Relatório TP2-SO

Equipe:

Lucas de Souza Cerveira Pereira
Mikaelle Costa de Santana
Pedro Gabriel Motta Vieira
Michael Willian Pereira Vieira

Questão 1 - Produto Escalar entre vetores

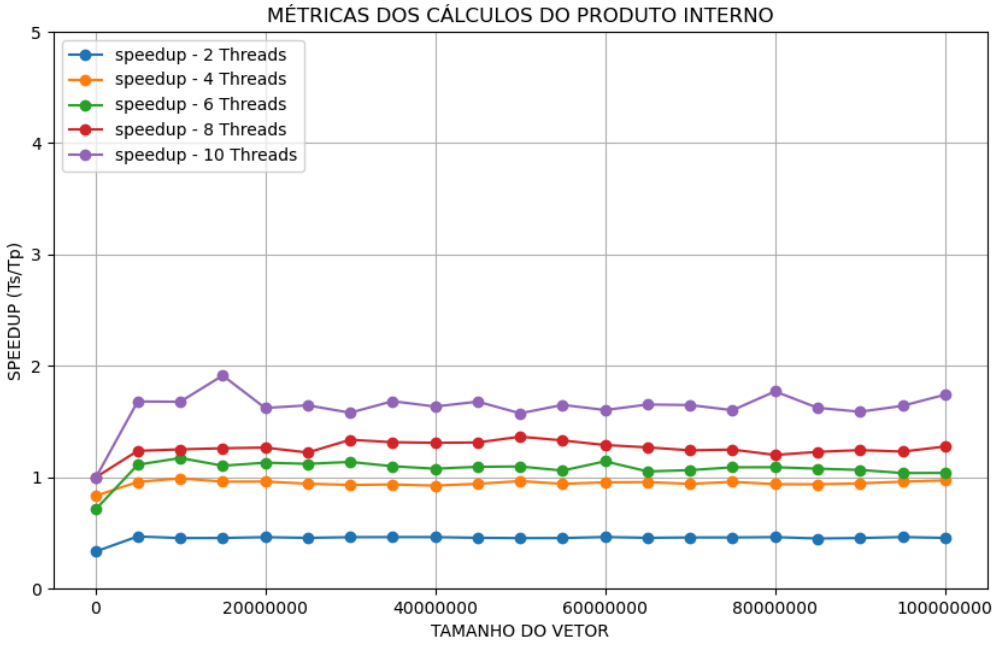
Método escolhido: A versão paralela da multiplicação de matrizes utiliza uma estratégia de decomposição de domínio, distribuindo a responsabilidade pelo cálculo das linhas da matriz resultante entre as threads disponíveis. Cada thread é encarregada de computar integralmente um subconjunto de linhas da matriz C, realizando o produto escalar entre as linhas atribuídas da matriz A e todas as colunas da matriz B dentro de um intervalo de índices [inicio, fim).

A quantidade de linhas processadas por cada bloco é definida pela divisão inteira da ordem da matriz (n) pelo número de threads. Caso essa divisão não seja exata (o número de linhas não é múltiplo do número de threads), a última thread assume o processamento de todas as linhas remanescentes até o final da matriz (n), garantindo que todas as linhas do resultado sejam calculadas.

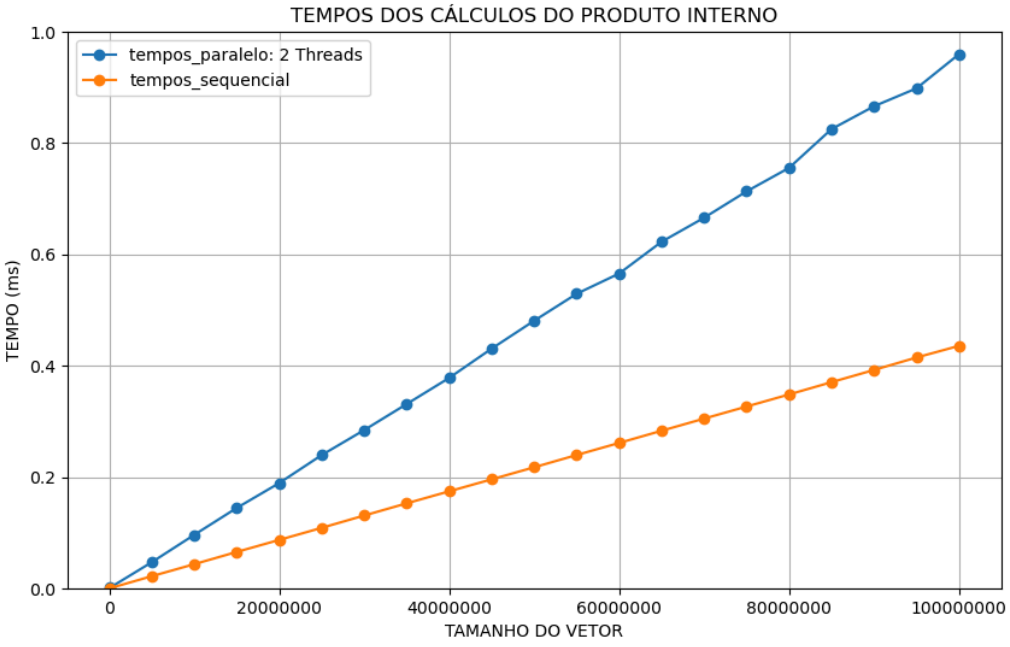
Durante a execução concorrente, cada thread escreve os resultados diretamente nas posições de memória correspondentes da matriz C (dados->C[i][j]). Essa abordagem evita condições de corrida e dispensa o uso de *mutexes* ou operações de redução para a escrita dos valores, uma vez que cada thread opera sobre um conjunto exclusivo de índices de linha (i), garantindo isolamento na escrita dos dados.

