Alfonso De La Rosa
00001051360
COEN 241: Cloud Computing
Professor Sean Choi
02/01/2023

COEN 241 Homework 1

Introduction

The goal of this assignment is to analyze the performance of Ubuntu 16.04 in both QEMU and Docker. This homework assignment will provide a summary of configurations of the computer system used for these experiments, enabling both QEMU and Docker, screenshots of the experiment, how the bash script performs these experiments, the code of the bash script and contents of the commands.txt file, how these measurements were conducted, summaries of each experiment, conclusion of the experiment, and an appendix with tables and figures.

Git Repository Information

- **https://github.com/alfonsodelarosa4/SCU_COEN_241_Cloud_Computing_Assignments.git**

Detailed configurations (CPU, Mem, etc...) of your experimental setup

- Dell XPS 13 Windows Laptop
  - Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2419 Mhz, 4 Core(s), 8 Logical Processor(s)
  - System type: x64-based PC
  - RAM: 16.0 GB
  - OS: Microsoft Windows 11 Home

Steps to enable a QEMU VM

To install:

1. I followed the following instructions from this article: https://medium.com/@dhanar.santika/installing-wsl-with-gui-using-vcxsrv-6f307e96fac0
   a. I went to Windows features and turned on *Windows SUbsystem for Linux* and *Virtual Machine Platform*
   b. I downloaded and installed *Ubuntu 22.04.1 LTS* from the Microsoft Store

c. I downloaded and installed the Linux kernel update package from this link: https://learn.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
d. I created a username: alfdelarosa4
e. I ran the following commands:
    i.

Installs WSL: sudo apt-get update sudo apt-get upgrade

f. I downloaded and installed the vcxsrv server from this link: https://sourceforge.net/projects/vcxsrv/files/latest/download
g. I opened *XLlaunch*
    i. Selected One Large Window
    ii. Start no client
    iii. Check Disable access control
h. Entered
    i.

Installed Desktop Environment for your WSL sudo apt-get install xfce4
More commands cd ~ nano .bashrc

i. Added the following to the last line:
    i. export DISPLAY="`grep nameserver /etc/resolv.conf | sed 's/nameserver //'`:0"
    ii. export LIBGL_ALWAYS_INDIRECT=1
j. Saved the file
k. Closed and ran WSL

2. Then, I followed the linux instructions
    a.

Install Ubuntu sudo apt-get install qemu
Go to where Ubuntu server iso file is located cd ~ cd /mnt/c/Users/Alf/”VirtualBox VMs”
Created a qemu image sudo qemu-img create ubuntu.img 10G -f qcow2
Install VM sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ./ubuntu-16.04.7-server-amd64.iso -m 2046 -boot strict=on -enable-kvm

3. After running the above command, the Ubuntu installation installer loaded.

4. Answered the questions in Ubuntu 16.04 installer
5. After the installation, it rebooted back to installation.
6. Closed WSL
7. Opened WSL again
8. Then, I rebooted the following command without the cdrom parameter

Start VM sudo qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -boot strict=on -enable-kvm

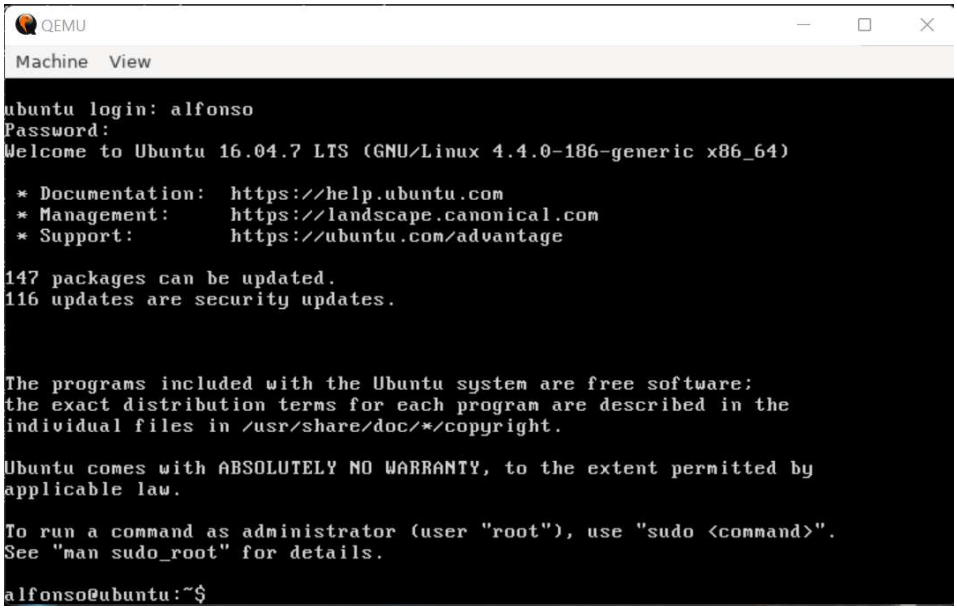

Figure 1: Screenshot of QEMU

Steps to enable the Docker container

Steps to install Docker

1. I followed the following instructions to install Docker in Windows: https://docs.docker.com/desktop/install/windows-install/
    a. Downloaded and installed Docker Desktop for Windows
    b. Opened Docker Desktop for Windows
2. Opened the command line and ran the following commands
    a.

Download image of ubuntu 16.04 to match with ubuntu 16.04 on qemu docker pull ubuntu:16.04
Run image of ubuntu 16.04 docker run -it ubuntu:16.04

3. Install sysbench on docker's Ubuntu and on QEMU's Ubuntu
    a.

```
sudo apt update
sudo apt install sysbench
```

Commands to Manage Docker

Create a docker container

Create a container from docker image of ubuntu 16.04 (in interactive mode)
docker run -it ubuntu:16.04

Close a docker container

Exit container
docker stop [container name]

Restart container and used terminal

Restart a specific container
docker restart [container name]

Attach to the terminal of docker container
docker attach [container name]

Run commands in a container without being in interactive mode

docker exec [container name] [actual command]

