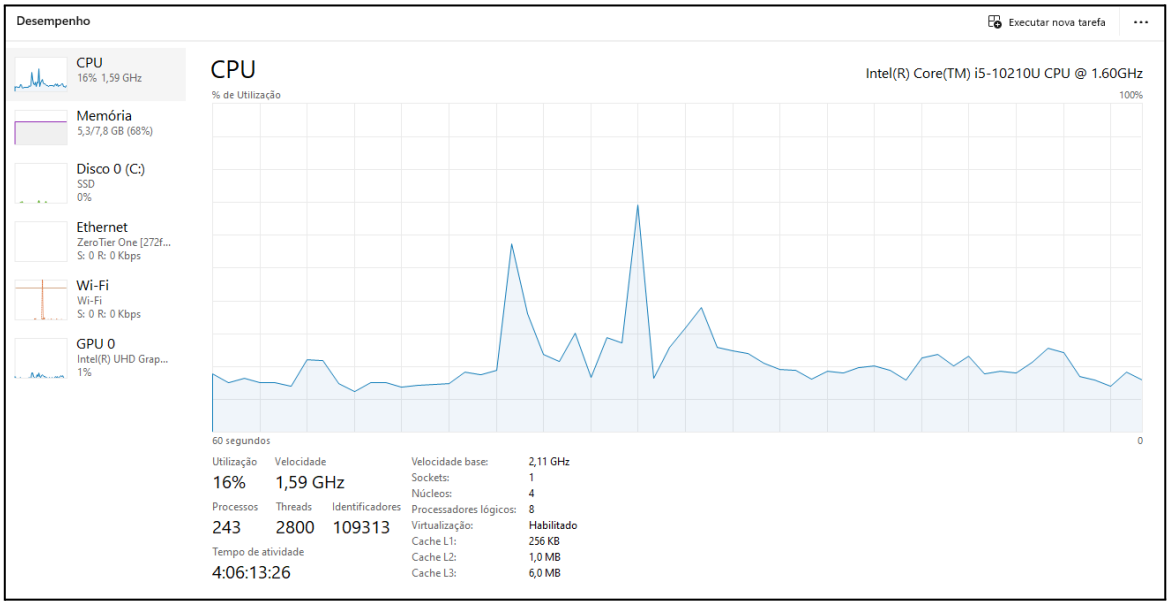
Aluno: Nicholas de Araujo Figueira DRE: 121088218

Relatório - Laboratório 3

Configurações do Computador



Para essa relatório, executou-se os programas de multiplicação de matrizes de tamanho 500 x 500, 1000 x 1000 e 2000 x 2000, sequencialmente e concorrentemente. No segundo caso, especificamente, utilizaram-se 1, 2, 4 e 8 threads. Deve-se atentar que o computador utilizado possui 4 núcleos.

Anotaram-se os tempos necessários para a leitura das matrizes e alocação de memória, multiplicação e liberação dos espaços de memória. Com esses valores, pode-se calcular os valores de aceleração e eficiência utilizando as equações:

$$A_{(n,t)} = \frac{T_s(n)}{T_p(n,t)} \; ; \; E_{(n,t)} = \frac{A(n,t)}{t}$$

Onde:

A = aceleração;

E = eficiência;

n = dimensão das matrizes;

t = quantidade de threads utilizadas;

Ts = tempo sequencial (tempo de inicialização + tempo de finalização);

Tp = tempo de processamento (tempo de multiplicação).

500 x 500

Sequencial						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.002383	0.000884	0.001900	0.004117	0.008118	0.003480
Multiplicação	0.536356	0.550774	0.540365	0.607979	0.557563	0.558607
Finalização	0.002640	0.001941	0.001853	0.004633	0.002614	0.002614

Concorrente - 1 Thread						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.009074	0.000914	0.000868	0.002217	0.001865	0.002988
Multiplicação	0.551772	0.563849	0.534102	0.548241	0.547116	0.549016
Finalização	0.002772	0.002005	0.001439	0.001193	0.001131	0.001708

Aceleração: 0.00855348
Eficiência: 0.00855348

Concorrente - 2 Threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.008972	0.001777	0.005080	0.001733	0.005715	0.004655
Multiplicação	0.316108	0.312111	0.322071	0.276642	0.270323	0.299451
Finalização	0.002174	0.001106	0.001013	0.001158	0.001434	0.001377

Aceleração: 0.02014353
Eficiência: 0.01007176

Concorrente - 4 Threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.001123	0.001208	0.000986	0.000872	0.005636	0.001965
Multiplicação	0.183734	0.195758	0.203794	0.174685	0.198783	0.191351
Finalização	0.004930	0.001012	0.001261	0.001398	0.001005	0.001921

