.1 Eficiência Temporal 12
7.1.2 Otimização para Buscas em Texto Científico 12
7.1.3 Adequação ao Sistema Distribuído 13
7.2 Vantagens sobre Alternativas 13
7.2.1 Comparação com Força Bruta: 13
7.2.2 Comparação com Boyer-Moore: 13
8 CONCLUSÃO 14
REFERÊNCIAS 15

1 INTRODUÇÃO

O repositório arXiv, um dos principais arquivos de pré-publicações científicas, hospeda atualmente mais de 2 milhões de artigos, demandando soluções computacionais robustas para processamento e recuperação de informações.

Neste contexto, o relatório apresenta uma análise teórica de um sistema de busca distribuído projetado para processar dados do arXiv. Esse projeto utiliza dois servidores independentes interconectados através de sockets em Java, onde cada servidor é responsável por processar metade do conjunto total de artigos científicos. Esta abordagem visa otimizar o desempenho das buscas através da distribuição de carga e processamento paralelo. Também será abordado a computação distribuída, escalabilidade e outros assuntos para melhor compreensão do programa.

2 INTRODUÇÃO À COMPUTAÇÃO DISTRIBUÍDA

A computação distribuída refere-se a um modelo de computação no qual diferentes componentes de um sistema computacional estão localizados em computadores conectados em rede, que se comunicam e coordenam suas ações através da troca de mensagens. Segundo Tanenbaum e Van Steen (2006), um sistema distribuído pode ser definido como "uma coleção de computadores independentes que aparece para seus usuários como um único sistema coerente". (TANENBAUM, Andrew S, VAN STEEN, Maarten (2006))

2.1 Características fundamentais

Para entender de fato a computação distribuída, é necessário examinar suas características fundamentais. A concorrência representa um aspecto central, onde diversos processos executam ao mesmo tempo, compartilhando recursos entre diferentes usuários e processos, o que demanda sofisticados mecanismos de coordenação. Outro aspecto fundamental é a falta de um relógio global, que torna impossível a sincronização perfeita entre processos, exigindo a implementação de algoritmos de sincronização aproximada e ordenação de eventos baseada em causalidade. Além disso, os sistemas distribuídos caracterizam-se pela possibilidade de falhas independentes, onde componentes podem falhar isoladamente sem comprometer todo o sistema, embora isso exija robustos mecanismos de detecção e recuperação.

2.2 Cliente-Servidor

As arquiteturas de sistemas distribuídos podem ter diferentes formas, cada uma com suas próprias características e aplicações. O modelo cliente-servidor representa uma das abordagens mais tradicionais, estabelecendo uma clara separação entre provedores e consumidores de serviços. Esta arquitetura facilita a manutenção e atualização do sistema, embora possa criar pontos únicos de falha. Por outro lado, as arquiteturas peer-to-peer (P2P) distribuem as responsabilidades igualmente entre todos os nós, que atuam simultaneamente como clientes e servidores. Esta abordagem oferece maior escalabilidade e resistência a falhas, embora introduza desafios significativos em termos de coordenação e consistência. Já as arquiteturas

híbridas, por sua vez, buscam combinar os benefícios de ambas as abordagens, oferecendo maior flexibilidade na distribuição de recursos.

3 ESCALABILIDADE

A escalabilidade, segundo (BONDI, André B. (2006)), refere-se à capacidade de um sistema de acomodar um número crescente de elementos ou objetos, processar volumes crescentes de trabalho com graça e/ou estar preparado para crescer. No contexto do sistema de busca distribuído para o arXiv, este conceito assume certa importância devido ao crescimento contínuo do repositório de artigos científicos. A escalabilidade é uma característica essencial para redes, sistemas ou processos. Uma escalabilidade inadequada pode causar uma queda no desempenho, exigindo a reformulação ou duplicação de sistemas.

A arquitetura proposta, baseada em dois servidores que dividem a responsabilidade de busca, implementa a escalabilidade de várias formas. A escalabilidade horizontal é alcançada através da possibilidade de adicionar novos servidores ao sistema, permitindo a distribuição do conjunto de dados em partições ainda menores. Por exemplo, se o volume de artigos crescer consideravelmente, é possível expandir o sistema de dois para quatro ou mais servidores, cada um responsável por uma fração menor do conjunto total de dados.

A escalabilidade vertical, por sua vez, manifesta-se na capacidade de aumentar os recursos computacionais de cada servidor individual. Isso pode envolver o upgrade de processadores, adição de memória RAM ou expansão do armazenamento em disco. Esta flexibilidade é particularmente valiosa quando se lida com consultas complexas que demandam maior poder de processamento.

4 VANTAGENS DA ARQUITETURA PROPOSTA

A arquitetura distribuída apresenta diversas vantagens significativas para o sistema de busca. O processamento paralelo permite a execução simultânea de consultas em diferentes partes do conjunto de dados, reduzindo o tempo de resposta para buscas complexas. Por exemplo, uma busca que envolveria a análise sequencial de milhões de artigos pode ser dividida entre os servidores, com cada um processando sua parte ao mesmo tempo.

