ALGORITMO DE STRASSEN

IMPLEMENTAÇÃO PARALELA E ANÁLISE DE DESEMPENHO

Sobre o Projeto

O Objetivo do trabalho é multiplicar duas matrizes de grau $n = 2^k$, k= 1, 2, 3, 4, 5, 6, 7,...

Serão mostrados os algoritmos de Strassen implementados em formato sequencial, e paralelamente por meio do MPI e OpenMP.

Também se mostra um experimento onde serão comparados os tempos de CPU dos algoritmos com diferentes configurações de CPU a medida que k cresce.

O Algoritmo de Strassen

- Sejam A e B matrizes quadradas de ordem 2^n x 2^n, e seja C o produto dessas matrizes, para calcular esse produto particiona-se A, B e C em quatro submatrizes de mesmo tamanho

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

- Obtêm-se então as matrizes P_k da seguinte forma:

$$\begin{aligned}
\mathbf{P}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{P}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{P}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{P}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{P}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{P}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{P}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}$$

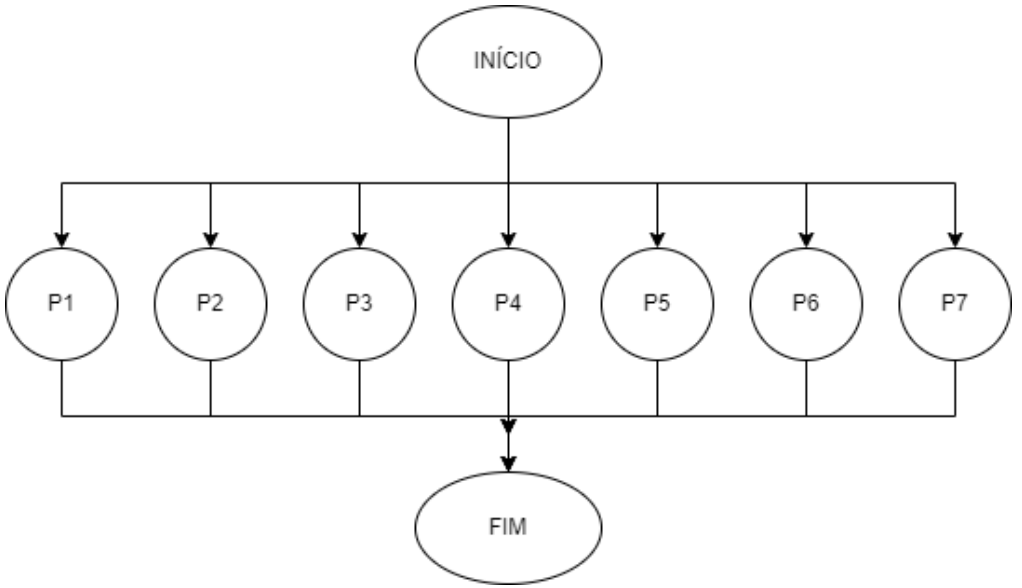
- Expressa-se $C_{i,j}$ em termos dos P_k.
- Devido a definição das matrizes P, pode-se eliminar uma multiplicação de matrizes e reduzir para 7 a sua quantidade (Uma multiplicação para cada P_k),
- expressando assim os $C_{i,j}$ como:

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{P}_1 + \mathbf{P}_4 - \mathbf{P}_5 + \mathbf{P}_7 \\
\mathbf{C}_{1,2} &= \mathbf{P}_3 + \mathbf{P}_5 \\
\mathbf{C}_{2,1} &= \mathbf{P}_2 + \mathbf{P}_4 \\
\mathbf{C}_{2,2} &= \mathbf{P}_1 - \mathbf{P}_2 + \mathbf{P}_3 + \mathbf{P}_6
\end{aligned}$$

- Ao todo, o Algoritmo de *Strassen* realiza 7 operações de multiplicação e 18 operações de soma/subtração, essas menos custosas;

Método Paralelo

- A ideia central do algoritmo consiste em dividir esses cálculos entre os processos disponíveis para acelerar a obtenção do resultado;
- A implementação serial de *Strassen* também foi usada no caso da implementação em MPI, em que temos uma função principal que vai se paralelizar e chamar recursivamente as funções seriais, assim sendo, cada um dos processos estará ocupado realizando as recursões para o cálculo final de uma das matrizes P.
- Graficamente falando esta é a divisão proposta:




Experimento

O Experimento

- Para o experimento foi utilizado uma máquina com a seguinte configuração:
  - Execução em ambiente Linux, por meio do WSL;
  - Processador Intel Core i7 9750h (6 núcleos/12 threads);
  - 32GB de Ram DDR4 (16Gb disponível via WSL);
  - SSD Nvme 500GB;

Comprovando Funcionamento

Para testar se o código estava multiplicando corretamente as matrizes, foram realizados testes com matrizes menores, onde as mesmas foram impressas e assim, pode-se comparar os resultados:

```
thiagofreitas@Xtreme-PH315:~/projetos/uff/labprogparalela/LPP-trabalho1-Strassen_Alan_Matheus_Thiago$ ./strassen-serial

Inserir dimensão n da matriz: 8

Imprimindo matriz A:
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2


Imprimindo matriz B:
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4


Tempo de execução do Strassen Sequencial: 5e-06


Imprimindo matriz C:
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
```

```
thiagofreitas@Xtreme-PH315:~/projetos/uff/labprogparalela/LPP-trabalho1-Strassen_Alan_Matheus_Thiago$ ./strassen-openmp

Insira o número de threads: 8

Insira a dimensão n da matriz: 8

Imprimindo matriz A:
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2


Imprimindo matriz B:
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4


Imprimindo matriz C:
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
```

```
thiagofreitas@Xtreme-PH315:~/projetos/uff/labprogparalela/LPP-trabalho1-Strassen_Alan_Matheus_Thiago$ mpiexec -n 8 --use-hwthread-cpus strassen-mpi

Inserir dimensão n da matriz: 8

Imprimindo matriz A:
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2


Imprimindo matriz B:
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4

Numero de processos: 8

Tempo de execução do Strassen Paralelo (MPI): 0.0002277


Imprimindo matriz C:
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
64 64 64 64 64 64 64 64
```

