



Universidade de São Paulo  
Instituto de Ciências Matemáticas e de Computação (ICMC)  
SSC0903 - Computação de Alto Desempenho

## Resultados para o Método Iterativo de Jacobi-Richardson

Matheus Yasuo Ribeiro Utino - 11233689

Pedro Ribas Serras - 11234328

Vinícius Silva Montanari - 11233709

Docente: Dr. Paulo Sérgio Lopes de Souza

São Carlos  
27 de maio de 2022

# **Sumário**

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Resultados</b>	<b>2</b>
2.1	Dimensão da matriz: 100 . . . . .	2
2.2	Dimensão da matriz: 1000 . . . . .	5
2.3	Dimensão da matriz: 10000 . . . . .	8
2.4	Dimensão da matriz: 20000 . . . . .	11
<b>3</b>	<b>Conclusão</b>	<b>14</b>

## **Lista de Figuras**

1	Tempo de resposta para ordem de matriz de 100. . . . .	4
2	Speedup para ordem de matriz de 100. . . . .	4
3	Eficiência para ordem de matriz de 100. . . . .	5
4	Tempo de resposta para ordem de matriz de 1000. . . . .	7
5	Speedup para ordem de matriz de 1000. . . . .	7
6	Eficiência para ordem de matriz de 1000. . . . .	8
7	Tempo de resposta para ordem de matriz de 10000. . . . .	10
8	Speedup para ordem de matriz de 10000. . . . .	10
9	Eficiência para ordem de matriz de 10000. . . . .	11
10	Tempo de resposta para ordem de matriz de 20000. . . . .	13
11	Speedup para ordem de matriz de 20000. . . . .	13
12	Eficiência para ordem de matriz de 20000. . . . .	14

## **Lista de Tabelas**

1	Resultados para a ordem da matriz de 100.	3
2	Resultados para a ordem da matriz de 1000	6
3	Resultados para a ordem da matriz de 10000	9
4	Resultados para a ordem da matriz de 20000	12

## 1 Introdução

Agora serão analisados todos os resultados para o método iterativo de Jacobi-Richardson, tanto para o algoritmo sequencial como para o paralelo. No caso, foi variado o tamanho da matriz em quatro valores: 100, 1000, 10000 e 20000. Além disso, para o algoritmo paralelo o número de *threads* também foi variado para 4, 6 e 8 *threads*.

Para o cálculo dos tempo será executado 30 vezes o algoritmo e posteriormente calcular a média dos valores, desvio padrão, speedups e eficiências. Com isso, torna-se possível comparar efetivamente os resultados e tomar as conclusões.

## 2 Resultados

Agora efetivamente serão demonstrados os resultados, em que será inicialmente analisado para cada dimensão da matriz escolhida.

### 2.1 Dimensão da matriz: 100

Os resultados para a dimensão 100 estão contidos na tabela 1.

Iteração	Sequencial	T = 4	T = 6	T = 8
1	0,000399	0,000767	0,003744	0,001942
2	0,000395	0,000313	0,002427	0,003828
3	0,000556	0,000329	0,001956	0,002010
4	0,000396	0,002062	0,004275	0,003810
5	0,000396	0,001417	0,000270	0,003619
6	0,000395	0,000405	0,000756	0,001787
7	0,000578	0,000637	0,001087	0,003091
8	0,000693	0,000299	0,000479	0,002378
9	0,000517	0,000403	0,002113	0,002192
10	0,000372	0,000426	0,002333	0,002483
11	0,000353	0,000584	0,002334	0,001110
12	0,000346	0,000412	0,000596	0,002831
13	0,000499	0,000436	0,002209	0,002202
14	0,000682	0,000344	0,002129	0,000562
15	0,000505	0,000483	0,002437	0,002389
16	0,000607	0,000393	0,000376	0,006479
17	0,000606	0,000286	0,000272	0,004260
18	0,000602	0,000365	0,002201	0,002650
19	0,000384	0,000370	0,000547	0,002089
20	0,000382	0,000418	0,002166	0,004334
21	0,000376	0,000301	0,001655	0,001922
22	0,000513	0,000459	0,002394	0,001623
23	0,000576	0,000353	0,000763	0,002702
24	0,000534	0,002200	0,001472	0,001357
25	0,000554	0,000490	0,002722	0,002392
26	0,000555	0,000460	0,002582	0,003109
27	0,000807	0,000346	0,002113	0,002253
28	0,000478	0,000450	0,001581	0,001788
29	0,000520	0,000769	0,000695	0,002519
30	0,000491	0,000492	0,002939	0,002630
Total	0,015067	0,017469	0,053623	0,078341
Média	0,000509	0,000582	0,002113	0,0023905
Desvio	0,000114	0,000473	0,001029	0,001140
Speedup	-	0,874120	0,240890	0,212926
Eficiência	-	0,218530	0,040148	0,026616

Tabela 1: Resultados para a ordem da matriz de 100.

