COEN 241 Assignment : 1

Name: Yash Bhargava
Student ID : W1630215

1. Installing QEMU and making a QEMU disk image

In this section of the report, we will go over the steps of how to install QEMU and then proceed to make a QEMU Disk Image. This is the image that we will be using for all our experiments throughout the course of this report. The installation will depend on every individual's system configuration. For this report, I'll be going over the installation steps for MacBook Pro with M1 chips (Also known as Apple Silicon).

a. Downloading Ubuntu ISO

As the first step, it is essential that we download the correct ISO image for our system's configuration as we will be using this image for all our experiments. We have chosen Ubuntu 20.04 for our experiments and thus we will be making a Ubuntu QEMU disk image. For M1 MacBook, the link for downloading the image is as follows:

Ubuntu 20.04 3 Server for ARM

After downloading it, place the image in the appropriate directory and remember the address of the directory as we will require it in the further steps.

b. Installing QEMU

Open the terminal of your MacBook and type the following command to install QEMU.

```
brew install qemu
```

That's it! We have successfully installed QEMU but our work is not done yet.

NOTE – For the above command to work, we need to have homebrew installed. I assume that the reader has homebrew installed in their system. If that's not the case for you kindly check this link.

c. Creating QEMU Image

Now, we create a QEMU image where we will install our VM. For simplicity, we create the image where we have placed our Ubuntu installation ISO. Once you have opened the terminal, type in the following command.

```
qemu-img create ubuntu.img 20G -f qcow2
```

where 20G is the space that our image will take (in this case, 20GBs) and -f stands for file format, which in this case, we have chosen qcow2.

d. Installing VM

To install our VM, run the following commands

```
qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2G \
```

```
-smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/6.2.0_1/share/qemu/edk2-aarch64-
code.fd,if=pflash,format=raw,readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="trial_2" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-cdrom ubuntu-20.04.4-live-server-arm64.iso \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic
```

Then, follow the installation instructions on screen as directed and you'll have successfully installed a QEMU Ubuntu VM image in your system.

To run your image, we use the same set of commands as above, we just remove the cdrom argument.

NOTE: Some notable arguments are:

a. -m : Stands for memory. In our command above, we have supplied 2G as the argument value which means we are allocating 2GB RAM memory for our VM
b. -accel : Stands for hardware acceleration.
c. -smp : Stands for number of cores. In our commands, we have given the argument value as 2, which means that we have allocated 2 cores for our VM.

Thus, we have successfully installed QEMU and made a QEMU VM in our system

2. Experimental Setup

For conducting these experiments, my system configurations are listed as follows:

System – 2021 MacBook Pro
Chip – M1 Pro (ARM Apple Silicon) 10 core CPU
Memory – 16GB RAM

For the VM, I will consider three test conditions to conduct my experiment. The conditions are as follows:

a. 2 cores with 2GB memory allocation
b. 4 cores with 4GB memory allocation
c. 6 cores with 6GB memory allocation

The same test conditions will be repeated for Docker experiments to maintain consistency between results.

For Docker, we are using the zyclonite/sysbench image which comes with sysbench preinstalled.

3. Docker Installation and Important Instructions

In this section of the report, we will be going over the installation steps for Docker, which is used to create and manage containers which in turn are used to run and manage different images. For the purpose of this report, we will list the installation instructions for Apple M1 chips (Apple Silicon)

For Docker to work, we first need to install Rosetta 2. If rosetta 2 is not installed in your system, we can install it by typing the following command in our terminal.

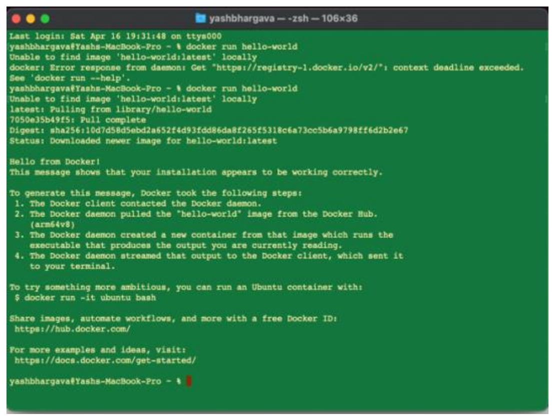
softwareupdate –install-rosetta

Post that, we can proceed to install docker engine, which we will be using throughout the entirety of these experiments. To install docker engine, simply download the docker dmg image from the following link.

Post installation, we can check if the docker has installed correctly on the system. To test docker, we simply type in the following command:

docker run hello-world

You should see the following output:



Thus, We have successfully installed Docker in our system.

