

# COEN 241: Cloud Computing - Homework 1

Author: Fangfang Lin

Student ID: W1630994

## 1. Experiment Environment

### a. Host Machine(Physical Machine)

Detailed configurations (CPU, Mem, etc...) of your experimental setup:(5 points)

Equipment Name	DESKTOP-2BHUVHG
Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
Operating System	64-bit Windows 10 system
RAM	8.00 GB
Disk	120 GB

Because the host system I used is Windows 10, I followed the assignment instructions to install some necessary modules to set up the environment for QEMU and Docker.

#### i. QEMU

I installed WSL and VcXsrv to connect Ubuntu to the display. The instructions are [here](#).

#### ii. Docker

I installed the Docker Desktop and updated the WSL to WSL 2 to support the Docker Desktop. The instructions are [here](#).

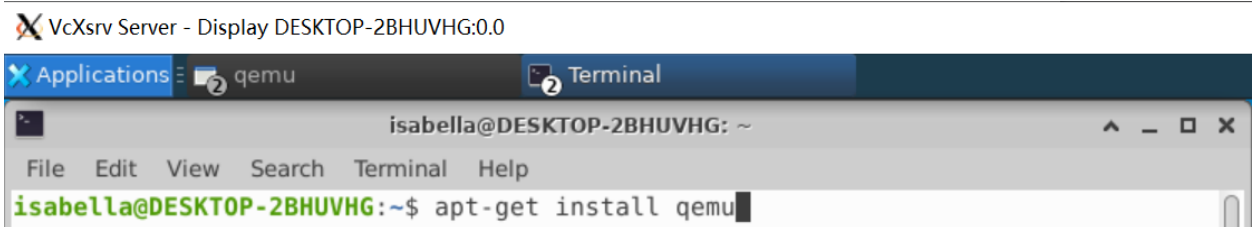
### b. Guest OS & Docker

Main steps to enable a QEMU VM and detailed QEMU commands(10 points)

I. Download the appropriate Ubuntu 16.04 server ISO image from the [link](#).

II. Install QEMU and create QEMU image

Although I use Windows system, I can follow the Ubuntu installation instructions to install the QEMU and create QEMU since I already installed the WSL and VcXsrv



```
VcXsrv Server - Display DESKTOP-2BHUVHG:0.0
Applications | qemu | Terminal
isabella@DESKTOP-2BHUVHG: /mnt/d/Coen241/HW1
File Edit View Search Terminal Help
isabella@DESKTOP-2BHUVHG:~$ cd /mnt/d/Coen241/HW1
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ sudo qemu-img create ubuntu.img 10G
-f qcow
```

```
isabella@DESKTOP-2BHUVHG: /mnt/d/Coen241/HW1$ ls
hw1  ubuntu-16.04.7-server-amd64.iso  ubuntu.img
```

III. Given the image, install the VM using the command below (which takes the iso file as a “cdrom” and the qemu image as a “hard disk”):

```
VcXsrv Server - Display DESKTOP-2BHUVHG:0.0
Applications | qemu | Terminal
isabella@DESKTOP-2BHUVHG: /mnt/d/Coen241/HW1
File Edit View Search Terminal Help
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ sudo qemu-system-x86_64 -hda ubuntu
.img -m 2048 -smp cores=1,threads=2 -boot strict=on
```

### VM configurations

Based on the above command, the virtual machine’s configuration is controlled by the arguments, such as ‘-m’ to control the memory size, ‘-smp’ to set the number of cores or threads. So the created virtual machine will have 2 GB memory, 1 cores, and 2 threads per core.

### main steps to enable the Docker container(10 points)

I. Install Docker Desktop and upgrade the WSL 1 to 2

II. Pull the Ubuntu 16.4 image from Docker Hub

```
isabella@DESKTOP-2BHUVHG: /mnt/d/Coen241/HW1$ docker pull ubuntu:16.04
[2023-02-04T03:47:29.365541100Z][docker-credential-desktop.exe][W] Windows version might not be up-to-date: The system c
annot find the file specified.
16.04: Pulling from library/ubuntu
58690f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:1f1a2d56deld604801a9671f301190704c25d604a416f59e03c04f5c6ffee0d6
Status: Downloaded newer image for ubuntu:16.04
docker.io/library/ubuntu:16.04
```

```
isabella@DESKTOP-2BHUVHG: /mnt/d/Coen241/HW1$ docker images
REPOSITORY    TAG          IMAGE ID      CREATED      SIZE
ubuntu        16.04       b6f507652425  17 months ago  135MB
```