Gráficos da máquina com 12 CPUs:

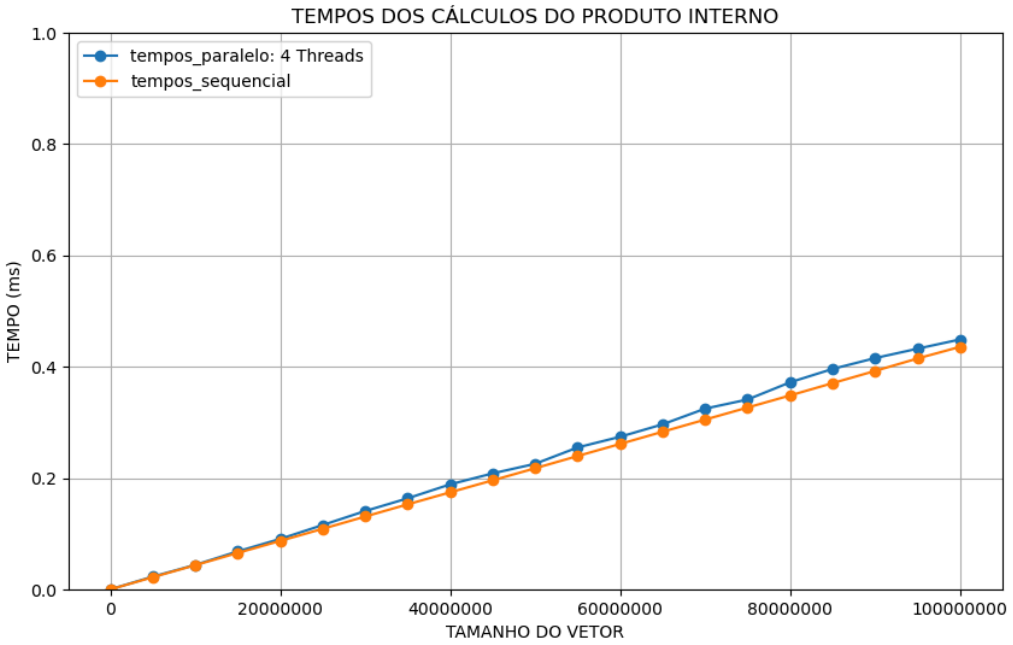
métrica speedup:



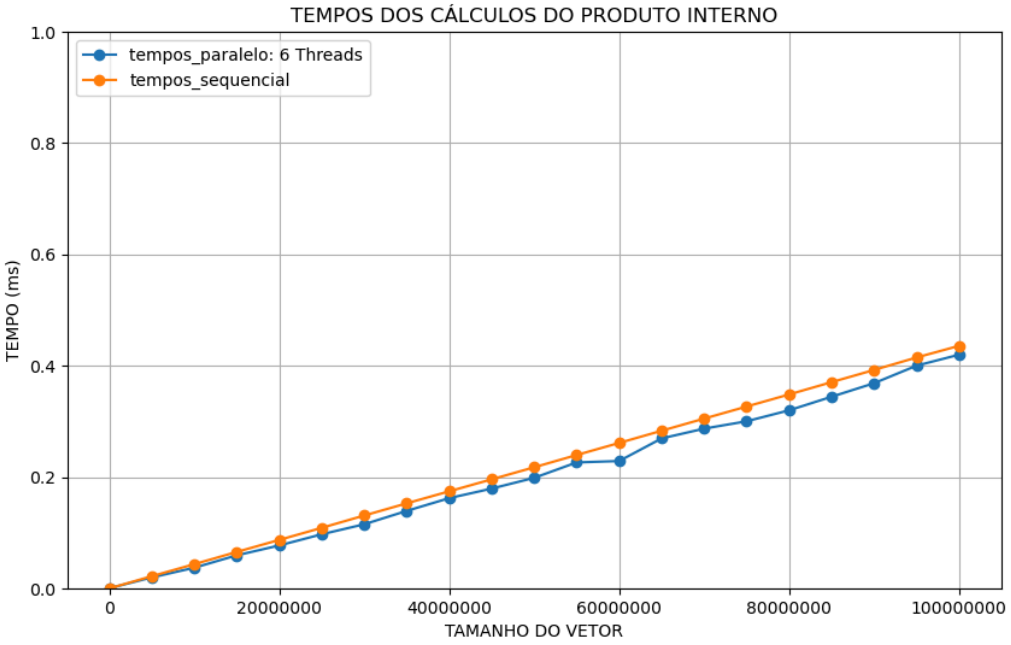
Comparação com 2 threads:



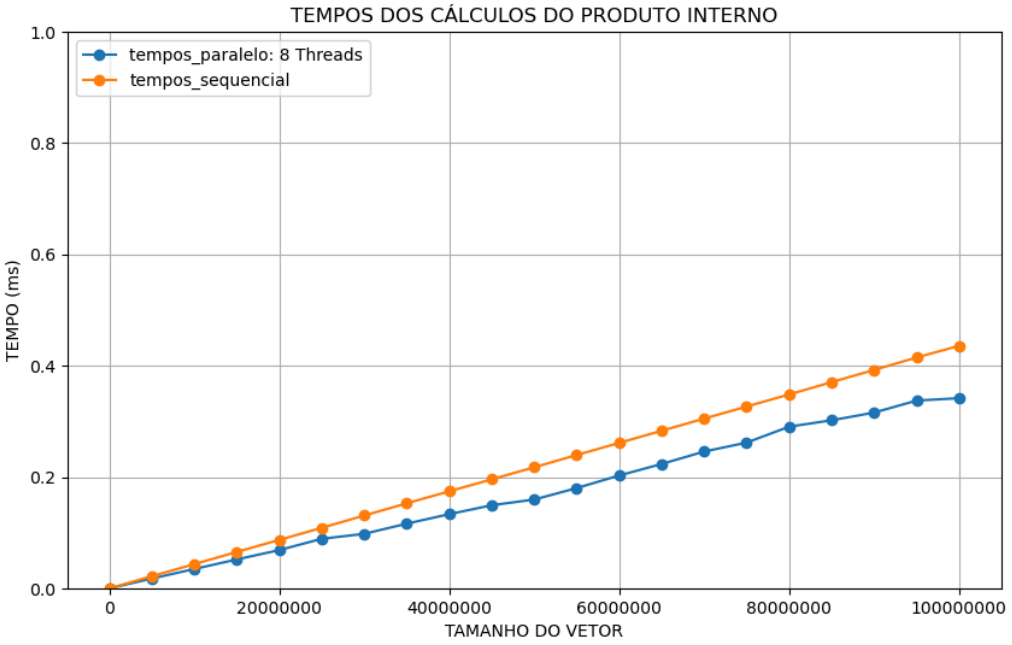
Comparação com 4 threads:



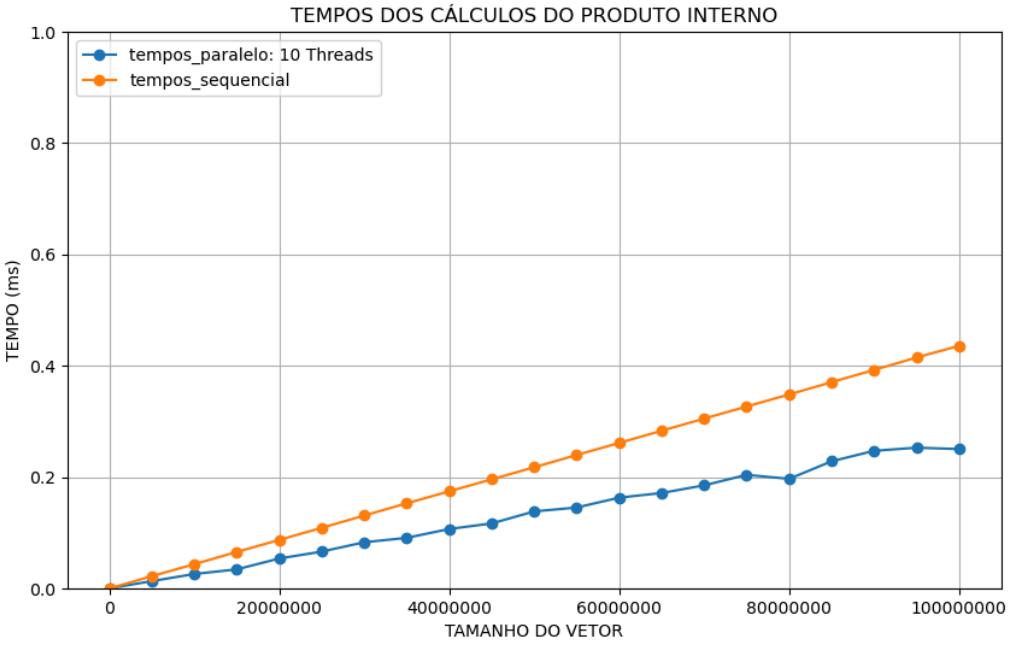
Comparação com 6 threads:



Comparação com 8 threads:

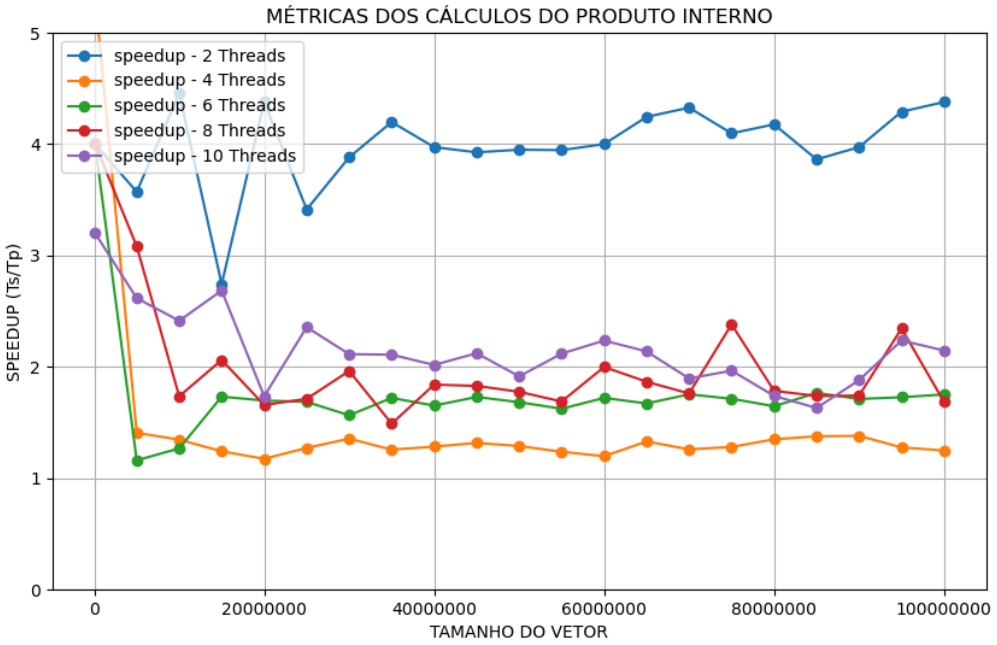


Comparação com 10 threads:

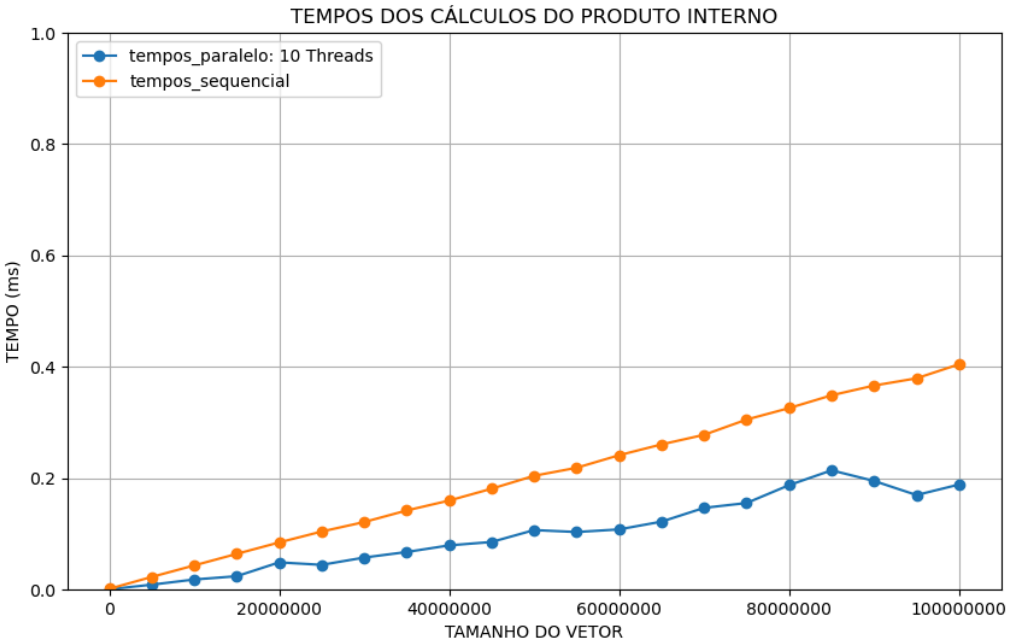


Gráficos da máquina com 16 CPUs:

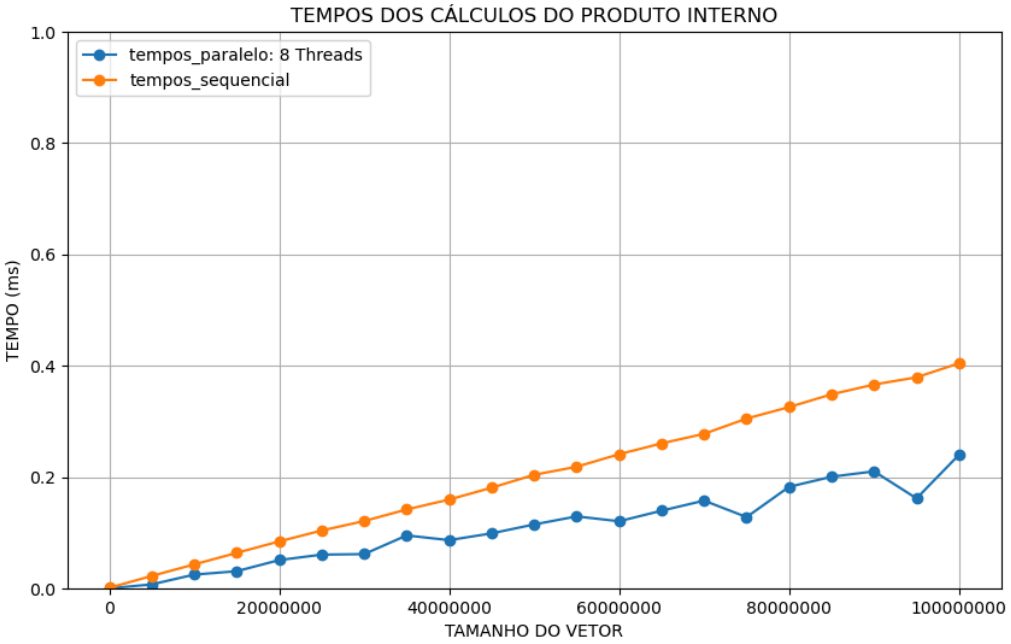
métrica speedup:



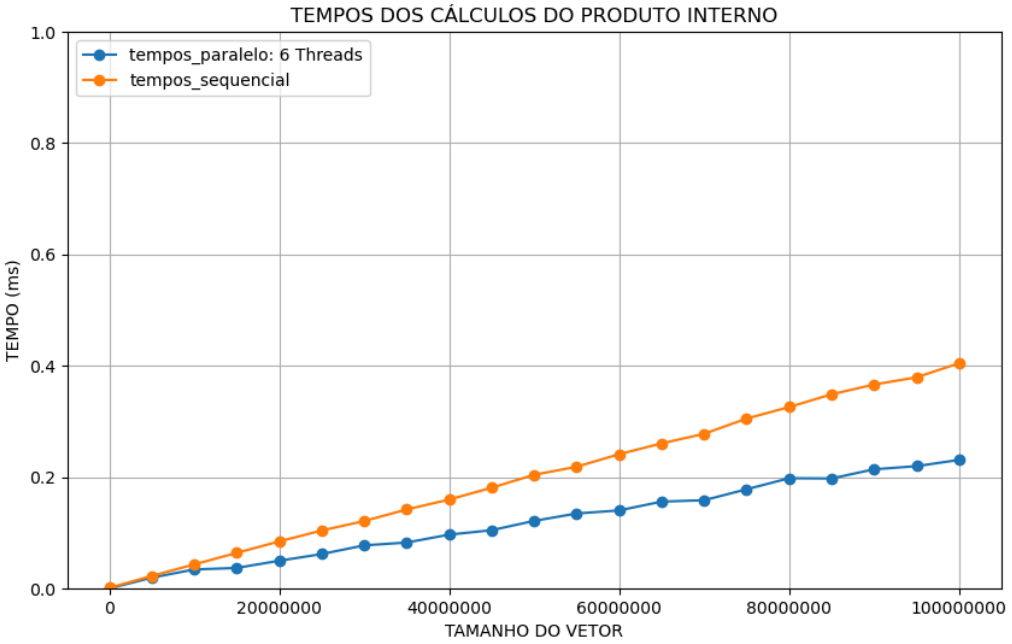
Comparação com 10 threads:



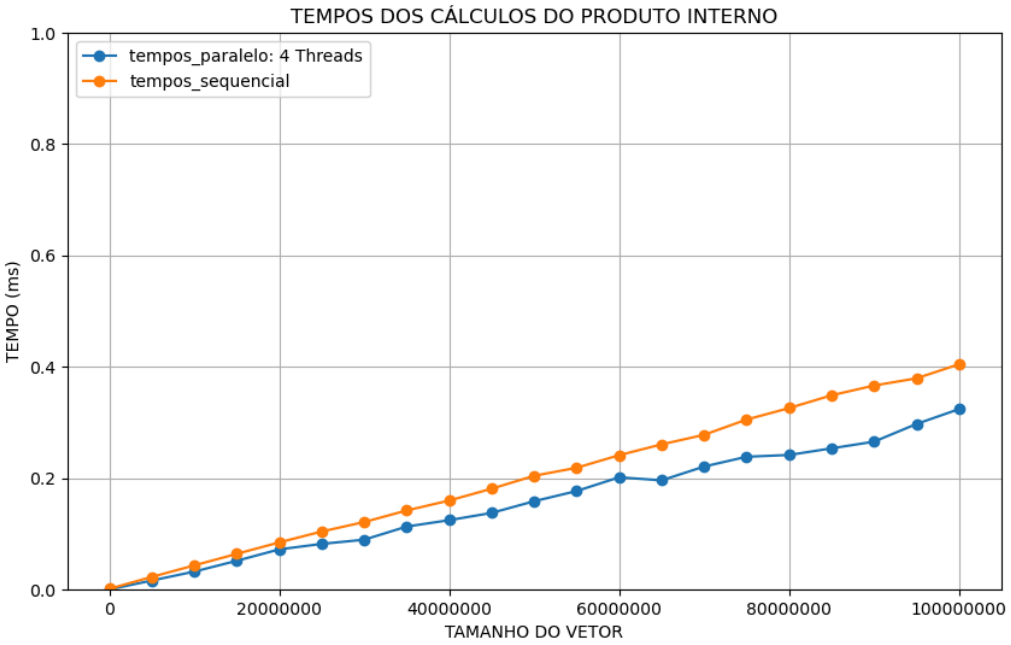
Comparação com 8 threads:



Comparação com 6 threads:



Comparação com 4 threads:



Comparação com 2 threads:

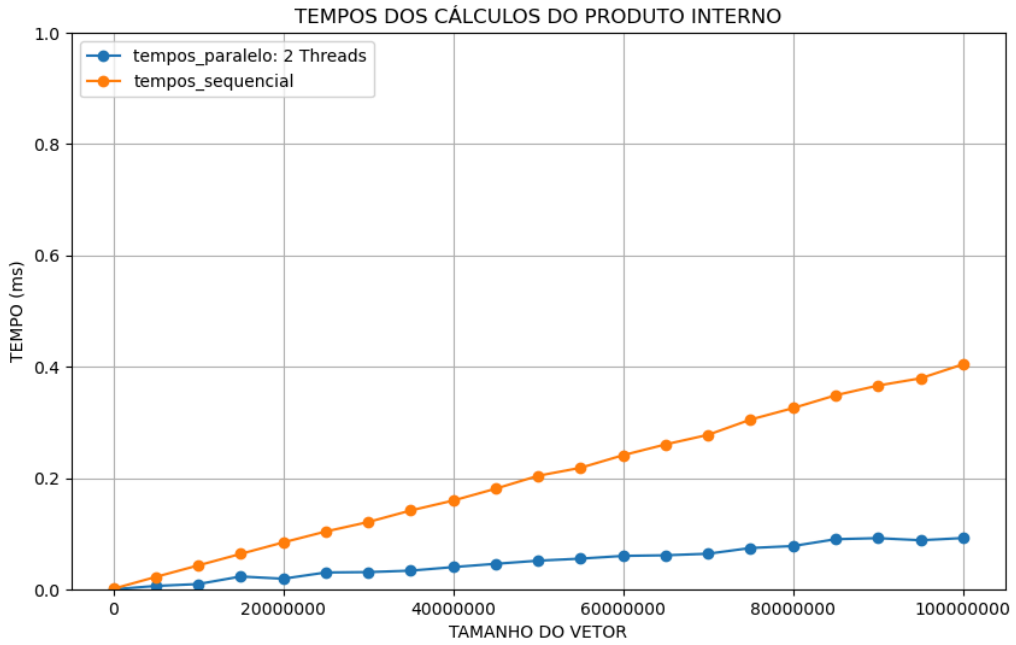


Tabela com os dados da versão sequencial:

Máquina com 12 CPUs:

tamanho_vet	resultado_PI	tempo(ms)
100000	300000	0.0005
5000000	15000000	0.0220
10000000	30000000	0.0436
15000000	45000000	0.0656
20000000	60000000	0.0872
25000000	75000000	0.1089
30000000	90000000	0.1310
35000000	105000000	0.1529
40000000	120000000	0.1745
45000000	135000000	0.1960
50000000	150000000	0.2177
55000000	165000000	0.2397
60000000	180000000	0.2614
65000000	195000000	0.2833
70000000	210000000	0.3051
75000000	225000000	0.3268
80000000	240000000	0.3485
85000000	255000000	0.3706
90000000	270000000	0.3925
95000000	285000000	0.4149
100000000	300000000	0.4358

Máquina com 16 CPUs:

tamanho do vetor	resultado_PI	tempo(ms)
100000	300000	0.0016
5000000	15000000	0.0225
10000000	30000000	0.0432
15000000	45000000	0.0639
20000000	60000000	0.0845

25000000	75000000	0.1041
30000000	90000000	0.1211
35000000	105000000	0.1419
40000000	120000000	0.1597
45000000	135000000	0.181
50000000	150000000	0.2042
55000000	165000000	0.2186
60000000	180000000	0.2412
65000000	195000000	0.2606
70000000	210000000	0.2778
75000000	225000000	0.3052
80000000	240000000	0.3258
85000000	255000000	0.3488
90000000	270000000	0.3663
95000000	285000000	0.3793
100000000	300000000	0.4045

Tabela com os dados da versão paralela:
Máquina com 8 CPUs:

qtd_threads	tamanho_vet	resultado_PI	tempo(ms)
2	100000	300000	0.0015
2	5000000	15000000	0.0471
2	10000000	30000000	0.0963
2	15000000	45000000	0.1446
2	20000000	60000000	0.1893
2	25000000	75000000	0.2396
2	30000000	90000000	0.2845
2	35000000	105000000	0.3313
2	40000000	120000000	0.3782
2	45000000	135000000	0.4308
2	50000000	150000000	0.4809
2	55000000	165000000	0.5293
2	60000000	180000000	0.5657
2	65000000	195000000	0.6228
2	70000000	210000000	0.6659

2	75000000	225000000	0.7133
2	80000000	240000000	0.7555
2	85000000	255000000	0.8252
2	90000000	270000000	0.8662
2	95000000	285000000	0.8984
2	100000000	300000000	0.9598
4	100000	300000	0.0006
4	5000000	15000000	0.0230
4	10000000	30000000	0.0441
4	15000000	45000000	0.0684
4	20000000	60000000	0.0908
4	25000000	75000000	0.1157
4	30000000	90000000	0.1409
4	35000000	105000000	0.1637
4	40000000	120000000	0.1888
4	45000000	135000000	0.2085
4	50000000	150000000	0.2257
4	55000000	165000000	0.2551
4	60000000	180000000	0.2743
4	65000000	195000000	0.2966
4	70000000	210000000	0.3248
4	75000000	225000000	0.3412
4	80000000	240000000	0.3721
4	85000000	255000000	0.3961
4	90000000	270000000	0.4156
4	95000000	285000000	0.4325
4	100000000	300000000	0.4491
6	100000	300000	0.0007
6	5000000	15000000	0.0198
6	10000000	30000000	0.0372
6	15000000	45000000	0.0595
6	20000000	60000000	0.0772
6	25000000	75000000	0.0973
6	30000000	90000000	0.1153

6	35000000	105000000	0.1393
6	40000000	120000000	0.1622
6	45000000	135000000	0.1794
6	50000000	150000000	0.1987
6	55000000	165000000	0.2264
6	60000000	180000000	0.2289
6	65000000	195000000	0.2695
6	70000000	210000000	0.2871
6	75000000	225000000	0.3003
6	80000000	240000000	0.3200
6	85000000	255000000	0.3444
6	90000000	270000000	0.3687
6	95000000	285000000	0.4002
6	100000000	300000000	0.4199
8	100000	300000	0.0005
8	5000000	15000000	0.0178
8	10000000	30000000	0.0349
8	15000000	45000000	0.0521
8	20000000	60000000	0.0689
8	25000000	75000000	0.0892
8	30000000	90000000	0.0981
8	35000000	105000000	0.1164
8	40000000	120000000	0.1334
8	45000000	135000000	0.1495
8	50000000	150000000	0.1597
8	55000000	165000000	0.1803
8	60000000	180000000	0.2029
8	65000000	195000000	0.2236
8	70000000	210000000	0.2459
8	75000000	225000000	0.2620
8	80000000	240000000	0.2906
8	85000000	255000000	0.3021
8	90000000	270000000	0.3159
8	95000000	285000000	0.3375

8	100000000	300000000	0.3418
10	100000	300000	0.0005
10	5000000	15000000	0.0131
10	10000000	30000000	0.0260
10	15000000	45000000	0.0343
10	20000000	60000000	0.0538
10	25000000	75000000	0.0662
10	30000000	90000000	0.0830
10	35000000	105000000	0.0909
10	40000000	120000000	0.1068
10	45000000	135000000	0.1168
10	50000000	150000000	0.1384
10	55000000	165000000	0.1454
10	60000000	180000000	0.1631
10	65000000	195000000	0.1715
10	70000000	210000000	0.1852
10	75000000	225000000	0.2041
10	80000000	240000000	0.1969
10	85000000	255000000	0.2284
10	90000000	270000000	0.2473
10	95000000	285000000	0.2529
10	100000000	300000000	0.2504

Máquina com 16 CPUs:

qtd_threads	tamanho_vet	resultado_PI	tempo(ms)
2	100000	300000	0.0004
2	5000000	15000000	0.0063
2	10000000	30000000	0.0097
2	15000000	45000000	0.0233
2	20000000	60000000	0.0193
2	25000000	75000000	0.0305
2	30000000	90000000	0.0312
2	35000000	105000000	0.0338

