COEN 241 HW 1: System vs OS Virtualization

Xiongsheng Yi

Host Machine

My Windows desktop:

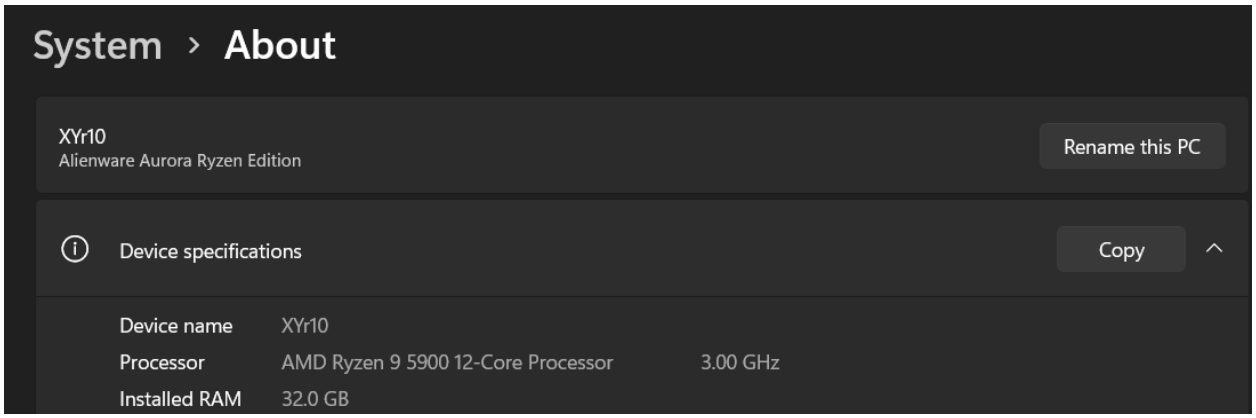
Processor: AMD Ryzen 9 5900 12-Core Processor, 3.00 GHz

