

* 1. UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Pthreads 1 - Matriz quadradas

**Pedro Cardoso Carvalho**

**NATAL-RN**

**2020**

**Sumário**

[**Introdução**](#_6xvfwkeumjbo) **3**

[**Desenvolvimento**](#_pkk2c8j6rvtm) **3**

[Ambiente de teste](#_jgh6kbg8vm3z) 3

[Programa serial](#_nclregthvvg7) 4

[Programa Paralelo](#_dh390irdhy5) 6

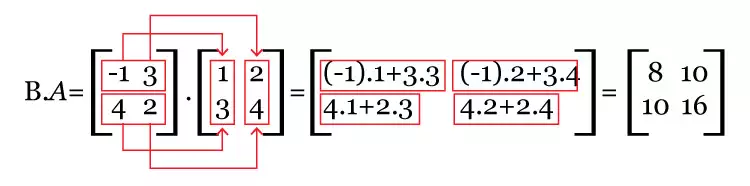
[Resultados](#_jybx269xcp5p) 8

[Considerações Finais](#_25qqtac1s9ie) **11**

# Introdução

Neste relatório iremos realizar um comparativo entre a utilização da programação paralela à tradicional programação serial, aplicado na resolução de uma multiplicação de matriz quadrada.Como demonstrado na figura 1, a resolução da multiplicação de matrizes que possuem mesmo número de linhas e colunas, chamada de matrizes quadradas, é realizado por uma somatório da multiplicação da coluna da primeira matriz com a linha da segunda matriz.

**Figura 1-** Demonstração de resolução de multiplicação de matriz quadrada



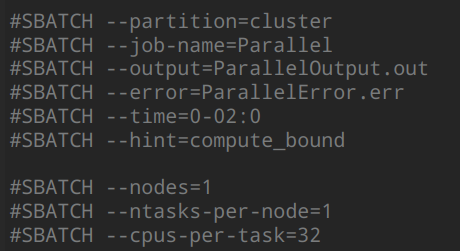
**Fonte** [**Só matemática**](https://www.somatematica.com.br/emedio/matrizes/matrizes4.php)

# Desenvolvimento

## Ambiente de teste

Todos os teste foram realizados em uma máquina do supercomputador disponibilizada pelo NPAD sendo para os códigos seriais a utilização de um núcleo e para o código paralelo uma maquina de até 32 cores(figura 2 possui o código do cabeçalho onde temos a configuração de uso da máquina paralela solicitada) . Foram selecionados 4 tamanhos de problemas que correspondem os valores M para duas matrizes MxM, sendo eles 1440, 1600, 1800 e 2080 , para cada uma desses casos foram realizados 10 execuções e para o caso do código paralelo para cada ambiente deste temos os casos de utilização de 4, 8, 16 e 32 processos. Para geração dos valores da matriz serem iguais para todos os casos foi utilizado a semente com valor fixo de 10 em hardcode.

**Figura 2-** Configuração da maquina virtual do codigo paralelo



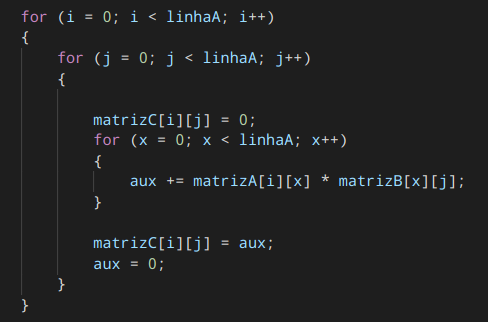
Fonte: Autor

## Programa serial

Para código serial, como forma de melhorar o estudo do desempenho, foram criados dois codigo. Um que realizaria o cálculos das matrizes seguindo a ordem linhas da primeira matriz

Para código serial, seguindo a ideia do problema relatado na introdução, foi gerado três sequência de loopings onde o primeiro responsável por percorrer os index das linhas da matriz resultante, segundo os index das colunas da matriz resultante e o terceiro looping que iria percorrer os valores das linha da primeira matriz e da coluna da segunda matriz realizando a multiplicação dos valores e somatório total desses valores, como demonstrado na figura 3

