Projeto Prático de Sistemas Operacionais

Disciplina: TT304A - Sistemas Operacionais

1º semestre de 2024

Prof. Dr. André Leon S. Gradvohl

Nome do grupo: Grupo M

Integrantes:

Marcelo Expedito Costa de Oliveira - 248007

Miguel Gomes Fecchio - 187871

1 INTRODUÇÃO

Neste trabalho, desenvolvemos um programa, em Linguagem C (pura), capaz de, ao ser iniciado, receber como entrada 3 arquivos de matrizes $n \times n$ e, a partir delas, realizar operações e gravar as novas matrizes geradas em outros arquivos de formato semelhante aos primeiros. Além disso, o programa utiliza de manipulação de threads para possibilitar que operações possam ser realizadas com mais eficiência a partir da simultaneidade da atuação dessas threads.

Aqui está o link (https://www.youtube.com/watch?v=dRHyWffX9CQ) para o vídeo no Youtube, no qual é mostrado o código fonte, a compilação do programa, trechos dos arquivos de entrada e as execuções do programa para os seis experimentos. O link também pode ser encontrado no fim deste relatório junto ao endereço do repositório GitHub.

2 DETALHAMENTO

Primeiramente, o programa deve ser capaz de receber todas as entradas necessárias logo ao ser iniciado, a partir da linha de comando. Dessa forma, os argumentos recebidos serão as únicas entradas utilizadas durante toda a execução do programa.

Devemos nos atentar ao formato do comando utilizado para iniciar o programa, que deve ser o seguinte:

./programa T n arqA.txt arqB.txt arqC.txt arqD.txt arqE.txt

Onde:

- *./programa* é o nome do programa que resolverá o problema.
- *T* é o número de threads de processamento (*tp*).
- *n* é o número de linhas e colunas das matrizes.
- *arqA.txt* é o nome do arquivo que contém a Matriz A.
- *arqB.txt* é o nome do arquivo que contém a Matriz B.
- *arqC.txt* é o nome do arquivo que contém a Matriz C.
- *arqD.txt* é o nome do arquivo que contém a matriz resultante da soma das matrizes A e B.
- *arqE.txt* é o nome do arquivo que contém a matriz resultante da multiplicação das matrizes D e C.

As threads estão divididas em três tipos:

- *tp*, thread de processamento;

- *tl*, thread de leitura;
- *te*, thread de escrita.

Na linha de comando de início do programa, o número de threads de processamento, "T", é utilizado de acordo com a Tabela 1 a seguir:

Passo	Tarefas	Tipos e quantidades de threads
Passo 1	Leitura da Matriz A // Leitura da Matriz B	2 threads do tipo *tl*
Passo 2	Soma das Matrizes A + B = D	T threads do tipo *tp*
Passo 3 // Passo 4	Gravação da Matriz D // Leitura da Matriz C	1 thread do tipo *te* e 1 thread do tipo *tl*
Passo 5	Multiplicação das Matrizes D x C = E	T threads do tipo *tp*
Passo 6 // Passo 7	Gravação da Matriz E // Redução da Matriz E	1 thread do tipo *te* e T threads do tipo *tp*

Tabela 1

Antes de finalizar sua execução, o programa deverá exibir ao usuário o valor da redução da Matriz E e os tempos gastos para a realização das operações com as matrizes, além de exibir também o tempo total.

3 COMPILAÇÃO DO PROGRAMA

Para compilar o programa há um arquivo *makefile* junto aos arquivos do programa no repositório GitHub (https://github.com/miguelgomesf/TrabalhoSO), que deve ser executado através do comando *make* no terminal.

4 RELATÓRIO DE TESTES

Seguindo as especificações do projeto, realizamos os testes solicitados em ambiente Linux nos computadores da FT e listamos a seguir os resultados obtidos:

1. *Teste 1*

   Número de threads de processamento (T) = 1

   Tamanho das matrizes (n × n) = 100

   Redução: 50292002

   Tempo soma: 0.000047 segundos.

   Tempo multiplicação: 0.001887 segundos.

   Tempo redução: 0.000018 segundos.

   Tempo total: 0.004494 segundos.

2. *Teste 2*

Número de threads de processamento (T) = 1

Tamanho das matrizes (n × n) = 1000

Redução: 49988426711

Tempo soma: 0.003959 segundos.

Tempo multiplicação: 1.776460 segundos.

Tempo redução: 0.001192 segundos.

Tempo total: 1.973131 segundos.

3. *Teste 3*

   Número de threads de processamento (T) = 2

   Tamanho das matrizes (n × n) = 100

   Redução: 50292002

   Tempo soma: 0.000067 segundos.

   Tempo multiplicação: 0.000065 segundos.

   Tempo redução: 0.000058 segundos.

   Tempo total: 0.001063 segundos.

