Evaluating Docker for Teaching and Learning MPI

By

Hamed Mohammed Alghamdi

201442664

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

October 2, 2020
ABSTRACT

With the increasing development of computer hardware, the performance of computers has increased by using multicores per computer. With the improvement of the multi-core computer design, the teaching of High-performance Computing has become fundamental for Computer Science students to teach them the concept of parallelism. Since the curriculum of teaching HPC does not including setting up a working environment, the teaching of HPC traditionally required access to HPC facility. Moreover, the teaching of HPC required environment reproducibility between teachers and students so that students can reproduce teachers’ environment from the class to run codes in their own time. In addition, students share their environment with teachers so that teachers can marking students’ work.

In this thesis, we evaluate the use of Docker containers to support the teaching pedagogy of HPC, especially the learning of MPI. The main aims are: (i) provide students with a container that has an MPI compiler and text editor (ii) evaluate Docker for teaching and learning MPI. We explain how we can initiate a Docker image and share it with students using Docker Hub repository; in addition, how students can build their own Docker image and share it with teachers. Furthermore, we evaluate Docker containers in terms of scalability of MPI code on local machine and on Azure Virtual Machine instances.
DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Hamed Mohammed Alghamdi
# TABLE OF CONTENT

1. Introduction ......................................................................................................................... 1  
   1.1 Environment Reproducibility ...................................................................................... 1  
   1.2 Docker for Environment Reproducing ................................................................. 1  
   1.3 Overview ..................................................................................................................... 2  
2. Background .......................................................................................................................... 3  
   2.1 Teaching of High-Performance Computing ............................................................ 3  
   2.2 Differences between Container and Virtual Machine ............................................ 3  
   2.3 Docker in Education .................................................................................................. 4  
3. Aims and Objectives .............................................................................................................. 5  
   3.1 Aims ............................................................................................................................. 5  
   3.2 objects .......................................................................................................................... 5  
4. Design and Realisation ........................................................................................................ 6  
   4.1 Docker image .............................................................................................................. 6  
   4.2 Use Case ....................................................................................................................... 6  
   4.2.1 teachers sharing their environment ..................................................................... 6  
   4.2.2 students sharing their environment ..................................................................... 10  
5. Evaluation .............................................................................................................................. 11  
   5.1 Size of the Image ........................................................................................................ 11  
   5.3 Scalability of Docker Container .............................................................................. 11  
6. Discussion .............................................................................................................................. 13  
   6.1 learning points .......................................................................................................... 13  
   6.2 professional issues ..................................................................................................... 13  
7. Conclusion ............................................................................................................................ 15  
   7.1 Summary ................................................................................................................... 15  
   7.2 Future Work ............................................................................................................... 15  
8. References ............................................................................................................................ 16  
9. Appendix .............................................................................................................................. 18  
   9.1 The MPI code was used to test the scalability ......................................................... 18  
   9.2 Dockerfile ..................................................................................................................... 20
1. Introduction

1.1 Environment Reproducibility

In this project, the main purpose is to find a solution for the environment reproducibility which is a common problem in teaching subjects such as Big Data and High-Performance Computing (HPC). Reproducing the environment can be useful in two aspects of the class. The first aspect is that the tutors sharing their environment with their students so that the students can reproduce examples in the class in their own time. Another aspect is that students share their environment with the tutor to marking the students’ work in the practical assessment. Students sharing their software environment with tutors will make marking assignments easier than marking the assignment on a paper report.

The concept of containerization[1], which is used in this project, provides developers and researchers the ability to share their environment and run it in a different environment[2][3]. Containerization is a lightweight application that comprises all its dependencies and libraries that are required for a container to run in a different computing environment. The most popular tool used to build and run containers is Docker[4].

1.2 Docker for Environment Reproducing

Docker is a tool to create, maintain, deploy, and run applications by using containers technology. Hence, Docker provides the ability to package up the application with its all required. Therefore, the application runs inside an isolated container. The isolation of the container allows the users to run many containers at the same time on the same host. The following diagram shows a simple workflow diagram to develop Docker images.
This project aims to provide an initial environment to the students on which they can run it on their laptops, university facilities, and cloud instances. The environment is built by using Docker which will be used to support teaching and learning HPC, especially to support the teaching of the parallel programming paradigm MPI[5].

