Programação Concorrente - Laboratório 3
IC/UFRJ

Nome: Nathan Andrade dos Santos Lobo
DRE: 120082390

---

1. Implementação

Foi feito um shell script para realizar a tarefa de compilar os programas auxiliares juntamente com o programa principal que realizará os cálculos concorrentes e sequencial (quando necessário) das matrizes de entrada.

Para executar o programa, basta executar as instruções que está no README na página da pasta do lab3.

2. Experimentos

Dos experimentos, foi realizado com um computador com as seguintes especificações:

Processador: AMD Ryzen 5 5500U
Núcleos: 6
Threads: 12
Memória RAM: 12GB

portanto, os dados computados foram feitos com 1, 2, 4, 8 e 12 threads, podendo serem consultados no link: https://docs.google.com/spreadsheets/d/1boX8CLNS5OWaJ3YfjAbjeDumNXpoSytMKmcMKXepnhA/edit?usp=sharing

mas que serão resumidos abaixo.

Primeiramente, importante ressaltar que durante a computação dos dados, foi gerado com a flag **diff**, permitindo o algoritmo calcular também tempos sequenciais para cada execução, de forma que fosse possível montar um gráfico com todo o detalhamento e que fosse possível interpretar a situação do tempo médio sequencial com maior facilidade.

Ficou fácil de visualizar que em todas as situações o tempo médio concorrente foi superior ao sequencial criando uma curva abaixo da reta constante nos gráficos.

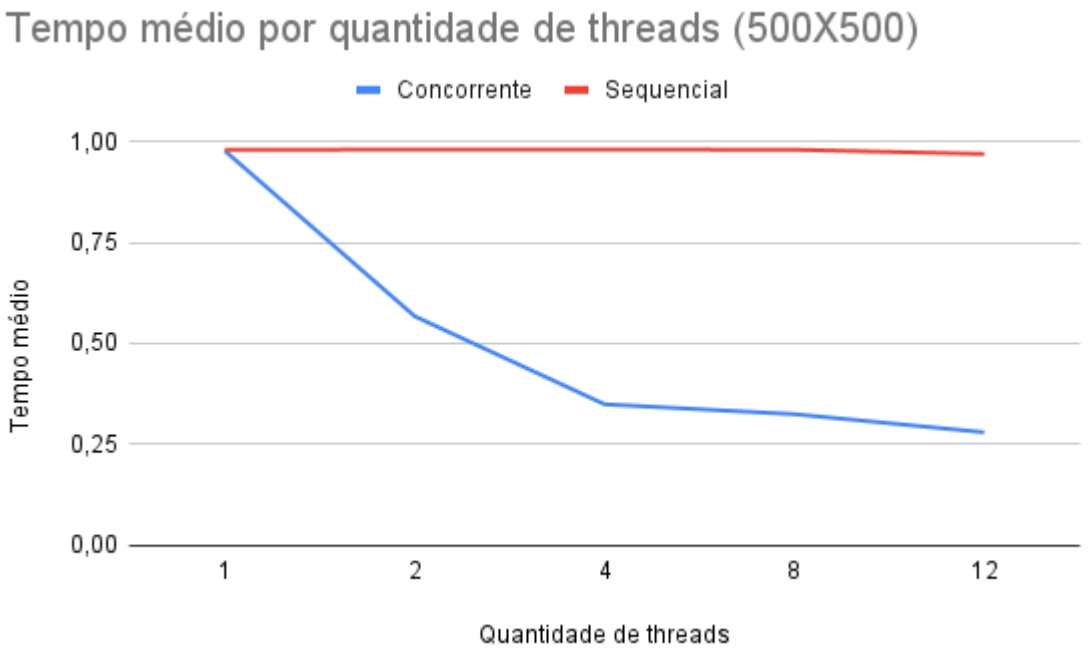
Os dados utilizados para gerar os gráficos foram: **tempo médio sequencial, tempo médio concorrente, quantidade de threads e o número de dimensão da matriz** para uma determinada execução de algoritmo que originou um arquivo de saída com os tempos em questão.