NOTE: The following docker commands are useful for reference. These are as follows:

4. Experiments – Reports and Findings

In this section of the report, we will design and deal with test cases related particularly to sysbench CPU and File I/O commands. In this section, we will also test various configurations of the QEMU VM to check if we do indeed get different results by changing the configurations of the VM.

Configuration 1 : 2GB Ram and 2 Cores

a. CPU TESTING

For CPU testing between QEMU and Docker, we will use the following 3 test cases to check for performance. For our testing purposes, we will use the following sysbench command and the following test cases:

sysbench cpu --cpu-max-prime={some_value} --num-threads={some_value} --time= {some_value} run

1. max-prime = 2000 and time = 30 seconds
2. max-prime = 20,000 and time = 30 seconds
3. max-prime = 200,000 and time = 30 seconds

a. QEMU and Docker Test Results

For our first test case, we choose a relatively smaller value of max prime numbers, in this case, 2000. We run the sysbench command 5 times to get accurate readings The screenshots of the execution and findings of the experiment are as follows :

NOTE: For execution of each of these commands, we use a bash shell script which automates the execution of commands. The bash shell script will be attached along with this report.

```
Last login: Sun Apr 17 23:58:49 UTC 2022 on ttyAMA0
yash@yash:~$ ./QEMU_2000_CPU.sh
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: 113197.67

General statistics:
    total time:                          30.0001s
    total number of events:              3396012

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    4.92
         95th percentile:                        0.01
         sum:                                29711.12

Threads fairness:
    events (avg/stddev):           3396012.0000/0.00
    execution time (avg/stddev):   29.7111/0.00
```

iteration 1

```
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: 114008.22

General statistics:
    total time:                          30.0002s
    total number of events:              3420342

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    5.00
         95th percentile:                        0.01
         sum:                                29731.06

Threads fairness:
    events (avg/stddev):           3420342.0000/0.00
    execution time (avg/stddev):   29.7311/0.00
```

iteration 2

```
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: 113923.41

General statistics:
    total time:                          30.0001s
    total number of events:              3417783

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    4.95
         95th percentile:                        0.01
         sum:                                29739.03

Threads fairness:
    events (avg/stddev):           3417783.0000/0.00
    execution time (avg/stddev):   29.7390/0.00
```

iteration 3

```
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: 113948.60

General statistics:
    total time:                          30.0001s
    total number of events:              3418535

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    3.54
         95th percentile:                        0.01
         sum:                                29739.28

Threads fairness:
    events (avg/stddev):           3418535.0000/0.00
    execution time (avg/stddev):   29.7393/0.00
```

iteration 4

```
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: 114268.76

General statistics:
    total time:                          30.0001s
    total number of events:              3428159

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    8.15
         95th percentile:                        0.01
         sum:                                29736.73

Threads fairness:
    events (avg/stddev):           3428159.0000/0.00
    execution time (avg/stddev):   29.7367/0.00

yash@yash:~$
```

iteration 5

A table detailing the result of our experiment is as follows:

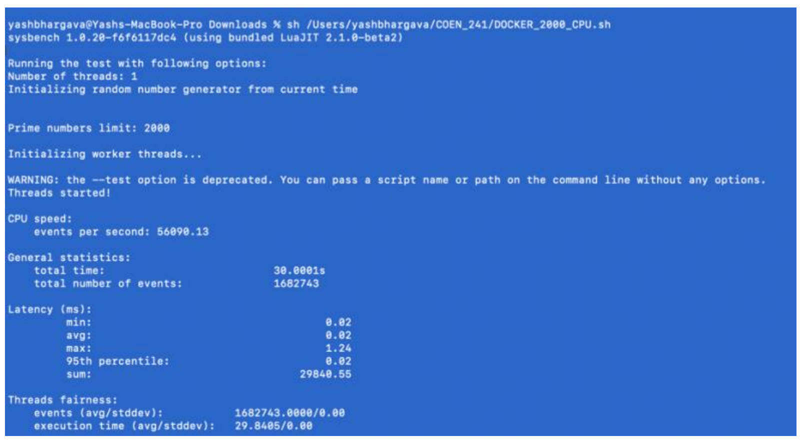
Serial Number of Iteration	Events per second
1	113197.67
2	114008.22
3	113923.41
4	113948.60
5	114268.76

Some useful observations:

- Average events per second – 113869.332
- Maximum Number of events per second recorded – 114268.76
- Minimum Number of events per second recorded – 113197.67