The initial environment is based on the Ubuntu operating system on which the MPICH[6] compiler is installed along with the Nano Editor. I have tested the initiated environment in different local computing environment. Besides, an Azure Virtual Machine has been used to test the environment on a cloud instance.

1.3 Overview

The next chapter of this dissertation, chapter 2, will be background reading on the necessary information to carry out the project. Then, in chapter 3, the list of hypotheses besides the aims and the objectivise of the project. In chapter 4, description of the design of the use case approach and the realisation on how the design can be approached. In chapter 5, the evaluation of the work was produced. Then, in chapter 6, a discussion related to the aims and the objectives of the project, the learning points, and professional issues. The last chapter, chapter 7, is the summary of the project and future work.
2. Background

2.1 Teaching of High-Performance Computing

High-performance computing (HPC) underpins much scientific research[7][8][9] and industrial R&D[10]. Increasing of cores number per computer rather than increasing the speed of the core has improved the performance of the computer[11]. With the improvement of the multi-core computer design, the teaching of HPC has become fundamental for Computer Science(CS) students to teach them the concept of parallelism.

In a paper by Richard Brown in 2010, Brown and his working group has suggested strategies for teaching HPC for undergraduate CS students. They have discussed the critical points that HPC students must know after they got educated. They claim that the key concepts of teaching HPC, such as scalability, speedup, efficiency, and more are much important than the technical details. Hence, the changing of theses concepts is much slower than the changing of hardware and software capability[11].

Another paper by Violeta Holmes and Ibad Kureshi in 2015, they reviewed HPC courses at different UK universities and showed their experience on their own module HPC, Cluster, and Grid. The main topic of their paper is to show the approach to teach HPC for undergraduate CS students. Moreover, the topics that they teach and the tools they use are presented in the paper[12].

2.2 Differences between Container and Virtual Machine

Containers and Virtual Machines (VMs) are both considering a virtualization technique. In cloud computing, the virtualization technique is used to maintain the elasticity of large scaling shared resources. In addition, both containers and VMs are working on the infrastructure layer of cloud where they provide the cloud with a virtual operating system[13].

The biggest advantage of containers comparing to VMs is the lightweight virtualization solution that containers provide. The lightweight virtualization is beneficial in many cases when working with cloud services. Because of the concept of containers, the resources and time consuming are lesser than VMs which is providing a better solution for packaging applications in the cloud[14].

Both containers and VMs provide the developer with an operating system to work with. However, VMs is a full implementation of the OS, where in containers, it gives the developer a view of the OS that already running. Therefore, using VMs or containers is depending on the developer itself. Hence, if the developers want to deploy an application with less infrastructure, they will go with the approach of the container[13].
2.3 Docker in Education

Docker is a kernel-level technology that was developed based on the Linux operating system[15]. Therefore, Docker images are the packaging of the application inside a Docker container, and Docker containers are the execution environment of the application. In addition, Docker is a client-server implementation where Docker client is a command-line binary which used to control Docker containers by communication with the Docker server or daemon. Docker supports almost all operating systems and it can be run on desktops, laptops, and cloud.

Docker has been used to support the sharing of the environment between researchers and developers[2][3][16]. Moreover, Docker has been used to support Information Technology education[17] and has been used as a cloud-based lab management system for teaching programming languages[18]. Even though Docker has limitations of using it in education, it shows advantages as well.

In a study done by Keyuan Jiang and Qunhao Song[17] where they used container-based virtualization to teach a variety of subjects in the Information Technology curriculum. Their study showed the advantages of using Docker in education instead of the hypervisor-based virtual machine. Problems that can be occurred by using VMs, such as consuming system resources, running many VM on limited resources hardware at the same time, and the heavyweight of VMs image, can be solved by using Docker. The lightweight images that Docker produces, the ability to run many containers at the same time on one host, the ease to create a Docker container, and more are advantages of using container-based virtualization rather than hypervisor-based virtual. However, there are some limitations they faced such as the lack of graphical user interface features, the need for knowledge of Dockerfile commands to create and update Docker images, the challenge of running the user interface application.
3. Aims and Objectives

3.1 Aims

1. Using Docker to provide an initial environment to students that can be run on their own laptops, their university facilities, and on cloud instances.
2. Explore the pros & cons of using Docker to support HPC teaching
3. Evaluate the use of Docker for teaching and learning MPI.