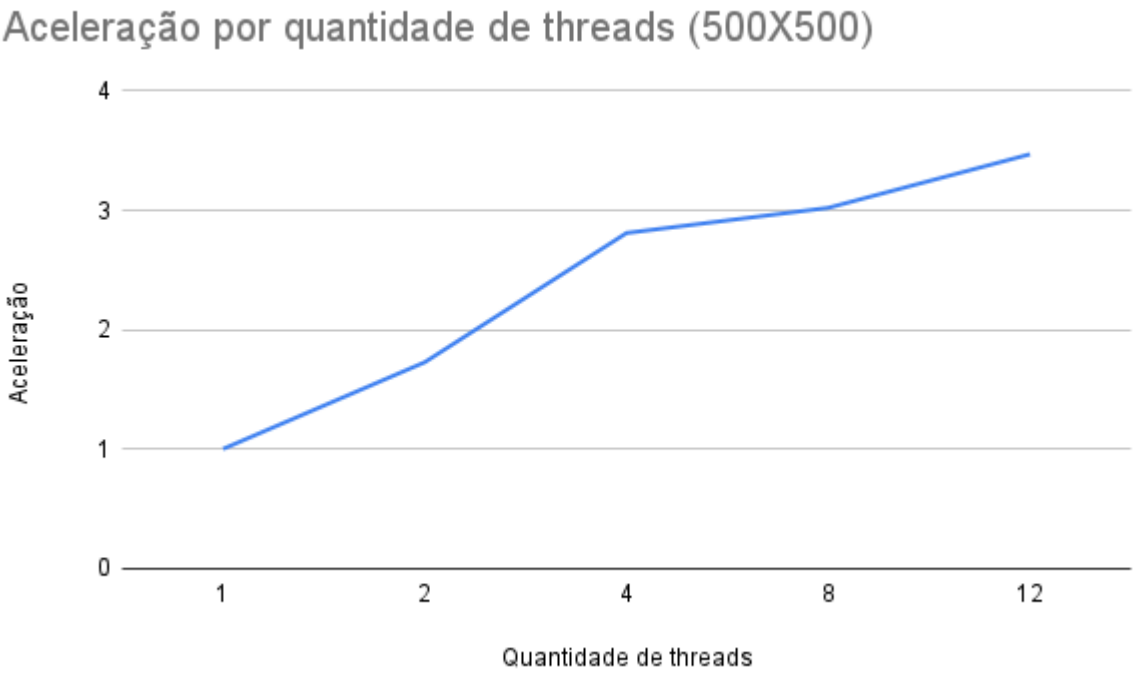
Para calcular a aceleração, foi utilizado cada tempo sequencial dividido por um tempo concorrente (variando pela quantidade de threads).

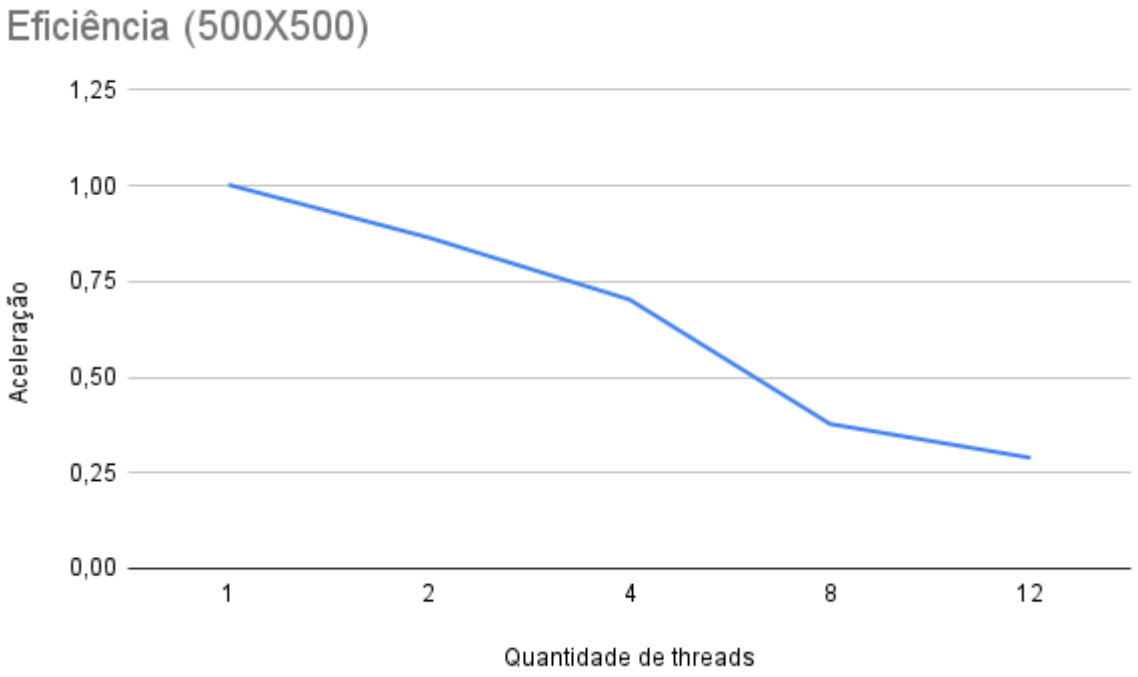
Já para calcular a eficiência, conhecendo a aceleração foi feito a divisão do valor da aceleração pela quantidade de threads relacionada a este mesmo valor.

