Relatório de Avaliação do Laboratório 3

Manoel Marcelo da Silva
Instituto de Computação – Universidade Federal do Rio de Janeiro
manoelms@ic.ufrj.br

14 de abril de 2024

Resumo

Este relatório mostrará as informações decorrentes do tempo de execução e o ganho de desempenho obtido em uma solução concorrente para o problema de multiplicação de matrizes.

1 Introdução

O problema a ser solucionado é uma simples multiplicação matricial, de tal forma que as matrizes sejam A_{nxm} e B_{mxk} e seu resultado C_{nxk}, de forma sequencial e concorrente. Apesar de existir algoritmos mais eficientes para a multiplicação, usaremos um método "ingênuo"para justamente calcularmos mais facilmente a eficiência e velocidade do método concorrente em relação ao sequencial. O método de divisão de Threads escolhida foi a separação por linha em blocos divididos igualmente entre as threads (com o resto calculado na thread principal), devido aos poucos núcleos dedicados em processador comum, na ausência de um acelerador gráfico/GPU ou NPU.

2 Metodologia

Iremos rodar 3 tipos de matrizes randomizadas (float) quadradas de tamanho 500x500, 1000x1000 e 2000x2000 em 4 valores de threads criadas, sendo elas 1,2,4 e 8. Tais cálculos serão feitos em um MacBook Air M2 de 8 GB, que possui 8 núcleos - sendo 4 deles de eficiência - com um clock de 3.2 GHz cinco vezes para cada thread e em cada tamanho da matriz.

Após os cálculos, as tabelas e a avaliação do desempenho serão feitas, calculando o tempo de execução em três etapas:

- Inicialização - Que se destina à leitura e alocação das matrizes
- Processamento - A multiplicação em si
- Finalização - Escrita e liberação da memória alocada

3 Dados Levantados

Com os tempos de execução calculados, foi calculado a média e variância entre as execuções de mesmo tipo, ou seja, mesmo número de threads e a mesma dimensão da matriz processada. Foi constatado que a variância é extremamente pequena entre os testes e que o tempo de inicialização e de finalização é irrisório em relação ao processamento. Os dados tabelados se encontram no Github e no Google Sheets

3.1 Matriz 500x500

3.2 Matriz 1000x1000

3.3 Matriz 2000x2000

Nº Threads	Inicialização (s)	Processamento (s)	Finalização(s)	Total(s)
seq	0.002311	0.449358	0.000260	0.451929
seq	0.000660	0.457047	0.000302	0.458009
seq	0.000497	0.453700	0.000241	0.454438
seq	0.000521	0.453118	0.000255	0.453895
seq	0.000673	0.458231	0.000255	0.459159
1	0.001959	0.463450	0.000212	0.465621
1	0.000627	0.455324	0.001190	0.457141
1	0.000427	0.452762	0.000235	0.453425
1	0.002205	0.454423	0.001513	0.458141
1	0.000708	0.455586	0.000242	0.456536
2	0.000634	0.246350	0.000228	0.247212
2	0.000723	0.246920	0.000235	0.247878
2	0.000723	0.245708	0.000243	0.246674
2	0.000660	0.246088	0.000268	0.247017
2	0.000611	0.244086	0.000264	0.244961
4	0.000447	0.137721	0.000229	0.138397
4	0.000647	0.129552	0.000215	0.130414
4	0.000497	0.135309	0.000214	0.136021
4	0.000747	0.132902	0.000244	0.133893
4	0.000541	0.134241	0.000221	0.135003
8	0.000510	0.104644	0.000251	0.105405
8	0.000451	0.107751	0.000214	0.108416
8	0.000649	0.110337	0.000222	0.111208
8	0.000580	0.103544	0.000234	0.104358
8	0.000591	0.113958	0.000222	0.114772