Validando Resultados

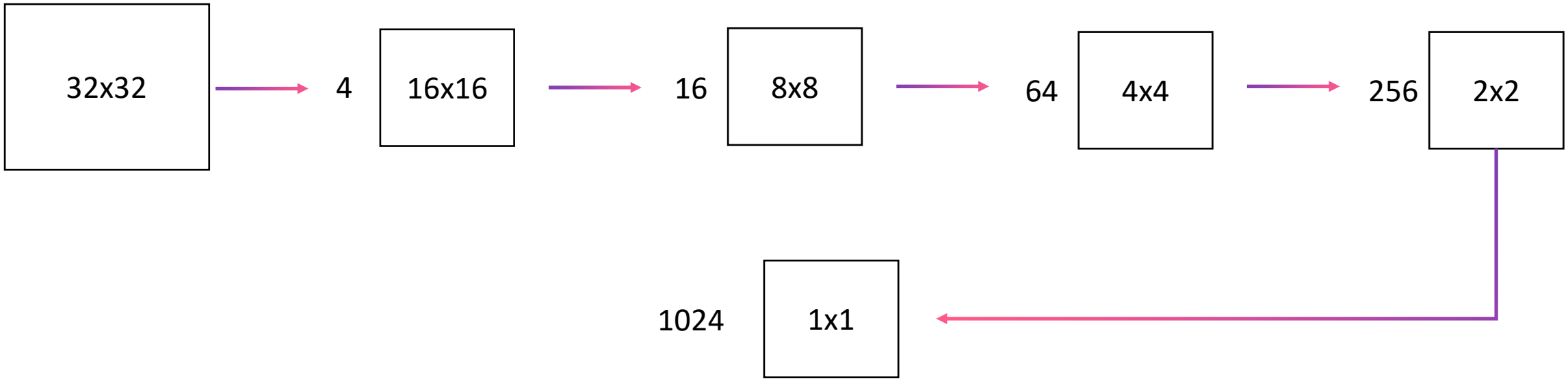
- Comparando ao obtido em site de multiplicação de matrizes:

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
1	64	64	64	64	64	64	64	64
2	64	64	64	64	64	64	64	64
3	64	64	64	64	64	64	64	64
4	64	64	64	64	64	64	64	64
5	64	64	64	64	64	64	64	64
6	64	64	64	64	64	64	64	64
7	64	64	64	64	64	64	64	64
8	64	64	64	64	64	64	64	64

Algumas Observações

- Strassen usa muito mais memória;
- Divisão em excesso do problema;

Problema no Algoritmo de Strassen



Solução: Redução do Corte

- Como o algoritmo direto se mostra mais eficiente para matrizes menores;
- Solução Reduzir o corte da recursão:

```
//Função strassen Serial
//entradas: tamanho n da matriz, ponteiro para matriz 1 e ponteiro para matriz 2
//saida: ponteiro para matriz produto da multiplicação
int** strassen(int n, int** mat1, int** mat2)
{

    //Condição de parada de recursão
    if (n <= 64)
    {
        return multiplica_matrizes(n, mat1, mat2);
    }
```

Execução

- O algoritmo foi executado para N = 128, 256, 512, 1024, 2048 e 4096;
- Em todos os casos a matriz A foi preenchida com 2 e a matriz B preenchida com 4.
- Cada algoritmo foi executado por pelo menos 5x para aproximar os resultados e desconsiderar valores ocasionalmente discrepantes;
- O teste foi realizado considerando 4, 6 e 8 processos/threads quando paralelo;
- Em todos os casos fez-se uso de hiperthreading;

Apresentação dos Algoritmos


Compilando e Executando

Resultado do Experimento

Para n=128

N = 128							
execução	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
1	0,0069	0,0063	0,0071	0,0064	0,0024	0,0035	0,0030
2	0,0078	0,0052	0,0065	0,0051	0,0034	0,0031	0,0029
3	0,0067	0,0047	0,0054	0,0055	0,0029	0,0031	0,0033
4	0,0085	0,0063	0,0053	0,0046	0,0026	0,0031	0,0030
5	0,0066	0,008	0,0058	0,0043	0,0028	0,0029	0,0034
MÉDIA	0,0073	0,0061	0,0060	0,0052	0,0028	0,0031	0,0031