3. Gráficos

a. Execução para matrizes de dimensão 500X500

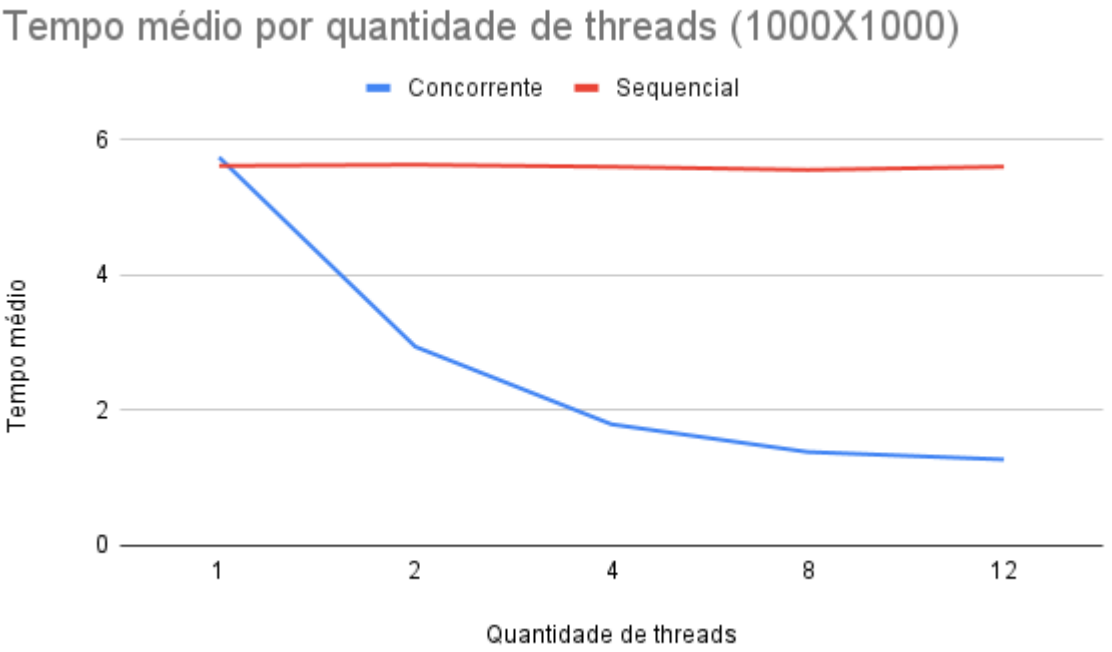




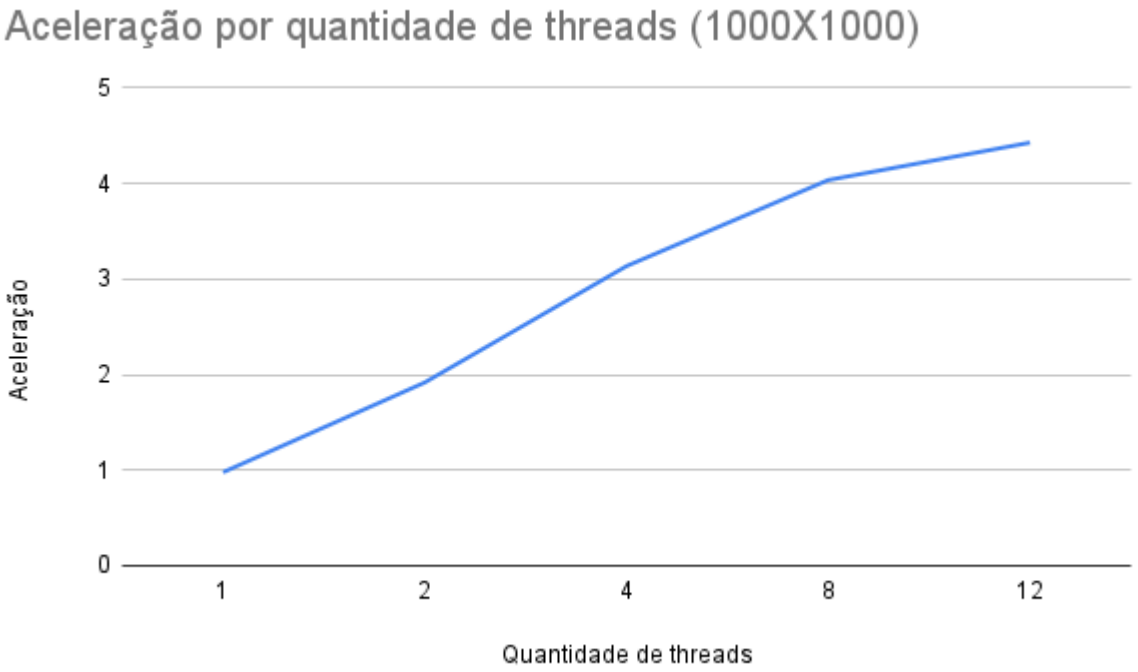


