Report of HW1 - System vs OS Virtualization

Configurations of experimental setup

CPU: 1.6 GHz Dual-Core Intel Core i5
Memory: 8GB
OS: macOS Big Sur 11.5.2

Steps to enable a QEMU VM

First, download the appropriate Ubuntu 20.04 server ISO image.

Then, Install Homebrew:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Then, Install QEMU using the homebrew method:

```
brew install qemu
```

Then, cd into project environment and create the QEMU image:

```
sudo qemu-img create ubuntu.img 10G
```

Then, install the VM:

```
sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ./ubuntu-20.04.3-live-server-amd64.iso -m 1536
```

After installation, reboot ubuntu. Or using following command to boot ubuntu:

```
sudo qemu-system-x86_64 -hda ubuntu.img -boot c -cdrom ./ubuntu-20.04.3-live-server-amd64.iso -m 1536
```

The VM has 10G disk space and 1536M memory.

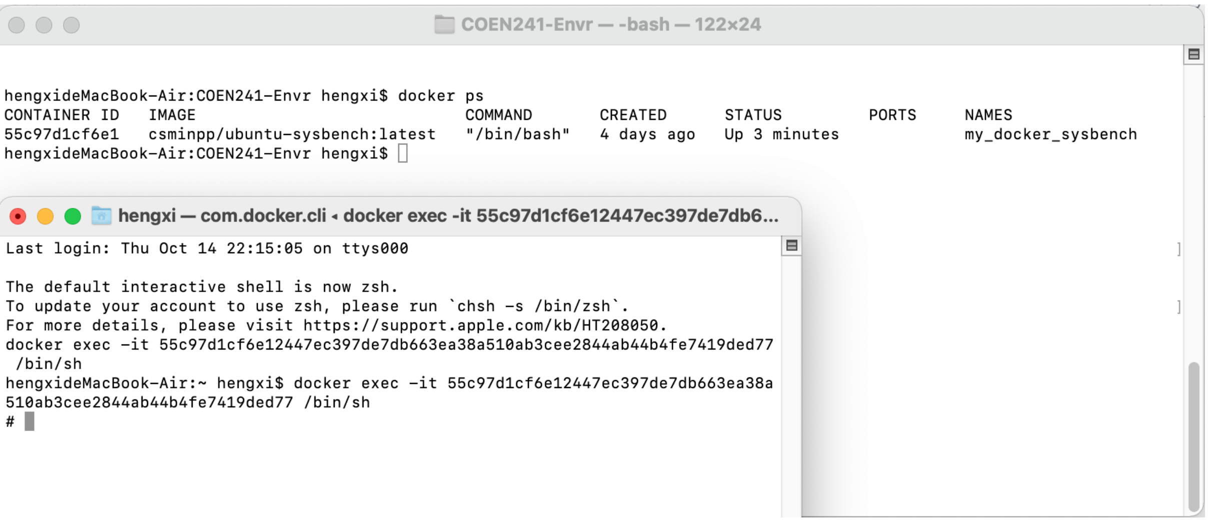
Steps to enable a Docker container

Install Docker Desktop which includes Docker Engine, Docker CLI client, Docker Compose. Use `docker pull` to download images from Docker Hub. With Docker Desktop it's easy to run, enter, close or remove a container, which can also be done by commands.

`docker run` is to run a container from an image.
`docker start` is to run an existing container.
`docker exec` is to enter a running container.
`docker stop` is to stop a running container.
`docker rm` is to delete a container.
`docker images` is to list all images.
`docker ps` is to list all running containers.