3.2 Objectives

1. Build Docker image comprising MPI compiler and MPI examples.
2. Run Docker image on Azure cloud instance.
3. Evaluate the use of Docker Hub repository.
4. Evaluate the scalability of Docker image.
4. Design and Realisation

4.1 Docker image

A Docker Hub repository was used to keep the Docker image updated all the time during the stage of testing and evaluating the image. The image was pushed to the Docker Hub every time a change was made to the Dockerfile. The use of the Docker Hub repository allows me to share the container image as same it runs in the developing environment as run in a different environment.

On the stage of developing the image, three key points I had to consider. The first point is on which base image I should build my image. The base image is the image on which the MPI compiler and the editor will be built on top of. I have chosen the Ubuntu base image to be the base image for my image. Ubuntu considers being one of the most stable and lightweight Linux base image. For more comparison, refer to[24].

The second point is the MPI compiler. I have chosen the MPICH compiler to be the MPI compiler to be installed on top of Ubuntu. MPICH compiler is a free license compiler. Moreover, it can be found and installed using the Advanced Package Tools (apt) in Ubuntu. The last point is the editor. The Nano editor was chosen to be the editor for this image. The lightweight, free license, and the less User Interface make nano the perfect match for the HPC experience.

4.2 Use Case

Since the main aim of this project is to use Docker to support teaching and learning of HPC, especially learning MPI, there are two scenarios we have to consider. The two scenarios are the use cases of dockers in a real class. The first use case is providing the student with an initial environment so that the students can run examples from the class on their own time. And the second use case is that students share their environment with the teacher so the teacher can mark the students' work.

4.2.1 teachers sharing their environment

Traditionally, when students want to get examples that were used in class by the teacher, the teacher would upload the examples into a web page or send the examples via email. However, by using Docker, teachers can comprise the examples aside with a compiler and a text editor into one container. Then, share that container with students so they can examine and rerun examples on their own time.
Figure 1 shows an approach where teacher can build Docker image and push the image to Docker Hub. Then, students can pull image from Docker Hub to their computer and run teacher’s environment on their own time.

On this approach, teachers can use the following Dockerfile which is the same Dockerfile was used on this project.

```
1 #### BASE IMAGE ####
2 FROM ubuntu
3
4 MAINTAINER HAMED ALGHAMDI <H.Alghamdi2@liverpool.ac.uk>
5
6 #### INSTALL THE REQUIRED DEPENDENCY ####
7 ARG REQUIRE="sudo build-base"
8 RUN apt-get update -y && apt-get upgrade -y
9 RUN apt-get install -y mpich
10 RUN apt-get install -y nano
11
12 #### ADD DEFAULT USER ####
13 USER root
14
15 #### CREATE WORKING DIRECTORY FOR USER ####
16 ARG WORKDIR=/project
17 ENV WORKDIR ${WORKDIR}
18 WORKDIR ${WORKDIR}
19
20 #### ADD ALL FILES IN THE DOCKERFILE DIRECTORY ####
21 ADD . ${WORKDIR}
22```

Screenshot 1: Dockerfile

Figure 2: Teacher Sharing Environment
The following steps explain the steps need to be done by teachers in order to provide students the Docker container.

i. First thing to do is installing Docker engine. Docker engine is an application for building and running containers in your local machine [4]. To install the Docker engine, please refer to the Docker documentation [19].

ii. After installing the Docker engine, the teachers have to create a free account in Docker Hub. Docker Hub is a repository on which the teachers can share their container image with the students [20]. Docker Hub provides two kind of repository, public and private.

iii. Now, teachers might log in to their repository from Docker engine CLI by using the command `docker login`, and then entering the username and the password of the Docker Hub.

![Screenshot 2: Docker login](image)

Login Succeed is the indicator that you have successfully logged in to your repository. This step will allow you to push the container image to your repository after building the image.

iv. Before building up the container image, teachers need to prepare the examples that they are going to use in the class alongside the Dockerfile. So, create a folder that contain the Dockerfile and another folder contain the examples.

![Screenshot 3:](image)

The Dockerfile must be named as Dockerfile.

v. On the Dockerfile directory, run:

`docker build -t <repository_username>/container_name .`  

and you should see output similar to:
After building the image, you might run `docker image ls` to see the image information.

vi. Finally, run `docker push <repository_username>/container_name` to push the docker image to the Docker Hub.

Now in order to use the image by students, the students have to do the following steps:

i. Installing Docker engine in their working environment. Since Docker support all operating system, students can install Docker in their own laptop or in a cloud instance.

ii. Second step is pulling the image from the teacher repository to the Docker engine. Run `docker pull <teacherRepositoryName>/imageName:latest`. This command will pull the last image tag was pushed by the teacher.

iii. Finally, run `docker run -it --rm <teacherRepositoryName>/imageName:latest`. The argument -it is to attach a console for the container; and the --rm is to remove the file system after exit the container. When run the command, a console will start where students can run mpicc, and mpirun.
4.2.2 students sharing their environment

The second experiment performed in this project is students sharing their environment with the teacher. The main purpose of this experiment is in case of doing a practical assessment, for which traditionally the students submit their work via a zip file. The zip file will contain their code and a report, for example. However, by using Docker containers, students can comprise their code and the report inside a container and share the container with the teacher. The use case here has no difference from the use case was shown in section 4.2.1 when teachers sharing their environment with the students. Meaning, students will build the image in their environment then push the image to Docker Hub to share the image with the teacher.

Students can use the teacher’s image as a base image for their own image. So, students do not have to learn the Docker command to write up Dockerfile. However, a Dockerfile must be written in order to build their image. The following screenshot is Dockerfile can be used by students.