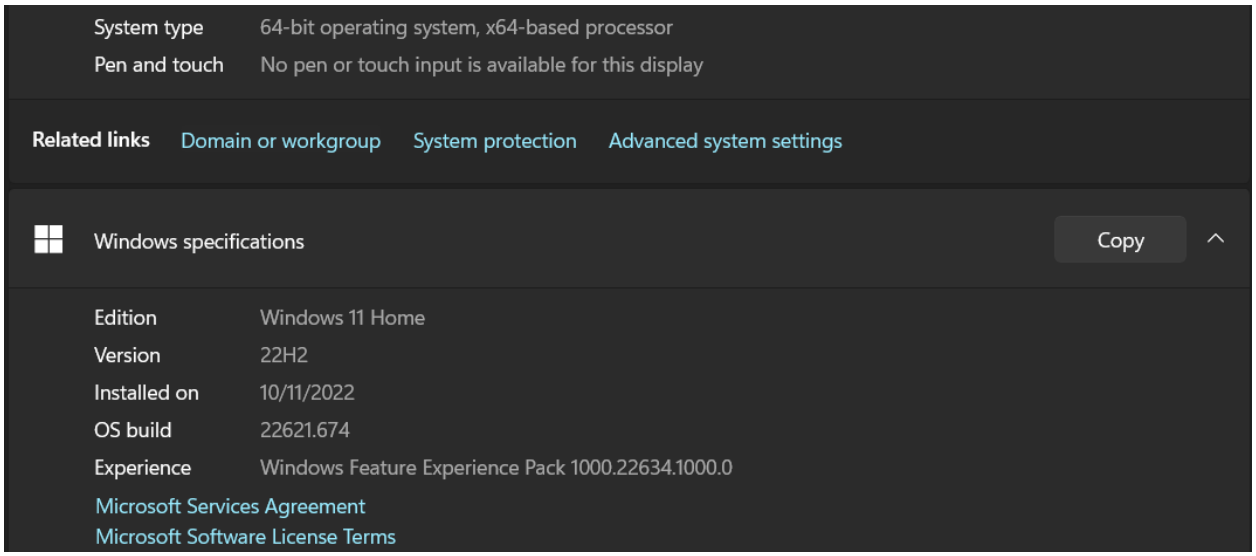
RAM: 32GB

System type: 64-bit operating system, x64-based processor

Windows edition: Windows 11 Home

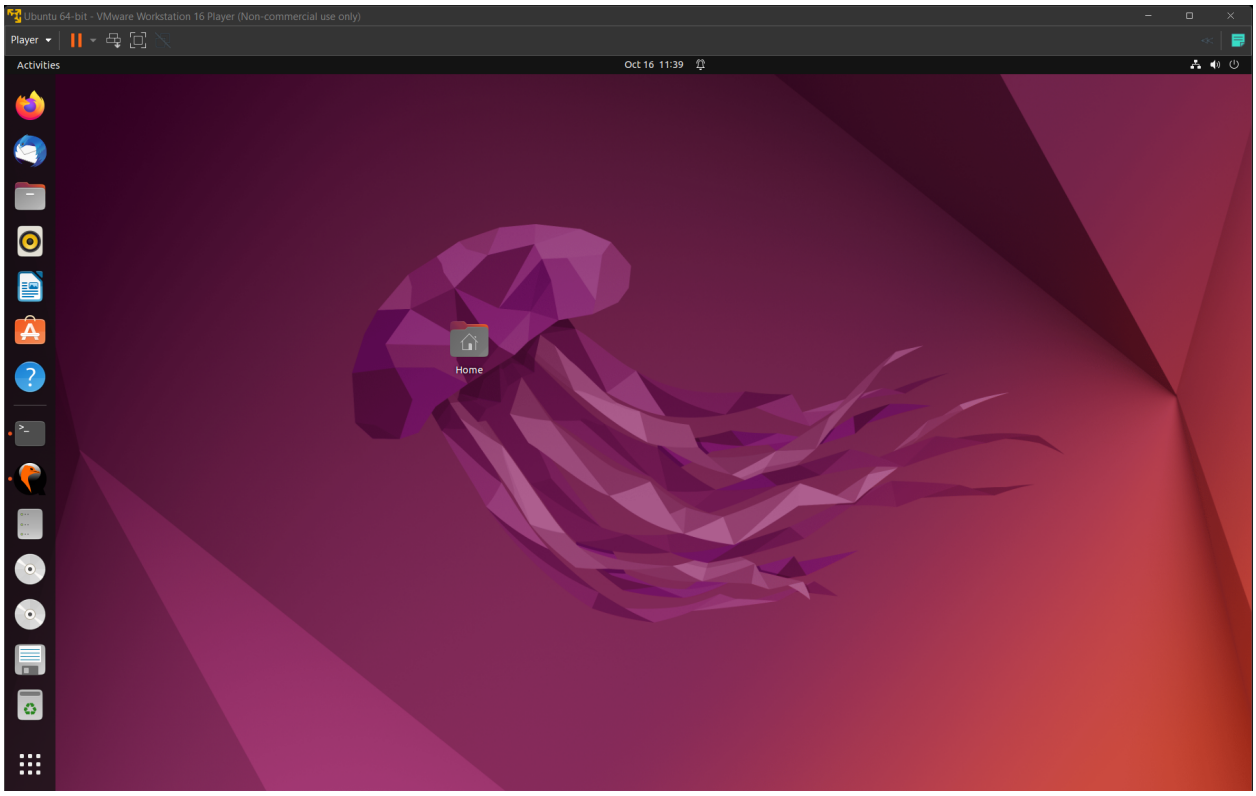
Storage: 700GB free of 936GB





Linux Environment Setup:

Since I was using Windows, I used VMware to create a Linux environment. I downloaded and installed VMware for Windows computer, and created a new Linux OS with the latest Ubuntu 22.04.



Reference Links:

https://www.vmware.com/products/workstation-player.html

https://ubuntu.com/download/desktop

QEMU Setup

1. Download the given Ubuntu server ISO image to local.

ubuntu-20.04.5-live-server-amd64.iso is downloaded. (For Linux)

2. Open the terminal and Install QEMU.

```
$ sudo apt-get install git
$ sudo apt-get install qemu
$ sudo apt-get install qemu-utils
$ sudo apt-get install qemu-system
```

3. Create a QEMU image. (10G disk space, image format of QEMU copy-on-write v2)

```
$ sudo qemu-img create ubuntu.img 10G -f qcow2
```

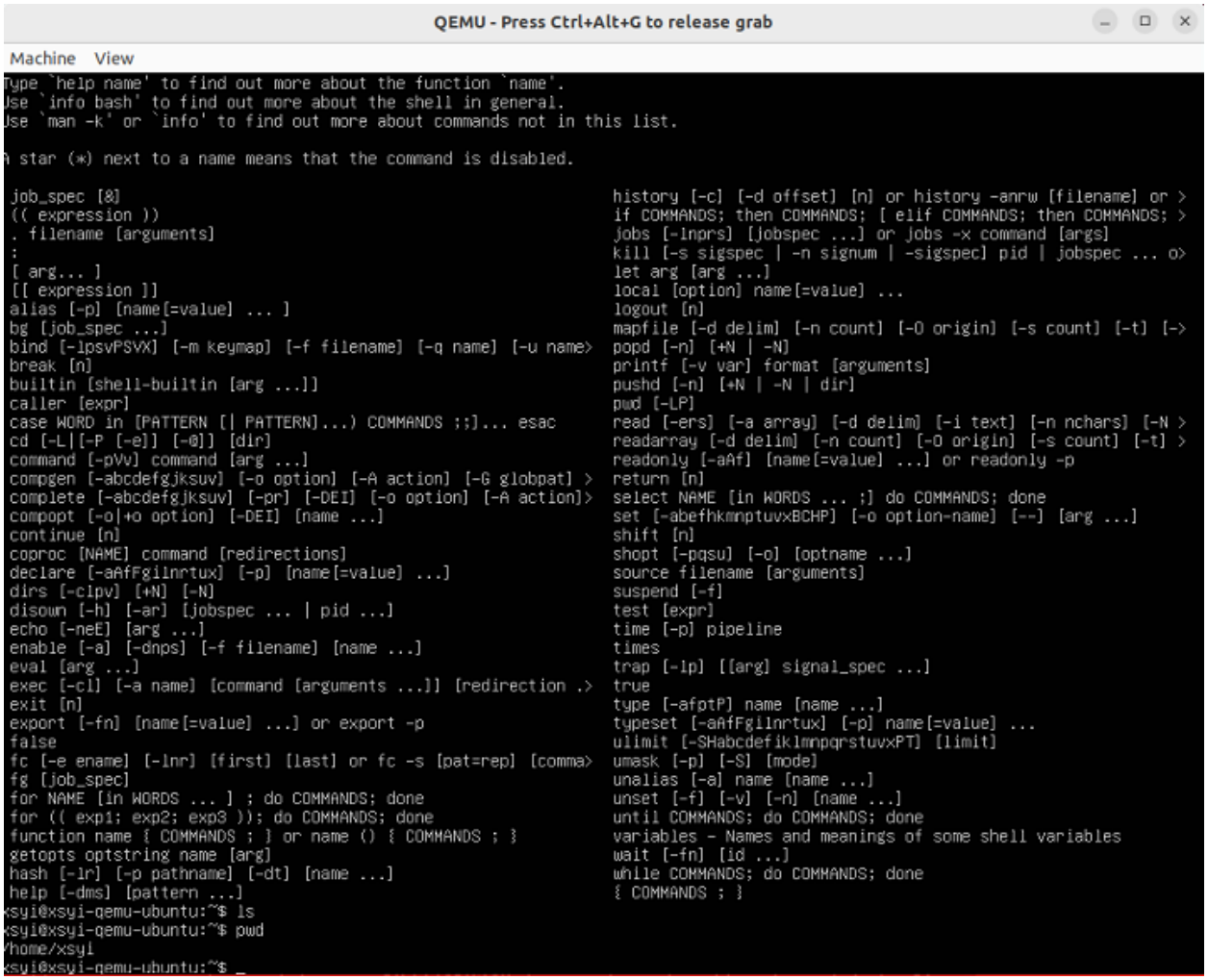
4. Then start the VM installation.

```
$ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom
./ubuntu-20.04.5-live-server-amd64.iso -m 2046 -boot strict=on
```