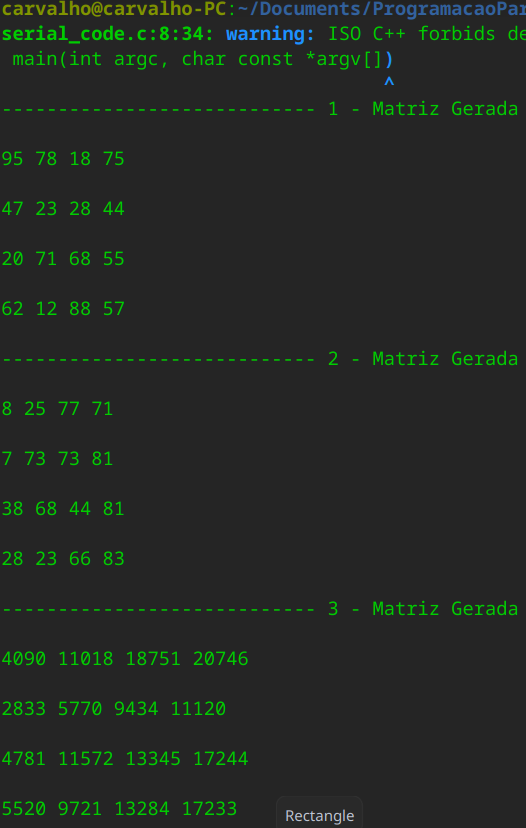
**Figura 3-** Código serial de multiplicação de matriz quadrada



**Fonte -**Autor

Um exemplo de execução do código mostrado pode ser visto na figura 4 com matriz valor M de 4.

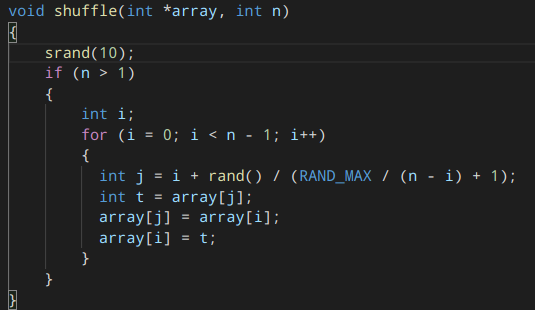
**Figura 4-** Execução Código serial de multiplicação de matrizes quadradas



**Fonte -**Autor

Como forma de estudar o comportamento do princípio da localidade espacial, que em resumo diz que a probabilidade de acessar um dado e e instruções são maiores em endereços próximos àqueles acessados recentemente. Foi criado um segundo código serial onde a seleção do index da linhas realizado pelo primeiro looping seria de forma aleatória. Para isso foi criado uma lista de index de 0 até o valor total de linhas da matriz menos um e a partir de uma função de embaralhamento, mostrada na figura 5, seria embaralhada. Assim o primeiro looping iria percorrer essa lista de index no final calculando aleatoriamente as linhas da matriz resultante.

**Figura 5-** Função emparelhamento de listas



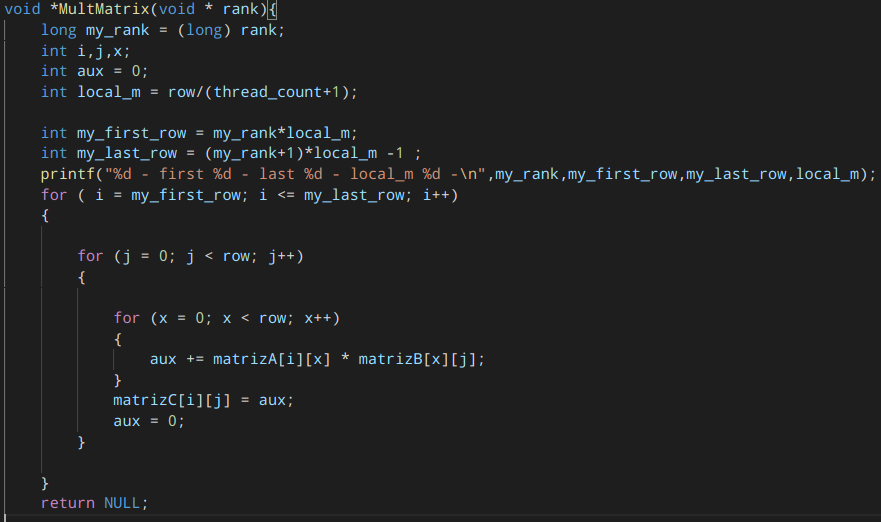
Fonte: Autor

Por fim será analisando o código que possuir o melhor tempo de execução será utilizado para análise com o código paralelo.