4. *Teste 4*

   Número de threads de processamento (T) = 2

   Tamanho das matrizes (n × n) = 1000

   Redução: 49988426711

   Tempo soma: 0.000092 segundos.

   Tempo multiplicação: 2.310736 segundos.

   Tempo redução: 0.000755 segundos.

   Tempo total: 2.504940 segundos.

5. *Teste 5*

   Número de threads de processamento (T) = 4

   Tamanho das matrizes (n × n) = 100

   Redução: 50292002

   Tempo soma: 0.000192 segundos.

   Tempo multiplicação: 0.000097 segundos.

   Tempo redução: 0.000084 segundos.

   Tempo total: 0.001143 segundos.

6. *Teste 6*

   Número de threads de processamento (T) = 4

Tamanho das matrizes (n × n) = 1000

Redução: 49988426711

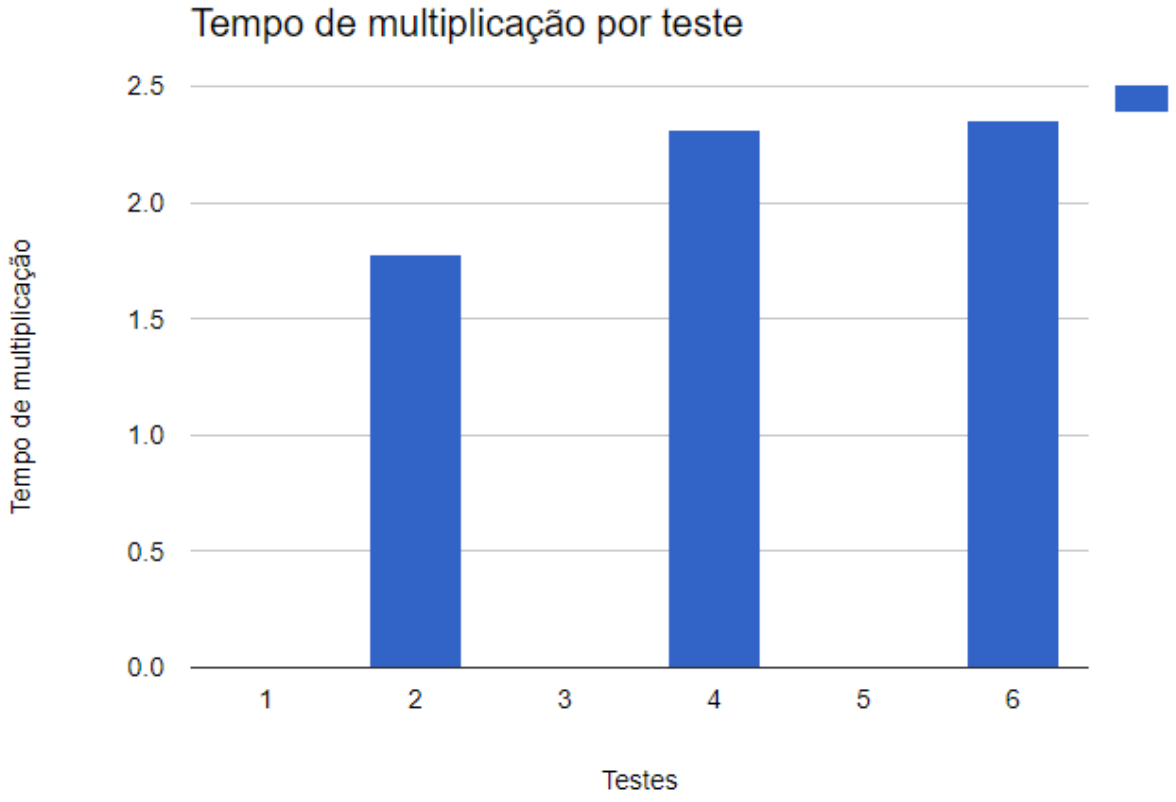
Tempo soma: 0.000161 segundos.

Tempo multiplicação: 2.352040 segundos.