Aceleração: 0.02030823
Eficiência: 0.00507706

Concorrente - 8 Threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.006057	0.001141	0.001629	0.001157	0.001001	0.002197
Multiplicação	0.159885	0.134983	0.170249	0.146258	0.138890	0.150053
Finalização	0.001627	0.001198	0.005078	0.001383	0.001255	0.002108

Aceleração: 0.02868986
Eficiência: 0.00358623

1000 x 1000

Sequencial						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.003614	0.005880	0.008480	0.003293	0.003342	0.004922
Multiplicação	6.239289	7.452598	6.814552	7.296838	7.084803	6.977616
Finalização	0.024474	0.003269	0.007533	0.003858	0.004140	0.008655

Concorrente - 1 thread						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.003168	0.004438	0.001980	0.010842	0.003293	0.00474
Multiplicação	7.383159	6.847156	7.393807	7.257560	7.343179	7.24497
Finalização	0.004033	0.004735	0.008250	0.009504	0.003366	0.00598

Aceleração: 0.00147965
Eficiência: 0.00147965

Concorrente - 2 threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.003011	0.026515	0.013435	0.009819	0.003219	0.01120
Multiplicação	3.580383	5.487085	3.395617	3.506291	3.441793	3.88223
Finalização	0.003907	0.007207	0.004040	0.003689	0.007180	0.00520

Aceleração: 0.00422438
Eficiência: 0.00211219

Concorrente - 4 threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.004170	0.003726	0.017696	0.056455	0.037055	0.023820
Multiplicação	2.139673	2.265952	2.265732	2.082076	2.006329	2.151952
Finalização	0.003818	0.004450	0.004208	0.004714	0.003492	0.004136

Aceleração: 0.01299100
Eficiência: 0.00324775

Concorrente - 8 threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.009208	0.015249	0.009025	0.006137	0.004661	0.008856
Multiplicação	1.976792	2.038203	1.877311	1.904801	1.870559	1.933533
Finalização	0.007109	0.007922	0.004818	0.003681	0.003440	0.005394

Aceleração: 0.00736993
Eficiência: 0.00092124

2000 x 2000

Sequencial						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.012252	0.030380	0.023346	0.022312	0.016247	0.02091
Multiplicação	75.356599	74.830785	75.279375	76.939865	71.377540	74.75683
Finalização	0.053968	0.028011	0.040970	0.019529	0.017016	0.03190

Concorrente - 1 Thread						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.029482	0.078919	0.010904	0.022312	0.020301	0.034450
Multiplicação	62.535484	62.964872	62.815403	62.178740	63.858191	62.870538
Finalização	0.013308	0.031016	0.010518	0.011230	0.010129	0.015240

Aceleração: 0.00079035
Eficiência: 0.00079035

Concorrente - 2 Threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.010774	0.035122	0.021623	0.021719	0.021623	0.022172
Multiplicação	36.405816	37.503415	36.790775	35.782804	36.790775	36.654717
Finalização	0.012943	0.011383	0.011186	0.011501	0.011186	0.011640

Aceleração: 0.00092245
Eficiência: 0.00046122

Concorrente - 4 Threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.015343	0.011311	0.011092	0.010283	0.021827	0.013971
Multiplicação	26.115806	27.155259	28.095080	26.083250	27.296011	26.949081
Finalização	0.011970	0.013265	0.010214	0.013893	0.015764	0.013021

