COEN241 HW1 report

Tinghui Zhang

W1652191

1.Detailed configurations of your experimental setup

Host machine:

CPU: Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz 2.30 GHz

RAM: 8GB

OS: windows10

Virtual machine(run on VMware):

RAM: 5.7GB

OS:Ubuntu 20.04.5

2.Main steps to enable a QEMU VM. In addition, please present the detailed QEMU commands, and VM configurations

QEMU(installed on the virtual machine):

Main steps:

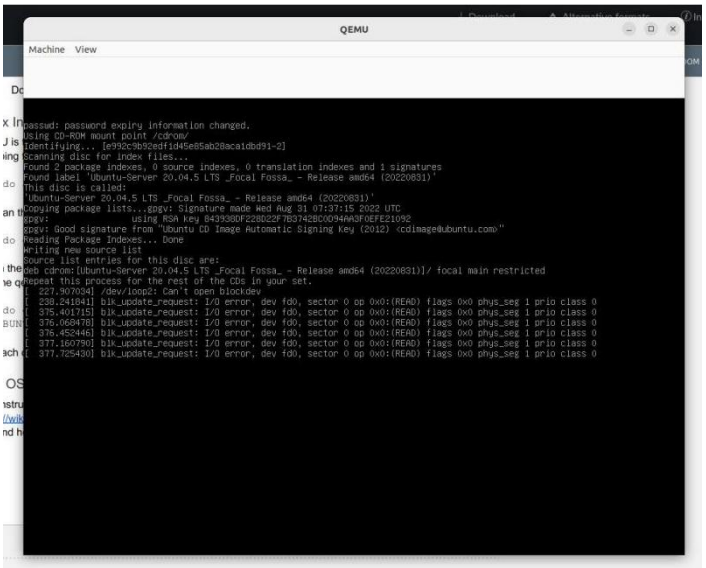
Commands used while installing:

Install QEMU using the following command:

```
sudo apt-get install qemu
```

Create QEMU image and install the VM which takes the iso file as a “cdrom” and the qemu image as a hard disk(iso file used is also 20.04.5 and the memory was 2046).

```
sudo qemu-img create ubuntu.img 10G -f qcow2
sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ./241/20.04.iso -m 2046 -boot strict=on
```



Installing QEMU on Ubuntu

After the installation completed, restart the VMware and removing cdrom flag while rebooting the QEMU.

```
sudo qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046
```

```
root@ubuntu:/home/hui# 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
```

QEMU version

3.Main steps to enable the Docker container. steps in creating my own image and your image history

Docker(installed on the virtual machine):

Commands used while installing:

Set up the repository

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

2. Add Docker's official GPG key:

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

1. Update the `apt` package index, and install the *latest version* of Docker Engine, containerd, and Docker Compose, or go to the next step to install a specific version:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

3. Verify that Docker Engine is installed correctly by running the `hello-world` image.

```
$ sudo service docker start
$ sudo docker run hello-world
```

```
root@ubuntu:/home/hui# docker --version
Docker version 20.10.19, build d85ef84
root@ubuntu:/home/hui#
```

Docker version

Install the docker image zyclonite/sysbench using the command

pull zyclonite/sysbench

```
root@ubuntu:/home/hui# docker pull zyclonite/sysbench
Using default tag: latest
latest: Pulling from zyclonite/sysbench
Digest: sha256:016020c3b53c7e65cdb58e7d4a98afd14f8a3e2f5781cf4c368596b2e448602b
Status: Image is up to date for zyclonite/sysbench:latest
docker.io/zyclonite/sysbench:latest
```