Screenshots of Docker

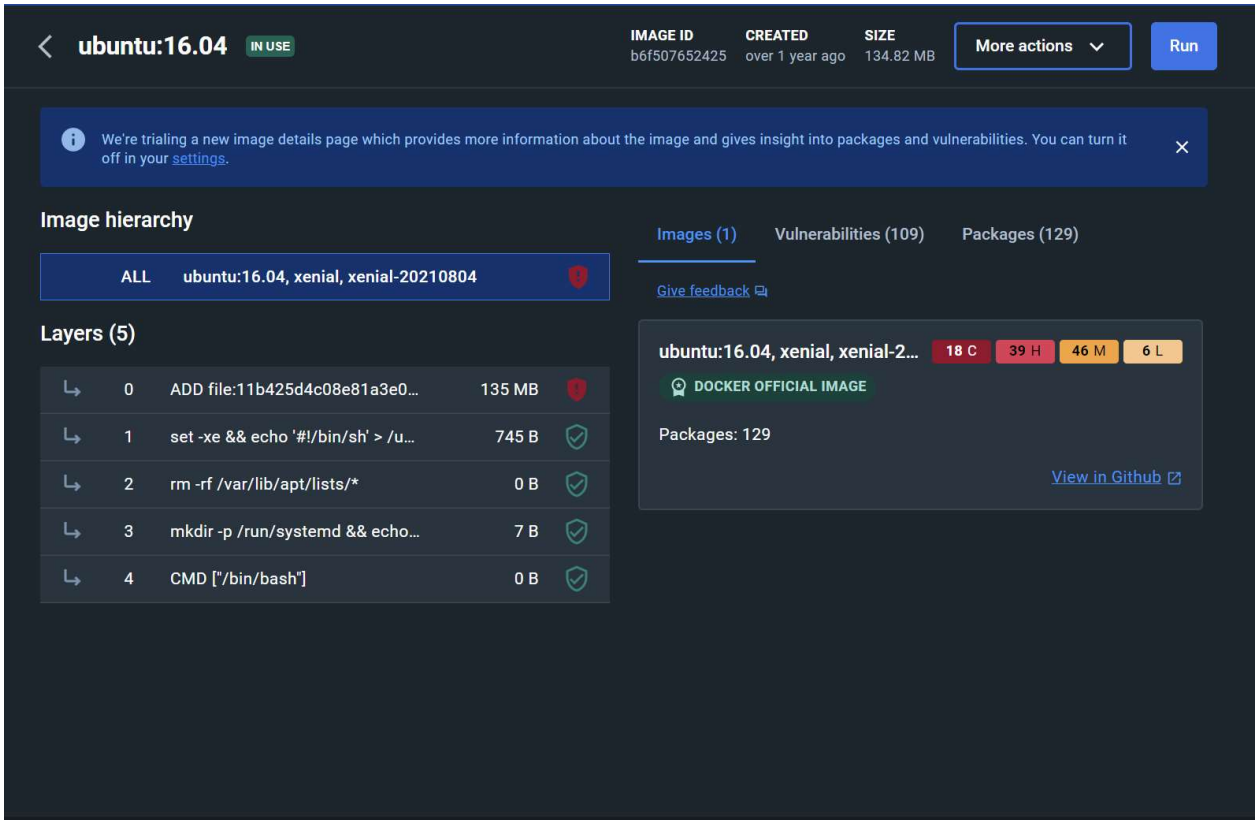

Figure 2: Image creation history of Docker Image

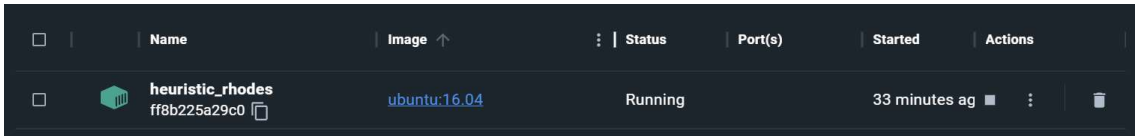

Figure 3: Screenshot of container created from Docker image

Proof of experiment

Both Docker's and QEMU's Ubuntu images have the same version of sysbench: 0.4.12

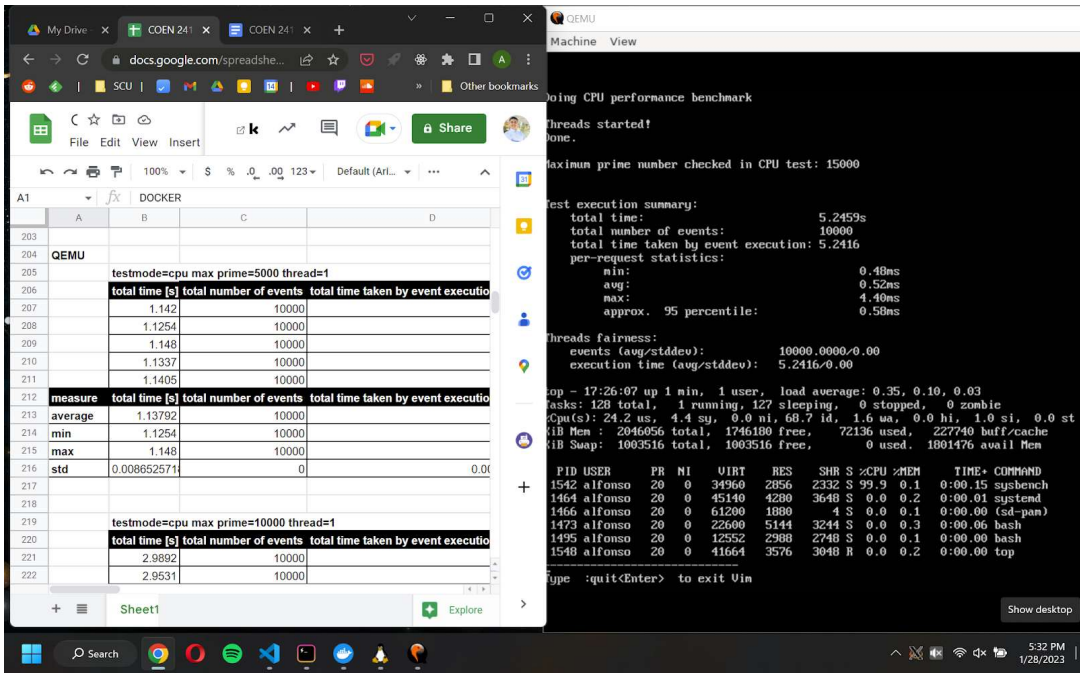

Figure 2: Proof of Experiment on QEMU

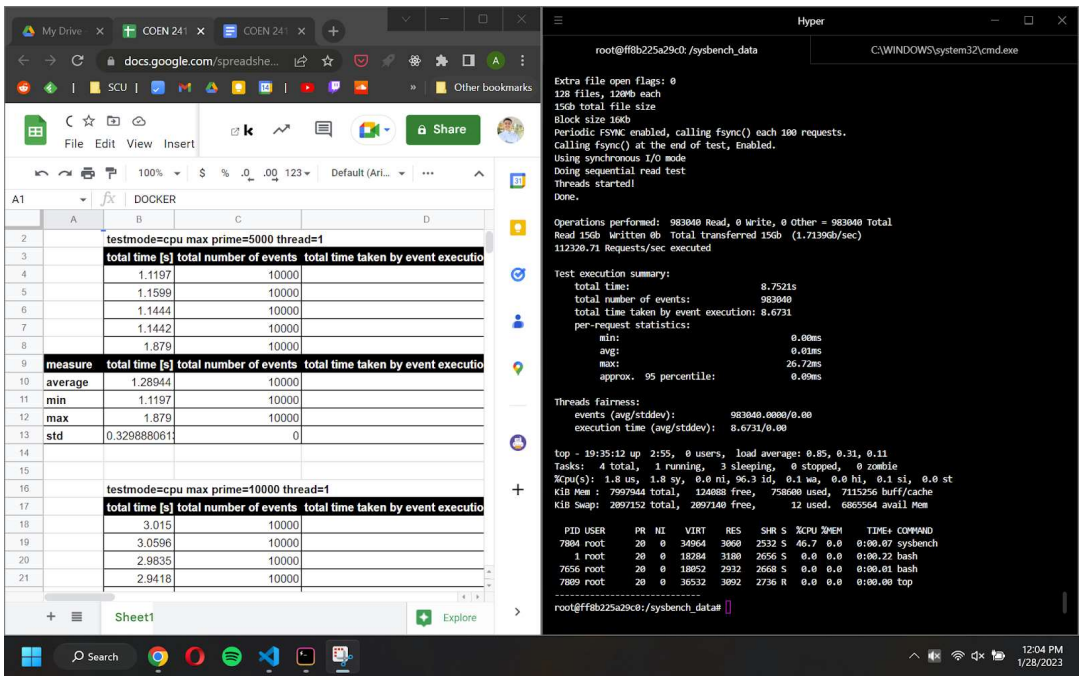

Figure 3: Proof of Experiment on Docker

The use of performance tools

For this assignment, I created a bash script to conduct several sysbench experiments. The same bash script is used for Ubuntu in both a Docker container and QEMU. Several experiments will be conducted for each virtual technology, such as testing different cpu-max-prime values and different thread counts for the cpu benchmark and testing different thread counts and different file sizes for the fileio benchmark.

In high-level, the bash script conducts different experiments. Each different experiment has different test cases each represented as a sysbench command with specific parameters. Since the commands.txt file has a list of sysbench commands, the bash script would retrieve each sysbench command from the command.txt file. Every four adjacent sysbench commands in commands.txt are all the test cases of the same experiment.

The results of each test case of an experiment are stored in a separate text file. For each test case or sysbench command, the program would execute the same sysbench command five times to collect averages, maximum values, minimum values, and the standard deviation. For each execution of the five executions, the results of the execution of a sysbench command and a snapshot of the running processes (with the linux top command) will be appended to the txt file of that test case. After the script is finished, each text file will be reviewed to collect the data of each test case for each experiment.

These were the different experiments conducted:

```
# Experiment 1: Different cpu-max-prime values
sysbench --test=cpu --cpu-max-prime=5000 run
sysbench --test=cpu --cpu-max-prime=10000 run
sysbench --test=cpu --cpu-max-prime=15000 run
sysbench --test=cpu --cpu-max-prime=20000 run
# Experiment 2: Different thread counts for the cpu tests
sysbench --test=cpu --num-threads=1 run
sysbench --test=cpu --num-threads=5 run
sysbench --test=cpu --num-threads=10 run
sysbench --test=cpu --num-threads=15 run
# Experiment 3: Different thread counts for fileio write
sysbench --test=fileio --num-threads=1 --file-test-mode=seqwr run
sysbench --test=fileio --num-threads=5 --file-test-mode=seqwr run
sysbench --test=fileio --num-threads=10 --file-test-mode=seqwr run
sysbench --test=fileio --num-threads=15 --file-test-mode=seqwr run
# Experiment 4: Different file sizes for fileio reads
sysbench --test=fileio --file-total-size=2G --file-test-mode=seqrd run
sysbench --test=fileio --file-total-size=4G --file-test-mode=seqrd run
```

Shell scripts for running the experiment

experiments.sh

```
#!/bin/bash

# This is a bash script used COEN 241 Homework 1
# Alfonso De La Rosa
# This bash script will be used for two virtual technologies
# Docker and QEMU

# the commands.txt file with sysbench commands will be required for this experiment
# In addition, every four adjacent commands are test cases of an experiment.

# creates new folder
mkdir sysbench_data

# iterate through each test case of an experiment
for index in 1 2 3 4 5 6 7 8 9 10 11 12 13 14
do
