Relatório da atividade de paralelismo através de threads

Computação Escalável

Dominique de Vargas de Azevedo
Pedro Thomaz Conzatti Martins
Tatiana Lage
Thiago Franke Melchiors

Professor: Thiago Pinheiro de Araújo

RIO DE JANEIRO
2024

Conteúdo

1 Metodologia

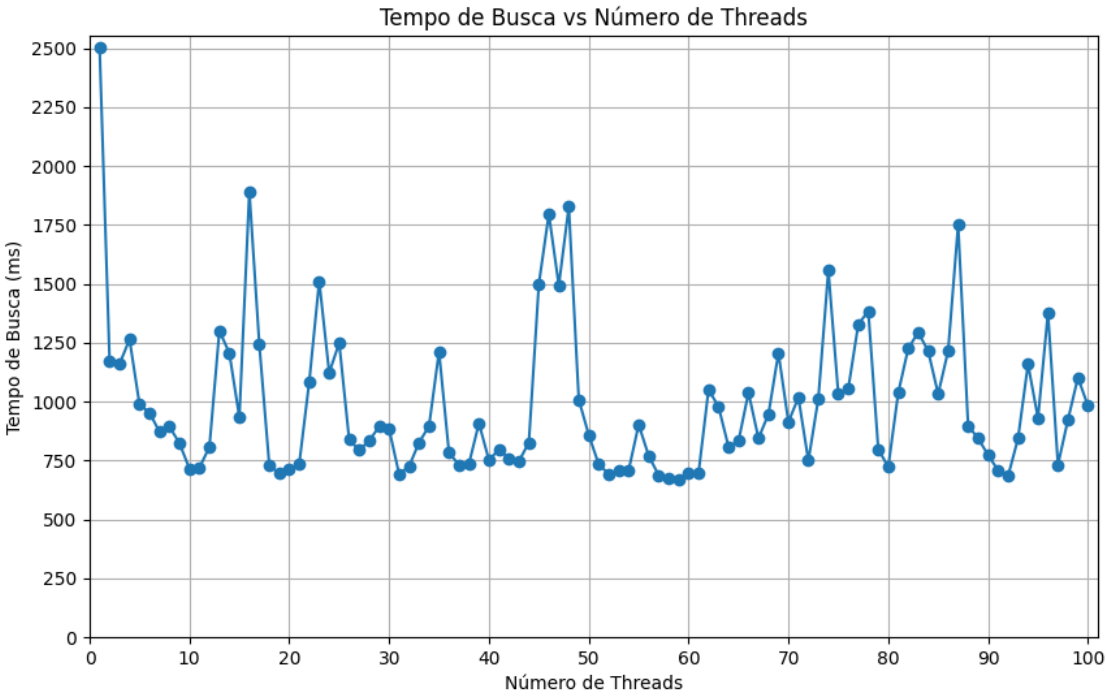
Primeiramente, realizamos o download da versão textual completa das obras de William Shakespeare no site do Project Gutenberg. O arquivo possui aproximadamente 190.000 linhas. Em seguida, utilizamos o script `increaseVolume.py` para aumentar o volume de dados do conjunto em 300 vezes, permitindo assim uma percepção significativa de aumento de velocidade no processamento dos dados por meio de threads.

Na sequência, o arquivo `wordCounter.cpp` utiliza threads para realizar a contagem de palavras em um texto. A configuração padrão procura pelas palavras "love"e "hate", mas o código é flexível para incluir outros casos. A constante global `MAX_THREADS` define o número máximo de threads que podem ser utilizadas. O programa possui duas funções auxiliares: `wordCount`, que conta o número de ocorrências de uma palavra específica dentro de um intervalo de texto, e `wordsCount`, que utiliza a função `wordCount` para contar ocorrências das múltiplas palavras solicitadas e armazenar os resultados em um vetor.

Na função pincipal, o programa solicita ao usuário o caminho de um arquivo de texto e, após a leitura dele, inicia um cronômetro para computar o tempo de execução. O texto é então dividido em blocos, e cada thread é responsável por contar as palavras em um bloco específico. O programa é capaz de configurar o número de threads e medir o tempo de preparação dos dados, o tempo de execução da pesquisa e o tempo total, além de exibir o número de ocorrências de cada palavra pesquisada e qual palavra foi mais utilizada. Como funcionalidade adicional, todos os dados coletados pelo programa são salvos em `results.csv`.

Finalmente, com os dados computados e registrados em `results.csv`, utilizamos o script `graphMaker.py` para extrair o conjunto e criar um gráfico que correlaciona o número de threads com o tempo de execução da busca pelas palavras "love"e "hate"no texto. Esse processamento paralelo com threads permite otimizar a busca e análise das ocorrências dessas palavras, tornando o processo mais eficiente e rápido. O gráfico gerado nos ajudará a visualizar como o desempenho varia conforme o número de threads utilizado.

2 Resultados



O gráfico apresenta uma análise detalhada do tempo de busca em relação ao número de threads utilizadas no programa.

Inicialmente, à medida que o número de threads aumenta, observamos uma diminuição notável no tempo de busca. Isso indica que a paralelização do processo de pesquisa está resultando em um processamento mais eficiente dos dados. O aumento no número de threads permite que várias partes do conjunto de dados sejam processadas simultaneamente, reduzindo assim o tempo necessário para concluir a busca.

No entanto, essa tendência positiva não é sustentada indefinidamente. Após atingir um certo ponto, aproximadamente 16 threads no nosso caso, o tempo de busca começa a aumentar novamente. Isso sugere a presença de um ponto de saturação, onde adicionar mais threads não proporciona ganhos significativos de desempenho. Pelo contrário, pode até levar a um aumento no tempo de busca devido a possíveis sobrecargas no sistema.

Essa observação indica que pode haver limitações adicionais no sistema que estão impactando o desempenho do programa. Essas limitações podem incluir fatores como a capacidade de processamento da CPU, a disponibilidade de recursos de memória ou até mesmo questões relacionadas à comunicação entre as threads.

Portanto, é necessário uma análise mais aprofundada para identificar a causa específica desse comportamento, podendo envolver o monitoramento detalhado do uso de recursos do sistema durante a execução do programa, bem como a revisão e otimização do algoritmo de pesquisa para garantir um melhor aproveitamento dos recursos disponíveis.