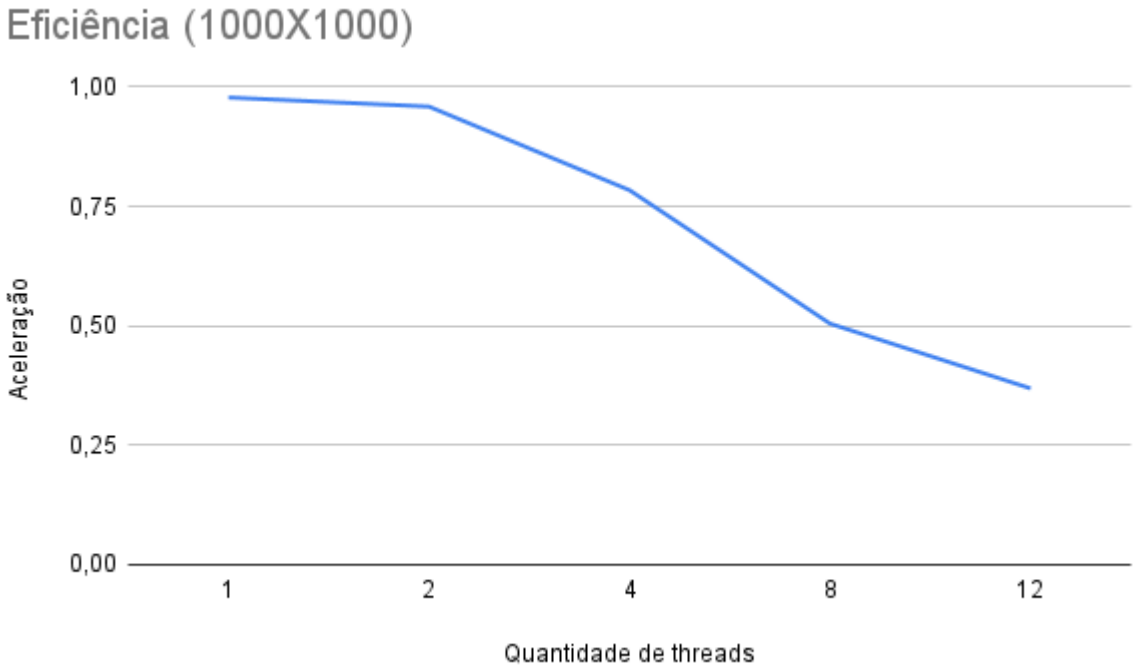
Agora, podemos analisar que quanto maior a dimensão da matriz, maior a disparidade entre o concorrente e o sequencial.