   # outputs the sysbench command of that experiment
   head -$index commands.txt | tail -1

   # add line separator to text file
   echo "-----------------------------" >> sysbench_data/"sys_output$((index))".txt

   # first line in new text file = command
   echo "COMMAND: $(head -$index commands.txt | tail -1)" >>
sysbench_data/"sys_output$((index))".txt

   # since the last 2 test cases read GB of data,
   # these commands create the GB of data to read
   if ((index == 13)); then
      sysbench --test=fileio --file-total-size=2G --file-test-mode=seqrd prepare
   elif ((index == 14)); then
      sysbench --test=fileio --file-total-size=4G --file-test-mode=seqrd prepare
   fi
   # executes each test case of an experiment five times
   for trial in 0 1 2 3 4
   do

      # execute sysbench command in the background and append results to a separate
text file
      $(head -$index commands.txt | tail -1) >> sysbench_data/"sys_output$((index))".txt &

      # delay a few millisecond, so that the sysbench process appears in the running
processes
      sleep 0.004s

      # outputs snapshot of top command to txt file
```

```
        top -b -n 1 > sysbench_data/top_out.txt

        # waits until sysbench command finishes running
        wait

        # appends contents of top_out.txt to sys_output.txt
        cat sysbench_data/top_out.txt >> sysbench_data/"sys_output$((index))".txt

        # removes top_out.txt file
        rm sysbench_data/top_out.txt

        # tells user trial of test case is finished
        echo "done"

        echo "----------------------------" >> sysbench_data/"sys_output$((index))".txt

    done
    # since the last two test cases finished reading GB of data,
    # those files of data will be deleted not to take up space
    if ((index == 13)); then
        sysbench --test=fileio --file-total-size=2G --file-test-mode=seqrd cleanup
    elif ((index == 14)); then
        sysbench --test=fileio --file-total-size=4G --file-test-mode=seqrd cleanup
    fi
done

# removes all test files
rm test_file*
```

commands.txt file

```
sysbench --test=cpu --cpu-max-prime=5000 run
sysbench --test=cpu --cpu-max-prime=10000 run
sysbench --test=cpu --cpu-max-prime=15000 run
sysbench --test=cpu --cpu-max-prime=20000 run
sysbench --test=cpu --num-threads=1 run
sysbench --test=cpu --num-threads=5 run
sysbench --test=cpu --num-threads=10 run
sysbench --test=cpu --num-threads=15 run
sysbench --test=fileio --num-threads=1 --file-test-mode=seqwr run
sysbench --test=fileio --num-threads=5 --file-test-mode=seqwr run
sysbench --test=fileio --num-threads=10 --file-test-mode=seqwr run
sysbench --test=fileio --num-threads=15 --file-test-mode=seqwr run
sysbench --test=fileio --file-total-size=2G --file-test-mode=seqrd run
sysbench --test=fileio --file-total-size=4G --file-test-mode=seqrd run
```

Measurements in three different scenarios for each virtualization technology

After the bash script is finished, the measurements must be recorded. The process is to iterate through the text of each text file and record the important values of each trial of each test case onto a table on excel. Then, the excel will calculate the maximum, minimum, average, and standard deviation of the values onto the table. Afterwards, the averages are placed onto the graph. Each graph will contain the averages of each test case of an experiment and contain the test cases from both Docker and QEMU to compare and analyze. In addition, the tables of measurements and graphs of the relationships between Docker and QEMU for each experiment are in the appendix.

