# Relatório de Avaliação
# (Programação Concorrente - Laboratório 3)

**Matheus da Cruz Percine Pinto (DRE: 121068501)**
**Universidade Federal do Rio de Janeiro**

## 1. Introdução

Este relatório mostra e analisa dados relativos a programas que objetivam realizar a multiplicação de duas matrizes de tamanho NxM, por meio de tabela com dados, gráficos de aceleração e eficiência e configuração da máquina usada. Neste relatório, compara-se a versão sequencial e concorrente. Quanto à forma concorrente, foi utilizado o método separação por linha alternada. Foram testadas multiplicações entre matrizes de tamanho 500x500, 1000x1000 e 2000x2000 para 1, 2, 4 e 8 threads. As médias foram calculadas por meio de uma média aritmética e seus valores foram considerados para geração de gráficos e cálculos de aceleração e eficiência.

Fórmulas usadas para o cálculo de aceleração eficiência:

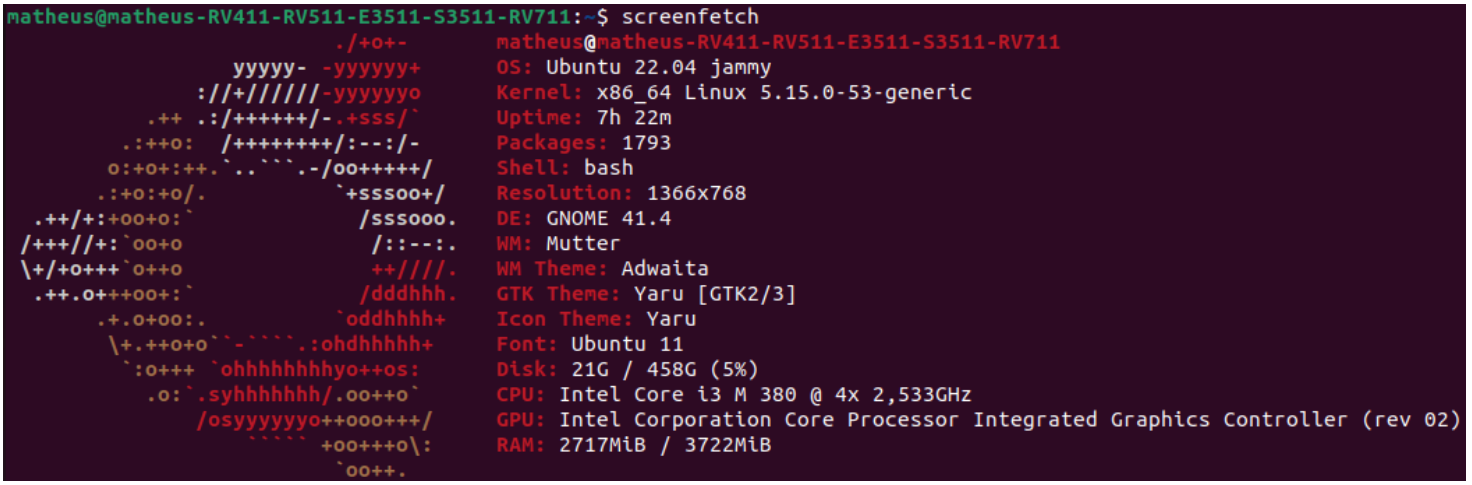
$$A = \frac{T_{sequencial}}{T_{paralelo}}$$