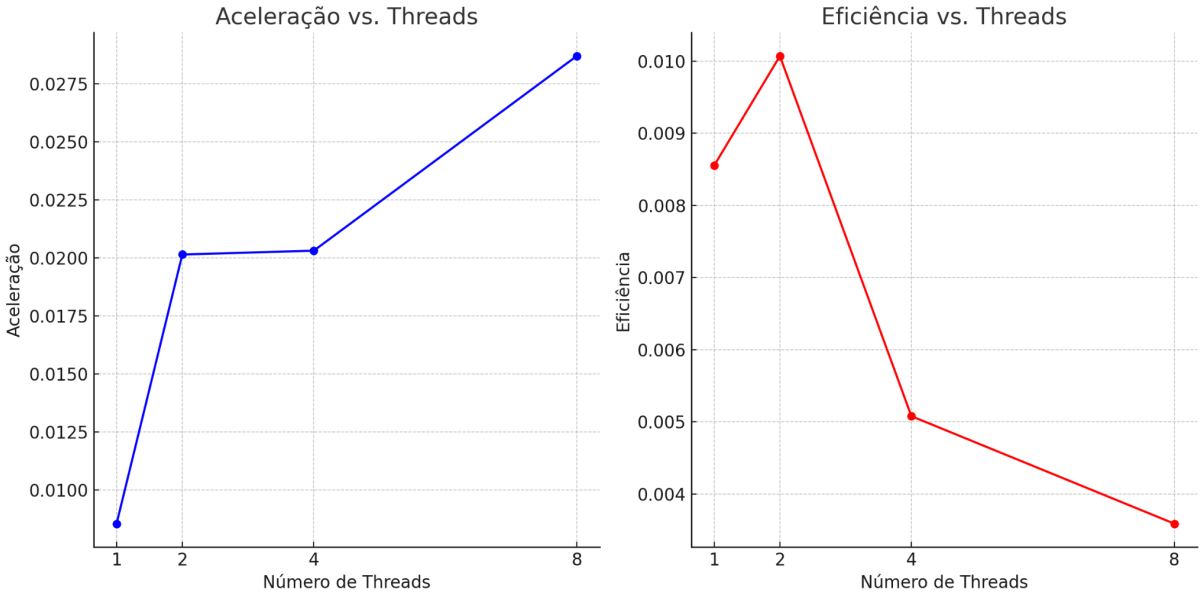
Aceleração: 0.00100159
Eficiência: 0.00025040

Concorrente - 8 Threads						
	1°	**2°**	**3°**	**4°**	**5°**	**Média**
Inicialização	0.012361	0.009979	0.018407	0.007079	0.009976	0.011560
Multiplicação	23.315649	23.741099	26.165635	28.958642	27.129145	25.862034
Finalização	0.102475	0.032727	0.012136	0.013531	0.017112	0.035596

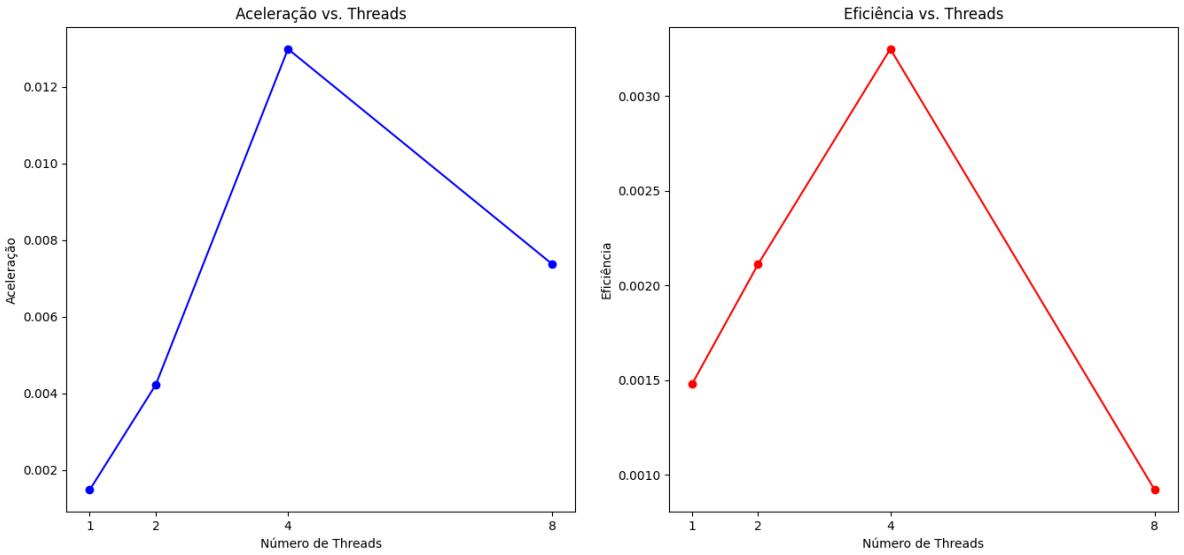
Aceleração: 0.00182337
Eficiência: 0.00022792

