Laboratório 3 - Atividade 1

Nome: Julia Deroci Lopes
DRE: 117257871

Tempo de Execução

O tempo de execução do código sequencial e concorrente foi medido para diferentes tamanhos de matrizes (500x500, 1000x1000, 2000x2000). No código sequencial, o tempo médio de processamento aumentou à medida que o tamanho da matriz aumentava. Já no código concorrente, o uso de múltiplos threads (1, 2, 4 e 8 threads) mostrou uma diminuição significativa no tempo de execução, especialmente com 4 e 8 threads, demonstrando o ganho de performance.

Abaixo podemos ver os dados obtidos para cada código:

- **Código Sequencial**

Tempo de Execução (Processamento) - Sequencial			
Nº	500 X 500	1000 X 1000	2000 X 2000
1	0,251315	2,209637	20,284473
2	0,248088	2,209101	20,325378
3	0,249157	2,212613	20,499651
4	0,252480	2,197720	20,614900
5	0,250942	2,195324	20,668775
Média	0,250396	2,204879	20,478635

- **Código Concorrente**

Tempo de Execução (Processamento) - Concorrente						
N° Threads	1			2		
Nº	500 X 500	1000 X 1000	2000 X 2000	500 X 500	1000 X 1000	2000 X 2000
1	0,252108	2,196311	20,157780	0,126886	1,102883	10,751335
2	0,254204	2,203545	20,943165	0,131320	1,107619	10,795020
3	0,254242	2,203231	20,751536	0,131023	1,103807	10,758936
4	0,254245	2,211162	20,184140	0,130258	1,107501	10,934133
5	0,253322	2,265879	20,599978	0,128572	1,102002	10,951880
Média	0,2536242	2,2160256	20,5273198	0,1296118	1,1047624	10,8382608

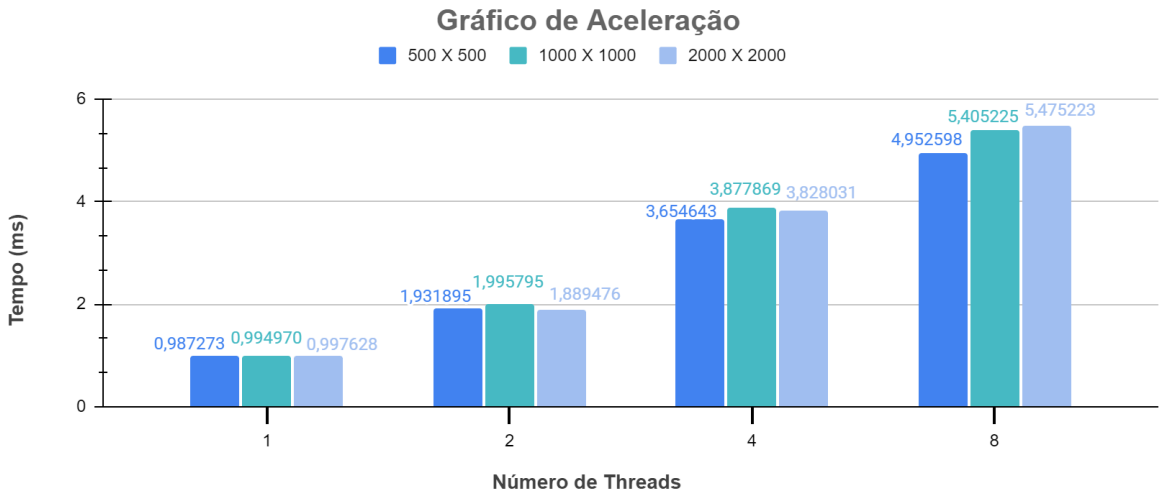
Tempo de Execução (Processamento) - Concorrente						
N° Threads	4			8		
Nº	500 X 500	1000 X 1000	2000 X 2000	500 X 500	1000 X 1000	2000 X 2000
1	0,065143	0,567425	5,149016	0,047756	0,419240	3,717008
2	0,069703	0,568403	5,210569	0,048956	0,401175	3,757970
3	0,068982	0,572375	5,263787	0,055800	0,407636	3,691808
4	0,068842	0,567570	5,525797	0,049710	0,399772	3,816798
5	0,069903	0,567127	5,599090	0,050571	0,411758	3,717605
Média	0,0685146	0,56858	5,3496518	0,0505586	0,4079162	3,7402378

Desempenho

- **Aceleração**

  A aceleração foi calculada comparando os tempos de execução dos códigos concorrentes com o sequencial. Com o aumento do número de threads, a aceleração foi maior, atingindo um pico de desempenho com 8 threads. A aceleração variou de cerca de 1,89 para 2 threads até 5,47 para 8 threads nas matrizes maiores.