III. Using the base ubuntu image, create a container and install the necessary libraries/tools, such as sysbench and git.

```
isabella@DESKTOP-2BHUVHG: /mnt/d/Coen241/HW1$ docker run -it --name=ubuntu ubuntu:16.04
root@f531849f43ee:/# apt-get update
```

#### IV. Exit from the container and create my own image based on this container

```
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker commit -m "Fangfang Lin's own ubuntu image" -a "Isabella" ubuntu isa
bella/sysbench
sha256:cef63893c7c052b797ed3cabd9c21a312efe516f3c8cf7433351731c7117fe36
```

The return line is the created image's ID

#### V. History of created image:

```
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker history isabella/sysbench
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
cef63893c7c0   About a minute ago  /bin/bash          177MB     Fangfang Lin's own ubuntu
image
b6f507652425   17 months ago     /bin/sh -c #(nop)  CMD ["/bin/bash"]    0B
<missing>      17 months ago     /bin/sh -c mkdir -p /run/systemd && echo 'do... 7B
<missing>      17 months ago     /bin/sh -c rm -rf /var/lib/apt/lists/*        0B
<missing>      17 months ago     /bin/sh -c set -xe  && echo '#!/bin/sh' > /... 745B
<missing>      17 months ago     /bin/sh -c #(nop)  ADD file:11b425d4c08e81a3e... 135MB
```

#### VI. Useful command

Check the containers I have

```
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS      NAMES
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS      NAMES
f531849f43ee   ubuntu:16.04   "/bin/bash"             9 minutes ago  Exited (0)    4 minutes ago          ubuntu
```

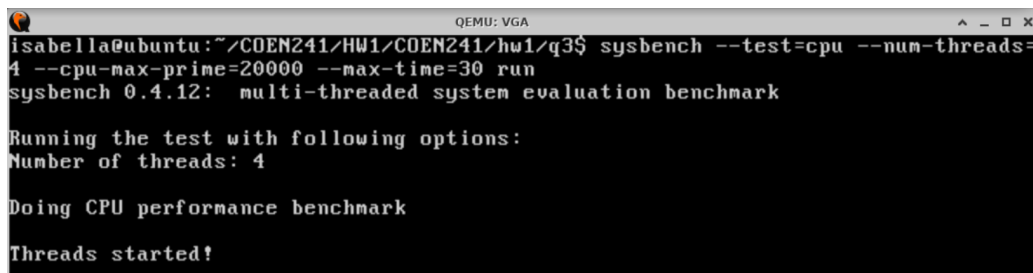
Some containers operations

```
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker start ubuntu
ubuntu
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker stop ubuntu
ubuntu
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker restart ubuntu
ubuntu
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker attach ubuntu
root@f531849f43ee:/#
isabella@DESKTOP-2BHUVHG:/mnt/d/Coen241/HW1$ docker rm ubuntu
```

2. Experiment(Present how you conduct your measurements in three different scenarios for each virtualization technology:20 points)

a. **Proof of experiment. Include screen snapshots of your Docker and QEMU running environments for each experiment:(5 points)**

The following snapshot is the running process on QEMU



```
QEMU: VGA
isabella@ubuntu:~/COEN241/HW1/COEN241/hw1/q3$ sysbench --test=cpu --num-threads=
4 --cpu-max-prime=20000 --max-time=30 run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4

Doing CPU performance benchmark

Threads started!
```

```

Doing CPU performance benchmark

Threads started!
Time limit exceeded, exiting...
(last message repeated 3 times)
Done.

Maximum prime number checked in CPU test: 20000

Test execution summary:
total time:                      30.0096s
total number of events:          9520
total time taken by event execution: 119.8135
per-request statistics:
    min:                          5.59ms
    avg:                          12.59ms
    max:                          74.15ms
    approx. 95 percentile:        22.62ms

Threads fairness:
events (avg/stddev):              2380.0000/12.25
execution time (avg/stddev):      29.9534/0.01

isabella@ubuntu:~/COEN241/HW1/COEN241/hw1/q3$ _

