Universidade Estadual de Campinas - FT

Projeto prático da disciplina

Sistemas Operacionais

TT304A – Sistemas Operacionais
2º semestre de 2024
Prof. André Leon S. Gradvohl, Dr

Alunos:
Antonio Carlos Rosendo da Silva - 174258
Eduardo Castro Brito - 281409

1. Descrição do Problema

O problema envolve processar vários arquivos de entrada que contêm listas de valores inteiros desordenados, e mesclá-los em um único arquivo de saída ordenado. O programa deve permitir a utilização de um número específico de threads para paralelizar o processamento, onde cada thread será responsável por ler e ordenar os dados de um arquivo. A implementação deve ser capaz de trabalhar com uma quantidade de threads especificada pela linha de comando, e o número de threads permitido deve ser de 1, 2, 4 ou 8.

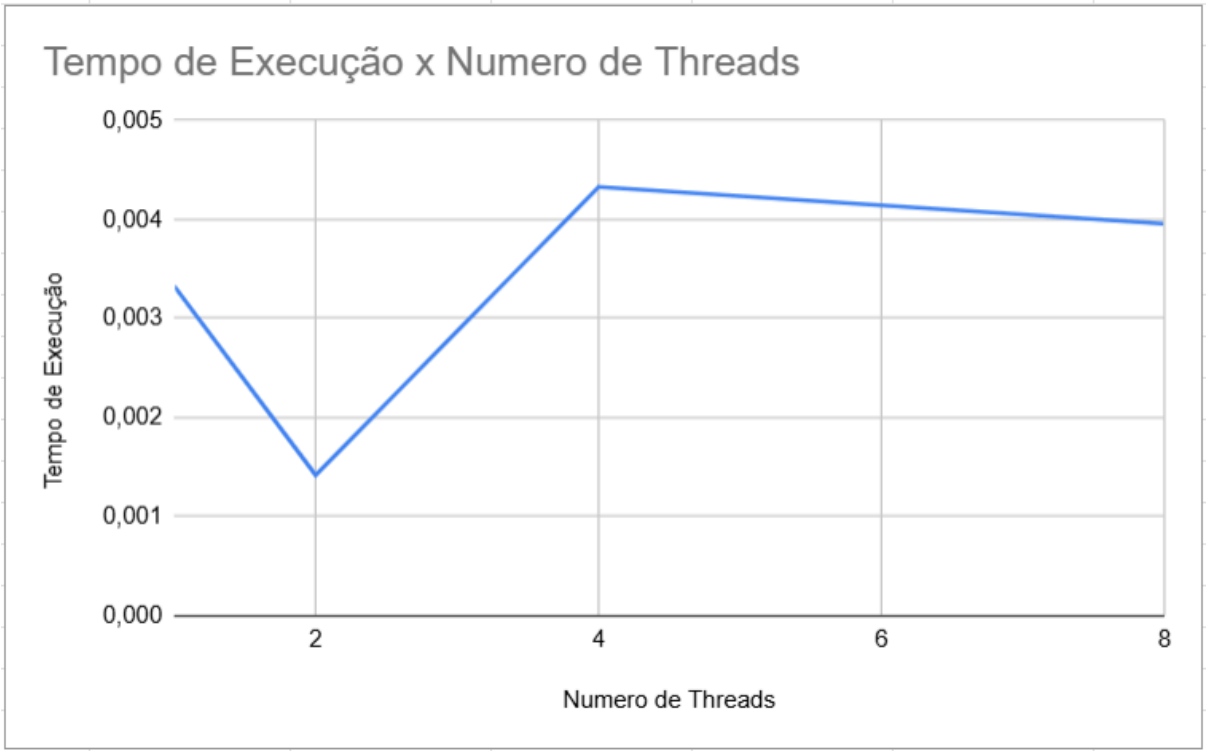
2. Instruções para compilar o programa

Com a utilização do arquivo makefile para auxiliar na compilação e execução do programa de maneira padronizada, o usuário deve utilizar os seguintes comandos em Sistemas Linux:

- Compilação: utilizar o comando make para executar a linha de comando padrão presente no arquivo makefile: gcc -o mergesort mergesort.c -lpthread
- Execução com 1 thread: utilizar o comando make 1, que executará o seguinte comando: ./mergesort 1 arq1.dat arq2.dat arq3.dat -o saida.dat
- Execução com 2 threads: utilizar o comando make 2, que executará o seguinte comando: ./mergesort 2 arq1.dat arq2.dat arq3.dat -o saida.dat
- Execução com 4 threads: utilizar o comando make 4, que executará o seguinte comando: ./mergesort 4 arq1.dat arq2.dat arq3.dat -o saida.dat. Sendo esse o teste solicitado pelo professor, visto que o código é idêntico ao utilizado na explicação do problema que nos foi passado.
- Execução com 8 threads: utilizar o comando make 8, que executará o seguinte comando: ./mergesort 8 arq1.dat arq2.dat arq3.dat -o saida.dat

Vale ressaltar também que nesse programa a utilização dos comandos de execução do makefile estão definidos para utilizar somente 3 arquivos de entrada contendo 1000 inteiros em cada, porém, o programa pode ser utilizado com até 8 arquivos de entrada.

3. Gráficos com os resultados



Os testes mostram que 2 threads ofereceram o melhor desempenho, pois dividiram as tarefas de forma equilibrada com menor sobrecarga de gerenciamento. Com 4 e 8 threads, o tempo aumentou devido ao número de threads exceder o de arquivos, causando threads ociosos e sobrecarga desnecessária. Conclusão: o número de threads deve ser ajustado ao número de arquivos para otimizar o desempenho.

4. Conclusões

Este projeto apresenta uma solução eficiente para o problema de mesclagem de múltiplos arquivos de inteiros desordenados em um único arquivo de saída ordenado, utilizando paralelismo com threads para melhorar o desempenho. Com o uso do mergesort em cada thread, e uma mesclagem final dos resultados, o programa consegue lidar de maneira eficaz com grandes volumes de dados. Além disso, a medição do tempo de execução de cada thread e do tempo total do programa permite avaliar o desempenho e identificar possíveis problemas. Esta implementação demonstra a aplicação prática de conceitos de concorrência e paralelismo, sendo um exemplo útil de otimização em processamento de dados usando threads e ordenação paralela.

Endereço do repositório GitHub: https://github.com/antxniozr/TT304A-Hunters

Link do vídeo: https://www.youtube.com/watch?v=EiMWwXNIDb8&ab_channel=AntonioCarlosRosendoDaSilva