UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

TT304 - Sistemas Operacionais

Integrantes:

Brenda Cristina de Souza Silva	194836
Mariana Albino Q. Antonio	202919

Limeira - SP
2019

Repositório no Git:
https://github.com/CristinaBrenda/SOProject
Link para o vídeo:
https://drive.google.com/file/d/1s7tH9hRMowdlboU4br41YDaEWcYNJrOI/view?usp=sharing

1. PROBLEMA

O projeto consiste em desenvolver um programa que leia diversos valores inteiros de n arquivos e armazene, de forma ordenada em um único arquivo de saída, deve utilizar múltiplas threads para tais funções.

2. SOLUÇÃO

O programa elaborado para a disciplina utiliza a seguinte lógica, primeiramente lê os valores dos arquivos de entrada e os armazena em uma matriz dinâmica, de modo que cada cada linha do vetor será preenchida pelos valores presentes em cada arquivo. Em seguida a quantidade de elementos é dividida pelo número de threads passado ao executar a aplicação, cada thread ficará responsável por ordenar a respectiva parte do vetor, o resto da divisão será anexado ao último subvetor gerado após a divisão do arquivo integral.

Após a ordenação dos subvetores, as threads são encerradas e a thread principal fará a adjeção dos subvetores utilizando a função de junção de forma ordenada “merge()”, posterior às junções necessárias, os vetores menores do que o maior vetor deve ser preenchido de maneira que todos os vetores tenham o mesmo número de componentes ao final desse processo.

Por fim, os vetores ordenados, preenchidos com “0” (zeros) se preciso, são armazenados no arquivo de saída referido pelo usuário na execução do software. Há “printfs” comentados a partir da linha 288, os mesmos podem e devem ser utilizadas com o propósito de constatar se os vetores finais estarão em conformidade com a proposta do projeto, ainda com a mesma finalidade há arquivos binários com diferentes quantidades de elementos.

Imprescindível destacar alguns pontos, a aplicação foi elaborada para ler apenas arquivos binários, o arquivo de saída gerado também será binário, o programa suporta apenas 2, 4, 8 ou 16 threads. O programa deve ser executado no seguinte formato e seguindo a ordem:

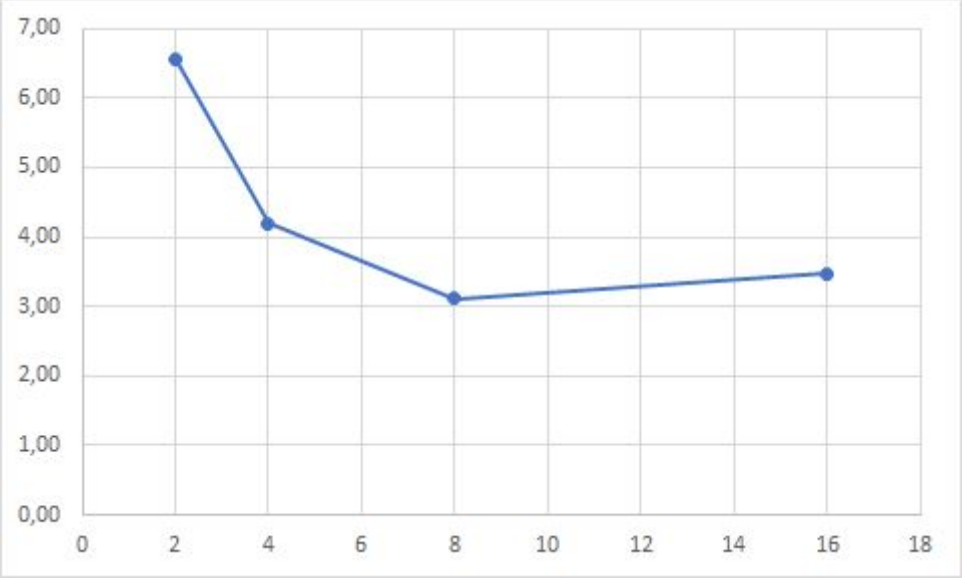
- ./nome_do_programa;
- número_de_threads;
- arquivos_de_entrada;
- -o;
- arquivo_de_saida.

Por exemplo: ./buildmatrix 2 arq1.dat arq2.dat -o saida.dat

Tal orientação também se encontra no comentário inicial do código fonte. A desconsideração de qualquer uma das questões evidenciadas ocasionará em erro fatal do programa. Vale mencionar que a compilação deve conter o comando -lpthread no final. Por exemplo: gcc buildmatrix.c -o buildmatrix -lpthread

3. COMPARAÇÕES DE DESEMPENHO

O teste foi realizado em um computador com sistema Arch Linux e um processador Intel i5-7300HQ com 4 núcleos.



Tempos em milissegundos (ms) na vertical, para cada quantidade de threads utilizadas na horizontal. Os tempos apresentados no gráfico são os mesmos do vídeo.

O tempo de processamento diminui consideravelmente utilizando 4 threads comparado ao tempo de 2 threads. Ao lidar com 8 ou 16 threads não há contraste evidente.

4. CONCLUSÃO

O uso das threads mostra-se vantajoso se baseado na capacidade no computador que executará a aplicação. Ao utilizar um aparelho com 4 núcleos, o desempenho será significamente melhor ao empregar 4 threads se comparado à 2 threads. Já ao utilizar 8 ou 16 não há melhora relevante. Tal situação ocorre devido a capacidade de processamento do computador utilizado, que é de apenas 4 execuções concomitantes.