Ashita Jayesh Shetty
Student Id: 00001607967

COEN- 241
Cloud Computing

HW1: System vs OS Virtualization

Configurations for the experimental setup:

CPU : 2.6 GHz 6 -core intel core i7

Memory: 32 GB 2667 MHz DDR4

OS: mac OS Monterry version 12.3.1

Disk space: 120 GB

Note: Borrowed a mac for this assignment as windows is not recommended. Terminal screenshots will show a different userId.

Installation of QEMU and enabling QEMU VM:

Installed QEMU using homebrew and the following command:

```
$ brew install qemu
```

Followed the homework document to create QEMU image and to install the VM:

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

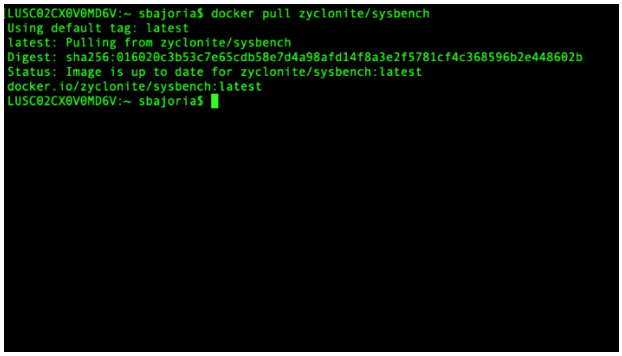
```
$ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom
/Users/sbajoria/downloads/ubuntu-20.04.4-live-server-amd64.iso -m 2046
-boot strict=on
```

Installation of Docker and enabling Docker container

Installed the docker desktop using the link:
https://docs.docker.com/desktop/mac/install/

Installed the docker image zyclonite/sysbench using the command:

```
docker pull zyclonite/sysbench
```



Note: Terminal id shows sbajoria, since it's a borrowed laptop

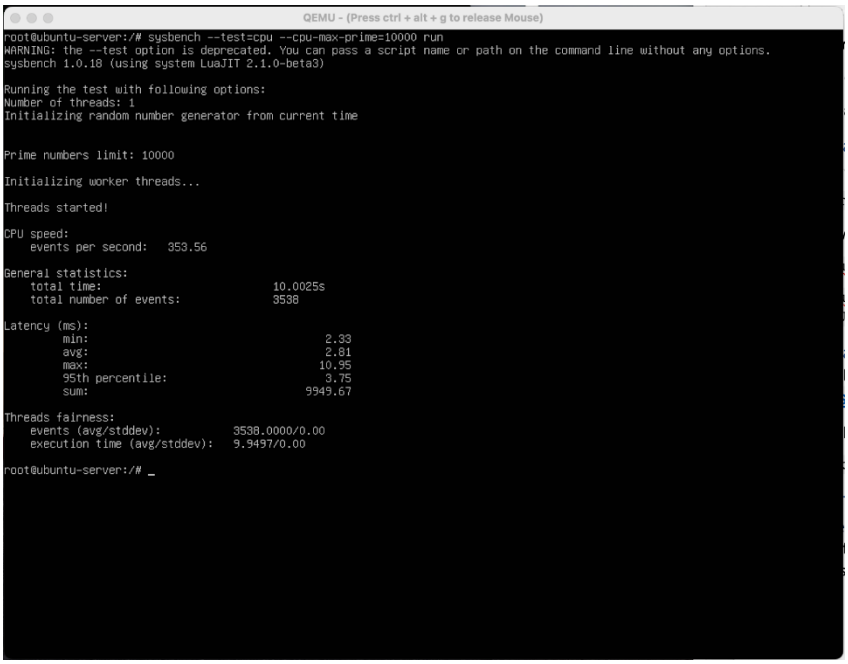
Screenshots of the experiments

CPU test:

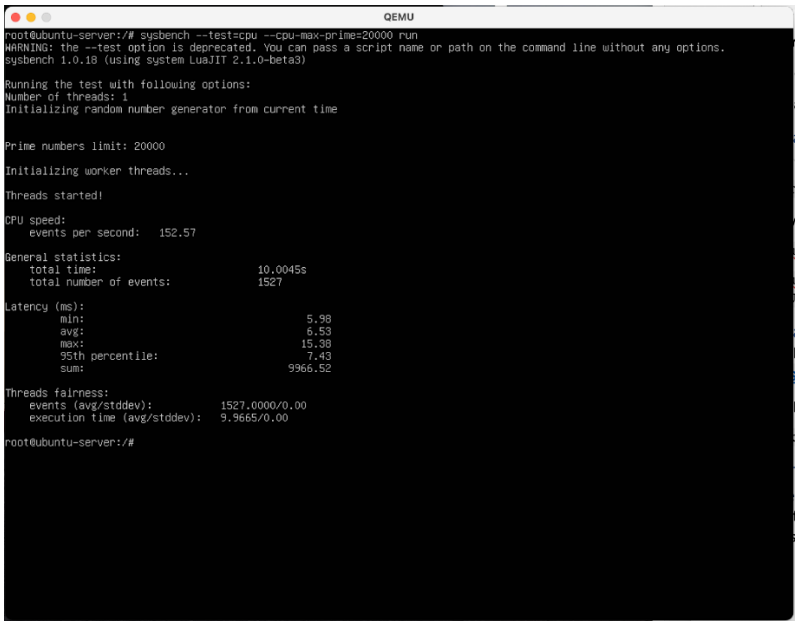
Ran different prime numbers on both QEMU and Docker. Results of all three tests are added as screenshots.

QEMU:

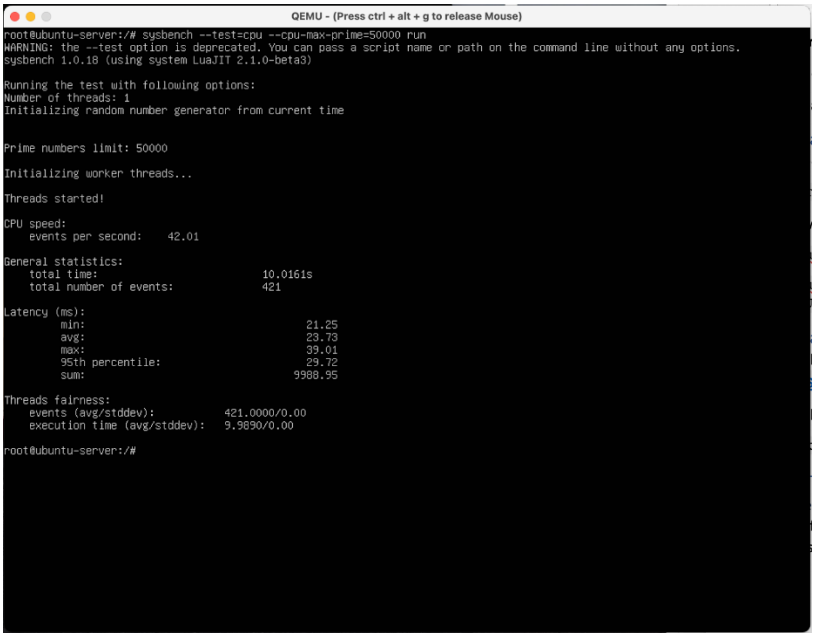
Test 1: `sysbench --test=cpu --cpu-max-prime=10000 run`



