

Coen 241-HW-1-Report

Xiao Yang

xyang12@scu.edu

My device information



QEMU Setup:

1. Download Ubuntu 20.04 server ISO image from given link.

2. Install QEMU

\$ brew install qemu for installing qemu on my local device.

3. Create QEMU image

\$ sudo qemu-img create ubuntu.img 10G -f qcow2 for create a 10G image.

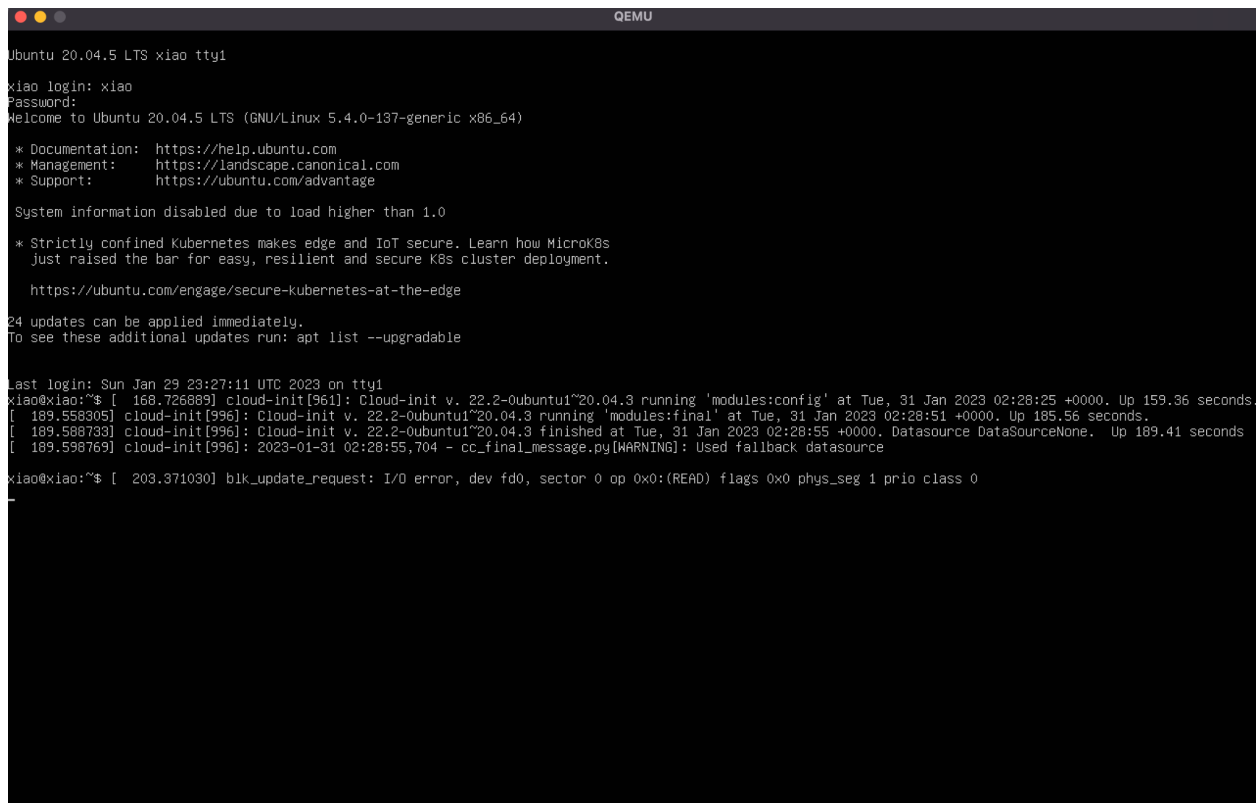
4. Install VM

\$ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom

/Users/fossil/Downloads/ubuntu-20.04.5-live-server-amd64.iso -m 2046 -boot strict=on for install the VM, (all use default settings and configurations).