```

On Docker, the following screenshot shows the proof:

The screenshot shows a Docker container named 'dumUbuntu' with ID '42260a834242' running the 'isabella/sysbench' image. The terminal output shows the execution of the sysbench CPU benchmark with the following options: `--test=cpu --num-threads=4 --cpu-max-prime=20000 --max-time=30 run`. The output includes the number of threads (4), the benchmark results, and the test execution summary.

```

root@42260a834242: /home/COEN241/hw1/docker
root@42260a834242: /home/COEN241/hw1/docker# sysbench --test=cpu --num-threads=4 --cpu-max-prime=20000 --max-time=30 run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000

Test execution summary:
total time:                      7.3060s
total number of events:          10000
total time taken by event execution: 29.2033
per-request statistics:
    min:                          2.43ms
    avg:                          2.92ms
    max:                          30.82ms
    approx. 95 percentile:        3.71ms

Threads fairness:
events (avg/stddev):              2500.0000/8.86
execution time (avg/stddev):      7.3008/0.01

root@42260a834242: /home/COEN241/hw1/docker#

```

**b. Shell scripts for running the experiment:(20 points)**

CPU test scripts:

I totally have three different scenarios for CPU testing, each scenario has a different max prime. Let the sysbench verify different prime numbers to test the CPU workload of QEMU VM and Docker container.

I used the same shell scripts to test both environments, the following shell script shows the source code of the CPU test.

```
#!/bin/bash

dir=q1_cpu_test_out1.txt
if [ ! -f $dir ]
then
    touch $dir
fi

echo "Test 1 =====> num of threads 4; max prime 20000" >> $dir

for((i=1;i<6;i++))
do
    echo "$i iteration" >> $dir
    sysbench --test=cpu --num-threads=4 --cpu-max-prime=20000 --max-time=30 run >> $dir
done
```

```
sysbench --test=cpu --num-threads=4 --cpu-max-prime=30000 --max-time=30 run >> $dir
```

This statement is the most important part, it sets the configuration of the CPU test. First the “--test=cpu” means we want to test the CPU. The second one “--num-threads” sets how many threads to do the CPU test. “--cpu-max-prime” is the max prime. “--max-time=30” is setting the limited time to execute.

My goal is testing the CPU using different max prime, so I just need to change the --cpu-max-prime attribute to other values. Here, I set the value from 20000 to 30000.

#### Fileio Test

Fileio test also has three different scenarios. The difference is the total size of file it will operate. I set the total size of the file from 2G to 4G.

And here is the shell script:

```
#!/bin/bash

dir=q1_fileio_test_out1.txt
if [ ! -f $dir ]
then
    touch $dir
fi

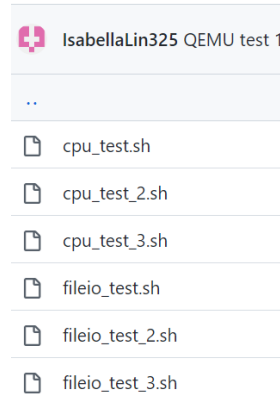
echo "Test 1 =====> num of threads 16, total file size 2G" >> $dir

for((i=1;i<6;i++))
do
    echo "$i iteration" >> $dir
    sysbench --test=fileio --num-threads=16 --max-requests=0 --file-total-size=2G --file-test-mode=rndrw --max-time=30 prepare
    sysbench --test=fileio --num-threads=16 --max-requests=0 --file-total-size=2G --file-test-mode=rndrw --max-time=30 run >> $dir
    sysbench --test=fileio --num-threads=16 --max-requests=0 --file-total-size=2G --file-test-mode=rndrw --max-time=30 cleanup
done
```

When using fileio, you will need to create a set of test files to work on. It is recommended that the size is larger than the available memory to ensure that file caching does not influence the workload too much. As you can see, for the fileio test, we need to do three steps: preparing, running, and cleaning up. The “prepare” statement will generate the temporary files for testing. And the running is executing the testing using the test mode. Here, I used “rndrw” which stands

for random read/write. “--max-time” is the same argument as the CPU test. The duration of the test is given through the “--max-time” option (in seconds). After the testing is finished, we need the “cleanup” statement to delete the temporary files.