## Programa Paralelo

Para o modelo paralelo foi pensado em uma divisão do número de linhas igualmente para cada core da aplicação(contando com o processo pai) e assim cada thread iria calcular de forma serial cada bloco de linhas que ficou responsável. Como podemos ver na figura 6.

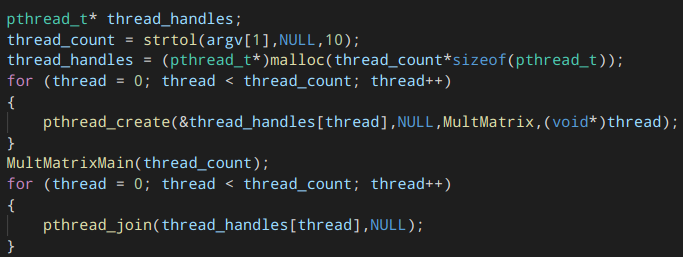
**Figura 6-** função utilizado por cada thread



**Fonte -** Autor

Com a utilização da biblioteca pthread foi realizada a criação das threads( figura 7) e o enviou das função mostrada na figura 6, posterior a isso o processo main realiza o cálculos das linhas que ficou responsável e ao final espera as demais threads finalizarem sua execução para finalizar o processo.

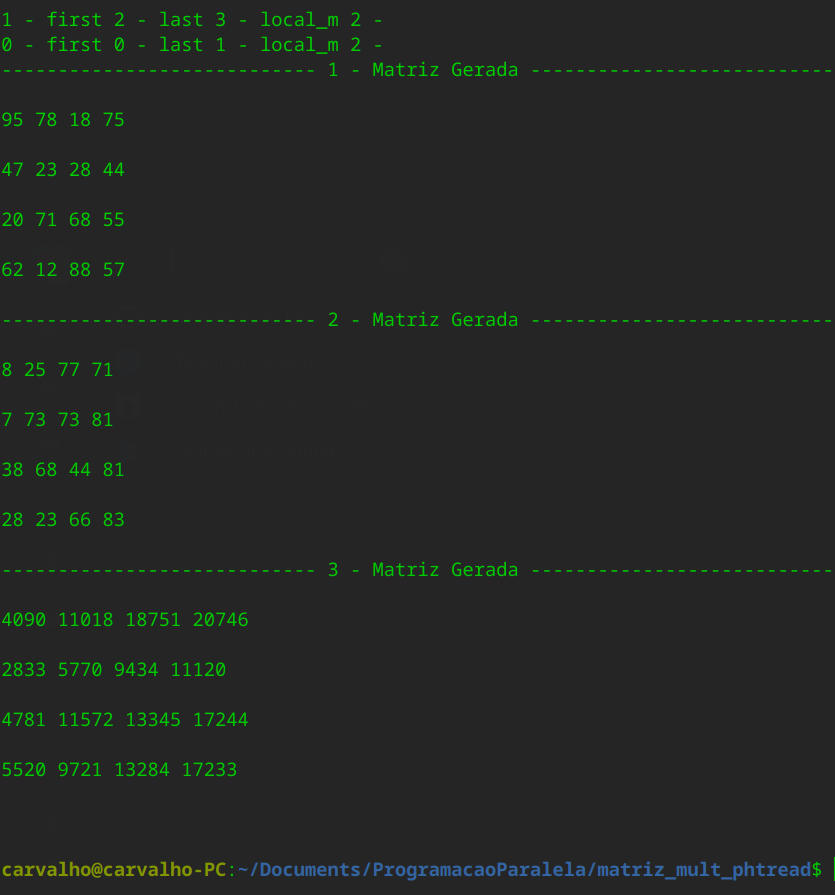
**Figura 7-** Código da criação de threads e execução da função



**Fonte -**Autor

Um exemplo de execução do código pode ser visto na figura 8, com a utilização de uma matriz 4x4 e utilização de 2 cores.

**Figura 8-** Execução Código paralelo

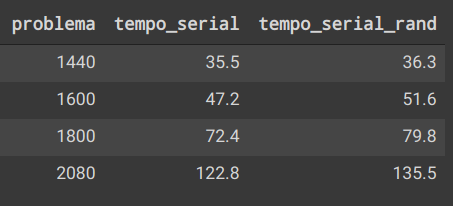


**Fonte -**Autor

## Resultados

A primeira análise que iremos realizar é o comparativos dos tempos médios resultados da execução dos dois códigos seriais.