create my image and verify image history

Docker images

Docker ps -a

Docker commit 3a56ff4626ca bench_image

Docker history bench_image

```
root@ubuntu:/home/hui# docker images
REPOSITORY           TAG       IMAGE ID       CREATED         SIZE
zyclonite/sysbench   latest    31638b096d0e   10 months ago   9.75MB
root@ubuntu:/home/hui# docker rm -f efcca948866c
efcca948866c
root@ubuntu:/home/hui# docker ps -a
CONTAINER ID   IMAGE                COMMAND              CREATED          STATUS
 PORTS     NAMES
3a56ff4626ca   zyclonite/sysbench   "sysbench /bin/sh"   10 minutes ago   Exited (1) 10 minutes ago
           awesome_ardinghelli
root@ubuntu:/home/hui# docker commit 3a56ff4626ca
sha256:8764345608b49695cd0282929f54d8ddc5cb27ea30370417d2a8fc9a7b5a397c
root@ubuntu:/home/hui# docker commit 3a56ff4626ca bench_image
sha256:bc17440703d5b0eecddd8962e978b17cbc5c0ca8bea89bd9c8f706185afac618
root@ubuntu:/home/hui# docker history bench_image
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
bc17440703d5   24 seconds ago   /bin/sh                                         0B
31638b096d0e   10 months ago    /bin/sh -c #(nop) CMD [ "--help" ]              4.17MB    FROM docker
.io/library/alpine:3.15
<missing>      10 months ago    /bin/sh -c #(nop) ENTRYPOINT [ "sysbench" ]     0B
<missing>      10 months ago    |1 version=1.0.20-r0 /bin/sh -c apk add --no…   0B
<missing>      10 months ago    /bin/sh -c #(nop) ARG version=1.0.20-r0         0B
<missing>      10 months ago    /bin/sh -c #(nop) LABEL description "Sysbenc…   0B
<missing>      10 months ago    /bin/sh -c #(nop) LABEL version "1.0.20"        0B
<missing>      10 months ago    /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      10 months ago    /bin/sh -c #(nop) ADD file:9233f6f2237d79659…   5.59MB
```

```
root@ubuntu:/home/hui# docker ps -a
CONTAINER ID   IMAGE                COMMAND              CREATED       STATUS                   PORTS
     NAMES
3a56ff4626ca   zyclonite/sysbench   "sysbench /bin/sh"   2 hours ago   Exited (1) 2 hours ago
     awesome_ardinghelli
root@ubuntu:/home/hui# docker images
REPOSITORY           TAG       IMAGE ID       CREATED         SIZE
bench_image          latest    bc17440703d5   2 hours ago     9.75MB
<none>               <none>    8764345608b4   2 hours ago     9.75MB
zyclonite/sysbench   latest    31638b096d0e   10 months ago   9.75MB
root@ubuntu:/home/hui#
```

Check the container and images

the operations used to manage Docker containers

Useful operations to manage the docker containers:

Docker pull: pull images from from a remote repository

Docker images: show the images

Docker ps: Display the containers in the docker

Docker run: create a container from a image

Docker kill: kill containers

Docker stop: stop a container

Docker start: restart a container

Docker --version: show the current version of docker

Docker rm: remove a container

Docker rmi: remove a image

Docker push: push an image of repository

Proof of experiment. Including screen snapshots of Docker and QEMU running environments for each experiment

CPU performance test:

Test conditions: cpu max prime:10000,20000,30000 run

Each repeated 5 times, 15 tests in total. Proof of experiment(only part of all) are the following screenshots.

Used the following command to delete caches to decrease variations.

```
root@ubuntu:/home/hui# sync; echo 3 > /proc/sys/vm/drop_caches
root@ubuntu:/home/hui#
```

QEMU CPU test:

Sysbench --test=cpu --cpu-max-prime=10000 run

```
hui@hui:~$ sysbench --test=cpu --cpu-max-prime=10000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options
sysbench 1.0.18 (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:   216.90

General statistics:
    total time:                          10.0060s
    total number of events:              2172

Latency (ms):
         min:                                    3.70
         avg:                                    4.56
         max:                                   39.06
         95th percentile:                        5.28
         sum:                                 9910.46

Threads fairness:
    events (avg/stddev):           2172.0000/0.00
    execution time (avg/stddev):   9.9105/0.00

hui@hui:~$ _
```

DOCKER CPU test:

Sysbench --test=cpu --cpu-max-prime=10000 run

```
root@ubuntu:/home/hui# sysbench --test=cpu --cpu-max-prime=10000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line wit
hout any options.
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:   841.05

General statistics:
    total time:                          10.0009s
    total number of events:              8413

Latency (ms):
         min:                                    0.94
         avg:                                    1.19
         max:                                    4.29
         95th percentile:                        1.55
         sum:                                 9987.24

Threads fairness:
    events (avg/stddev):           8413.0000/0.00
    execution time (avg/stddev):   9.9872/0.00

root@ubuntu:/home/hui#
```

FILE I/O test:

Test conditions: “random read” “random write” “random read/write”

8 threads with file size as 1GB.