Test 2: `sysbench --test=cpu --cpu-max-prime=20000 run`

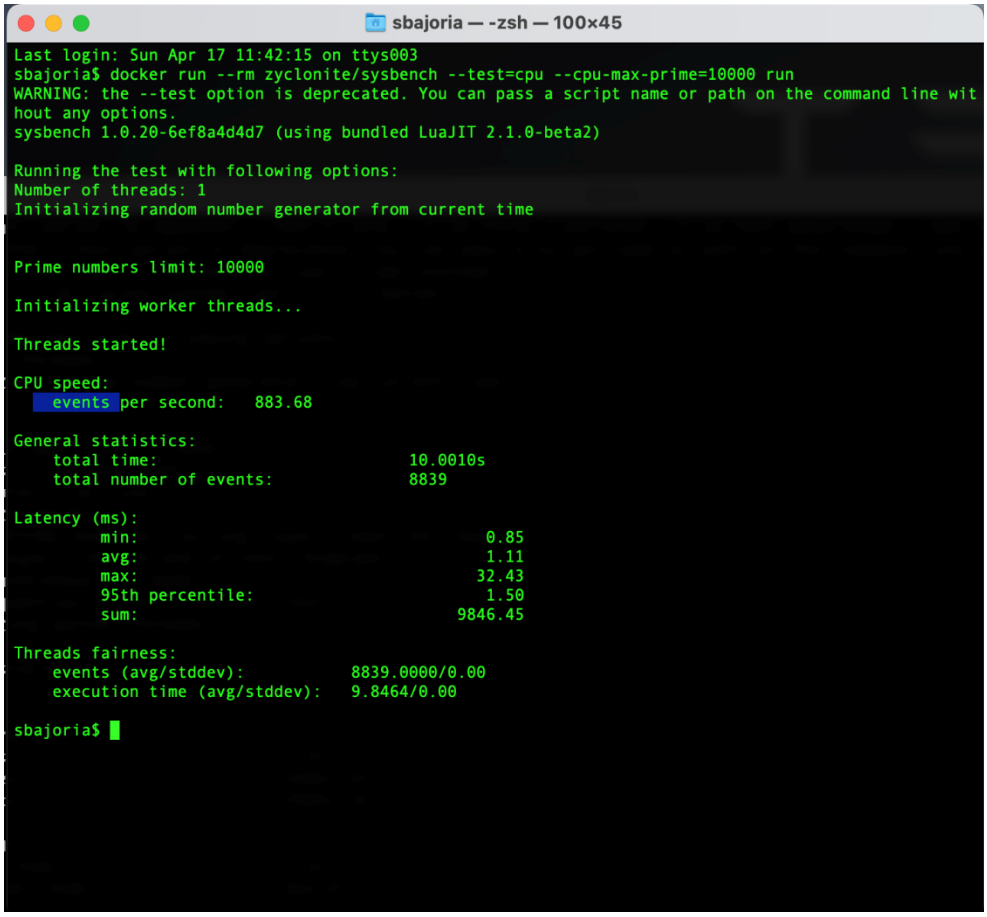


Test 3: `sysbench --test=cpu --cpu-max-prime=50000 run`

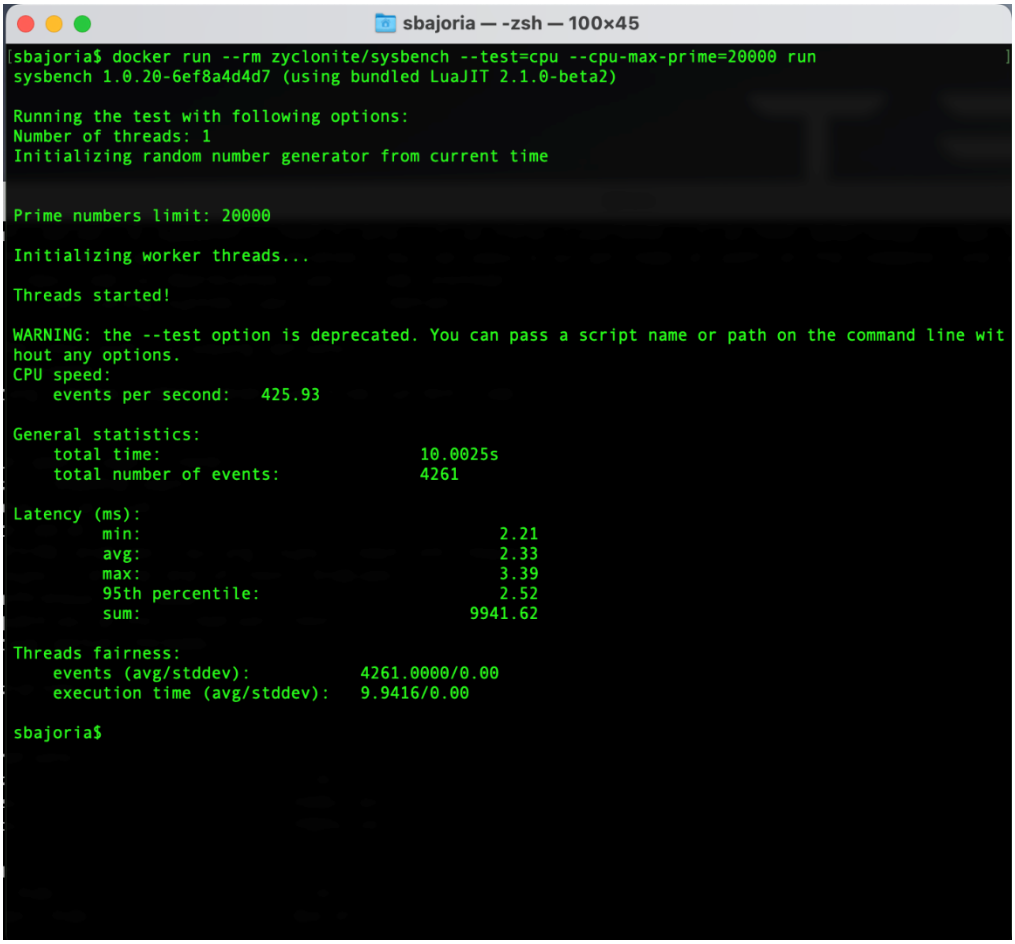


Docker:

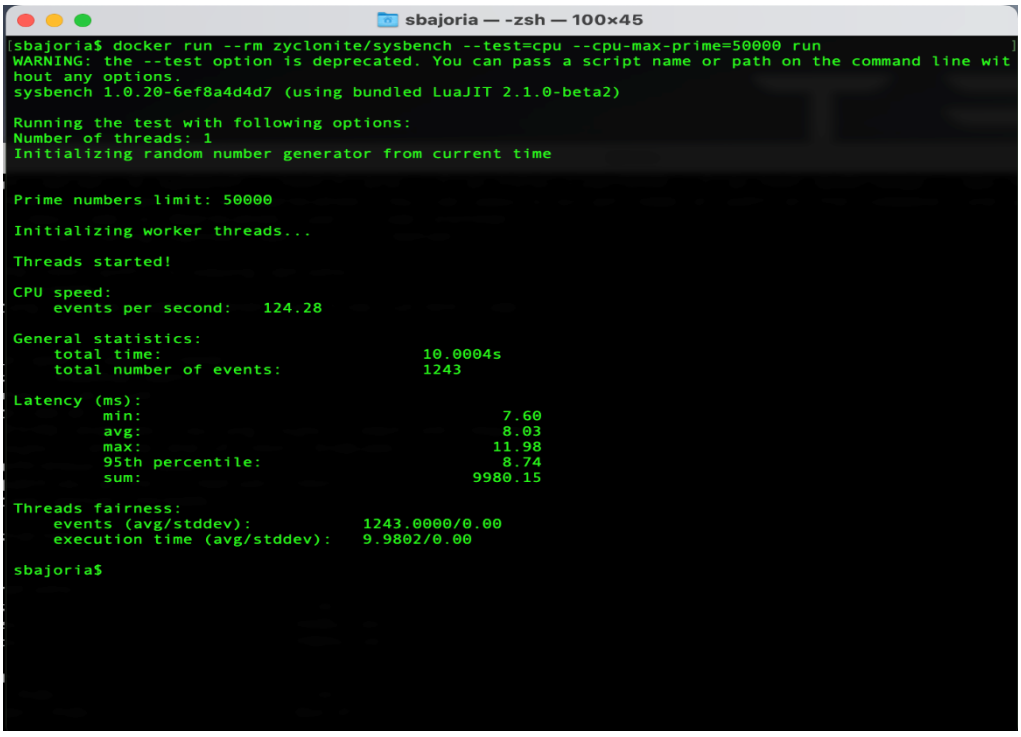
Test 1: `docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=10000 run`



