Laboratório 3
Multiplicação de Matrizes Concorrente

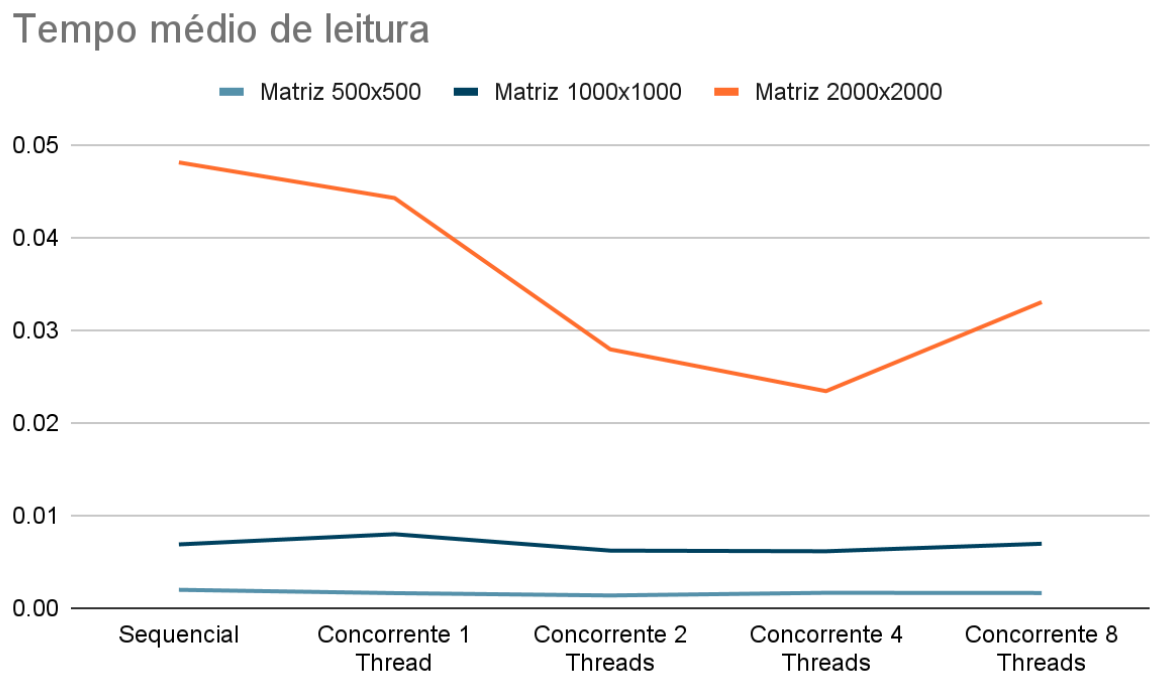
Programação Concorrente (ICP-361) 2024-2
Prof. Silvana Rossetto

