COEN 241 Cloud Computing

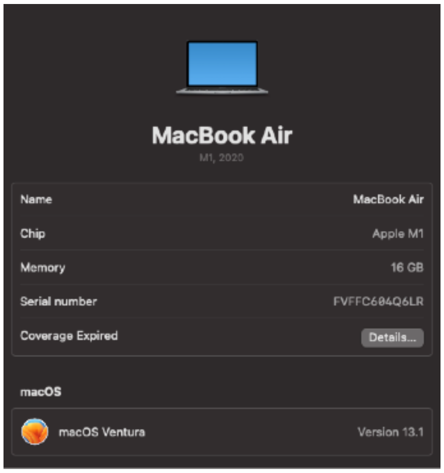
HW 1 Report

Student Name: Yu Bi

Student ID: W1628995

Git URL: https://github.com/biu425/COEN241_HW.git

Detailed configurations of experimental setup



MacBook Air

M1, 2020

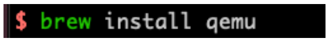
Name	MacBook Air
Chip	Apple M1
Memory	16 GB
Serial number	FVFFC684Q6LR
Coverage Expired	Details...

macOS

macOS Ventura Version 13.1

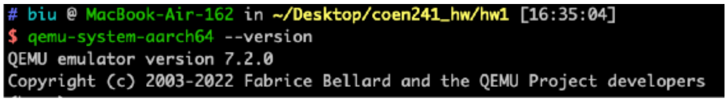
QEMU VM Installation

1. Use HomeBrew to install QEMU.



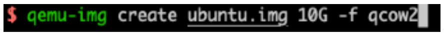
```
$ brew install qemu
```

2. Check QEMU version.

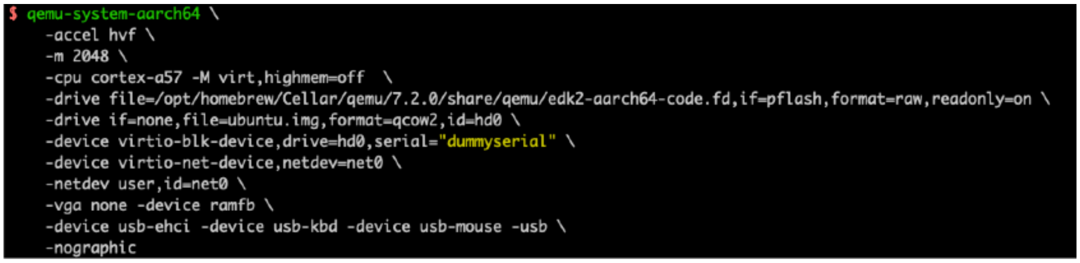


```
# biu @ MacBook-Air-162 in ~/Desktop/coen241_hw/hw1 [16:35:04]
$ qemu-system-aarch64 --version
QEMU emulator version 7.2.0
Copyright (c) 2003-2022 Fabrice Bellard and the QEMU Project developers
```

3. Create QEMU image and install VM.

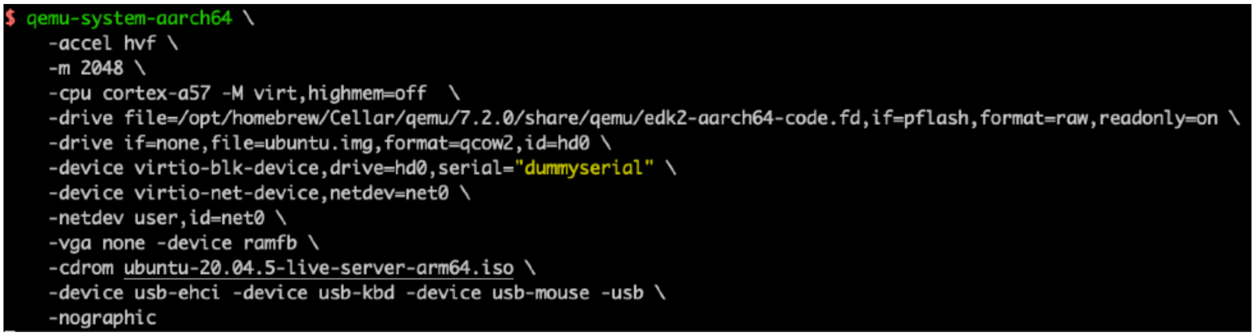


```
$ qemu-img create ubuntu.img 10G -f qcow2
```

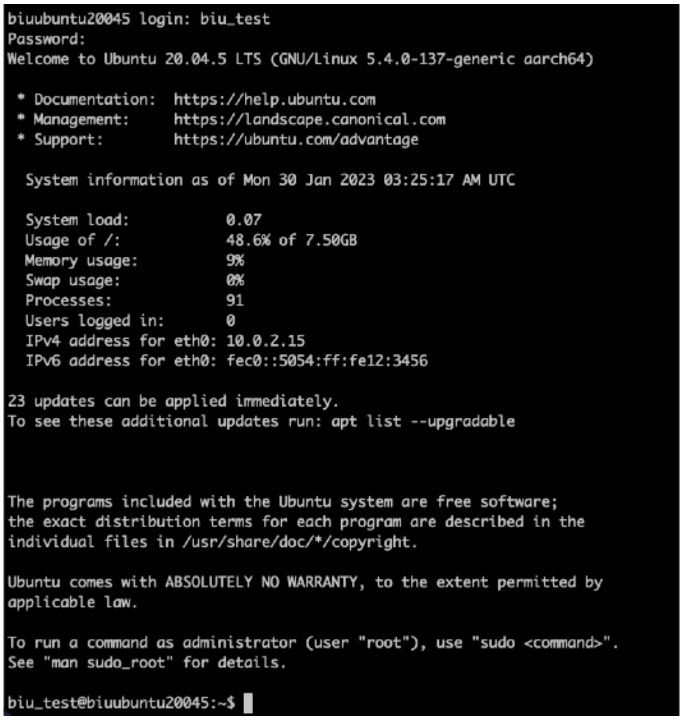


```
$ qemu-system-aarch64 \
    -accel hvf \
    -m 2048 \
    -cpu cortex-a57 -M virt,highmem=off  \
    -drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
    -drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
    -device virtio-blk-device,drive=hd0,serial="dummyserial" \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0 \
    -vga none -device ramfb \
    -device usb-ehci -device usb-kbd -device usb-mouse -usb \
    -nographic
```

