Pradnya Yeole(W1629375)

SYSTEM VS OS VIRTUALIZATION
Homework 1

Table of Contents

Pradnya Yeole(W1629375)

Introduction:

This report compares two virtualization approaches and covers their setup, configuration, operations, benchmarking, and detailed findings.

System Virtualization (Qemu)

OS Virtualization (Docker)

Installations steps:

The procedures for installing QEMU and Docker in this portion of the report. The installation will vary based on each person's system setup. I'll describe how did I install MacBook air with M1 processors in this section.

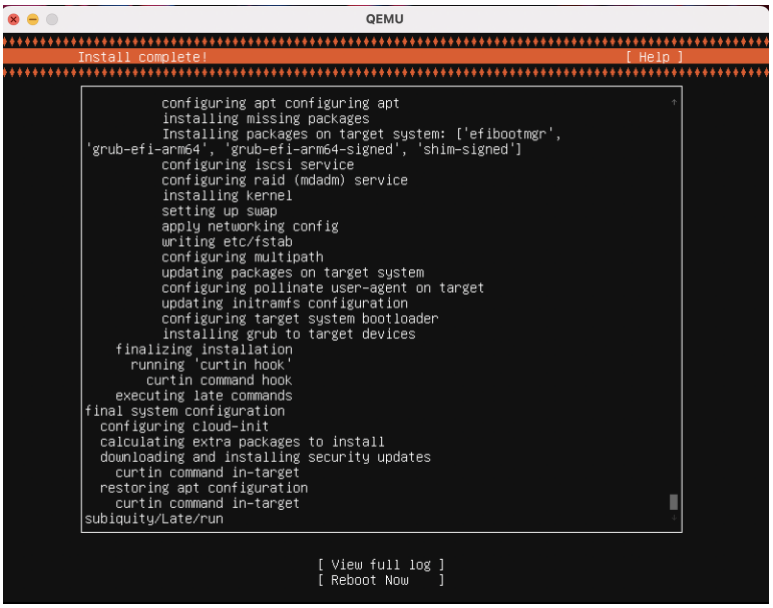
Qemu:

1) Downloaded ARM (Apple Silicon): **Ubuntu 22.04 Server for ARM**
2) Installed qemu with **brew install qemu**
3) Run qemu command with options including iso image, raw image etc.

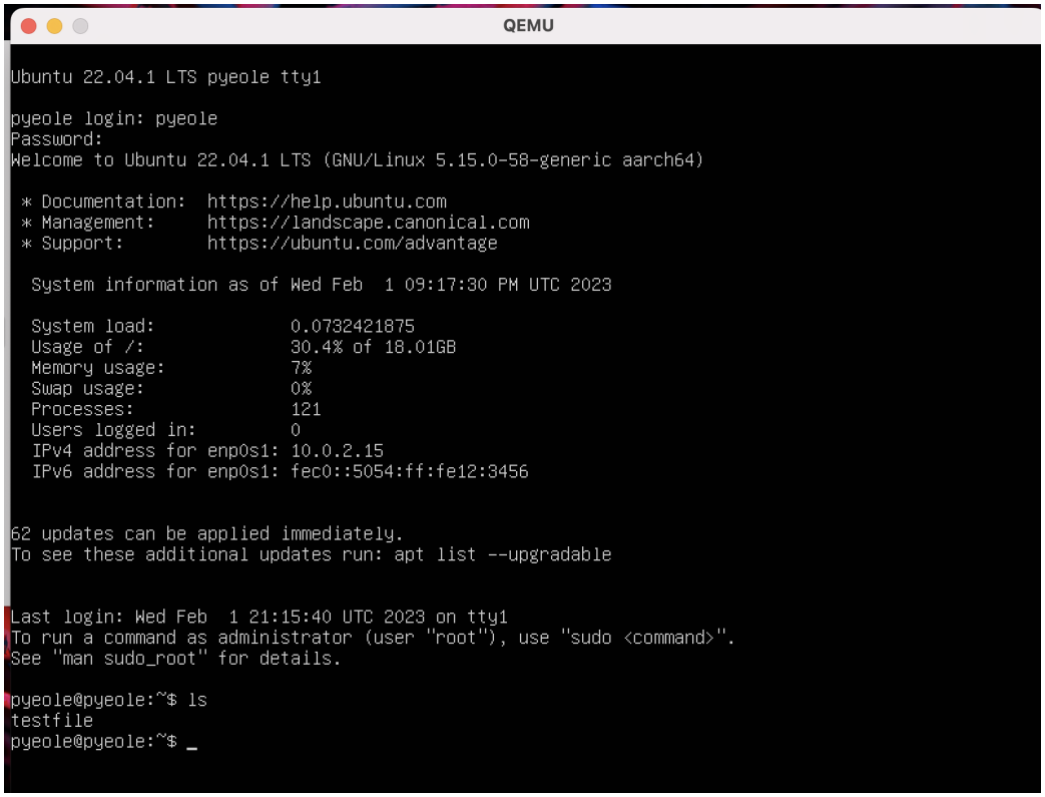
```
qemu-system-aarch64 \
-monitor stdio \
-M virt,highmem=off \
-accel hvf \
-cpu host \
-smp 4 \
-m 3000 \
-bios QEMU_EFI.fd \
-device virtio-gpu-pci \
-display default,show-cursor=on \
-device qemu-xhci \
-device usb-kbd \
-device usb-tablet \
-device intel-hda \
-device hda-duplex \
-drive file=ubuntu-latest.raw,format=raw,if=virtio,cache=writethrough
-cdrom ubuntu-22.04.1-live-server-arm64.iso
```