5. After the installation is done, boot the VM from the image ubuntu.img created.

```
$ qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -boot strict=on
```

6. Then login to the QEMU with username and password. You will see a new QEMU window.



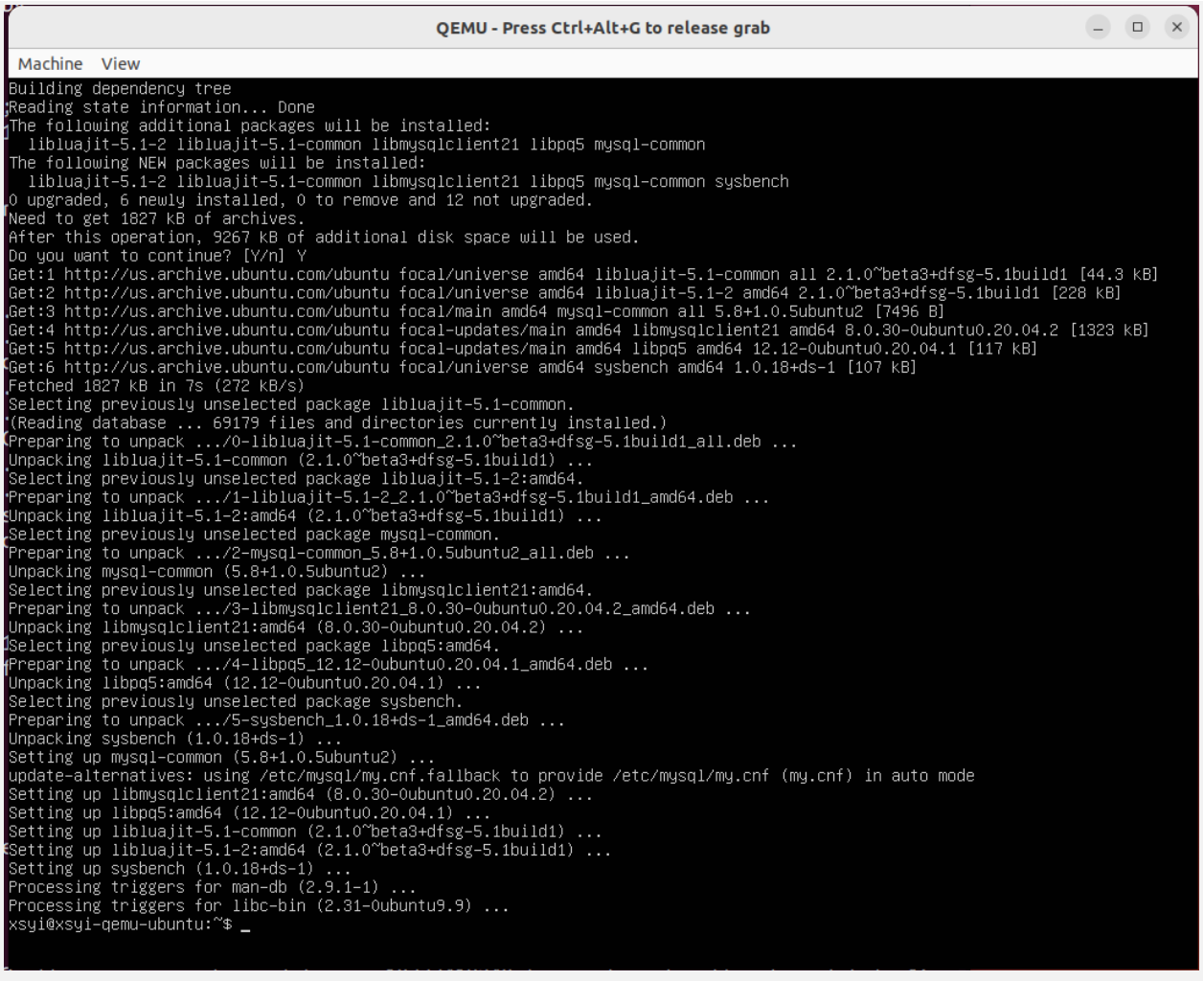
QEMU version:

```
xsyi@xsyi-qemu-ubuntu:~$ qemu-system-x86_64 -version
QEMU emulator version 4.2.1 (Debian 1:4.2-3ubuntu6.23)
Copyright (c) 2003-2019 Fabrice Bellard and the QEMU Project developers
```

7. Install the sysbench in the QEMU.

```
$ sudo apt update
```

```
$ sudo apt install sysbench
```



It is ready for testing now.

QEMU sysbench version

```
xsyi@xsyi-qemu-ubuntu:~$ sysbench --version
sysbench 1.0.18
xsyi@xsyi-qemu-ubuntu:~$ _
```

Docker container Setup

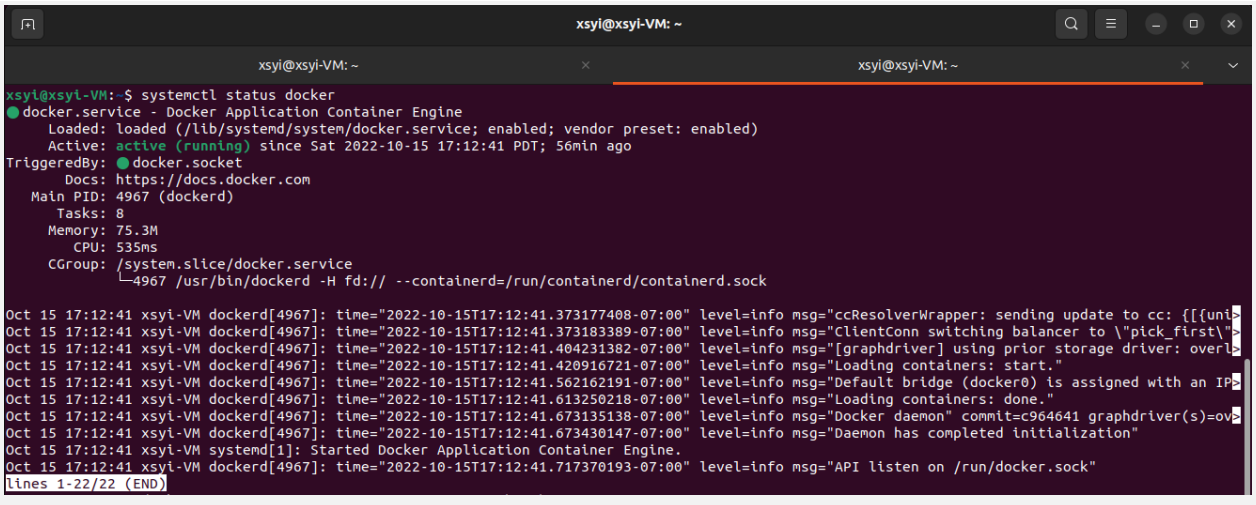
I used this tutorial to install the docker:

https://docs.docker.com/engine/install/ubuntu/

Verify that Docker is installed and running correctly

```
$ sudo systemctl start docker
```