I just posted the example shell scripts here, all source shell scripts are on my git repository, I will attach the git repository information at the end of the report.



### c. Performance Analysis

CPU test:

Test 1: Max Prime 20000

Scenario	Total num of events(Test execution summary)	events (avg/stddev, threads fairness)
QEMU(2G, 1core, 2threads)	8548	2137.5/17.174
QEMU(4G, 2cores, 2threads)	8771	2192.9/8.21
QEMU(4G, 4cores, 8threads)	10000	2500/13.658
Docker Container	10000	2500/9.286

Test 2: Max Prime 30000

Scenario	Total num of events(Test execution summary)	events (avg/stddev, threads fairness)
QEMU(2G, 1core, 2threads)	5115	1278.85/8.738
QEMU(4G, 2cores, 2threads)	5189.8	1297.45/4.688
QEMU(4G, 4cores, 8threads)	6851.6	1712.9/12.268

Docker Container	10000	2500/9.324
------------------	-------	------------

### Test 3: Max Prime 40000

Scenario	Total num of events(Test execution summary)	events (avg/stddev, threads fairness)
QEMU(2G, 1core, 2threads)	3463.8	885.95/8.298
QEMU(4G, 2cores, 2threads)	3660.6	915.15/3.68
QEMU(4G, 4cores, 8threads)	4723.6	1180.9/5.992
Docker Container	10000	2500/5.168

(Notes: **all the data are average results after five iterations.** I ran each test five times. And the original results are on git repository.)

Total num of events(Test execution summary) stands for all threads completed a total of  $n$  events within  $x$  seconds. And events (avg/stddev, threads fairness) stands for each thread completes  $m$  events, with a standard deviation of  $std$ .

stddev (standard deviation): In the same time, whether the number of prime number calculations completed by multiple threads is stable, if the value is lower, it means that the results of multiple threads are closer (that is, more stable).

If there are 2 servers for CPU performance comparison, when the upper limit of prime numbers is consistent with the number of threads:

- (1) At the same time, compare who has more events
- (2) For the same event, whoever takes less time compares
- (3) The time and event are the same, which one is lower compared to stddev (standard deviation)

Based on this rules, just for comparing QEMUs, I found the trend of the result is correct. QEMU that has more memory, cores, and threads will have better performance. And Let's compare the QEMU with Docker container, I can find that the performance of Docker container is better than all QEMU VMs. It's because the docker container is close to physical machine.

### Fileio Test

#### Test 1: Total Size 2G

Scenario	I/O Throughput	events (avg/stddev, threads fairness)	Latency
QEMU(2G, 1core, 2threads)	21883	1367.6875/84.576	5.054
QEMU(4G, 2cores, 2threads)	16572	1035.75/201.56	1.1

QEMU(4G, 4cores, 8threads)	21618	1351.1125/198.7	0.618
Docker Container	261961.2	16372.575	0.01

Test 2: Total Size 3G

Scenario	I/O Throughput	events (avg/stddev, threads fairness)	Latency
QEMU(2G, 1core, 2threads)	18888	1180.5/54.984	8.368
QEMU(4G, 2cores, 2threads)	26030.8	1626.925/163.57	1.174
QEMU(4G, 4cores, 8threads)	20833.6	1302.1/208.03	0.854
Docker Container	254970	15935.52494/522.32	0.02

