Programação Concorrente
Atividade 3
Ian de Andrade Camargo
118089205

Objetivo:

Partindo do começo, seguindo os passos designados no roteiro do laboratório, foi implementado um programa sequencial e um programa concorrente de multiplicação matricial, para verificar a corretude da solução, assim como sugerido utilizei o comando diff.

Porém como nosso intuito real é fazer comparações entre os resultados, foram realizados alguns testes, estes que seguem nas planilhas abaixo, tendo cada planilha o número de testes, a proporção da matriz a qual se refere, e a média destes testes, além destas informações, no caso concorrente também é expressado a quantidade de threads utilizadas.

Tempo de Execução (Processamento) - Sequencial			
Nº	500 X 500	1000 X 1000	2000 X 2000
1	0,140431	1,220929	13,883780
2	0,118285	1,153574	12,800603
3	0,142037	1,341624	12,842922
4	0,118636	1,148747	14,657605
5	0,119925	1,148747	12,427075
Média	0,127863	1,202724	13,322397

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	0,114004	1,527352	11,715374	0,117933	1,050980	12,384960
2	0,119187	1,182156	11,393980	0,123478	1,068885	12,073685
3	0,132463	0,996387	11,822267	0,120152	0,968926	11,668633
4	0,125709	1,136078	11,548411	0,120482	1,026888	12,569672
5	0,129647	0,983307	11,854925	0,133635	0,963577	14,795702
Média	0,124202	1,165056	11,6669914	0,123136	1,0158512	12,6985304

4			8		
500 X 500	1000 X 1000	2000 X 2000	500 X 500	1000 X 1000	2000 X 2000
0,131223	0,962322	12,710931	0,120423	0,951175	10,132002
0,122713	0,094629	12,727526	0,135530	0,985982	12,021654
0,115898	1,000509	12,005115	0,117372	1,003250	12,897534
0,130817	1,158072	13,022210	0,117213	1,058432	11,929921
0,127411	1,501485	11,629086	0,000401	1,058432	12,557214
0,1256124	0,94340342	12,4189736	0,0981878	1,0114542	11,907665

Desempenho

Por último iremos analisar um pouco os parâmetros de aceleração e eficiência que são obtidos através da fórmula:

$$A(n,t) = T_s(n)/T_p(n,t)$$

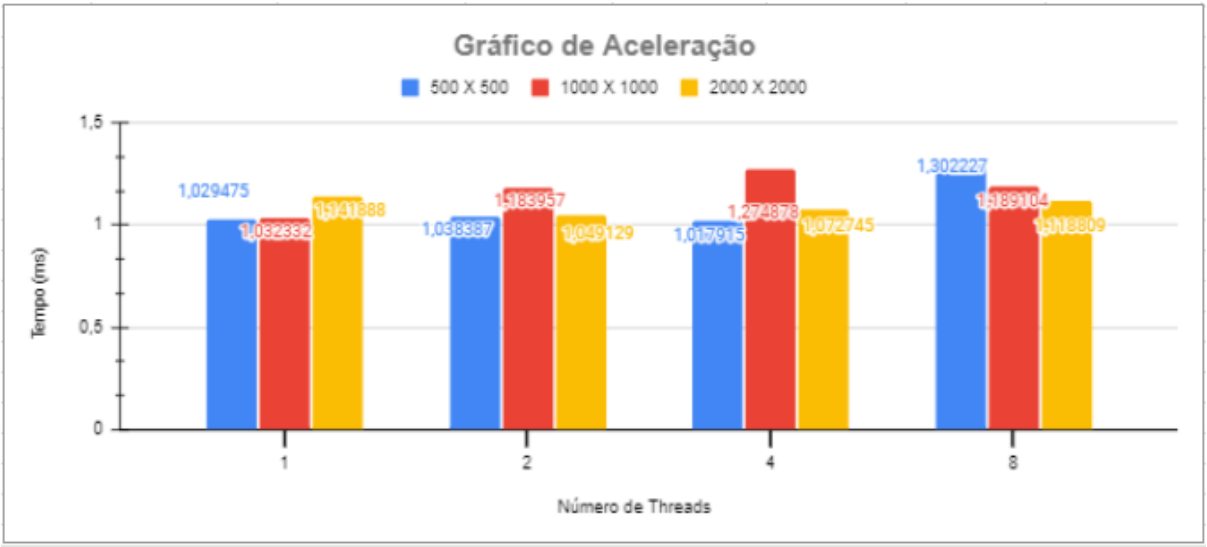
$$E(n,t) = A(n,t)/t$$

onde a fórmula da aceleração consiste, no tempo médio obtido no código sequencial, dividido pelo tempo médio obtido no código concorrente de acordo com a quantidade de threads utilizadas.

