Study of Time Taken By Different Matrix Multiplication Approach For Different value N of N Square Matrix

by

BITTU GOND

001611001013

NITYANANDA DEBSHARMA

001611001033

Index

- Introduction
- Applications of Matrix Multiplication
- Normal matrix multiplication
- Strassen's Algorithm
- Matrix multiplication using thread
- Matrix multiplication using mpi
- Input/output
- chart representations of data
- Reference

Introduction

- Matrix=It is 2d array with size m*n. Most of data are stored in the form of matrix.
- **Matrix multiplication:** The product of two matrix a and b (a*b) is Only possible if number of columns of
- First matrix ‘a’ is equal to number of rows of second matrix ‘b’. Let ‘a’ be a matrix of size m*n and ‘b’ be a matrix of size n*o then the product a*b will produce ‘c’ matrix of size m*o.
- Such that
- C[i][j]=a[i][1]*b[1][j]+a[i][2]*b[2][j]+.....+a[i][n]*b[n][j]

  For i=1, 2, 3....m and j=1, 2, 3.....o

Applications of Matrix Multiplication

- In graph most of the data is represented in the form of matrix. Matrix multiplication is one of the most important operation on matrix. Lots of matrix-graph based algorithms are based on matrix multiplication.

  egg Transitive closure

- It is used in network theory, solution of linear system of equation, transformation of coordinate system, statistics and linear algebra, geology, robotics, AI, Machine learning.

- It is used in cryptography .For example it is used in encryption and decryption in Hill Cipher and other techniques.

Normal matrix multiplication

- Function of normal matrix multiplication:-
- int odinaryMatrixMultiplication(int **a,int **b,int **c,int n)
- {
- int i,j,k;
- for(i=o;i<n;i++)
- for(j=o;j<n;j++)
- {
- c[i][j]=o;
- for(k=o;k<n;k++)
- {
- c[i][j] +=a[i][k] * b[k][j];
- }
- }
- return o;
- }

- In normal matrix multiplication we took 4 argument
- 1st a[n][n] matrix let say a[2][2]=

1	0
3	1

- 2nd b[n][n] matrix , b[2][2]=

2	1
0	1

- 3rd c[n][n] matrix to store result, c[2][2]
- 4th n as size, n=2

- In loop
- Case 1:i=0,j=0,k=0-1 then c[0][0]=a[0][0]*b[0][0]+a[0][1]*b[1][0]
- Case 1:i=0,j=1,k=0-1 then c[0][1]=a[0][0]*b[0][1]+a[0][1]*b[1][1]
- Case 1:i=1,j=0,k=0-1 then c[1][0]=a[1][0]*b[0][0]+a[1][1]*b[1][0]
- Case 1:i=1,j=1,k=0-1 then c[1][1]=a[1][0]*b[0][1]+a[1][1]*b[1][1]

- c=

2	1
6	4

- Total 8 addition took place for n =2. their were 3 loop each running n time all 3 were nested which shows time complexity of normal method =O(n^3) .

Strassen's Algorithm

Algorithm 3 Strassen's Matrix Multiplication Algorithm

Input: $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ and $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \in \mathbb{R}^{n \times n}$

1: **if** $n = 1$ **then**
2: $C = A \cdot B$
3: **else**
4: $M_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$
5: $M_2 = (A_{21} + A_{22}) \cdot B_{11}$
6: $M_3 = A_{11} \cdot (B_{12} - B_{22})$
7: $M_4 = A_{22} \cdot (B_{21} - B_{11})$
8: $M_5 = (A_{11} + A_{12}) \cdot B_{22}$
9: $M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$
10: $M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$
11: $C_{11} = M_1 + M_4 - M_5 + M_7$
12: $C_{12} = M_3 + M_5$
13: $C_{21} = M_2 + M_4$
14: $C_{22} = M_1 - M_2 + M_3 + M_6$

Output: $A \cdot B = C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \in \mathbb{R}^{n \times n}$

- So in each recursion of multiplication

1. if n=1 do normal multiply a[0][0]*b[0][0]
2. Else
3. Divide a and b assign a11,a12,a21,a22,b11,b12,b21,b22
4. Find m1,m2,m3,m4,m5,m6,m7 (n/2)^2 matrix
5. Then assign value in c11,c12,c21,c22 using formula from Strassen's method

- Time complexity=O(n^2.81).

