

COEN 241: Cloud Computing System vs OS Virtualization

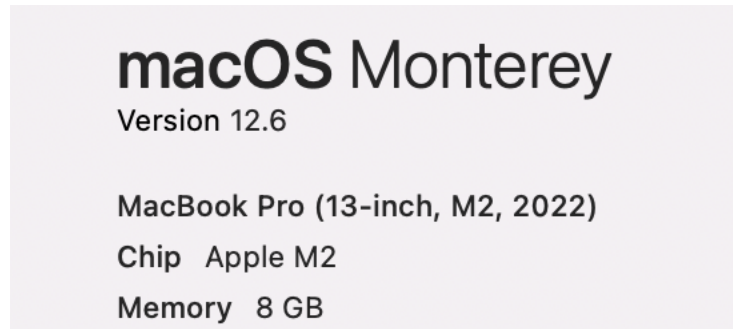
Xiao He (W1649820)

Table of contents

I.	System Configuration	2
II.	Steps to enable QEMU	3
III.	Steps to enable Docker container	4
IV.	Proof of experiment	6
V.	Performance measurement	12
VI.	Performance Analysis	16
VII.	Shell scripts for test execution	17
VIII.	CPU Utilization	18
IX.	Dockerfile for VM	20

I. System configuration

The experiments for the virtual QEMU machine and the docker container have been run on Mac M2 with Apple Silicon Chip with 8GB RAM.



1) QEMU

The ubuntu-server iso has been mounted on a QEMU image having 10G hard disk space and 2GB RAM and 1 CPU.

2) Docker container

The docker container is created with similar specifications so as to ensure that the test environment is as similar as possible. The memory limit on the container is 2G.

II. Steps to enable QEMU

As I am using MAC with an M2 chip, I installed QEMU using homebrew. After installing QEMU I created an image with 10GB of hard disk space. I downloaded the ubuntu server iso and then installed the by loading it as a cdrom. Below are the commands for these steps:

- brew install qemu
- qemu-img create -f qcow2 disk.qcow2 10G
- qemu-system-aarch64 -accel hvf -m 2048 -cpu cortex-a57 \
 - M virt,highmem=off -drive
 - file=/opt/homebrew/Cellar/qemu/7.1/share/qemu/edk2-aarch64-
 - code.fd,if=pflash,format=raw,readonly=on \
 - drive file=ovmf_vars.fd,if=pflash,format=raw serial
 - telnet::4444,server,nowait \
 - drive if=none,file=disk.qcow2,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
 - /Users/janangandhi/Documents/Courses/CC/ubuntu-20.04.4-live-server-ar
 - m64.iso \
 - device usb-ehci -device usb-kbd -device usb-mouse -usb \
 - monitor stdio

Following are the important parameters in the above QEMU command:

1. -m - It is used to assign RAM to the virtual machine. I have assigned a RAM of 2G to the machine.
2. -drive - It is used to attach the qemu-image we created in the previous step.
3. -cdrom - It is used to provide the ubuntu server iso image as a cdrom for ubuntu installation.
4. -accel - It is used to enable an accelerator
5. -cpu - It is used to select amongst different cpu models available

III. Steps to enable Docker container

As I was running into issues while trying to install native Docker onto my Apple Silicon MAC, I had to install Docker Desktop for Apple Silicon from the official Docker website - <https://docs.docker.com/desktop/mac/apple-silicon/>. After installation, I pulled the zyclonite/sysbench image using the docker pull command. It stored a copy of the image from the docker hub to my local. I was able to check this image using the below command:

```
[MacBook-Pro-42:~ alex$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	d63f752103bb	12 days ago	69.2MB
zyclonite/sysbench	latest	8731aa4184ff	10 months ago	9.19MB
docker101tutorial	latest	92f4492e2890	13 months ago	28.2MB
alpine/git	latest	b8f176fa3f0d	17 months ago	25.1MB
scrapinghub/splash	latest	9364575df985	2 years ago	1.89GB

It displays all the copies of the images that are present on the local machine.

Post that I started an instance of that image using the command:

```
docker run --rm -it --memory="2g" --entrypoint /bin/sh zyclonite/sysbench
```