Tabela 1: 500x500

Nº Threads	Inicialização (s)	Processamento (s)	Finalização(s)	Total(s)
Seq	0.000932	0.454291	0.000263	0.455486
1	0.001185	0.456309	0.000678	0.458173
2	0.000670	0.245830	0.000248	0.246748
4	0.000576	0.133945	0.000225	0.134746
8	0.000556	0.108047	0.000229	0.108832

Tabela 2: Média dos valores em 500x500

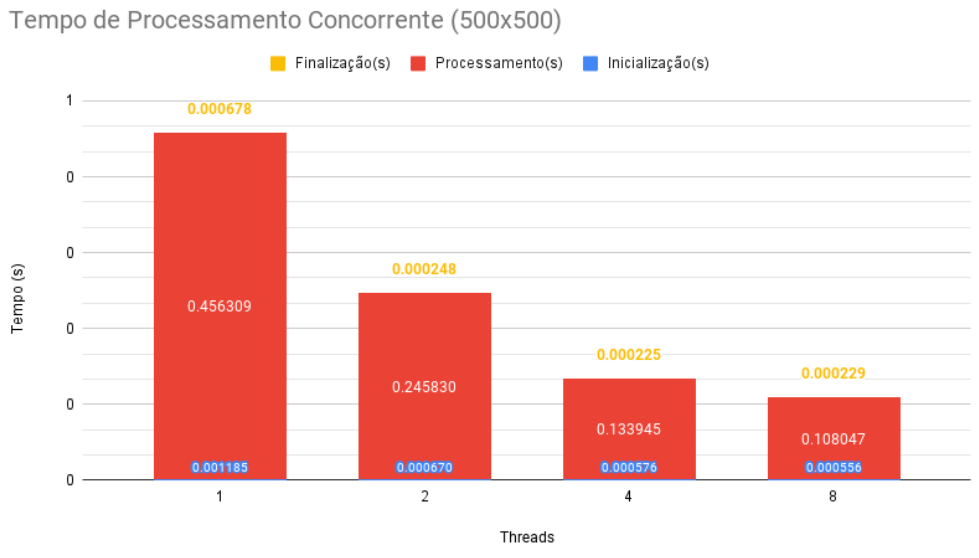

Figura 1: Tempo em média de processamento na multiplicação de matrizes 500x500

Nº Threads	Inicialização (s)	Processamento (s)	Finalização(s)	Total(s)
seq	0.005624	3.483351	0.001614	3.490589
seq	0.003766	3.480884	0.001312	3.485963
seq	0.002108	3.501307	0.001358	3.504773
seq	0.002064	3.507335	0.000916	3.510315
seq	0.002095	3.466902	0.001168	3.470166
1	0.005723	3.519441	0.000894	3.526058
1	0.002086	3.515594	0.001264	3.518944
1	0.002190	3.551591	0.000933	3.554714
1	0.001938	3.555950	0.001530	3.559418
1	0.002056	3.556259	0.001183	3.559498
2	0.002009	1.840374	0.001559	1.843943
2	0.002129	1.841255	0.000898	1.844282
2	0.002041	1.876243	0.000793	1.879077
2	0.002106	1.838323	0.000651	1.841081
2	0.001927	1.840269	0.000937	1.843133
4	0.002292	0.984691	0.000664	0.987647
4	0.002116	0.978958	0.001130	0.982204
4	0.002189	0.976746	0.001340	0.980275
4	0.001089	0.968883	0.000701	0.970673
4	0.002000	0.983530	0.000673	0.986203
8	0.001999	0.761660	0.001340	0.764999
8	0.002168	0.766118	0.001382	0.769668
8	0.002156	0.785634	0.001299	0.789090
8	0.002129	0.775413	0.000783	0.778325
8	0.002233	0.775086	0.001244	0.778564

Tabela 3: 1000x1000

Nº Threads	Inicialização (s)	Processamento (s)	Finalização(s)	Total(s)
Seq	0.003131	3.487956	0.001274	3.492361
1	0.002799	3.539767	0.001161	3.543726
2	0.002042	1.847293	0.000968	1.850303
4	0.001937	0.978562	0.000902	0.981400
8	0.002137	0.772782	0.001210	0.776129