Tabela 1 – Resultados para n=128

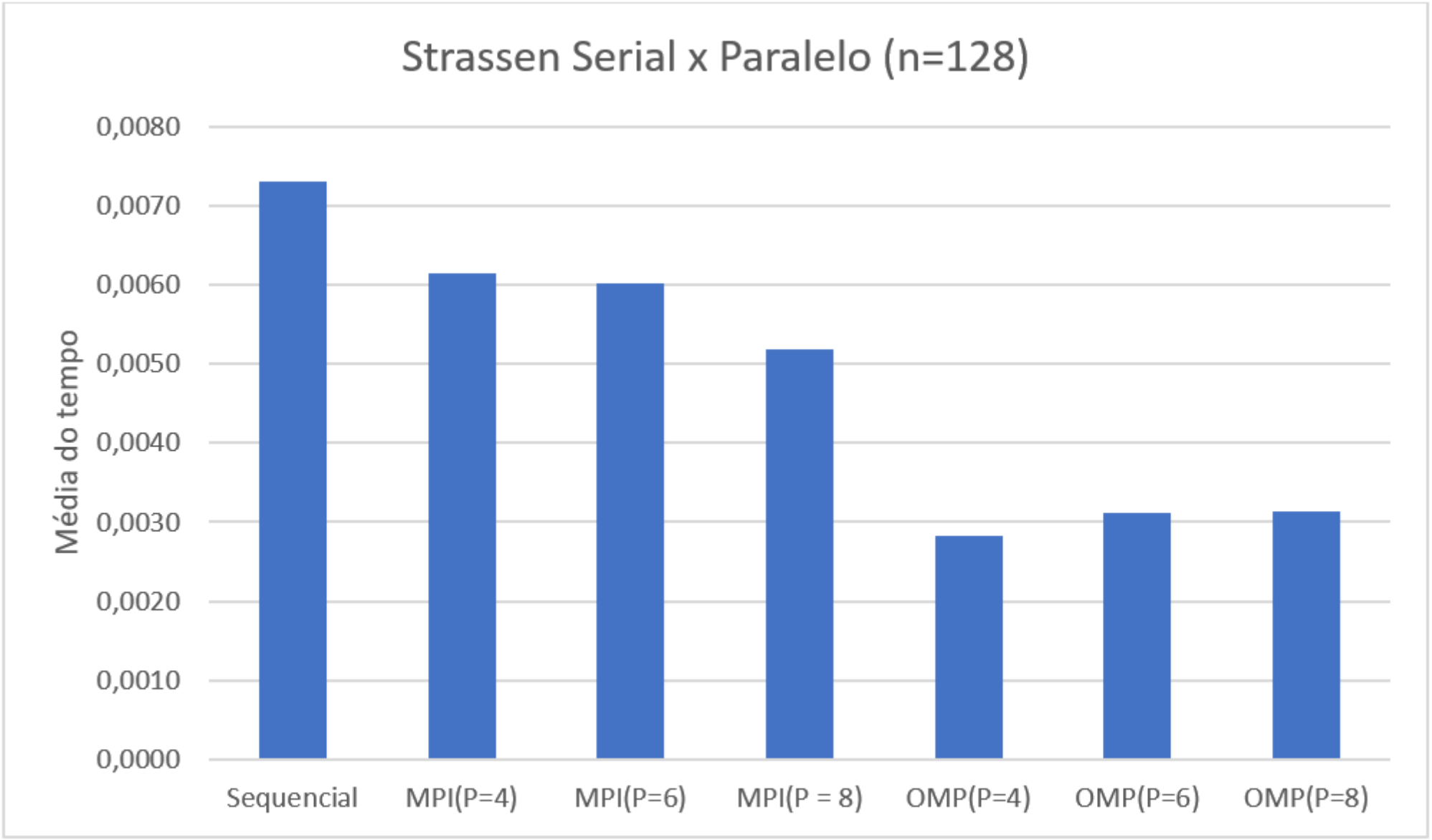


Gráfico 1 – Média de tempo obtido para multiplicação de matrizes com n=128

Para n=256

N = 256							
execução	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
1	0,0543	0,0287	0,0328	0,0329	0,0178	0,0187	0,0195
2	0,0527	0,0393	0,0335	0,0277	0,0221	0,0197	0,0192
3	0,0609	0,0307	0,0343	0,0204	0,0169	0,0193	0,0153
4	0,0500	0,0290	0,0233	0,0212	0,0207	0,0158	0,0161
5	0,0551	0,0387	0,0245	0,0201	0,0224	0,0251	0,0178
MEDIA	0,0546	0,0333	0,0297	0,0245	0,0200	0,0197	0,0176

