UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Tecnologia - FT

SISTEMAS OPERACIONAIS

PROF. ANDRÉ LEON SAMPAIO GRADVOHL

PROJETO FINAL SISTEMAS OPERACIONAIS

Projeto_1

Grupo Inventado

Nome:	RA:
Nikolas D'Andrea Freire	222970
Victor Akio Uchiyama	244974
Wallace Benedito da Silva	245752

Limeira - Novembro de 2019

Sumário

1. REPOSITÓRIO GIT

Segue o link para o repositório GitHub com toda a documentação do projeto, assim como o código fonte documentado e o link para acesso ao vídeo de execução do programa.

1.1 Link GitHub

https://github.com/akiouchiyama/grupoinventadoSO

1.2 Link vídeo

https://www.youtube.com/watch?v=AJ1fzDdR2YM&feature=youtu.be

2. O PROBLEMA

Foi proposto para a nossa equipe, elaborar um programa que que utilize múltiplas threads para ler valores inteiros de N arquivos de entrada, armazenando esses valores em ordem crescente em um único arquivo de saída.

A utilização de múltiplas threads teve como objetivo a quantificação de desempenho operacionais da execução desse sistema, utilizando a unidade de tempo (segundos e milissegundos) para cada combinação de threads. A partir desse contexto foi possível gerar um relatório de desempenho e de análise operacional.

2.1 Descrição da Solução do Problema

O projeto criado pela equipe *Inventado* conseguiu introduzir a função de múltiplas threads (2, 4, 8, 16) para ordenar uma sequência de números inteiros em múltiplos arquivos e posteriormente ordená-los num único arquivo.

Nossa lógica para a resolução do problema foi a seguinte: separar etapas do projeto em funções diferentes como leitura dos diferentes arquivos realizados sem o uso de threads; ordenação multithread e escrita do arquivo final sem o uso de threads, onde a main realiza a operação de apenas receber os argumentos da linha de comando; criar as threads; chamar a função leitura, a qual lê todos os arquivos e salva em um único vetor.

A contagem do tempo é feita com a função clock(), e depois enviamos cada thread para a função ordena com um laço onde cada thread fará a ordenação do vetor, como passamos o endereço do vetor, ela simultaneamente ordenará este. Por fim, recebemos o tempo final que levamos para realizar esta operação e enviamos o vetor ordenado para a função escreve, a qual imprime o vetor no arquivo de saída.

2.2 Instruções de compilação

Para a compilação é necessário acessar o link do repositório no GitHub, realizar o download em https://github.com/akiouchiyama/grupoinventadoSO, em seguida, acessar via terminal do sistema operacional Linux a pasta no qual estão os arquivos baixados e inserir os seguintes comandos:

```
gcc -pthread thread.c -o thread
```

Em seguida digite:

```
./thread numero_de_threads arqN.txt saida.txt
```

Segue abaixo a ilustração da compilação do programa:

./antigo 2 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt **//PARA 2 THREADS**

./antigo 4 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt **//PARA 4 THREADS**

./antigo 8 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt **//PARA 8 THREADS**

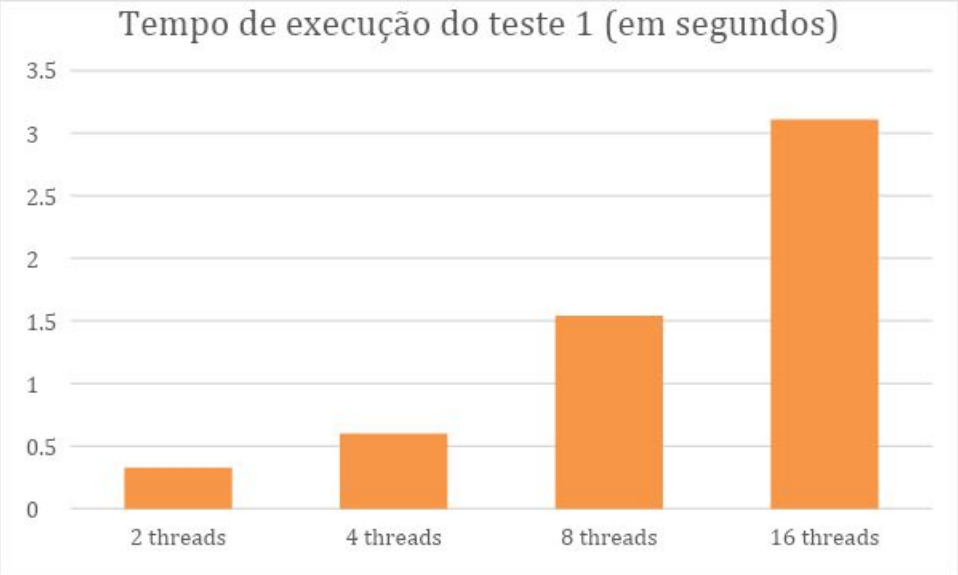
./antigo 16 arq0.txt arq1.txt arq2.txt arq3.txt arq4.txt saida.txt **//PARA 16 THREADS**