Proof of experiment

Docker running environment:

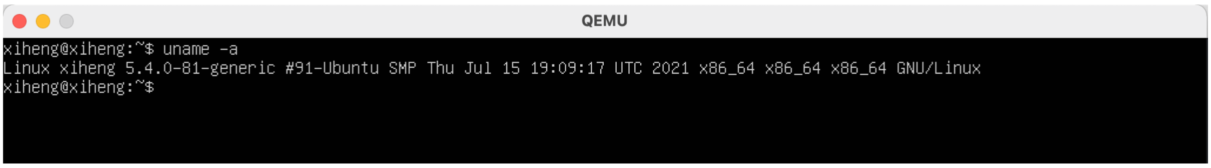

COEN241-Envr — -bash — 122×24
hengxi — com.docker.cli ◂ docker exec -it 55c97d1cf6e12447ec397de7db6...

```
hengxideMacBook-Air:COEN241-Envr hengxi$ docker ps
CONTAINER ID   IMAGE                           COMMAND       CREATED      STATUS         PORTS     NAMES
55c97d1cf6e1   csminpp/ubuntu-sysbench:latest   "/bin/bash"   4 days ago   Up 3 minutes             my_docker_sysbench
hengxideMacBook-Air:COEN241-Envr hengxi$ 
```

```
Last login: Thu Oct 14 22:15:05 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
docker exec -it 55c97d1cf6e12447ec397de7db663ea38a510ab3cee2844ab44b4fe7419ded77
 /bin/sh
hengxideMacBook-Air:~ hengxi$ docker exec -it 55c97d1cf6e12447ec397de7db663ea38a
510ab3cee2844ab44b4fe7419ded77 /bin/sh
# 
```

QEMU running environment:



```
xiheng@xiheng:~$ uname -a
Linux xiheng 5.4.0-81-generic #91-Ubuntu SMP Thu Jul 15 19:09:17 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
xiheng@xiheng:~$
```

Conduct measurements in three different scenarios

For both Docker container and QEMU VM, follow the same steps to conduct measurements:

First, `git clone` from GitHub;

Then, `cd` into the file;

Then, `chmod +777` to give the permission to `run_cpu1.sh, run_cpu2.sh, run_cpu3.sh, run_fileio1.sh, run_fileio2.sh, run_fileio3.sh`;

Run `run_cpu1.sh` and `run_fileio1.sh` to conduct measurements in the first scenario.

For this scenario, arguments are set as:

`--cpu-max-prime=20000` for CPU in sysbench and

`--file-total-size=1G` for I/O in sysbench.

Run `run_cpu2.sh` and `run_fileio2.sh` to conduct measurements in the second scenario;

For this scenario, arguments are set as:
`--cpu-max-prime=22000` for CPU in sysbench and
`--file-total-size=2G` for I/O in sysbench.

Run `run_cpu3.sh` and `run_fileio3.sh` to conduct measurements in the third scenario.

For this scenario, arguments are set as:
`--cpu-max-prime=24000` for CPU in sysbench and
`--file-total-size=3G` for I/O in sysbench.

And the sysbench printing results are saved into `test_cpu1.txt, test_cpu2.txt, test_cpu3.txt, test_fileio1.txt, test_fileio2.txt, test_fileio3.txt`;

Shell scripts

run_cpu1.sh

```
#!/bin/bash
chmod +777 ./test_cpu1.sh &&
./test_cpu1.sh > ./test_cpu1.txt
```

run_cpu2.sh

```
#!/bin/bash
chmod +777 ./test_cpu2.sh &&
./test_cpu2.sh > ./test_cpu2.txt
```

run_cpu3.sh

```
#!/bin/bash
chmod +777 ./test_cpu3.sh &&
./test_cpu3.sh > ./test_cpu3.txt
```

run_fileio1.sh

```
#!/bin/bash
chmod +777 ./test_fileio1.sh &&
./test_fileio1.sh > ./test_fileio1.txt
```

run_fileio2.sh

```
#!/bin/bash
chmod +777 ./test_fileio2.sh &&
./test_fileio2.sh > ./test_fileio2.txt
```

run_fileio3.sh

```
#!/bin/bash
chmod +777 ./test_fileio3.sh &&
./test_fileio3.sh > ./test_fileio3.txt
```

test_cpu1.sh

```
#!/bin/bash
for((i=0;i<5;i++))
do
  sysbench --test=cpu --cpu-max-prime=20000 run
done
```