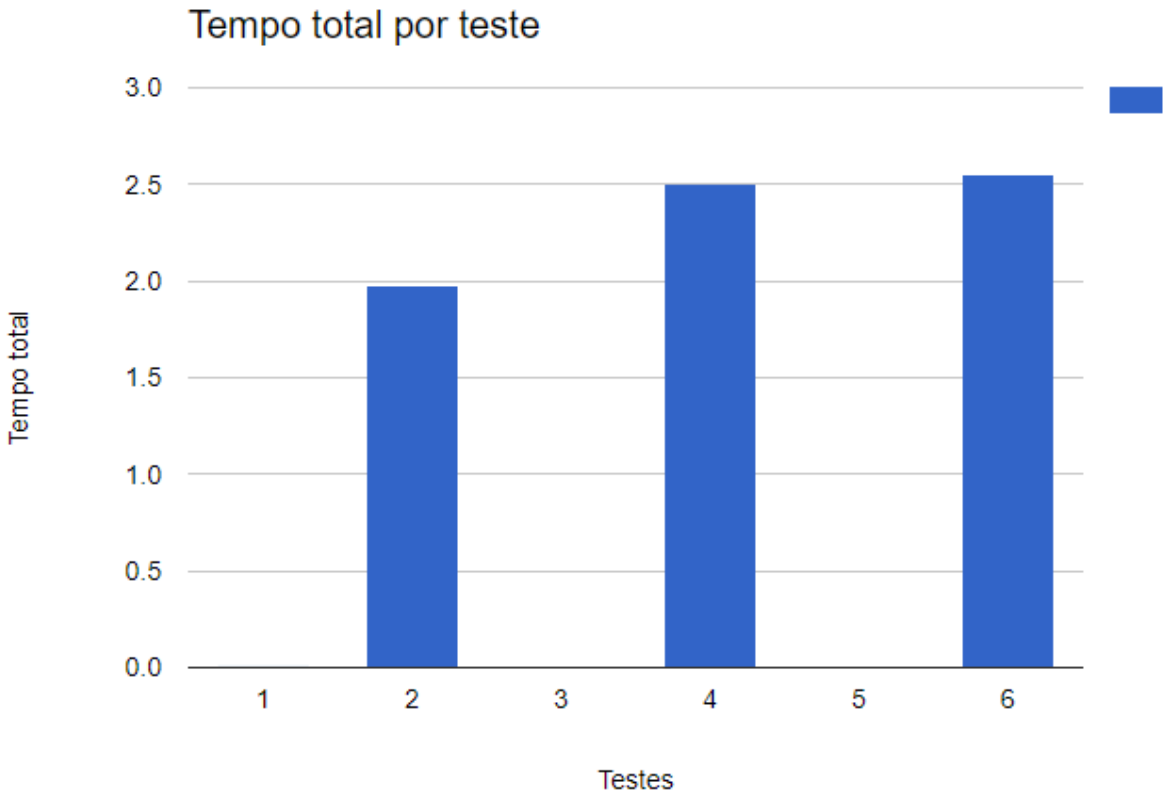
Tempo redução: 0.000107 segundos.

Tempo total: 2.548894 segundos.

4.1 Gráficos


Tempo de multiplicação por teste
Tempo de multiplicação
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
Testes


Tempo de redução por teste
Tempo de redução
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
1
2
3
4
5
6
Testes



4.2 Conclusões

A partir dos testes realizados, nota-se que na operação de soma, o uso de threads de processamento nas matrizes 1000x1000 diminui muito o tempo de processamento, enquanto para as matrizes 100x100, o uso de threads parece aumentar levemente o tempo de processamento. Já para a operação de multiplicação, é visível que, para as matrizes 1000x1000, o uso de threads aumenta o tempo de processamento da operação, e para as matrizes 100x100, o uso de threads diminui levemente o tempo. Porém, também é notável que a operação consome muito mais tempo quanto maiores são as matrizes, diferente da operação de soma em que o tempo aumenta levemente das matrizes 100x100 para as matrizes 1000x1000.

Para a operação de redução, observa-se uma grande diminuição no tempo de processamento das matrizes 1000x1000 com o uso de threads, enquanto para as matrizes 100x100 há um leve aumento no tempo ao usar threads. Por fim, o tempo total foi maior nas matrizes 100x100 quando foi usada apenas a thread principal, apesar da diferença ser muito pequena quando se compara a execução com uso de threads de processamento. Já quando

foram usadas as matrizes 1000x1000, o uso de threads além da principal tornou o tempo de processamento total maior.

Ao analisar as diferentes execuções, é visível que, além do uso das threads de processamento, o tamanho da matriz manipulada e o tipo de operação influenciam no tempo de processamento do programa. Conclui-se também que as threads podem tanto aumentar quanto diminuir o tempo de processamento das operações, de acordo com as condições nas quais elas são usadas. Para a execução do programa com matrizes 1000x1000, pode-se responsabilizar o maior tempo de processamento das execuções com uso de threads ao fato de que a operação de multiplicação com threads de processamento tem seu tempo consideravelmente aumentado, já que nas outras operações, o uso de threads torna o tempo de processamento menor. Já na execução com matrizes 100x100, o tempo de processamento não apresenta diferenças significativas com ou sem uso de threads além da principal. Contudo, conclui-se que o uso de threads torna o programa levemente mais rápido também devido à operação de multiplicação, que faz a operação ser mais rápida com matrizes desse tamanho.

5 LINKS DO PROJETO

Link do repositório GitHub: https://github.com/miguelgomesf/TrabalhoSO

Link do vídeo: https://www.youtube.com/watch?v=dRHyWffX9CQ