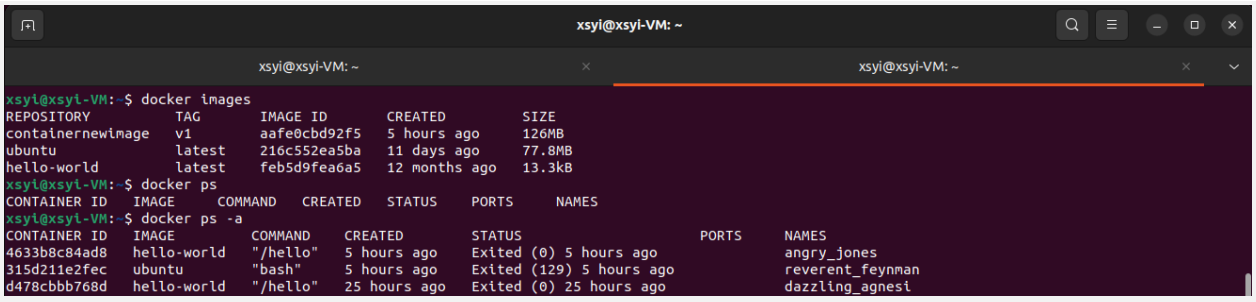
```
$ sudo systemctl status docker
```



Docker version:

Create my own image and verify image history:

```
$ docker commit -m "install sysbench" 315d211e2fec containernewimage:v1
$ docker images
$ docker ps -a
```



Commands for Docker management:

`$ docker images`	Display the list of current images in the docker.
`$ docker pull`	Pull images from a specific repository.
`$ docker ps`	Display the list of containers in the docker.
`$ docker images`	Display the list of current images in the docker.
`$ docker run`	create a container from an image.
`$ docker stop`	Stop running a specific container from the docker.
`$ docker rm`	Remove a container from the docker.

`$ docker --version` Display current version of docker.

`$ docker exec` Run a command in a running container.

`$ docker restart` Restart one or more containers.

`$ docker kill` Kill one or more containers.

`$ docker commit` Create a new image from the container image.

`$ docker push` Push an image or repository to a registry.

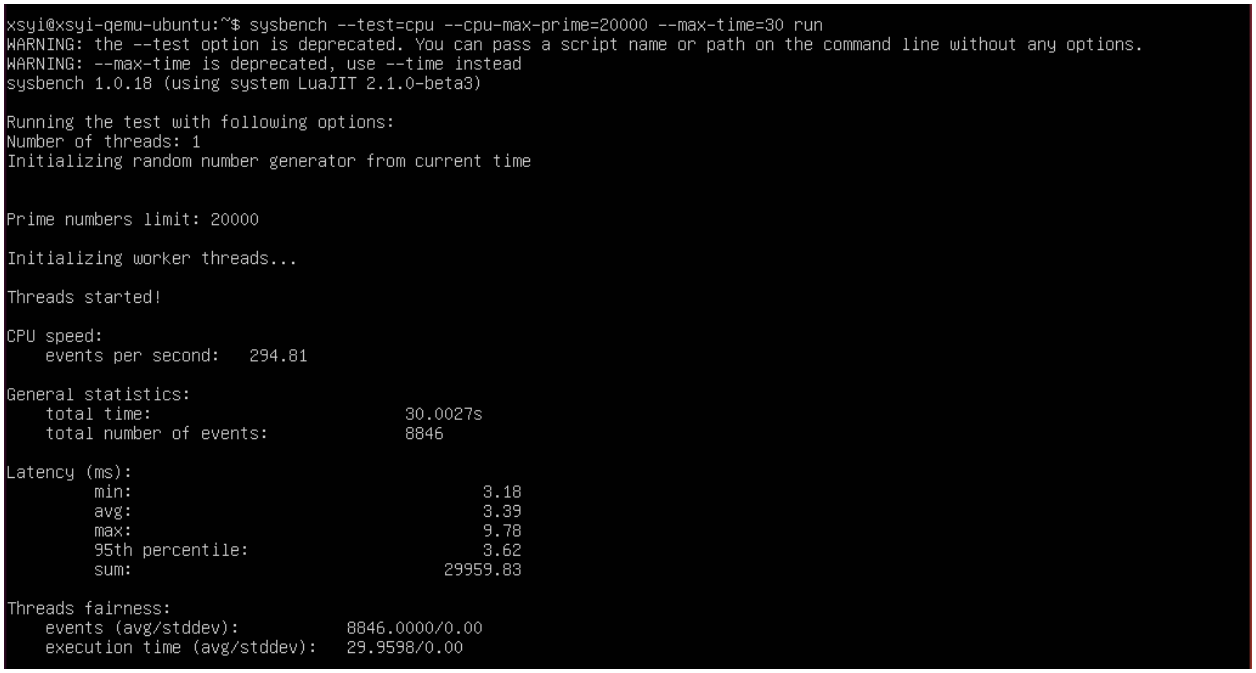
`ctl + p + q` Go back to terminal from running container, but keep container running

Proof of experiment

Try the first CPU performance comparison

```
$ sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
```