Matrix multiplication using thread

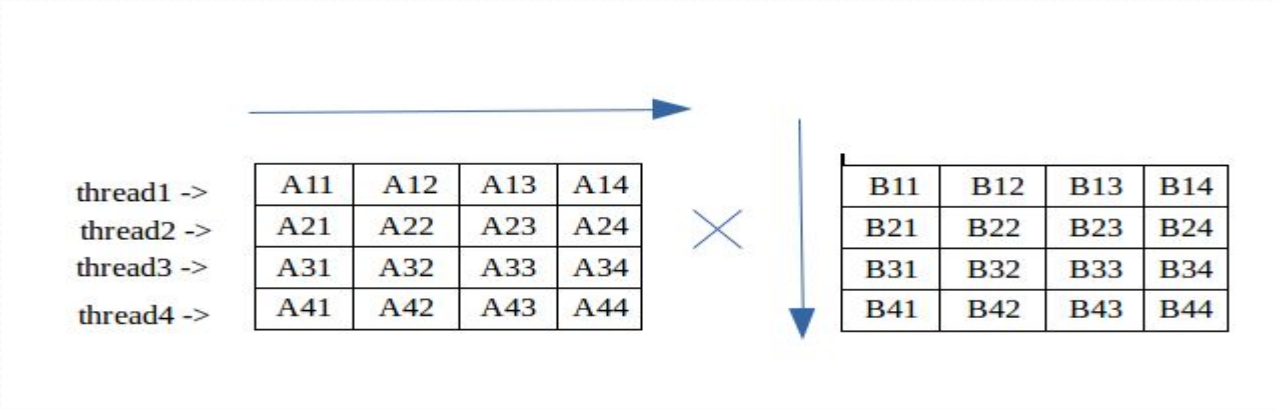
- Create 4 thread
- Let each thread multiply a[p][n]*b[n][n] where p in n/4.1st thread have p=0 to n/4 -1,

  2nd thread have p=0+n/4 to n/4 -1+n/4

  3rd thread have p=0+n/4+n/4 to n/4 -1+n/4+n/4

  3rd thread have p=0+n/4+n/4+n/4 to n-1

Since it works paralleled manner its execution time is better



Matrix multiplication using mpi

- It work similar to thread approach dividing 1st matrix with process other than 0 do normal multiplication on a[p]*[n]*b[n][n].where p =n/(np-1) n is size and np is count of process created after mpi_init.

Process 0 act as master and remaining process act as worker p(O) interchange data with other process by use of MPI_Send and MPI_Recv .

It works in parallel as well as in distributed environment

Input /output for normal matrix multiplication

```
addr:192.168.43.204>gcc matrixMultiplication1.c -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=2
matrix a:
 3 2
 3 1
matrix b:
 1 0
 0 2
done
time taken by odanary method of matrix multiplication=0.000003 sec
matrix c:
 3 4
 3 2
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=4
matrix a:
 3 2 3 1
 4 2 0 3
 0 2 1 2
 2 2 2 4
matrix b:
 1 0 0 2
 1 2 4 1
 1 1 3 4
 0 3 0 2
done
time taken by odanary method of matrix multiplication=0.000003 sec
matrix c:
 8 10 17 22
 6 13 8 16
 3 11 11 10
 6 18 14 22
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=16
done
time taken by odanary method of matrix multiplication=0.000134 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c  -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=256
done
time taken by odanary method of matrix multiplication=0.168110 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c  -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=512
done
time taken by odanary method of matrix multiplication=1.002248 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c  -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=1024
done
time taken by odanary method of matrix multiplication=15.214676 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c  -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=2048
done
time taken by odanary method of matrix multiplication=123.969043 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc matrixMultiplication1.c  -o 1
addr:192.168.43.204>./1
enter value of n for n^2 matrix a and b=4096
done
time taken by odanary method of matrix multiplication=842.484933 sec
addr:192.168.43.204>
```

