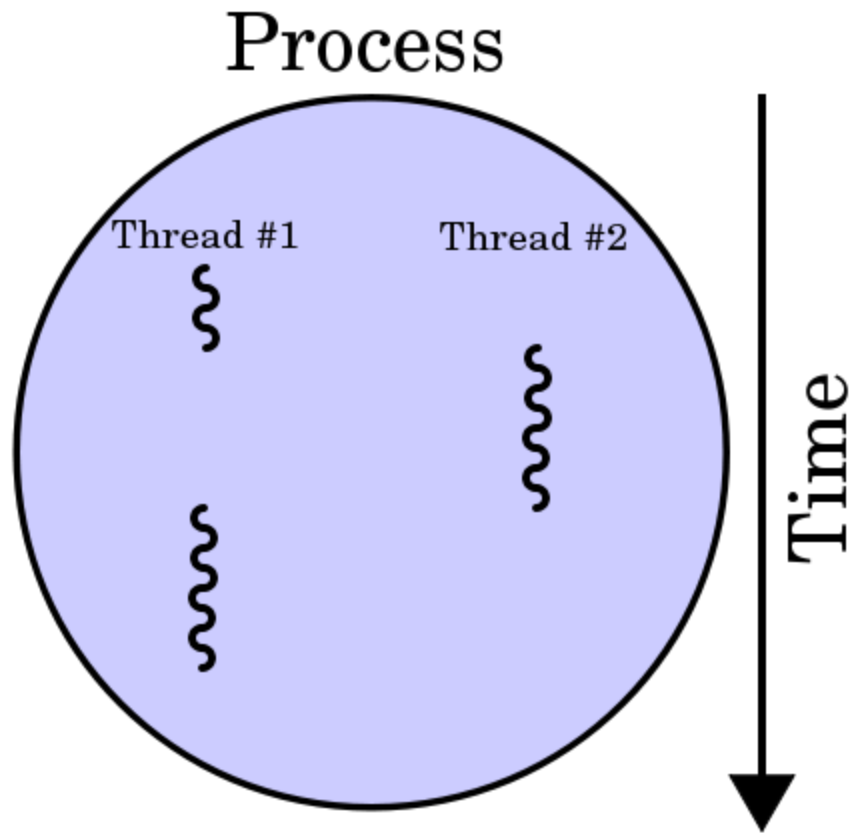


Threads Project

Name: Hisham Osama Abd Al-salam Ghazy

ID: 77



Introduction

In this project we are implementing a multithreaded matrix multiplication program. The input to the program is two matrices $A(x*y)$ and $B(y*z)$ that are read from corresponding files. The output is a matrix $C(x*z)$ that is written to an output file. A parallelized version of matrix multiplication will be done using 2 different methods. Details and comparison between the two methods will be shown.

Code Organization

The code is divided into 3 main folders:

1. Files

This folder is responsible for the file managing through the project where it is responsible for **setting** the **files** of the input and output matrices

It consists of 2 files:

- a. File Processing responsible for writing the output matrices to the standard format to files
- b. File Setup responsible for setting up the names of file of the_ of the input and output matrices

2. Matrices

This folder is considered a warm up step in the project flow where it is responsible for **creating** and **initializing** the input and output **matrices**

It consists of 2 files:

- a. Matrix Setup can be considered as a builder where it builds the input by allocating memory for them and filling their data from files and output matrices by allocating memory for them also but clearing their data from any previous garbage
- b. Utils containing helper methods in creating and initializing the input and output matrices

3. Executor

This folder is considered the **backbone** of the project where it contains the main functions implementing the two different methods of **matrix multiplication** using threads

Code Main Functions

For the main functions found in our project:

1. Files

- void **write_matrix**(char* path, int** matrix, int row, int column)
This function is responsible for writing the values of the matrix in its standard form to a file
- void **set_matrices_file_names**(char* file_names[])
This function is responsible for setting the file names of the input and output matrices

2. Matrices

- void **set_up_matrices**()
This function is working as a builder for building the input and output matrices
- void **set_metrics**()
This function is responsible for setting the rows and columns of the matrix
- void **set_matrix**(int** matrix)
This function is responsible for setting the values of the matrix
- void **set_matrix_dimensions**(int*** matrix, int row, int column)
This function is responsible for allocating memory for matrix
- void **clear_matrix**(int** matrix, int row, int column)
This function is responsible for clearing the contents of matrix

3. **Executers**

- void* **calculate_by_row**(void* tdata)

This function is responsible for calculating matrix multiplication row by row

- void* **calculate_by_element**(void* tdata)

This function is responsible for calculating matrix multiplication element by element

- void **method_1**()

This function is responsible for calling the method interested in calculating matrix multiplication row by row

- void **method_2**()

This function is responsible for calling the method interested in calculating matrix multiplication element by element

- void **run_method_1**()

This function is considered the interface for calling the method interested in calculating matrix multiplication row by row showing simple report for the user about execution

- void **run_method_2**()

This function is considered the interface for calling the method interested in calculating matrix multiplication element by element showing simple report for the user about execution