test_cpu2.sh

```
#!/bin/bash
for((i=0;i<5;i++))
do
  sysbench --test=cpu --cpu-max-prime=22000 run
done
```

test_cpu3.sh

```
#!/bin/bash
for((i=0;i<5;i++))
do
  sysbench --test=cpu --cpu-max-prime=24000 run
done
```

test_fileio1.sh

```
#!/bin/bash
sysbench --num-threads=16 --test=fileio --file-total-size=1G
--file-test-mode=rndrw prepare
for((i=0;i<5;i++))
do
  sysbench --num-threads=16 --test=fileio --file-total-size=1G
--file-test-mode=rndrw run
done
sysbench --num-threads=16 --test=fileio --file-total-size=1G
--file-test-mode=rndrw cleanup
```

test_fileio2.sh

```
#!/bin/bash
sysbench --num-threads=16 --test=fileio --file-total-size=2G
--file-test-mode=rndrw prepare
for((i=0;i<5;i++))
do
```

```
  sysbench --num-threads=16 --test=fileio --file-total-size=2G
--file-test-mode=rndrw run
done
sysbench --num-threads=16 --test=fileio --file-total-size=2G
--file-test-mode=rndrw cleanup
```

test_fileio3.sh

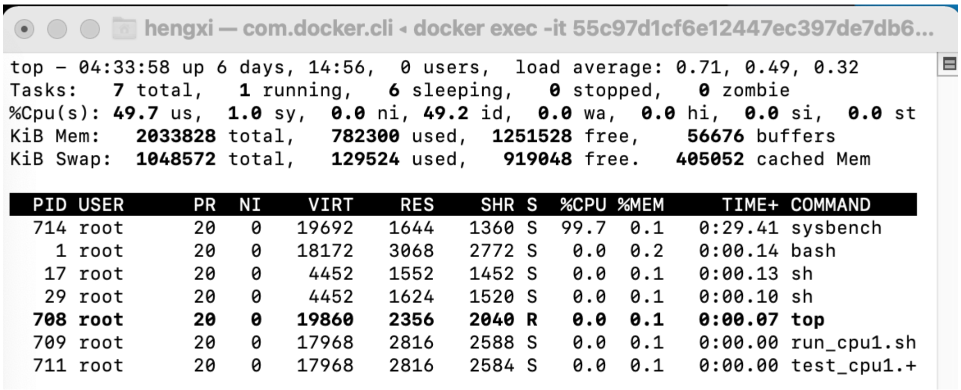
```
#!/bin/bash
sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw prepare
for((i=0;i<5;i++))
do
  sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw run
done
sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw cleanup
```

Performance tools

For CPU:

Use the top as the performance tool.

In Docker container:

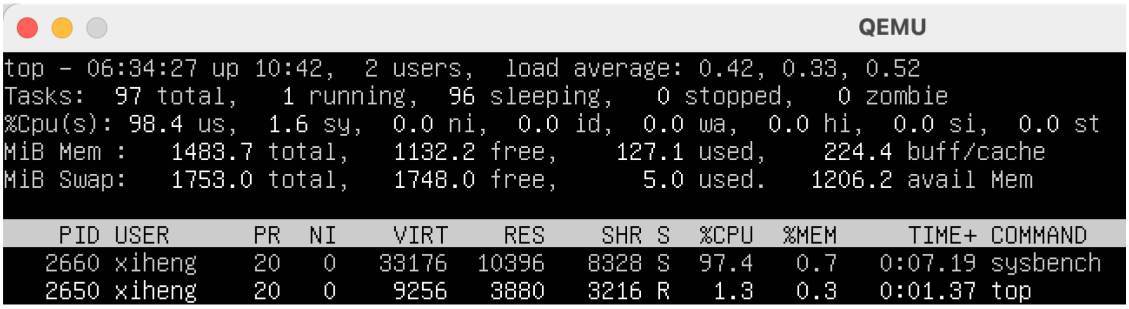

hengxi — com.docker.cli ◂ docker exec -it 55c97d1cf6e12447ec397de7db6...

```
top - 04:33:58 up 6 days, 14:56,  0 users,  load average: 0.71, 0.49, 0.32
Tasks:   7 total,   1 running,   6 sleeping,   0 stopped,   0 zombie
%Cpu(s): 49.7 us,  1.0 sy,  0.0 ni, 49.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2033828 total,   782300 used,  1251528 free,    56676 buffers
KiB Swap:  1048572 total,   129524 used,   919048 free.   405052 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  714 root      20   0   19692   1644   1360 S  99.7  0.1   0:29.41 sysbench
    1 root      20   0   18172   3068   2772 S   0.0  0.2   0:00.14 bash
   17 root      20   0    4452   1552   1452 S   0.0  0.1   0:00.13 sh
   29 root      20   0    4452   1624   1520 S   0.0  0.1   0:00.10 sh
  708 root      20   0   19860   2356   2040 R   0.0  0.1   0:00.07 top
  709 root      20   0   17968   2816   2588 S   0.0  0.1   0:00.00 run_cpu1.sh
  711 root      20   0   17968   2816   2584 S   0.0  0.1   0:00.00 test_cpu1.+
```

User-level CPU utilization: 49.7%
Kernel-level CPU utilization: 1.0%

In QEMU VM:



```
top - 06:34:27 up 10:42,  2 users,  load average: 0.42, 0.33, 0.52
Tasks:  97 total,   1 running,  96 sleeping,   0 stopped,   0 zombie
%Cpu(s): 98.4 us,  1.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1483.7 total,   1132.2 free,    127.1 used,    224.4 buff/cache
MiB Swap:   1753.0 total,   1748.0 free,      5.0 used.   1206.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2660 xiheng    20   0   33176  10396   8328 S  97.4   0.7   0:07.19 sysbench
   2650 xiheng    20   0    9256   3880   3216 R   1.3   0.3   0:01.37 top
```

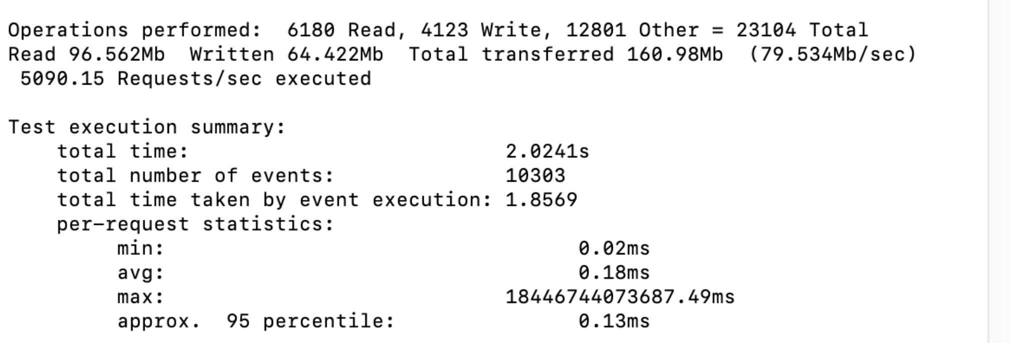
User-level CPU utilization: 98.4%
Kernel-level CPU utilization: 1.6%

For IO:

Use the sysbench’s output and top as the performance tools.

In Docker container:

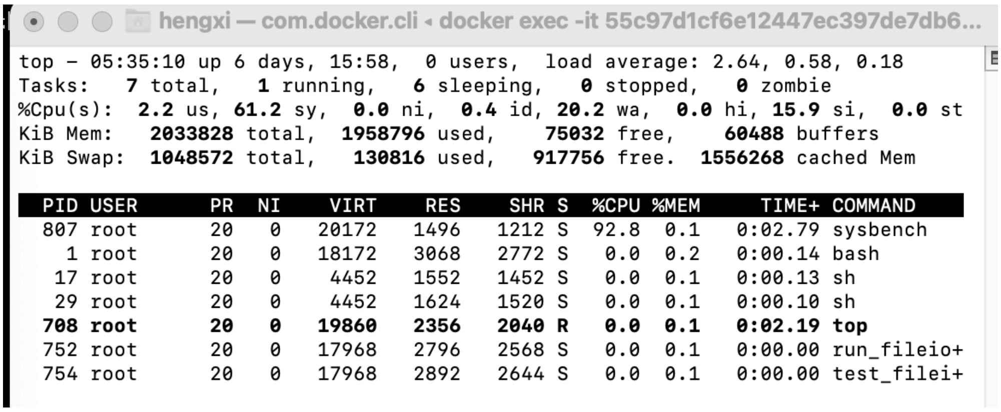
I/O’s throughput, latency are shown below:



```
Operations performed:  6180 Read, 4123 Write, 12801 Other = 23104 Total
Read 96.562Mb  Written 64.422Mb  Total transferred 160.98Mb  (79.534Mb/sec)
 5090.15 Requests/sec executed

Test execution summary:
    total time:                          2.0241s
    total number of events:              10303
    total time taken by event execution: 1.8569
    per-request statistics:
         min:                                  0.02ms
         avg:                                  0.18ms
         max:                            18446744073687.49ms
         approx.  95 percentile:               0.13ms
```

I/O's disk utilization is shown below:

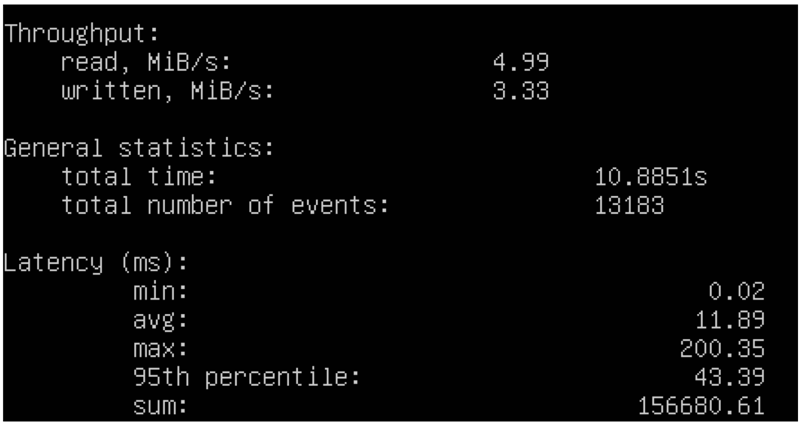

hengxi — com.docker.cli ◂ docker exec -it 55c97d1cf6e12447ec397de7db6...

```
top - 05:35:10 up 6 days, 15:58,  0 users,  load average: 2.64, 0.58, 0.18
Tasks:   7 total,   1 running,   6 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.2 us, 61.2 sy,  0.0 ni,  0.4 id, 20.2 wa,  0.0 hi, 15.9 si,  0.0 st
KiB Mem:  2033828 total,  1958796 used,    75032 free,    60488 buffers
KiB Swap: 1048572 total,   130816 used,   917756 free.  1556268 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  807 root      20   0   20172   1496   1212 S  92.8  0.1   0:02.79 sysbench
    1 root      20   0   18172   3068   2772 S   0.0  0.2   0:00.14 bash
   17 root      20   0    4452   1552   1452 S   0.0  0.1   0:00.13 sh
   29 root      20   0    4452   1624   1520 S   0.0  0.1   0:00.10 sh
  708 root      20   0   19860   2356   2040 R   0.0  0.1   0:02.19 top
  752 root      20   0   17968   2796   2568 S   0.0  0.1   0:00.00 run_fileio+
  754 root      20   0   17968   2892   2644 S   0.0  0.1   0:00.00 test_filei+
```

In QEMU VM:

I/O's throughput, latency are shown below:

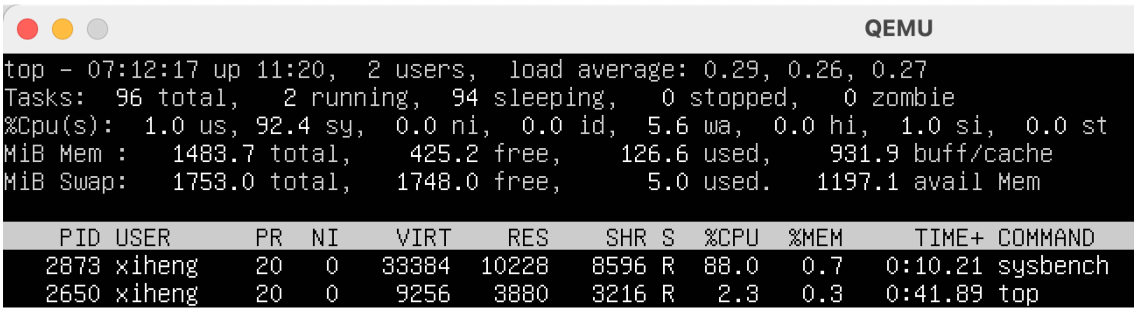


```
Throughput:
    read, MiB/s:                  4.99
    written, MiB/s:               3.33

General statistics:
    total time:                          10.8851s
    total number of events:              13183

Latency (ms):
         min:                                    0.02
         avg:                                   11.89
         max:                                  200.35
         95th percentile:                       43.39
         sum:                               156680.61
```

I/O's disk utilization is shown below:



```
top - 07:12:17 up 11:20,  2 users,  load average: 0.29, 0.26, 0.27
Tasks:  96 total,   2 running,  94 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us, 92.4 sy,  0.0 ni,  0.0 id,  5.6 wa,  0.0 hi,  1.0 si,  0.0 st
MiB Mem :  1483.7 total,   425.2 free,   126.6 used,   931.9 buff/cache
MiB Swap:  1753.0 total,  1748.0 free,     5.0 used.  1197.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2873 xiheng    20   0   33384  10228   8596 R  88.0   0.7   0:10.21 sysbench
 2650 xiheng    20   0    9256   3880   3216 R   2.3   0.3   0:41.89 top
```

Presentation and Analysis of performance data

For CPU:

In Docker container:

scenario	round	min/ms	average/ms	approx. 95 percentile/ms	std	number of events	time of events/s	events per second	average events per second for each scenario
--cpu-max-prime=20000	1	2.57	3.65	5.16	0	10000	36.4722	274.1814313	255.4609447
	2	2.61	4.56	7.91	0	10000	45.6046	219.2761257	
	3	2.84	3.84	4.87	0	10000	38.4086	260.3583572	
	4	2.73	3.8	4.77	0	10000	37.9642	263.406051	
	5	2.87	3.84	4.78	0	10000	38.4493	260.0827583	
--cpu-max-prime=22000	1	2.98	4.47	7.09	0	10000	44.7238	223.5945962	223.7007596
	2	3.14	4.35	5.45	0	10000	43.471	230.0384164	
	3	3.13	4.55	6.12	0	10000	45.5194	219.6865512	
	4	3.37	4.65	6.12	0	10000	46.46	215.2389152	
	5	3.24	4.35	5.31	0	10000	43.4886	229.945319	
--cpu-max-prime=24000	1	3.35	4.77	6.26	0	10000	47.7051	209.6211935	183.9042917
	2	3.49	6.06	10.35	0	10000	60.5863	165.0538158	
	3	3.61	6.33	10.68	0	10000	63.2868	158.0108332	
	4	3.78	5.29	7.44	0	10000	52.9184	188.9701881	
	5	3.47	5.05	6.69	0	10000	50.5394	197.8654278	