1) # Experiment 1: Different cpu-max-prime values

The following sysbench commands are test cases for this experiment.

```
sysbench --test=cpu --cpu-max-prime=5000 run
sysbench --test=cpu --cpu-max-prime=10000 run
sysbench --test=cpu --cpu-max-prime=15000 run
sysbench --test=cpu --cpu-max-prime=20000 run
```

After the execution of the program, the results from the test cases were placed in a table of an excel document, which is shown in the appendix. The main results recorded for each test case are: total time, total number of events, total time taken by event execution, user cpu time, and kernel cpu time. Excel then calculates the averages, minimum, maximum, and the standard deviation for each of these values. Then, the averages of the total time to execute a test case for each virtual technology are plotted on a graph, shown in a later section. Although both trends seem close to each other, Docker outperforms QEMU according to Table 18.

2) # Experiment 2: Different thread counts for the cpu tests

The following sysbench commands are test cases for this experiment.

```
sysbench --test=cpu --num-threads=1 run
sysbench --test=cpu --num-threads=5 run
sysbench --test=cpu --num-threads=10 run
sysbench --test=cpu --num-threads=15 run
```

Once the bash script is finished, the results from the test cases are then placed in a table of an excel document, which is after the conclusion. The results of this experiment are similar to the previous one: total time, total number of events, total time taken by event execution, user cpu time, and kernel cpu time. Excel computes the averages, minimum, maximum, and the standard deviation for each of these values. Afterwards, the averages of the total time values to execute a test case for both Docker and QEMU are plotted on a graph, later. Compared to the previous experiment, total time values for both Docker and QEMU are easier to compare, according to graph Figure 2 in the appendix. In this experiment, Docker still performs faster than QEMU.

3) # Experiment 4: Different file sizes for fileio reads

The following sysbench commands are test cases for this experiment.

```
sysbench --test=fileio --file-total-size=2G --file-test-mode=seqrd run
sysbench --test=fileio --file-total-size=4G --file-test-mode=seqrd run
```

After the program finished, the results from the test cases were placed in an Excel document. The main results recorded for each test case are: total time, total number of events, latency: total time taken by event execution, disk utilization [GB], and throughput [GB/s]. From these values, Excel computes the averages, minimum, maximum, and the standard deviation for each of these values. Then, the averages of the total time to execute a test case are plotted on a graph, shown later. According to Figure 4, both trends are very different from each other. The total time for QEMU to execute these commands is longer than the total time for Docker to execute these commands.

Experiment Summaries

The following sections will describe why the parameters used for each experiment were chosen. As mentioned earlier, the tables of measurements and graphs of the relationships of Ubuntu's performance between Docker and QEMU for each experiment are in the appendix.

Experiment 1: Different cpu-max-prime values
This experiment was difficult to perform. The following parameters were chosen for cpu max prime in the cpu test: 20000, 30000, 40000, and 50000. Even though the total time values might seem close to each other before 30000 milliseconds, or 30 seconds, afterwards, there is a small difference between the total time values, where the total time values for QEMU is greater than the total time values for Docker.

Experiment 2: Different thread counts for the cpu tests
For this experiment, the following parameters were chosen for thread count for the cpu test: 1, 5, 10, and 15. These values were chosen because they were large enough to see the difference between total time between Docker and QEMU and small enough to prevent the benchmark from computing for too long. According to Figure 2, total time values for Docker are shorter than the total times for QEMU.

Experiment 3: Different thread counts for fileio write
For this experiment, the following values were chosen for thread count for the fileio test: 1, 5, 10, and 15. These values were used for this experiment for similar reasons. Even though these values are small, they were big enough to highlight the discrepancies in total time between Docker and QEMU, where the total time values of QEMU are longer than those of Docker.

Experiment 4: Different file sizes for fileio reads
In this experiment, the following file sizes were used for the fileio test: 2G and 4GB. Previous tests demonstrated that values greater than and equal to 6 GB caused the benchmark to run out of space for QEMU. Despite the limitation, these two values were enough to show the difference in performance between Docker and QEMU, where Docker's Ubuntu performs those tasks faster than QEMU's Ubuntu.

Conclusion

Overall, the experiments for this homework showed one important insight: that the total time values for QEMU were longer than the total time values for Docker. In addition, QEMU is an example of system virtualization while Docker is an example of OS virtualization. Because of this fact, it makes sense that Ubuntu performs faster in Docker than in QEMU.

Appendix 1: Measurements and Graphs

Experiment 1: Different cpu-max-prime values

Docker:

Table 1: testmode=cpu max prime=5000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	1.1197	10000	1.1189	3.3	3.2
	1.1599	10000	1.591	3.3	3.2
	1.1444	10000	1.429	3.3	3.2
	1.1442	10000	1.434	3.3	3.2
	1.879	10000	1.1869	3.3	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	1.28944	10000	1.35196	3.3	3.2
min	1.1197	10000	1.1189	3.3	3.2
max	1.879	10000	1.591	3.3	3.2
std	0.3298880613	0	0.194529954	0	0

Table 2: testmode=cpu max prime=10000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	3.015	10000	3.0139	3.3	3.2
	3.0596	10000	3.0586	3.3	3.2
	2.9835	10000	2.9825	3.3	3.2
	2.9418	10000	2.9408	3.3	3.2
	2.9573	10000	2.9563	3.3	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	2.99144	10000	2.99042	3.3	3.2
min	2.9418	10000	2.9408	3.3	3.2
max	3.0596	10000	3.0586	3.3	3.2
std	0.04715286842	0	0.04714039669	0	0

Table 3: testmode=cpu max prime=15000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	5.1654	10000	5.1642	3.3	3.2
	5.1384	10000	5.1372	3.4	3.2
	5.1927	10000	5.1915	3.4	3.2
	5.22	10000	5.2185	3.4	3.2
	5.2069	10000	5.2056	3.4	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	5.18468	10000	5.1834	3.38	3.2
min	5.1384	10000	5.1372	3.3	3.2
max	5.22	10000	5.2185	3.4	3.2
std	0.03285996044	0	0.03276255485	0.04472135955	0