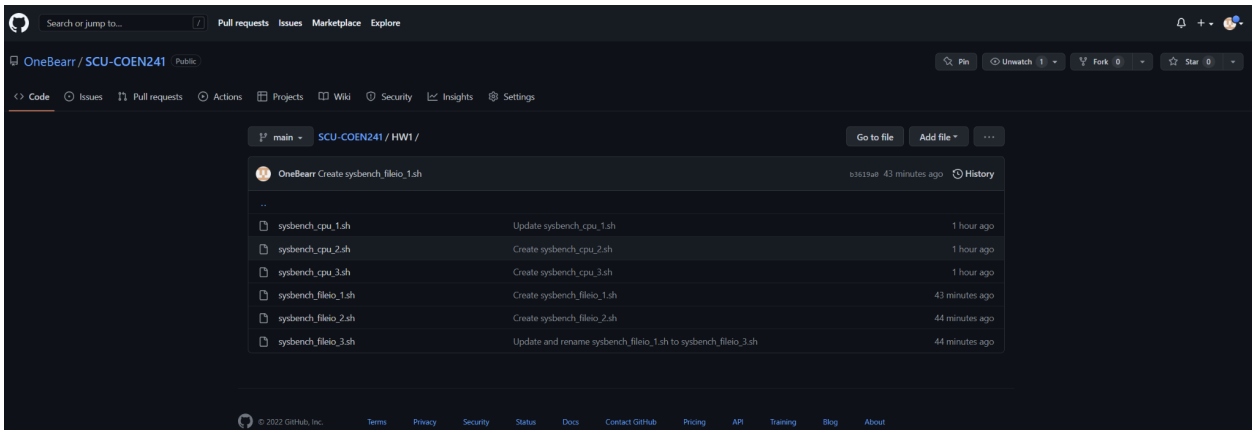
For QEMU:



For docker:

```
xsyi@xsyi-VM:~$ docker run -it --cpus=2 -m 2G zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-6ef8a4d4d7 (using bundled LuaJIT 2.1.0-beta2)

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


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  1969.34

General statistics:
    total time:                          30.0000s
    total number of events:              59081

Latency (ms):
         min:                                    0.50
         avg:                                    0.51
         max:                                    2.98
         95th percentile:                        0.54
         sum:                                29988.94

Threads fairness:
    events (avg/stddev):           59081.0000/0.00
    execution time (avg/stddev):   29.9889/0.00
```

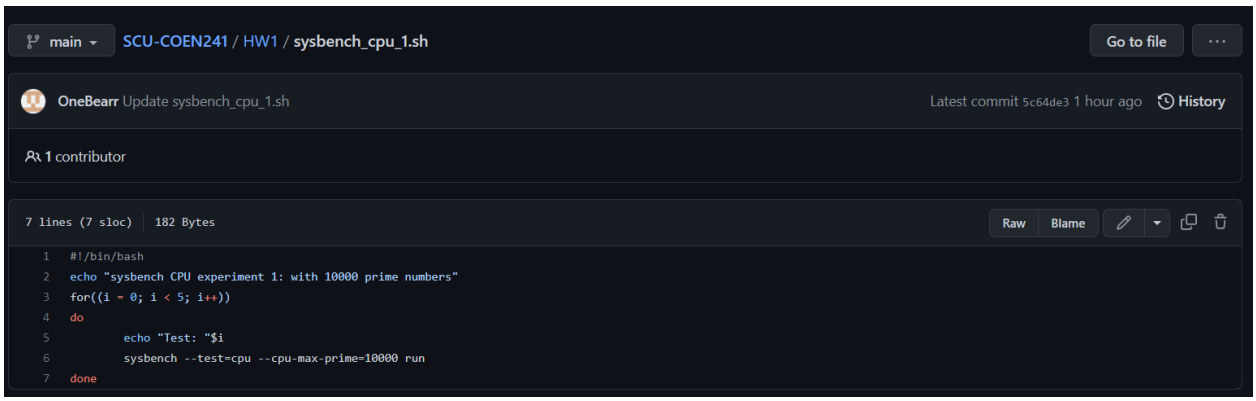
All the test cases shell scripts are saved in the Github repository



For CPU test cases:

I tested “cpu max prime” 10000, 20000, 30000

Each shell script is like: (repeat 5 times)

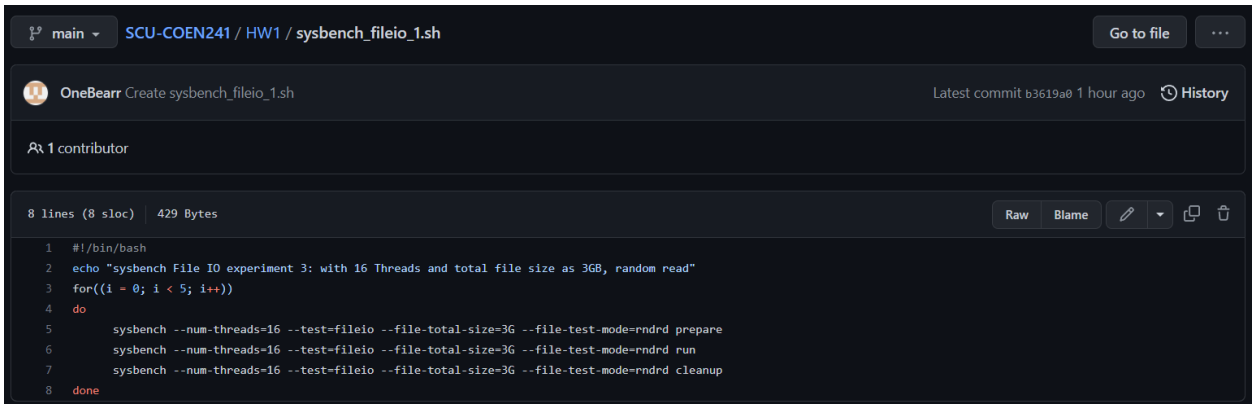


For File I/O test cases:

I tested with 16 Threads and total file size as 3GB,

“random read”, “random write” and “combined random read/write”

Each shell script is like:(repeat 5 times)



The previous test result will speed up the following test from the cache, which is unfair and inaccurate.

Thus, prepare -> run -> cleanup

For QEMU:

Installations:

$ apt update

$ apt install sysbench

$ apt install git

$ apt install vim

Use “git clone https://github.com/OneBearr/SCU-COEN241.git” to pull the repository to the local VM.