4. Complete installation by following the instructions displayed.
5. Run Ubuntu Server without -cdrom param and login.



```
$ qemu-system-aarch64 \
    -accel hvf \
    -m 2048 \
    -cpu cortex-a57 -M virt,highmem=off  \
    -drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
    -drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
    -device virtio-blk-device,drive=hd0,serial="dummyserial" \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0 \
    -vga none -device ramfb \
    -cdrom ubuntu-20.04.5-live-server-arm64.iso \
    -device usb-ehci -device usb-kbd -device usb-mouse -usb \
    -nographic
```



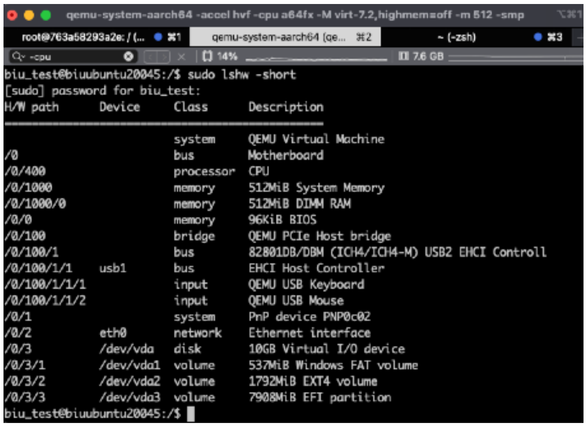
```
biuubuntu20045 login: biu_test
Password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-137-generic aarch64)

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

  System information as of Mon 30 Jan 2023 03:25:17 AM UTC

  System load:           0.07
  Usage of /:            48.6% of 7.50GB
  Memory usage:          9%
  Swap usage:            0%
  Processes:             91
  Users logged in:       0
  IPv4 address for eth0: 10.0.2.15
  IPv6 address for eth0: fec0::5054:ff:fe12:3456

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



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

biu_test@biuubuntu20045:~$ 
```

Test different arguments of QEMU

1. -m [size=]megs[,slots=n,maxmem=size] Sets guest startup RAM size to megs megabytes.
2. -M Select the emulated machine by name.
3. -cdrom file Use file as CD-ROM image
4. -smp [cpus=]n[,cores=cores][,threads=threads][,dies=dies][,sockets=sockets][,maxcpus=maxcpus] Simulate an SMP system with n CPUs.
5. -cpu model Select CPU model.
6. -accel name[,prop=value[,…]] This is used to enable an accelerator.
7. -vga type Select type of VGA card to emulate.
8. -netdev user,id=id[,option][,option][,…] Configure user mode host network backend which requires no administrator privilege to run.
9. -nographic Run QEMU as a simple command line application.

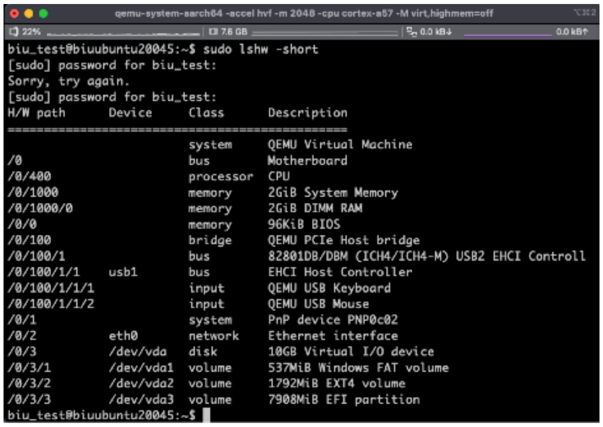
I tried to change some of arguments to run a different Ubuntu server:

> *qemu-system-aarch64 -accel hvf -cpu a64fx -M virt-7.2,highmem=off -m 512 -smp 4 -drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on -drive if=none,file=ubuntu.img,format=qcow2,id=hd0 -device virtio-blk-device,drive=hd0,serial="dummyserial" -device virtio-net-device,netdev=net0 -netdev user,id=net0 -vga none -device ramfb -device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic*


qemu-system-aarch64 -accel hvf -cpu a64fx -M virt-7.2,highmem=off -m 512 -smp
root@763a58293a2e: / (... ⌘1
qemu-system-aarch64 (qe... ⌘2
~ (-zsh) ⌘3
-cpu
14%
7.6 GB

```
biu_test@biuubuntu20045:/$ sudo lshw -short
[sudo] password for biu_test:
H/W path        Device     Class      Description
==================================================
                           system     QEMU Virtual Machine
/0                         bus        Motherboard
/0/400                     processor  CPU
/0/1000                    memory     512MiB System Memory
/0/1000/0                  memory     512MiB DIMM RAM
/0/0                       memory     96KiB BIOS
/0/100                     bridge     QEMU PCIe Host bridge
/0/100/1                   bus        82801DB/DBM (ICH4/ICH4-M) USB2 EHCI Controll
/0/100/1/1      usb1       bus        EHCI Host Controller
/0/100/1/1/1               input      QEMU USB Keyboard
/0/100/1/1/2               input      QEMU USB Mouse
/0/1                       system     PnP device PNP0c02
/0/2            eth0       network    Ethernet interface
/0/3            /dev/vda   disk       10GB Virtual I/O device
/0/3/1          /dev/vda1  volume     537MiB Windows FAT volume
/0/3/2          /dev/vda2  volume     1792MiB EXT4 volume
/0/3/3          /dev/vda3  volume     7908MiB EFI partition
biu_test@biuubuntu20045:/$ 
```