1. -m: Denotes memory. Since we specified 2G as the input value in the command above, we are allocating 2GB of RAM capacity for our VM.
2. Hardware acceleration is denoted by the suffix "-accel."
3. -smp: The number of cores is indicated. We have specified the value of the parameter as 2, which indicates that we have provided our VM with 2 cores.

4) It launched the Ubuntu installation process in qemu terminal.
5) Configured Ubuntu with proxy, username, password etc.



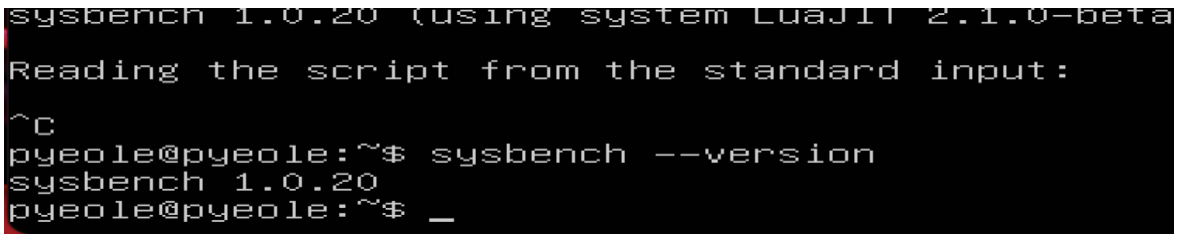
6) Using the same command with same set of options except -cdrom, qemu terminal is launched after installation



7) Then First sysbench is installed qemu ubuntu with following commands:

apt update

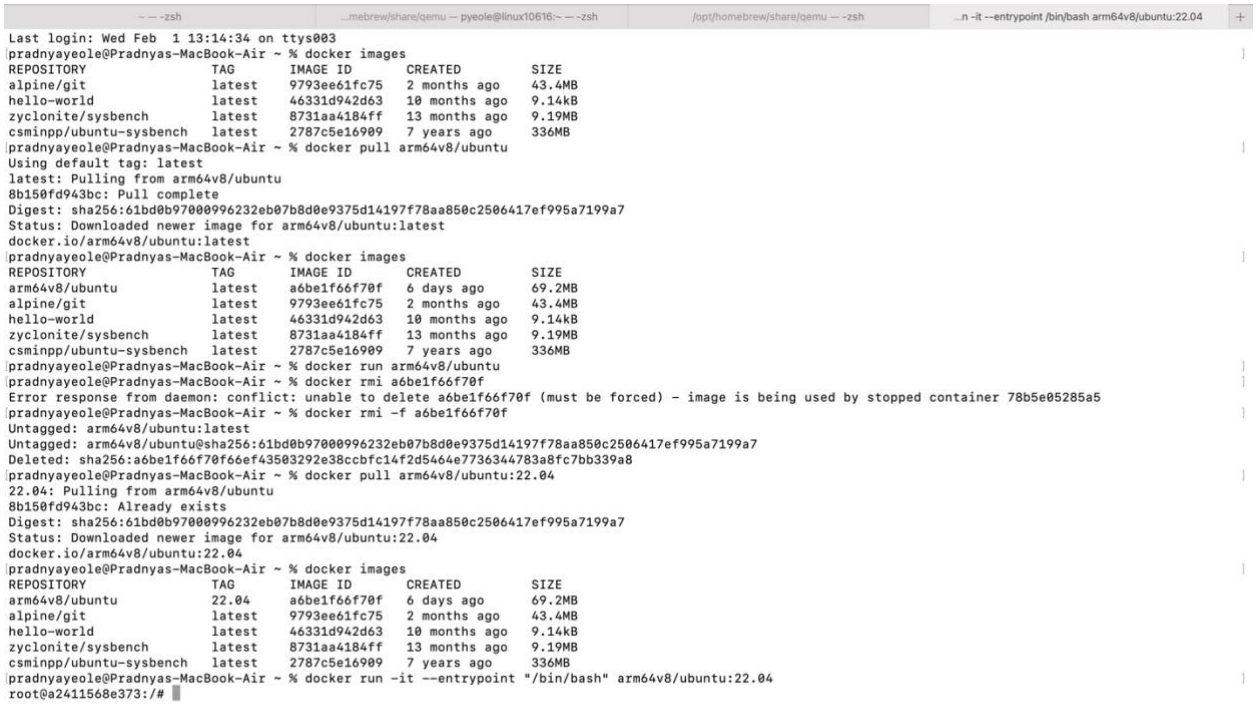
apt install sysbench



Docker:

1) Installed docker with https://docs.docker.com/desktop/mac/install/
2) Pulled the ubuntu image with the same version like qemu: i.e Ubuntu:22.04
3) Run the docker image with the following command which will start the container

   Docker run -it --entrypoint “/bin/bash” arm64v8/ubuntu:22.04



4) Installed Sysbench version in docker container with same commands.

```
Setting up libsasl2-2:arm64 (2.1.27+dfsg2-3ubuntu1.1) .
Setting up libaio1:arm64 (0.3.112-13build1) ...
Setting up libluajit-5.1-2:arm64 (2.1.0~beta3+dfsg-6) .
Setting up libldap-2.5-0:arm64 (2.5.13+dfsg-0ubuntu0.22
Setting up libpq5:arm64 (14.6-0ubuntu0.22.04.1) ...
Setting up sysbench (1.0.20+ds-2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
root@a2411568e373:/# sysbench --version
sysbench 1.0.20
root@a2411568e373:/#
```