4. Boot the new VM from ubuntu.img

\$ sudo qemu-system-x86_64 -m 2046 -hda ubuntu.img and then initiate the vm by instructions. Here is the part of screenshot for opening the new VM



```

QEMU
Ubuntu 20.04.5 LTS xiao tty1
xiao login: xiao
Password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-137-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

24 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Last login: Sun Jan 29 23:27:11 UTC 2023 on tty1
xiao@xiao:~$ [ 168.726889] cloud-init[961]: Cloud-init v. 22.2-0ubuntu1~20.04.3 running 'modules:config' at Tue, 31 Jan 2023 02:28:25 +0000. Up 159.36 seconds.
[ 189.558305] cloud-init[996]: Cloud-init v. 22.2-0ubuntu1~20.04.3 running 'modules:final' at Tue, 31 Jan 2023 02:28:51 +0000. Up 185.56 seconds.
[ 189.588733] cloud-init[996]: Cloud-init v. 22.2-0ubuntu1~20.04.3 finished at Tue, 31 Jan 2023 02:28:55 +0000. Datasource DataSourceNone. Up 189.41 seconds
[ 189.598769] cloud-init[996]: 2023-01-31 02:28:55,704 - cc_final_message.py[WARNING]: Used fallback datasource
xiao@xiao:~$ [ 203.971030] blk_update_request: I/O error, dev fd0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 0

```

4. Install sysbench on qemu image

\$ sudo apt update

```
$ sudo apt install sysbench
```

The version of sysbench is: 1.0.18

```
xiao@xiao:~$ sysbench --version
sysbench 1.0.18
xiao@xiao:~$ _
```

Docker Setup:

I had docker on my laptop so skip for installation.

1. Verify if Docker is running correctly

```
$ docker --version
```

3. Create and list Docker containers and images

```
$ docker images
```

```
$ docker ps -a
```

Based on my device and the version of ubuntu server, I choose amd64/ubuntu:22.04 version

Image ID is : 6b7dfa7e8fdb18ad425dd965a1049d984f31cf0ad57fa6d5377cca355e65f03

<input type="checkbox"/>	amd64/ubuntu 6b7dfa7e8fdb 	22.04	In use	about 2 months ago	77.79 MB			
--------------------------	--	-------	------------------------	--------------------	----------	---	---	---

```
[(base) fossils-MacBook-Pro:~ fossil$ docker --version
Docker version 20.10.22, build 3a2c30b
[(base) fossils-MacBook-Pro:~ fossil$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
amd64/ubuntu         22.04        6b7dfa7e8fdb      7 weeks ago      77.8MB
warter_prd           latest       45e01b046ca1     19 months ago    2.49GB
docker/getting-started latest       083d7564d904     19 months ago    28MB
hello-world          latest       d1165f221234     23 months ago    13.3kB
[(base) fossils-MacBook-Pro:~ fossil$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(base) fossils-MacBook-Pro:~ fossil$
```

3. Create ubuntu container and run

\$ docker run -it

6b7dfa7e8fdb18ad425dd965a1049d984f31cf0ad57fa6d5377cca355e65f03 bash

4. Install sysbench on docker image

Firstly need to install sudo because of lightweight:

\$ apt update

\$ apt install sudo

And then install sysbench as following:

\$ sudo apt update

\$ sudo apt install sysbench

```
[(base) fossils-MacBook-Pro:~ fossil$ docker run -it 6b7dfa7e8fdb18ad425dd965a1049d984f31cf0ad57fa6d5377cca355e65f03 bash
[root@c507c1dc8d22:/# sysbench --version
sysbench 1.0.20
root@c507c1dc8d22:/#
```

5. Some other useful docker commands

\$ docker pull Pull a image from a repo

\$ docker push Push a local image to hub

\$ docker exec Run commands in a running container

\$ docker stop Stop a running container

\$ docker rm Remove a container from docker

Experiment Setup:

1. CPU Performance

\$ sysbench --test=cpu --cpu-max-prime=10000 --max-time=30 run

Basic command for testing CPU performance is like above, I choose cpu-max-prime as 10000, 20000, 30000 respectively and integrate it as a shell file.

eg:cpu-test1.sh

```
1  echo "sysbench CPU test with 10000 cpu-max-prime :"
```

```
2  for((i = 0; i < 5; i++))
```

```
3  do
```

```
4      echo "Test: "$i
```

```
5      sysbench --test=cpu --cpu-max-prime=10000 run
```

```
6  done
```