Tabela 2 – Resultados para n=256

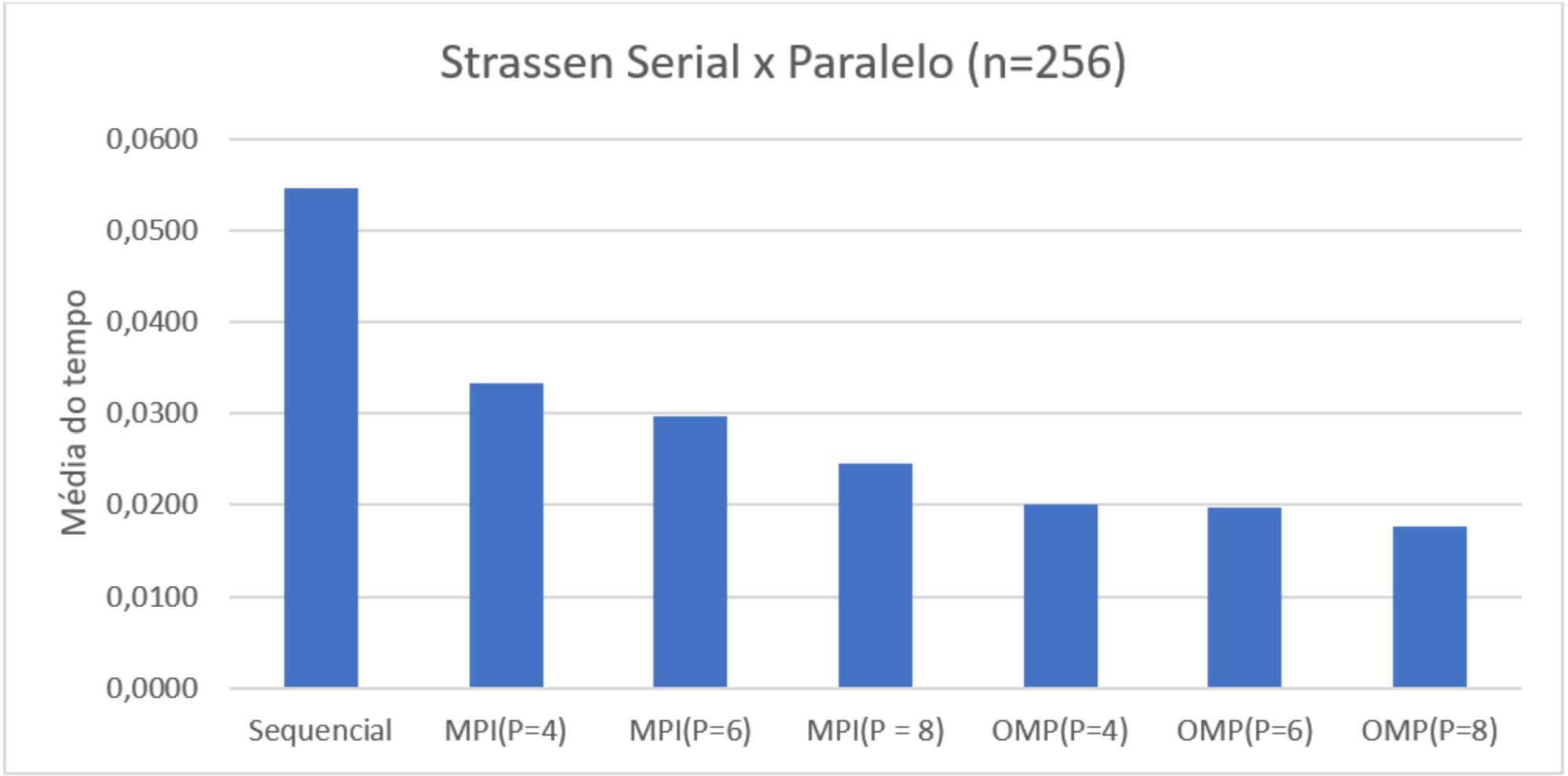


Gráfico 2 - Média de tempo obtido para multiplicação de matrizes com n=256

Para n=512

N = 512							
execução	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
1	0,3732	0,2222	0,2254	0,1413	0,1364	0,1275	0,1051
2	0,3641	0,2277	0,1888	0,1430	0,1552	0,1114	0,1106
3	0,3604	0,2506	0,1735	0,1243	0,1309	0,1165	0,1071
4	0,3807	0,2651	0,1723	0,1227	0,1447	0,1153	0,1127
5	0,3864	0,2323	0,1971	0,1211	0,1410	0,1232	0,1038
MÉDIA	0,3730	0,2396	0,1914	0,1305	0,1416	0,1188	0,1079