- $A$: Aceleração
- $T_{sequencial}$: Tempo de execução da versão sequencial do programa
- $T_{paralelo}$: Tempo de execução da versão paralela do programa

$$E = \frac{A}{P}$$

- $E$: Eficiência
- $A$: Aceleração
- $P$: Número de processadores ou threads

## 2. Configuração da Máquina Usada




## 3. Tabela de dados

| SEQUENCIAL > | | | | | | |
|---|---|---|---|---|---|---|
| | 2000x2000 | | execução 1 | execução 2 | execução 3 | |
| | | tempo inicialização | 0. 113000 | 0. 97707 | 0. 109561 | média: 0.395226 |
| | | tempo multiplicação | 123 . 452775 | 123. 279830 | 115. 203130 | média: 120.645245 |
| | | tempo finalização | 0. 229301 | 0. 223504 | 0. 220776 | média: 0.224527 |
| | | tempo total | 123. 795076 | 123. 601040 | 115. 533467 | média: 120.976527 |
| | | | | | | |
| | 1000x1000 | | execução 1 | execução 2 | execução 3 | |
| | | tempo inicialização | 0. 036119 | 0. 038047 | 0. 037792 | media: 0.037319 |
| | | tempo multiplicação | 14. 640248 | 14. 727953 | 15. 538423 | media: 14.968875 |
| | | tempo finalização | 0. 015142 | 0. 011784 | 0. 011386 | media: 0.012771 |
| | | tempo total | 14. 691509 | 14. 777785 | 15. 587601 | media: 15.018965 |
| | | | | | | |
| | 500x500 | | execução 1 | execução 2 | execução 3 | |
| | | tempo inicialização | 0. 005717 | 0. 007625 | 0. 010639 | media: 0.007994 |
| | | tempo multiplicação | 1. 438722 | 1. 537857 | 1. 568133 | media: 1.514904 |
| | | tempo finalização | 0. 004659 | 0. 002741 | 0. 003093 | media: 0.003498 |
| | | tempo total | 1. 449098 | 1. 548223 | 1. 581865 | media: 1.526395 |
| | | | | | | |

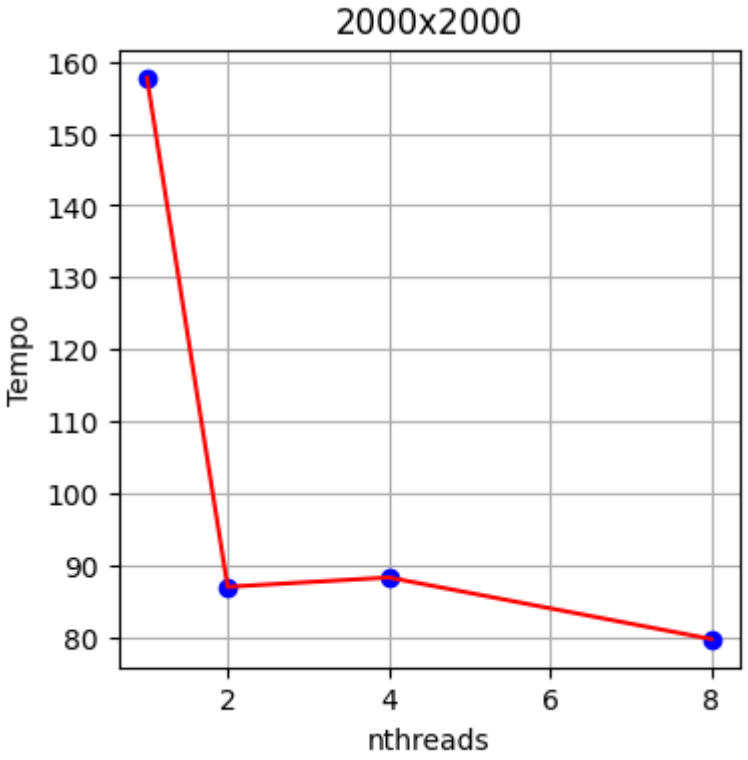
| CONCORRENTE > | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 2000x2000 (1 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | | tempo inicialização | 0. 110004 | 0. 507706 | 0. 236039 | media: 0.284583 | 0. 766785 | 0. 766785 |
| | | tempo multiplicação | 152. 661603 | 159. 751803 | 159. 310898 | media: 157.241434 | | |
| | | tempo finalização | 0. 207847 | 0. 266886 | 0. 260484 | media: 0.2450723 | | |
| | | tempo total | 152. 979454 | 160. 526395 | 159. 807421 | media: 157.771090 | | |
| | | | | | | | | |
| | 2000x2000 (2 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | | tempo inicialização | 0. 405361 | 0. 094516 | 0. 103791 | media: 0.201223 | 1. 390481 | 0. 695241 |
| | | tempo multiplicação | 86. 105040 | 84. 753949 | 88. 865253 | media: 86.574747 | | |
| | | tempo finalização | 0. 192639 | 0. 232842 | 0. 256693 | media: 0.227391 | | |
| | | tempo total | 86. 703040 | 85. 081307 | 89. 225738 | media: 87.003362 | | |
| | | | | | | | | |
| | 2000x2000 (4 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | | tempo inicialização | 0. 105708 | 0. 110750 | 0. 114392 | media: 0.110283 | 1. 370011 | 0. 342503 |
| | | tempo multiplicação | 93. 286575 | 85. 218590 | 85. 307661 | media: 87.937609 | | |
| | | tempo finalização | 0. 210865 | 0. 281583 | 0. 273907 | media: 0.255452 | | |
| | | tempo total | 93. 603148 | 85. 610923 | 85. 695960 | media: 88.303344 | | |
| | | | | | | | | |
| | 2000x2000 (8 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | | tempo inicialização | 0. 097521 | 0. 100403 | 0. 103548 | media: 0.100491 | 1. 517186 | 0. 189648 |
| | | tempo multiplicação | 77. 981612 | 87. 115128 | 80. 049917 | media: 81.715552 | | |
| | | tempo finalização | 0. 273047 | 0. 276613 | 0. 276613 | media: 0.275424 | | |
| | | tempo total | 78. 352180 | 80. 430078 | 80. 430078 | media: 79.737445 | | |