Da tabela 1, nota-se que para matriz de ordem 100 o algoritmo sequencial acabou apresentando um resultado melhor do que o paralelo, para qualquer número de threads, isso se deve ao grande custo para criação de threads e o overhead de comunicação. E conforme esperado, quanto maior o número de threads maior será o tempo para uma ordem pequena. Os tempos podem ser visualmente comparados pelo gráfico de barras da figura 1

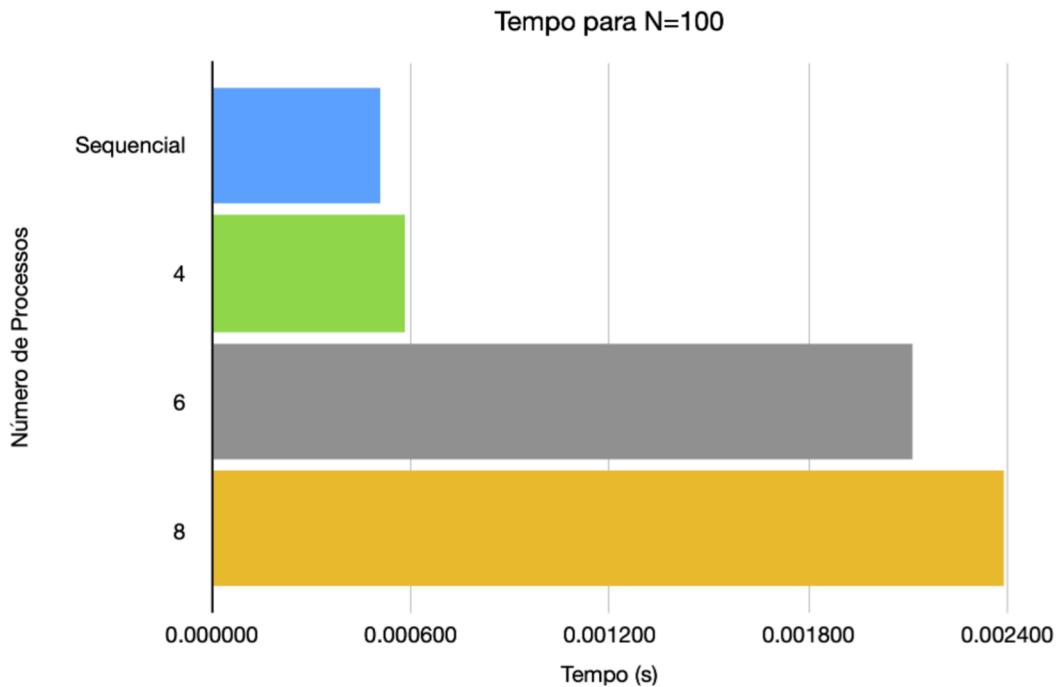


Figura 1: Tempo de resposta para ordem de matriz de 100.

Com esses dados é possível também plotar os gráficos de speedup que está contido na figura 2 e de eficiência na figura 3 para a ordem da matriz de 100.

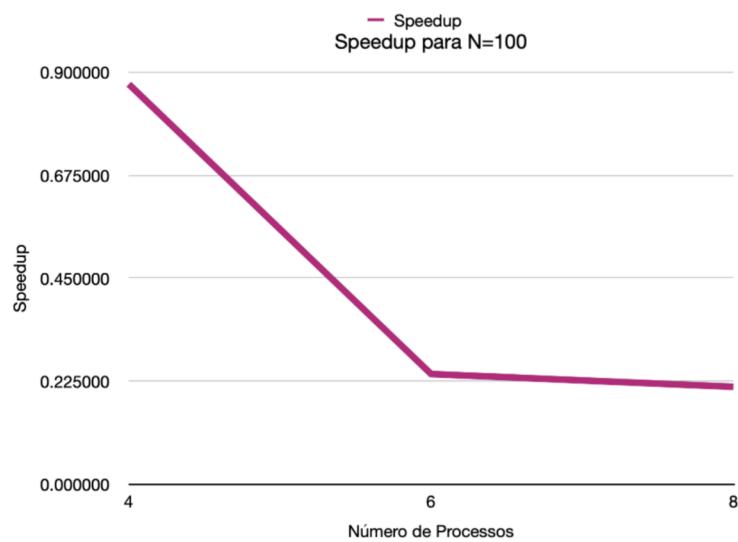


Figura 2: Speedup para ordem de matriz de 100.