```
### BASE IMAGE ###
FROM hamedmg/demo

### ADD DEFAULT USER ###
USER root

### ADD ALL THE FILES IN THE DIRECTORY ###
ADD ./${WORKDIR}
```

*Screenshot 5: Students Dockerfile*

This Dockerfile will add all the files in the Dockerfile directory. Students can use any editor to complete their MPI assignment and then use Dockerfile to build a container to compile and run their program.

To use Docker Hub effectively, teachers have to create an organization to use Docker Hub with the team. Teams and organizations which allow controlling the access to the shared image[21]. This feature allows the only owner of the team, which in this case the teacher, to have access to all repositories in the organization.
5. Evaluation

In this chapter, we discuss the performance of the Docker image in terms of the size of the image was produced, the use of Docker repository, and the scalability of the produced image.

5.1 Size of the Image

Docker produces a lightweight application’s image. we used Ubuntu as a base image on which MPICH compiler and Nano editor installed on top of. The image is built by using the commands on the Dockerfile was shown in section 4.2.1. This Dockerfile produces an image with a size of 360MB. The lightweight nature of the image is going to be useful when using Docker in the real class. Since sharing between teachers and students will not be a problem because of the small size. In addition, the lightweight image will take almost nothing on the students’ computer space nor on the cloud instance.

5.3 Scalability of Docker Container

Docker has no limit accessing machine resources[23]. We have used two machines to test the scalability of Docker container image. The first machine is a local computer with Intel Core i7-8650 CPU. The number of cores in use is 4 physical cores. The following tables show the speed of the code on local machine without using docker in the first table and with using Docker container in the second one. The code we used can be found in appendix.

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>elapsed time</td>
<td>5.894 second</td>
<td>2.953 second</td>
<td>1.981 second</td>
<td>1.555 second</td>
</tr>
</tbody>
</table>

*Table 1: Local Machine without Using Docker*

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>elapsed time</td>
<td>5.822 second</td>
<td>3.011 second</td>
<td>1.968 second</td>
<td>1.597 second</td>
</tr>
</tbody>
</table>

*Table 2: Local Machine with Using Docker Container*

The second machine is Azure VM instance D4s v3 which has 4 vCPUs. The following table shows the speed of the code on Azure VM.

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>elapsed time</td>
<td>6.048 second</td>
<td>3.363 second</td>
<td>3.372 second</td>
<td>2.122 second</td>
</tr>
</tbody>
</table>

*Table 3: Docker Container on Azure VM*
And to compare the performance, we used Barkla which is a supercomputer facility in University of Liverpool. We run the code on Barkla login node without using Docker container. The following table shows the speed of the code on Barkla.

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>elapsed time</td>
<td>6.8 second</td>
<td>3.505 second</td>
<td>2.445 second</td>
<td>2.146 second</td>
</tr>
</tbody>
</table>

*Table 4: Barkla*

These numbers show the notion scalability which is one of the concept that HPC course taught[11]. Scaling is the ability of using multiple CPUs to increase the power of computational when the number of CPUs increase. In parallel computing, the concept of Speedup is essential. It means how much faster the parallel code than the sequential one. And to calculate the speedup, we use:

\[
\text{Speedup} = \frac{t_1}{t_C}
\]

Where \(t_1\) is the running time of the sequential code, and \(t_N\) is the running time when using \(C\) number of cores. Another concept is Efficiency. We use efficiency to see how efficient the parallel code on using cores capacity. And to calculate the efficiency, we use

\[
\text{Efficiency} = \frac{\text{Speedup}_C}{C}
\]

By applying these two equations to the result we obtained, we got:

<table>
<thead>
<tr>
<th></th>
<th>Sequential Time</th>
<th>Parallel Time</th>
<th>Speedup</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local Machine</strong></td>
<td>5.822</td>
<td>2 cores 3.011</td>
<td>1.933</td>
<td>0.966</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 cores 1.968</td>
<td>2.958</td>
<td>0.986</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 cores 1.597</td>
<td>3.645</td>
<td>0.911</td>
</tr>
<tr>
<td><strong>Azure Machine</strong></td>
<td>6.048</td>
<td>2 cores 3.363</td>
<td>1.798</td>
<td>0.899</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 cores 3.372</td>
<td>1.793</td>
<td>0.597</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 cores 2.122</td>
<td>2.850</td>
<td>0.712</td>
</tr>
<tr>
<td><strong>Barkla</strong></td>
<td>6.8</td>
<td>2 cores 3.505</td>
<td>1.94</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 cores 2.445</td>
<td>2.781</td>
<td>0.927</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 cores 2.146</td>
<td>3.168</td>
<td>0.792</td>
</tr>
</tbody>
</table>

*Table 5: Speedup and Efficiency of Docker Data*