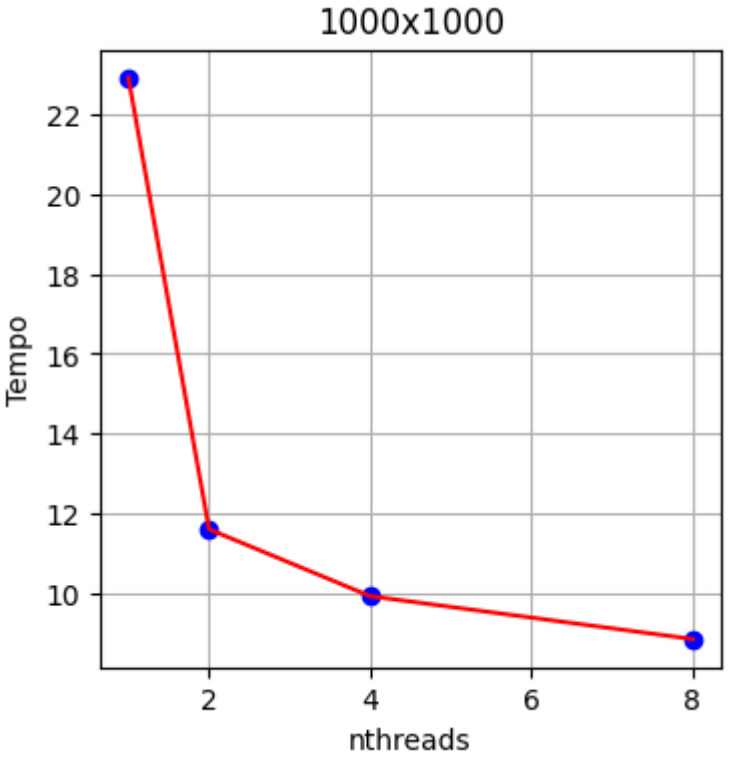
| 1000x1000 (1 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
|---|---|---|---|---|---|---|---|
| | tempo inicialização | 0. 038134 | 0. 029111 | 0. 029592 | media: 0.032279 | 0. 654773 | 0. 654773 |
| | tempo multiplicação | 22. 046073 | 23. 163880 | 23. 441231 | media: 22.883728 | | |
| | tempo finalização | 0. 040897 | 0. 012990 | 0. 011126 | media: 0.021671 | | |
| | tempo total | 22. 125105 | 23. 205981 | 23. 481949 | media: 22.937678 | | |
| | | | | | | | |
| 1000x1000 (2 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 033137 | 0. 037777 | 0. 038447 | media: 0.036454 | 1. 295548 | 0. 647774 |
| | tempo multiplicação | 11. 163379 | 11. 846618 | 11. 622576 | media: 11.544191 | | |
| | tempo finalização | 0. 013236 | 0. 011525 | 0. 011554 | media: 0.012105 | | |
| | tempo total | 11. 209753 | 11. 895920 | 11. 672577 | media: 11.592750 | | |
| | | | | | | | |
| 1000x1000 (4 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 031291 | 0. 027968 | 0. 032349 | media: 0.030536 | 1. 515389 | 0. 378847 |
| | tempo multiplicação | 9. 675988 | 9. 748426 | 10. 174658 | media: 9.866357 | | |
| | tempo finalização | 0. 014736 | 0. 015343 | 0. 012125 | media: 0.014068 | | |
| | tempo total | 9. 722016 | 9. 791737 | 10. 219132 | media: 9.910962 | | |
| | | | | | | | |
| 1000x1000 (8 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 027671 | 0. 028348 | 0. 031024 | media: 0.029014 | 1. 700233 | 0. 212529 |
| | tempo multiplicação | 9. 021923 | 9. 105754 | 8. 245832 | media: 8.791170 | | |
| | tempo finalização | 0. 014315 | 0. 013204 | 0. 012352 | media: 0.013290 | | |
| | tempo total | 9. 063909 | 9. 147306 | 8. 289208 | media: 8.833474 | | |