**Figura 9-** Comparativo do tempos de execução dos códigos seriais

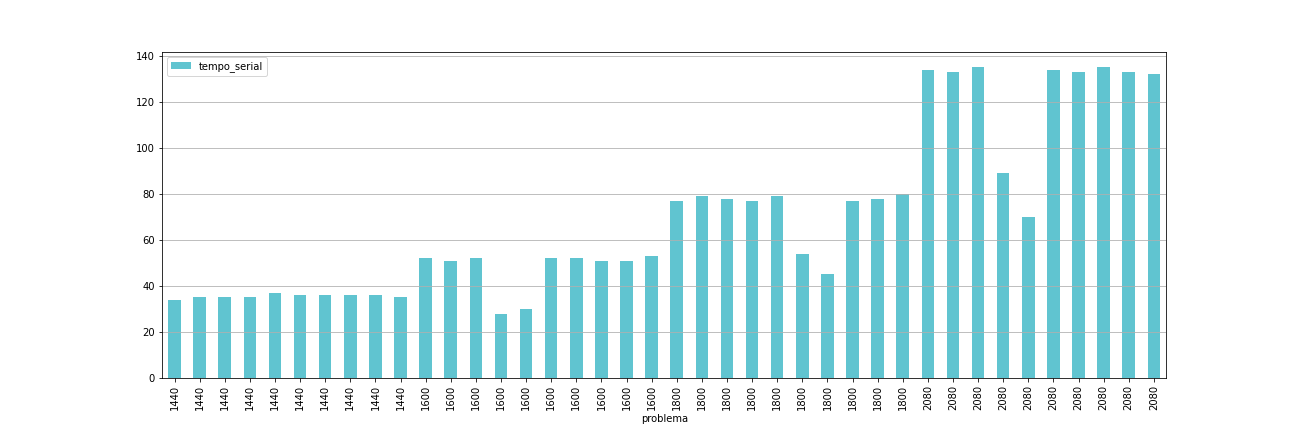


**Fonte -**Autor

Na figura 9 temos o “tempo\_serial” como tempo de execução das linhas das matrizes em sequência enquanto “tempo\_serial\_rand” o tempo de execução da seleção aleatórias de linhas. Como já previsto pelo princípio da localidade espacial, a diferença das médias dos tempos de execução do primeiro código se tornaram cada vez mais relevante de acordo com aumento do problema, chegando a uma diferença de aproximadamente 13 segundos na execução de uma matriz de 2080 x 2080.

Tendo confirmado isso, iremos utilizar ,a partir de agora, somente os tempos de execução do primeiro código para quesito de análises. A figura 10 temos o gráfico de barras que representa todos tempos de execução do código serial.

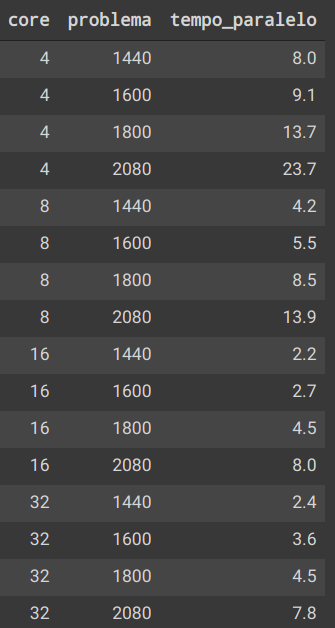
**Figura 10-** Tempos de execução código serial