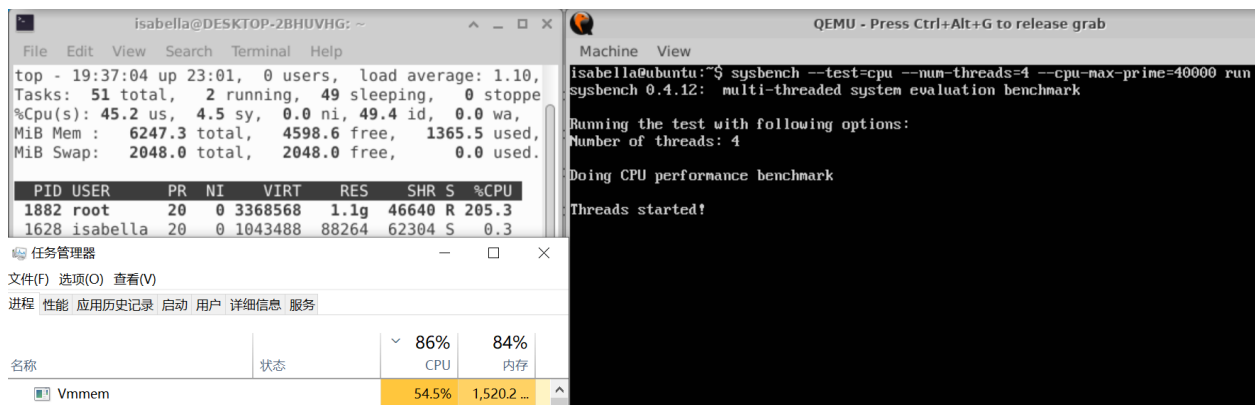
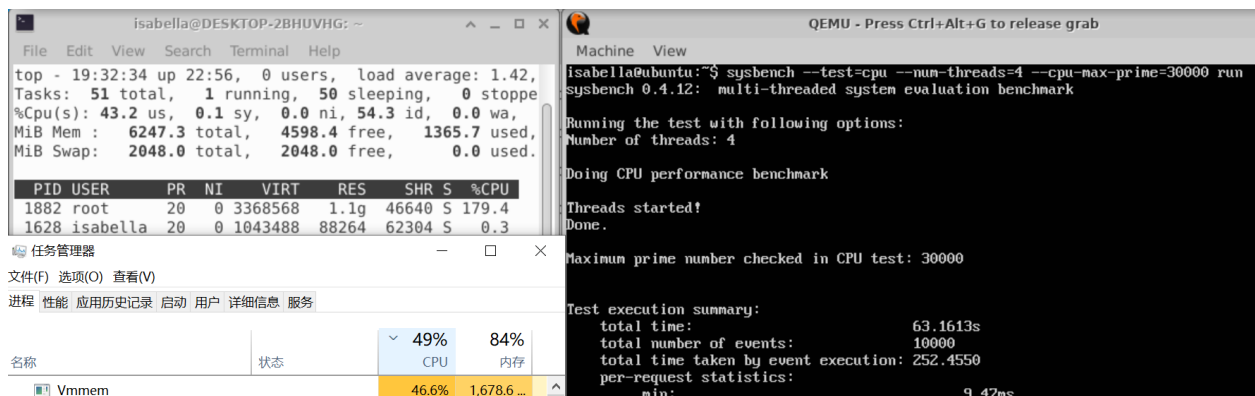
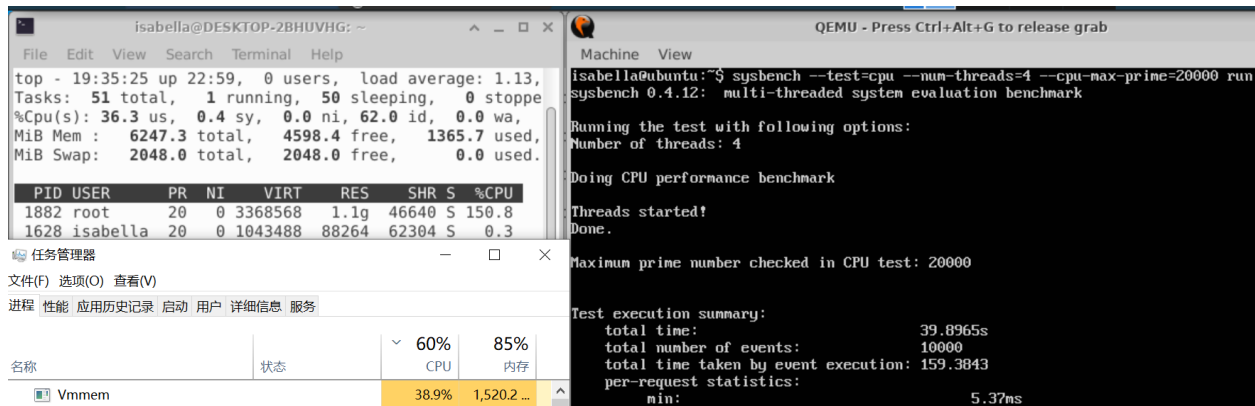
Test 3: Total Size 4G