Test 2: `docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=20000 run`

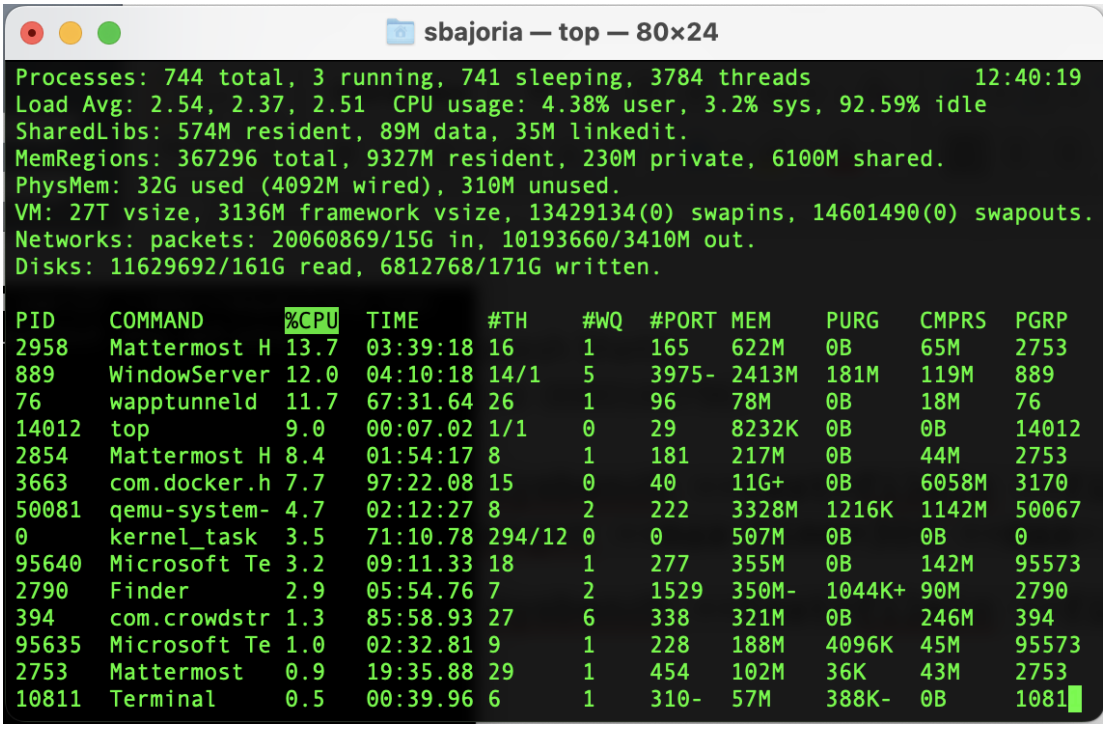


Test 3: `docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=50000 run`

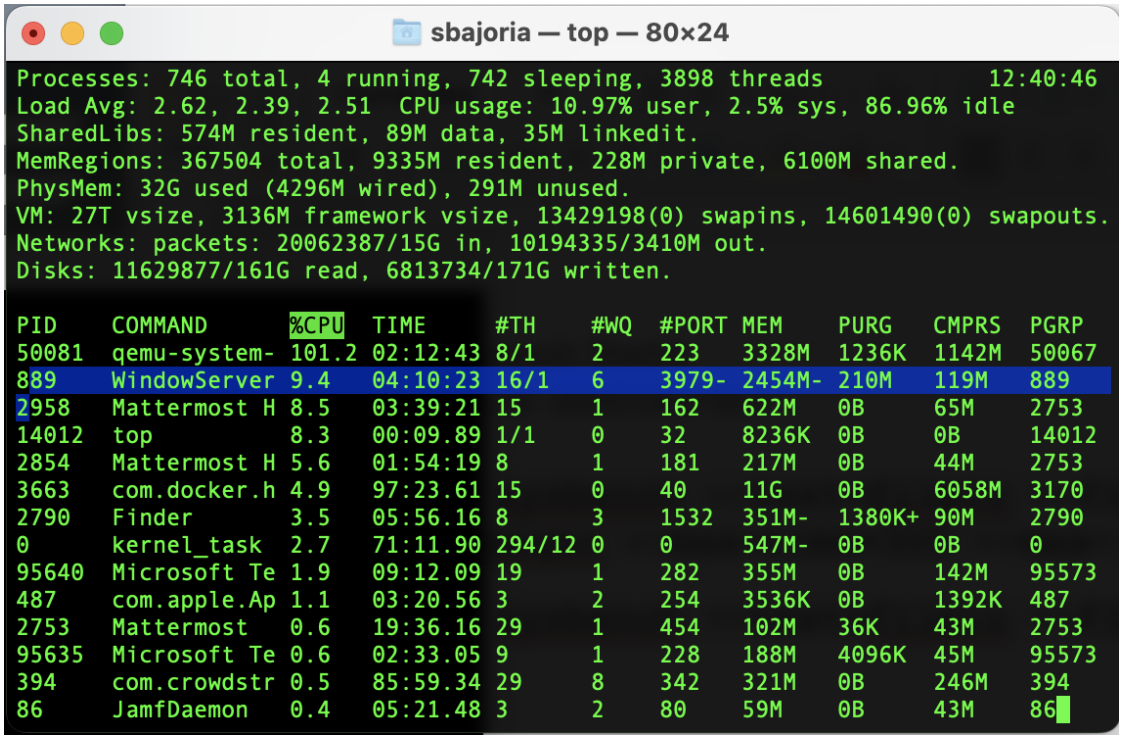


CPU utilization for user-level and kernel -level while running the CPU test:

Screenshot of activity monitor when QEMU is idle:



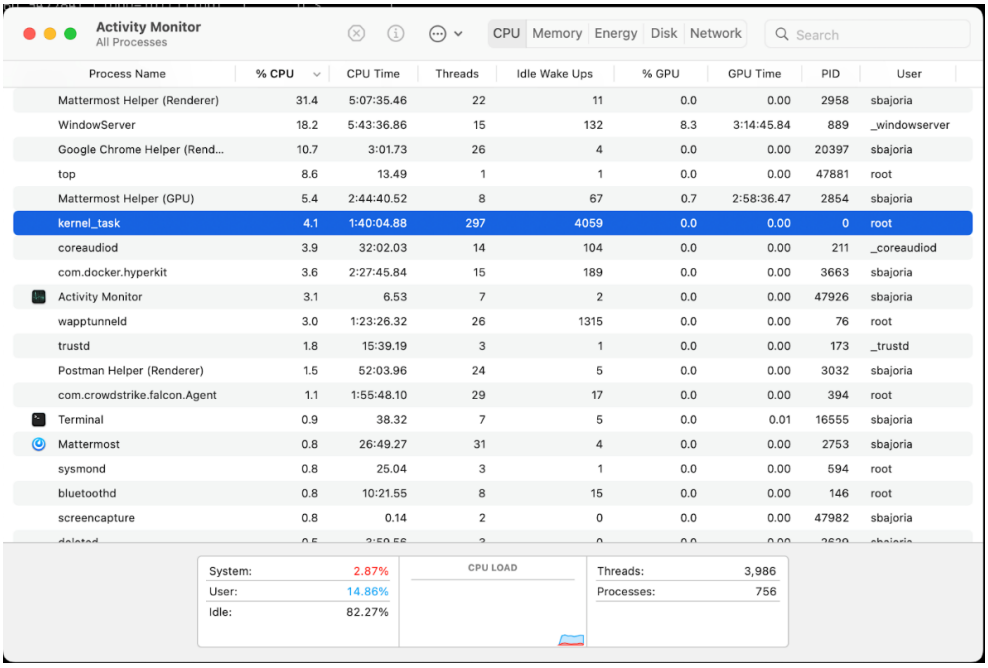
Using activity monitor when running cpu test on sysbench on QEMU:



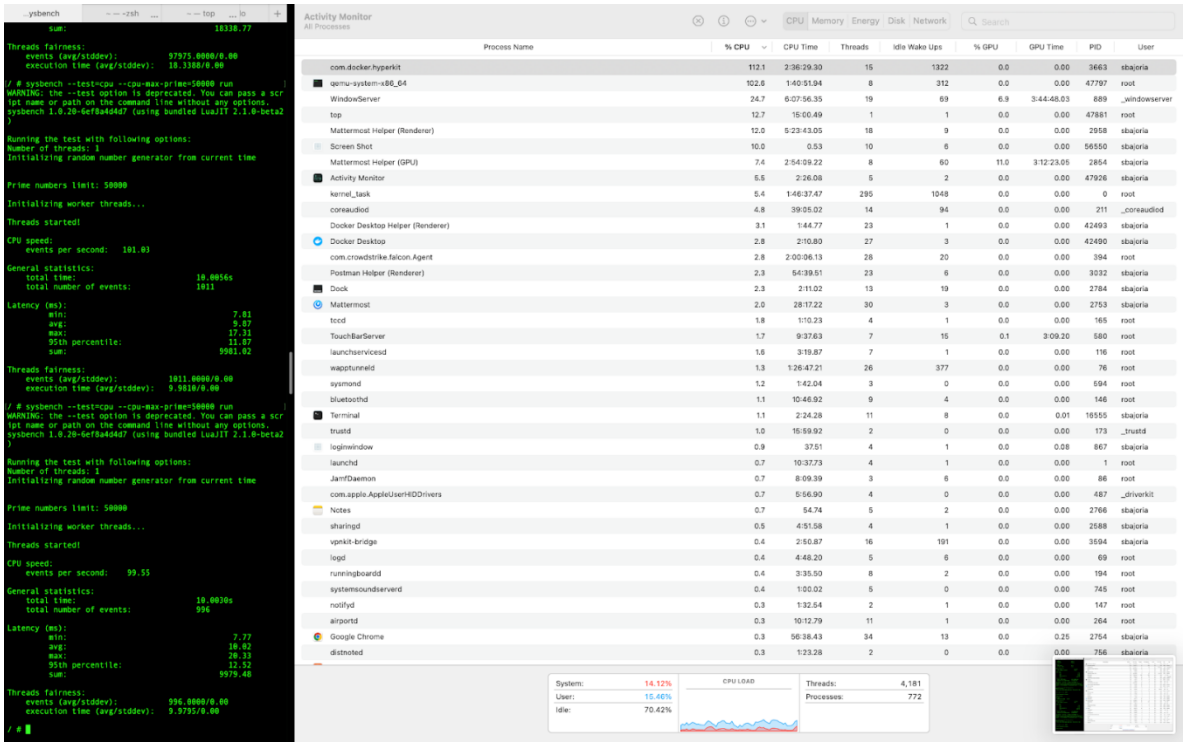
From the above screenshots we can see that the CPU usage has increased from 13.7 to 101.2 but kernel_task remains more or less the same.

Now, Running the same test for docker.