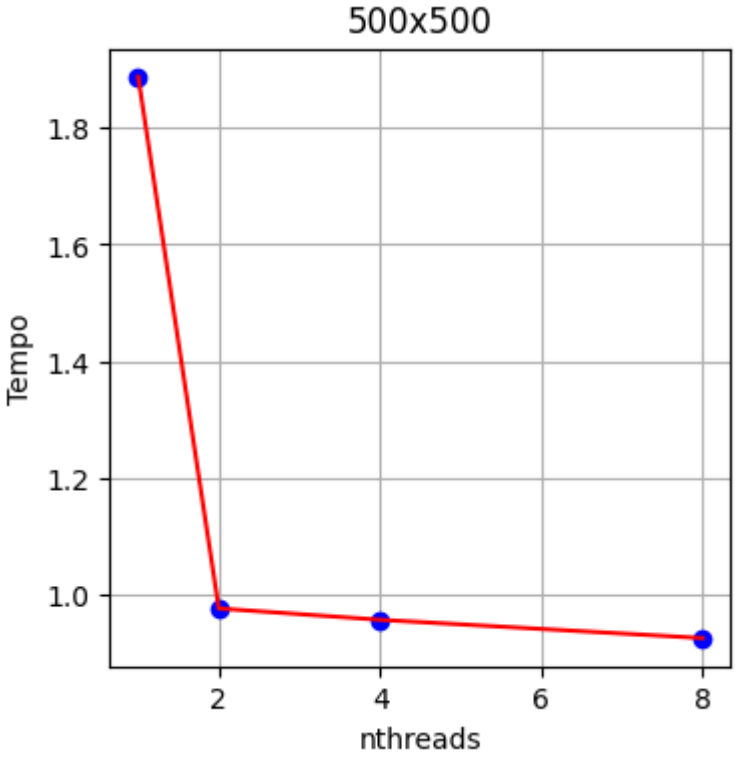
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 500x500 (1 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 010580 | 0. 007960 | 0. 008805 | media: 0.009115 | 0. 808465 | 0. 808465 |
| | tempo multiplicação | 1. 879792 | 1. 875610 | 1. 870879 | media: 1.875427 | | |
| | tempo finalização | 0. 003488 | 0. 003474 | 0. 003465 | media: 0.003476 | | |
| | tempo total | 1. 893860 | 1. 887044 | 1. 883148 | media: 1.888017 | | |
| | | | | | | | |
| 500x500 (2 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 008337 | 0. 007622 | 0. 008307 | media: 0.008089 | 1. 563635 | 0. 781817 |
| | tempo multiplicação | 0. 910583 | 0. 877834 | 1. 106299 | media: 0.964905 | | |
| | tempo finalização | 0. 003854 | 0. 002673 | 0. 003044 | media: 0.003190 | | |
| | tempo total | 0. 922774 | 0. 888129 | 1. 117650 | media: 0.976184 | | |
| | | | | | | | |
| 500x500 (4 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 007728 | 0. 006985 | 0. 007359 | media: 0.007357 | 1. 595353 | 0. 398838 |
| | tempo multiplicação | 1. 020936 | 0. 862645 | 0. 953795 | media: 0.945792 | | |
| | tempo finalização | 0. 003206 | 0. 004471 | 0. 003203 | media: 0.003627 | | |
| | tempo total | 1. 031870 | 0. 874101 | 0. 964357 | media: 0.956776 | | |
| | | | | | | | |
| 500x500 (8 thread(s)) | | execução 1 | execução 2 | execução 3 | | Aceleração | Eficiência |
| | tempo inicialização | 0. 006992 | 0. 008145 | 0. 008243 | media: 0.007793 | 1. 649451 | 0. 206181 |
| | tempo multiplicação | 0. 937843 | 0. 952915 | 0. 853812 | media: 0.914857 | | |
| | tempo finalização | 0. 002698 | 0. 002883 | 0. 002658 | media: 0.002746 | | |
| | tempo total | 0. 947532 | 0. 963943 | 0. 864713 | media: 0.925396 | | |