b. Execução para matrizes de dimensão 1000X1000

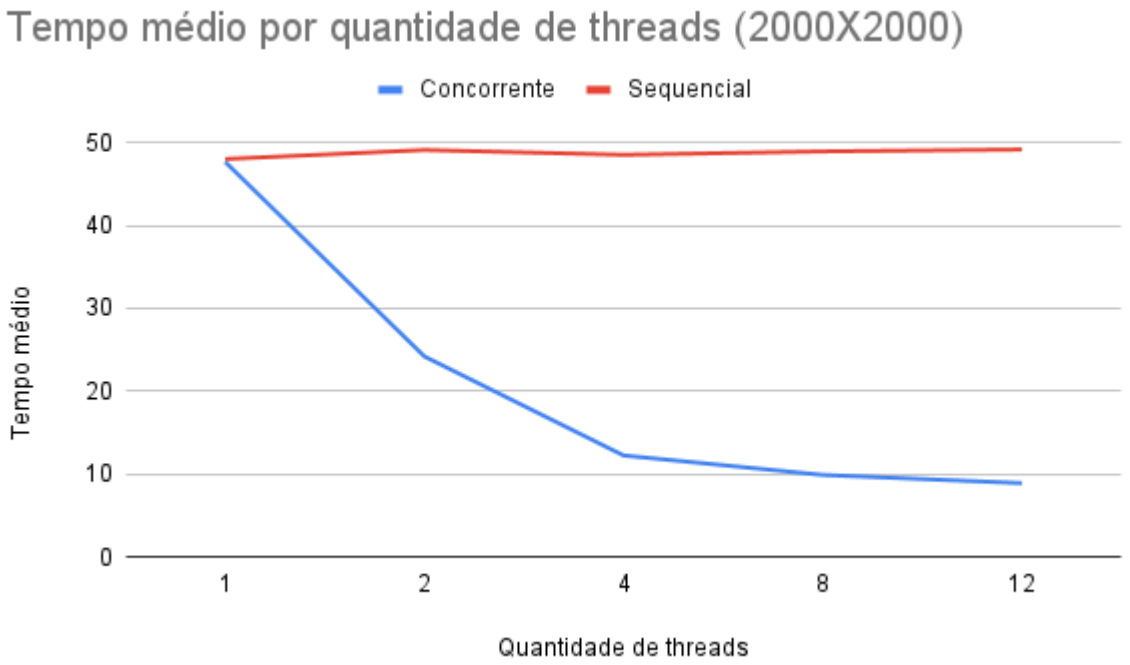


neste gráfico começamos a ver a diferença pois quanto maior a thread, mais se aproxima de 0 enquanto para o programa sequencial está fixo em 6 u.t (unidade de tempo)..



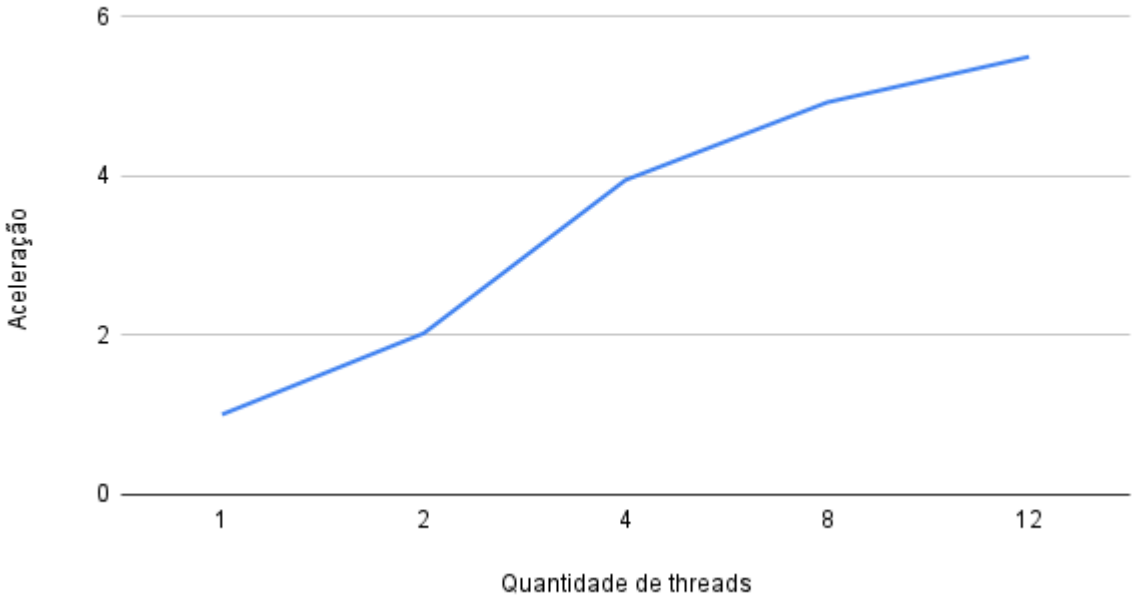


c. Execução para matrizes de dimensão 2000X2000



Analogamente do no caso do 1000X1000, ao chegarmos no gráfico de matriz com dimensão 2000X2000, percebe-se que fica maior ainda a disparidade de valores do tempo médio para o sequencial e o concorrente.

Aceleração por quantidade de threads (2000X2000)



Eficiência (2000X2000)