These two concepts are important in teaching HPC so that students know how well their parallel solution is.
6. Discussion

The aim of evaluating the use of Docker for teaching and learning MPI was not fully fulfilling. Docker containers show very well scalability. In addition, Docker Hub provides a solid solution for environment reproducibility. However, we have not evaluated the approach was shown in section 4.2 in a real class, yet.

During the work of this project, I countered some pros and cons of using Docker. The pros are 1) the lightweight images. It does not matter how many images I have created, images do not have much space in my computer nor affect the performance of my computer. 2) running multi containers at the same time, which is something I would never be able to do with virtual machines. 3) the ease of sharing the images I was creating in my computer to test them in Azure VMs by using Docker Hub. With the pull and push commands, it is a matter of seconds to share or to get the image.

On the other hand, the cons of Docker are: 1) Docker engine is not compatible with VM in Windows OS. Docker uses the Hyper-V feature on Windows, so every time I wanted to start the virtual machine I had to turn off the Hyper-V feature. 2) writing a robust Dockerfile needs a high level of experience on Docker command and on Linux filesystem.

6.1 learning points

During working on this project, a lot of skills and knowledge were gained. The technology of containerization was a new concept to me. Therefore, as I was working on the project the concept was getting clearer to me. Since I work as a teaching assistant in the Department of Computer Science at Alba University in Saudi Arabia, the knowledge I gained in this project can be beneficial to the growth of the CS department.

In addition, I have got a general idea about how we can use Docker to support the teaching of other subjects in Computer Science such as Data Bases and Programming Languages. Moreover, how we can use the Azure cloud to support the teaching pedagogy.

Carrying out individual work gives me the opportunity to improve my communication skills as well. This project is my first project as a research project. Therefore, my formal writing skill has been improved. In addition, two presentations were delivered during the period of the project. As a second language English speaker, presenting in English was a great opportunity to improve my oral communication.

6.2 professional issues

- Public interest

The project was about improving a pedagogy practice in teaching Computer Science subjects. Even though, the aim of the project is to support the teaching an learning MPI, the concept of using docker in education can be used in almost every other practical subject in Computer Science.
• **Professional Competence and Integrity**

The background reading that I did before starting the project gave me the knowledge and skills I needed to carry out this project. The background reading was mainly about pedagogy practices and containerization. Therefore, I did not clam any competence that I do not have.

• **Duty to Relevant Authority**

The Docker image was published for the public in Docker Hub repository. Therefore, the result was not misrepresented. Moreover, no confidential information was used in this project so there was no disclosing of the result.

• **Duty to the Profession**

To support other professionals, as one of my responsibility as member of BCS, the Dockerfile was used in this project is published in a public Docker Hub repository along with the Docker image was used in the project.
7. Conclusion

7.1 Summary

In summary, using of Docker to support the learning of MPI shows a well result in terms of scalability. In addition, provide students with Docker image so they can use it to run MPI code is easier than install MPI compiler in their machine. Hence, installing MPI compiler is not what the HPC curriculum aiming to teach. Using of Docker also support marking students work. By using Docker Hub, sharing environment will make the marking process easier for the teachers.

7.2 Future Work

Until this point of the project, we evaluate how well containers scaling. However, evaluating how students feel about using containers still missing. So, a possible future work can be evaluating how students feel about using containers. Another possible work is to test containers on HPC facility and how containers will work in multi node.
8. References


4. Docker Engine Overview: https://docs.docker.com/engine/

5. MPI Forum: https://www.mpi-forum.org/docs/

6. MPICH Website: https://www.mpich.org/


19. Install Docker Engine: [https://docs.docker.com/engine/install/](https://docs.docker.com/engine/install/)


21. Docker Teams and Organization: [https://docs.docker.com/docker-hub/orgs/](https://docs.docker.com/docker-hub/orgs/)

22. Docker: [https://www.docker.com/](https://www.docker.com/)

23. Run Time Options with Memory, CPUs, and GPUs: [https://docs.docker.com/config/containers/resource_constraints/](https://docs.docker.com/config/containers/resource_constraints/)

9. Appendix

9.1 The MPI code was used to test the scalability

**func1.c:**