VM for arguments test

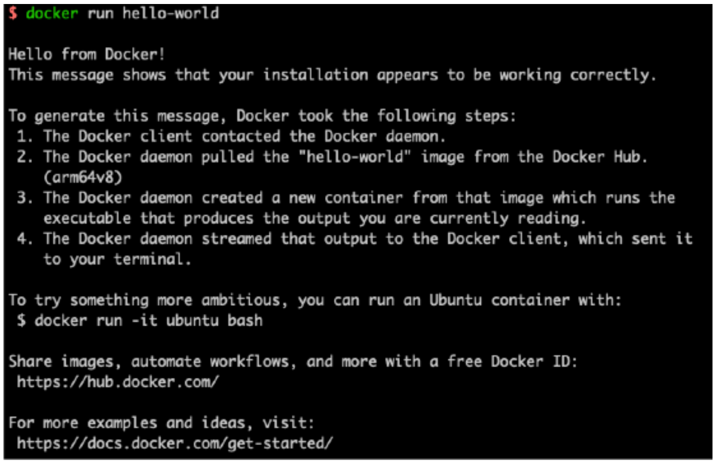

qemu-system-aarch64 -accel hvf -m 2048 -cpu cortex-a57 -M virt,highmem=off
22%
7.6 GB
0.0 kB↓
0.0 kB↑

```
biu_test@biuubuntu20045:~$ sudo lshw -short
[sudo] password for biu_test:
Sorry, try again.
[sudo] password for biu_test:
H/W path        Device     Class      Description
==================================================
                           system     QEMU Virtual Machine
/0                         bus        Motherboard
/0/400                     processor  CPU
/0/1000                    memory     2GiB System Memory
/0/1000/0                  memory     2GiB DIMM RAM
/0/0                       memory     96KiB BIOS
/0/100                     bridge     QEMU PCIe Host bridge
/0/100/1                   bus        82801DB/DBM (ICH4/ICH4-M) USB2 EHCI Controll
/0/100/1/1      usb1       bus        EHCI Host Controller
/0/100/1/1/1               input      QEMU USB Keyboard
/0/100/1/1/2               input      QEMU USB Mouse
/0/1                       system     PnP device PNP0c02
/0/2            eth0       network    Ethernet interface
/0/3            /dev/vda   disk       10GB Virtual I/O device
/0/3/1          /dev/vda1  volume     537MiB Windows FAT volume
/0/3/2          /dev/vda2  volume     1792MiB EXT4 volume
/0/3/3          /dev/vda3  volume     7908MiB EFI partition
biu_test@biuubuntu20045:~$ 
```

VM for hw1

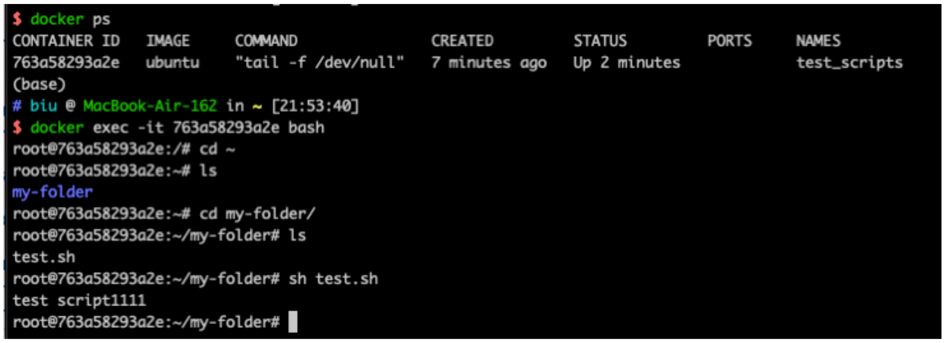
Docker Container Installation

1. After installing Docker, run hello-world for testing.



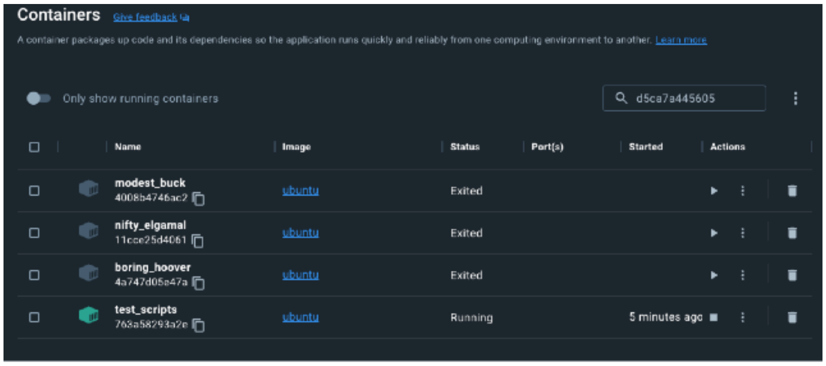
```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

2. Use Docker Mount Volume to mount my local directories and files (scripts) to ubuntu container. And make this container runs forever (unless we kill it manualy).
$ docker run --name test_scripts -v ~/Desktop/coen241_hw/hw1/scripts/:/root/my-folder ubuntu tail -f /dev/null
(edit: did not use mounted directory later, since I realized that mounting will impact the I/O throughput performance significantly…)