2. I/O Performance

\$ sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrd

run

Basic command for testing IO performance is like above, I choose num-thread as 16 and total file size as 1G. Particularly for the fileio test, after one test, the operating system will cache the accessed files. If we don't manually drop such cache, the following tests will finish much faster,

as most I/O data will be served directly from the 1 memory cache instead of disks. So I add prepare and cleanup command for fair testing.

eg:io-test1.sh(random read operation)

```
echo "sysbench IO experiment with 16 Threads and total file size as 1GB, random read"
for((i = 0; i < 5; i++))
do
    sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrd prepare
    sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrd run
    sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrd cleanup
done
```

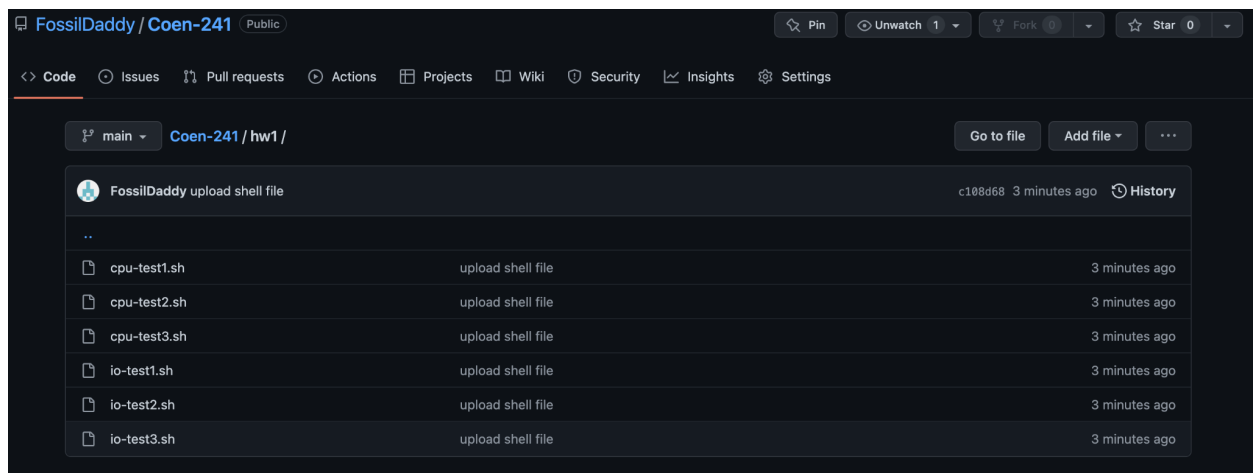
3. Upload shell file and setup on VM

\$ git add [file_name]

\$ git commit -m [msg]

\$ git push

After creating those shell file locally and all github configurations, I upload them to github for two VM to run.



On both qemu and docker VM, install git and clone this repo from github.

\$ git clone [repo-url]

\$ git init

```
xiao@xiao:~$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3.8).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 28 not upgraded.

xiao@xiao:~$
xiao@xiao:~$ git clone https://github.com/FossilDaddy/Coen-241.git
Cloning into 'Coen-241'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 29 (delta 5), reused 16 (delta 4), pack-reused 0
Unpacking objects: 100% (29/29), 4.53 KiB | 23.00 KiB/s, done.
xiao@xiao:~$ ls
Coen-241
xiao@xiao:~$ cd Coen-241
xiao@xiao:~/Coen-241$ git init
Reinitialized existing Git repository in /home/xiao/Coen-241/.git/
xiao@xiao:~/Coen-241$ _
```

4. Run shell files on VM and upload output files

Run .sh file and save output result

\$ chmod 755 [file_name] Give permission to shell file

\$./cpu-test1.sh > output-qemu-cpu-test1.txt Run shell file and save the result as file

\$ vim output-qemu-cpu-test1.txt Edit and read output file

```
General statistics:
  total time:                10.0049s
  total number of events:    3319

Latency (ms):
  min:                        2.51
  avg:                        2.99
  max:                        52.80
  95th percentile:          3.75
  sum:                        9935.73

Threads fairness:
  events (avg/stddev):       3319.0000/0.00
  execution time (avg/stddev): 9.9357/0.00

Test: 4
sysbench 1.0.18 (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: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second:    346.14

General statistics:
  total time:                10.0018s
  total number of events:    3464

Latency (ms):
  min:                        2.51
  avg:                        2.87
  max:                        18.23
  95th percentile:          3.43
  sum:                        9937.08

Threads fairness:
  events (avg/stddev):       3464.0000/0.00
  execution time (avg/stddev): 9.9371/0.00
```


Upload those testing result to github again from docker and qemu respectively both for cpu-test and io-test.

