Paralelismo através de threads
Computação Escalável

Rian Freitas, Hanna Rodrigues e
Yonathan Rabinovici Gherman

Março 2024

1 Introdução

O objetivo deste relatório é analisar os resultados obtidos a partir de um exercício de paralelismo realizado como parte da disciplina de Computação Escalável. O exercício consistiu na implementação de um programa em C++ para contar a frequência das palavras "hate" e "love" em um arquivo de texto, utilizando paralelismo para melhorar o desempenho.

2 Implementação

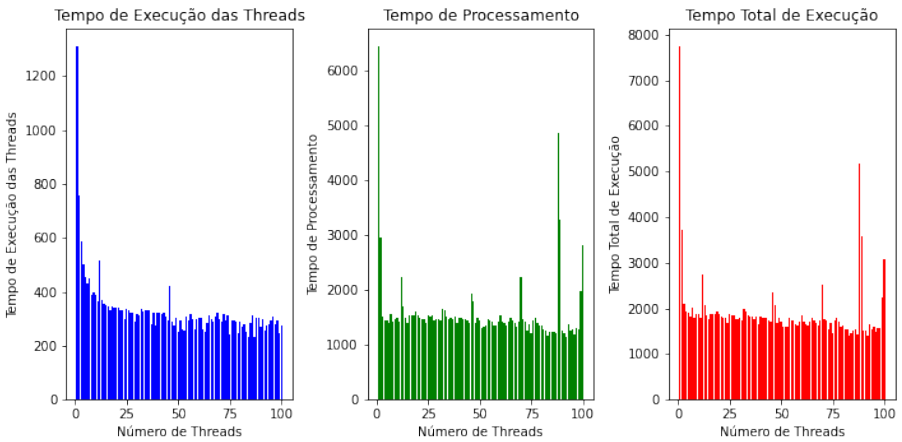
O programa foi desenvolvido em C++ e emprega o uso de threads para processar diferentes partes do arquivo de texto simultaneamente. O arquivo é dividido em blocos de forma a evitar que palavras sejam cortadas na divisão, e cada bloco é atribuído a uma thread para realizar a contagem das palavras. O número de threads utilizado varia de 1 a 100, permitindo uma análise do impacto do paralelismo no desempenho do programa.

3 Resultados

Os resultados do experimento foram registrados e analisados. E a palavra "love" é a mais frequente tendo sido contada 387.086 vezes e a palavra "hate" 34.916 vezes.Além disso os principais pontos observados foram:

1. **Tempo de Processamento:**
    - O tempo de processamento diminui à medida que o número de threads aumenta até um certo ponto. Isso ocorre devido à distribuição da carga de trabalho entre as threads, permitindo uma execução mais rápida.

- No entanto, após um determinado número de threads, o tempo de processamento começa a aumentar. Isso pode ser atribuído ao overhead de criação e gerenciamento de threads, que se torna mais significativo à medida que o número de threads aumenta.



2. **Concorrência entre as threads:**

   - Inicialmente, uma variável global foi criada para armazenar a contagem das palavras "love" e "hate". No entanto, à medida que o número de threads aumentava além de 1, ocorria a concorrência de acesso a essas variáveis, resultando em instabilidade nos resultados.
   - Uma alternativa adotada foi utilizar variáveis de contagem internas a cada thread e, após o término das operações, somar essas contagens às variáveis globais. Isso evitou o fenômeno da concorrência e garantiu a consistência dos resultados.

4 Conclusões

Com base nos resultados obtidos, podemos concluir que o paralelismo pode ser uma ferramenta eficaz para melhorar o desempenho de programas computacionais, especialmente em tarefas que podem ser facilmente divididas em partes independentes. No entanto, é importante considerar o overhead e a concorrência de recursos associado à criação e gerenciamento de threads, o que pode limitar a escalabilidade do programa em ambientes com um grande número de threads ou comprometer os resultados obtidos.