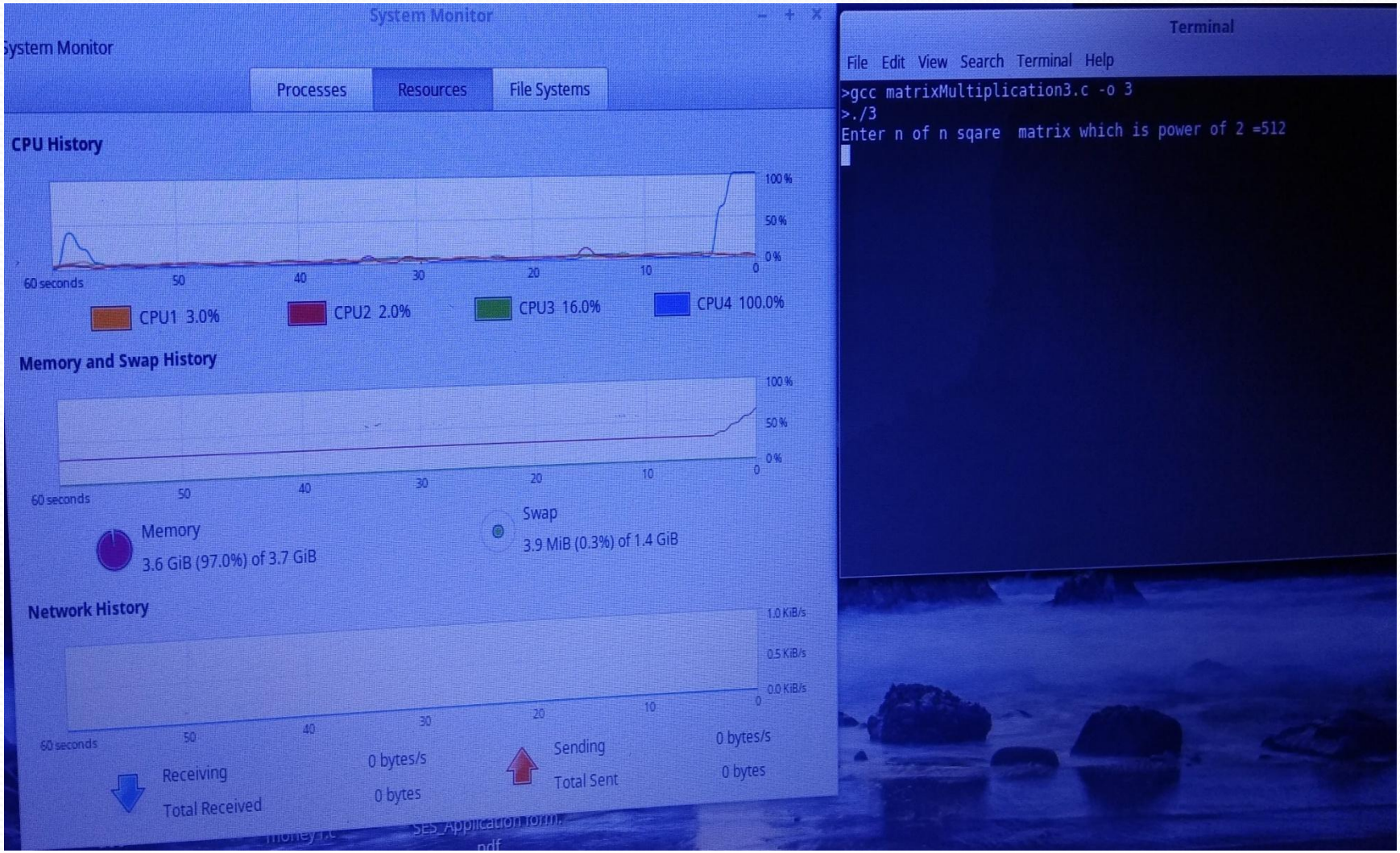
Input /output for Strassen's algorithm

```
>gcc matrixMultiplication3.c -o 3
>./3
Enter n of n sqare  matrix which is power of 2 =2
matrix a:
 3 2
 3 1
matrix b:
 1 0
 0 2
done
time taken by strassen's algorithm for matrix multiplication=0.000026 sec
matrix c:
 3 4
 3 2
>
```

```
>gcc matrixMultiplication3.c -o 3
>./3
Enter n of n sqare  matrix which is power of 2 =4
matrix a:
 3 2 3 1
 4 2 0 3
 0 2 1 2
 2 2 2 4
matrix b:
 1 0 0 2
 1 2 4 1
 1 1 3 4
 0 3 0 2
done
time taken by strassen's algorithm for matrix multiplication=0.000139 sec
matrix c:
 8 10 17 22
 6 13 8 16
 3 11 11 10
 6 18 14 22
>
```

```
>gcc matrixMultiplication3.c -o 3
>./3
Enter n of n sqare  matrix which is power of 2 =16
done
time taken by strassen's algorithm for matrix multiplication=0.004274 sec
>
```

```
>gcc matrixMultiplication3.c -o 3
>./3
Enter n of n sqare  matrix which is power of 2 =256
done
time taken by strassen's algorithm for matrix multiplication=2.584046 sec
>
```