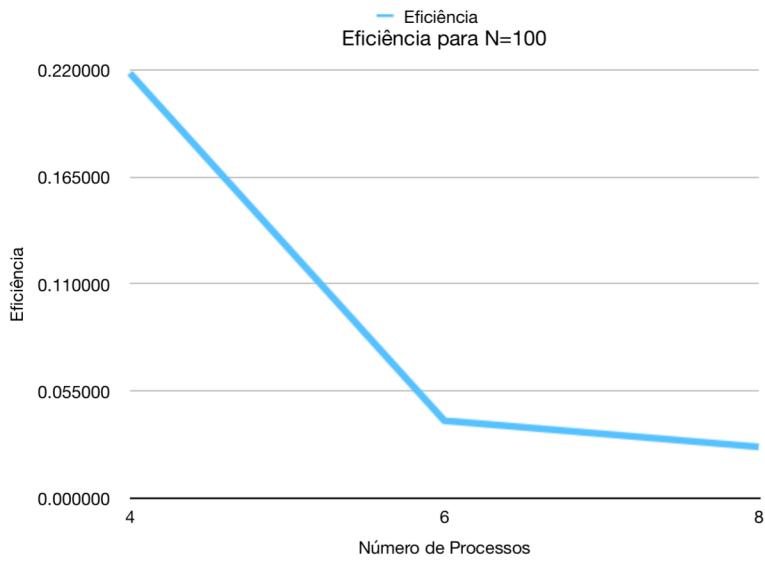


Figura 3: Eficiência para ordem de matriz de 100.

Assim como discutido anteriormente, como o tempo paralelo foi aumentando com o número de threads, o speedup acabou decaindo, demonstrando que para esses números de threads escolhidos não valia a pena sua versão paralela. Ademais, a eficiência também foi caindo drasticamente com o aumento do número de threads.

## 2.2 Dimensão da matriz: 1000

Os resultados para a dimensão 1000 estão contidos na tabela 2.

Iteração	Sequencial	T = 4	T = 6	T = 8
1	0,033240	0,017197	0,011762	0,010292
2	0,033117	0,017364	0,012022	0,010306
3	0,033174	0,017275	0,011808	0,008949
4	0,033166	0,008993	0,011835	0,009279
5	0,033155	0,009043	0,011780	0,010334
6	0,033047	0,009374	0,011751	0,008975
7	0,033180	0,009012	0,011841	0,010391
8	0,033154	0,009102	0,011730	0,010315
9	0,033190	0,009083	0,011829	0,010348
10	0,033168	0,009204	0,011924	0,010371
11	0,033136	0,017285	0,011766	0,009923
12	0,033110	0,009057	0,011794	0,010163
13	0,033340	0,009187	0,011724	0,009280
14	0,033148	0,017259	0,011783	0,009837
15	0,033038	0,009135	0,011882	0,010361
16	0,033161	0,009169	0,011964	0,008955
17	0,033073	0,009217	0,011798	0,008965
18	0,033108	0,009108	0,011781	0,010366
19	0,033170	0,009361	0,011828	0,010311
20	0,033142	0,009406	0,011774	0,010147
21	0,033104	0,009382	0,011836	0,010265
22	0,033088	0,009117	0,011896	0,009029
23	0,033266	0,009240	0,011717	0,008937
24	0,033155	0,009048	0,011779	0,010558
25	0,033194	0,009345	0,011772	0,008990
26	0,033118	0,009172	0,011745	0,010358
27	0,033072	0,009460	0,011773	0,008962
28	0,033132	0,009202	0,011674	0,010290
29	0,033272	0,009207	0,011729	0,008945
30	0,033133	0,009167	0,011760	0,008951
Total	0,994551	0,316171	0,354057	0,293153
Média	0,033151	0,010539	0,0117805	0,010155
Desvio	0,000066	0,003067	0,000075	0,000644
Speedup	-	3,145545	2,814057	3,264500
Eficiência	-	0,786386	0,469010	0,408063

Tabela 2: Resultados para a ordem da matriz de 1000

Da tabela 2, nota-se que para matriz de ordem 1000 o algoritmo paralelo se saiu melhor que o sequencial para qualquer número de threads. Mas entre os valores de threads o melhor foi para 8 threads, com isso o aumento do tamanho da matriz acabou por compensar o tempo de criação das threads e do overhead de comunicação. Deve-se ressaltar que ao considerar o pior caso, isto é, o tempo médio mais o desvio padrão, o tempo é decrescente com o aumento do número de threads.