Table 4: testmode=cpu max prime=20000 thread=1

	total time [s]	total number of events	5	us: user cpu time [s]	sy: kernel cpu times [s]
	7.6398	10000	7.6386	3.4	3.2
	7.565	10000	7.5637	3.4	3.2
	7.6614	10000	7.6601	3.4	3.2
	7.7405	10000	7.7391	3.4	3.2
	7.7177	10000	7.7163	3.5	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	7.66488	10000	7.66356	3.42	3.2
min	7.565	10000	7.5637	3.4	3.2
max	7.7405	10000	7.7391	3.5	3.2
std	0.06914742945	0	0.06909195322	0.04472135955	0

QEMU

Table 5: testmode=cpu max prime=5000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	1.142	10000	1.14409	9.3	2.5
	1.1254	10000	1.1237	9.3	2.5
	1.148	10000	1.1457	9.3	2.5

	1.1337	10000	1.1309	9.4	2.5
	1.1405	10000	1.139	9.4	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	1.13792	10000	1.136678	9.34	2.5
min	1.1254	10000	1.1237	9.3	2.5
max	1.148	10000	1.1457	9.4	2.5
std	0.008652571872	0	0.009267411721	0.05477225575	0

Table 6: testmode=cpu max prime=10000 thread=1

	total time [s]	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
	2.9892	10000	2.9859	9.4	2.5
	2.9531	10000	2.952	9.5	2.5
	3.0204	10000	3.0163	9.5	2.5
	3.0264	10000	3.0246	9.6	2.5
	2.9432	10000	2.9416	9.6	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	2.98646	10000	2.98408	9.52	2.5
min	2.9432	10000	2.9416	9.4	2.5
max	3.0264	10000	3.0246	9.6	2.5
std	0.03787793553	0	0.03713821482	0.08366600265	0

Table 7: testmode=cpu max prime=15000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	5.1537	10000	5.1522	9.7	2.5
	5.1776	10000	5.1712	9.8	2.5
	5.1357	10000	5.1291	9.9	2.5
	5.1495	10000	5.1449	10	2.5
	5.2201	10000	5.2174	10.1	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	5.16732	10000	5.16296	9.9	2.5
min	5.1357	10000	5.1291	9.7	2.5
max	5.2201	10000	5.2174	10.1	2.5
std	0.033144713	0	0.03398548219	0.158113883	0

Table 8: testmode=cpu max prime=20000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	7.7725	10000	7.7709	10.2	2.5
	7.6491	10000	7.6463	10.3	2.5
	7.6403	10000	7.6345	10.3	2.5
	7.6583	10000	7.6553	10.4	2.5
	7.7953	10000	7.7925	10.7	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	7.7031	10000	7.6999	10.38	2.5
min	7.6403	10000	7.6345	10.2	2.5
max	7.7953	10000	7.7925	10.7	2.5
std	0.074471605 33	0	0.0754238689	0.1923538406	0

Table 9: EXPERIMENT 1: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

cpu max prime	Docker (average total time)	QEMU (average total time)
5000	1.28944	1.13792
10000	2.99144	2.98646
15000	5.18468	5.16732
20000	7.66488	7.7031

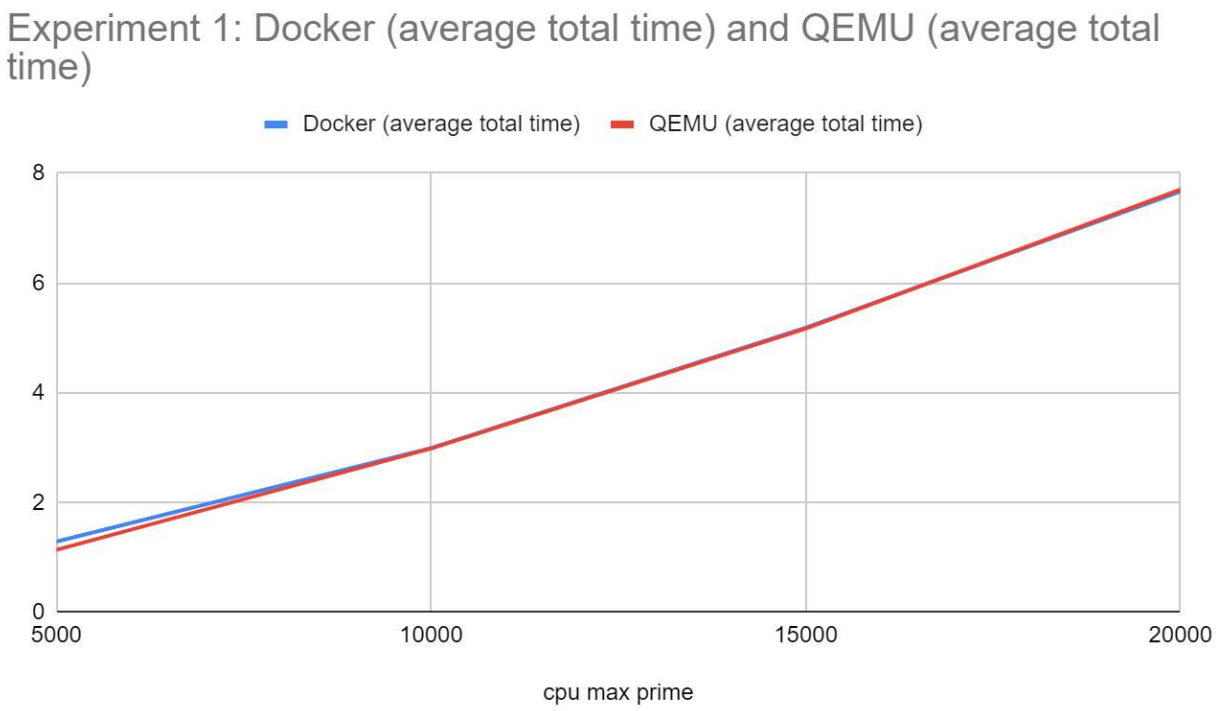

Figure 1: EXPERIMENT 1: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Experiment 2: Different thread counts for the cpu tests

Docker:

Table 10: testmode=cpu max prime=10000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	2.9598	10000	2.9588	3.5	3.2
	2.9549	10000	2.9539	3.5	3.2
	2.9606	10000	2.9596	3.5	3.2
	3.1298	10000	3.1287	3.5	3.2
	3.3003	10000	3.2989	3.5	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	3.06108	10000	3.05998	3.5	3.2
min	2.9549	10000	2.9539	3.5	3.2
max	3.3003	10000	3.2989	3.5	3.2
std	0.1529515512	0	0.1527839226	0	0

Table 11: testmode=cpu max prime=15000 thread=5

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	0.8728	10000	4.3601	3.5	3.2
	0.8818	10000	4.4058	3.5	3.2
	0.9257	10000	4.6251	3.5	3.2
	0.9052	10000	4.5226	3.5	3.2
	0.899	10000	4.4908	3.5	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	0.8969	10000	4.48088	3.5	3.2
min	0.8728	10000	4.3601	3.5	3.2
max	0.9257	10000	4.6251	3.5	3.2
std	0.02068550217	0	0.1035171338	0	0