All output files upload to github again for analysis at last.

FossilDaddy qemu result upload			395c8f4 2 minutes ago History
..			
cpu-test1.sh	x		9 minutes ago
cpu-test2.sh	x		9 minutes ago
cpu-test3.sh	x		9 minutes ago
io-test1.sh	x		9 minutes ago
io-test2.sh	x		9 minutes ago
io-test3.sh	x		9 minutes ago
output-docker-cpu-test1.txt	x		9 minutes ago
output-docker-cpu-test2.txt	x		9 minutes ago
output-docker-cpu-test3.txt	x		9 minutes ago
output-docker-io-test1.txt	x		9 minutes ago
output-docker-io-test2.txt	x		9 minutes ago
output-docker-io-test3.txt	x		9 minutes ago
output-qemu-cpu-test1.txt	qemu result upload		2 minutes ago
output-qemu-cpu-test2.txt	qemu result upload		2 minutes ago
output-qemu-cpu-test3.txt	qemu result upload		2 minutes ago
output-qemu-io-test1.txt	qemu result upload		2 minutes ago
output-qemu-io-test2.txt	qemu result upload		2 minutes ago
output-qemu-io-test3.txt	qemu result upload		2 minutes ago

Result display and analysis:

1. For Qemu

CPU-Performance result(Average for 5 tests inside):

	cpu-max-prime = 10000	cpu-max-prime = 20000	cpu-max-prime = 30000
Events per second	750	295	172
Total Time	10s	10	10

Total numbers of events	7500	2950	1720
--------------------------------	------	------	------

Analysis: CPU1 has highest CPU speed while CPU3 has lowest one, which means that the cpu speed will tend to lower down with the increase of cpu-max-prime.

IO-Performance result(Average for 5 tests inside):

	Read	Write	Read/Write
Throughput	1250MB/s read	22.98MB/s write	20MB/s read 13MB/s write
Total Time	10s	10	10
Total numbers of events	800000	35000	50000
Events(avg/stddev)	49078 / 585	2189 / 273	3038 / 274
Execution time	9.29 / 0.17	9.96 / 0.02	9.96 / 0.02

Analysis: Read operation is much more faster than write operation.

Use **\$ htop** command to show cpu utilization table for all processes, these shell files(cpu testing and IO testing) were ran with 95-97% cpu-utilization.

2. For Docker

CPU-Performance result(Average for 5 tests inside):

	cpu-max-prime = 10000	cpu-max-prime = 20000	cpu-max-prime = 30000
Events per second	1248	484	276
Total Time	10s	10	10
Total numbers of events	12480	4840	2760

Analysis: CPU1 has highest CPU speed while CPU3 has lowest one, which means that the cpu speed will tend to lower down with the increase of cpu-max-prime.

IO-Performance result(Average for 5 tests inside):

	Read	Write	Read/Write
Throughput	14978MB/s read	314MB/s write	425MB/s read 280MB/s write
Total Time	10s	10	10
Total numbers of events	10200000	461000	1033720
Events(avg/stddev)	616737 / 23978	28813 / 2757	64607 / 6649
Execution time	9.38 / 0.15	9.96 / 0.02	9.96 / 0.02

Analysis: Read operation is much more faster than write operation.

Use **\$ htop** command to show cpu utilization table for all processes, these shell files(cpu testing and IO testing) were ran with 80-90% cpu-utilization.

3. Compare Qemu with Docker

Apparently, docker containers are much faster than Qemu for both type of testing.

For cpu performance testing, the cpu speed for same arguments of docker is about 2 times faster.

For IO testing, the throughput of docker is about 12-15 times faster.

Git Repo:

<https://github.com/FossilDaddy/Coen-241>