Instituto de Computação/UFRJ

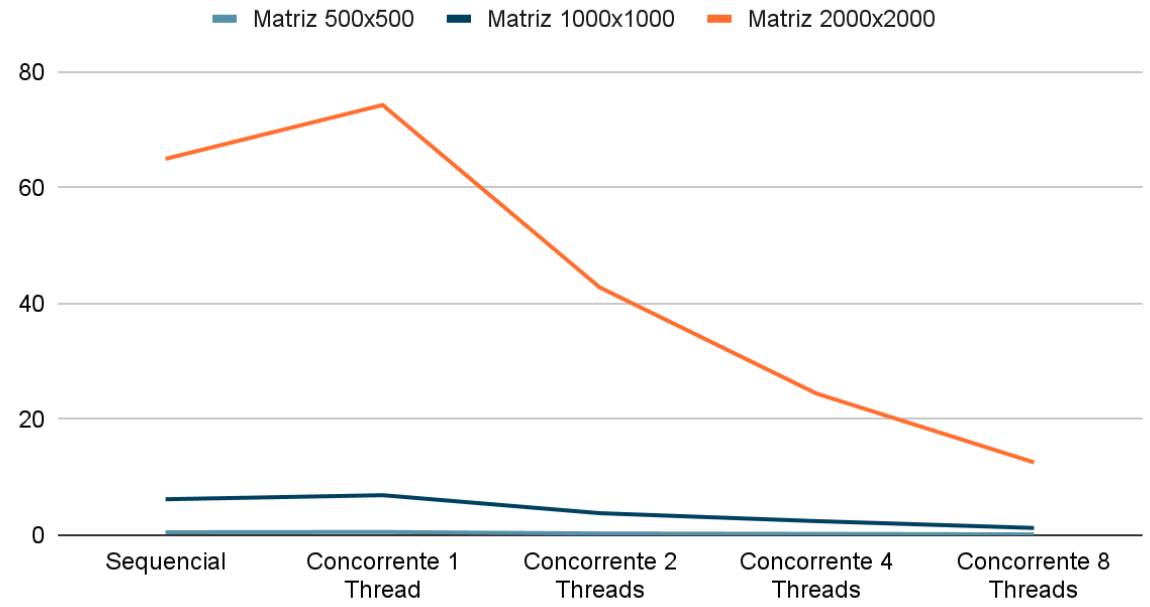
João Vitor Alvarenga - 120152276

Tabela:
A tabela está no repositório do código

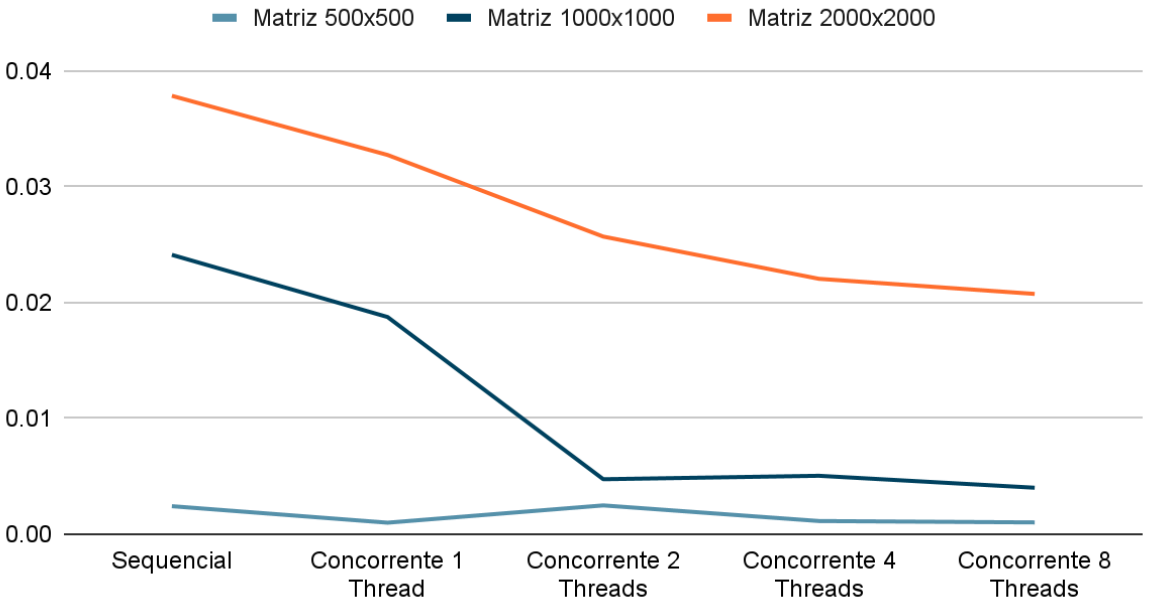
Gráficos:

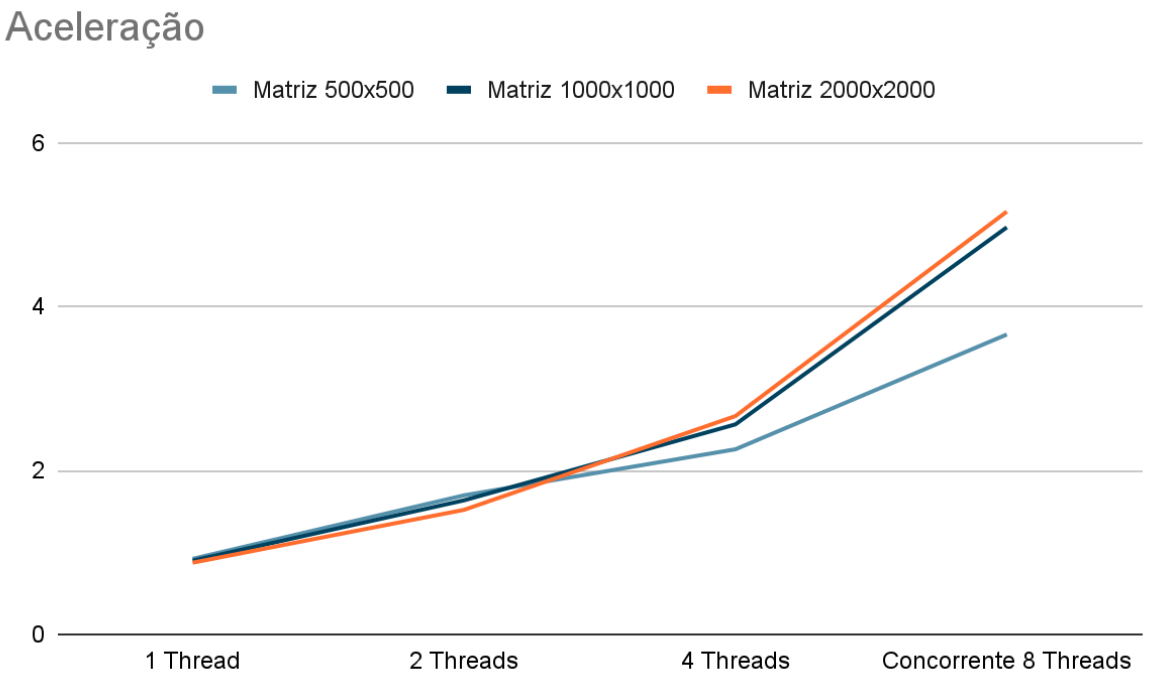


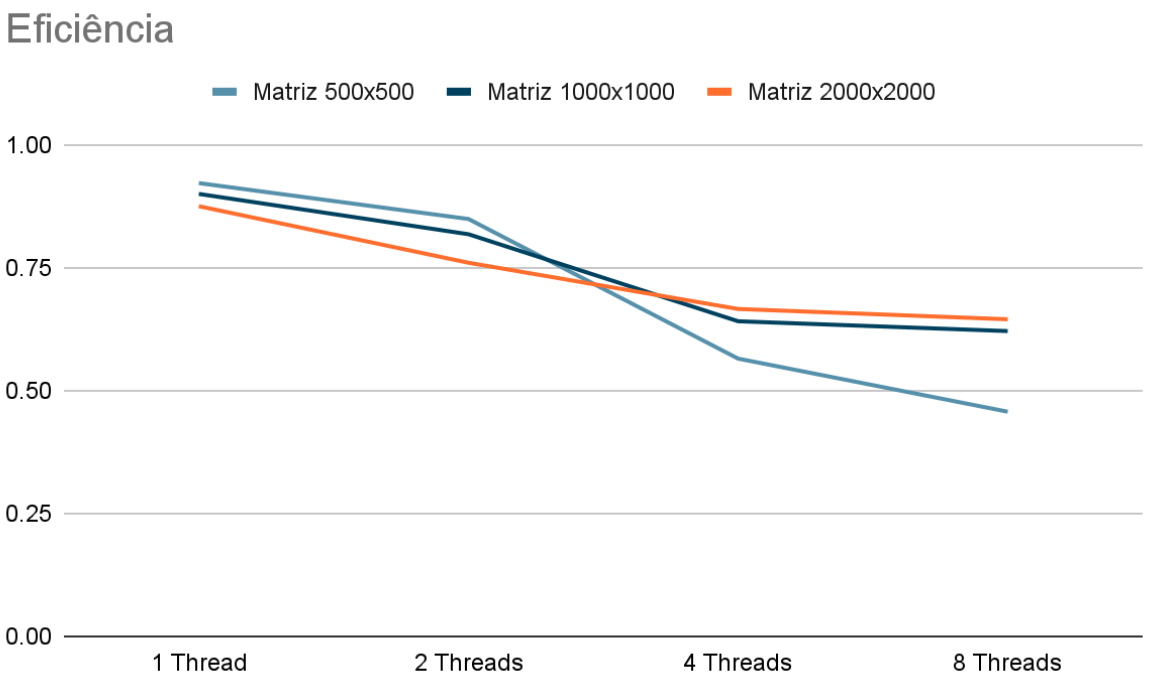
Tempo médio de processamento



Tempo médio de finalização







Configuração da Máquina:
Processador: 12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz
Número de Núcleos: 10
Número de Threads: 12
RAM: 16.0 GB (15.7 GB usable)

Conclusões Gerais:

Redução de Tempo Total:

Matriz 500x500: A execução sequencial é mais rápida em comparação com a execução concorrente com 1 thread, o que pode ser devido ao overhead associado à criação e gerenciamento de threads. A partir de 2 threads, o desempenho melhora significativamente, e com 8 threads, o tempo total é reduzido de forma substancial.

Matriz 1000x1000: A execução concorrente mostra uma melhoria significativa em comparação com a execução sequencial, especialmente com 4 e 8 threads. O tempo total reduz bastante à medida que o número de threads aumenta.

Matriz 2000x2000: O tempo total de execução é consideravelmente menor para a execução concorrente com 4 e 8 threads em comparação com a execução sequencial. O tempo total é reduzido para menos de um quarto do tempo sequencial ao usar 8 threads, o que destaca o impacto positivo da paralelização.

Aceleração:

Aumenta com o número de threads: Em geral, a aceleração aumenta conforme o número de threads aumenta, especialmente para matrizes maiores. Isso é esperado, pois mais threads podem dividir o trabalho de processamento, reduzindo o tempo total.