```
$ docker ps
CONTAINER ID   IMAGE     COMMAND               CREATED         STATUS         PORTS     NAMES
763a58293a2e   ubuntu    "tail -f /dev/null"   7 minutes ago   Up 2 minutes             test_scripts
(base)
# biu @ MacBook-Air-162 in ~ [21:53:40]
$ docker exec -it 763a58293a2e bash
root@763a58293a2e:/# cd ~
root@763a58293a2e:~# ls
my-folder
root@763a58293a2e:~# cd my-folder/
root@763a58293a2e:~/my-folder# ls
test.sh
root@763a58293a2e:~/my-folder# sh test.sh
test script1111
root@763a58293a2e:~/my-folder# █
```

3. Run ubuntu container and install Sysbench.



Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

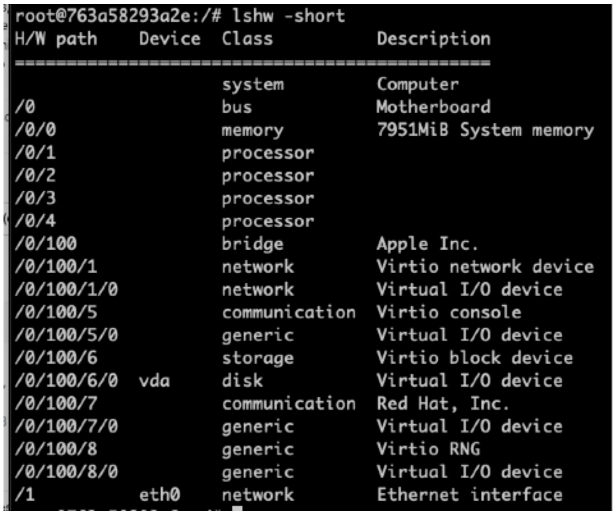
d5ca7a445605

□	Name	Image	Status	Port(s)	Started	Actions
□	modest_buck 4008b4746ac2	ubuntu	Exited			
□	nifty_elgamal 11cce25d4061	ubuntu	Exited			
□	boring_hoover 4a747d05e47a	ubuntu	Exited			
□	test_scripts 763a58293a2e	ubuntu	Running		5 minutes ago	

$ docker run -it ubuntu bash
$ apt update
$ apt install sysbench

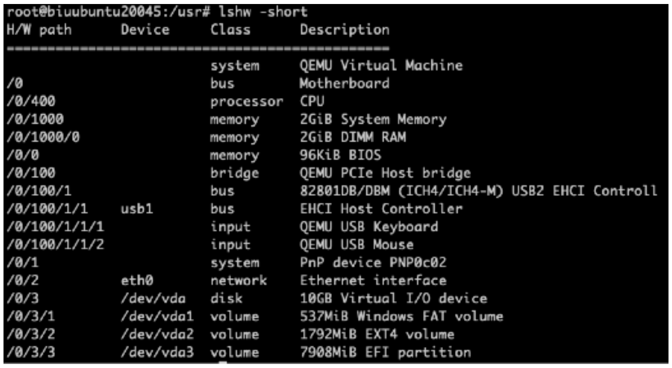
CPU Performance Tests

- Hardware environment info:



```
root@763a58293a2e:/# lshw -short
H/W path     Device  Class          Description
===================================================
                     system         Computer
/0                   bus            Motherboard
/0/0                 memory         7951MiB System memory
/0/1                 processor
/0/2                 processor
/0/3                 processor
/0/4                 processor
/0/100               bridge         Apple Inc.
/0/100/1             network        Virtio network device
/0/100/1/0           network        Virtual I/O device
/0/100/5             communication  Virtio console
/0/100/5/0           generic        Virtual I/O device
/0/100/6             storage        Virtio block device
/0/100/6/0   vda     disk           Virtual I/O device
/0/100/7             communication  Red Hat, Inc.
/0/100/7/0           generic        Virtual I/O device
/0/100/8             generic        Virtio RNG
/0/100/8/0           generic        Virtual I/O device
/1           eth0    network        Ethernet interface
```

Docker VM

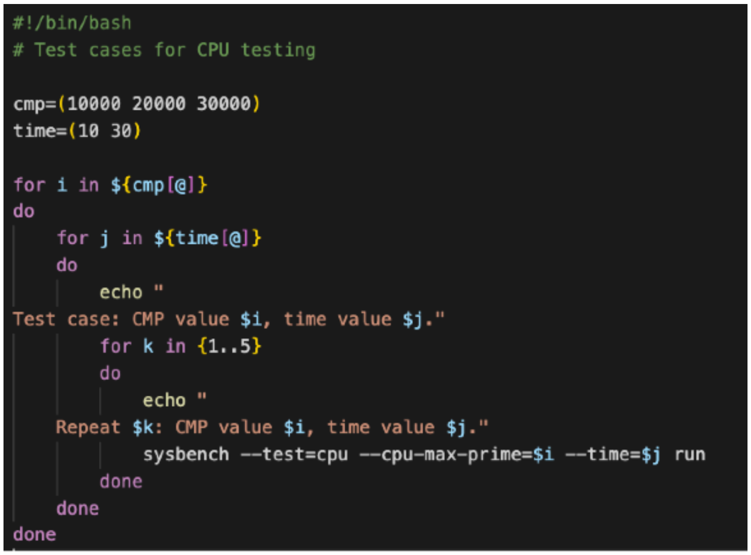