Tabela 4: Média dos valores em 1000x1000

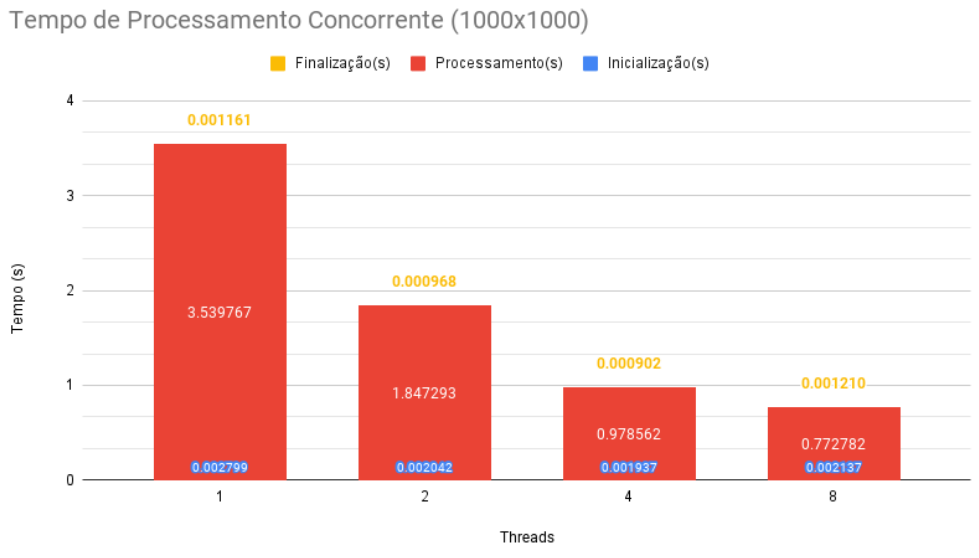

Figura 2: Tempo em média de processamento na multiplicação de matrizes 1000x1000

Nº Threads	Inicialização (s)	Processamento (s)	Finalização(s)	Total(s)
seq	0.019171	29.610459	0.004504	29.634134
seq	0.008602	28.967252	0.004481	28.980335
seq	0.007831	29.750721	0.005468	29.764020
seq	0.007524	28.985092	0.003483	28.996099
seq	0.005621	29.120774	0.002991	29.129387
1	0.019152	30.100505	0.004038	30.123695
1	0.005476	30.092788	0.004087	30.102351
1	0.007139	30.165426	0.003110	30.175675
1	0.008150	29.899561	0.003021	29.910732
1	0.006134	29.864412	0.003042	29.873588
2	0.007404	15.587707	0.003126	15.598237
2	0.007581	15.370734	0.003026	15.381341
2	0.007495	15.418099	0.004343	15.429938
2	0.007677	15.477249	0.002984	15.487910
2	0.006930	15.492148	0.002846	15.501924
4	0.006049	7.897382	0.002931	7.906362
4	0.007864	8.173372	0.002982	8.184218
4	0.005973	8.261168	0.002939	8.270081
4	0.006230	8.399885	0.003935	8.410050
4	0.007455	8.233865	0.002797	8.244117
8	0.008172	6.517680	0.002819	6.528672
8	0.007440	6.311100	0.002631	6.321171
8	0.007804	6.615469	0.002616	6.625889
8	0.007219	6.371789	0.002605	6.381613
8	0.004255	6.366431	0.002741	6.373427

Tabela 5: 2000x2000

Nº Threads	Inicialização (s)	Processamento (s)	Finalização(s)	Total(s)
Seq	0.009750	29.286860	0.004185	29.300795
1	0.009210	30.024538	0.003460	30.037208
2	0.007417	15.469187	0.003265	15.479870
4	0.006714	8.193134	0.003117	8.202966
8	0.006978	6.436494	0.002682	6.446154