```
[MacBook-Pro-42:~ alex$ docker run --rm -it --memory="2g" --entrypoint /bin/sh zyclonite/sysbench
[/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
[/ # ls -ltr
total 56
drwxr-xr-x 12 root    root    4096 Nov 24  2021 var
drwxrwxrwt  2 root    root    4096 Nov 24  2021 tmp
drwxr-xr-x  2 root    root    4096 Nov 24  2021 srv
drwxr-xr-x  2 root    root    4096 Nov 24  2021 opt
drwxr-xr-x  2 root    root    4096 Nov 24  2021 mnt
drwxr-xr-x  5 root    root    4096 Nov 24  2021 media
drwxr-xr-x  2 root    root    4096 Nov 24  2021 home
drwxr-xr-x  2 root    root    4096 Nov 24  2021 sbin
drwxr-xr-x  2 root    root    4096 Nov 24  2021 bin
drwxr-xr-x  1 root    root    4096 Dec 20  2021 run
drwxr-xr-x  1 root    root    4096 Dec 20  2021 usr
drwxr-xr-x  1 root    root    4096 Dec 20  2021 lib
dr-xr-xr-x 13 root    root         0 Oct 17 08:17 sys
dr-xr-xr-x 184 root    root         0 Oct 17 08:17 proc
drwxr-xr-x  1 root    root    4096 Oct 17 08:17 etc
drwxr-xr-x  5 root    root    360 Oct 17 08:17 dev
drwx----- 1 root    root    4096 Oct 17 08:17 root
/ #
```

The command launches an instance of the image using the memory limit of 2GB in interactive mode. I was then able to perform sysbench tests on this instance.

Other important docker commands include:

- `docker ps` - It is used to see all containers that are currently running. You can all the `-a` parameter to see all exited containers as well.
- `docker rm` - it is used to remove/clean up containers.
- `docker network ls` - to list all the networks created for the docker containers to allow communication between different containers.

IV. Proof of experiment

I have executed three tests each for CPU and File IO on both the docker container and QEMU. Each test was executed 5 times to get the average results.

The tests are as follows:

CPU tests

Test 1: sysbench --test=cpu --cpu-max-prime=2000 run

QEMU

```
xiaohe@xiaohe:~/Desktop$ sysbench --test=cpu --cpu-max-prime=2000 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: 2000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 124896.70

General statistics:
  total time:          10.0001s
  total number of events: 1249076

Latency (ms):
  min:                 0.01
  avg:                 0.01
  max:                 0.48
  95th percentile:    0.01
  sum:                 9912.54

Threads fairness:
  events (avg/stddev): 1249076.0000/0.00
  execution time (avg/stddev): 9.9125/0.00
```

Docker

```
MacBook-Pro-42:~ alex$ docker run --rm -it --memory="2g" --entrypoint /bin/sh zyclonite/sysbench
# sysbench --test=cpu --cpu-max-prime=2000 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-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 2000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 61146.46

General statistics:
   total time:          10.0001s
   total number of events: 611509

Latency (ms):
   min:                 0.02
   avg:                 0.02
   max:                 3.67
   95th percentile:    0.02
   sum:                 9950.96

Threads fairness:
   events (avg/stddev): 611509.0000/0.00
   execution time (avg/stddev): 9.9510/0.00
```

Test 2: sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run QEMU

```
xiaohe@xiaohe:~/Desktop$ sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --max-time is deprecated, use --time instead
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: 10000000

Initializing worker threads...

Threads started!

CPU speed:
  events per second:      0.94

General statistics:
  total time:              30.8132s
  total number of events:  29

Latency (ms):
  min:                     1051.95
  avg:                     1062.51
  max:                     1102.39
  95th percentile:        1089.30
  sum:                     30812.69

Threads fairness:
  events (avg/stddev):      29.0000/0.00
  execution time (avg/stddev): 30.8127/0.00
```

Docker

```
[/ # sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run
WARNING: the --test option is deprecated. You can pass a script option or path on the command line without any options.
WARNING: --max-time is deprecated, use --time instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Prime numbers limit: 10000000

Initializing worker threads...

Threads started!

CPU speed:
  events per second:      0.93

General statistics:
  total time:              30.1960s
  total number of events:  28

Latency (ms):
  min:                     1066.42
  avg:                     1078.41
  max:                     1150.05
  95th percentile:        1109.09
  sum:                     30195.38

Threads fairness:
  events (avg/stddev):      28.0000/0.00
  execution time (avg/stddev): 30.1954/0.00
```


Test 3: sysbench --num-threads=8 --test=cpu --cpu-max-prime=10000 run QEMU

```
xiaohe@xiaohe:~/Desktop$ sysbench --num-threads=8 --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.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 8
Initializing random number generator from current time

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 42071.06

General statistics:
  total time:          10.0002s
  total number of events: 420734