Table 12: testmode=cpu max prime=20000 thread=10

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	0.8046	10000	8.0105	3.6	3.2
	0.7949	10000	7.8788	3.6	3.2
	0.7985	10000	7.9445	3.6	3.2
	0.9985	10000	9.9104	3.6	3.2
	0.8792	10000	8.6875	3.6	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	0.85514	10000	8.48634	3.6	3.2
min	0.7949	10000	7.8788	3.6	3.2
max	0.9985	10000	9.9104	3.6	3.2
std	0.08735303658	0	0.8598737367	0	0

Table 13: testmode=cpu max prime=1000 thread=15

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	0.8484	10000	12.4931	3.6	3.2
	0.8668	10000	12.8327	3.6	3.2
	0.8207	10000	12.1366	3.7	3.2
	0.843	10000	12.461	3.7	3.2
	0.8203	10000	12.1859	3.7	3.2
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	0.83984	10000	12.42186	3.66	3.2
min	0.8203	10000	12.1366	3.6	3.2
max	0.8668	10000	12.8327	3.7	3.2
std	0.01973735038	0	0.2794847813	0.05477225575	0

QEMU:

Table 14: testmode=cpu max prime=10000 thread=1

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	2.9509	10000	2.9496	10.8	2.5
	2.9451	10000	2.9409	10.9	2.5
	2.9383	10000	2.9362	10.9	2.5
	2.9486	10000	2.9451	11	2.5
	2.9599	10000	2.9529	11	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	2.94856	10000	2.94494	10.92	2.5
min	2.9383	10000	2.9362	10.8	2.5
max	2.9599	10000	2.9529	11	2.5
std	0.007926411546	0	0.006667308302	0.08366600265	0

Table 15: testmode=cpu max prime=15000 thread=5

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	2.981	10000	14.8485	11.1	2.5
	2.9289	10000	14.5984	11.1	2.5
	2.9715	10000	14.8243	11.2	2.5
	3.0069	10000	14.9885	11.3	2.5
	3.0343	10000	15.1363	11.3	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	2.98452	10000	14.8792	11.2	2.5
min	2.9289	10000	14.5984	11.1	2.5
max	3.0343	10000	15.1363	11.3	2.5
std	0.039542407 62	0	0.2005318179	0.1	0

Table 16: testmode=cpu max prime=20000 thread=10

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	3.1199	10000	30.995	11.4	2.5
	2.9857	10000	29.6767	11.4	2.5
	2.9596	10000	29.3988	11.5	2.5
	2.9489	10000	29.3299	11.5	2.5
	2.96	10000	29.3829	11.6	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	2.99482	10000	29.75666	11.48	2.5
min	2.9489	10000	29.3299	11.4	2.5
max	3.1199	10000	30.995	11.6	2.5
std	0.07122097303	0	0.7052949688	0.08366600265	0

Table 17: testmode=cpu max prime=1000 thread=15

	total time [s]	total number of events	total time taken by event execution [s]	us: user cpu time [s]	sy: kernel cpu times [s]
	3.0317	10000	45.0592	11.6	2.5
	2.9451	10000	43.7652	11.7	2.5
	2.9456	10000	43.7289	11.7	2.5
	3.0063	10000	44.6861	11.8	2.5
	2.9426	10000	43.669	11.8	2.5
measure	**total time [s]**	**total number of events**	**total time taken by event execution [s]**	**us: user cpu time [s]**	**sy: kernel cpu times [s]**
average	2.97426	10000	44.18168	11.72	2.5
min	2.9426	10000	43.669	11.6	2.5
max	3.0317	10000	45.0592	11.8	2.5
std	0.04183291766	0	0.6453268528	0.08366600265	0

Table 18: EXPERIMENT 2: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Thread count for cpu test	Docker (average total time)	QEMU (average total time)
1	3.06108	2.94856
5	0.8969	2.98452
10	0.85514	2.99482
15	0.83984	2.97426

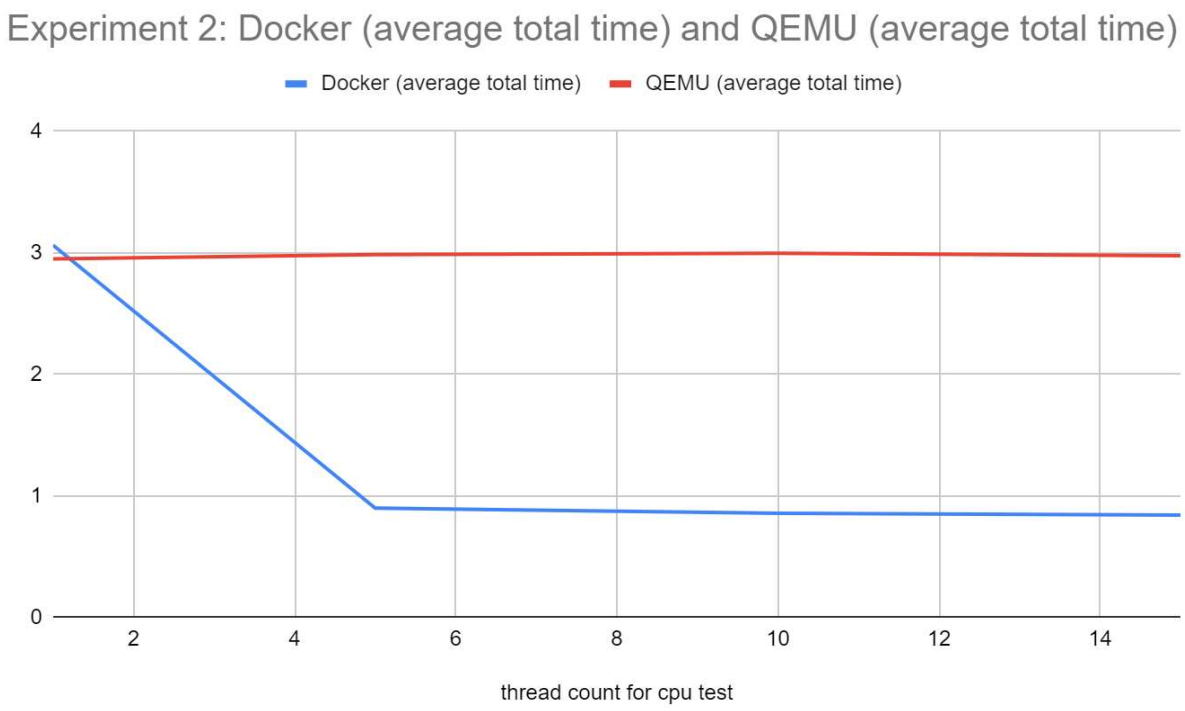

Figure 2: EXPERIMENT 2: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Experiment 3: Different thread counts for fileio write
Docker