Melhor para matrizes maiores: A aceleração tende a ser maior para matrizes maiores (2000x2000), indicando que o benefício do paralelismo é mais pronunciado para tarefas mais intensivas em cálculo. Para matrizes menores (500x500), a aceleração não é tão significativa porque o tempo de processamento é menor e o overhead de criar e gerenciar threads pode ser mais perceptível.

Eficiência:

Diminui com o aumento do número de threads: A eficiência, que é a aceleração dividida pelo número de threads, tende a diminuir conforme o número de threads aumenta. Isso acontece porque, embora a aceleração aumente, o benefício adicional por thread se torna menor. Ou seja, adicionar mais threads contribui menos para a aceleração adicional à medida que você adiciona mais threads.

Melhor para matrizes menores: A eficiência tende a ser mais alta para matrizes menores quando se usa um menor número de threads, apesar da tendência geral de diminuição. Isso sugere que, para tarefas mais intensivas em cálculos, a adição de mais threads não traz um ganho em termos de desempenho.

Sequencial vs Concorrente:

Desempenho sequencial vs. concorrente: O tempo total sequencial é geralmente mais alto em comparação com o tempo total concorrente. A implementação concorrente, com apenas uma thread, tende a ter um tempo total semelhante ou até melhor que o desempenho concorrente. À medida que o número de threads aumenta, o desempenho concorrente se destaca ainda mais, especialmente para matrizes maiores.