Latency (ms):
  min:                 0.09
  avg:                 0.19
  max:                 28.15
  95th percentile:    0.10
  sum:                 79783.52

Threads fairness:
  events (avg/stddev): 52591.7500/864.85
  execution time (avg/stddev): 9.9729/0.02
```

Docker

```
[/ # sysbench --num-threads=8 --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.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 8
Initializing random number generator from current time

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 29320.84

General statistics:
  total time:          10.0004s
  total number of events: 293229

Latency (ms):
  min:                 0.13
  avg:                 0.27
  max:                 22.47
  95th percentile:    0.19
  sum:                 79836.24

Threads fairness:
  events (avg/stddev): 36653.6250/105.60
  execution time (avg/stddev): 9.9795/0.01
```

File IO tests

Test 1:

sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw run
QEMU

```
Extra file open flags: (none)
128 files, 8MiB each
1GiB 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!

I
File operations:
  reads/s:                16811.98
  writes/s:               11207.98
  fsyncs/s:              36069.23

Throughput:
  read, MiB/s:            262.69
  written, MiB/s:         175.12

General statistics:
  total time:              10.0494s
  total number of events:  642047

Latency (ms):
  min:                     0.00
  avg:                     0.25
  max:                     17.22
  95th percentile:        0.73
  sum:                     159750.57

Threads fairness:
  events (avg/stddev):    40127.9375/605.38
  execution time (avg/stddev): 9.9844/0.00
```

Docker

```
/ # sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mod
e=rndrw run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)
```

```
Running the test with following options:
Number of threads: 16
Initializing random number generator from current time
```

```
Extra file open flags: (none)
128 files, 8MiB each
1GiB 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:          17264.41
  writes/s:         11509.44
  fsyncs/s:         37023.65
```

```
Throughput:
  read, MiB/s:      269.76
  written, MiB/s:   179.84
```

```
General statistics:
  total time:       10.0397s
  total number of events: 658583
```

```
Latency (ms):
  min:              0.00
  avg:              0.24
  max:              10.04
  95th percentile: 0.74
  sum:              159751.09
```

```
Threads fairness:
  events (avg/stddev): 41161.4375/520.64
  execution time (avg/stddev): 9.9844/0.00
```

Test 2:

sysbench --num-threads=8 --test=fileio --file-total-size=1G --file-test-mode=seqrd run
QEMU

```
Extra file open flags: (none)
128 files, 8MiB each
1GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential read test
Initializing worker threads...

Threads started!

File operations:
  reads/s:                2091107.20
  writes/s:               0.00
  fsyncs/s:               0.00

Throughput:
  read, MiB/s:            32673.55
  written, MiB/s:         0.00

General statistics:
  total time:              10.0008s
  total number of events:  20913508

Latency (ms):
  min:                     0.00
  avg:                     0.00
  max:                     33.90
  95th percentile:        0.00
  sum:                     61411.78

Threads fairness:
  events (avg/stddev):    2614188.5000/45547.40
  execution time (avg/stddev): 7.6765/0.05
```

Docker

```
[/ # sysbench --num-threads=8 --test=fileio --file-total-size=1G --file-test-mode=seqrd run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line wi
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)
```

Running the test with following options:

Number of threads: 8

Initializing random number generator from current time

Extra file open flags: (none)

128 files, 8MiB each

1GiB total file size

Block size 16KiB

Periodic FSYNC enabled, calling fsync() each 100 requests.

Calling fsync() at the end of test, Enabled.

Using synchronous I/O mode

Doing sequential read test

Initializing worker threads...

Threads started!

File operations:

reads/s: 1538888.37

writes/s: 0.00

fsyncs/s: 0.00

Throughput:

read, MiB/s: 24045.13

written, MiB/s: 0.00

General statistics:

total time: 10.0006s

total number of events: 15390371

Latency (ms):

min: 0.00

avg: 0.00

max: 24.23

95th percentile: 0.00

sum: 62424.11

Threads fairness:

events (avg/stddev): 1923796.3750/20798.93

execution time (avg/stddev): 7.8030/0.11

Test 3:

sysbench --num-threads=8 --test=fileio --file-total-size=1G --file-test-mode=seqwr run
QEMU

```
Extra file open flags: (none)
128 files, 8MiB each
1GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:                0.00
  writes/s:               21405.49
  fsyncs/s:              27494.29

Throughput:
  read, MiB/s:            0.00
  written, MiB/s:         334.46

General statistics:
  total time:              10.0367s
  total number of events:  489797

