

`

National University of Computer and Emerging Sciences

Department of Computer Science

**CS-326 : Parallel and Distributed Computing (3) BSCS- 5C Spring 2021**

**Matrix Multiplication using OpenMP in Docker environment**

**Project Report**

**Course Instructor**: Miss Nausheen Shoaib

**Group Members:**

* Ahmed Saqib (18K-0279)
* Shaheer Amir (18K-1154)
* Abeer Fatima (18K-0304)

**Date of Submission:** 22th May 2021

**Section**: 6C

**Objective:**

The objective of this project is to evaluate and compare the computation speed of matrix multiplication on different environments such as Serial, OpenMP and MPI in Dockers environment. Docker is an open platform for developing, shipping, and running applications.

**Project Description:**

* **Problem Statement:**

To analyze the computation power of processor by computing high dimensions of matrix multiplication.

* **Introduction:**

This project will do matrix multiplication on Dockers environment. Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. Docker provides the ability to package and run an application in a loosely isolated environment called a container. Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.

* **Methodology:**

We create different files for matrix multiplication i.e for serial, OpenMP and MPI.

Now in Ubuntu we first install Dockers by using **sudo apt-get install dockers.io**.

After installation we create a folder and we place the files in it, then we create a **Dockerfile.** A Dockerfile is simply a text file named as Dockerfile (without any extensions) with some commands and rules that Docker uses to create an image.

Setting up the Dockerfile we use **FROM** is used to define the base image to start the build process. Every Dockerfile must start with the FROM instruction. The idea behind this is that you need a starting point to build your image. **WORKDIR** tells Docker that the rest of the commands will be run in the context of the /app folder inside the image.**RUN** command runs within the container at build time. The main purpose of a CMD is to provide defaults when executing a container. These will be executed after the entry point. In Dockerfiles, you can define CMD defaults that include an executable.

After creating all the files we then run the command of

**sudo docker build . –t openmp**

This command build the image of process then we execute this image using command. Here –t is the tag that is given to the image. The **.(period)** symbol tells that the image should be created in the same directory.

We can view the created images by using the command **sudo dockers images.**

Then to run we use:

**sudo docker run –rm –it openmp**

Here –rm means if there exist any image with same tag and name remove it than run it in interactive mode. And we compare this multithreading execution with serial execution and mpi on dockers environment.

**Result (Comparison via Graphs)**

The above graphs shows the comparison between serial, OpenMP and MPI.

**Conclusion:**

In the beginning serial execution was quicker than multithreading but as the size of matrix increases the performance of serial execution decreases while multithreading performance is better. Hence for greater data and computation multithreading is better than serialize computation. Whereas comparatively OpenMP performs slightly better than MPI as the data increases. It can be seen that they are likely similar to one another on the same work load.