Tabela 3 – Resultados para n=512

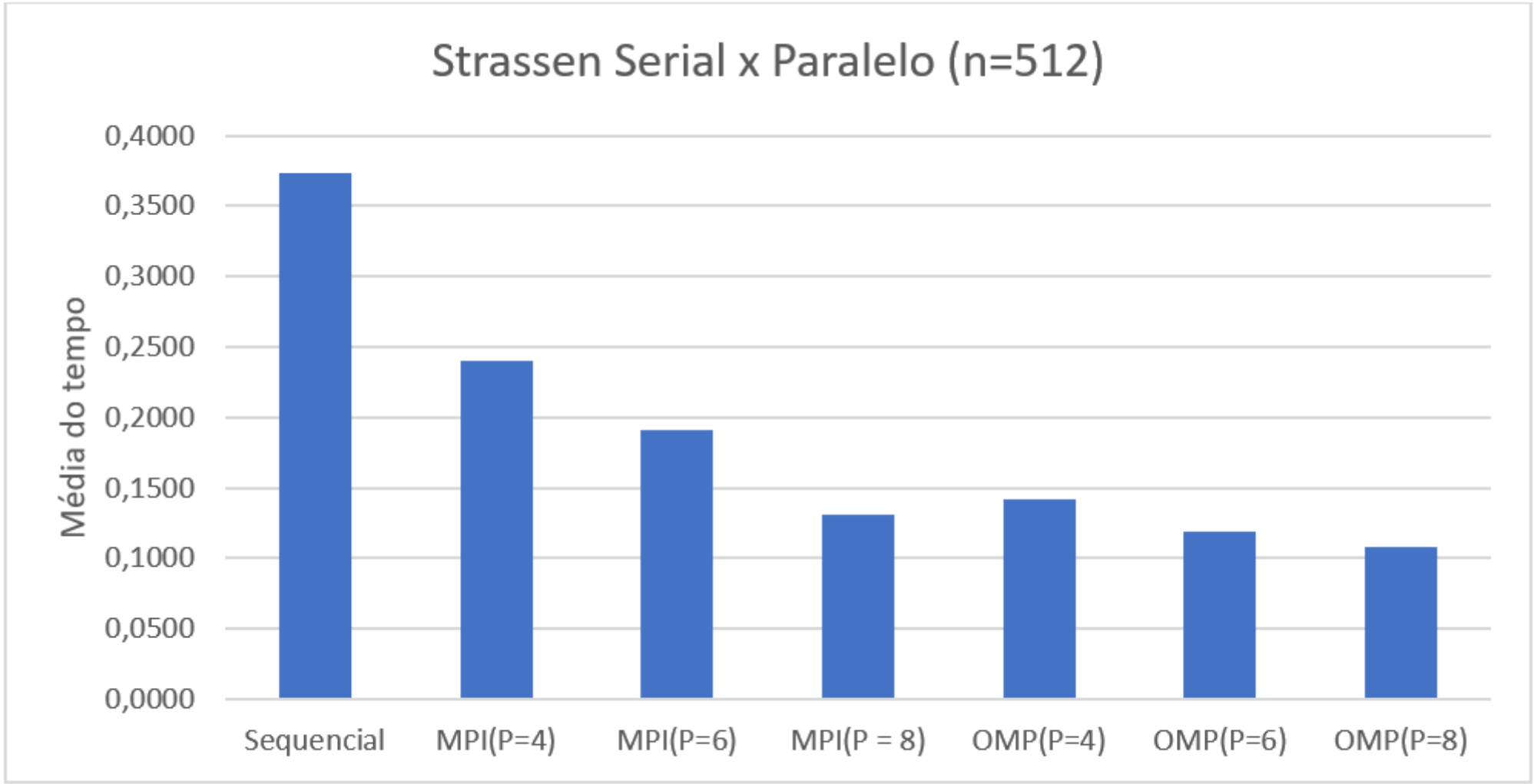


Gráfico 3 - Média de tempo obtido para multiplicação de matrizes com n=512

Para n=1024

N = 1024							
execução	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
1	2,6397	1,7836	1,3161	0,8118	1,0482	0,8310	0,7469
2	2,8666	1,5053	1,2755	0,8081	1,0018	0,8003	0,7441
3	2,5929	1,6139	1,2539	0,8555	1,0758	0,8624	0,7760
4	2,8459	1,6398	1,3547	0,8269	1,0515	0,7920	0,7554
5	2,7669	1,5994	1,2610	0,9007	1,0743	0,8130	0,7321
MÉDIA	2,7424	1,6284	1,2922	0,8406	1,0503	0,8198	0,7509