```c
#include <math.h>
#define MAX(x,y) ( x>y ? x : y )
double func (double x) {
    return pow(x,1.5)/3.1 - x/log(3.0);
}
```

**mpi_quad.c:**

```c
/*
 * COMP328: MPI solution for quadrature
 * (c) m.k.bane@liverpool.ac.uk
 */
#include <stdio.h>
#include <math.h>
#include <mpi.h>       // header for MPI
int main(void) {
    const double a=0.0, b=100.0;             // bounds
    const int numQuads = 90000000;
    double stepsize=(b-a)/(double) numQuads;  // stepsize in x-space
    double x, y, meanHeight;
    int i;
    double t0, t1;                            // timing
    double sum=0.0;
    double localSum=0.0;
    /* for MPI */
    int numProcesses, rankNum;
    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &numProcesses);
    MPI_Comm_rank(MPI_COMM_WORLD, &rankNum);
    if(rankNum==0) t0 = MPI_Wtime();
    /* split iterations over MPI processes */
    int numQuads_perProcess = numQuads/numProcesses;
    if (rankNum == numProcesses-1) {
        numQuads_perProcess = numQuads - (numQuads_perProcess*rankNum);
    }
    int start_perProcess = rankNum*numQuads_perProcess;
    int finish_perProcess = start_perProcess + numQuads_perProcess - 1;
    // printf("rank %d\%d start=%d finish=%d\n", rankNum, numProcesses, start_perProcess, finish_perProcess);
    /* by having different sub-ranges of the iteration space,
     * we have divided the work between the MPI processes */
    for (i=start_perProcess; i<=finish_perProcess; i++) {
        x = a + i*stepsize;
```
y = x + stepsize;
meanHeight = 0.5 * (func(x) + func(y));
localSum += meanHeight*stepsize;
}

/* now we have to sum all 'localSum' from each rank into 'sum' on rank=0 */
MPI_Reduce(&localSum, &sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
if (rankNum==0) {
    t1 = MPI_Wtime();
    printf("TOTAL SUM: %f
%d iterations took total wallclock %f milliseconds\n", sum,numQuads,1000.0*(t1-t0));
} 
MPI_Finalize();
}
9.2 Dockerfile

```bash
#### BASE IMAGE ####
FROM ubuntu
MAINTAINER HAMED ALGHAMDI <H.Alghamdi2@liverpool.ac.uk>
#### INSTALL THE REQUIRED DEPENDENCY ####
ARG REQUIRE="sudo build-base"
RUN apt-get update -y && apt-get upgrade -y
RUN apt-get install -y mpich
RUN apt-get install -y nano
#### ADD DEFAULT USER ####
USER root
#### CREATE WORKING DIRECTORY FOR USERS ####
ARG WORKDIR=/project
ENV WORKDIR ${WORKDIR}
WORKDIR ${WORKDIR}
#### ADD ALL FILES IN THE DOCKERFILE DIRECTORY ####
ADD . ${WORKDIR}
```