**Universidade Estadual de Campinas - FT**
**Data**

# Relatório
# Trabalho de Sistemas Operacionais

Disciplina: [TT304] - Sistemas Operacionais
1º semestre de 2024
Prof. Dr. André Leon S. Gradvohl

Alunos:
José Vitor Dutra Antônio - 187174
Nicolas Suzuki - 173762

# Descrição do Problema

Neste trabalho, nosso desafio era desenvolver um programa em linguagem C que utiliza-se alocação dinâmica de memória e threads para realizar alguns cálculos de matrizes NxN.

Conforme o enunciado: Dadas três matrizes de entrada, $An\times n$,$Bn\times n$ e $Cn\times n$, o programa deverá calcular inicialmente a matriz $Dn\times n$, tal que $Dn\times n$ = ($An\times n$ + $Bn\times n$) e gravá-la em arquivo. Em seguida, o programa deverá calcular a matriz $En\times n$tal que $En\times n$ = ($Cn\times n$ × $Dn\times n$). Por último, a matriz $En\times n$ deve ser gravada em arquivo e reduzida por soma

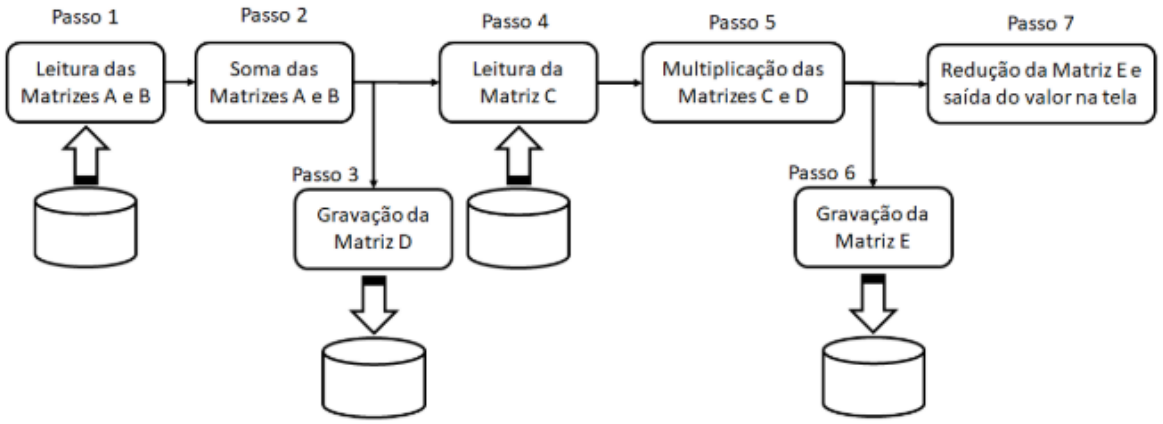



*Figura 1 Passos para a execução do programa*

# Compilação

Os requerimentos de compilação desse programa são: utilizar um ambiente Linux (neste trabalho utilizamos Linux Mint 21.3, WSL também se mostrou funcional para execução e compilação) e ter instalado o compilador GCC (GNU Compiler Collection).
Tendo o projeto baixado em sua máquina, basta acessar a raiz do projeto e executar os seguintes comandos.

**Compilando código fonte**

```
Unset
gcc -o matrix_operations trabalho_so.c -lpthread
```

**Executando**

```
Unset
./matrix_operations T n arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
```

Substitua **T** pelo número de threads desejada, e **n** pelo tamanho da matriz.

**Alternativa utilizando Makefile**
Para facilitar o processo de compilação é possível utilizar o arquivo Makefile, basta executar o comando abaixo e então repetir o passo de execução:

```
Unset
Make
```

# Experimento

Para este trabalho produzimos alguns experimentos a fim de observar o comportamento do programa e das threads com diferentes entradas, sendo elas:

- Número de threads de processamento (T): T = 1, T = 2 e T = 4 threads
- Tamanho das matrizes ($n \times n$): $n$ = 100 e $n$ = 1000

**Nossos resultados:**

Entrada para: T = 1, n = 100

```
ze@ze-VirtualBox:~/Área de Trabalho/SistemasOperacionais$ ./matrix_operations 1 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Reducao: 665
Tempo Soma: 0.000741 segundos
Tempo Multiplicacao: 0.009880 segundos
Tempo Reducao: 0.001114 segundos
Tempo total: 0.022724 segundos
```

Entrada para: T = 2, n = 100

```
ze@ze-VirtualBox:~/Área de Trabalho/SistemasOperacionais$ ./matrix_operations 2 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Reducao: 665
Tempo Soma: 0.000560 segundos
Tempo Multiplicacao: 0.010895 segundos
Tempo Reducao: 0.002625 segundos
Tempo total: 0.028249 segundos
```

Entrada para: T = 4, n = 100

```
ze@ze-VirtualBox:~/Área de Trabalho/SistemasOperacionais$ ./matrix_operations 4 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Reducao: 665
Tempo Soma: 0.002264 segundos
Tempo Multiplicacao: 0.011064 segundos
Tempo Reducao: 0.002400 segundos
Tempo total: 0.024603 segundos
```

Entrada para: T = 1, n = 1000

```
ze@ze-VirtualBox:~/Área de Trabalho/SistemasOperacionais$ ./matrix_operations 1 1000 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Reducao: 665
Tempo Soma: 0.004721 segundos
Tempo Multiplicacao: 9.445291 segundos
Tempo Reducao: 0.005520 segundos
Tempo total: 9.899024 segundos
```