Give the files permission to be executed by

$ chmod 755 file_name.sh

Run each shell script and save results by

$ ./file_name.sh > output_file_name.txt

To open and read the output files by

$ vim output_file_name.txt

After running all the test case files,

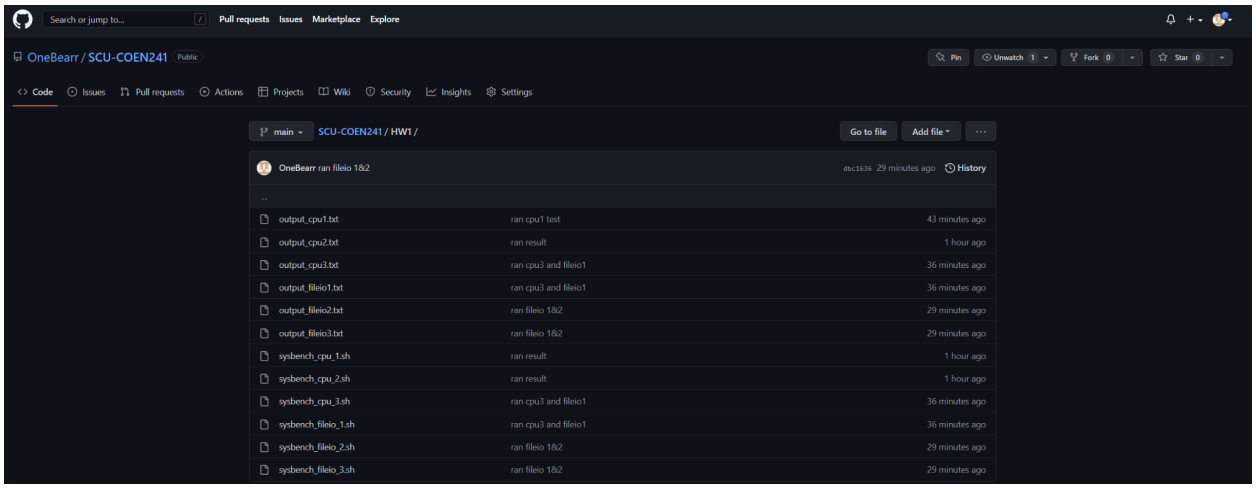
```
xsyi@xsyi-qemu2:~/SCU-COEN241/HW1$ ls
output_cpu1.txt  output_cpu3.txt     output_fileio2.txt  sysbench_cpu_1.sh  sysbench_cpu_3.sh     sysbench_fileio_2.sh
output_cpu2.txt  output_fileio1.txt  output_fileio3.txt  sysbench_cpu_2.sh  sysbench_fileio_1.sh  sysbench_fileio_3.sh
xsyi@xsyi-qemu2:~/SCU-COEN241/HW1$ _
```

After all the Github authorization setup(ssh, etc.), I pushed all the output files to my github,

$ git add

$ git commit

$ git push origin main



Ref: https://github.com/OneBearr/SCU-COEN241/tree/main/HW1

For cpu1 with 10000 prime: (Average of 5 tests)

CPU speed:

events per second: 750

General statistics:

total time: 10s

total number of events: 7500

For cpu2 with 20000 prime: (Average of 5 tests)

CPU speed:

events per second: 295

General statistics:

total time: 10s

total number of events: 2950

For cpu3 with 30000 prime: (Average of 5 tests)

CPU speed:

events per second: 172

General statistics:

total time: 10s
total number of events: 1720

Analysis:

CPU1 has the highest CPU speed with most events per sec, CPU3 is the lowest.

For fileio1 with 16 Threads and total file size as 1GB, random read: (Average of 5 tests)

Throughput:
read, MiB/s: 1250
written, MiB/s: 0.00

General statistics:
total time: 10s
total number of events: 800000

Threads fairness:
events (avg/stddev): 49078.3125/585.55
execution time (avg/stddev): 9.2867/0.17

For fileio2 with 16 Threads and total file size as 1GB, random write: (Average of 5 tests)

Throughput:
read, MiB/s: 0.00
written, MiB/s: 22.98

General statistics:
total time: 10s
total number of events: 35000

Threads fairness:
events (avg/stddev): 2189.3750/272.67
execution time (avg/stddev): 9.9626/0.02

For fileio3 with 16 Threads and total file size as 1GB, combined random read/write: (Average of 5 tests)