2	40000000	120000000	0.0402
2	45000000	135000000	0.0461
2	50000000	150000000	0.0517
2	55000000	165000000	0.0554
2	60000000	180000000	0.0603
2	65000000	195000000	0.0614
2	70000000	210000000	0.0642
2	75000000	225000000	0.0745
2	80000000	240000000	0.078
2	85000000	255000000	0.0903
2	90000000	270000000	0.0922
2	95000000	285000000	0.0884
2	100000000	300000000	0.0924
4	100000	300000	0.0003
4	5000000	15000000	0.016
4	10000000	30000000	0.0321
4	15000000	45000000	0.0515
4	20000000	60000000	0.072
4	25000000	75000000	0.0819
4	30000000	90000000	0.0894
4	35000000	105000000	0.113
4	40000000	120000000	0.1245
4	45000000	135000000	0.1375
4	50000000	150000000	0.1585
4	55000000	165000000	0.1768
4	60000000	180000000	0.2015
4	65000000	195000000	0.196
4	70000000	210000000	0.2208
4	75000000	225000000	0.2384
4	80000000	240000000	0.2416
4	85000000	255000000	0.2535
4	90000000	270000000	0.2655
4	95000000	285000000	0.2973
4	100000000	300000000	0.324
6	100000	300000	0.0004
6	5000000	15000000	0.0194
6	10000000	30000000	0.0341

6	15000000	45000000	0.0369
6	20000000	60000000	0.0498
6	25000000	75000000	0.0618
6	30000000	90000000	0.0774
6	35000000	105000000	0.0825
6	40000000	120000000	0.0967
6	45000000	135000000	0.1047
6	50000000	150000000	0.1214
6	55000000	165000000	0.1347
6	60000000	180000000	0.1402
6	65000000	195000000	0.156
6	70000000	210000000	0.1585
6	75000000	225000000	0.1782
6	80000000	240000000	0.198
6	85000000	255000000	0.1976
6	90000000	270000000	0.2141
6	95000000	285000000	0.2197
6	100000000	300000000	0.2309
8	100000	300000	0.0004
8	5000000	15000000	0.0073
8	10000000	30000000	0.0249
8	15000000	45000000	0.031
8	20000000	60000000	0.051
8	25000000	75000000	0.0608
8	30000000	90000000	0.0617
8	35000000	105000000	0.0953
8	40000000	120000000	0.0868
8	45000000	135000000	0.099
8	50000000	150000000	0.115
8	55000000	165000000	0.1295
8	60000000	180000000	0.1207
8	65000000	195000000	0.1397
8	70000000	210000000	0.1578
8	75000000	225000000	0.1282
8	80000000	240000000	0.1826
8	85000000	255000000	0.2007
8	90000000	270000000	0.2104

8	95000000	285000000	0.1618
8	100000000	300000000	0.2402
10	100000	300000	0.0005
10	5000000	15000000	0.0086
10	10000000	30000000	0.0179
10	15000000	45000000	0.0238
10	20000000	60000000	0.0487
10	25000000	75000000	0.0442
10	30000000	90000000	0.0573
10	35000000	105000000	0.0673
10	40000000	120000000	0.0792
10	45000000	135000000	0.0853
10	50000000	150000000	0.1067
10	55000000	165000000	0.1032
10	60000000	180000000	0.1079
10	65000000	195000000	0.1219
10	70000000	210000000	0.1466
10	75000000	225000000	0.1553
10	80000000	240000000	0.1874
10	85000000	255000000	0.214
10	90000000	270000000	0.1949
10	95000000	285000000	0.1696
10	100000000	300000000	0.1885

Análise da equipe sobre os impactos das quantidades diferentes de CPU:

A análise dos resultados do produto interno evidencia que o desempenho do paralelismo está diretamente relacionado tanto ao número de threads utilizadas quanto às características do hardware empregado nos testes. Para os experimentos, foram utilizados dois computadores distintos, um com 12 processadores lógicos (threads/CPUs) e outro com 16 processadores lógicos, sendo empregados os mesmos tamanhos de vetores e os mesmos valores dos elementos em ambos os ambientes, garantindo a comparabilidade dos resultados.

No computador com 12 processadores lógicos, observou-se que, ao utilizar apenas 2 threads, a versão sequencial apresentou desempenho superior à

versão paralela. Esse comportamento ocorre devido ao overhead associado à paralelização, como a criação e sincronização das threads, que não é compensado quando o grau de paralelismo é reduzido. Com 4 threads, os tempos de execução das versões sequencial e paralela mostraram-se equivalentes, caracterizando um ponto de equilíbrio entre o custo de gerenciamento das threads e o ganho obtido com a divisão do trabalho. A partir de mais de 4 threads, a versão paralela passou a apresentar melhor desempenho, indicando que o aumento do paralelismo e da carga computacional tornou o custo adicional da paralelização amortizado ao longo da execução.

Já no computador com 16 processadores lógicos, a versão paralela apresentou desempenho superior ao sequencial em todas as configurações de threads testadas. Esse resultado sugere que a maior disponibilidade de CPUs permite uma melhor distribuição das threads, reduzindo a contenção por recursos e possibilitando que o ganho obtido com a execução concorrente supere o overhead do paralelismo mesmo para números menores de threads.

Em ambos os ambientes, os testes foram realizados com vetores de tamanhos médios e grandes, variando de 10^5 a 10^8 elementos, o que permitiu observar claramente a influência do crescimento do vetor no desempenho. À medida que o tamanho do vetor aumenta, o custo computacional do cálculo do produto interno passa a dominar o tempo total de execução, favorecendo a versão paralela. Esse comportamento é consistente com o esperado para algoritmos paralelos de granularidade grossa, nos quais o paralelismo se torna mais eficiente conforme a quantidade de trabalho por thread aumenta.