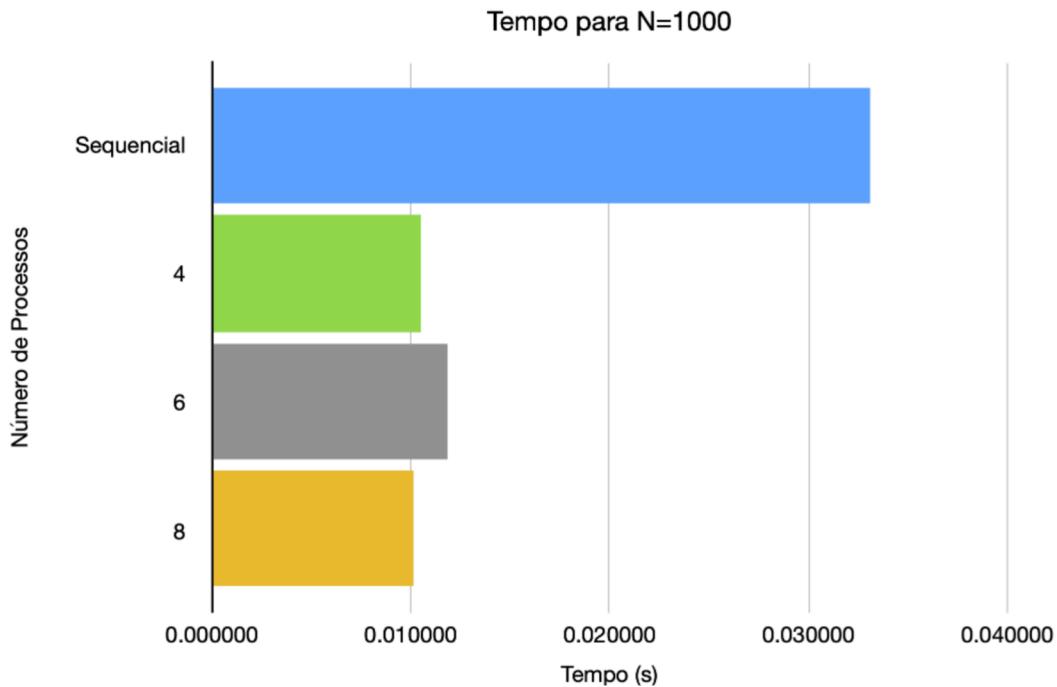


Figura 4: Tempo de resposta para ordem de matriz de 1000.

Plotando novamente os gráficos de speedup que está contido na figura 5 e de eficiência na figura 6 para a ordem da matriz de 1000.

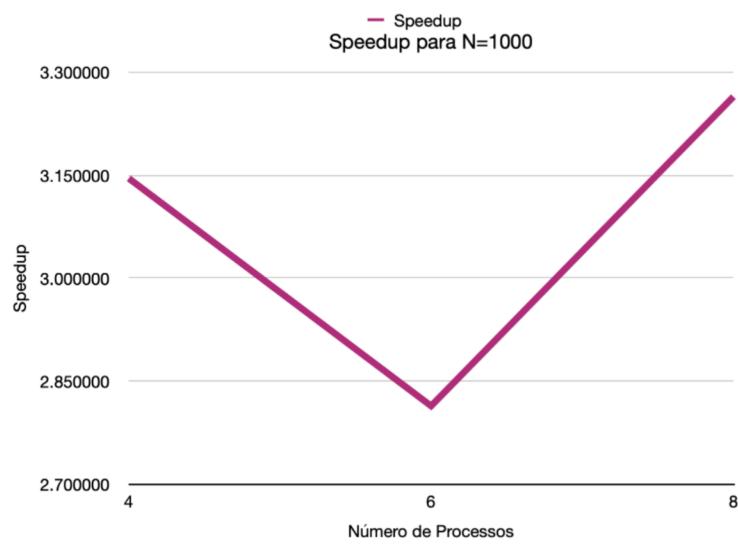


Figura 5: Speedup para ordem de matriz de 1000.

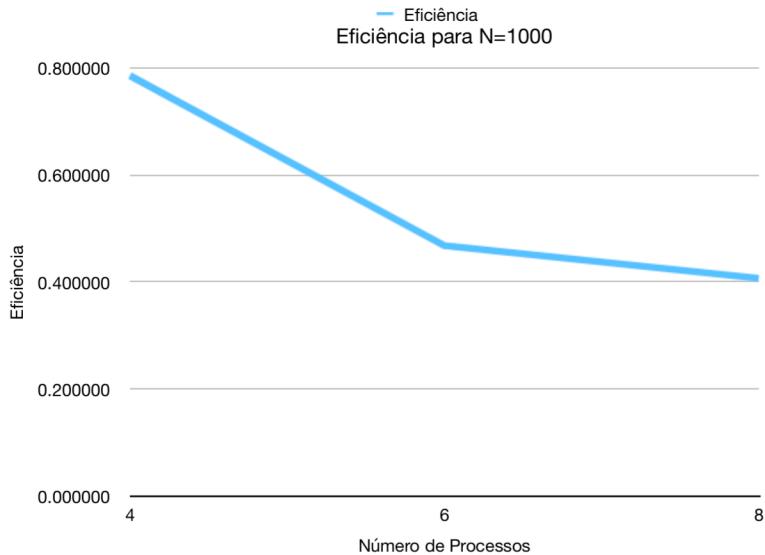


Figura 6: Eficiência para ordem de matriz de 1000.