Tabela 4 – Resultados para n=1024

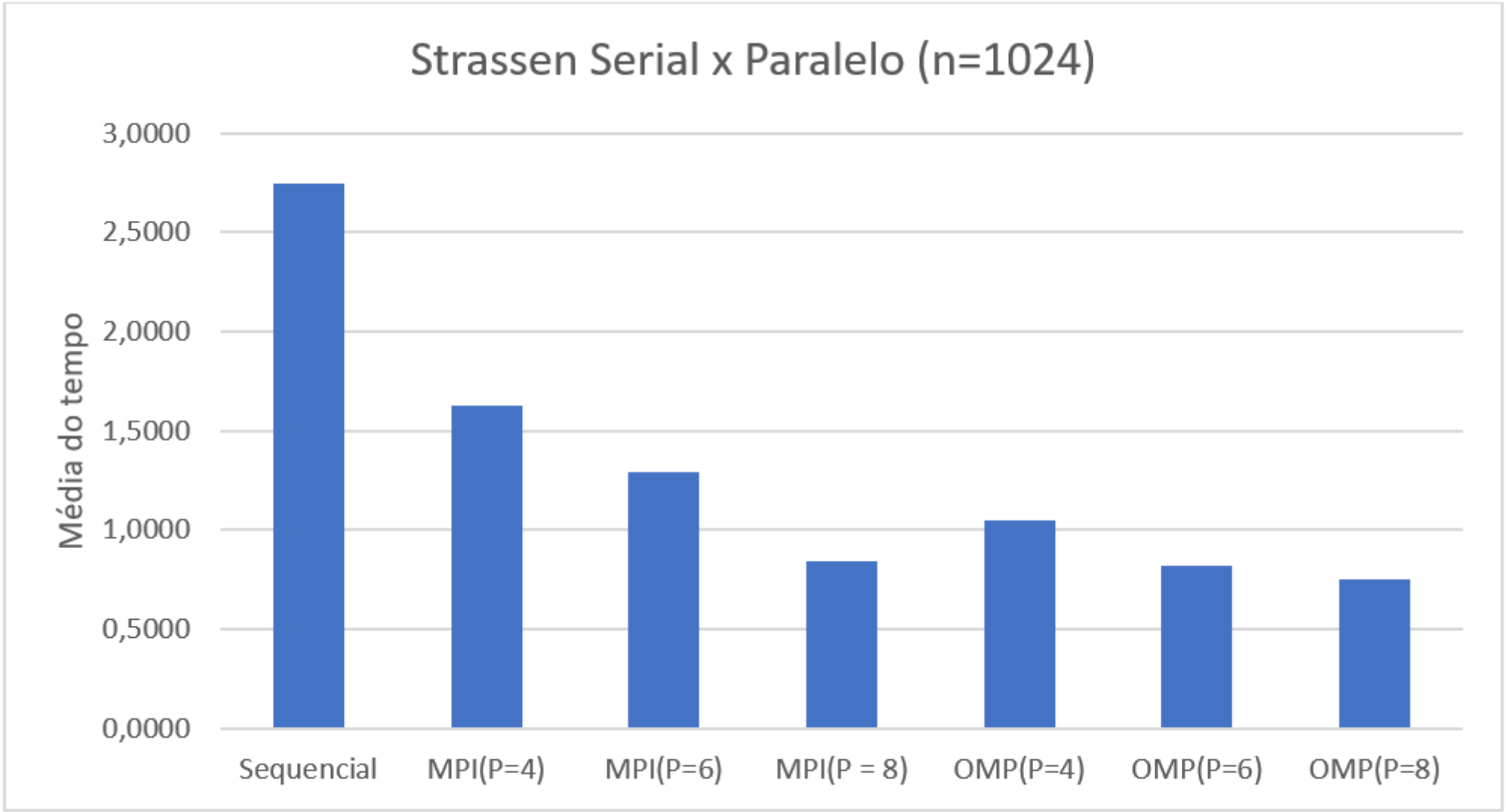


Gráfico 4 - Média de tempo obtido para multiplicação de matrizes com n=1024

Para n=2048

N = 2048							
execução	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
1	18,5100	11,1500	9,3013	6,0516	7,4392	5,7490	5,5318
2	19,0800	11,2470	9,1649	5,6135	7,8574	6,0082	5,2286
3	18,7770	11,6460	9,0820	5,7353	7,3557	5,9319	5,0991
4	18,8190	11,5010	8,9399	5,5879	7,3621	5,7073	5,1015
5	18,7740	11,3230	8,8398	5,5365	7,2726	5,7094	4,9395
MÉDIA	18,7920	11,3734	9,0656	5,7050	7,4574	5,8212	5,1801

Tabela 5 – Resultados para n=2048

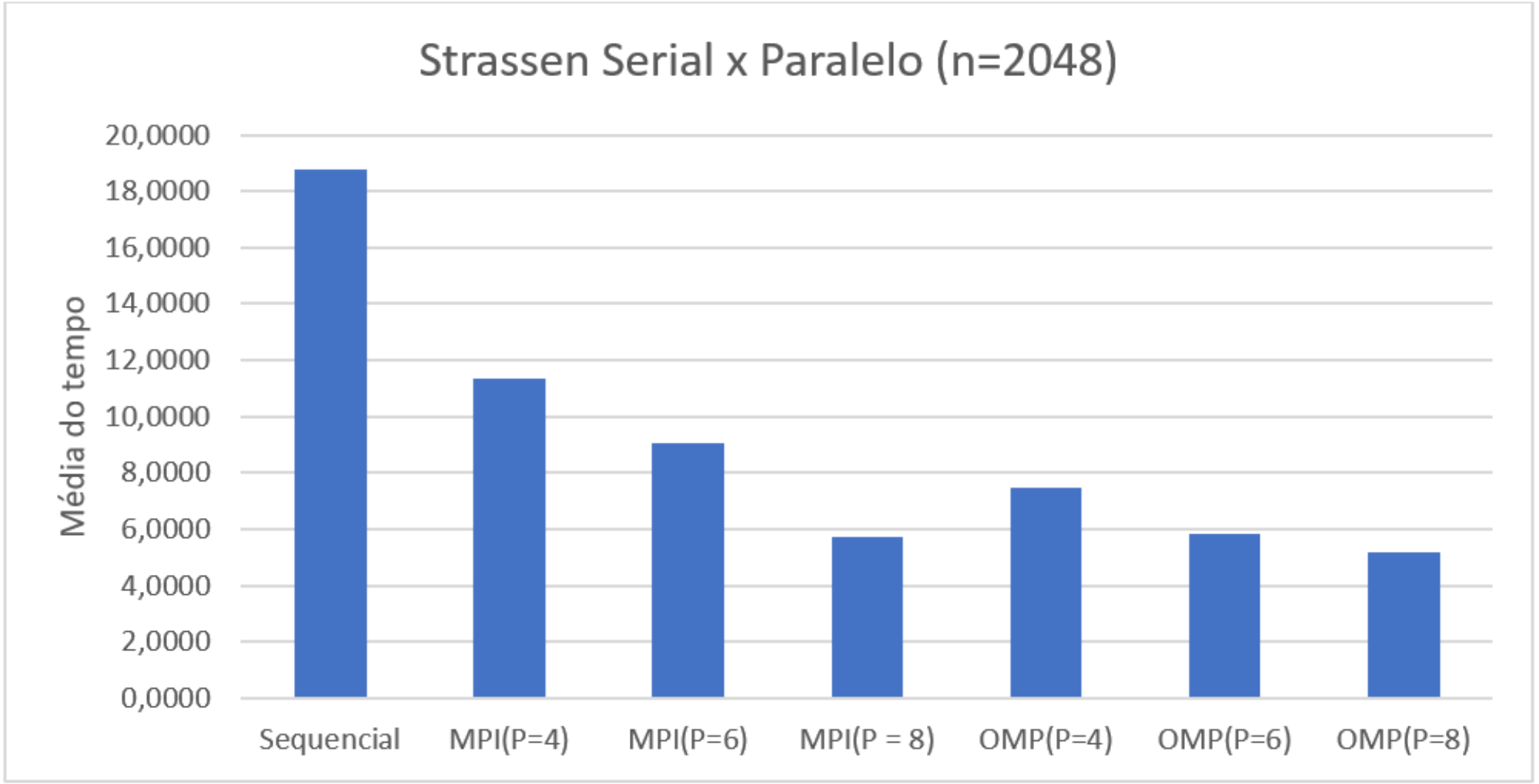


Gráfico 5 - Média de tempo obtido para multiplicação de matrizes com n=2048

Para n=4096

N = 4096							
execução	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
1	129,9800	79,2250	61,4720	38,4610	55,3200	42,7120	37,618
2	131,8300	80,9650	62,3120	38,9830	53,4170	41,8140	38,3400
3	131,5700	90,6070	63,2230	38,9100	50,7650	40,7090	37,3650
4	132,2700	79,9580	61,2810	38,9390	55,9900	41,9340	36,5800
5	131,5900	78,9390	61,0420	39,0180	51,0510	40,5540	36,4410
MEDIA	131,4480	81,9388	61,8660	38,8622	53,3086	41,5446	37,2688