A distribuição natural da carga entre os servidores evita gargalos de processamento e permite um uso mais eficiente dos recursos disponíveis. A arquitetura também facilita a manutenção do sistema, pois é possível realizar atualizações ou manutenções em um servidor enquanto o outro continua operacional, reduzindo o tempo de inatividade.

A redundância presente ao sistema distribuído aumenta a disponibilidade e confiabilidade do serviço. Mesmo em casos de falha parcial, o sistema pode continuar oferecendo funcionalidade reduzida, em vez de uma falha total que tornaria o serviço completamente indisponível

5 DESVANTAGENS E DESAFIOS

Apesar de todas suas vantagens, a arquitetura distribuída também apresenta desafios importantes. A complexidade do sistema aumenta consideravelmente em comparação com uma solução centralizada. A necessidade de coordenação entre os servidores, sincronização de dados e gerenciamento de falhas acrescenta camadas adicionais de complexidade tanto no desenvolvimento quanto na manutenção do sistema.

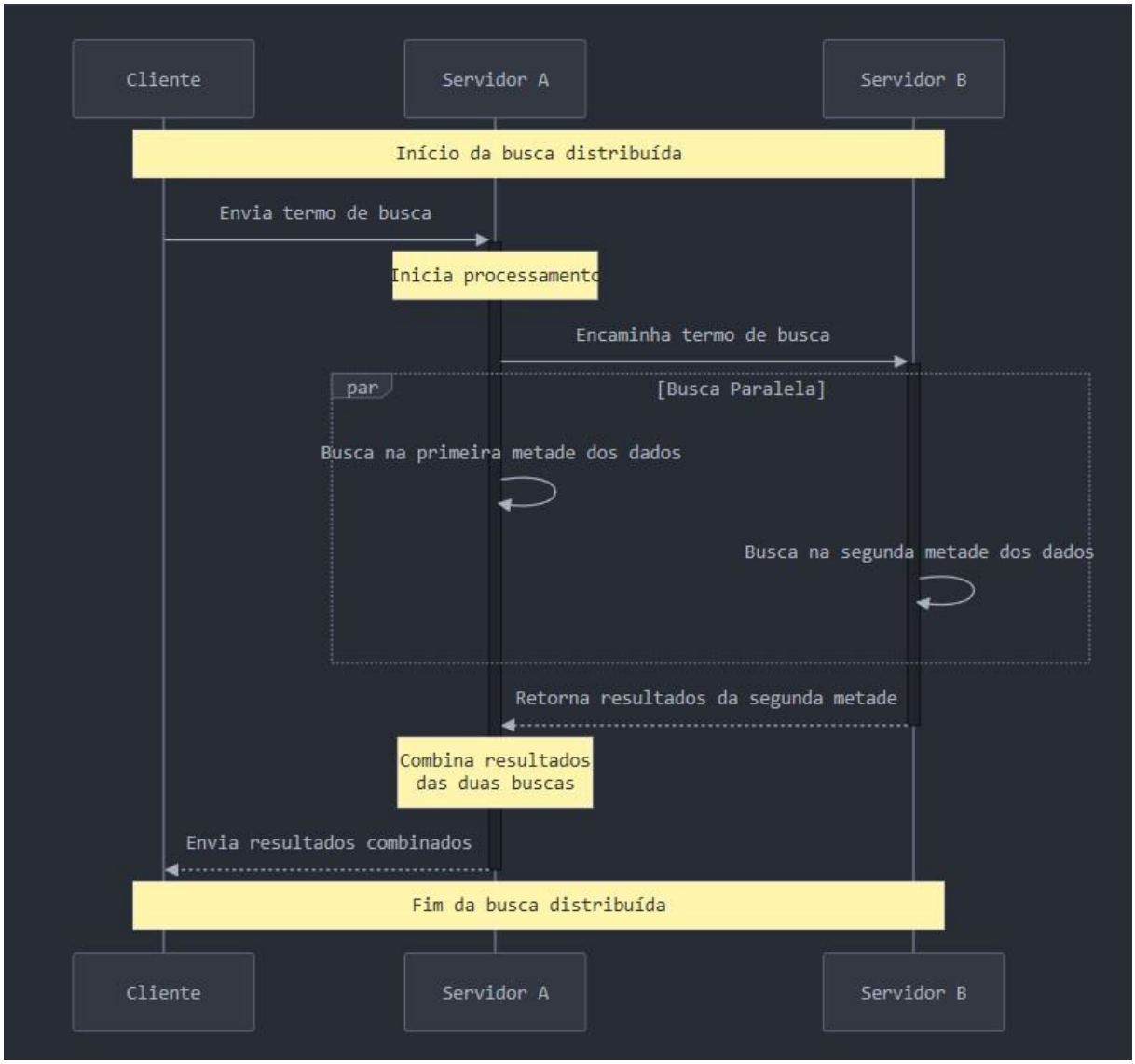
O overhead de comunicação entre os servidores, realizado através de sockets, pode introduzir latência adicional nas operações do sistema. Esse aspecto é especialmente importante em consultas que precisam unir resultados de ambos os servidores, já que o tempo total de resposta abrange não apenas o processamento paralelo, mas também a comunicação e a combinação dos resultados.