Para esse caso, o tempo paralelo foi superior ao sequencial, assim o speedup fica superior a 1 em todos os casos. Nota-se que para 6 threads o speedup acaba decaindo em relação a 4, mas posteriormente sobe a um patamar maior que o de 6 quando escolhemos 8 threads. A queda entre 4 e 6 threads acontece quando analisamos a média dos tempos encontrados, mas se analisarmos o pior caso de 4 e 6, fazendo o tempo médio mais o desvio padrão, temos que o pior caso de 6 é melhor que o de 4 threads. Ainda por meio do desvio padrão é interessante notar que os tempos calculados para T igual a 6 são mais constantes do que para T igual 4. Em relação a eficiência, ela acabou decaindo progressivamente com o aumento do número de threads.

### 2.3 Dimensão da matriz: 10000

Os resultados para a dimensão 10000 estão contidos na tabela 3.

Iteração	Sequencial	T = 4	T = 6	T = 8
1	3,293415	1,487506	1,154920	1,280553
2	3,289930	1,483017	1,156399	1,279872
3	3,292011	1,488026	1,161661	1,279897
4	3,298442	1,493055	1,161173	1,282332
5	3,286683	0,943261	1,072851	0,872173
6	3,278967	0,946196	1,079707	0,870897
7	3,313991	0,989066	1,083854	0,870431
8	3,304908	0,940736	1,055514	0,871428
9	3,293563	0,901453	1,077609	0,870635
10	3,306106	0,928098	1,081358	0,871235
11	3,299447	0,943665	1,074598	0,870711
12	3,308833	0,879112	1,083040	0,870662
13	3,272581	0,947631	1,085574	0,871214
14	3,272782	0,944213	1,076222	0,871782
15	3,270753	0,940330	1,070788	0,870668
16	3,270276	0,944438	1,068887	0,874455
17	3,272383	0,940277	1,070204	0,869503
18	3,269542	0,938605	1,070033	0,871806
19	3,272818	0,938237	1,071367	0,869090
20	3,271056	0,897123	1,080616	0,869914
21	3,273334	0,944817	1,074255	0,871484
22	3,272124	0,942934	1,067721	0,870161
23	3,271951	0,947216	1,081331	0,873306
24	3,272211	0,936787	1,086000	0,871696
25	3,271646	0,943536	1,068991	0,873510
26	3,272403	0,902571	1,071630	0,871560
27	3,270619	0,948194	1,079815	0,870736
28	3,272245	0,901674	1,084994	0,871007
29	3,271629	0,938500	1,074648	0,871852
30	3,270421	0,889538	1,080160	0,869324
Total	98,457070	30,209812	32,605920	27,773894
Média	3,272682	1,006994	1,078658	0,8713315
Desvio	0,014150	0,193116	0,029359	0,141575
Speedup	-	3,249952	3,034031	3,755955
Eficiência	-	0,812488	0,505672	0,469494

Tabela 3: Resultados para a ordem da matriz de 10000

Novamente, conforme esperado o algoritmo paralelo se saiu melhor que o algoritmo sequencial para a ordem 10000 da matriz. Ocorreu o mesmo comportamento do que a matriz de ordem 100, em que o algoritmo para 4 threads se saiu superior ao de 6, e utilizando 8 threads teve uma boa redução do tempo. A queda entre 4 e 6 threads acontece novamente, apresentando a mesma explicação do que para a ordem de matriz 1000.