Screenshot of the activity monitor when docker is idle:



Screenshot while running the test:



	Sysbench running	Sysbench not running
User level CPU	15.46%	14.86%
Kernel level CPU	14.12%	2.87%

As we can see OS virtualization uses kernel level cpu.

FileIO test:

Since my laptop was running out of memory, tested the fileIO for a small memory, both on QEMU and Docker. Setting the max-requests to 0, meaning unlimited number of requests, and max-time to 20.

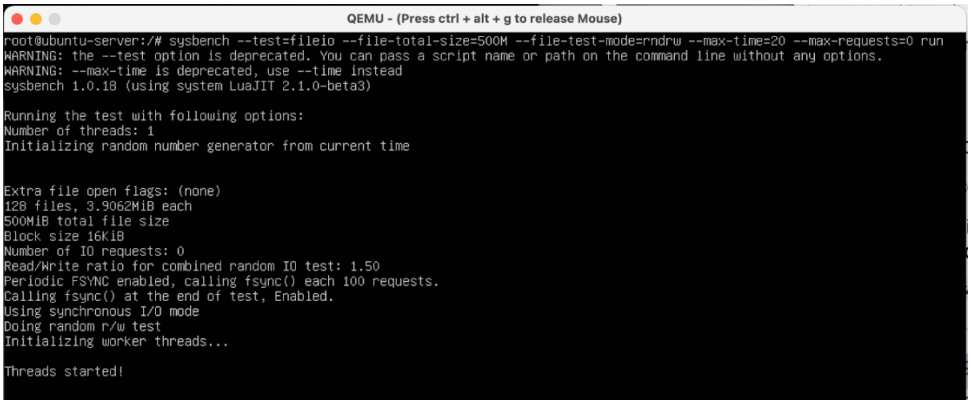
QEMU:

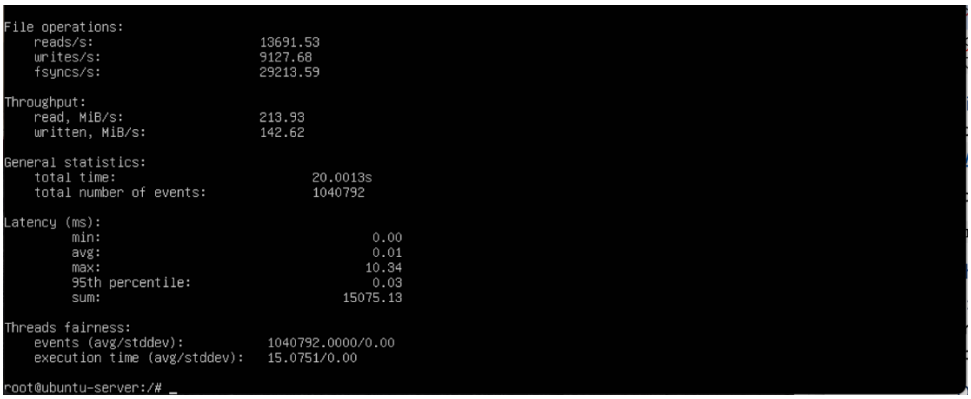
Test 1: Randown read/write

```
sysbench --test=fileio --file-total-size=500MB prepare
```

```
sysbench --test=fileio --file-total-size=500MB --file-test-mode=rndrw --max-time=20 --max-requests=0 run
```

```
sysbench --test=fileio --file-total-size=500MB cleanup
```





Test 2: Random write

```
sysbench --test=fileio --file-total-size=500MB prepare
```

```
sysbench --test=fileio --file-total-size=500MB --file-test-mode=rndwr --max-time=20 --max-requests=0 run
```

```
sysbench --test=fileio --file-total-size=500MB cleanup
```

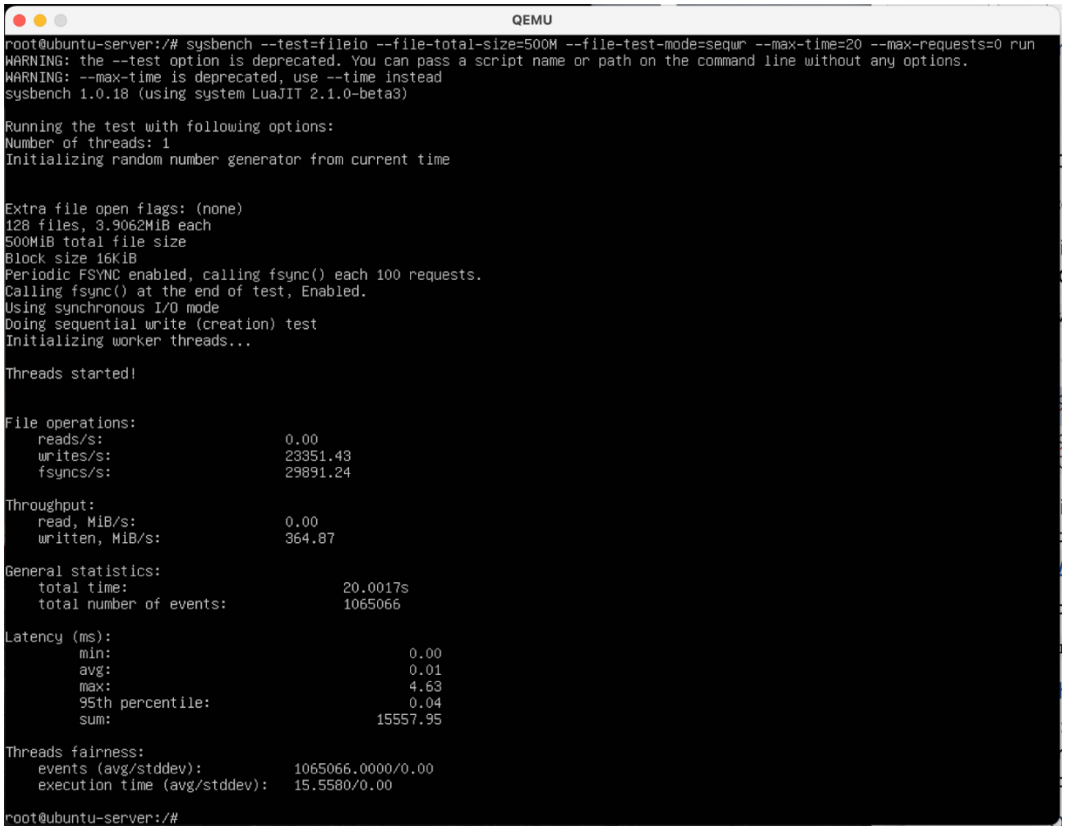
```
File operations:
    reads/s:                      0.00
    writes/s:                     24202.65
    fsyncs/s:                     30981.05

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               378.17

General statistics:
    total time:                          20.0013s
    total number of events:              1103822

Latency (ms):
         min:                                    0.00
         avg:                                    0.01
         max:                                    4.03
         95th percentile:                        0.03
         sum:                                15291.60

Threads fairness:
    events (avg/stddev):           1103822.0000/0.00
    execution time (avg/stddev):   15.2916/0.00

root@ubuntu-server:/#
```