```
root@biuubuntu20045:/usr# lshw -short
H/W path       Device      Class      Description
=================================================
                           system     QEMU Virtual Machine
/0                         bus        Motherboard
/0/400                     processor  CPU
/0/1000                    memory     2GiB System Memory
/0/1000/0                  memory     2GiB DIMM RAM
/0/0                       memory     96KiB BIOS
/0/100                     bridge     QEMU PCIe Host bridge
/0/100/1                   bus        82801DB/DBM (ICH4/ICH4-M) USB2 EHCI Controll
/0/100/1/1     usb1        bus        EHCI Host Controller
/0/100/1/1/1               input      QEMU USB Keyboard
/0/100/1/1/2               input      QEMU USB Mouse
/0/1                       system     PnP device PNP0c02
/0/2           eth0        network    Ethernet interface
/0/3           /dev/vda    disk       10GB Virtual I/O device
/0/3/1         /dev/vda1   volume     537MiB Windows FAT volume
/0/3/2         /dev/vda2   volume     1792MiB EXT4 volume
/0/3/3         /dev/vda3   volume     7908MiB EFI partition
```

QEMU VM

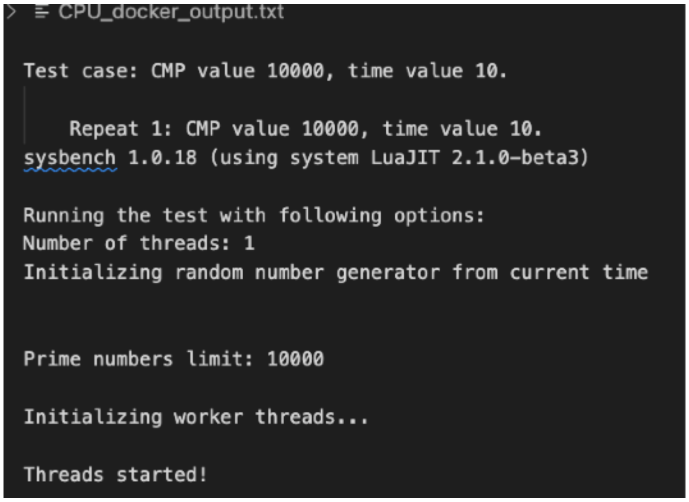
- Scripts for CPU testing: sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
- Test cases: all are single thread
    1. --cpu-max-prime=10000 --time=10
    2. --cpu-max-prime=20000 --time=10
    3. --cpu-max-prime=30000 --time=10
    4. --cpu-max-prime=10000 --time=30
    5. --cpu-max-prime=20000 --time=30
    6. --cpu-max-prime=30000 --time=30

- Use scripts for automation test.



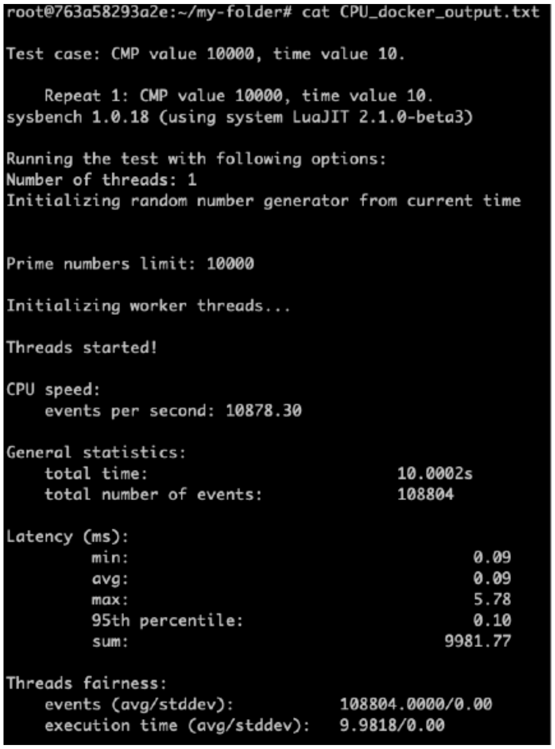
```
#!/bin/bash
# Test cases for CPU testing

cmp=(10000 20000 30000)
time=(10 30)

for i in ${cmp[@]}
do
    for j in ${time[@]}
    do
        echo "
Test case: CMP value $i, time value $j."
        for k in {1..5}
        do
            echo "
    Repeat $k: CMP value $i, time value $j."
            sysbench --test=cpu --cpu-max-prime=$i --time=$j run
        done
    done
done
```

- Write all output into single txt file.
  bash CPU_tests.sh > CPU_<VM>_output.txt



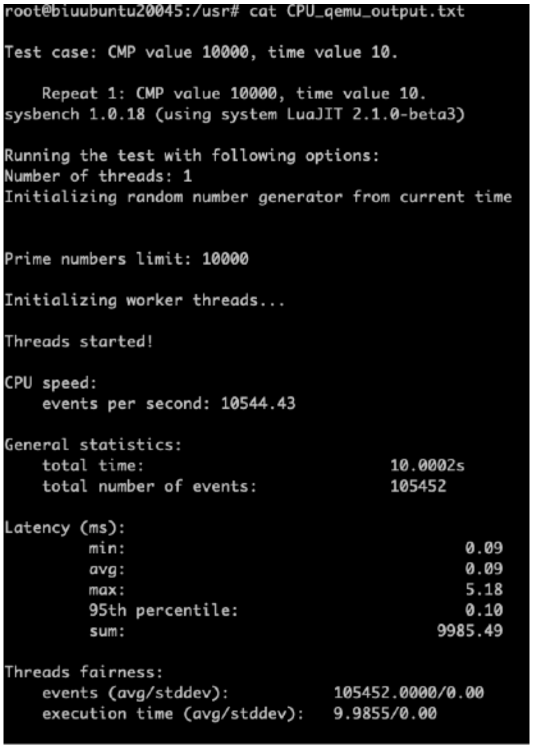
```
> ≡ CPU_docker_output.txt

Test case: CMP value 10000, time value 10.

    Repeat 1: CMP value 10000, time value 10.
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!
```

- Test output example:



```
root@763a58293a2e:~/my-folder# cat CPU_docker_output.txt

Test case: CMP value 10000, time value 10.

    Repeat 1: CMP value 10000, time value 10.
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: 10878.30

General statistics:
    total time:                          10.0002s
    total number of events:              108804

Latency (ms):
         min:                                    0.09
         avg:                                    0.09
         max:                                    5.78
         95th percentile:                        0.10
         sum:                                 9981.77

Threads fairness:
    events (avg/stddev):           108804.0000/0.00
    execution time (avg/stddev):   9.9818/0.00
```

Docker test result



```
root@biuubuntu20045:/usr# cat CPU_qemu_output.txt

Test case: CMP value 10000, time value 10.

