Group A

Assignment No: 6

MINI PROJECT

Title: Write a program to implement matrix multiplication. Also implement multithreaded matrix multiplication with either one thread per row or one thread per cell. Analyze and compare their performance.

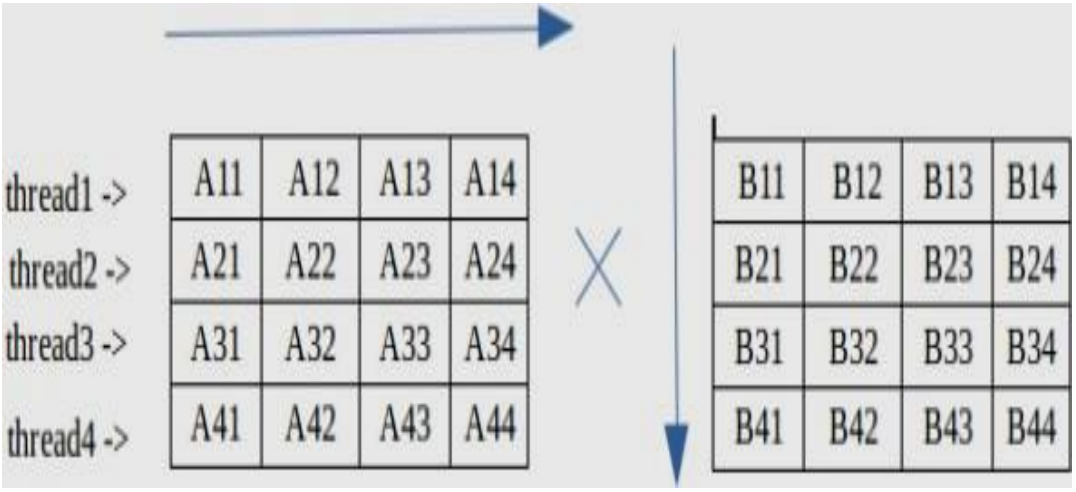
Multiplication of matrix does take time surely. Time complexity of matrix multiplication is O(n^3) using normal matrix multiplication. And Strassen algorithm improves it and its time complexity is O(n^(2.8074)).

But, Is there any way to improve the performance of matrix multiplication using the normal method.

Multi-threading can be done to improve it. In multi-threading, instead of utilizing a single core of your processor, we utilizes all or more core to solve the problem.

We create different threads, each thread evaluating some part of matrix multiplication.

Depending upon the number of cores your processor has, you can create the number of threads required. Although you can create as many threads as you need, a better way is to create each thread for one core. In second approach,we create a separate thread for each element in resultant matrix. Using pthread_exit() we return computed value from each thread which is collected by pthread_join(). This approach does not make use of any global variables