Test 3: Sequential write

`sysbench --test=fileio --file-total-size=500MB prepare`

`sysbench --test=fileio --file-total-size=500MB --file-test-mode=seqwr --max-time=20 –num-threads=4 --max-requests=0 run`

`sysbench --test=fileio --file-total-size=500MB cleanup`



Docker:

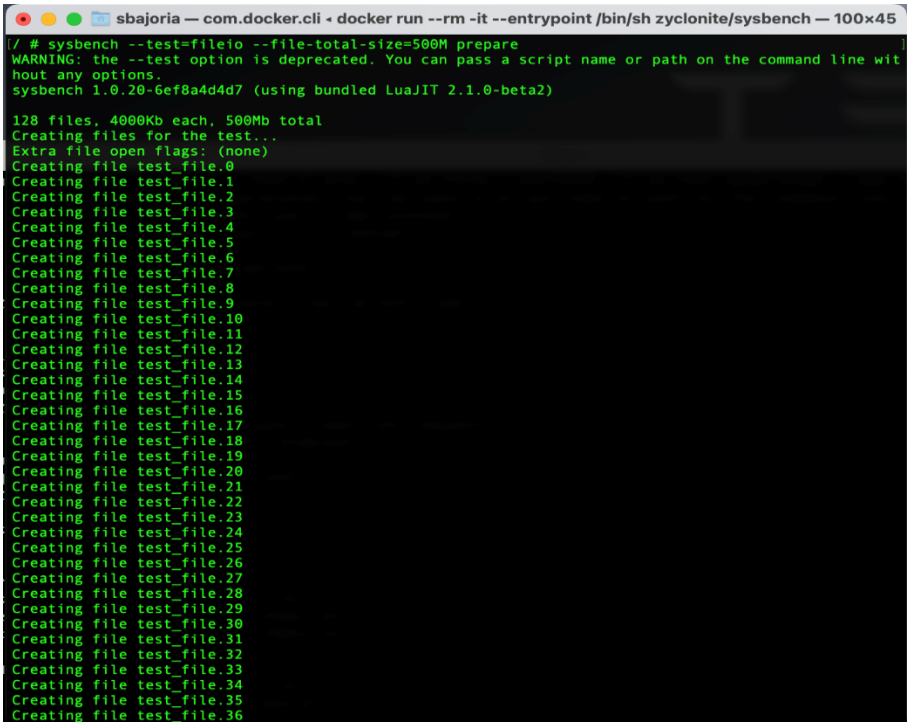
Run the command: `docker run --rm -it --entrypoint /bin/sh zyclonite/sysbench` before running the tests.

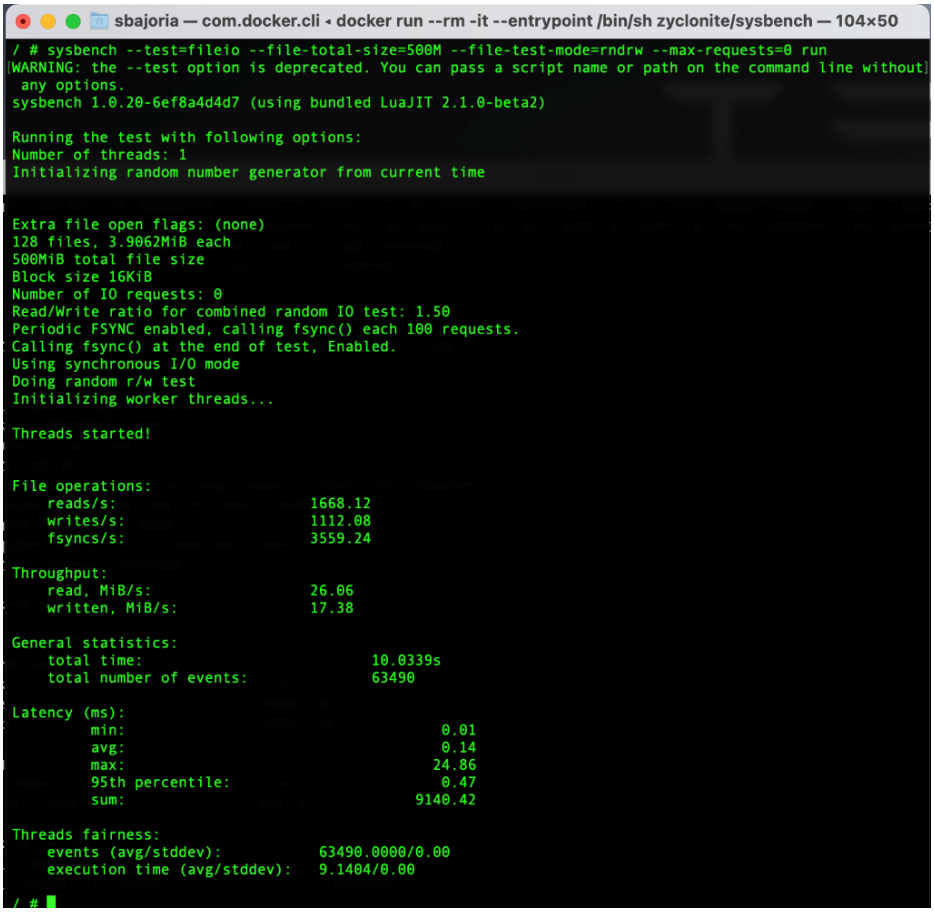
Test 1: Randown read/write

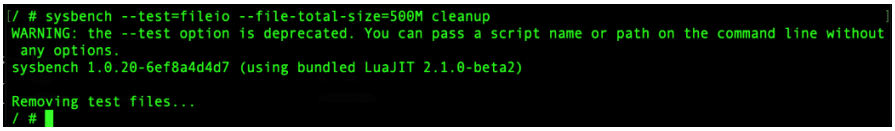
`sysbench --test=fileio --file-total-size=500M prepare`

`sysbench --test=fileio --file-total-size=500M --file-test-mode=rndrw --max-time=20 --max-requests=0 run`