    Repeat 1: CMP value 10000, time value 10.
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: 10544.43

General statistics:
    total time:                          10.0002s
    total number of events:              105452

Latency (ms):
         min:                                    0.09
         avg:                                    0.09
         max:                                    5.18
         95th percentile:                        0.10
         sum:                                 9985.49

Threads fairness:
    events (avg/stddev):           105452.0000/0.00
    execution time (avg/stddev):   9.9855/0.00
```

QEMU test result

- Other useful commands:
  - Copy file from docker container to host: $ docker cp <containerID>:<filePathInHost>.
  - Copy file from QEMU VM to host:

Test Result:
All detailed data please check:
https://docs.google.com/spreadsheets/d/1vv_p0Dv_Jq3FRMXocnNmw7mHjolGXhPXzRxQ7eLHmPQ/edit?usp=sharing

Test case 1: sysbench --test=cpu --cpu-max-prime=10000 --time=10 run

Docker

test	min	max	avg	events/sec	total events
1	0.09	5.78	0.09	10878.30	108804.00
2	0.09	0.71	0.09	10985.73	109878.00
3	0.09	5.18	0.09	10883.61	108855.00
4	0.09	14.02	0.09	10849.46	108503.00
5	0.09	0.91	0.09	10730.28	107311.00
std				91.50	920.08
avg	0.09	5.32	0.09	10865.48	108670.20

QEMU

test	min	max	avg	events/sec	total events
1	0.09	5.18	0.09	10544.43	105452.00
2	0.09	0.47	0.09	10651.07	106520.00
3	0.09	1.59	0.09	10658.00	106587.00
4	0.09	0.48	0.09	10661.94	106628.00
5	0.09	0.41	0.09	10654.96	106558.00
std				50.27	503.00
avg	0.09	1.63	0.09	10634.08	106349.00

Test case 2: sysbench --test=cpu --cpu-max-prime=10000 --time=30 run

test	min	max	avg	events/sec	total events
1	0.09	25.51	0.09	10941.08	328241.00
2	0.09	21.13	0.09	10881.35	326471.00
3	0.09	4.69	0.09	10887.77	326643.00
4	0.09	20.13	0.09	10944.32	328338.00
5	0.09	106.87	0.10	10497.13	314924.00
std				188.54	5656.99
avg	0.09	35.67	0.09	10830.33	324923.40

test	min	max	avg	events/sec	total events
1	0.09	0.85	0.09	10566.18	316995.00
2	0.09	0.53	0.09	10668.55	320065.00
3	0.09	0.52	0.09	10569.46	317092.00
4	0.09	0.49	0.09	10654.97	319657.00
5	0.09	1.51	0.09	10614.44	318441.00
std				47.23	1416.54
avg	0.09	0.78	0.09	10614.72	318450.00

Test case 3: sysbench --test=cpu --cpu-max-prime=20000 --time=10 run

test	min	max	avg	events/sec	total events
1	0.23	0.31	0.23	4304.49	43049.00
2	0.23	6.73	0.23	4249.20	42496.00
3	0.23	11.44	0.24	4221.47	42223.00
4	0.23	3.51	0.23	4292.11	42925.00
5	0.23	0.55	0.24	4243.57	42440.00
std				34.85	347.22
avg	0.23	4.51	0.23	4262.17	42626.60

test	min	max	avg	events/sec	total events
1	0.23	1.38	0.24	4121.48	41218.00
2	0.23	0.99	0.24	4184.52	41848.00
3	0.23	5.73	0.24	4112.59	41129.00
4	0.23	5.31	0.24	4160.15	41605.00
5	0.23	2.38	0.24	4167.60	41680.00
std				30.88	308.87
avg	0.23	3.16	0.24	4149.27	41496.00

Test case 4: sysbench --test=cpu --cpu-max-prime=20000 --time=30 run