Table 19: testmode=fileio thread=1 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	1.3146	131072	1.0425	2	1.5214
	1.3147	131072	0.8929	2	1.5212
	1.2177	131072	0.8067	2	1.6424
	1.3058	131072	0.8728	2	1.5317
	1.2327	131072	0.8085	2	1.6224
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	1.2771	131072	0.88468	2	1.56782
min	1.2177	131072	0.8067	2	1.5212
max	1.3147	131072	1.0425	2	1.6424
std	0.04781061598	0	0.09617708667	0	0.05952740545

Table 20: testmode=fileio thread=5 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	1.3344	131072	5.223	2	1.4988
	1.2804	131072	4.9147	2	1.562
	1.2806	131072	4.9565	2	1.5618
	1.3504	131072	5.3258	2	1.481
	1.3233	131072	5.1002	2	1.5113
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	1.31382	131072	5.10404	2	1.52298
min	1.2804	131072	4.9147	2	1.481
max	1.3504	131072	5.3258	2	1.562
std	0.0319059869	0	0.1738975359	0	0.03712468182

Table 21: testmode=fileio thread=10 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	1.7485	131072	14.5601	2	1.1438
	1.7018	131072	14.6954	2	1.1752
	1.6538	131072	13.6747	2	1.2094
	1.5581	131072	12.4758	2	1.2836
	1.553	131072	12.6024	2	1.2878
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	1.64304	131072	13.60168	2	1.21996
min	1.553	131072	12.4758	2	1.1438
max	1.7485	131072	14.6954	2	1.2878
std	0.08662033826	0	1.047173642	0	0.06435765689

Table 22: testmode=fileio thread=15 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	1.8151	131072	22.6419	2	1.1019
	1.6736	131072	20.358	2	1.195
	1.7797	131072	22.2888	2	1.1238
	1.9266	131072	24.5967	2	1.0381
	2.0122	131072	25.6253	2	1.0178
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	1.84144	131072	23.10214	2	1.09532
min	1.6736	131072	20.358	2	1.0178
max	2.0122	131072	25.6253	2	1.195
std	0.1314221176	0	2.06177673	0	0.07084135092

QEMU

Table 23: testmode=fileio thread=1 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	8.0003	131072	5.9078	11.9	2.5
	7.9716	131072	5.5965	11.9	2.5
	7.7027	131072	5.7111	11.8	2.5
	7.826	131072	5.5098	11.8	2.5
	8.5809	131072	6.1337	11.8	2.6
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	8.0163	131072	5.77178	11.84	2.52
min	7.7027	131072	5.5098	11.8	2.5
max	8.5809	131072	6.1337	11.9	2.6
std	0.3374856664	0	0.2512599192	0.05477225575	0.04472135955

Table 24: testmode=fileio thread=5 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	7.8471	131072	27.6958	11.8	2.6
	8.3873	131072	28.8891	11.8	2.6
	8.1726	131072	28.7717	11.8	2.6
	7.9012	131072	28.0584	11.8	2.6
	7.555	131072	29.4448	11.7	2.6
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	7.97264	131072	28.57196	11.78	2.6
min	7.555	131072	27.6958	11.7	2.6
max	8.3873	131072	29.4448	11.8	2.6
std	0.3190666278	0	0.6952919768	0.04472135955	0

Table 25: testmode=fileio thread=10 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	8.1361	131072	57.6956	11.7	2.7
	7.9052	131072	54.9762	11.7	2.7
	8.409	131072	58.335	11.7	2.7
	7.9404	131072	53.085	11.7	2.7
	8.7234	131072	63.1016	11.7	2.7
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	8.22282	131072	57.43868	11.7	2.7
min	7.9052	131072	53.085	11.7	2.7
max	8.7234	131072	63.1016	11.7	2.7
std	0.3440229673	0	3.806454967	0	0

Table 26: testmode=fileio thread=15 file-test-mode=sequential write

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	7.9297	131072	80.0601	11.7	2.8
	8.6436	131072	87.8255	11.6	2.8
	8.0293	131072	80.5049	11.6	2.8
	8.3084	131072	85.8643	11.6	2.8
	8.6423	131072	80.5119	11.6	2.8
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	8.31066	131072	82.95334	11.62	2.8
min	7.9297	131072	80.0601	11.6	2.8
max	8.6436	131072	87.8255	11.7	2.8
std	0.3335915212	0	3.624153577	0.04472135955	0

Table 27: EXPERIMENT 3: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Thread count for fileio test	Docker (average total time)	QEMU (average total time)
1	1.2771	8.0163
5	1.31382	7.97264
10	1.64304	8.22282
15	1.84144	8.31066

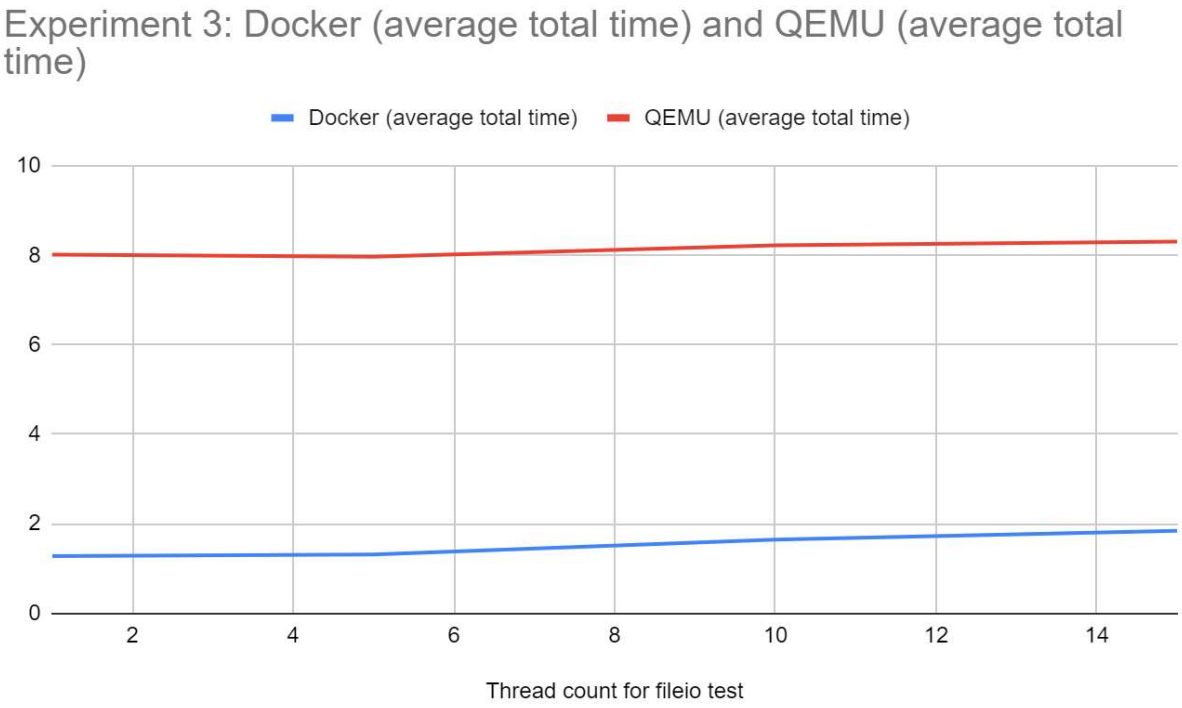

Figure 3: EXPERIMENT 3: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Experiment 4: Different file sizes for fileio reads