# 4. Gráficos

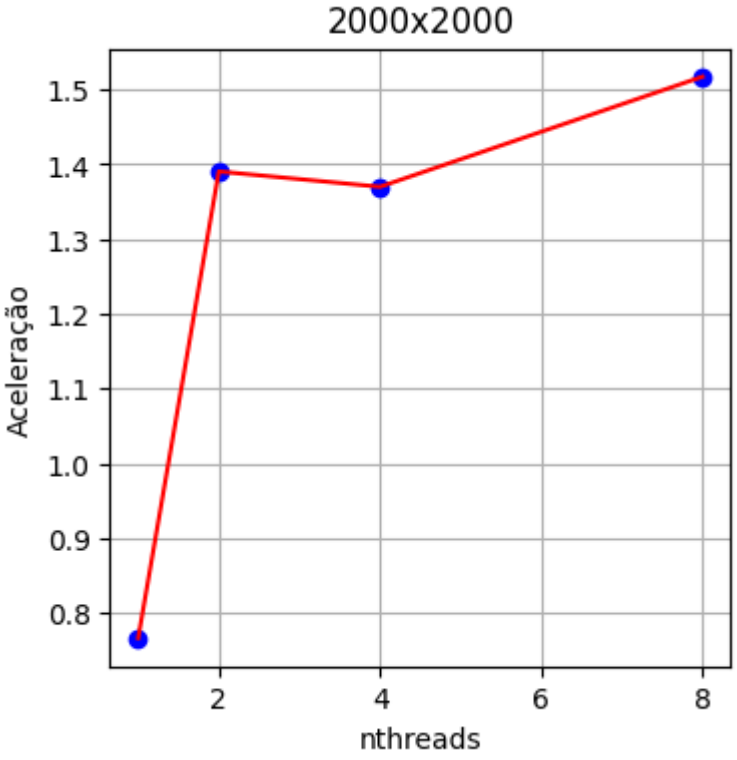
## 4.1. Gráficos de: Tempo Total de Execução x Número de Threads