Each repeated 5 times, 15 tests in total. Proof of experiment(only part of all) are the following screenshots.

QEMU FILE I/O test:

Random write

Sysbench --test=fileio --file-total-size=500MB prepare

Sysbench --test=fileio --file-total-size=500MB --file-test-mode=rndwr --time=30 --max-requests=0 run

Sysbench --test=fileio --file-total-size=500MB cleanup

```
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
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


Extra file open flags: (none)
128 files, 3.9062MiB each
500MiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random write test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     393.47
    fsyncs/s:                     505.13

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               6.15

General statistics:
    total time:                          30.6188s
    total number of events:              27506

Latency (ms):
         min:                                    0.03
         avg:                                    1.03
         max:                                 1470.77
         95th percentile:                        2.91
         sum:                                28445.48

Threads fairness:
    events (avg/stddev):           27506.0000/0.00
    execution time (avg/stddev):   28.4455/0.00

root@hui:/home/hui#
```

DOCKER FILE I/O test:

Random write

Sysbench --test=fileio --file-total-size=500MB prepare

Sysbench --test=fileio --file-total-size=500MB --file-test-mode=rndwr --time=30 --max-requests=0 run

Sysbench --test=fileio --file-total-size=500MB cleanup

```
root@ubuntu:/home/hui# sysbench --test=fileio --file-total-size=500MB --file-test-mode=rndwr -
-time=30 --max-requests=0 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command li
ne without any options.
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


Extra file open flags: (none)
128 files, 3.9062MiB each
500MiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random write test
Initializing worker threads...

Threads started!
```

```
Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     6781.34
    fsyncs/s:                     8680.28

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               105.96

General statistics:
    total time:                          30.0092s
    total number of events:              464313

Latency (ms):
         min:                                    0.00
         avg:                                    0.06
         max:                                  241.92
         95th percentile:                        0.16
         sum:                                29687.26

Threads fairness:
    events (avg/stddev):           464313.0000/0.00
    execution time (avg/stddev):   29.6873/0.00

root@ubuntu:/home/hui#
```

Present how you conduct your measurements in three different scenarios for each virtualization technology

Present how you use performance tools to collect performance data. For CPU utilization, you should at least divide them into two parts including user-level and kernel-level. For I/O, you should present I/O throughput, latency, and disk uilization

CPU test:

QEMU CPU test:

Cpu-max-prime=10000

min	avg	max	Events per sec	Total time	Total number of events
3.70	4.56	39.06	216.9	10s	2172
3.77	4.37	27.00	222.49		2277
3.80	4.47	17.81	218.39		2221
3.69	5.40	357.92	168.11		1716
3.81	4.77	87.24	201.32		2052

Cpu-max-prime=20000

min	avg	max	Events per sec	Total time	Total number of events
9.85	11.18	100.36	83.28	10s	835
9.69	11.22	18.25	88.28		886
10.01	11.76	95.23	82.53		845
9.72	11.31	42.99	85.44		865
9.86	11.83	70.22	80.60		821

Cpu-max-prime=30000

min	avg	max	Events per sec	Total time	Total number of events
17.12	19.69	53.24	50.12	10s	506
16.99	20.35	49.35	48.72		490
17.46	19.85	56.12	50.16		504
17.33	19.37	46.66	50.84		514
17.10	19.91	152.15	49.52		501

DOCKER CPU test:

Cpu-max-prime=10000

min	avg	max	Events per sec	Total time	Total number of events
0.94	1.19	4.29	841.05	10s	8413
0.94	1.10	2.37	904.34		9046
0.94	1.11	11.95	896.15		8964
0.94	1.21	10.41	827.57		8278
0.94	1.21	4.83	888.41		8886

Cpu-max-prime=20000

min	avg	max	Events per sec	Total time	Total number of events
2.44	1.81	11.07	355.23	10s	3554
2.44	2.90	17.58	344.61		3447
2.46	2.91	14.43	343.13		3432
2.44	2.88	15.27	346.46		3466
2.44	2.85	4.80	350.02		3501

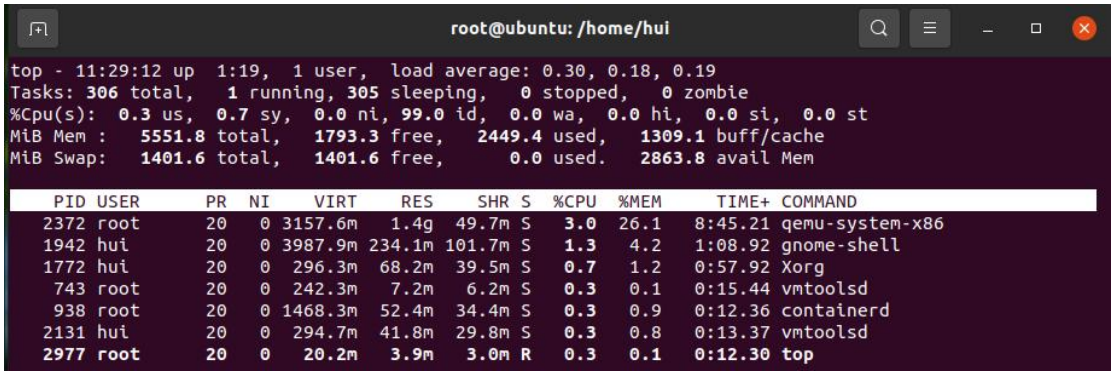
Cpu-max-prime=30000

min	avg	max	Events per sec	Total time	Total number of events
4.35	5.64	52.75	177.21	10s	1773
4.35	5.16	7.28	193.72		1938
4.38	5.67	277.39	171.43		1760
4.37	5.46	280.84	178.95		1830
4.34	5.10	12.30	195.80		1959

Show CPU utilization by command:

top

When DOCKER is idle, CPU utilization is about 3%



When DOCKER is running, CPU utilization is about 99.7%

```
root@ubuntu: /home/hui
top - 11:30:48 up  1:20,  1 user,  load average: 0.21, 0.16, 0.18
Tasks: 310 total,   1 running, 309 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.7 us,  0.6 sy,  0.0 ni, 73.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   5551.8 total,   1790.1 free,   2452.5 used,   1309.2 buff/cache
MiB Swap:   1401.6 total,   1401.6 free,      0.0 used.   2860.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3039 root      20   0   32.6m  10.0m   8.1m S  99.7   0.2   0:06.54 sysbench
   2372 root      20   0 3157.6m   1.4g  49.7m S   6.6  26.1   8:49.63 qemu-system-x86
   1942 hui       20   0 3987.9m 234.2m 101.7m S   1.7   4.2   1:10.95 gnome-shell
   1772 hui       20   0  296.3m  68.2m  39.5m S   1.0   1.2   1:01.09 Xorg
    436 root     -51   0    0.0m   0.0m   0.0m S   0.3   0.0   0:02.33 irq/16-vmwgfx
   2131 hui       20   0  295.4m  42.5m  29.8m S   0.3   0.8   0:15.05 vmtoolsd
   2977 root      20   0   20.2m   3.9m   3.0m R   0.3   0.1   0:13.06 top
```