Tabela 6: Média dos valores em 2000x2000

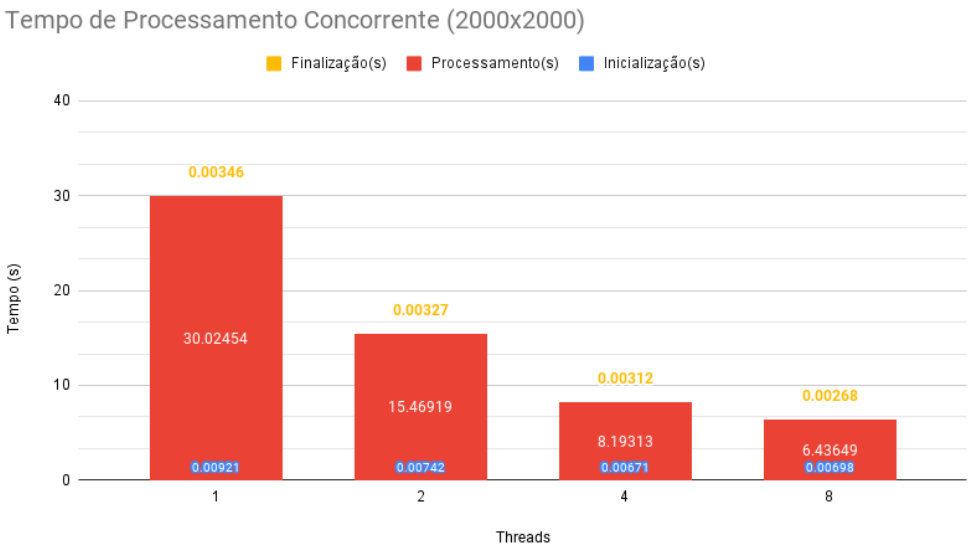

Figura 3: Tempo em média de processamento na multiplicação de matrizes 2000x2000

4 Analise da Aceleração e Eficiência

A partir dos dados coletados acima, podemos calcular a aceleração e a eficiência do código em relação ao sequencial. Para o cálculo da aceleração é a razão entre o tempo de execução da versão sequencial do algoritmo ($Ts(n)$) e o tempo de execução da versão concorrente ($Tp(n,p)$) usando p processadores. Enquanto o cálculo da eficiência é a razão entre a aceleração e o número de processadores usado.

4.1 Aceleração

Com o cálculo mostrado acima temos os seguintes gráficos:

4.1.1 500x500

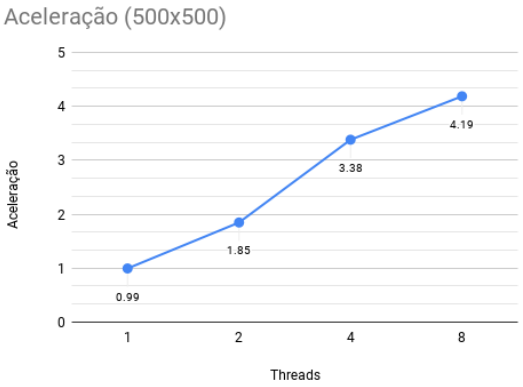

Figura 4: Aceleração (500x500).

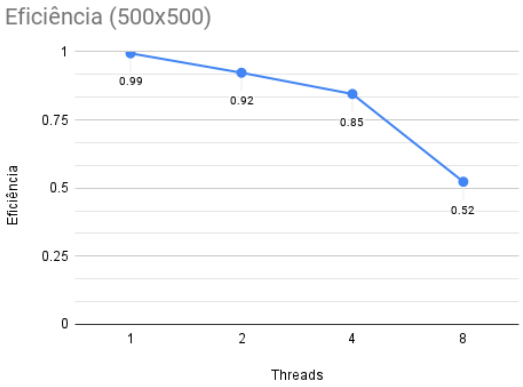

Figura 5: Eficiência (500x500).