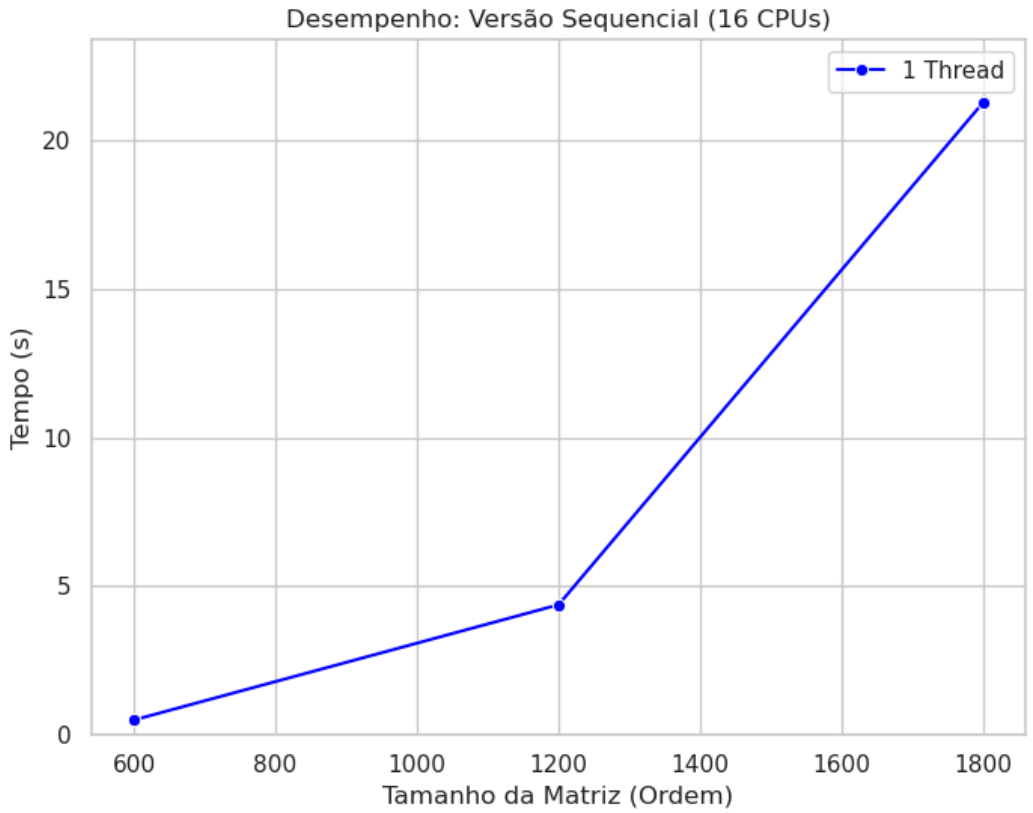
Por fim, assim como observado na multiplicação matricial, o desempenho do paralelismo depende do equilíbrio entre o número de threads criadas e a quantidade de CPUs disponíveis. Quando esse equilíbrio é respeitado, o algoritmo paralelo apresenta ganhos significativos de desempenho; caso contrário, o excesso de threads pode introduzir concorrência e reduzir a eficiência do processamento.

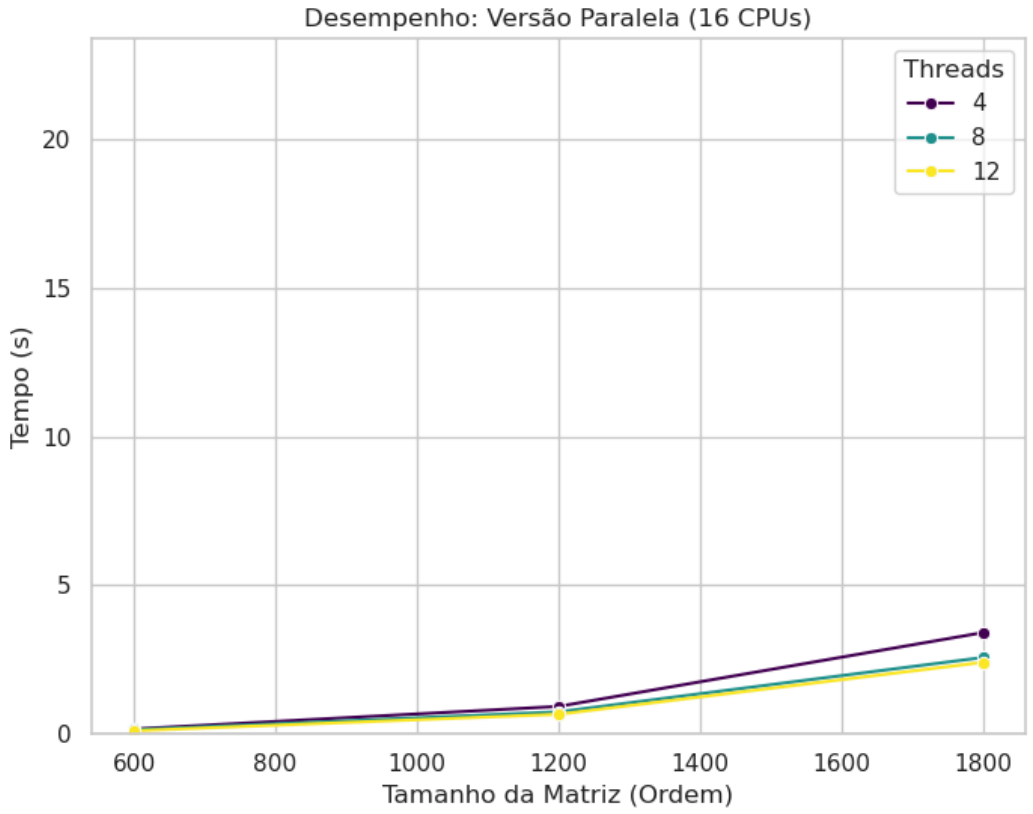
Questão 2 - Multiplicação de Matrizes

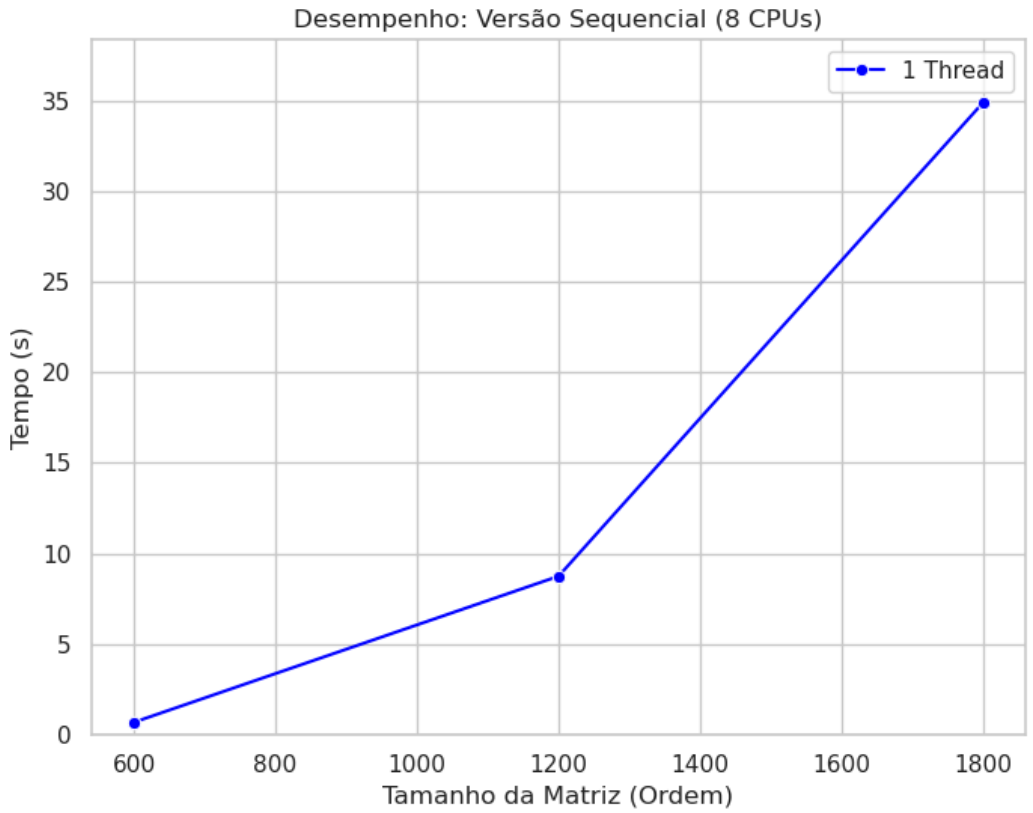
Método escolhido: A versão paralela utiliza o método de divisão por linhas. O loop externo da multiplicação, que percorre as linhas da matriz, foi distribuído entre as threads disponíveis. Cada thread calcula uma 'fatia' horizontal da matriz final independentemente. Se a divisão do número de