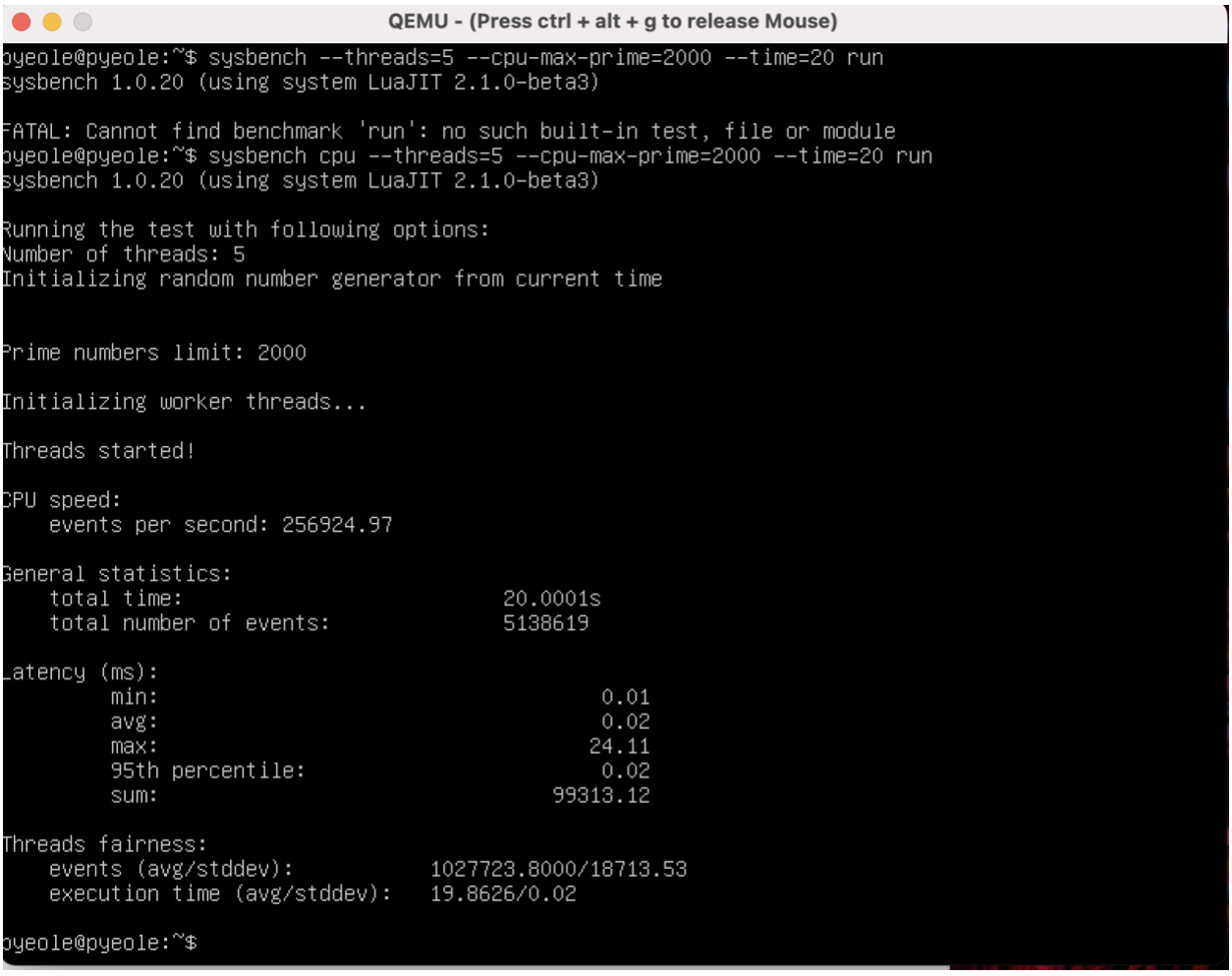
Here, the Installation process completes and now let's see how CPU and file IO tests are performed.

CPU tests:

Qemu:

Commands: sysbench –threads=5 –cpu-max-prime=2000 –time=20 run

With the same commands, different tests are performed with different sets of parameters. Example is shown in screenshot.



Bash scripts:

In the Qemu folder,

cpuTest.sh is a bash script which ran 3 test cases 5 times. And stored output in cpuTest_QEMU_op.txt output file.

Output table:

Test1:

QEMU	20000					
	Events/Sec	Min ms	Max ms	Avg ms	std	95 percentile
1	140	6.63	16.79	7.1	0	8.13
2	141	6.61	13.53	7.03	0	7.56
3	134	6.62	6.11	7.43	0	9.22
4	138	6.62	28.81	71.9	0	8.13
5	138	6.67	23.88	7.19	0	7.98

Test 2:

QEMU	50000					
	Events/Sec	min	max	avg	std	95 percentile
1	41.34	23.13	43.24	24.11	0	25.28
2	41.32	22.9	30.65	24.1	0	25.28
3	41.24	23.04	33.52	24.16	0	25.28
4	41.2	23.15	43.63	24.17	0	25.28
5	41.34	23.3	28.61	24.1	0	24.83

Test 3:

	Events/Sec	Min ms	Max ms	avgms	std	95 percentile
1	18.59	52.15	62.32	53.63	0	55.82
2	18.75	51.61	58.11	53.2	0	54.83
3	18.24	51.99	167.36	54.67	0	55.82
4	18.63	52.17	61.77	53.5	0	54.83

5	18.54	52.16	86.25	53.79	0	54.83

Docker:

In docker too, like qemu,
Commands: sysbench –threads=5 –cpu-max-prime=2000 –time=20 run