When QEMU is idle, CPU utilization is about 3%

```
root@ubuntu: /home/hui
top - 11:31:40 up  1:21,  1 user,  load average: 0.53, 0.27, 0.22
Tasks: 310 total,   1 running, 309 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.4 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   5551.8 total,   1791.1 free,   2451.4 used,   1309.4 buff/cache
MiB Swap:   1401.6 total,   1401.6 free,      0.0 used.   2861.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2372 root      20   0 3157.6m   1.4g  49.7m S   3.0  26.1   8:52.52 qemu-system-x86
   1942 hui       20   0 3987.9m 234.1m 101.7m S   0.7   4.2   1:12.49 gnome-shell
    743 root      20   0  242.3m   7.2m   6.2m S   0.3   0.1   0:16.13 vmtoolsd
    938 root      20   0 1468.3m  52.4m  34.4m S   0.3   0.9   0:12.63 containerd
   1772 hui       20   0  298.7m  70.7m  39.6m S   0.3   1.3   1:02.75 Xorg
   2131 hui       20   0  295.4m  42.5m  29.8m S   0.3   0.8   0:15.21 vmtoolsd
   2977 root      20   0   20.2m   3.9m   3.0m R   0.3   0.1   0:13.23 top
```

QEMU is running, CPU utilization is about 104.0%

```
root@ubuntu: /home/hui
top - 11:33:34 up  1:23,  1 user,  load average: 0.29, 0.25, 0.21
Tasks: 310 total,   1 running, 309 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.6 us,  6.8 sy,  0.0 ni, 84.8 id,  2.8 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   5551.8 total,   1792.2 free,   2450.2 used,   1309.4 buff/cache
MiB Swap:   1401.6 total,   1401.6 free,      0.0 used.   2862.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2372 root      20   0 3157.6m   1.4g  49.7m S 104.0  26.1   9:00.45 qemu-system-x86
   1772 hui       20   0  298.7m  70.7m  39.6m S  17.6   1.3   1:04.50 Xorg
   1942 hui       20   0 3987.9m 234.1m 101.7m S  17.6   4.2   1:14.51 gnome-shell
   2343 hui       20   0  805.3m  51.2m  38.0m S   6.0   0.9   0:13.54 gnome-terminal-
     40 root      20   0    0.0m   0.0m   0.0m I   0.3   0.0   0:02.80 kworker/0:2-events
    938 root      20   0 1468.3m  52.4m  34.4m S   0.3   0.9   0:12.93 containerd
   2042 hui       20   0  682.8m  32.2m  21.7m S   0.3   0.6   0:00.87 gsd-media-keys
   2131 hui       20   0  295.4m  42.5m  29.8m S   0.3   0.8   0:15.39 vmtoolsd
   2977 root      20   0   20.2m   3.9m   3.0m R   0.3   0.1   0:13.71 top
```

analysis of the performance data

1. Docker behaves about 3 times better than QEMU when cpu-max-prime is equal.
2. QEMU and Docker take similar CPU resources and memory when running.

FILE I/O test:

DOCKER FILE I/O test:

Random write

min	avg	max	Total time	read	write
0.00	0.06	241.92	30s	0.00	105.96
0.00	0.07	677.34		0.00	100.48
0.00	0.08	1368.34		0.00	87.32
0.00	0.05	112.24		0.00	126.43
0.00	0.05	98.94		0.00	133.26

Sequential write

min	avg	max	Total time	read	write
0.00	0.01	76.35	30s	0.00	643.46
0.00	0.01	129.36		0.00	611.47
0.00	0.01	317.13		0.00	640.08
0.00	0.01	163.63		0.00	662.14
0.00	0.01	75.15		0.00	641.39

Random read/write

min	avg	max	Total time	read	write
0.00	0.03	671.99	30s	121.34	80.89
0.00	0.03	814.12		122.60	81.73
0.00	0.03	350.93		132.38	88.13
0.00	0.03	129.25		124.23	82.82
0.00	0.03	897.61		123.91	82.61

QEMU FILE I/O test:

Random write

min	avg	max	Total time	read	write
0.03	1.03	1470.77	30s	0.00	6.15
0.04	1.69	336.34		0.00	4.01
0.03	1.30	1006.31		0.00	5.21
0.03	1.30	617.32		0.00	4.69
0.03	1.08	466.66		0.00	6.25