Análise dos Dados Obtidos

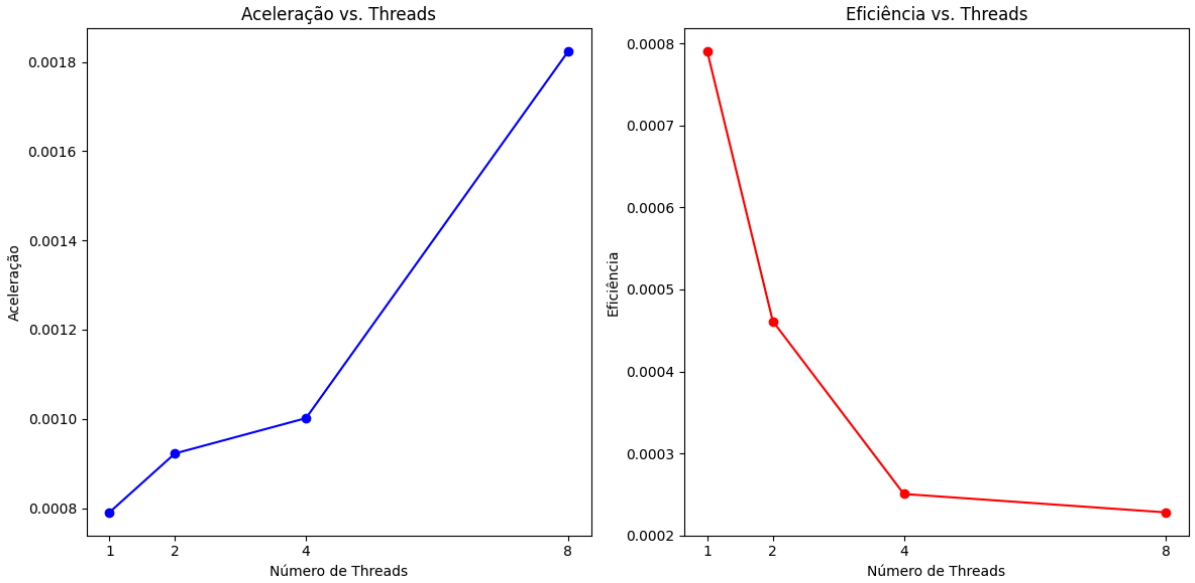
- **500 x 500**



- **1000 x 1000**



- **2000 x 2000**



Nas matrizes de 500 x 500 a diferença absoluta de tempo não foi perceptível, pois tanto sequencialmente quanto concorrentemente o tempo total ficou abaixo de 1 segundo, porém, percentualmente, a diferença foi absurda, chegando a uma queda de mais de 70% no

tempo de multiplicação no comparativo com 8 threads. A partir da multiplicação por 1000 x 1000, as diferenças começaram a ser mais visíveis. Entre a execução do programa sequencialmente e com 8 threads, foi possível ver uma diferença média de mais de 5 segundos (queda de mais de 70%). A multiplicação 2000 x 2000 seguiu um padrão semelhante com uma queda pouco maior de 60%. Em todos os casos, a diferença do tempo de multiplicação serviu como uma visualização do impacto que o uso de concorrência pode fornecer ao usuário.

Deve-se destacar que as quedas começaram a se mostrar a partir do uso de 2 threads, sendo que o uso de apenas uma thread era, por vezes, pior do que a multiplicação sequencial. Isso se dá, primeiro, ao tempo de criação da thread e que o uso de apenas uma thread é extremamente similar a própria aplicação sequencial.

Agora, a aceleração mede o quanto uma tarefa foi executada mais rapidamente conforme o número de recursos cresceu. Uma analogia seria que, se um funcionário realiza uma atividade 24 horas, três funcionários realizam em 8 horas e doze funcionários em 2 horas. Embora seja uma visão ingênua já que considera não haver comunicação entre funcionários e não haver queda de produtividade, ela pode ser parcialmente vista nesse problema. Em todos os casos, a aceleração cresce indicando que o aumento de threads provou-se útil para diminuir o tempo da multiplicação. O único caso destoando foi 1000 x 1000 em que houve uma considerável queda entre 4 e 8 threads. Aparentemente, indicando que o aumento de threads impactou pouco no ganho de tempo.

Por fim, discute-se a eficiência a qual mensura a queda de velocidade por cada thread. Na analogia anterior, os trabalhadores não dedicam 100% do tempo ao trabalho, pois podem diminuir o ritmo, por diversos fatores. Isso também impacta diretamente a aceleração. Se os

três funcionários anteriores trabalharem, a 80% do ritmo do caso de apenas um trabalhador, a aceleração real é de 2,4x ao invés de 3x. Em todos os casos, houve uma significativa queda de 4 para 8 threads no nível de eficiência, indicando, portanto, que cada thread passou, individualmente, a executar menos trabalho.

Em suma, compensa-se o uso de concorrência em problemas de larga escala e em computadores com altas quantidades de núcleos para maximizar a velocidade de execução da tarefa.

Referências

- https://stackoverflow.com/questions/29970813/parallel-speedup-and-efficiency
- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/kn-old/strc/html/node9.html