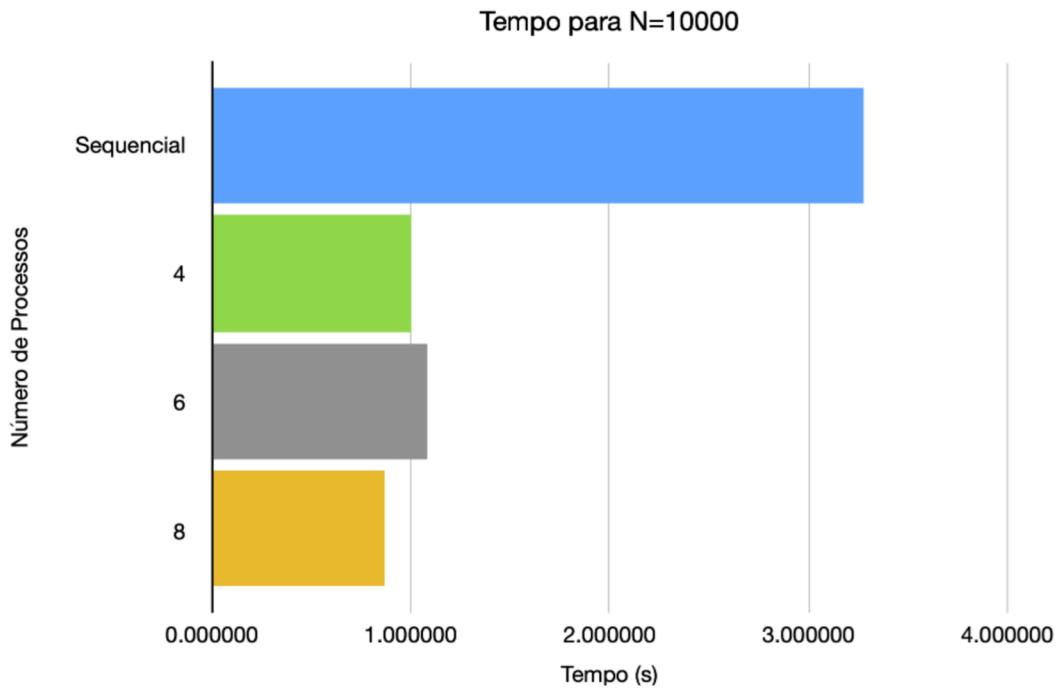


Figura 7: Tempo de resposta para ordem de matriz de 10000.

Agora plotando os gráficos de speedup que está contido na figura 8 e de eficiência na figura 9 para a ordem da matriz de 10000.

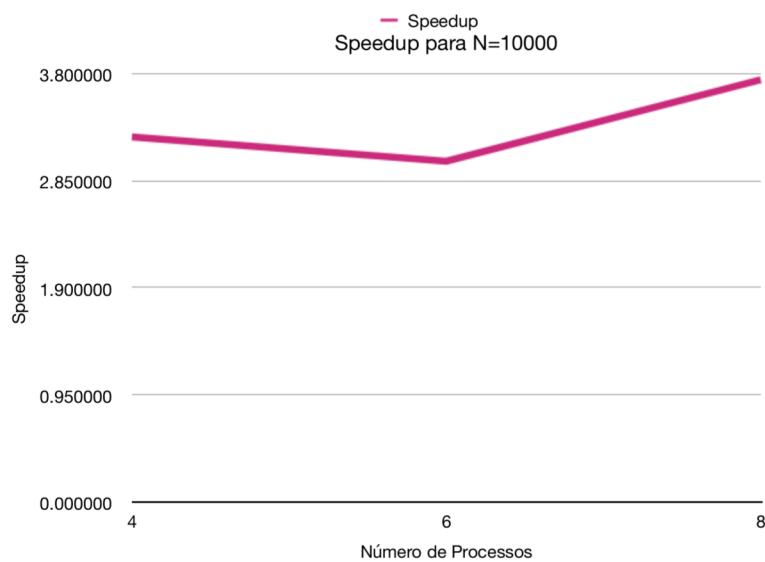


Figura 8: Speedup para ordem de matriz de 10000.

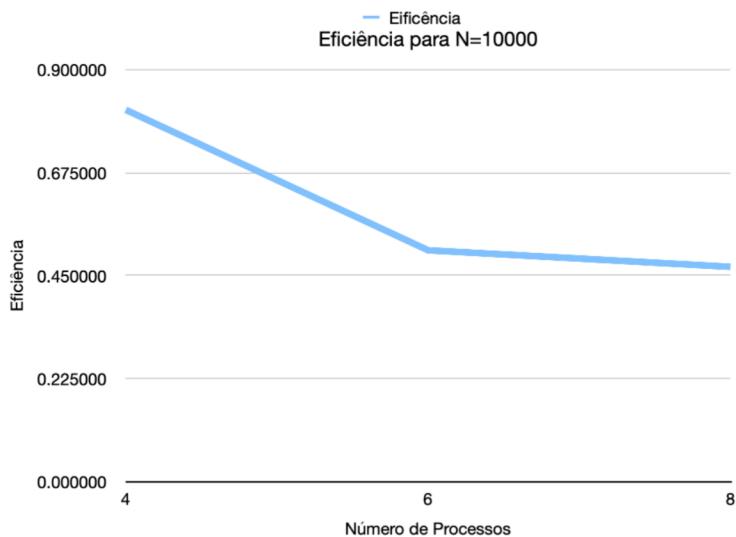


Figura 9: Eficiência para ordem de matriz de 10000.