Throughput:
read, MiB/s: 20
written, MiB/s: 13

General statistics:
total time: 10s
total number of events: 50000

Threads fairness:
events (avg/stddev): 3038.0625/274.21

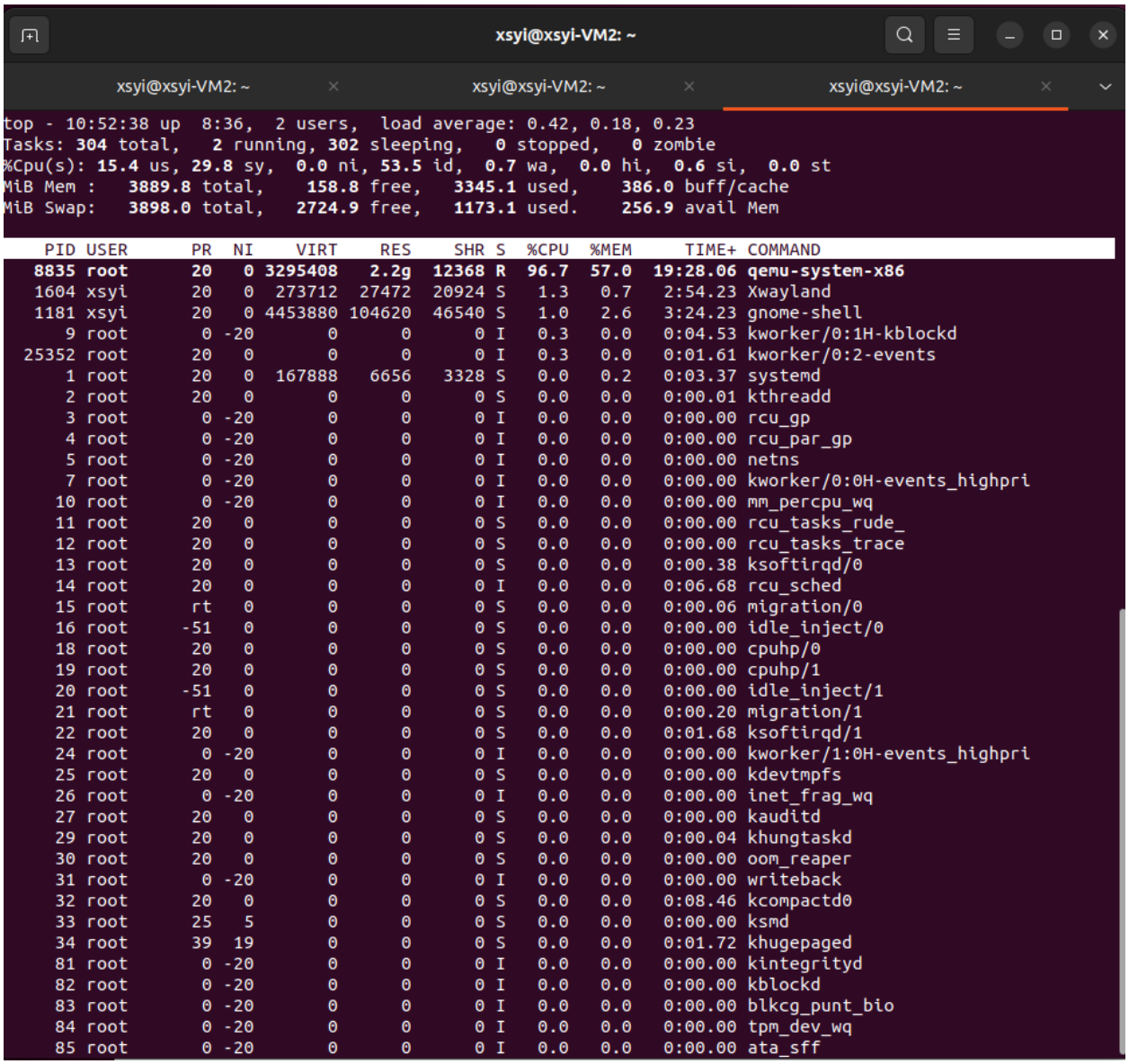
execution time (avg/stddev): 9.9558/0.02

Analysis:

File read is much faster than file write relatively, the throughput is about 50 times faster.

The CPU utilization when running tests:

About 96-97%



For Docker:

Docker preparation,

$ docker start

$ docker pull ubuntu

$ docker run -ti ubuntu

Installations:

apt update

apt install sysbench

apt install git

apt install vim

Use “git clone https://github.com/OneBearr/SCU-COEN241.git” to pull the repository to the local VM.

Give the files permission to be executed by

$ chmod +x *.sh

Run each shell script and save results by

$ ./file_name.sh > output_file_name.txt

To open and read the output files by

$ vim output_file_name.txt

After running all the test case files,

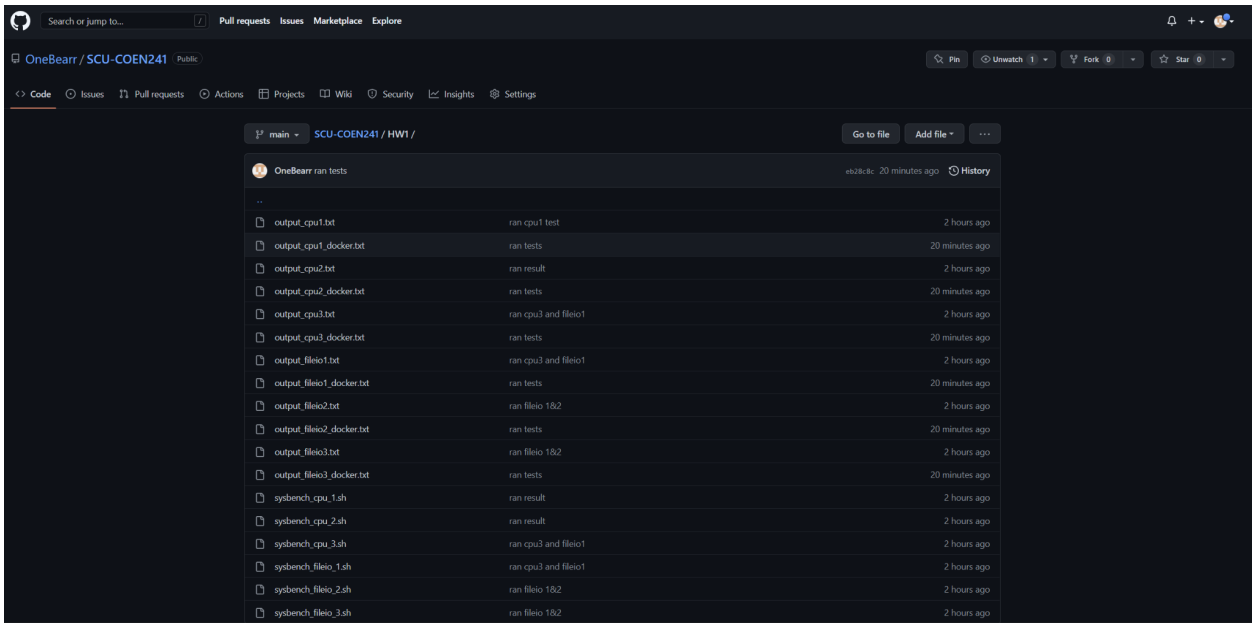
```
root@eafa0fdcb229:/SCU-COEN241/HW1# ls
output_cpu1.txt         output_cpu2_docker.txt  output_fileio1.txt         output_fileio2_docker.txt  sysbench_cpu_1.sh  sysbench_fileio_1.sh
output_cpu1_docker.txt  output_cpu3.txt         output_fileio1_docker.txt  output_fileio3.txt         sysbench_cpu_2.sh  sysbench_fileio_2.sh
output_cpu2.txt         output_cpu3_docker.txt  output_fileio2.txt         output_fileio3_docker.txt  sysbench_cpu_3.sh  sysbench_fileio_3.sh
```

After all the Github authorization setup(ssh, etc.), I pushed all the output files to my github,

$ git add

$ git commit

$ git push origin main



Ref: https://github.com/OneBearr/SCU-COEN241/tree/main/HW1

For cpu1 with 10000 prime: (Average of 5 tests)

CPU speed:

events per second: 5300

General statistics:

total time: 10s

total number of events: 53000

For cpu2 with 20000 prime: (Average of 5 tests)

CPU speed:

events per second: 2080

General statistics:

total time: 10s

total number of events: 20800

For cpu3 with 30000 prime: (Average of 5 tests)

CPU speed:

events per second: 1200

General statistics:

total time: 10s

total number of events: 12000

Analysis:

CPU1 has the highest CPU speed with most events per sec, CPU3 is the lowest.