**Fonte -**Autor

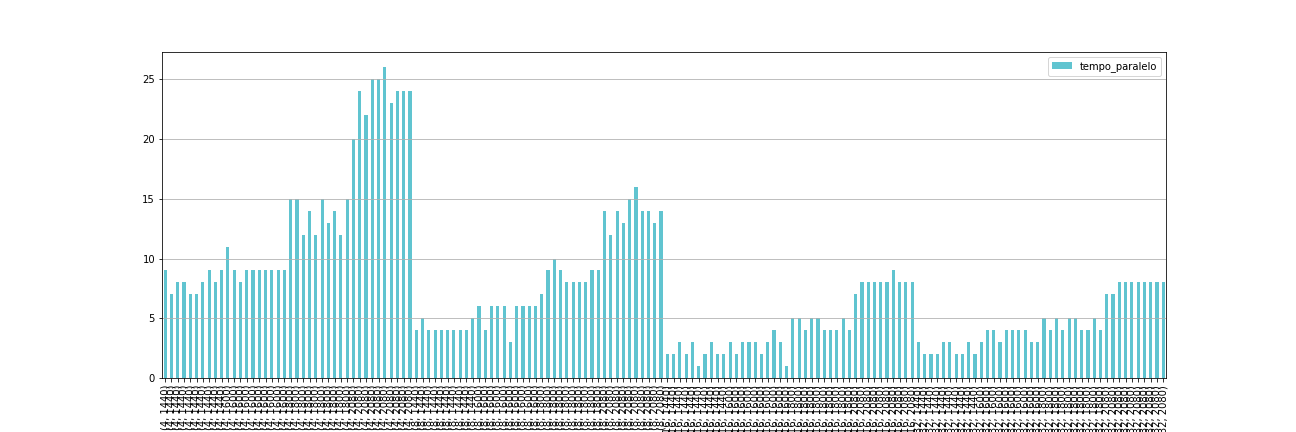
Nas figuras 11 vemos os tempos médios de execução do código paralelo e em seguida figura 12 temos os histórico de tempos de execução mostrados por barrar de todas as execuções em cada quantidade de cores e tamanho de problema.

**Figura 11-** Tabela de tempos médios de execução do código paralelo



**Fonte -**Autor

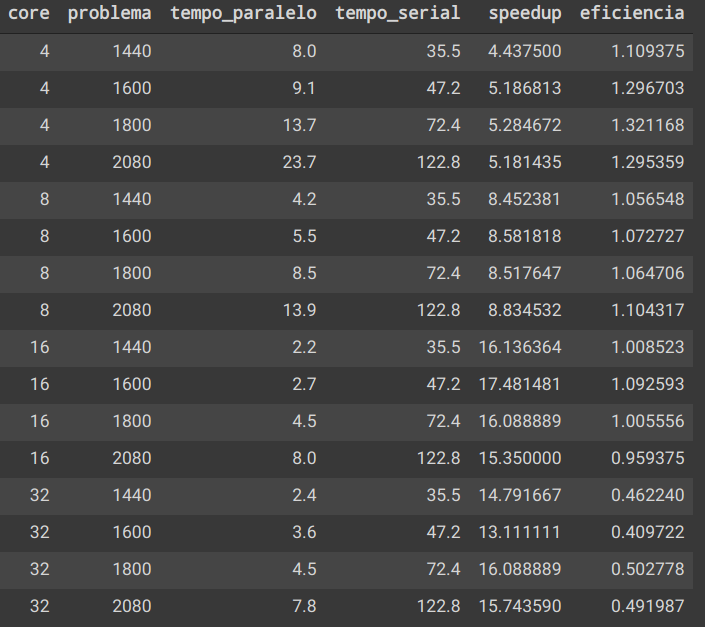
**Figura 12-** Tempos de execução código paralelol



**Fonte -**Autor

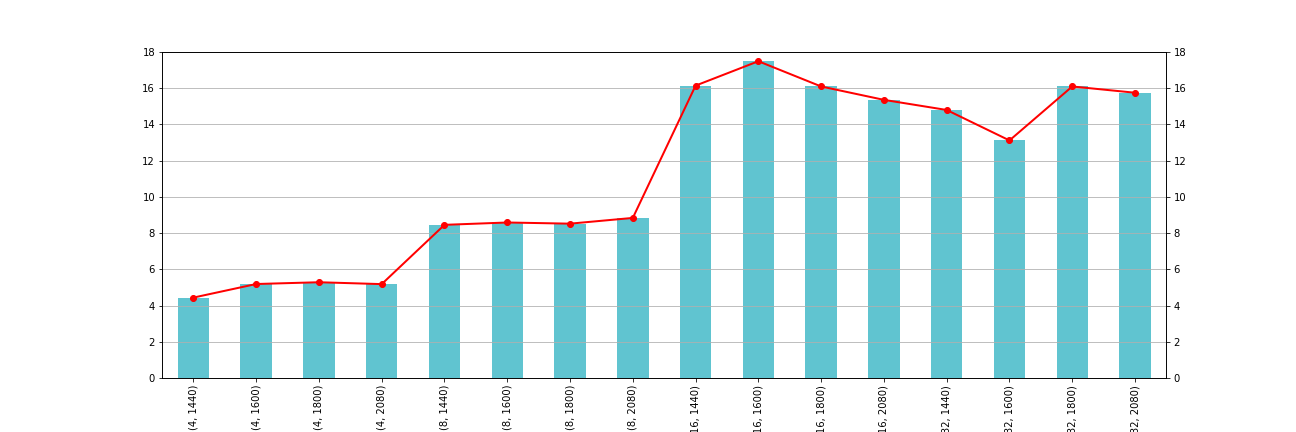
Foi concatenado as tabelas de média de tempos mostrados nas figuras 11 e 9 e calculamos o Speedup ( tempo serial/tempo paralelo) e a eficiência (Speedup/ números de processos) e geramos a tabela a seguir(figura 13). E para melhor visualização foi gerado os gráficos de Speedup e eficiência(figura 14 e 15, respectivamentes).