`sysbench --test=fileio --file-total-size=500M cleanup`


sbajoria — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh zyclonite/sysbench — 100×45
/ # sysbench --test=fileio --file-total-size=500M prepare
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-6ef8a4d4d7 (using bundled LuaJIT 2.1.0-beta2)
128 files, 4000Kb each, 500Mb 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
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
Creating file test_file.24
Creating file test_file.25
Creating file test_file.26
Creating file test_file.27
Creating file test_file.28
Creating file test_file.29
Creating file test_file.30
Creating file test_file.31
Creating file test_file.32
Creating file test_file.33
Creating file test_file.34
Creating file test_file.35
Creating file test_file.36


sbajoria — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh zyclonite/sysbench — 104×50
/ # sysbench --test=fileio --file-total-size=500M --file-test-mode=rndrw --max-requests=0 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-6ef8a4d4d7 (using bundled LuaJIT 2.1.0-beta2)
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 r/w test
Initializing worker threads...
Threads started!
File operations:
reads/s: 1668.12
writes/s: 1112.08
fsyncs/s: 3559.24
Throughput:
read, MiB/s: 26.06
written, MiB/s: 17.38
General statistics:
total time: 10.0339s
total number of events: 63490
Latency (ms):
min: 0.01
avg: 0.14
max: 24.86
95th percentile: 0.47
sum: 9140.42
Threads fairness:
events (avg/stddev): 63490.0000/0.00
execution time (avg/stddev): 9.1404/0.00
/ #

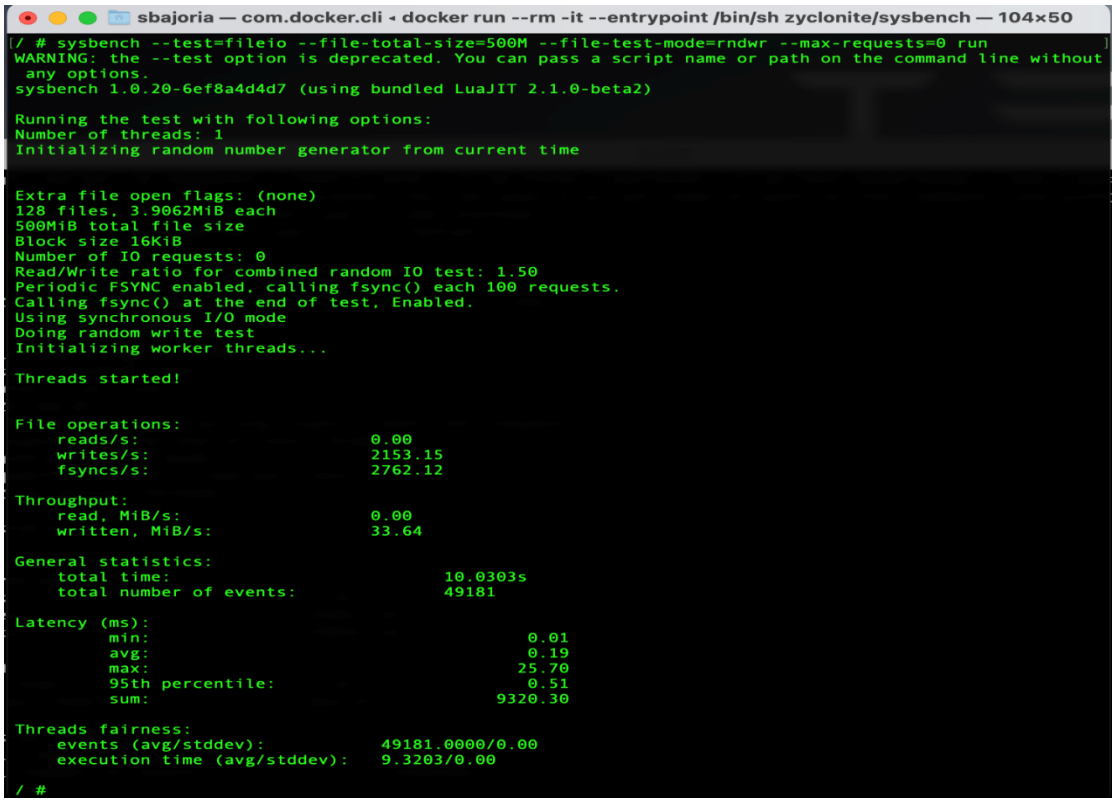

/ # sysbench --test=fileio --file-total-size=500M cleanup
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-6ef8a4d4d7 (using bundled LuaJIT 2.1.0-beta2)
Removing test files...
/ #

Test 2: Random write

```
sysbench --test=fileio --file-total-size=500MB prepare
```

```
sysbench --test=fileio --file-total-size=500MB --file-test-mode=rndwr --max-time=20 --max-requests=0 run
```

```
sysbench --test=fileio --file-total-size=500MB cleanup
```

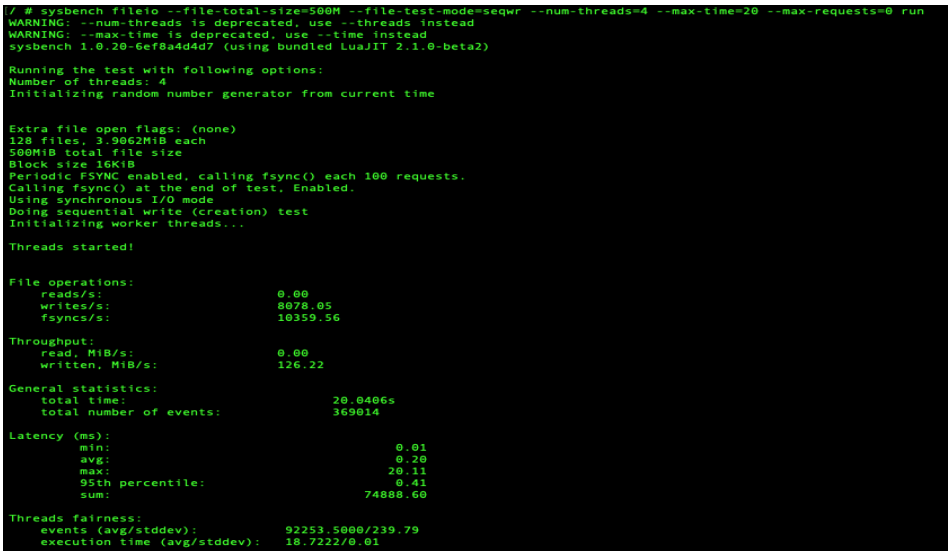


Test 3: Sequential write

```
sysbench --test=fileio --file-total-size=500MB prepare
```

```
sysbench --test=fileio --file-total-size=500MB --file-test-mode=seqwr –num-threads=4 --max-time=20 --max-requests=0 run
```

```
sysbench --test=fileio --file-total-size=500MB cleanup
```