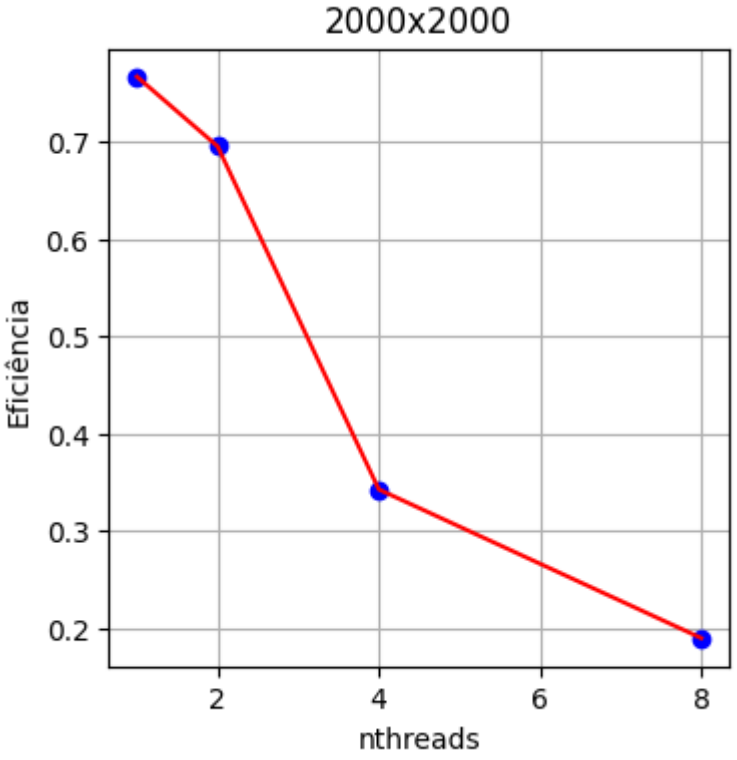


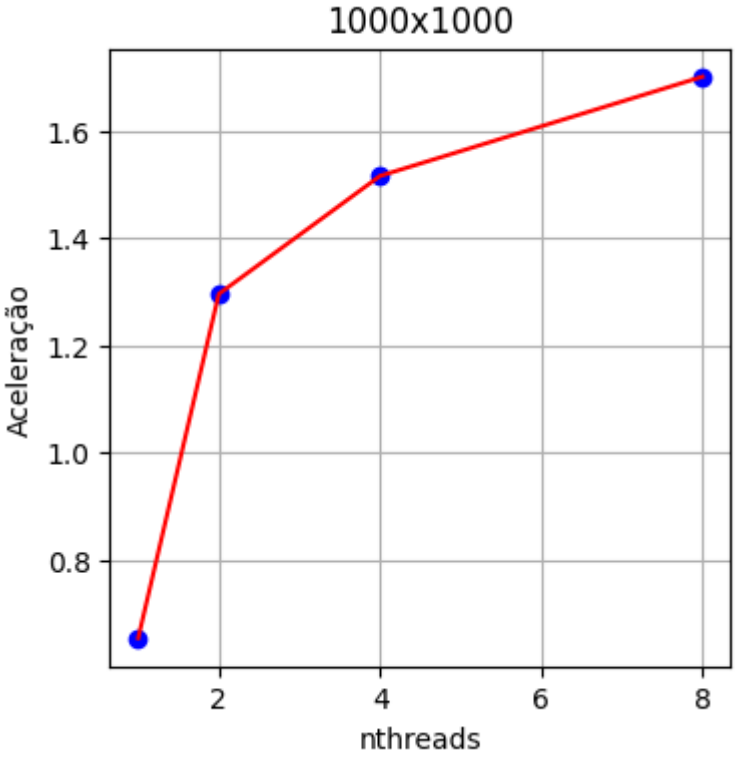



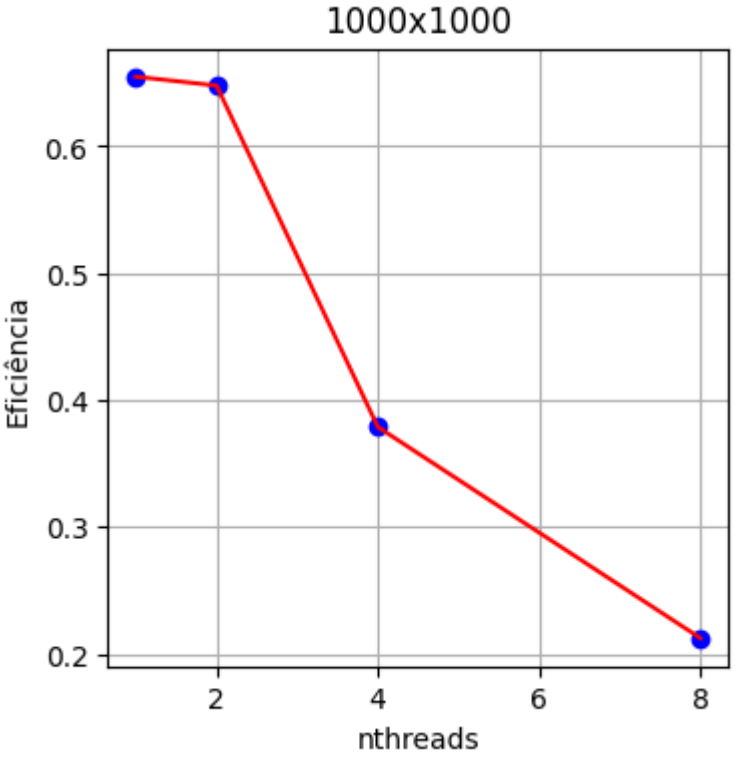


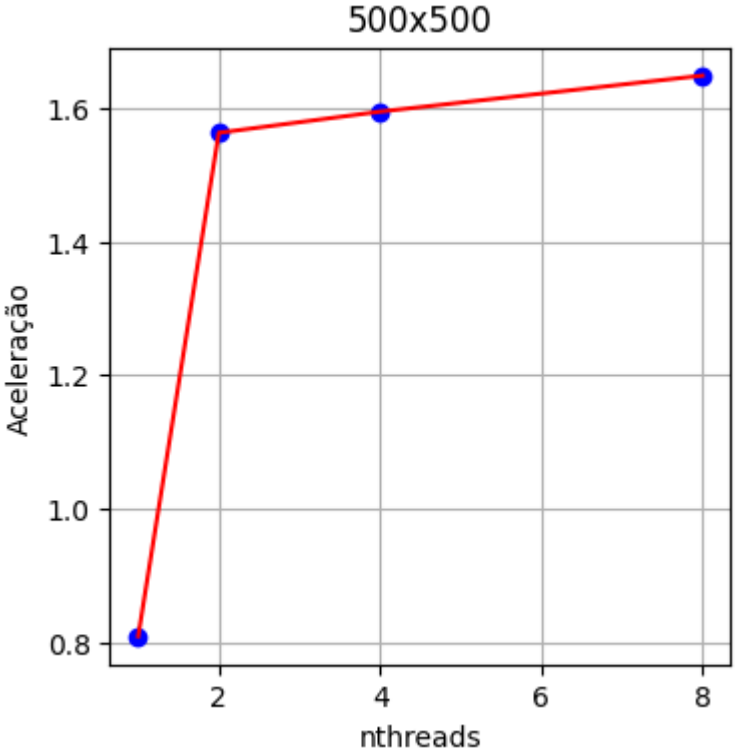

## 4.2. Gráficos de: Aceleração/Eficiência x Número de Threads

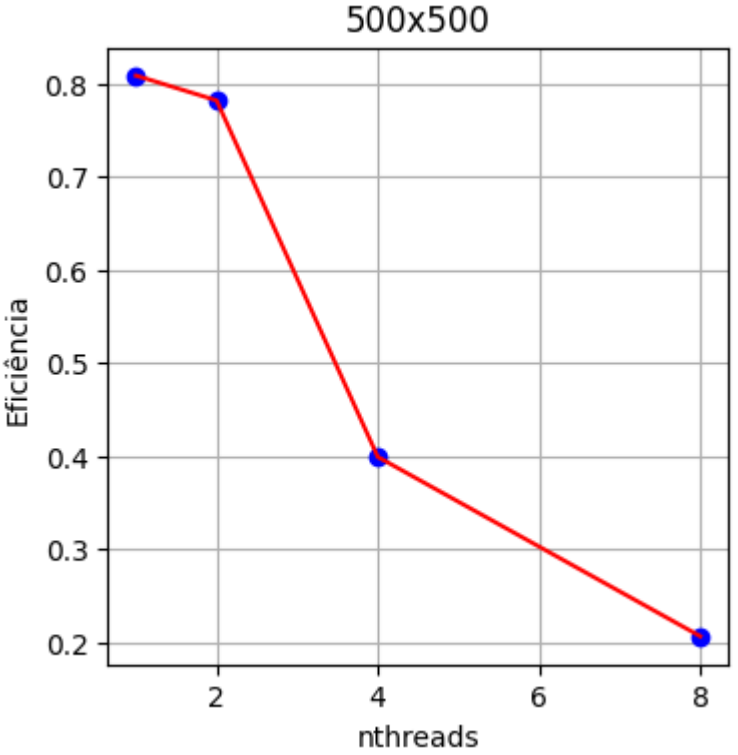



1000x1000
Aceleração
nthreads
1.6
1.4
1.2
1.0
0.8
2
4
6
8


1000x1000
Eficiência
nthreads
0.6
0.5
0.4
0.3
0.2
2
4
6
8


500x500
Aceleração
nthreads
1.6
1.4
1.2
1.0
0.8
2
4
6
8


500x500
Eficiência
nthreads
0.8
0.7
0.6
0.5
0.4
0.3
0.2
2
4
6
8

## 5. Conclusão

De forma geral, a forma concorrente demanda menos tempo de execução do que a forma sequencial, porém não podemos dizer isso no caso em que há somente 1 thread, pois há um tempo gasto na criação da thread que a forma sequencial não gasta.

Quanto aos gráficos de Tempo Total de Execução x Número de Threads, podemos observar que o tempo decai à medida que o número de threads aumenta, pois a tarefa é dividida entre vários “trabalhadores” (threads).