**Figura 13-** tabela resultante do estudo



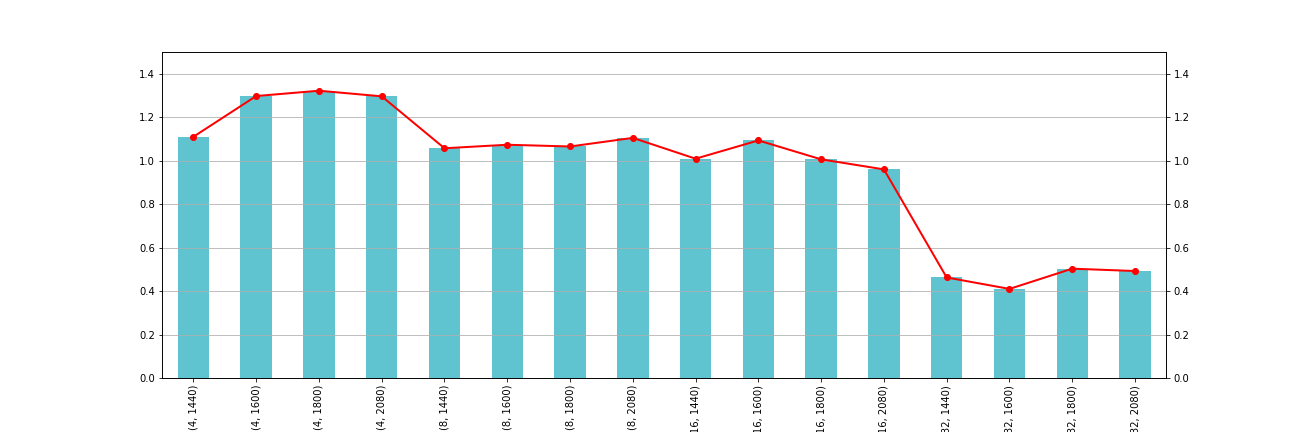
**Fonte -**Autor

**Figura 14-** gráfico de Speedup



**Fonte -**Autor

**Figura 15-** gráfico de eficiencia



**Fonte -**Autor

Em uma análise partindo da figura 11. Vemos uma queda expressiva nos tempos de execução em relação aos códigos em paralelo para o em série, mostrando um bom aproveitamento de soluções paralelas para multiplicações de matrizes quadradas. O que mais chama atenção são os valores speedup e eficiência onde obtemos valores de ganhos perfeitos nas utlização de 4, 8 e 16 threads tendo tendo valores de eficiência de 0,95 até 1,32 e Speedup com valores próximos ao números de cores.

Quando visualizamos os resultados de 32 cores percebemos valores de tempo iguais aos 16 e uma queda brusca nos valores Speedup e eficiência, que em um ambiente perfeito seriam de aproximadamente 1 e 32 temos os resultados aproximados de 15 e 0,50. Esse efeito pode ser justificado pelo limitação na diminuição da parcela de tempo da execução em paralelo fazendo com que seu tempo não consiga ser diminuído e que possivelmente grande parte do tempo final é resultante do tempo serial do código.

Ao fim podemos classificar o código como escalável por ter conseguido manter aproximadamente constante os valores de eficiência enquanto aumentava o número de cores e tamanho do problema .

# Considerações Finais

Em resumo os resultados dos estudo se apresentam gratificantes, obtendo resultados perfeito na utilização do codigo paralelo em até 16 cores .Para melhor visualizão os códigos (serial e paralelo) juntamente com o notebook em python, utilizado para realizar a analise dos dados e construção dos graficos, estão disponibilizados no github pelo link :

<https://github.com/PdrCarvalho/matriz_mult_phtread>