Universidade Estadual de Campinas - UNICAMP
Faculdade de Tecnologia - FT

Relatório - Trabalho final da disciplina

T_TT304A_2024S1 - Sistemas Operacionais

Docente:
André Leon S. Gravdhol Dr.

DIscentes:
Eric Henrique Ramos 174871
João Vitor Abrahão de Moraes 205903

Problema a ser resolvido:

No trabalho nos foi dada a tarefa de resolver um problema envolvendo leituras, gravações e operações matemáticas de matrizes na linguagem de programação C pura. Para a realizar o que foi pedido antes, precisamos criar threads para otimizar a atividade, as threads devem seguir o seguinte formato:

1. Thread de processamento (Tp): Tem a finalidade de realizar as operações matemáticas exigidas.
2. Thread de leitura (Tl): Tem a finalidade de ler os arquivos.
3. Thread de escrita(Te): Tem a finalidade de gravar as matrizes em arquivos.

Além disso, também contamos com cinco matrizes (A,B,C, D e E) que devem seguir a seguintes operações:

1. Leitura de A e B (2 Tl);
2. Somar A + B = D (T threads de Tp) ;
3. Gravar D e leitura de C (1 Te e 1 Tl);
4. Calcular D * C = E (T threads de Tp);
5. Gravar, reduzir por soma e printar E(1 Te e T Tp);

Segue o link para o vídeo do nosso trabalho:
https://www.youtube.com/watch?v=ejY8uT7Z6jI
Link para o repositório do github: https://github.com/AbrahaoDev/SO
No repositório há 4 arquivos: trabalhofinal.c (trabalho com todas as threads requeridas) e o código dos 3 testes (1, 2 e 4 threads)

Instruções para compilação do programa

Para compilar o arquivo, é necessário usar o compilador gcc com o seguinte comando:

```
gcc trabalhofinal.c -o nome_do_executavel -lm -pthread
```

Além disso, é necessário que os arquivos contendo as matrizes estejam na mesma pasta do arquivo.

PARTE TÉCNICA:

Os testes foram realizados no Linux, mais especificamente na versão Ubuntu 22.04.1 LTS e na versão Debian 12.
No Ubuntu o processador utilizado foi o Intel® Core™ i5-4590 CPU @ 3.30GHz × 4 (4 núcleos e 4 threads), já no Debian foi o AMD Ryzen 5 7520U with Radeon Graphics (8) (4 núcleos e 8 threads).

EXPERIMENTOS:

Além do projeto principal, também foi exigido a realização de experimentos envolvendo threads nos quais devemos trabalhar com dois tipos diferentes de matrizes, uma delas sendo 100x100 e a outra 1000x1000 tratando cada uma com as seguintes quantidades de threads: 1, 2 e 4.

Segue abaixo os resultados da execução com o AMD Ryzen 5 7520U:
Matriz 100x100 com uma thread:

```
abrahao@DESKTOP-5GKBQKC:~/Documents$ gcc v1.c -o trab -lm -pthread
abrahao@DESKTOP-5GKBQKC:~/Documents$ ./trab
A função produto() gastou 0.004556 segundos.
2000000
O programa gastou 0.009284 segundos no total.
abrahao@DESKTOP-5GKBQKC:~/Documents$
```

Matriz 100x100 com duas threads:

```
abrahao@DESKTOP-5GKBQKC:~/Documents$ gcc v2.c -o trab -lm -pthread
abrahao@DESKTOP-5GKBQKC:~/Documents$ ./trab
A função soma() gastou 0.000179 segundos.
A função produto() gastou 0.005626 segundos.
O programa gastou 0.009620 segundos no total.
abrahao@DESKTOP-5GKBQKC:~/Documents$
```

Matriz 100x100 com quatro threads:

```
abrahao@DESKTOP-5GKBQKC:~/Documents$ gcc v4.c -o trab -lm -pthread
abrahao@DESKTOP-5GKBQKC:~/Documents$ ./trab
A função soma() gastou 0.000084 segundos.
A função produto() gastou 0.003737 segundos.
A função writee() gastou 0.000704 segundos.
A função reduc() gastou 0.000018 segundos.
2000000
O programa gastou 0.010017 segundos no total.
```