Tabela 6 – Resultados para n=4096

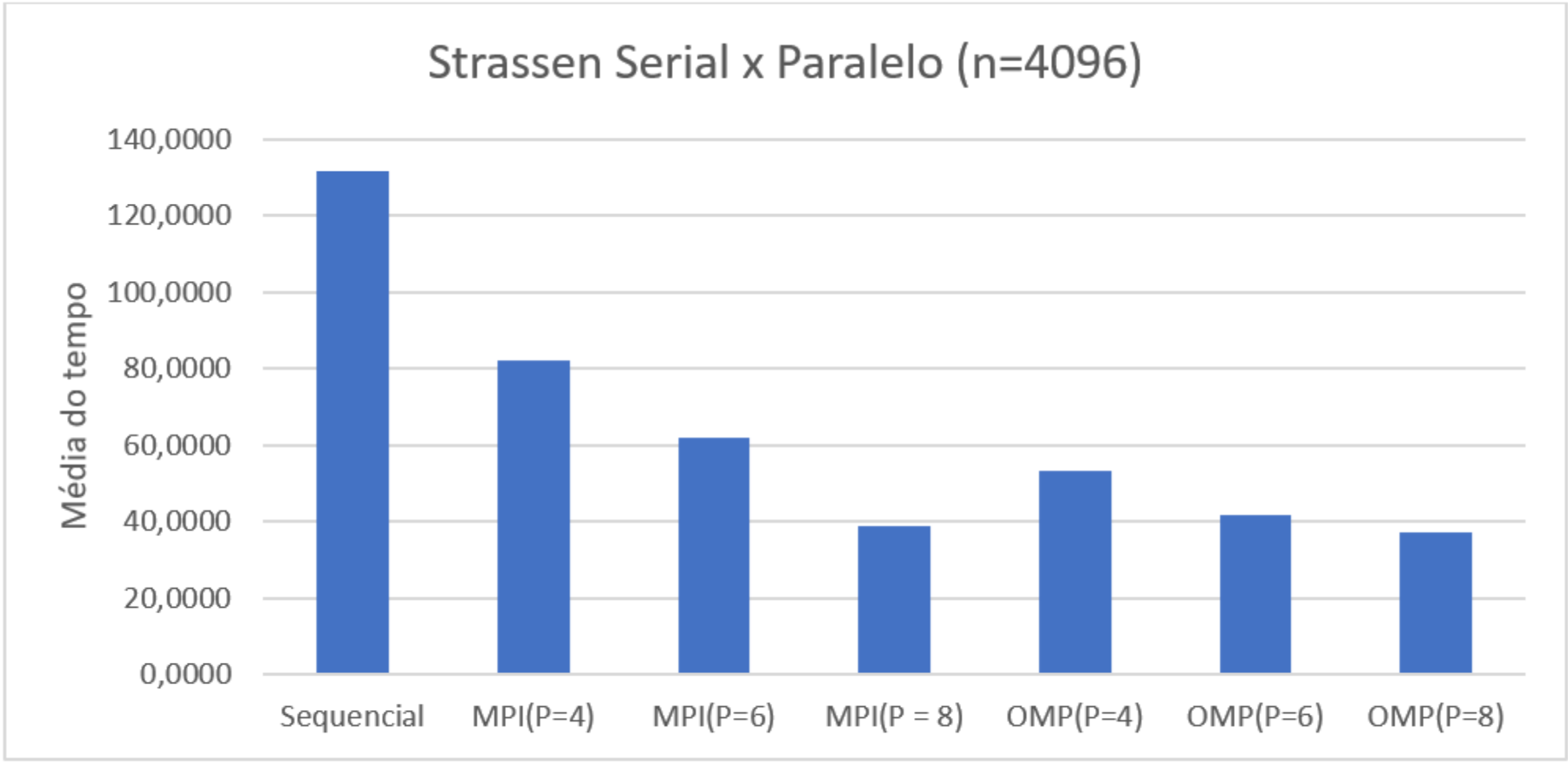
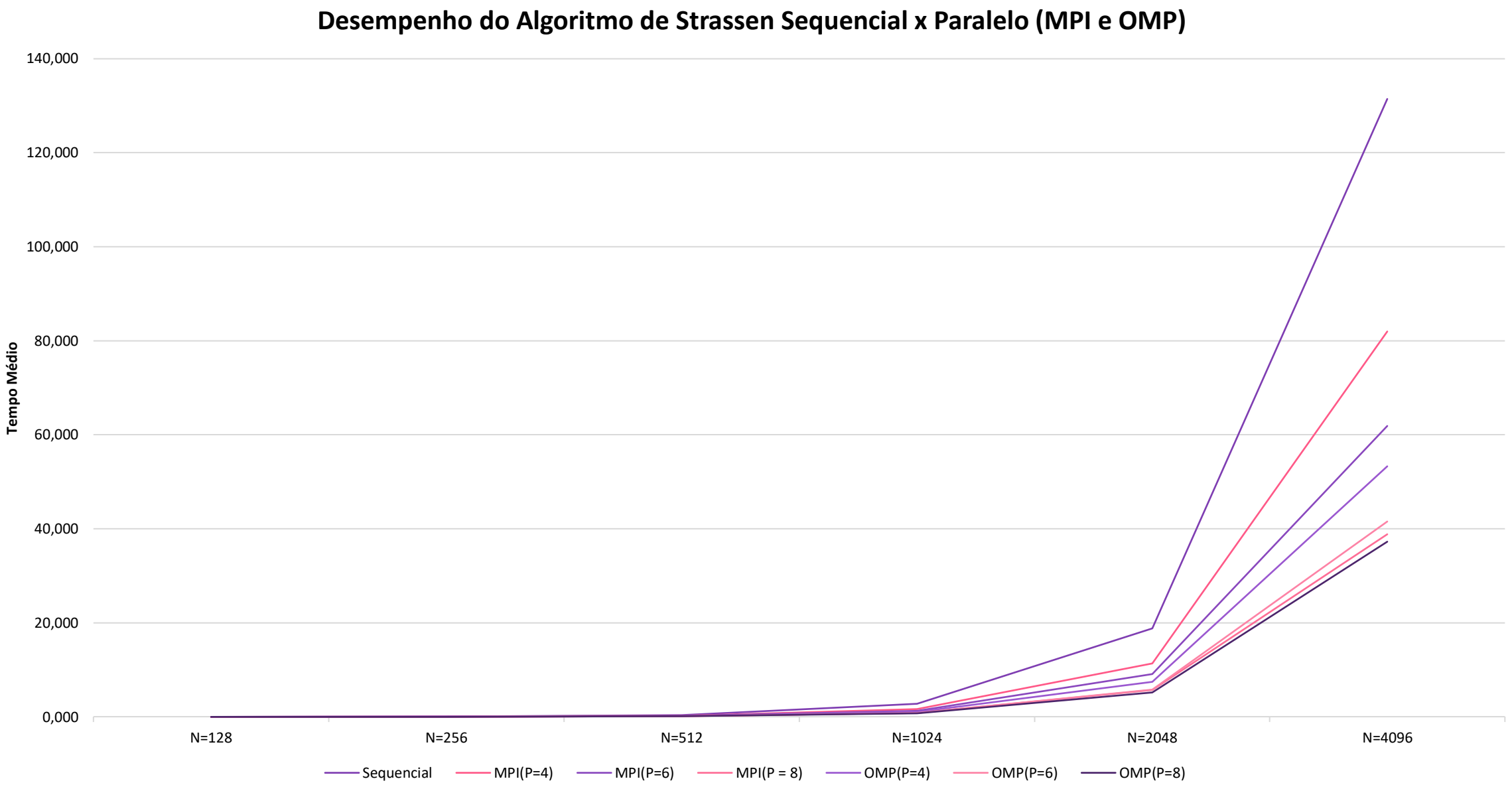