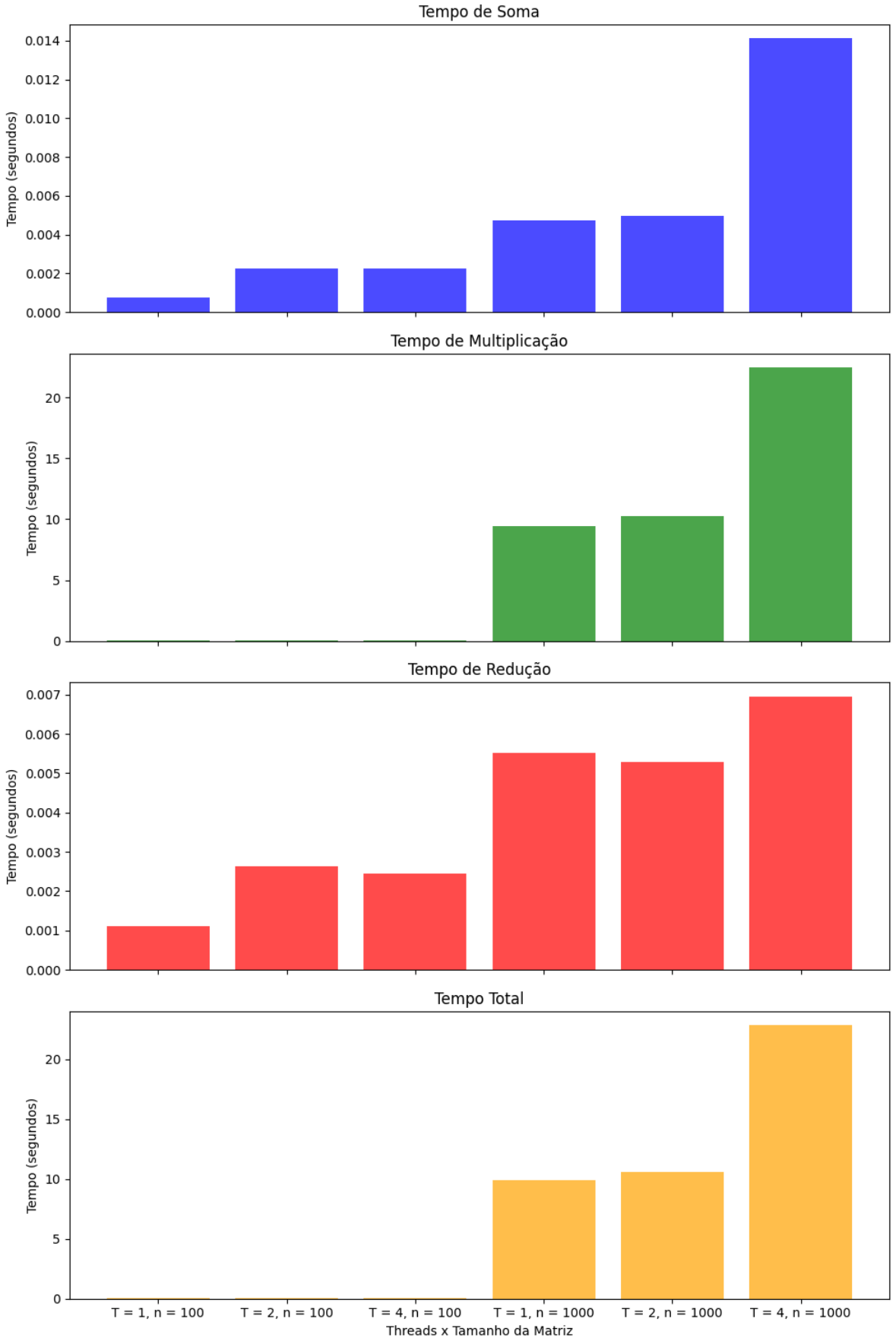
Entrada para: T = 2, n = 1000

```
ze@ze-VirtualBox:~/Área de Trabalho/SistemasOperacionais$ ./matrix_operations 2 1000 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Reducao: 665
Tempo Soma: 0.004940 segundos
Tempo Multiplicacao: 10.252611 segundos
Tempo Reducao: 0.005290 segundos
Tempo total: 10.595491 segundos
```

Entrada para: T = 4, n = 1000

```
ze@ze-VirtualBox:~/Área de Trabalho/SistemasOperacionais$ ./matrix_operations 4 1000 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Reducao: 665
Tempo Soma: 0.014112 segundos
Tempo Multiplicacao: 22.485454 segundos
Tempo Reducao: 0.006958 segundos
Tempo total: 22.870001 segundos
```

# Resultados


Tempo de Soma
Tempo (segundos)
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Tempo de Multiplicação
Tempo (segundos)
20
15
10
5
0
Tempo de Redução
Tempo (segundos)
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Tempo Total
Tempo (segundos)
20
15
10
5
0
T = 1, n = 100
T = 2, n = 100
T = 4, n = 100
T = 1, n = 1000
T = 2, n = 1000
T = 4, n = 1000
Threads x Tamanho da Matriz

# Conclusão

Com base nos dados fornecidos, podemos observar o impacto da quantidade de threads no tempo de execução das operações de soma, multiplicação e redução de matrizes, além do tempo total.

Contrário ao que esperávamos, o aumento do número de threads não leva necessariamente a uma diminuição do tempo de execução. Para matrizes menores (100x100), o impacto das threads é menor, e os tempos de execução não variam significativamente. No entanto, para matrizes maiores (1000x1000), o uso de mais threads parece aumentar o tempo de execução.

## Links externos

Código fonte: https://github.com/nicksuzuk1/SistemasOperacionais
Vídeo: https://www.youtube.com/watch?v=F_zoOyrUypQ