Scenario	I/O Throughput	events (avg/stddev, threads fairness)	Latency(avg)
QEMU(2G, 1core, 2threads)	18237.6	1139.85/50.618	9.586
QEMU(4G, 2cores, 2threads)	22308	1394.25/81.83	4.56
QEMU(4G, 4cores, 8threads)	15313.8	957.1125/127.346	5.93
Docker Container	220298	13768.625/337.108	0.088

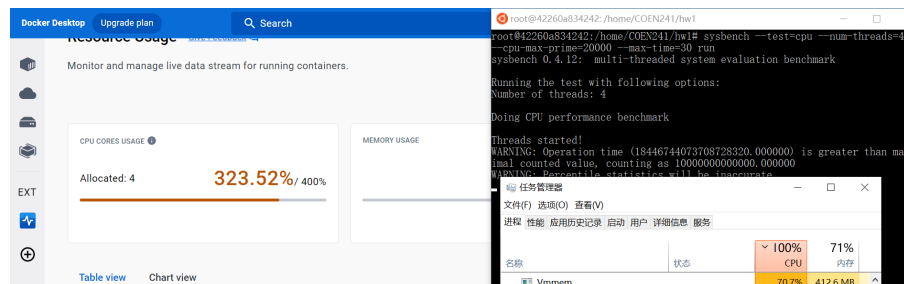
For Fileio test, I found there are some weird points for QEMU. QEMU scenario 1 has less memory, less cores, and less threads, so whatever I/O throughput or latency, it will be worse than other two. However, QEMU scenario 2 and 3 have the same memory but different cores and threads, the performance of scenario 2 will be better than scenario 3. Another normal finding is scenario 3 has less latency than other scenarios since it has more cores and threads. The same as CPU test, Docker container is better than the QEMU VMs since the docker container can use more host I/O to boot the I/O operations.

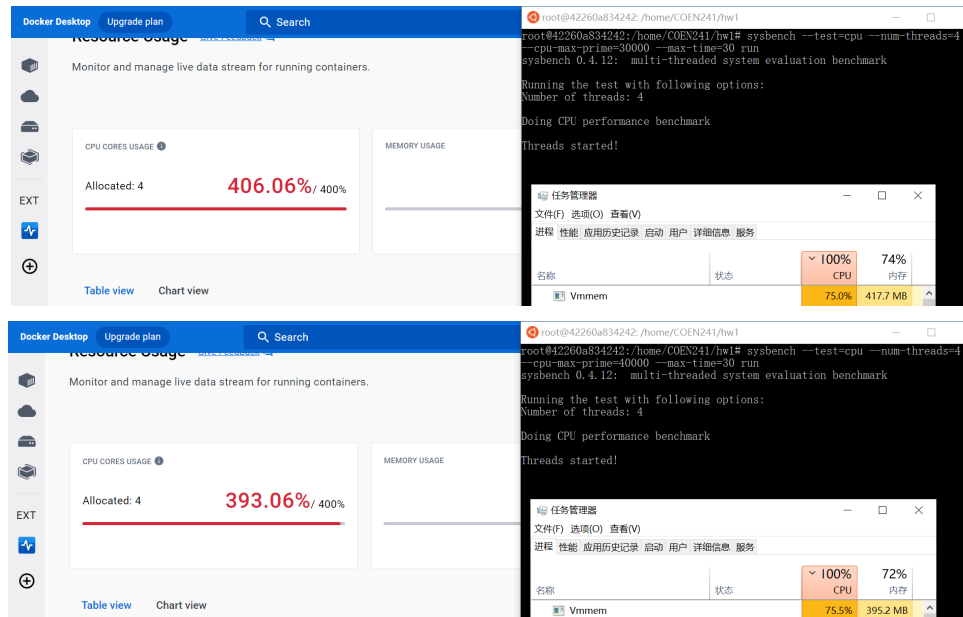
d. how I use performance tools to collect performance data(10 points)





The performance analysis tools I used here are the “top” command line and the windows task manager. “Top” tool and the task manager can show the details of all processes such as the PID, **user-level and kernel-level CPU utilization, and I/O throughput**. I mainly used these two tools to monitor the QEMU virtual machine's performance. The snapshots are the proofs.





### 3. Git Repository Information

Git Repository Http URL: <https://github.com/IsabellaLin325/COEN241.git>

Git User Name: IsabellaLin325

List of Contents:

- Four folders for shell scripts and output
- One homework report