Obs: Como o programa acima foi executado em um dos computadores dos membros da equipe, o nome do código fonte está diferente de como está no GitHub. Portanto, no momento da compilação, ao invés de digitar "./antigo", digite "./thread".

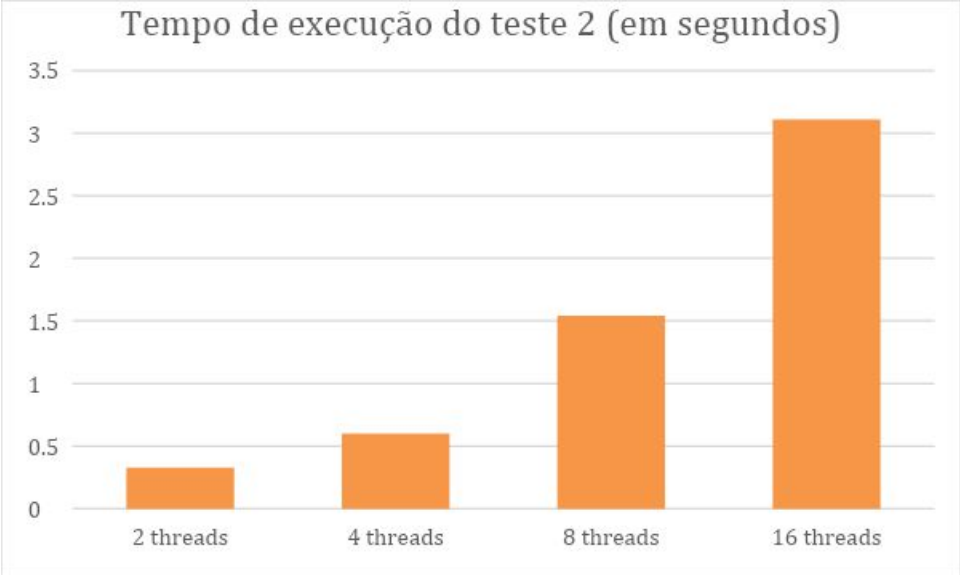
3. GRÁFICOS DE EXECUÇÃO

Para este teste foi utilizado uma máquina com processador Intel Core i7-8550U de 8°Geração (1.8GHz até 4.0GHz) 8Mb Cache, Memória RAM 8GB DDR4.