Para esse caso, ocorreu um aumento do speedup para  $T = 8$ , demonstrando que foi realmente superior a 4 e 6 threads. Já a eficiência foi sofrendo uma redução com o aumento do número de threads.

#### 2.4 Dimensão da matriz: 20000

Os resultados para a dimensão 20000 estão contidos na tabela 4.

Iteração	Sequencial	T = 4	T = 6	T = 8
1	13,137730	3,747938	3,898848	4,608062
2	13,080439	5,549169	3,942508	3,668392
3	13,138350	5,488813	4,825537	4,605203
4	13,137215	6,432422	3,911503	3,966555
5	13,071682	7,247940	3,918826	3,668373
6	13,135428	3,798870	4,803752	4,581963
7	13,123382	5,429143	4,745599	4,579305
8	13,133178	4,381702	4,834469	4,630918
9	17,348020	5,541931	4,834123	4,610381
10	13,361228	5,544564	4,843044	4,675516
11	13,168892	3,813635	3,882558	3,665178
12	13,077994	3,621593	3,896117	3,670689
13	13,088573	3,668751	4,640144	3,770889
14	13,083558	3,743013	3,874654	3,662913
15	13,080384	3,790989	3,896257	3,655936
16	13,083672	3,790834	3,903540	3,658396
17	13,083385	3,711297	3,910570	3,657160
18	13,087763	3,793912	3,899499	3,660652
19	13,083847	3,764573	3,881554	3,659411
20	13,071540	3,740295	3,870601	3,661525
21	13,083627	3,654936	3,903806	3,657011
22	13,072005	3,763793	3,914215	3,651194
23	13,090023	3,824018	3,918884	3,660561
24	13,082658	3,782225	3,915701	3,657524
25	13,079159	3,766783	3,874725	3,658928
26	13,109671	3,710967	3,872059	3,659284
27	13,080295	3,732374	3,916367	3,659627
28	13,135284	3,739595	3,907264	3,660783
29	13,074364	3,808917	3,911824	3,657433
30	13,077964	3,801861	3,907844	3,654306
Total	397,461310	128,186853	123,256392	116,894068
Média	13,083760	4,272895	3,9110365	3,661154
Desvio	0,776184	0,968463	0,383900	0,406573
Speedup	-	3,062036	3,345343	3,573671
Eficiência	-	0,765509	0,557557	0,446709

Tabela 4: Resultados para a ordem da matriz de 20000

Para esse caso, o tempo foi reduzido com o aumento do número threads, o que é esperado para ordem de matrizes maiores, pois o overhead de comunicação acaba reduzindo e o custo de criação passa a valer a pena.

