Programação Concorrente

2024.1

UFRJ

Laboratório 3

Iago Cesar Tavares de Souza - 122087998

Dados da Máquina

Processador: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Velocidade base: 2,4 GHz
Sockets: 1
Núcleos: 4
Processadores lógicos: 8

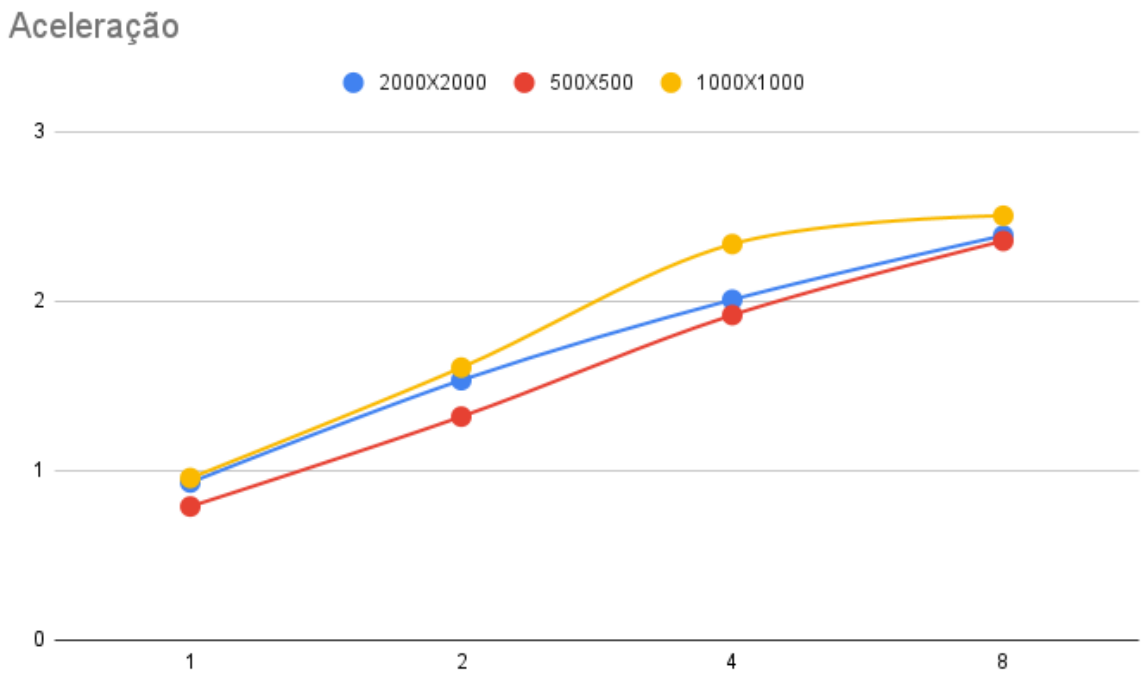
Implementação

O código foi construído de modo que a cada thread realizasse a multiplicação relativa às linhas da matriz de saída, ou seja, numa multiplicação de matrizes 2x2, AxB=C, realizada por 2 threads, a thread 1 ficaria responsável pela pelos valores C_{11}, C_{12} enquanto a thread 2 ficaria responsável por calcular todos os valores da segunda linha, isto é C_{21}, C_{22}. Além disso, para coleta de dados foram tomados os tempos de:

- Inicialização: O tempo para alocação de todos os espaços de memória necessários para o programa, com exceção da matriz resultado, assim como as variáveis utilizadas ao longo dele.
- Execução: Tempo de alocação da matriz resultado, criação das threads e multiplicação das matrizes
- Gravação: Tempo necessário para escrita da matriz resultado no arquivo.
- Finalização: Tempo para liberar os espaços de memória alocados no programa.

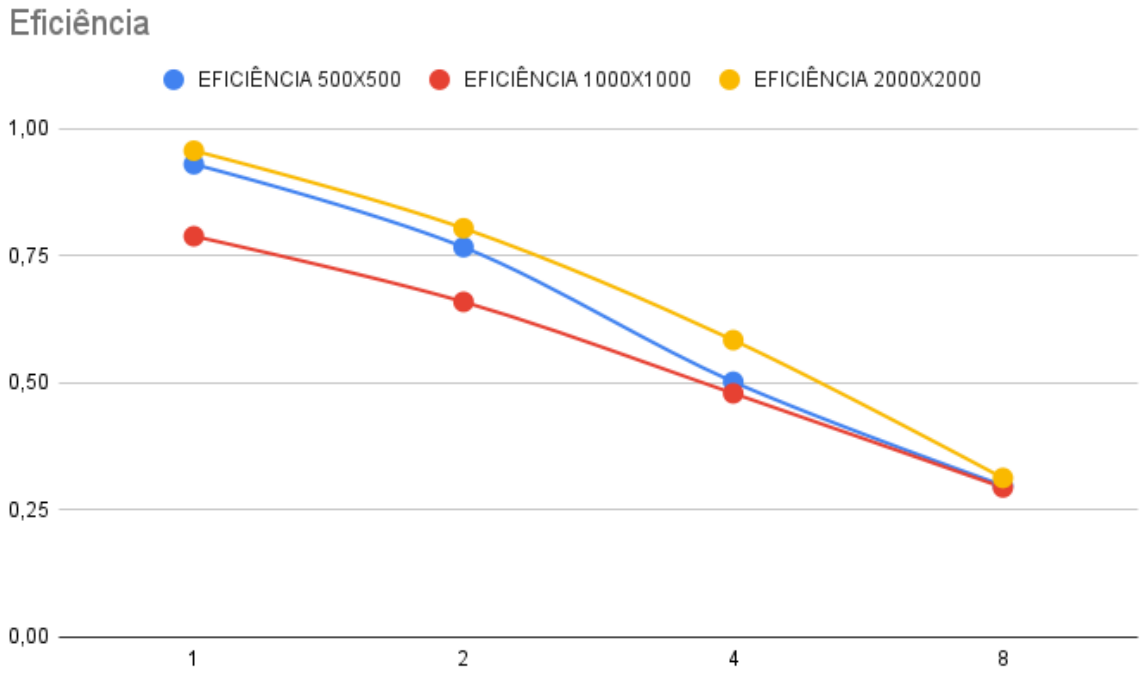
Além disso, foram realizadas 15 execução para cada tamanho de matriz, para cada quantidade de thread, assim como o sequencial com cada tamanho, totalizando 225 execuções, cada uma com os tempos mencionados acima.

Análise



Observando o gráfico de aceleração acima, é possível notar que a multiplicação para matrizes de tamanho 1000 apresentam um comportamento muito semelhante a de 2000 inicialmente para 1 e 2 threads, porém para 4 threads a matriz de 1000 possui um avanço muito bom em relação aos outros tamanhos de matriz.

Em relação a matriz de 500 e 2000, pode-se dizer que ambas possuem uma evolução da aceleração visualmente linear e muito próximas de maneira geral, porém na execução com 8 threads a aceleração de todas termina quase igual.



Diferente do esperado, conforme o número de threads aumenta, a eficiência acaba decrescendo, o que significa que o ganho em função do aumento do número de threads está longe do ideal para execução do programa.

Isso é facilmente compreensível já que o número de operações realizadas é quadrático, e considerando o overhead gerado pelas threads que, apesar de serem poucas nestes casos, acabam não gerando lucro suficiente para compensar o crescimento exponencial do tamanho das matrizes.