Latency (ms):
  min:                     0.00
  avg:                     0.16
  max:                     17.13
  95th percentile:        0.37
  sum:                     79873.86

Threads fairness:
  events (avg/stddev):    61224.6250/595.95
  execution time (avg/stddev): 9.9842/0.00
```

Docker

```
[/ # sysbench --num-threads=8 --test=fileio --file-total-size=1G --file-test-mode=seqwr run
WARNING: the --test option is deprecated. You can pass a script name or path on the command l
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)
```

Running the test with following options:

Number of threads: 8

Initializing random number generator from current time

Extra file open flags: (none)

128 files, 8MiB each

1GiB total file size

Block size 16KiB

Periodic FSYNC enabled, calling fsync() each 100 requests.

Calling fsync() at the end of test, Enabled.

Using synchronous I/O mode

Doing sequential write (creation) test

Initializing worker threads...

Threads started!

File operations:

reads/s:	0.00
writes/s:	19885.18
fsyncs/s:	25542.94

Throughput:

read, MiB/s:	0.00
written, MiB/s:	310.71

General statistics:

total time:	10.0320s
total number of events:	454738

Latency (ms):

min:	0.00
avg:	0.18
max:	15.73
95th percentile:	0.50
sum:	79851.51

Threads fairness:

events (avg/stddev):	56842.2500/516.07
execution time (avg/stddev):	9.9814/0.00

V. Performance measurement

I have used **Events per second** as the standard unit of measurement across all the below tests. The tests for CPU are designed to monitor system performance with varied CPU max prime, the number of threads as well as the max time. The parameters for all CPU tests are as follows:

1. CPU tests

	CPU max prime	Number of Threads	Max Time
Test 1	2000	1	0
Test 2	10000000	1	30
Test 3	10000	8	0

The result of each test rounded off to two decimal places with 5 runs in both Docker and QEMU are as follows:

Test 1 results: sysbench --test=cpu --cpu-max-prime=2000 run

	QEMU	Docker
Average	112974.25	55210.10
Min	111913.93	53268.05
Max	115140.45	56219.67
Std	1264.34	1167.08

Test 2 results: sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run

	QEMU	Docker
Average	0.842	0.85
Min	0.81	0.82
Max	0.85	0.87
Std	0.01	0.01

Test 3 results: sysbench --num-threads=8 --test=cpu --cpu-max-prime=10000 run

	QEMU	Docker
Average	10634.71	27152.73
Min	10499.60	26922.64
Max	10736.79	27532.89
Std	95.65	237.50

2. File IO tests

The tests for file IO are designed to monitor the system performance with varied file size, number of threads and modes. The parameters of these tests are as follows:

	File size	Number of Threads	Mode
Test 1	3G	16	rndrw
Test 2	2G	8	seqrd
Test 3	2G	8	seqwr

For each File IO test we need to ensure that the system does not go to cache for the file as that would taint the test results. For each test we run three stages : prepare, run and cleanup. The result of each test rounded off to two decimal places with 5 runs in both Docker and QEMU are as follows:

Test 1 results: sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw

	QEMU	Docker
Average	48839.02	34194.45
Min	40756.71	20056.79
Max	52989.26	43222.96
Std	4784.59	8548.62

Test 2 results: sysbench --num-threads=8 --test=fileio --file-total-size=2G
--file-test-mode=seqrd

	QEMU	Docker
Average	41347.68	38156.18
Min	35455.98	12355.28
Max	48135.42	60261.24
Std	4635.56	21426.84

Test 3 results: sysbench --num-threads=8 --test=fileio --file-total-size=2G
--file-test-mode=seqwr

	QEMU	Docker
Average	52914.12	36616.48
Min	48114.52	29624.89
Max	55605.25	39888.78
Std	3495.21	4115.24

VI. Performance Analysis

We can infer the below things from the results of all tests that we have collected:

- 1. QEMU is faster than Docker desktop on Mac M2 for CPU tests -**
Containers are expected to have better performance than Virtual Machines because they only provide OS virtualization and don't interact directly with the hardware. However, because the test machine is a Mac M1 the docker performance is relatively worse than QEMU for all CPU tests. From the results we can see that the events per seconds for docker are much less than the events per seconds in QEMU.
- 2. Increasing the number of threads in the test helps to increase performance of the system -** For all the tests where the system had more threads, the performance of the system is much better. However, just increasing the number of threads does not help. The performance also depends on how many cores are present in the machine.
- 3. Higher the cpu max prime, lower is the events per seconds -** The number of events that the system performs is directly related to the complexity of the event. From the tests we can see that for higher max prime parameters, the events per seconds fall down drastically since it is more complicated to calculate.
- 4. For file tests sequential writes perform much better than random writes -** As observed in the file IO test results, the system performs better for sequential writes. This is because the seek time for sequential write and read is reduced as compared to random read and writes where the head of the disk has to rotate more.

VII. Shell scripts for test execution

cpu-test.sh