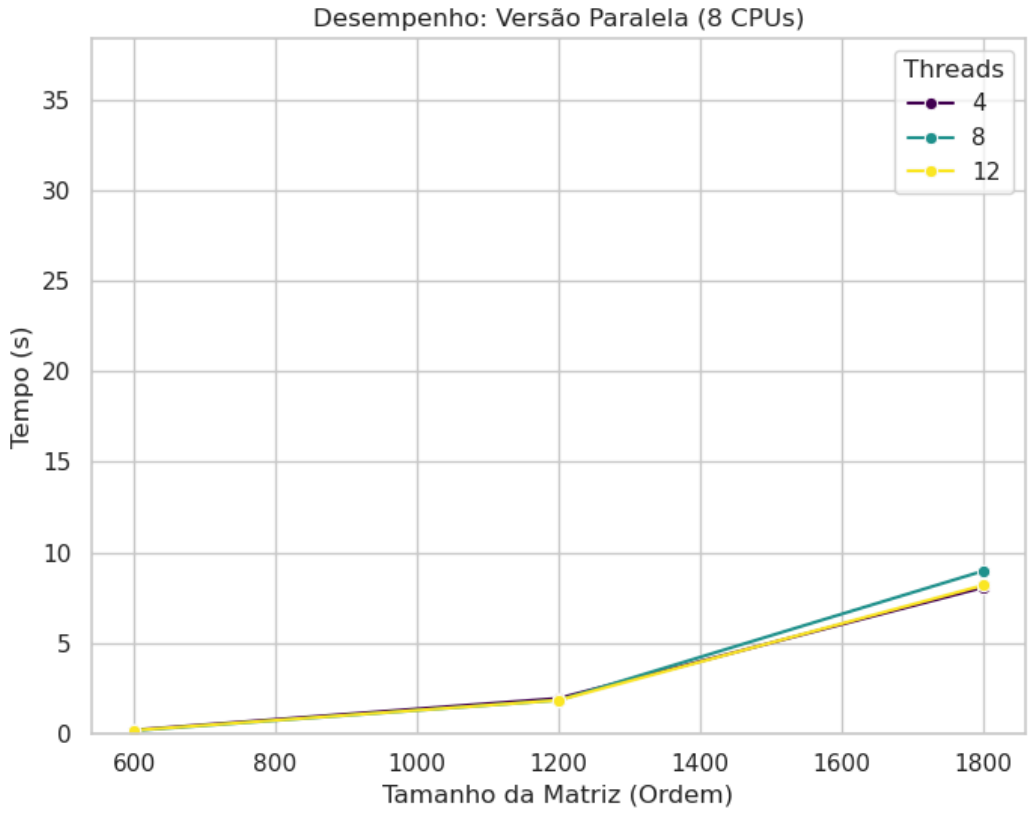
linhas pelo número de threads não for exata, a última thread assume o processamento das linhas restantes.

Gráficos gerados:


Desempenho: Versão Sequencial (16 CPUs)
1 Thread
Tempo (s)
0
5
10
15
20
600
800
1000
1200
1400
1600
1800
Tamanho da Matriz (Ordem)


Desempenho: Versão Paralela (16 CPUs)
Threads
4
8
12
Tempo (s)
0
5
10
15
20
600
800
1000
1200
1400
1600
1800
Tamanho da Matriz (Ordem)


Desempenho: Versão Sequencial (8 CPUs)
1 Thread
Tempo (s)
0
5
10
15
20
25
30
35
600
800
1000
1200
1400
1600
1800
Tamanho da Matriz (Ordem)


Desempenho: Versão Paralela (8 CPUs)
Threads
4
8
12
Tempo (s)
0
5
10
15
20
25
30
35
600
800
1000
1200
1400
1600
1800
Tamanho da Matriz (Ordem)

Análise da equipe sobre os impactos das quantidades diferentes de CPU:
Apesar do paralelismo apresentar desempenho superior ao sequencial em todos os casos testados, o seu desempenho começa a decair quando muitas threads são criadas. O provável é que, devido à quantidade finita de CPUs, conforme a quantidade de Threads vão aumentando elas vão causando mais concorrência entre si, o que faz com que algumas não consigam ser processadas pelas CPUs imediatamente, assim diminuindo a eficácia do algoritmo.

Resultados da máquina com 16 CPUs:

Ordem	Tempo sequencial (s)	Quantidade Threads	Tempo paralelo (s)	Speedup
600	0.495	4	0.158	3.132911392
600	0.495	8	0.1349	3.669384729
600	0.495	12	0.1007	4.915590864
1200	4.373	4	0.917	4.768811341
1200	4.373	8	0.7336	5.961014177
1200	4.373	12	0.6522	6.704998467
1800	21.2742	4	3.4025	6.252520206
1800	21.2742	8	2.5636	8.298564519
1800	21.2742	12	2.401	8.860558101

Resultados da máquina com 8 CPUs:

Ordem	Tempo sequencial (s)	Quantidade Threads	Tempo paralelo (s)	Speedup
600	0.686	4	0.206	3.330097087
600	0.686	8	0.17	4.035294118
600	0.686	12	0.187	3.668449198
1200	8.744	4	1.949	4.486403284
1200	8.744	8	1.845	4.739295393
1200	8.744	12	1.843	4.744438416
1800	34.877	4	8.065	4.324488531
1800	34.877	8	8.976	3.885583779

1800	34.877	12	8.185	4.261087355