With the same commands, different tests are performed with different sets of parameters. An example is shown in the screenshot.

```
root@a2411568e373:/HW1# cd ..
root@a2411568e373:/# sysbench cpu --threads=1 --cpu-max-prime=35000 --time=10 run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 35000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  1701.69

General statistics:
    total time:                          10.0004s
    total number of events:              17020

Latency (ms):
         min:                                    0.56
         avg:                                    0.59
         max:                                   10.97
         95th percentile:                        0.61
         sum:                                 9989.72

Threads fairness:
    events (avg/stddev):           17020.0000/0.00
    execution time (avg/stddev):   9.9897/0.00

root@a2411568e373:/#
```

```
root@a2411568e373:/# sysbench cpu --threads=15 --cpu-max-prime=15000 --time=50 run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 15
Initializing random number generator from current time


Prime numbers limit: 15000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 19166.15

General statistics:
    total time:                          50.0008s
    total number of events:              958329

Latency (ms):
         min:                                    0.20
         avg:                                    0.78
         max:                                   36.38
         95th percentile:                        0.23
         sum:                               749382.65

Threads fairness:
    events (avg/stddev):           63888.6000/1337.34
    execution time (avg/stddev):   49.9588/0.02

root@a2411568e373:/#
```

Bash scripts:

In Docker folder,

cpuTest.sh is a bash script which ran 3 test cases 5 times. And stored output in cpuTest_Docker_op.txt output file.

Output table:

Test1:

Docker	20000					
	Events/Sec	min	max	avg	std	95 percentile
1	468	2.01ms	18446744073678.15ms	2.21ms	0	2.54ms
2	469	1.96ms	18446744073677.96ms	2.11ms	0	2.47ms

3	466	2.00ms	18446744073678.05ms	2.12ms	0	2.41ms
4	462	1.99ms	18446744073677.48ms	2.16ms	0	2.33ms
5	467	2.01ms	4.82ms	2.14ms	0	2.26ms

Test2:

Docker	50000					
	Events/Sec	Min ms	max	Avg ms	std	95 percentile ms
1	132	7.08	18446744073683.76ms	7.55	0	8.29
2	135	7.04	18446744073682.76ms	7.42	0	7.79
3	134	7.08	18446744073687.00ms	7.45	0	7.89
4	128	7.05	18446744073686.71ms	7.82	0	8.62
5	14	7.07	18446744073685.14ms	69.31	0	10.54

Test 3:

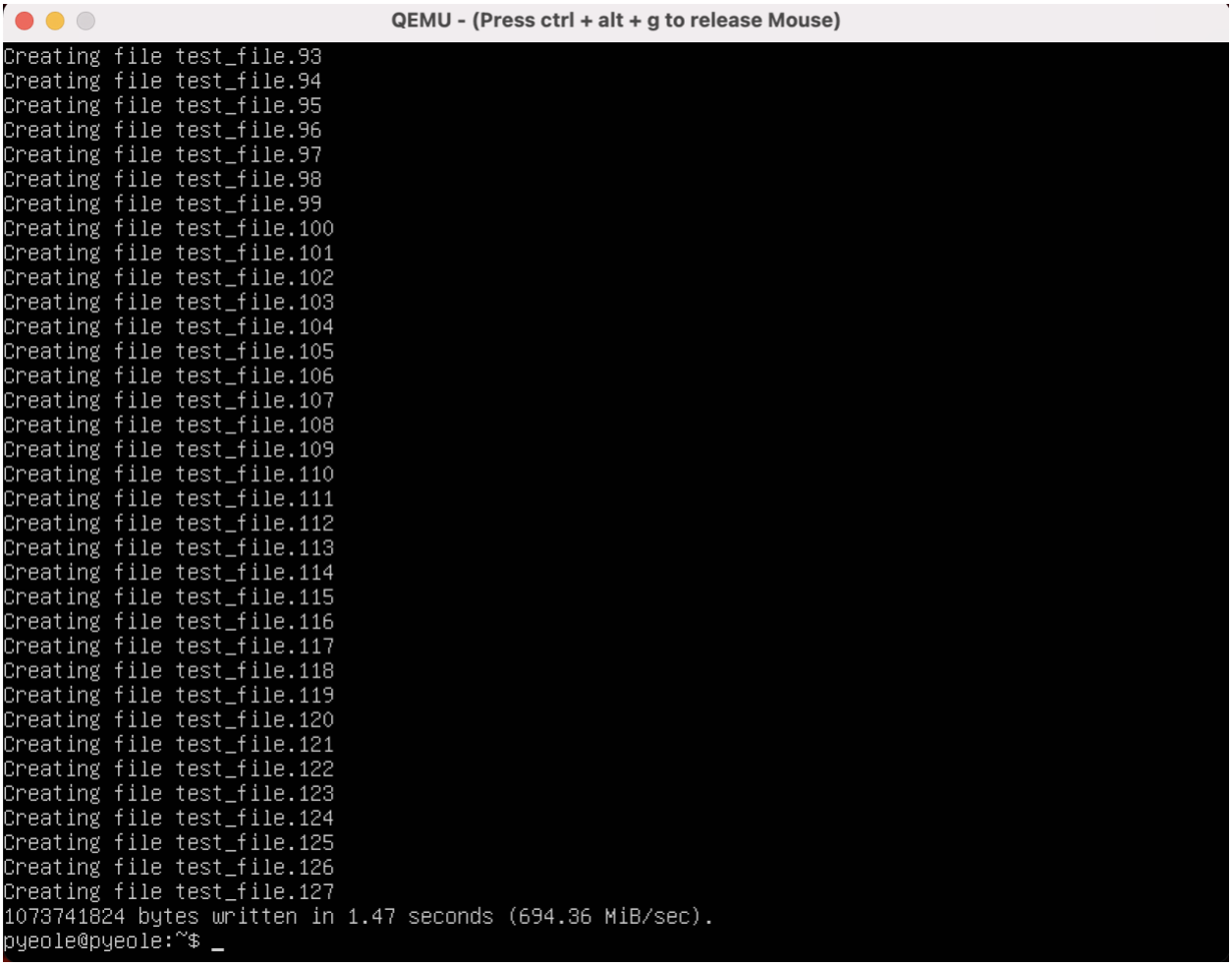
Dcoker	90000					
	Events/Sec	Min ms	Max ms	Avg ms	std	95 percentile ms
1	59	16.27	18446744073699.46ms	17.03	0	18.05
2	59	16.26	18446744073694.41ms	16.96	0	17.9
3	58	16.28	18446744073697.96ms	17.16	0	18.74
4	59	16.24	18446744073693.59ms	17.02	0	18.08
5	58	16.31	18446744073696.18ms	17.11	0	18.5