Enquanto isso, na fórmula de eficiência utilizamos a aceleração obtida, dividida pelo número de threads utilizado.

Aceleração:

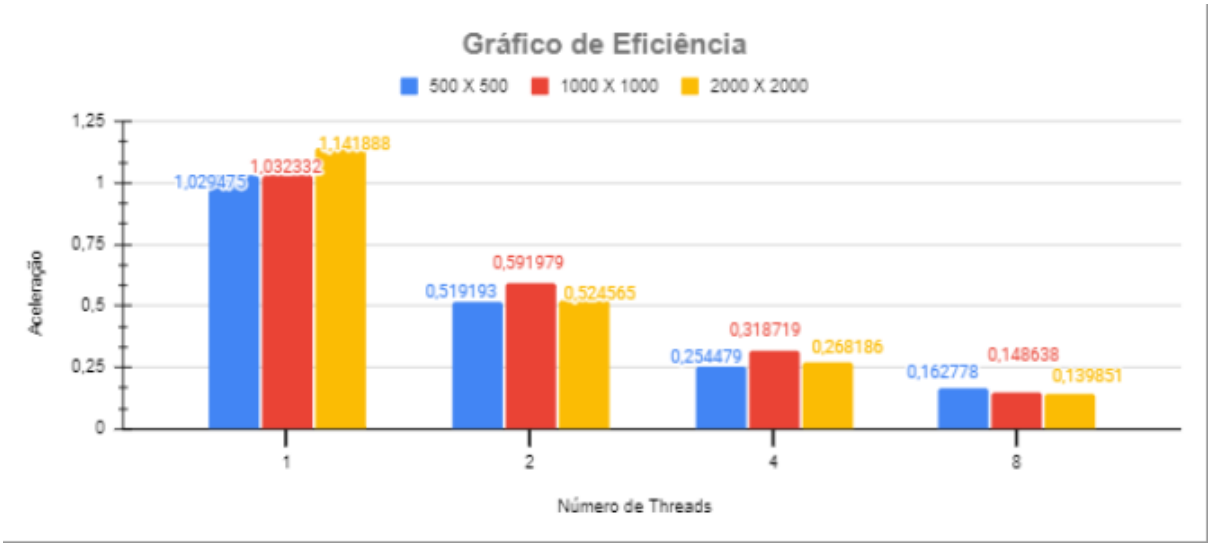
Aceleração				
N° Threads	1	2	4	8
500 X 500	1,029475	1,038387	1,017915	1,302227
1000 X 1000	1,032332	1,183957	1,274878	1,189104
2000 X 2000	1,141888	1,049129	1,072745	1,118809



Aqui, percebemos que, como esperado, a aceleração se mantém estável para uma única thread, próxima de 1. No entanto, para múltiplas threads, não há um crescimento consistente na aceleração. Isso pode ser atribuído ao overhead envolvido na criação, gerenciamento e sincronização das threads, o que reduz o ganho de desempenho esperado com o aumento do número de threads.

Eficiência:

Eficiência				
N° Threads	1	2	4	8
500 X 500	1,029475	0,519193	0,254479	0,162778
1000 X 1000	1,032332	0,591979	0,318719	0,148638
2000 X 2000	1,141888	0,524565	0,268186	0,139851



Já a tabela de eficiência mostra que, ao aumentar o número de threads, a eficiência diminui significativamente, indicando que o overhead de paralelismo (como sincronização e comunicação entre threads) é alto. O que imagino estar associado a uma utilização não tão eficiente das threads, não trazendo ganhos significativos no desempenho, principalmente nos problemas maiores.

Processador utilizado:

intel(r) core(tm) i5-4440 cpu @ 3.10ghz 3.10 ghz

4 núcleos - 4 threads