A consistência dos dados emerge como outro desafio significativo. Garantir que ambos os servidores mantenham visões consistentes do conjunto de dados, especialmente durante atualizações ou em casos de falha, requer implementação cuidadosa de protocolos de sincronização e resolução de conflitos.

Do ponto de vista econômico, a infraestrutura necessária para manter múltiplos servidores, incluindo hardware, rede e manutenção, representa um custo maior em comparação com sistemas centralizados. Este aspecto deve ser cuidadosamente considerado no planejamento e implementação do sistema.

6 DIAGRAMA

Figura 1 – Diagrama explicando comunicação durante uma busca entre cliente e servidores



O diagrama foi criado utilizando a linguagem Mermaid.

7 ALGORITMO KMP - ANÁLISE E JUSTIFICATIVA

O algoritmo KMP (Knuth-Morris-Pratt) é um algoritmo eficiente de busca de padrões em strings, desenvolvido por Donald Knuth, James H. Morris e Vaughan Pratt em 1977(KNUTH, Donald E. et al. (1977)). Sua escolha para o sistema de busca distribuído do arXiv é adequada devido a várias características fundamentais que o tornam ideal para esta aplicação.

O KMP funciona através de um mecanismo sofisticado que evita comparações desnecessárias ao buscar um padrão em um texto. O algoritmo utiliza as informações de correspondências parciais anteriores para determinar onde a próxima correspondência poderia começar, eliminando a necessidade de voltar no texto sendo pesquisado.

O algoritmo opera em duas fases principais: Pré-processamento do padrão e processo de busca. O algoritmo primeiro analisa o padrão a ser buscado e constrói uma tabela auxiliar (também chamada de tabela de prefixo-sufixo ou lps - longest proper prefix which is also suffix). Esta tabela armazena informações sobre o próprio padrão e é usada para otimizar a busca. Durante a busca real, o algoritmo usa a tabela lps para determinar quanto pode avançar no texto quando encontra uma incompatibilidade, evitando retroceder no texto principal.

7.1 Justificativa

A escolha do algoritmo KMP para o sistema de busca distribuído do arXiv é justificada por vários fatores críticos como:

7.1.1 Eficiência Temporal

- Complexidade temporal O(n + m), onde n é o comprimento do texto e m é o comprimento do padrão
- Não requer retrocesso no texto principal
- Particularmente eficiente para grandes volumes de texto, como artigos científicos
- Ideal para processamento paralelo em sistemas distribuídos

7.1.2 Otimização para Buscas em Texto Científico

- Eficaz na busca de termos técnicos e expressões complexas comuns em artigos científicos

- Capaz de lidar com padrões repetitivos que podem ocorrer em texto técnico
- Mantém eficiência mesmo com padrões longos (como títulos ou frases completas)

7.1.3 Adequação ao Sistema Distribuído

- Facilmente paralelizável, permitindo busca simultânea em diferentes partes do conjunto de dados
- Baixo overhead de memória, importante quando processando grandes volumes de dados
- Consistente em termos de performance, facilitando o balanceamento de carga entre servidores

7.2 Vantagens sobre Alternativas

7.2.1 Comparação com Força Bruta:

- KMP: O(n + m)
- Força Bruta: O(nm)
- KMP elimina retrocessos desnecessários

7.2.2 Comparação com Boyer-Moore:

- Embora Boyer-Moore possa ser mais rápido em alguns casos, KMP tem performance mais consistente
- KMP é mais simples de implementar e manter em um sistema distribuído
- Menor overhead de memória comparado ao Boyer-Moore

8 CONCLUSÃO

O desenvolvimento de um sistema de busca distribuído para artigos científicos do arXiv representa uma aplicação prática e sofisticada dos conceitos fundamentais de computação distribuída, demonstrando como diferentes aspectos teóricos se traduzem em soluções efetivas para problemas reais.

Este projeto demonstra a sinergia entre teoria e prática na computação distribuída, onde conceitos fundamentais como escalabilidade, tolerância a falhas, comunicação assíncrona e processamento paralelo se combinam com implementações práticas de algoritmos e protocolos de comunicação. O resultado é um sistema robusto e eficiente, capaz de atender às demandas crescentes de busca em repositórios científicos de grande escala.

REFERÊNCIAS

BONDI, A. B. Characteristics of scalability and their impact on performance. 2006.

KNUTH, D. E.; JR, J. H. M.; R., V. Fast pattern matching in strings. SIAM journal on computing, v. 6, n. 2, p. 323 – 350, Jun 1977.

TANENBAUM, A. S.; STEEN, M. V. Distributed systems: principles and paradigms. 2nd ed.. ed. Upper Saddle River: Pearson Prentice Hall, 2006.