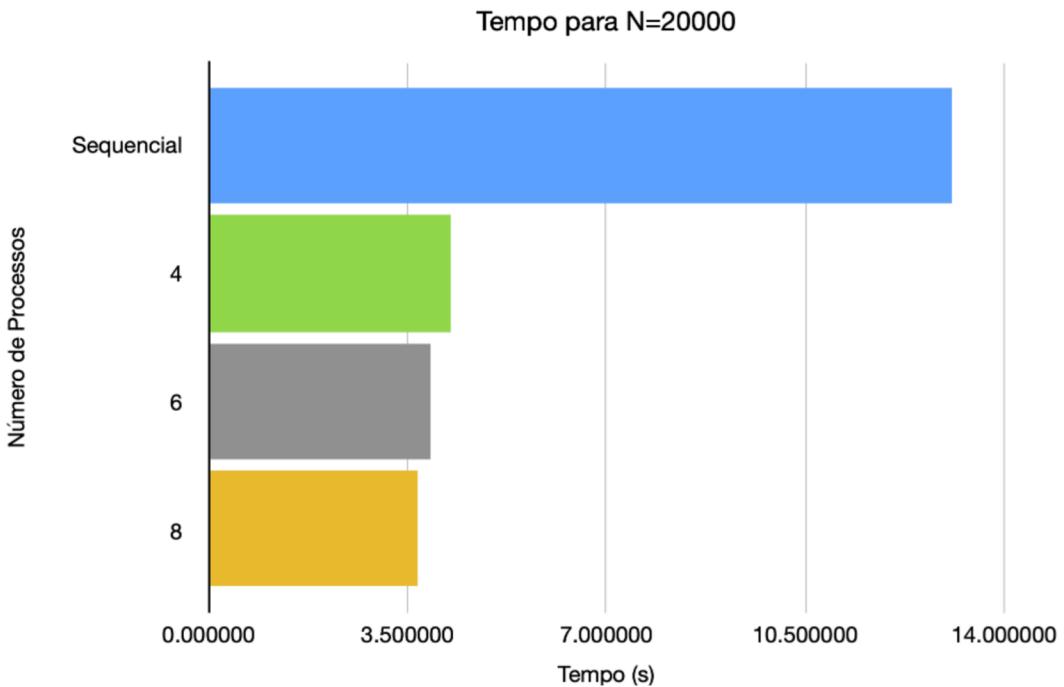


Figura 10: Tempo de resposta para ordem de matriz de 20000.

Por fim, plotando os gráficos de speedup que está contido na figura 11 e de eficiência na figura 12 para a ordem da matriz de 20000.

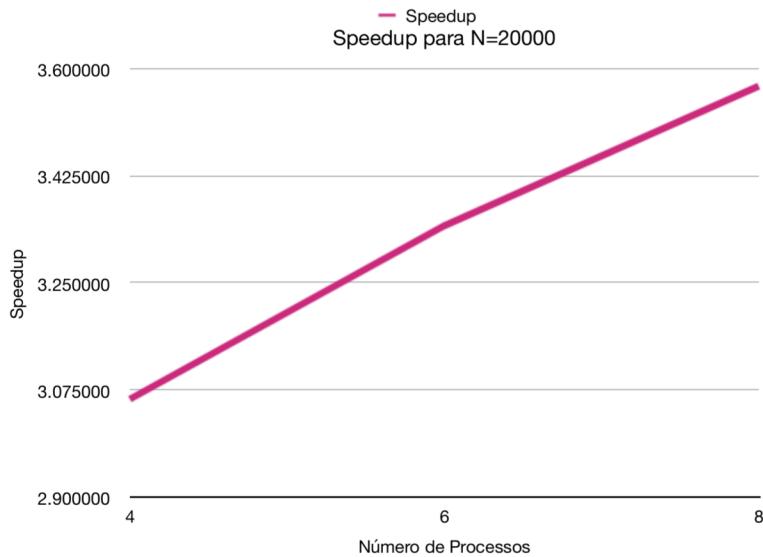


Figura 11: Speedup para ordem de matriz de 20000.

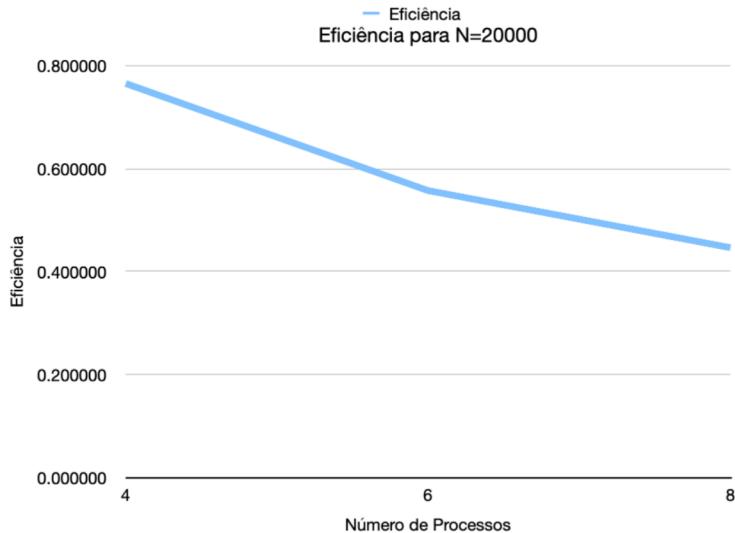


Figura 12: Eficiência para ordem de matriz de 20000.

O speedup apresentou um aumento contínuo, com uma tendência linear para a ordem de 20000. Por fim, a eficiência apresentou a mesma tendência das demais, em que ela vai reduzindo com o número de threads.

### 3 Conclusão

Nota-se pelo valores analisados que, conforme esperado, para uma boa paralelização é vital saber a máquina que estamos rodando, assim como o número de threads a ser escolhido para cada ordem de matriz.

Para ordens de matriz pequenas a paralelização não conseguem compensar o custo de criação de threads e comunicação, sendo, portanto, melhor optar pelo algoritmo sequencial. Já para ordens em faixas intermediárias pode ser interessante a escolha de um baixo número de threads, tendo que ser testado empiricamente para a máquina utilizada para uma boa tomada de decisão. Por fim, para valores elevados é interessante utilizar o número máximo de threads físicas da máquina que provavelmente terá um melhor desempenho.