test	min	max	avg	events/sec	total events
1	0.23	6.19	0.24	4153.02	124595.00
2	0.23	1.44	0.24	4160.04	124805.00
3	0.23	2.93	0.24	4131.00	123934.00
4	0.23	2.81	0.24	4164.68	124945.00
5	0.23	1.50	0.24	4131.29	123943.00
std				15.94	478.34
avg	0.23	2.97	0.24	4148.01	124444.40

test	min	max	avg	events/sec	total events
1	0.23	3.50	0.24	4136.24	124091.00
2	0.23	12.97	0.24	4148.76	124466.00
3	0.23	22.47	0.24	4123.71	123716.00
4	0.23	1.67	0.24	4214.74	126446.00
5	0.23	21.09	0.24	4232.82	126988.00
std				49.18	1475.06
avg	0.23	12.34	0.24	4171.25	125141.40

Test case 5: sysbench --test=cpu --cpu-max-prime=30000 --time=10 run

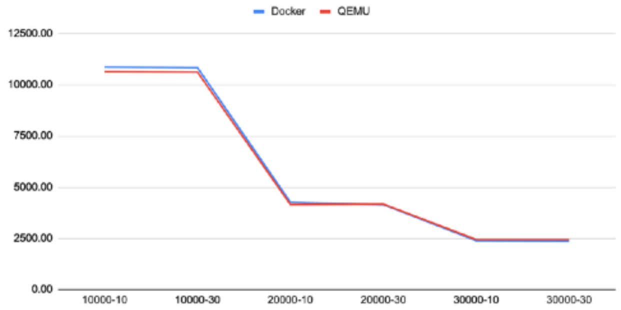
test	min	max	avg	events/sec	total events
1	0.41	1.76	0.42	2385.40	23859.00
2	0.41	2.34	0.42	2383.22	23838.00
3	0.41	2.19	0.42	2377.80	23781.00
4	0.41	0.56	0.42	2383.91	23842.00
5	0.41	2.70	0.42	2381.13	23814.00
std				2.94	30.23
avg	0.41	1.91	0.42	2382.29	23826.80

test	min	max	avg	events/sec	total events
1	0.40	1.91	0.41	2408.97	24092.00
2	0.40	1.44	0.41	2429.76	24300.00
3	0.40	1.04	0.41	2442.83	24431.00
4	0.39	1.56	0.41	2451.73	24520.00
5	0.39	10.69	0.41	2461.87	24621.00
std				20.54	205.43
avg	0.40	3.33	0.41	2439.03	24392.80

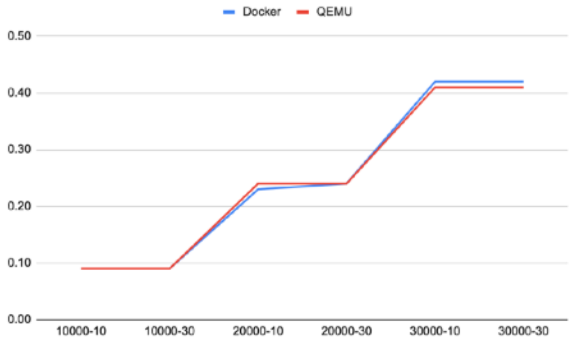
Test case 6: sysbench --test=cpu --cpu-max-prime=30000 --time=30 run

test	min	max	avg	events/sec	total events
1	0.41	4.00	0.42	2359.04	70774.00
2	0.41	9.68	0.42	2375.93	71280.00
3	0.41	3.97	0.42	2355.36	70664.00
4	0.41	21.84	0.42	2364.93	70951.00
5	0.41	17.89	0.43	2347.47	70429.00
std				10.68	319.41
avg	0.41	11.48	0.42	2360.55	70819.60

test	min	max	avg	events/sec	total events
1	0.39	3.74	0.41	2434.05	73024.00
2	0.39	10.42	0.41	2439.38	73184.00
3	0.39	25.96	0.41	2430.94	72930.00
4	0.39	5.90	0.41	2433.18	72998.00
5	0.39	1.74	0.41	2464.69	73943.00
				13.90	417.05
	0.39	9.55	0.41	2440.45	73215.80



CPU speed (events/sec)



Average latency

- Both in CPU performance, Docker VM is slightly better than QEMU VM, but not a significant difference.
- CPU speed will decrease with prime-max increasing.
- Average latency will increase with prime-max increasing.

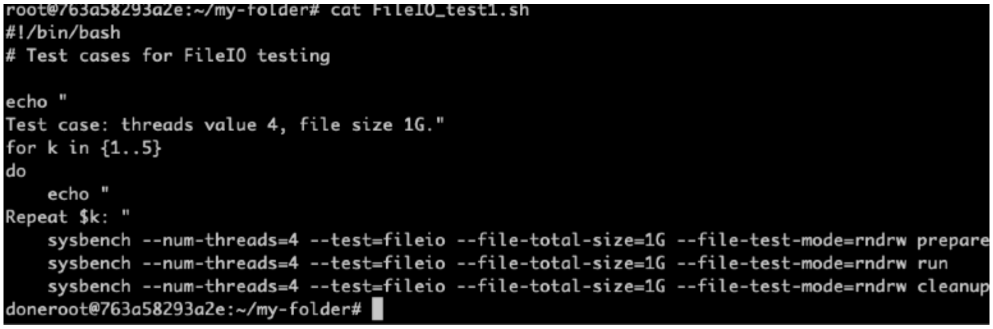
FileIO Performance Tests

- Scripts for FileIO testing:
$ sysbench --num-threads=<num> --test=fileio --file-total-size=<size> --file-test-mode=<mode> prepare
$ sysbench --num-threads=<num> --test=fileio --file-total-size=<size> --file-test-mode=<mode> run
$ sysbench --num-threads=<num> --test=fileio --file-total-size=<size> --file-test-mode=<mode> cleanup

- Manually drop cache in the host: sync && sudo purge

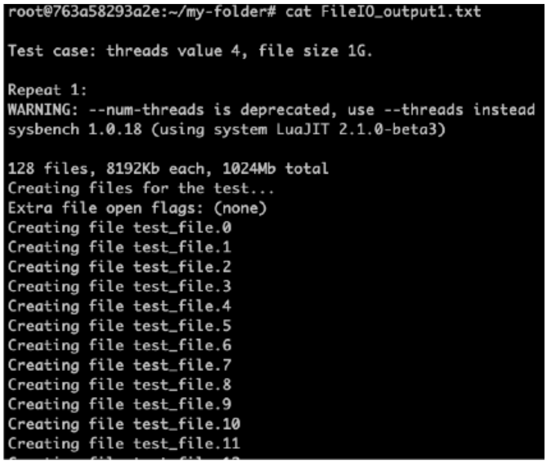
- Test cases:
    1. --num-threads=4 --test=fileio --file-total-size=1G --file-test-mode=rndrw
    2. --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw
    3. --num-threads=4 --test=fileio --file-total-size=3G --file-test-mode=rndrw
    4. --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw

- Use scripts for automation test:



```
root@763a58293a2e:~/my-folder# cat FileIO_test1.sh
#!/bin/bash
# Test cases for FileIO testing

echo "
Test case: threads value 4, file size 1G."
for k in {1..5}
do
    echo "
Repeat $k: "
    sysbench --num-threads=4 --test=fileio --file-total-size=1G --file-test-mode=rndrw prepare
    sysbench --num-threads=4 --test=fileio --file-total-size=1G --file-test-mode=rndrw run
    sysbench --num-threads=4 --test=fileio --file-total-size=1G --file-test-mode=rndrw cleanup
doneroot@763a58293a2e:~/my-folder# 
```

- Write all output into txt file.
  bash FileIO_testx.sh > FileIO_outputx.txt



```
root@763a58293a2e:~/my-folder# cat FileIO_output1.txt

Test case: threads value 4, file size 1G.

Repeat 1:
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

128 files, 8192Kb each, 1024Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
```

Test Result:

Test case 1: --num-threads=4 --test=fileio --file-total-size=1G --file-test-mode=rndrw

Docker

test	read, MiB/s	written, MiB/s	latency_avg
1	253.08	168.71	0.06
2	232.76	155.17	0.07
3	247.1	164.74	0.07
4	280.14	186.76	0.06
5	271.53	181.03	0.06
avg	256.922	171.282	0.064

QEMU

test	read, MiB/s	written, MiB/s	latency_avg
1	251.67	167.78	0.06
2	272.6	181.74	0.06
3	268.56	179.04	0.06
4	267.4	178.27	0.06
5	262.6	175.07	0.06
avg	264.566	176.38	0.06

Test case 2: --num-threads=16 --test=fileio --file-total-size=1G —file-test-mode=rndrw

test	read, MiB/s	written, MiB/s	latency_avg
1	455.92	303.95	0.14
2	262.53	175.02	0.25
3	261.66	174.44	0.25
4	245.84	163.9	0.27
5	241.13	160.76	0.27
avg	293.416	195.614	0.236

test	read, MiB/s	written, MiB/s	latency_avg
1	462.33	308.22	0.14
2	372.31	248.21	0.18
3	470.15	313.44	0.14
4	385.98	257.32	0.17
5	430.68	287.12	0.15
avg	424.29	282.862	0.156

Test case 3: --num-threads=4 --test=fileio --file-total-size=3G --file-test-mode=rndrw

test	read, MiB/s	written, MiB/s	latency_avg
1	249.64	166.42	0.07
2	211.83	141.22	0.08
3	174.72	116.48	0.09
4	181.54	121.03	0.09
5	236.55	157.7	0.07
avg	210.856	140.57	0.08

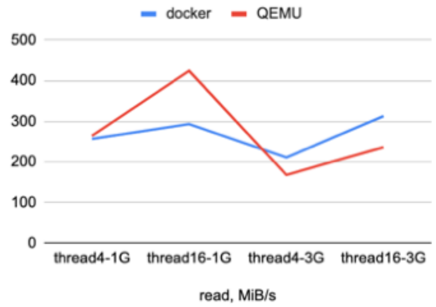
test	read, MiB/s	written, MiB/s	latency_avg
1	168.58	112.39	0.1
2	165.77	110.51	0.1
3	151.64	101.09	0.11
4	176.74	117.82	0.09
5	175.89	117.26	0.09
avg	167.724	111.814	0.098

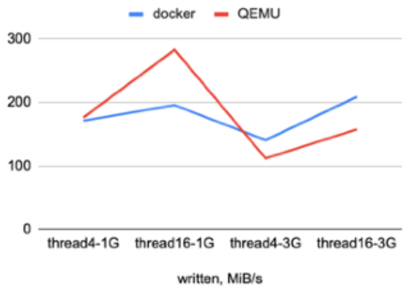
Test case 4: --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw

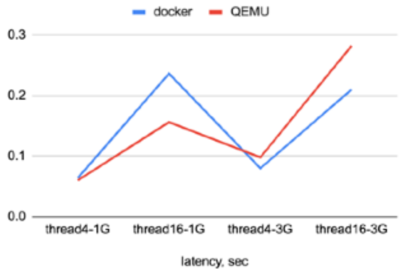
test	read, MiB/s	written, MiB/s	latency_avg
1	360.12	240.08	0.18
2	306.38	204.26	0.21
3	299.51	199.67	0.22
4	310.87	207.25	0.21
5	288.65	192.43	0.23
avg	313.106	208.738	0.21

test	read, MiB/s	written, MiB/s	latency_avg
1	276.66	184.42	0.24
2	186.06	124.03	0.35
3	212.8	141.85	0.31
4	237.84	158.55	0.27
5	268.36	178.89	0.24
avg	236.344	157.548	0.282

Throughput & latency:





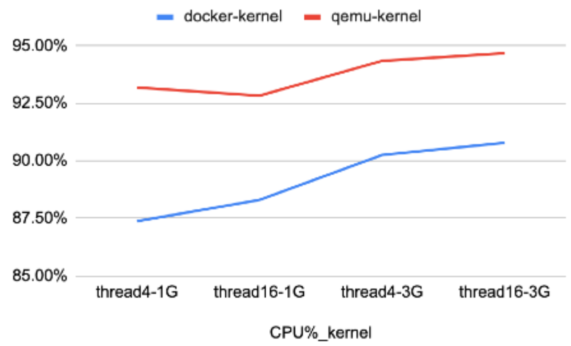


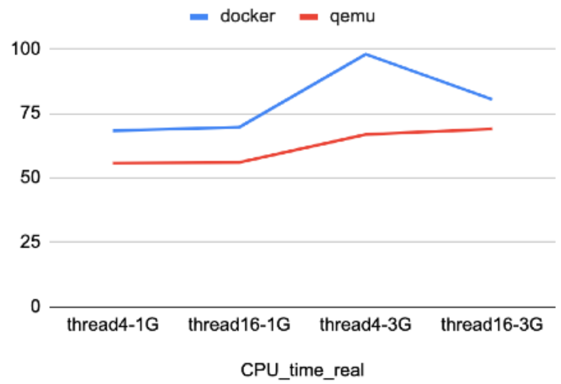
- Test cases may not enough to find a data pattern.
- With smaller file size, QEMU VM will have better performance, which indicates the higher throughput and lower latency.
- But generally, there is no significant difference with throughput for docker VM and QEMU VM.

CPU time, sec:

- The ‘real’ time is clock time from start to finish of the call including time used by other processes and time the process spends blocked.
- The ‘kernel’ time is the amount of CPU time spent in the kernel within the process.
- The ‘user’ time is the actual CPU time used in executing the process.

	thread4-1G	thread16-1G	thread4-3G	thread16-3G
docker-kernel	45.744	37.361	54.735	47.629
docker-user	6.606	4.948	5.909	4.835
docker-real	68.298	69.583	97.916	80.368
qemu-kernel	25.749	32.135	26.074	31.846
qemu-user	1.884	2.482	1.564	1.792
qemu-real	55.684	56.008	66.815	68.918





- The QEMU VM has higher kernel CPU usage proportion, which means during the whole CPU time, QEMU VM uses more CPU% to do the I/O.
- The docker VM needs more time to finish test cases, which means docker VM is slower in file I/O performance.