4.1.2 1000x1000

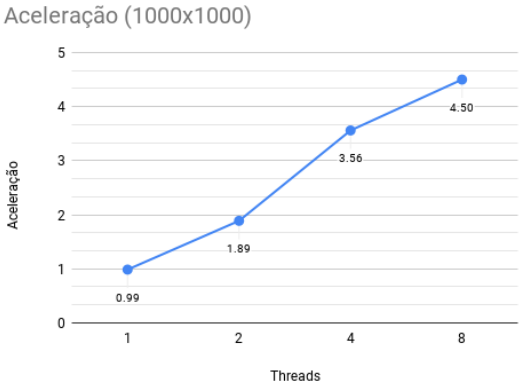

Figura 6: Aceleração (1000x1000).

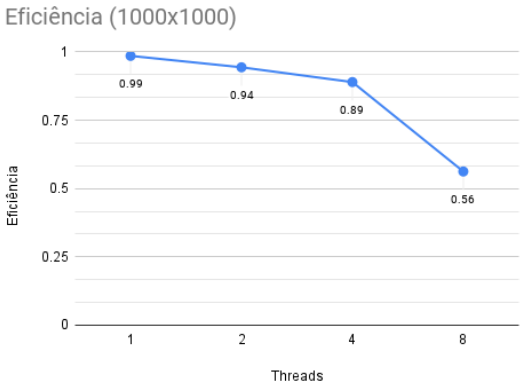

Figura 7: Eficiência (1000x1000).

4.1.3 2000x2000

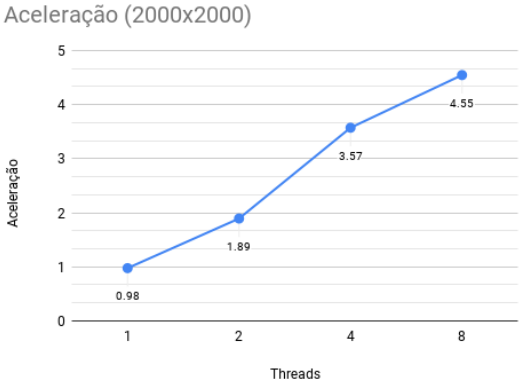

Figura 8: Aceleração (2000x2000).

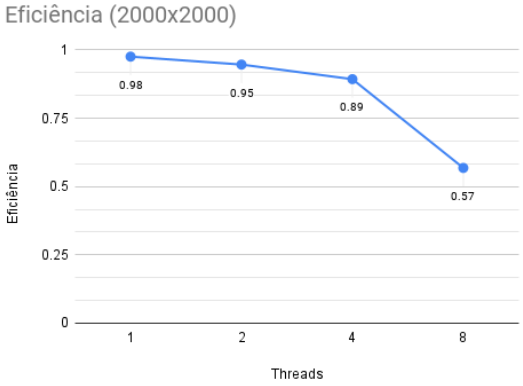

Figura 9: Eficiência (2000x2000.

5 Conclusão

Como podemos perceber pelos gráficos acima, a versão concorrente é muito mais eficiente e ágil que a sequencial, exceto no caso em que há apenas uma thread, já que ocorre uma penalidade em relação ao processamento de criação das threads.

Contudo, é visível que a eficiência decai após o uso de 4 núcleos. Isso pode ser explicado por conta da presença dos núcleos de eficiência do processador, já que o sistema operacional - no caso, MacOS (baseado em UNIX) - escalona a thread para esses núcleos que são mais fracos em termos de performance. Mas, até 4 threads a aceleração sobe em rítmo quase linear, com eficiência bem próxima de 1 - o ideal.

Portanto, o código concorrente traz muitas vantagens em relação ao tradicional sequencial em termos de eficiência e performance, podendo ser escalonado para um processador com mais núcleos ou um acelerador para mais desempenho.