Quanto aos gráficos de Aceleração x Número de Threads, podemos observar que a aceleração aumenta à medida que o número de threads aumenta. Isso ocorre, pois como a aceleração é [tempo Sequencial] / [tempo Concorrente] e mantendo o numerador fixo, o denominador vai diminuindo à medida que o número de threads aumenta e, consequentemente, o valor da fração aumenta. O denominador vai diminuindo pela explicação dada no parágrafo anterior.

Quanto aos gráficos de Aceleração x Número de Threads, podemos observar que a eficiência diminui à medida que o número de threads aumenta. Isso ocorre, pois como a eficiência é [aceleração] / [número de threads], se mantermos o numerador fixo e aumentarmos o denominador, consequentemente, o valor da fração vai diminuindo. Tomando uma abordagem do ponto de vista de arquitetura de computadores, podemos dizer que a eficiência vai diminuindo por conta de overhead de criação e gerenciamento de threads e concorrência em recursos compartilhados. A criação e o gerenciamento de threads em um programa paralelo também têm um custo associado. À medida que o número de threads aumenta, o sistema operacional e a própria aplicação precisam gastar mais tempo e recursos para criar, destruir e gerenciar as threads. Esse overhead pode se tornar significativo em cenários onde o tempo de computação de cada thread é relativamente curto. Em programas paralelos que compartilham recursos, como CPUs, memória e dispositivos de E/S, o aumento do número de threads pode levar a uma competição mais intensa por esses recursos. Isso pode resultar em contenção e atrasos à medida que as threads disputam acesso aos recursos compartilhados, reduzindo assim a eficiência.