Matriz 1000x1000 com uma thread:

```
abrahao@DESKTOP-5GKBQKC:~/Documents$ gcc v1.c -o trab -lm -pthread
abrahao@DESKTOP-5GKBQKC:~/Documents$ ./trab
A função produto() gastou 3.911882 segundos.
2000000000
O programa gastou 4.161394 segundos no total.
abrahao@DESKTOP-5GKBQKC:~/Documents$
```

Matriz 1000x1000 com duas threads:

```
abrahao@DESKTOP-5GKBQKC:~/Documents$ gcc v2.c -o trab -lm -pthread
abrahao@DESKTOP-5GKBQKC:~/Documents$ ./trab
A função soma() gastou 0.019856 segundos.
A função produto() gastou 4.665221 segundos.
O programa gastou 4.812730 segundos no total.
abrahao@DESKTOP-5GKBQKC:~/Documents$
```

Matrix 1000x1000 com quatro threads:

```
abrahao@DESKTOP-5GKBQKC:~/Documents$ gcc v4.c -o trab -lm -pthread
abrahao@DESKTOP-5GKBQKC:~/Documents$ ./trab
A função soma() gastou 0.003896 segundos.
A função produto() gastou 3.402850 segundos.
A função writee() gastou 0.052305 segundos.
A função reduc() gastou 0.002757 segundos.
2000000000
O programa gastou 3.661990 segundos no total.
```

Utilizando o Intel® Core™ i5-4590 obtivemos os seguintes resultados:

Com apenas uma thread na matriz 100x100:

```
j205903@li074:~/Downloads$ gcc v1.c -o trab -lm -pthread
j205903@li074:~/Downloads$ ./trab
A função produto() gastou 0.008557 segundos.
2000000
O programa gastou 0.019936 segundos no total.
j205903@li074:~/Downloads$
```

Com duas threads para 100x100:

```
e174871@ll068:~/Downloads$ gcc v2.c -o trab -lm -pthread
e174871@ll068:~/Downloads$ ./trab
A função soma() gastou 0.000243 segundos.
A função produto() gastou 0.005267 segundos.
O programa gastou 0.007607 segundos no total.
```

Com quatro threads para 100x100:

```
e174871@ll068:~/Downloads$ gcc v4.c -o trab -lm -pthread
e174871@ll068:~/Downloads$ ./trab
A função soma() gastou 0.000070 segundos.
A função produto() gastou 0.006220 segundos.
A função writee() gastou 0.000995 segundos.
A função reduc() gastou 0.000033 segundos.
12000000
O programa gastou 0.009872 segundos no total.
```

Uma thread 1000x1000:

```
j205903@li074:~/Downloads$ gcc v1.c -o trab -lm -pthread
j205903@li074:~/Downloads$ ./trab
A função produto() gastou 7.349174 segundos.
2000000000
O programa gastou 7.727077 segundos no total.
```