```
#!/bin/bash
threadCount=(1 1 8)
cpuMaxPrime=(2000 10000000 10000)
maxTime=(0 30 0)
for ((i=0; i<3 ;i++))
do
    echo "Running test number ${i}"
    for j in {1..5}
    do
        echo "Current run number: ${j}"
        sysbench --num-threads=${threadCount[$i]} --te
        echo "Run number ${j} is complete"
    done
done
```

file-test.sh



```
#!/bin/bash
threadCount=(16 8 8)
fileSize=("3G" "2G" "2G")
modes=("rndrw" "seqrd" "seqwr")
step=("prepare" "run" "cleanup")

for ((i=0; i<3 ;i++))
do
    echo "Running test number ${i}"
    for j in {1..5}
    do
        echo "Current run number: ${j}"
        for ((j=0; j<3; j++))
        do
            echo "Current Stage: ${step[$j]}"
            sysbench --num-threads=${threadCount[$i]} --test=fileio --file-total-size=${f
        done
        echo "Run number ${j} is complete"
    done
done
```

VIII. CPU Utilization

CPU usage has been measured on host OS using the activity monitor. Below is the CPU usage on the host machine:

When sysbench is not running on docker:

Process Name	% CPU <small>▼</small>	CPU Time	Threads	Idle Wake Ups	Kind	% GPU	GPU Time	PID
WindowServer	15.5	23:28.35	22	300	Apple	29.2	4:19.17	365
coreaudiod	11.6	13:41.25	12	146	Apple	0.0	0.00	407
corespeechd	10.3	11:58.61	12	0	Apple	0.0	0.00	668
kernel_task	9.3	9:50.70	485	1036	Apple	0.0	0.00	0
Google Chrome Helper (Rend...	9.1	4:53.62	34	109	Apple	0.0	0.00	1090
qemu-system-aarch64	3.5	9:28.92	8	109	Apple	0.0	0.00	2884
Docker Desktop Helper (Ren...	3.0	3:27.83	22	5	Apple	0.0	0.00	2920
locationd	2.5	7.71	8	1	Apple	0.0	0.00	340
airportd	2.5	21.18	10	0	Apple	0.0	0.00	397
 Activity Monitor	2.4	9.25	6	4	Apple	0.0	0.00	18733
 Microsoft Word	2.0	6:14.85	15	12	Apple	1.2	12.27	1811
bluetoothd	1.8	1:10.38	9	1	Apple	0.0	0.00	355

When cpu sysbench test is running on the docker:

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	Kind	% GPU	GPU Time	PID
qemu-system-aarch64	103.4	8:45.19	9	86	Apple	0.0	0.00	2884
WindowServer	11.1	22:20.26	22	190	Apple	12.8	4:08.55	365
coreaudiod	10.0	13:11.20	11	155	Apple	0.0	0.00	407
kernel_task	9.5	9:27.98	485	1560	Apple	0.0	0.00	0
corespeechd	8.7	11:32.71	12	0	Apple	0.0	0.00	668
Google Chrome Helper (Rend...	2.9	4:41.71	34	114	Apple	0.0	0.00	1090
Activity Monitor	2.7	5.36	7	4	Apple	0.0	0.00	18733

	Without sysbench	With sysbench
User level CPU usage%	8.46%	8.51%
System(kernel) level CPU usage%	8.43%	21.22%
Idle %	81.49%	72.4%

We can see from the above table that when we run sysbench, the kernel level CPU usage jumps from 8.43% to 21.22%. This is because the container is using the kernel of the host machine. Also, if we see, the usage of CPU by the qemu process which is used by docker to run the container increases to 105%.

IX. Dockerfile for VM

```
FROM zyclonite/sysbench

WORKDIR /test

COPY docker-start.sh /test/docker-start.sh
COPY cpu-test.sh /test/cpu-test.sh
COPY file-test.sh /test/file-test.sh

RUN chmod +x cpu-test.sh
RUN chmod +x docker-start.sh
RUN chmod +x file-test.sh

RUN apk update && apk add bash

ENTRYPOINT bash docker-start.sh
```