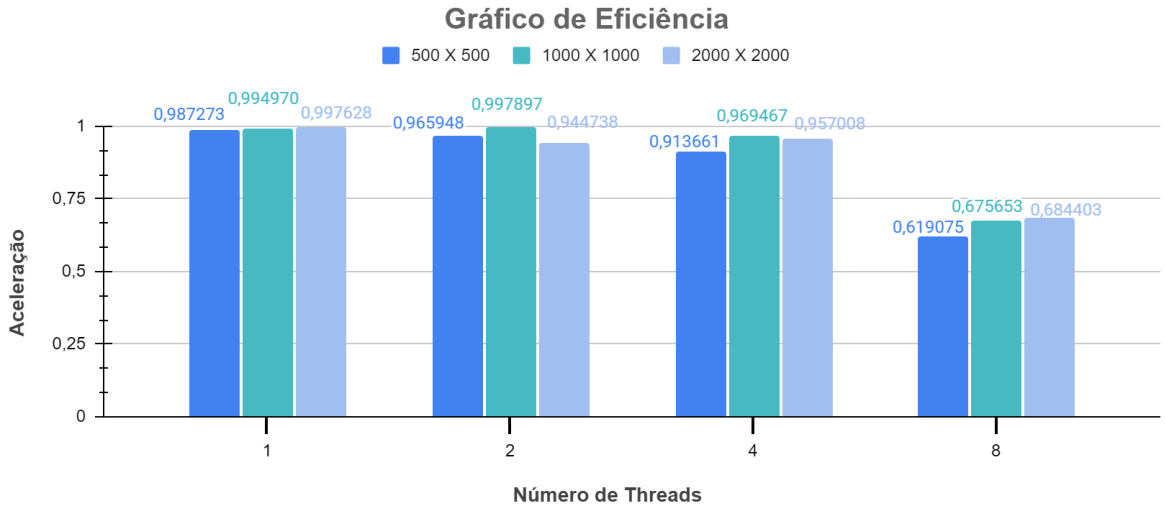
Aceleração				
N° Threads	1	2	4	8
500 X 500	0,987273	1,931895	3,654643	4,952598
1000 X 1000	0,994970	1,995795	3,877869	5,405225
2000 X 2000	0,997628	1,889476	3,828031	5,475223



- **Eficiência**

  A eficiência mostrou como o uso de múltiplos threads foi aproveitado. Para 1 thread, a eficiência foi quase total (perto de 1), mas à medida que mais threads foram adicionados, a eficiência diminuiu. No caso de 8 threads, por exemplo, a eficiência para a matriz 500x500 caiu para 0,61, refletindo uma perda de eficiência ao utilizar mais recursos do sistema.

Eficiência				
N° Threads	1	2	4	8
500 X 500	0,987273	0,965948	0,913661	0,619075
1000 X 1000	0,994970	0,997897	0,969467	0,675653
2000 X 2000	0,997628	0,944738	0,957008	0,684403



Configurações da Máquina

A máquina utilizada na realização dos testes possui as seguintes configurações:

- Sistema Operacional: Linux Mint 21 Cinnamon
- Versão do Cinnamon: 5.4.12
- Kernel do Linux: 5.15.0-101-generic
- Processador: 13ª geração Intel® Core™ i5-13500, com 14 núcleos

Conclusão

A partir dos resultados obtidos podemos observar que o uso de programação concorrente com múltiplos threads melhora significativamente o tempo de execução para grandes matrizes. No entanto, isso vem à custa da eficiência, que diminui com o aumento do número de threads, o que pode indicar uma saturação do sistema.