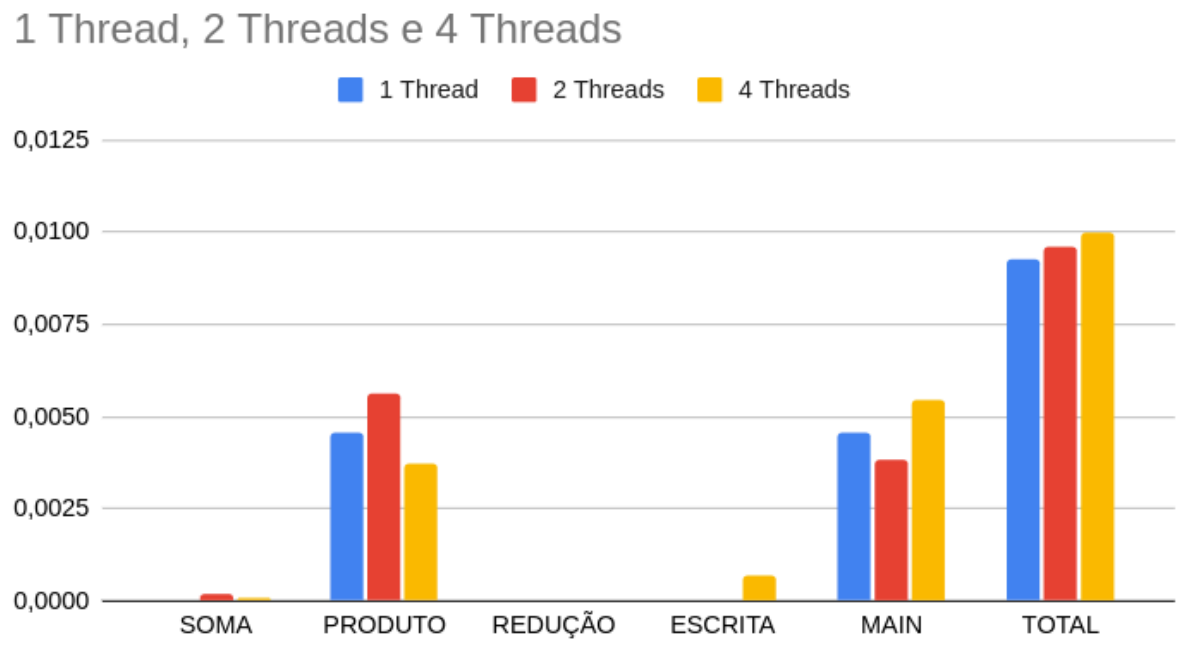
Duas threads 1000x1000:

```
e174871@ll068:~/Downloads$ gcc v2.c -o trab -lm -pthread
e174871@ll068:~/Downloads$ ./trab
A função soma() gastou 0.015889 segundos.
A função produto() gastou 6.165037 segundos.
O programa gastou 6.348196 segundos no total.
e174871@ll068:~/Downloads$
```

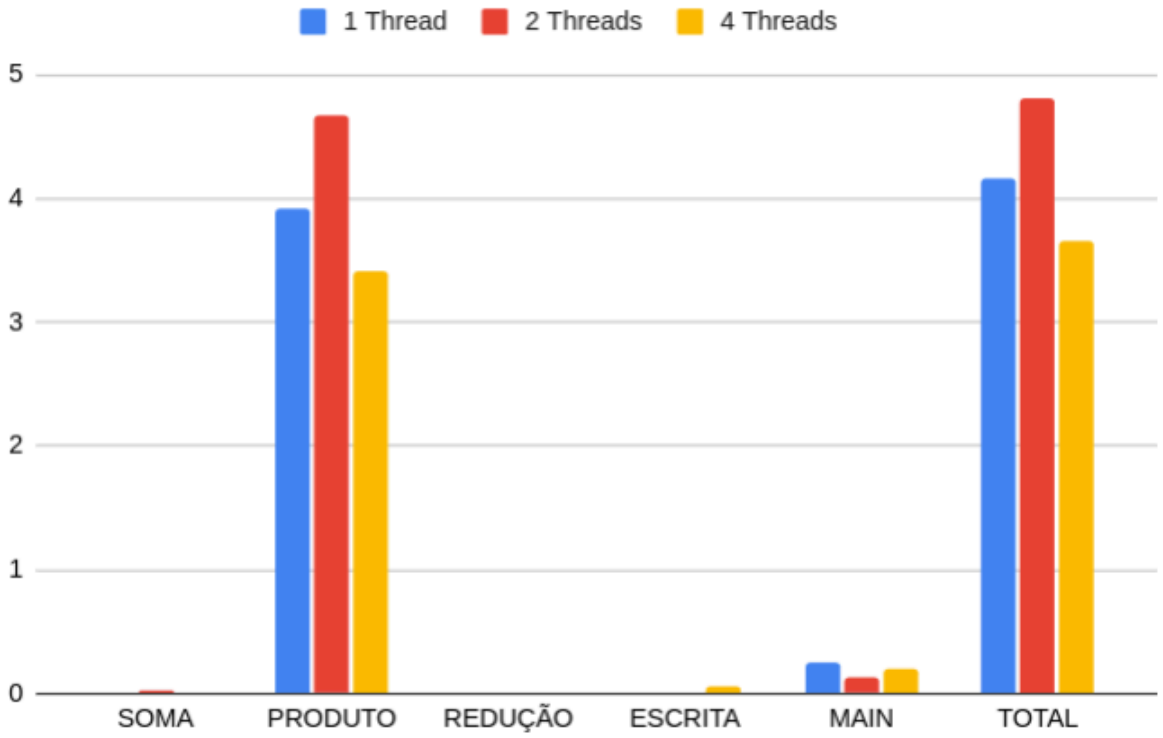
Quatro threads 1000x1000:

```
e174871@ll068:~/Downloads$ gcc v4.c -o trab -lm -pthread
e174871@ll068:~/Downloads$ ./trab
A função soma() gastou 0.004843 segundos.
A função produto() gastou 4.008778 segundos.
A função writee() gastou 0.066988 segundos.
A função reduc() gastou 0.002295 segundos.
-884901888
O programa gastou 4.326113 segundos no total.
```