Docker

Table 28: testmode=fileio thread=1 file-test-mode=sequential read --file-total-size=2G

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	0.2387	131072	0.23	2	8.3789
	0.3097	327680	0.3005	2	6.4572
	0.2401	327680	0.2312	2	8.3297
	0.2426	327680	0.2334	2	8.2448
	0.2366	327680	0.2282	2	8.4535
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	0.25354	288358.4	0.24466	2	7.97282
min	0.2366	131072	0.2282	2	6.4572
max	0.3097	327680	0.3005	2	8.4535
std	0.03147003337	87925.77058	0.0312726398	0	0.8506460292

Table 29: testmode=fileio thread=1 file-test-mode=sequential read --file-total-size=4G

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	0.5823	262144	0.5621	4	6.8688
	0.6339	262144	0.6156	4	6.3098
	0.5378	262144	0.5192	4	7.4381
	0.4737	262144	0.4571	4	8.4447
	0.4741	262144	0.4579	4	8.4377
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	0.54036	262144	0.52238	4	7.49982
min	0.4737	262144	0.4571	4	6.3098
max	0.6339	262144	0.6156	4	8.4447
std	0.06955068655	0	0.06836831869	0	0.947438693

QEMU

Table 30: testmode=fileio thread=1 file-test-mode=sequential read --file-total-size=2G

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	14.0134	131072	13.9927	11.6	2.9
	14.59	131072	14.5798	11.5	2.9
	15.4251	131072	15.3995	11.5	2.9
	15.6649	131072	15.6465	11.5	2.9
	15.0232	131072	15.0014	11.5	2.9
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	14.94332	131072	14.92398	11.52	2.9
min	14.0134	131072	13.9927	11.5	2.9
max	15.6649	131072	15.6465	11.6	2.9
std	0.6612028259	0	0.6595245613	0.04472135955	0

Table 31: testmode=fileio thread=1 file-test-mode=sequential read --file-total-size=4G

	total time [s]	total number of events	latency: total time taken by event execution [s]	disk utilization [Gb]	throughput: [GB/s]
	28.5339	262144	28.4975	11.4	3
	29.5596	262144	29.5234	11.4	3
	29.1051	262144	29.0658	11.3	3
	29.29	262144	29.2542	11.3	3
	30.1006	262144	30.0625	11.3	3.1
measure	**total time [s]**	**total number of events**	**latency: total time taken by event execution [s]**	**disk utilization [Gb]**	**throughput: [GB/s]**
average	29.31784	262144	29.28068	11.34	3.02
min	28.5339	262144	28.4975	11.3	3
max	30.1006	262144	30.0625	11.4	3.1
std	0.5770087547	0	0.576715057	0.05477225575	0.04472135955

Table 32: EXPERIMENT 4: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Disk utilization [GB]	Docker (average total time)	QEMU (average total time)
2	0.25354	5.26238
4	0.54036	5.30394

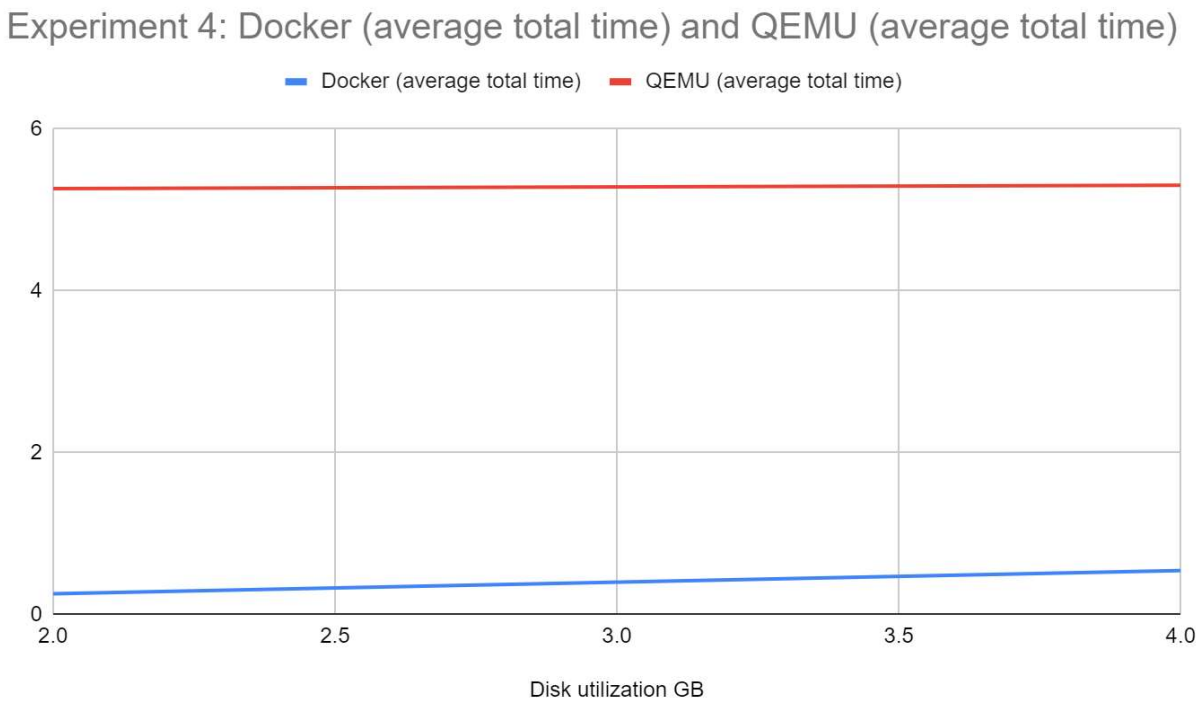

Figure 4: EXPERIMENT 4: AVERAGE TOTAL TIMES BETWEEN DOCKER AND QEMU

Experiment 5: Different RAM levels for QEMU (2046 MB and 4092 MB)

Table 33: Different RAM levels for QEMU (2046 MB and 4092 MB)

RAM [MB]	QEMU (average total time)
2046	5.26238
4092	5.30394