Sequential write

min	avg	max	Total time	read	write
0.11	1.09	1436.43	30s	0.00	6.22
0.12	1.09	1545.52		0.00	6.20
0.11	0.66	1617.81		0.00	10.17
0.11	0.97	1607.32		0.00	6.97
0.11	0.99	1605.75		0.00	6.79

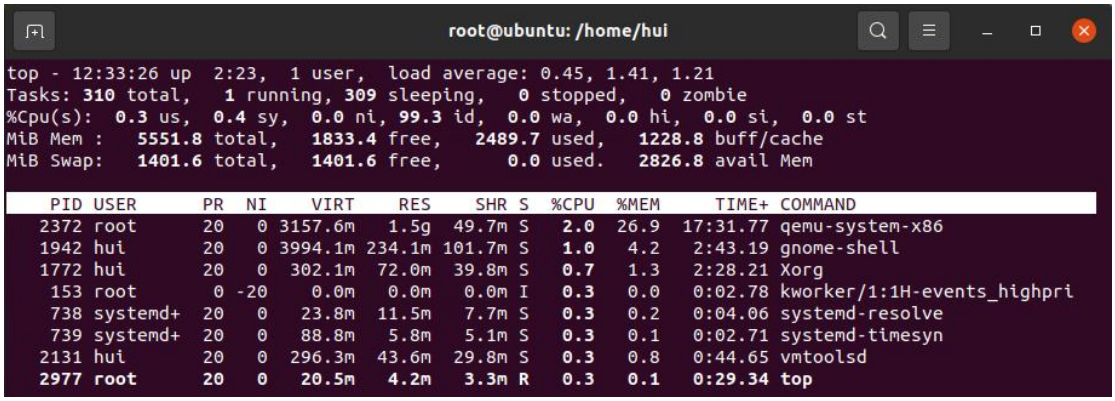
Random read/write

min	avg	max	Total time	read	write
0.01	0.71	2464.17	30s	5.66	3.77
0.02	0.92	904.56		4.33	2.89
0.02	0.93	530.54		4.37	2.91
0.02	1.03	657.82		3.91	2.61
0.02	0.89	450.37		4.40	2.93

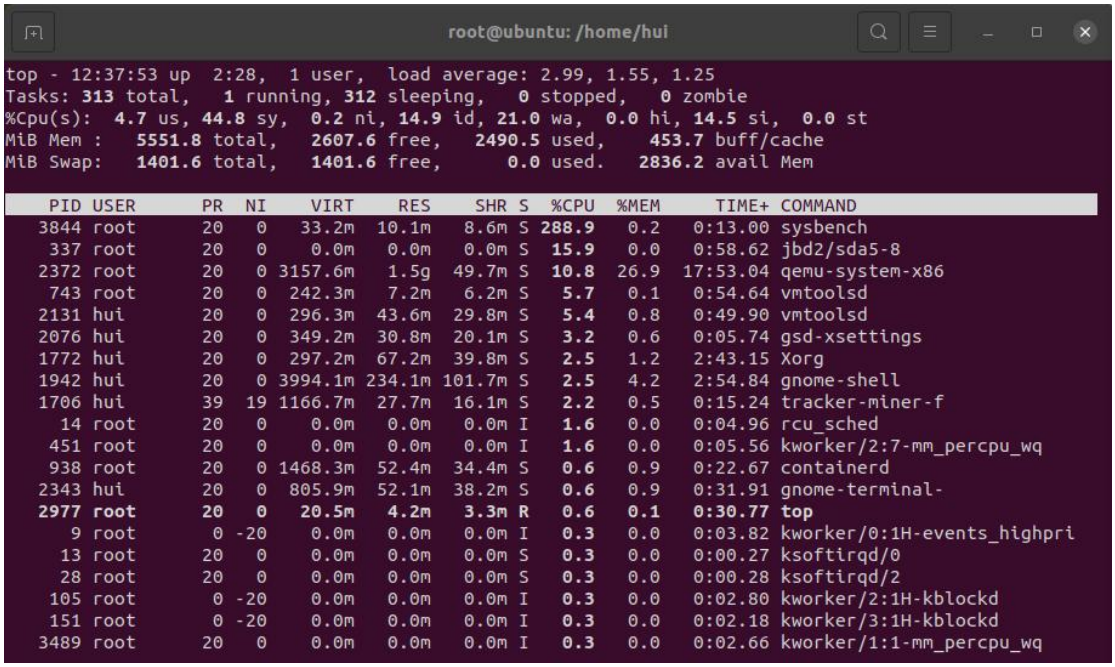
Show CPU utilization by command:

top

When DOCKER is idle, CPU utilization is about 2%



DOCKER is running, CPU utilization is about 288.9%



When QEMU is idle, CPU utilization is about 2.3%

```
root@ubuntu: /home/hui
top - 13:54:32 up  3:44,  1 user,  load average: 0.16, 0.79, 0.84
Tasks: 307 total,   1 running, 306 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.3 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   5551.8 total,   2625.0 free,   2547.8 used,    379.0 buff/cache
MiB Swap:   1401.6 total,   1401.6 free,      0.0 used.   2779.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2372 root      20   0 3237.6m   1.5g  49.7m S   2.3  27.5  27:01.11 qemu-system-x86
   1772 hui       20   0  302.3m  72.6m  40.3m S   0.7   1.3   3:55.52 Xorg
   1942 hui       20   0 3993.3m 234.1m 101.8m S   0.7   4.2   4:17.65 gnome-shell
    938 root      20   0 1468.3m  52.4m  34.4m S   0.3   0.9   0:32.84 containerd
   2131 hui       20   0  293.8m  41.2m  29.8m S   0.3   0.7   1:31.45 vmtoolsd
   2977 root      20   0   20.5m   4.2m   3.3m R   0.3   0.1   0:54.83 top
   3492 root      20   0    0.0m   0.0m   0.0m I   0.3   0.0   0:02.34 kworker/3:0-events
```

When QEMU is running CPU utilization is about 106.6%

```
root@ubuntu: /home/hui
top - 14:00:34 up  3:50,  1 user,  load average: 0.79, 0.54, 0.68
Tasks: 306 total,   1 running, 305 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.7 us, 16.8 sy,  0.0 ni, 76.5 id,  0.3 wa,  0.0 hi,  0.8 si,  0.0 st
MiB Mem :   5551.8 total,   1637.4 free,   3015.7 used,    898.7 buff/cache
MiB Swap:   1401.6 total,   1401.6 free,      0.0 used.   2305.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2372 root      20   0 3239.6m   1.9g  49.7m S 106.6  35.7  28:07.89 qemu-system-x86
   2131 hui       20   0  294.6m  41.7m  29.8m S   7.3   0.8   1:32.98 vmtoolsd
    743 root      20   0  242.3m   7.2m   6.2m S   4.3   0.1   1:40.21 vmtoolsd
   1942 hui       20   0 3993.3m 234.1m 101.8m S   3.7   4.2   4:23.25 gnome-shell
   1772 hui       20   0  302.3m  72.6m  40.3m S   2.3   1.3   4:01.06 Xorg
    436 root     -51   0    0.0m   0.0m   0.0m S   0.3   0.0   0:08.49 irq/16-vmwgfx
    938 root      20   0 1468.3m  52.4m  34.4m S   0.3   0.9   0:33.57 containerd
   2977 root      20   0   20.5m   4.2m   3.3m R   0.3   0.1   0:56.14 top
   3492 root      20   0    0.0m   0.0m   0.0m I   0.3   0.0   0:02.45 kworker/3:0-events
```

analysis of the performance data

1. When running Docker, the speed of sequential write is far more quicker than random write and random read/write
2. The speed of random write of Docker is 20 times faster than QEMU
3. The speed of sequential write of Docker is 100 times faster than QEMU
4. The speed of random read/write of Docker is 20 times faster than QEMU
5. QEMU takes two times more CPU resources than QEMU

Overall, docker containers are much faster than QEMU

Git repository: https://github.com/Burgerrrr/Tinghui_Zhang_COEN241HW