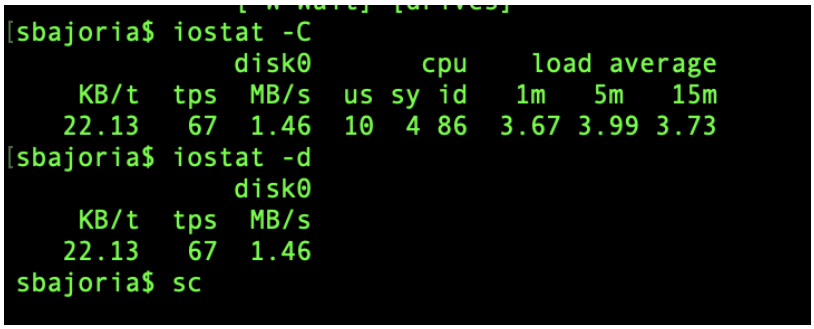


I/O Utilization while running File I/O test:

Used `iostat` command to find the disk utilization while running a file i/o test on docker:

```
$ iostat -x
```

Screenshot of the same:



Shell scripts:

1. FileIOQEMU.sh
2. FileIODocker.sh
3. cpuQEMU.sh
4. cpuDocker.sh

Analysis of the performance data:

Each test has been repeated 5 times, by clearing the cache after running each test using the command:
`sync && sudo purge`

CPU test for QEMU

Test 1:

	Total Time	Total number of events	Total Number of events per second
1	10.002	4005	400.4199
2	10.0042	3995	399.3323
3	10.0027	4018	401.6915
4	10.0038	3964	396.2494
5	10.0034	4022	402.0633

Test 2:

	Total Time	Total number of events		Total Number of events per second
1	10.0044	1551	155.0318	10.0044
2	10.0021	1578	157.7669	10.0021
3	10.0037	1579	157.8416	10.0037
4	10.0055	1585	158.4129	10.0055
5	10.0048	1581	158.0241	10.0048

Test 3:

	Total Time	Total number of events		Total Number of events per second
1	10.0196	444	44.31315	10.0196
2	10.0119	453	45.24616	10.0119

3	10.0164	450	44.92632	10.0164
4	10.01	454	45.35465	10.01
5	10.0049	455	45.47772	10.0049

CPU test for Docker

Test 1:

	Total Time	Total number of events	Execution Time	Total Number of events per second
1	10.0007	10237	9.856	1023.48
2	10.0007	10592	9.8586	1058.96
3	10.0005	10515	9.8577	1051.29
4	10.0003	10572	9.8573	1057.01
5	10.0002	10622	9.8589	1062.03

Test 2:

	Total Time	Total number of events	Execution Time	Total Number of events per second
1	10.0011	4296	9.9428	429.5
2	10.0011	4305	9.9423	430.39
3	10.0024	4257	9.9442	425.54
4	10.0005	4272	9.941	427.1
5	10.0005	4288	9.9414	428.72

Test 3:

	Total Time	Total number of events	Execution time	Total Number of events per second
1	10.0074	1274	9.9887	127.29
2	10.007	1272	9.982	127.17
3	10.007	1273	9.9879	127.19
4	10.0011	1270	9.9824	126.97
5	10.0051	1268	9.9866	126.72

Performance analysis:

Used events per second to calculate the average, min and max values.

CPU max time = 10000

	Average	Min	Max	Standard Deviation
QEMU	399.9513	396.2494	402.0633	2.087
Docker	1050.554	1023.48	1062.03	13.98

CPU max time = 20000

	Average	Min	Max	Standard Deviation
QEMU	157.4155	155.0318	158.4129	1.212
Docker	428.25	425.54	430.39	1.733

CPU max time = 50000

	Average	Min	Max	Standard Deviation
QEMU	45.0636	44.31315	45.47772	0.41
Docker	127.068	126.72	127.29	0.2026

FileIO test for QEMU

Test 1:

	Total Time	Total number of events	Total Number of events per second
1	20.0014	129391	6469.02
2	20.0015	129550	6477.29
3	20.0015	129020	6450.31
4	20.0022	119250	5961.89
5	20.0012	129575	6478.96

Test 2:

	Total Time	Total number of events	Total Number of events per second
1	20.0014	122712	6135.86
2	20.0014	122299	6114.42
3	20.0016	122341	6116.16
4	20.002	123114	6155.94
5	20.0012	122122	6105.44

Test 3:

	Total Time	Total number of events	Total Number of events per second
1	20.0015	123815	6190.06
2	20.0014	122317	6115.67
3	20.0014	123721	6185.22
4	20.0008	124782	6238.55
5	20.0012	123841	6191.83

FileIO test for Docker

Test 1:

	Total Time	Total number of events	Total time taken by event execution	Total Number of events per second
1	20.0192	125165	18.2931	6842.197
2	20.026	123941	18.2955	6774.398
3	20.012	125671	18.2928	6869.971
4	20.023	124212	18.2966	6788.802
5	20.0188	125887	18.2921	6882.042

Test 2:

	Total Time	Total number of events	Total time taken by event execution	Total Number of events per second
1	20.0308	95934	18.6764	5136.643
2	20.035	91561	18.6714	4903.81
3	20.0295	94561	18.6591	5067.822
4	20.031	94981	18.6722	5086.76
5	20.0288	94812	18.6531	5082.908

Test 3:

	Total Time	Total number of events	Total time taken by event execution	Total Number of events per second
1	20.0437	364598	18.7355	19460.28
2	20.041	361766	18.7282	19316.65
3	20.0399	366732	18.7082	19602.74
4	20.0425	374374	18.6755	20046.26
5	20.0406	369014	18.7222	19709.97

Performance analysis:

Used events per second to calculate the average, min and max values.

File test mode = rndrw

	Average	Min	Max	Standard Deviation
QEMU	6367.367	5961.844	6478.78	203.0043
Docker	6831.482	6774.398	6882.042	42.97

File test mode = rndwr

	Average	Min	Max	Standard Deviation
QEMU	6125.444	6105.734	6155.084	17.6478
Docker	5055.589	4903.81	5136.643	79.34

File test mode = `seqwr with 4 threads`

	Average	Min	Max	Standard Deviation
QEMU	6184.371	6115.422	6191.678	39.52
Docker	19627.18	20046.26	19316.65	247.94

In some cases, my QEMU events are greater than docker, which I think is because I could not install native docker on my laptop so had to use docker desktop instead. Docker desktop is slower than native docker.

Observation and Analysis:

I ran all the tests for both QEMU and Docker on a single thread. Both in CPU tests and for File I/O tests, Docker has a better performance. Number of events per second for Docker (OS Virtualization) is way higher than QEMU (System Virtualization). This remains true to all the test-modes in File I/O tests. By looking at the results for a single thread, OS virtualization performs better than System Virtualization.

Also ran a test for 8 threads, for file i/o and the results are more or less the same, OS virtualization provides better performance than system virtualization.

Git Repository:

Url : https://github.com/AshitaShetty/CloudComputing.git

Shell scripts: https://github.com/AshitaShetty/CloudComputing/tree/main/Homework1/shell%20scripts

Pdf: https://github.com/AshitaShetty/CloudComputing/tree/main/Homework1