In QEMU VM:

scenario	round	min/ms	average/ ms	max/ms	approx. 95 percentile /ms	std	events per second	average events per second for each scenario
--cpu-max-prime=20000	1	6.86	7.51	20.35	8.58	0	132.52	
	2	6.88	8.48	34.58	10.84	0	117.1	
	3	6.94	8.33	30.93	9.91	0	119.52	
	4	6.92	8.4	25.19	9.39	0	118.46	
	5	6.94	10.41	83.06	19.65	0	95.47	116.614
--cpu-max-prime=22000	1	7.87	11.48	425.26	20.74	0	86.38	
	2	7.87	10.08	90.16	16.71	0	98.69	
	3	8.02	9.69	25.65	12.08	0	102.64	
	4	8.01	9.8	26.29	11.04	0	101.49	
	5	8.15	11.08	98.79	15.27	0	89.78	95.796
--cpu-max-prime=24000	1	8.9	10.16	26.56	12.3	0	97.97	
	2	8.95	12.16	166.65	21.5	0	81.83	
	3	8.94	12.21	207.79	18.61	0	81.45	
	4	9.1	12.54	143.79	17.63	0	79.33	
	5	9.12	12.31	105.29	17.01	0	80.85	84.286

For IO:

In Docker container:

scenario	round	min/ms	average/ ms	max/ms	approx. 95 percentile /ms	std	throughput / Mib/s	average throughput for each scenario
--file-total-size=1G	1	0.02	0.18	invalid	0.13	0.02	79.534	
	2	0.02	0.2	12.96	0.14	0.03	69.57	
	3	0.02	0.15	10.78	0.1	0.01	85.825	
	4	0.02	0.14	15.75	0.1	0.01	82.94	
	5	0.02	0.17	19.95	0.12	0.02	81.718	79.9174
--file-total-size=2G	1	0.02	1.27	27.73	5.89	0.02	54.801	
	2	0.03	1.11	29.13	5.89	0.04	53.238	
	3	0.03	0.58	24.28	4.21	0.02	64.34	53.0886

	4	0.03	0.85	21.15	5.19	0.03	57.797	
	5	0.03	0.49	108.56	0.56	0.06	35.267	
--file-total-size=3G	1	0.02	1.91	34.35	6.73	0.05	47.913	53.624
	2	0.03	1.69	32.19	6.49	0.03	49.765	
	3	0.03	1.39	24.59	6.39	0.04	53.749	
	4	0.03	1.5	18.38	6.6	0.03	52.17	
	5	0.03	0.63	15.46	4.91	0.02	64.523	

In QEMU VM:

scenario	round	min/ms	average /ms	max/ms	approx. 95 percentile/ms	std	read / Mib/s	write / Mib/s	through put / Mib/s	average through put for each scenario
--file-total-size=1G	1	0.02	6.37	55.89	21.11	0.02	9.47	6.31	15.78	16.094
	2	0.02	6.21	49.29	20.37	0.02	10.03	6.69	16.72	
	3	0.02	6.32	53.25	21.11	0.02	9.54	6.36	15.9	
	4	0.02	6.4	63.05	21.11	0.03	9.72	6.48	16.2	
	5	0.02	6.4	86.75	21.11	0.02	9.52	6.35	15.87	
--file-total-size=2G	1	0.02	11.28	402.09	34.95	0.06	5.39	3.59	8.98	8.904
	2	0.03	11.3	349.29	34.95	0.05	5.25	3.49	8.74	
	3	0.03	11.03	194.76	35.59	0.06	5.27	3.51	8.78	
	4	0.03	11.91	375.88	36.89	0.05	4.96	3.31	8.27	
	5	0.03	10.39	284.52	32.53	0.05	5.85	3.9	9.75	
--file-total-size=3G	1	0.03	12.71	320.66	39.65	0.07	4.69	3.13	7.82	8.468
	2	0.03	12.14	373.73	37.56	0.11	4.93	3.29	8.22	
	3	0.03	12.86	871.41	41.1	0.06	4.77	3.16	7.93	
	4	0.03	10.29	134.64	32.53	0.05	5.7	3.8	9.5	
	5	0.03	11.58	500.71	36.89	0.07	5.33	3.54	8.87	

As data shown in tables, both cpu and i/o speed of the Docker container is higher than the speed of the QEMU virtual machine. It’s partially because containers don't need a guest OS and share the host OS.

Git Repository Information

Link of repository:

https://github.com/XihengY/COEN241_HW1

hash of commit ID:

```
68b75c7000c2f60db798e81e0102351c61a72aad
```