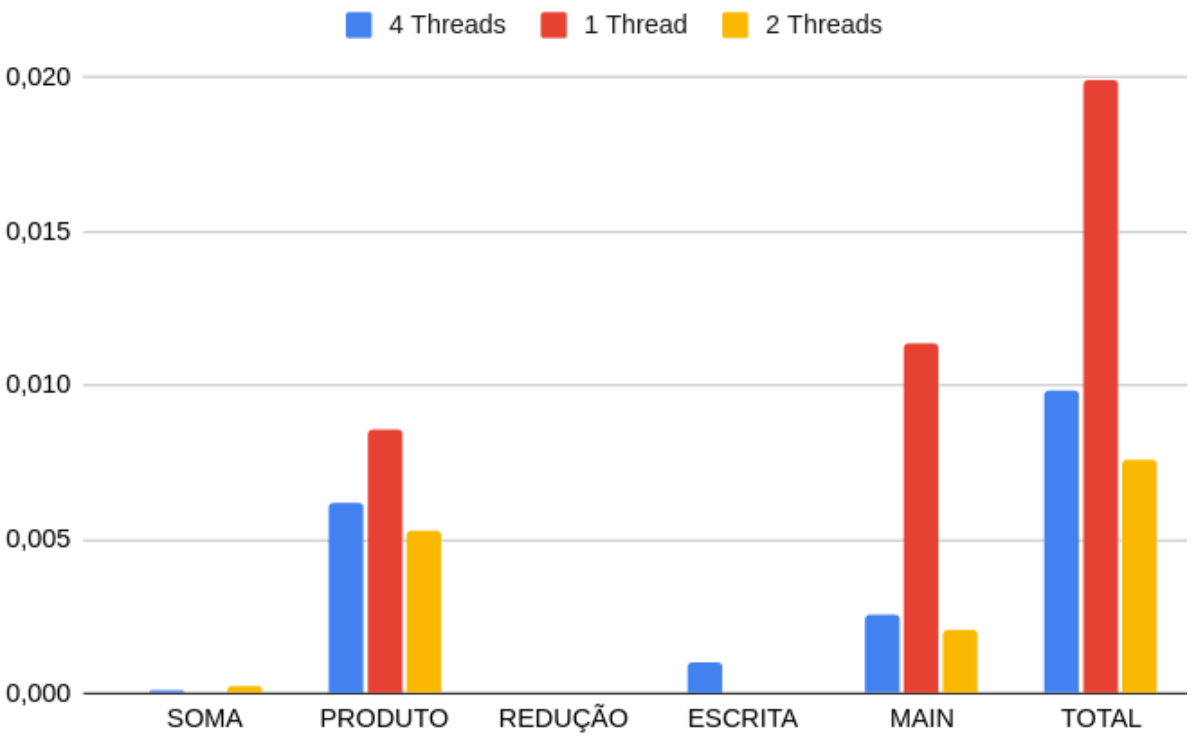
Aqui estão alguns gráficos para facilitar o entendimento do que foi feito:



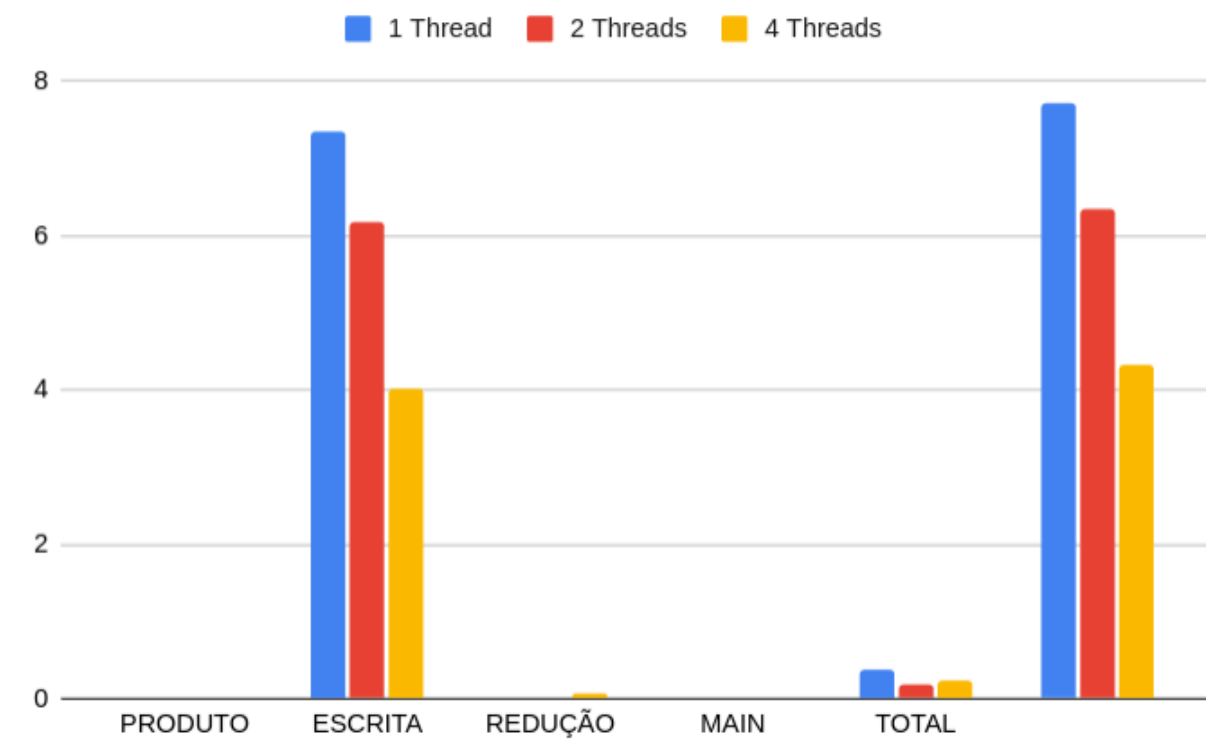
AMD Ryzen 5 7520U 100x100



AMD Ryzen 5 7250U 1000x1000



Intel® Core™ i5-4590 100x100



Intel® Core™ i5-4590 1000x1000

CONCLUSÃO:

Com tudo isso em mente chegamos a conclusão que as operações com o AMD Ryzen 5 7250U não nos deram resultados muito claros, porém com o Intel® Core™i5-4590, mesmo com menos threads disponíveis, obtivemos resultados que nos mostram que o tempo é inversamente proporcional ao número de threads utilizadas no processo. Vale ressaltar também como o trabalho nos mostrou como a utilização de threads é importante na linguagem C, vide quando usamos o IntelCore na matrix 1000 por 1000 onde a diferença de uma thread quando comparada com quatro thread temos uma diferença de 50% no tempo gasto.