For fileio1 with 16 Threads and total file size as 1GB, random read: (Average of 5 tests)

Throughput:

read, MiB/s:	15000
written, MiB/s:	0.00

General statistics:

total time:	10s
total number of events:	10200000

Threads fairness:

events (avg/stddev):	616737.9375/23978.66
execution time (avg/stddev):	9.3833/0.15

For fileio2 with 16 Threads and total file size as 1GB, random write: (Average of 5 tests)

Throughput:

read, MiB/s:	0.00
written, MiB/s:	315.90

General statistics:

total time:	10s
total number of events:	461015

Threads fairness:

events (avg/stddev):	28813.4375/2757.16
execution time (avg/stddev):	9.9626/0.02

For fileio3 with 16 Threads and total file size as 1GB, combined random read/write: (Average of 5 tests)

Throughput:

read, MiB/s:	424.99
written, MiB/s:	283.33

General statistics:

total time:	10s
total number of events:	1033712

Threads fairness:

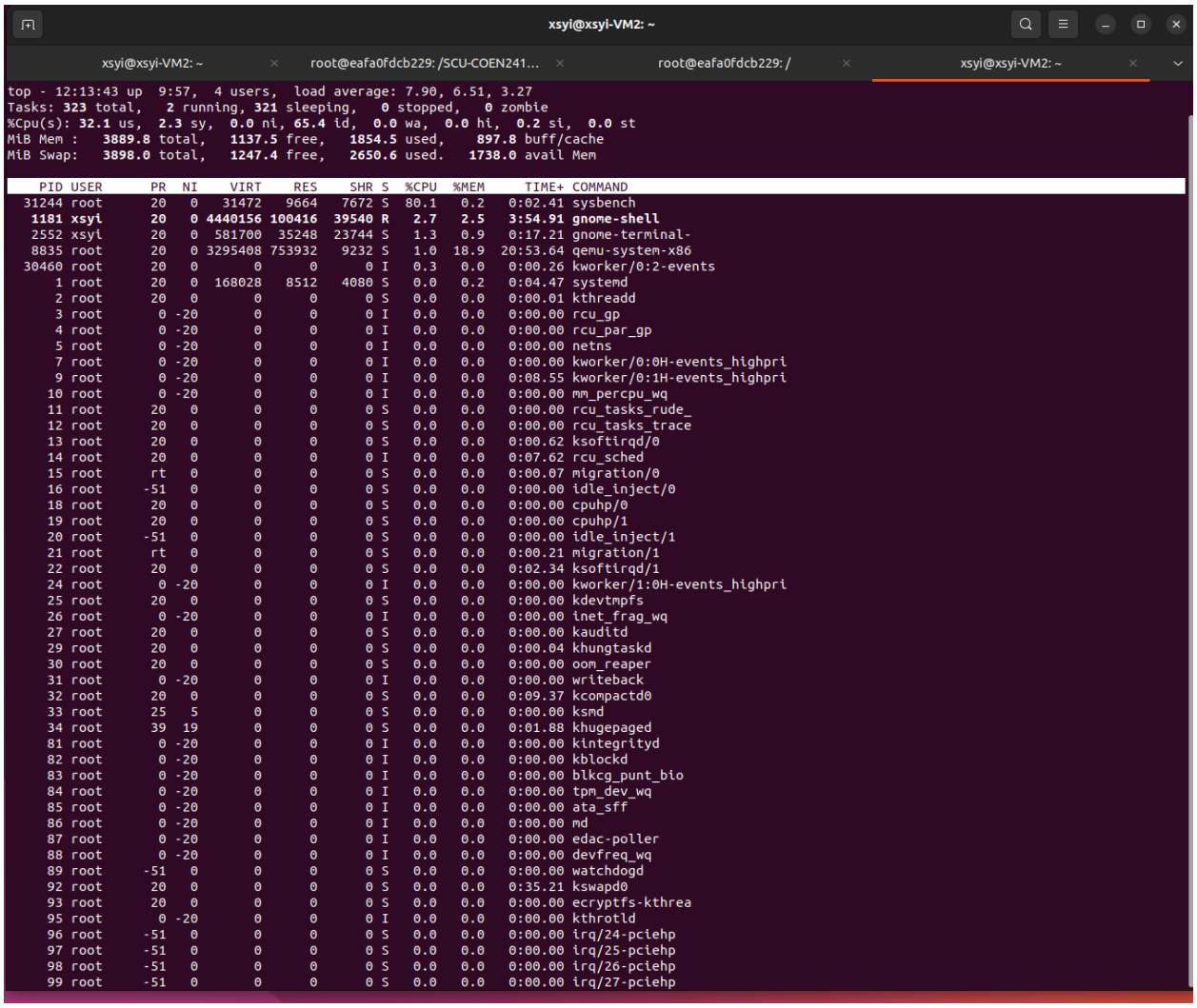
events (avg/stddev):	64607.0000/6649.11
execution time (avg/stddev):	9.9558/0.02

Analysis:

File read is much faster than file write relatively, the throughput is about 50 times faster.

The CPU utilization when running tests:

80%-90% varies.



QEMU vs Docker container Performance Comparisons

Apparently, docker containers are much faster than the QEMU.

For cpu tests,
The events per second is 7-8 times faster.

For fileio tests,

The throughputs are about 12-15 times faster in MiB/s.

Git Repository:
https://github.com/OneBearr/SCU-COEN241