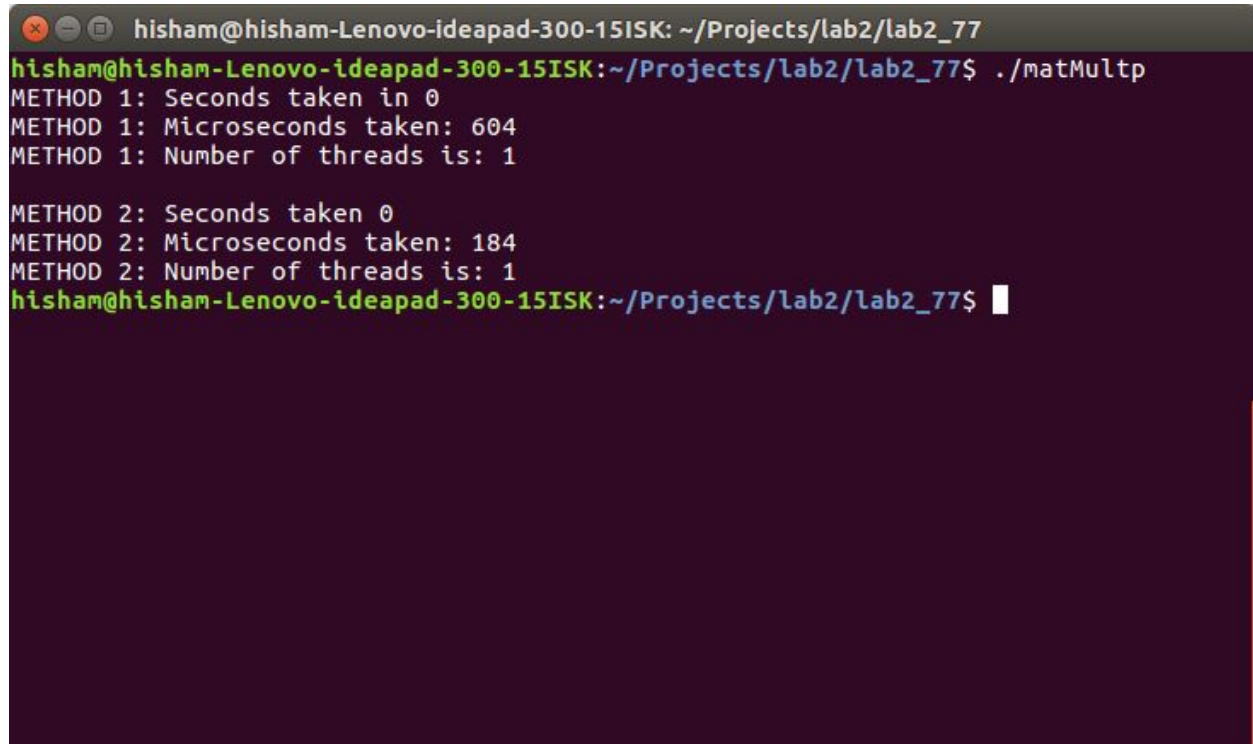
User Guide

To compile and run the code, follow the following steps :

1. Download the project to your machine
2. Open the terminal in the working directory of the project
3. Type “make”
 - This step will open the makefile supported with the project
 - The already found makefile will compile the .c files and link them together
 - It will and produce an executable file to run the program from, this file will be named “matMul”
4. Type “./matMul” to run the program using default file names
5. You can type “./matMul FILE_1 FILE_2 FILE_3” where FILE_1, FILE_2 and FILE_3 are the names of the first input matrix file, second input matrix file and the output matrix file respectively
6. Check the output folders where you can find the solution of the matrix multiplication using the two supported methods with the stdin showing a detailed report of the execution of the two methods

Sample Runs

Input Sizes 1x1, 5x5, 50x50 and 150x150 for both matrices will be used in the following test cases



```
hisham@hisham-Lenovo-ideapad-300-15ISK: ~/Projects/lab2/lab2_77
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$ ./matMultp
METHOD 1: Seconds taken in 0
METHOD 1: Microseconds taken: 604
METHOD 1: Number of threads is: 1

METHOD 2: Seconds taken 0
METHOD 2: Microseconds taken: 184
METHOD 2: Number of threads is: 1
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$
```

```
hisham@hisham-Lenovo-ideapad-300-15ISK: ~/Projects/lab2/lab2_77
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$ ./matMultp
METHOD 1: Seconds taken in 0
METHOD 1: Microseconds taken: 862
METHOD 1: Number of threads is: 5

METHOD 2: Seconds taken 0
METHOD 2: Microseconds taken: 2868
METHOD 2: Number of threads is: 25
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$
```

```
hisham@hisham-Lenovo-ideapad-300-15ISK: ~/Projects/lab2/lab2_77
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$ ./matMultp
METHOD 1: Seconds taken in 0
METHOD 1: Microseconds taken: 2091
METHOD 1: Number of threads is: 50

METHOD 2: Seconds taken 0
METHOD 2: Microseconds taken: 43141
METHOD 2: Number of threads is: 2500
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$
```

```
hisham@hisham-Lenovo-ideapad-300-15ISK: ~/Projects/lab2/lab2_77
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$ ./matMultp
METHOD 1: Seconds taken in 0
METHOD 1: Microseconds taken: 11829
METHOD 1: Number of threads is: 150

METHOD 2: Seconds taken 0
METHOD 2: Microseconds taken: 391347
METHOD 2: Number of threads is: 22500
hisham@hisham-Lenovo-ideapad-300-15ISK:~/Projects/lab2/lab2_77$
```

Comparison

It is clear from the previous test cases (and by repeating them for about 10 times to ensure that the function calculating the time is working in a good way) that:

- For small input size, the second method is found to be faster
- For large input size, the first method is found to be faster