UNIVERSIDADE ESTADUAL DE CAMPINAS - FACULDADE DE TECNOLOGIA

GUILHERME TUPINAMBÁ DURÃES
LUCAS DE SOUZA TIMOTEO
MARCOS GABRIEL NOCETTI SANTOS

PROJETO “MERGE AND SORT”

LIMEIRA, SÃO PAULO
2019

1. OBJETIVO

Este projeto visa a criação de um programa que utilize múltiplas threads para ler valores inteiros de N arquivos de entrada e armazenar em ordem crescente em um único arquivo de saída. Os dados dos arquivos de entrada não estão ordenados.

2. DESCRIÇÃO DO PROBLEMA A SER RESOLVIDO

Considere que existem N arquivos de entrada com diferentes quantidades de valores inteiros não ordenados e que podem ser repetidos ou não. O programa deverá ler os valores inteiros desses vários arquivos e, de forma ordenada, armazená-los em um arquivo de saída.

O programa deve ser testado para 2, 4, 8 e 16 threads, com arquivos com diferentes quantidades de números inteiros.

3. ENTRADAS E SAÍDAS DE VALORES PARA O PROGRAMA

Entradas: O número de threads que o programa deve utilizar, os nomes dos arquivos de entrada e o nome do arquivo de saída. Para os testes, considere pelo menos 5 arquivos com 1000 valores cada.

Saídas: Arquivo com os valores ordenados.

Por exemplo:

./mergesort 4 arq1.dat arq2.dat arq3.dat –o saida.dat

Onde:

- ./mergesort é o nome do programa;
- 4 é o número de threads;
- arq1.dat arq2.dat arq3.dat são os arquivos com dados de entrada;
- -o saida.dat é o arquivo que contém os dados de saída.

4. CÓDIGO FONTE

O código fonte do programa se encontra no seguinte diretório apresentado no GitHub:
<https://github.com/LucaStarT/Projeto-SO/blob/master/so.c>

5. VÍDEO DO CÓDIGO

O vídeo explicativo do código fonte e da apresentação dos resultados se encontra em:
<https://www.youtube.com/watch?v=brhAdGUIwMk&feature=youtu.be>

6. INSTRUÇÕES SOBRE O PROGRAMA

6.1 Descrição para a Compilação

Para a compilação do programa, é aconselhável a utilização do sistema Linux, as instruções aqui descritas serão baseadas neste ambiente:

1 - Primeiramente é necessário acessar o repositório disponível no seguinte endereço do Github: <https://github.com/LucaStarT/Projeto-SO/blob/master/so.c>;
2 - Entrando nele, deve ser feito o clone do arquivo ou então baixar os arquivos lá postados;
3 - Após isso, deve-se acessar o terminal de seu computador para acessar a pasta na qual os arquivos foram baixados (etapa 2), inserindo assim, os seguintes comandos respectivamente:

gcc -o co -lpthread
./so X arquivo.dat saida.out

Onde X = número de Threads (2, 4, 8 ou 16)

Seguindo esses passos, o código será executado.

6.2 Descrição do Código

Foi criado um arquivo para escrever o que vai ser a saída. Depois foi iniciado um número inteiro, com uma condição que lê os arquivos e criará outro arquivo de leitura, e em um laço de 'for' cria-se outro laço 'while' que vai ler o arquivo e alocar a variável número e escrever em outro arquivo de saída, assim fechando os dois arquivos.

Depois aloca-se um vetor de 1000 números. Depois vai imprimir os valores já ordenados em uma matriz e escrever os números presentes nele. Depois uma função para ordenar os números dentro vetor. Por fim passar os valores para a thread.

7. RESULTADOS E CONCLUSÕES

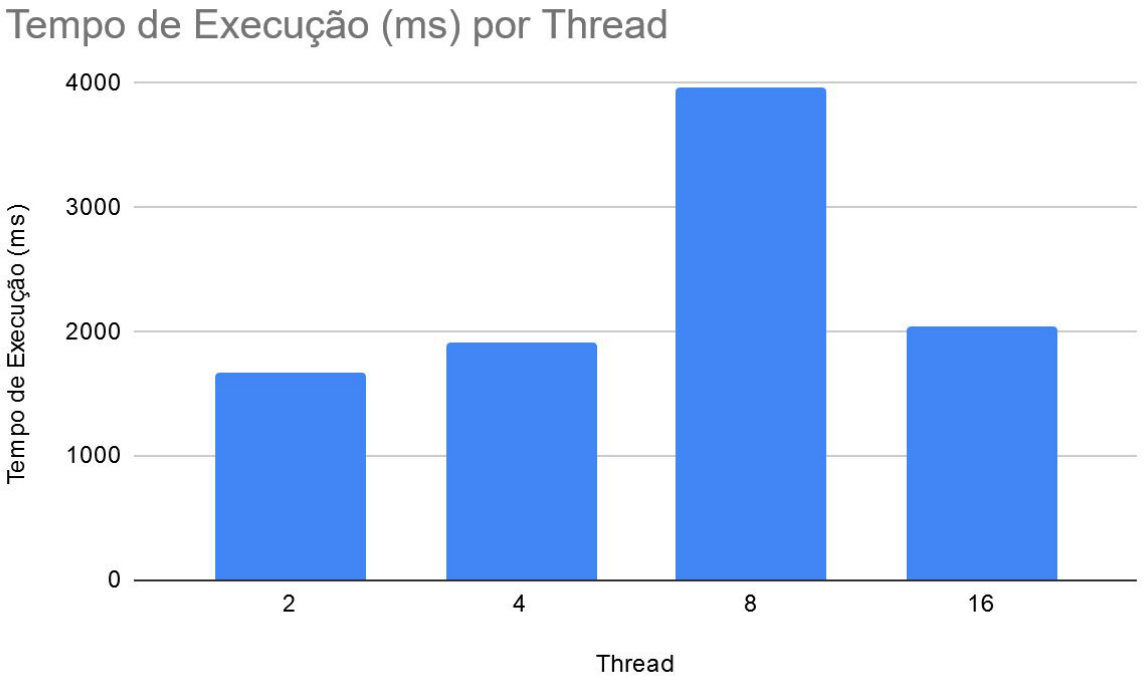

Gráfico 1

Como pode ser observado no Gráfico 1, pode-se concluir que nas duas primeiras Thread (2 e 4) quanto menor a Thread, menor o tempo de execução. Já nas Threads 8 e 16, a com menor número levou um maior tempo de execução em relação a outra. Outra coisa que pode influenciar no Tempo de Execução, é a quantidade de dados processados, pois uma Thread pode processar de tempo diferente em relação a outra.