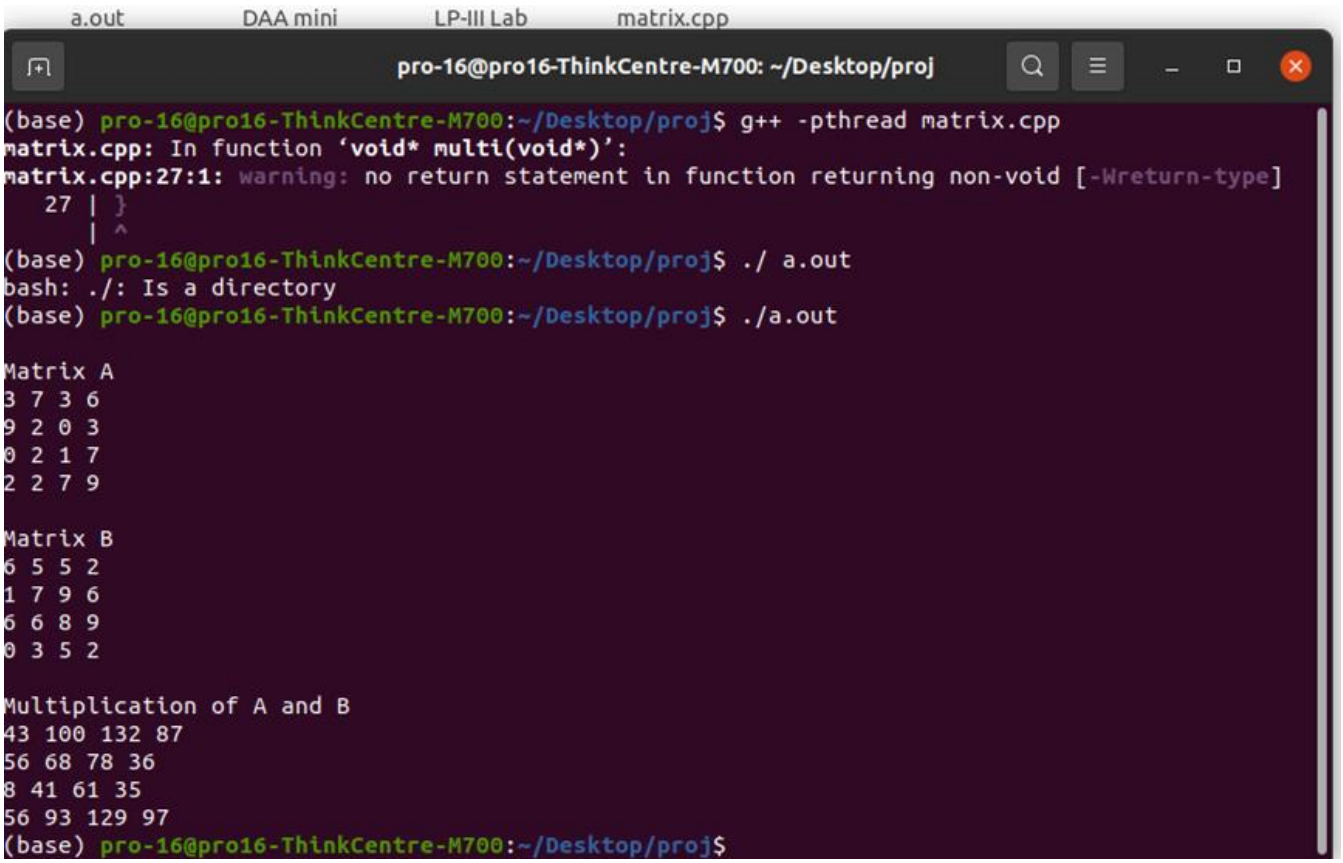
//CODE

Code :-

```
// CPP Program to multiply two matrix using  threads
#include <bits/stdc++.h>
using namespace std;

// maximum size of matrix
#define MAX 4

// maximum number of threads
#define MAX_THREAD 4

int matA[MAX][MAX];
int matB[MAX][MAX];
int matC[MAX][MAX];
int step_i = 0;

void* multi(void* arg)
{
	int i = step_i++; //i denotes row number of resultant matC

	for (int j = 0; j < MAX; j++)
	{
		for (int k = 0; k < MAX; k++)
		{
			matC[i][j] += matA[i][k] * matB[k][j];
		}
	}
}

// Driver Code
int main()
{
	// Generating random values in matA and matB
	for (int i = 0; i < MAX; i++) {
		for (int j = 0; j < MAX; j++) {
			matA[i][j] = rand() % 10;
			matB[i][j] = rand() % 10;
		}
	}

	// Displaying matA
	cout << endl
		<< "Matrix A" << endl;
	for (int i = 0; i < MAX; i++) {
		for (int j = 0; j < MAX; j++)
```

```
                cout << matA[i][j] << " ";
            cout << endl;
        }

        // Displaying matB
        cout << endl
            << "Matrix B" << endl;
        for (int i = 0; i < MAX; i++) {
            for (int j = 0; j < MAX; j++)
                cout << matB[i][j] << " ";
            cout << endl;
        }

        // declaring four threads
        pthread_t threads[MAX_THREAD];

        // Creating four threads, each evaluating its own part
        for (int i = 0; i < MAX_THREAD; i++) {
            int* p;
            pthread_create(&threads[i], NULL, multi, (void*)(p));
        }

        // joining and waiting for all threads to complete
        for (int i = 0; i < MAX_THREAD; i++)
            pthread_join(threads[i], NULL);

        // Displaying the result matrix
        cout << endl
            << "Multiplication of A and B" << endl;
        for (int i = 0; i < MAX; i++) {
            for (int j = 0; j < MAX; j++)
                cout << matC[i][j] << " ";
            cout << endl;
        }
        return 0;
}
```

Output:-


a.out
DAA mini
LP-III Lab
matrix.cpp
pro-16@pro16-ThinkCentre-M700: ~/Desktop/proj
(base) pro-16@pro16-ThinkCentre-M700:~/Desktop/proj$ g++ -pthread matrix.cpp
matrix.cpp: In function 'void* multi(void*)':
matrix.cpp:27:1: warning: no return statement in function returning non-void [-Wreturn-type]
27 | }
| ^
(base) pro-16@pro16-ThinkCentre-M700:~/Desktop/proj$ ./ a.out
bash: ./: Is a directory
(base) pro-16@pro16-ThinkCentre-M700:~/Desktop/proj$ ./a.out
Matrix A
3 7 3 6
9 2 0 3
0 2 1 7
2 2 7 9
Matrix B
6 5 5 2
1 7 9 6
6 6 8 9
0 3 5 2
Multiplication of A and B
43 100 132 87
56 68 78 36
8 41 61 35
56 93 129 97
(base) pro-16@pro16-ThinkCentre-M700:~/Desktop/proj$