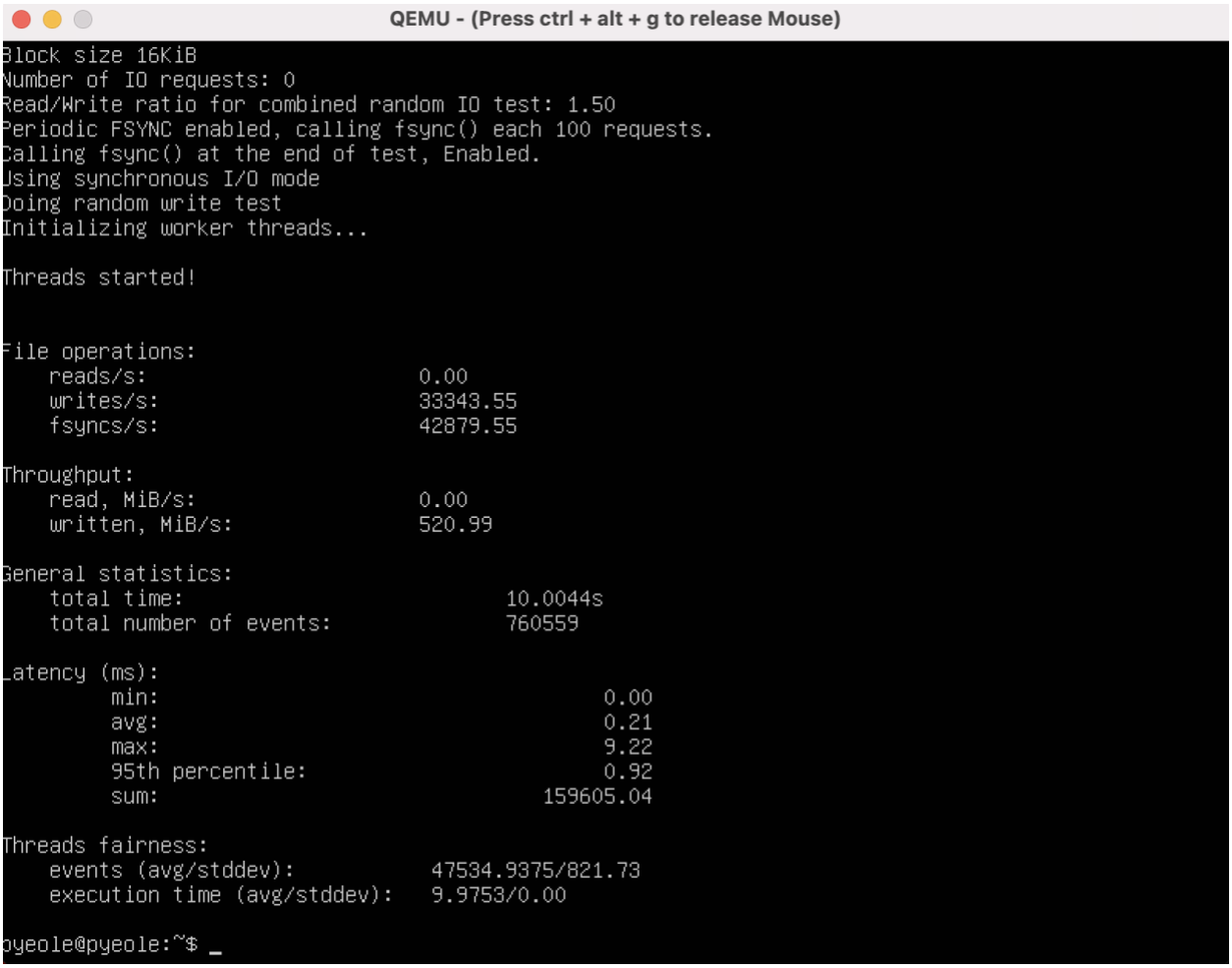
File IO tests:

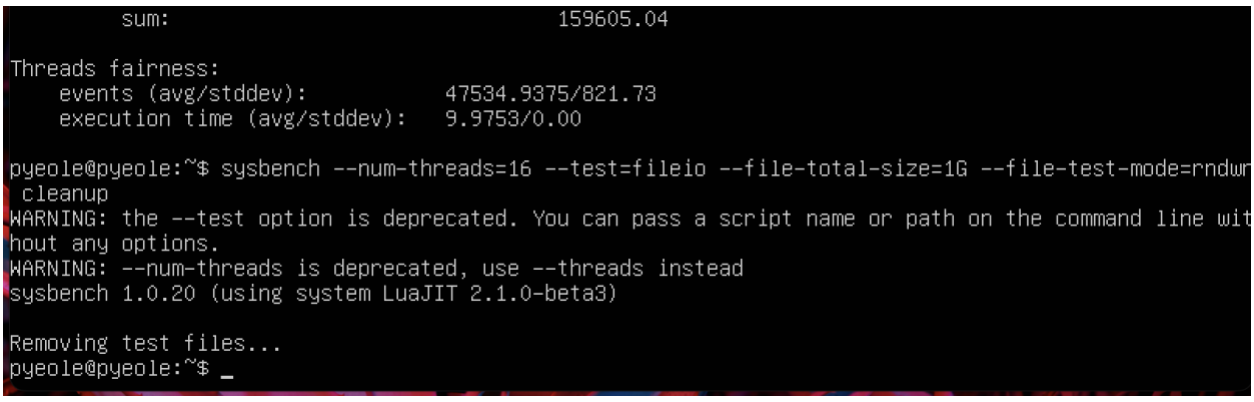
Qemu:

Commands:

sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndwr prepare
sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndwr run
sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndwr cleanup







Bash scripts:

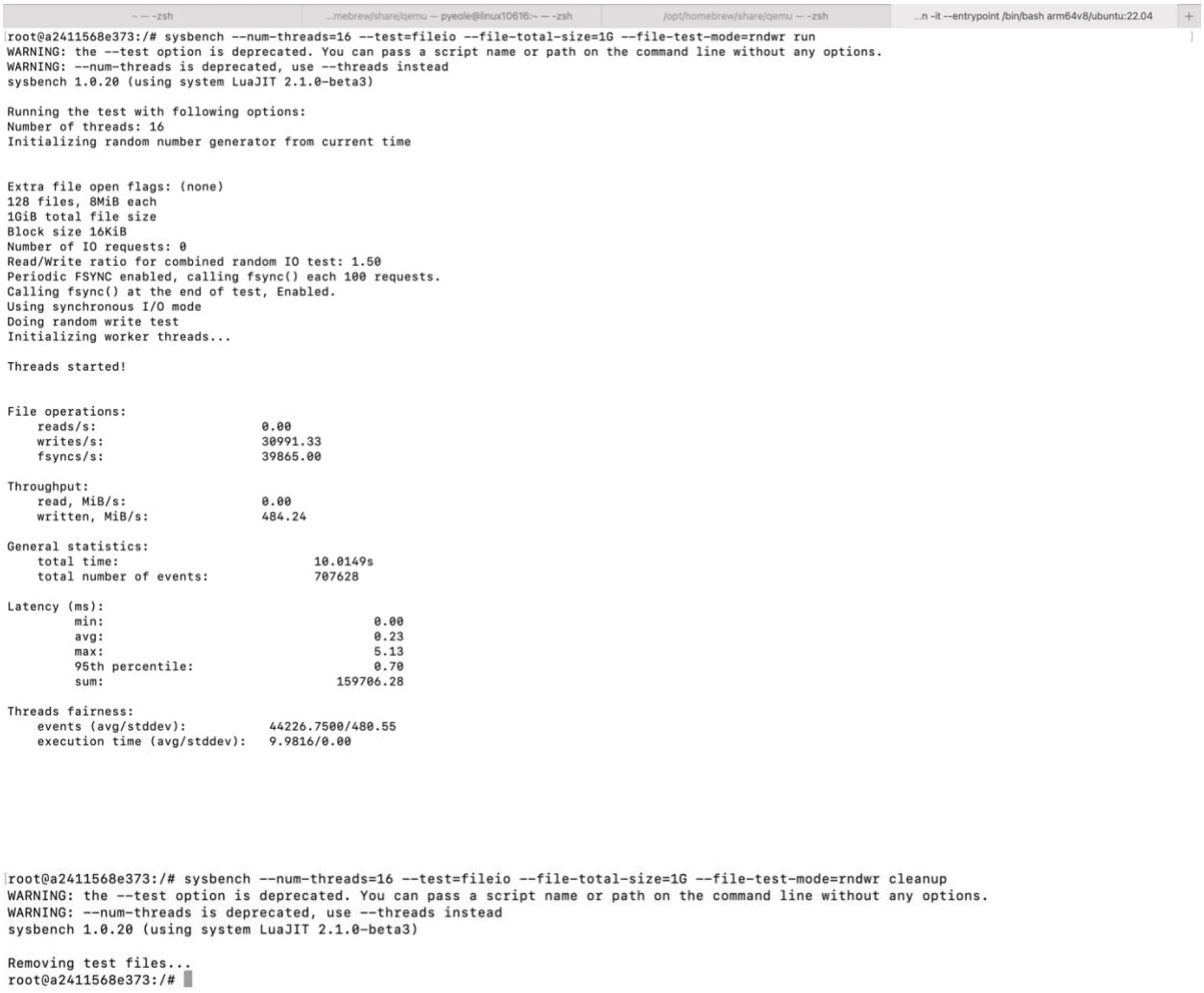
In the Qemu folder,

fileioTest.sh is a bash script that ran 3 test cases 5 times. And stored output in fileioTest_QEMU_op.txt output file.

Docker:

Commands:

sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndwr prepare
sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndwr run
sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndwr cleanup



Bash scripts:

In Docker folder,

fileioTest.sh is a bash script which ran 3 test cases 5 times. And stored output in fileioTest_Docker_op.txt output file.

Output table:

Docker						
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	72.75	4656	0.02	0.15	0.62	91.93
2	57.67	3691	0.02	0.16	22.69	72.73
3	44.683	2859.74	0.02	0.13	7.29	79.52
4	42.87	2745	0.02	0.18	26.02	104.92
5	67.12	4296.22	0.02	0.18	8.94	91.78

Conclusion:

1) The CPU events/second is higher in OS virtualization than in System Virtualization.
2) When both System and OS virtualization's maxprime limit is increased, the number of CPU events per second drops.
3) When we increase the number of threads for fileIO testing, the throughput rises.
4) According to the CPU tests and fileio tests, OS virtualization performs better than System virtualization.

Github Details:

Repository link https://github.com/pradnyayeole/COEN241_CC_Winter23_pyeole
Folder: HW1
Commit hash: b7d53e8e06d799ba5b6ccc6302e3e7b93ce0efa2