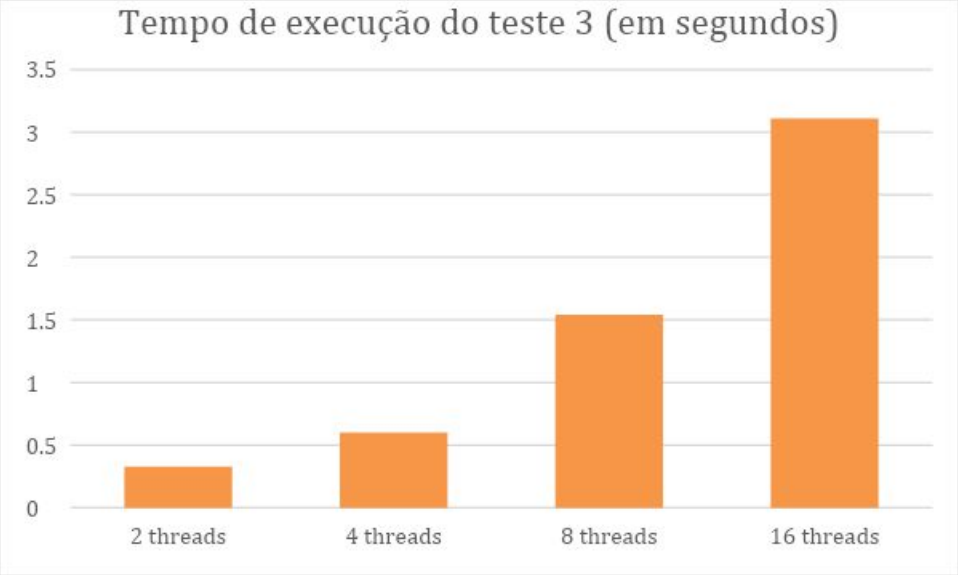
Teste 1:


Tempo de execução do teste 1 (em segundos)
3.5
3
2.5
2
1.5
1
0.5
0
2 threads
4 threads
8 threads
16 threads

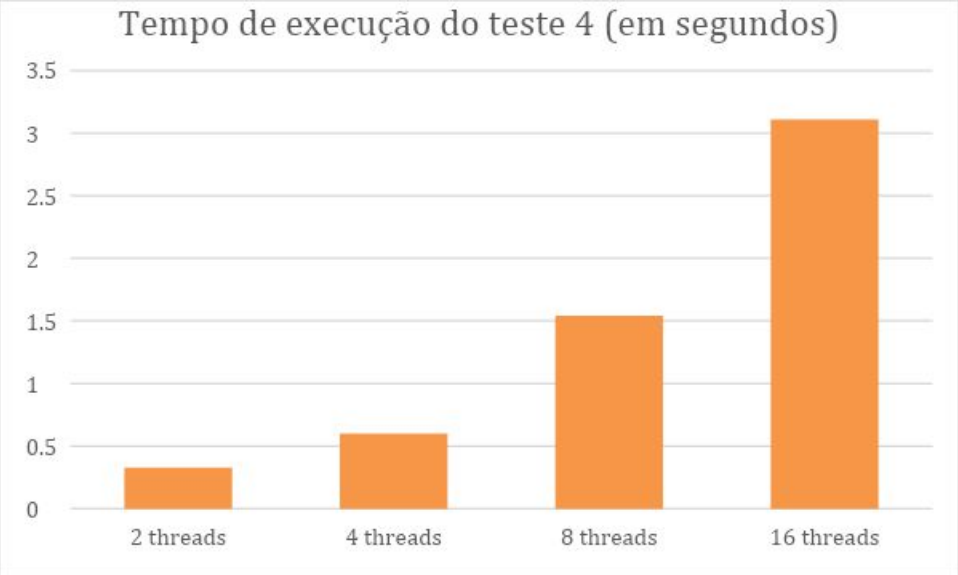
Teste 2:


Tempo de execução do teste 2 (em segundos)
3.5
3
2.5
2
1.5
1
0.5
0
2 threads
4 threads
8 threads
16 threads

Teste 3:


Tempo de execução do teste 3 (em segundos)
3.5
3
2.5
2
1.5
1
0.5
0
2 threads
4 threads
8 threads
16 threads

Teste 4:


Tempo de execução do teste 4 (em segundos)
3.5
3
2.5
2
1.5
1
0.5
0
2 threads
4 threads
8 threads
16 threads

4. CONCLUSÃO

Durante o processamento de um programa, a thread cria várias linhas de execução. Essas linhas têm o processamento dividido pelo sistema operacional e repassa ao processador. É basicamente assim que funciona o conceito de multi – thread.

Com base nos testes feitos pelo grupo, notamos que com um número maior de threads o tempo de execução aumenta proporcionalmente, pois cada thread faz a verificação inteira do vetor para verificar se este está ordenado sequencialmente.

Portanto, podemos concluir, que nem sempre ao adicionar mais threads será benéfico para diminuir o tempo de execução.