Memory consumption became 100% system start hanging

Input/output of matrix multiplication using thread

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=2
matrix a:
 3 2
 3 1
matrix b:
 1 0
 0 2
done
time taken for  matrix multiplication using  n square thread =0.000857 sec
matrix c:
 3 4
 3 2
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=4
matrix a:
 3 2 3 1
 4 2 0 3
 0 2 1 2
 2 2 2 4
matrix b:
 1 0 0 2
 1 2 4 1
 1 1 3 4
 0 3 0 2
done
time taken for  matrix multiplication using  n square thread =0.000440 sec
matrix c:
 8 10 17 22
 6 13 8 16
 3 11 11 10
 6 18 14 22
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=16
done
time taken for  matrix multiplication using  n square thread =0.000618 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=256
done
time taken for  matrix multiplication using  n square thread =0.099006 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=512
done
time taken for  matrix multiplication using  n square thread =0.502232 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=1024
done
time taken for  matrix multiplication using  n square thread =5.958136 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=2048
done
time taken for  matrix multiplication using  n square thread =51.464284 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>gcc -pthread matrixMultiplication2a.c -o 2
addr:192.168.43.204>./2
enter value of n for n^2 matrix a and b=4096
done
time taken for  matrix multiplication using  n square thread =528.827028 sec
addr:192.168.43.204>
```

Input/output of matrix multiplication using mpi

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =2
matrix a:
3   2
3   1
matrix b:
1   0
0   2
resultant matrix:
3   4
3   2
time taken by matrix multiplication using mpi=0.000109 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =4
matrix a:
3   2   3   1
4   2   0   3
0   2   1   2
2   2   2   4
matrix b:
1   0   0   2
1   2   4   1
1   1   3   4
0   3   0   2
resultant matrix:
8   10   17   22
6   13   8   16
3   11   11   10
6   18   14   22
time taken by matrix multiplication using mpi=0.000063 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =16
time taken by matrix multiplication using mpi=0.000185 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =256
time taken by matrix multiplication using mpi=0.096881 sec
addr:192.168.43.204>
```