Gráfico 6 - Média de tempo obtido para multiplicação de matrizes com n=4096

Comparação Final

N	Sequencial	MPI(P=4)	MPI(P=6)	MPI(P = 8)	OMP(P=4)	OMP(P=6)	OMP(P=8)
N=128	0,007	0,006	0,006	0,005	0,003	0,003	0,003
N=256	0,055	0,033	0,030	0,024	0,020	0,020	0,018
N=512	0,373	0,240	0,191	0,130	0,142	0,119	0,108
N=1024	2,742	1,628	1,292	0,841	1,050	0,820	0,751
N=2048	18,792	11,373	9,066	5,705	7,457	5,821	5,180
N=4096	131,448	81,939	61,866	38,862	53,309	41,545	37,269

Tabela 7 – Comparativo das médias obtidas


Desempenho do Algoritmo de Strassen Sequencial x Paralelo (MPI e OMP)
Tempo Médio
140,000
120,000
100,000
80,000
60,000
40,000
20,000
0,000
N=128
N=256
N=512
N=1024
N=2048
N=4096
Sequencial
MPI(P=4)
MPI(P=6)
MPI(P = 8)
OMP(P=4)
OMP(P=6)
OMP(P=8)

Conclusão

Conclusão

- O Algoritmo de *Strassen* com certeza se mostra bem eficiente quando o tamanho da matriz é suficientemente grande;
- O algoritmo sequencial já mostra ganhos de desempenho quando comparado à solução direta, mas podemos ver que com o uso do paralelismo essa vantagem fica mais evidente;
- Para matrizes de tamanho 2048 e 4096 nota-se uma melhora de desempenho que chega na casa dos 70% quando executado com 8 processadores;

Conclusão

- Considera-se que a implementação apresentada mostra eficiência;
- Consumo razoável de memória;
- Tempos excelentes (ainda que um pouco maiores quando comparado a outras implementações que chegam a gastar mais que o triplo de memória);
- Desta forma, considera-se que as soluções apresentam equilíbrio entre o consumo e o tempo de execução.
- Ideias futuras para este algoritmo incluem: A possibilidade de execução por meio de clusters/rede, possibilitando maior quantidade de nós/processos disponíveis; A possibilidade de implementação considerando paralelização por meio de GPU.

Referências

Matrix computations (em inglês) - Golub, Gene Howard; Van Loan, Charles F. (1996). 3 ed. [S.l.]: JHU Press. pags. 31 - 33.

Divisão e Conquista – Prof. Maria Inés Castiñeira - Disponível em: https://slideplayer.com.br/slide/385710/ - Acesso em Junho de 2022

EXPERIMENTS WITH STRASSEN'S ALGORITHM: FROM SEQUENTIAL TO PARALLEL - Fengguang Song, Jack Dongarra, Shirley Moore - Disponível em: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.6510&rep=rep1&type=pdf – Acesso em Junho/2022

Matrix Multiplication Using Strassen's Algorithm With MPI – Mazarakis Periklis, Papadopoulous Aristeidis, Tsapekos Theodoros. Disponível em: https://github.com/aristosp/StrassenMPI_Project/blob/main/Report_english.pdf Acesso em junho/2022


Obrigado