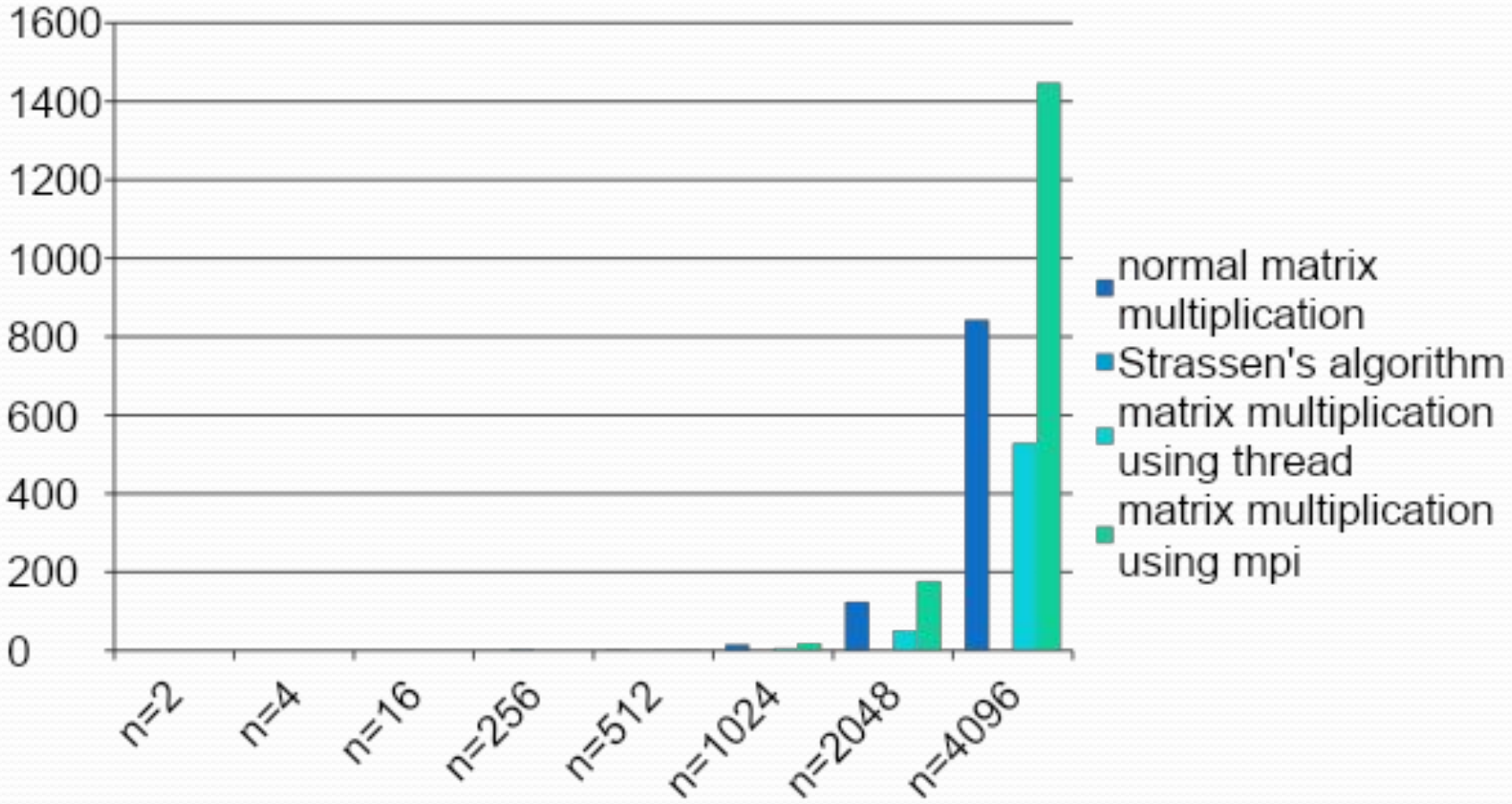
```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =512
time taken by matrix multiplication using mpi=0.918421 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =1024
time taken by matrix multiplication using mpi=18.204945 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =2048
time taken by matrix multiplication using mpi=176.621725 sec
addr:192.168.43.204>
```

```
addr:192.168.43.204>mpicc matrixMultiplication4.c -o 4
addr:192.168.43.204>mpirun 4
value of n =4096
time taken by matrix multiplication using mpi=1447.589081 sec
addr:192.168.43.204>
```

chart representations of data



Reference

- https://dl.acm.org/cms/attachment/6c24cb05-165b-4596-abec-21699eb1c282/ins03.gif/
- https://media.geeksforgeeks.org/wp-content/uploads/matmul.png/

Thank you