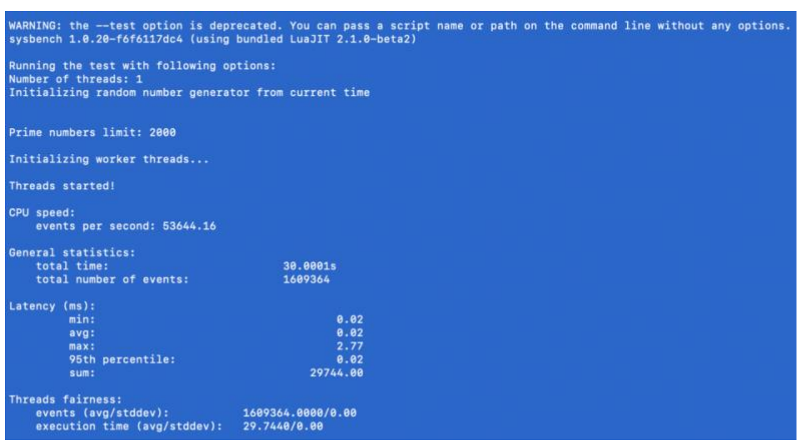
Docker:

For the same test case, the result for docker are as follows:



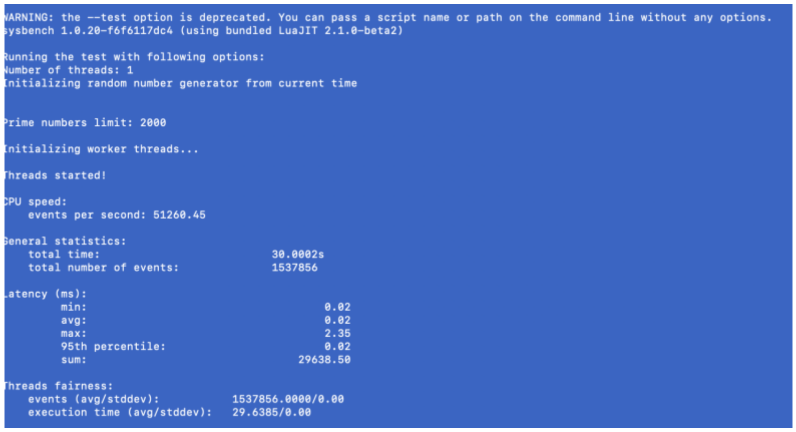
```
yashbhargava@Yashs-MacBook-Pro Downloads % sh /Users/yashbhargava/COEN_241/DOCKER_2000_CPU.sh
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

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


Prime numbers limit: 2000

Initializing worker threads...

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

CPU speed:
    events per second: 56090.13

General statistics:
    total time:                          30.0001s
    total number of events:              1682743

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    1.24
         95th percentile:                        0.02
         sum:                                29840.55

Threads fairness:
    events (avg/stddev):           1682743.0000/0.00
    execution time (avg/stddev):   29.8405/0.00
```

Iteration 1



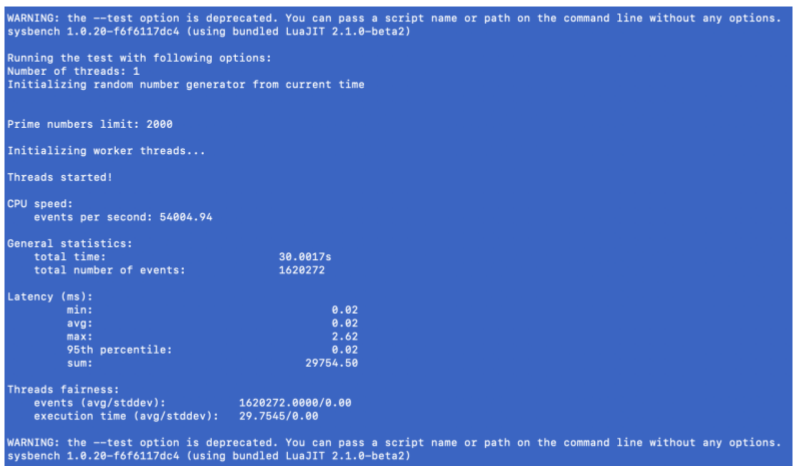
```
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

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: 53644.16

General statistics:
    total time:                          30.0001s
    total number of events:              1609364

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    2.77
         95th percentile:                        0.02
         sum:                                29744.00

Threads fairness:
    events (avg/stddev):           1609364.0000/0.00
    execution time (avg/stddev):   29.7440/0.00
```

Iteration 2



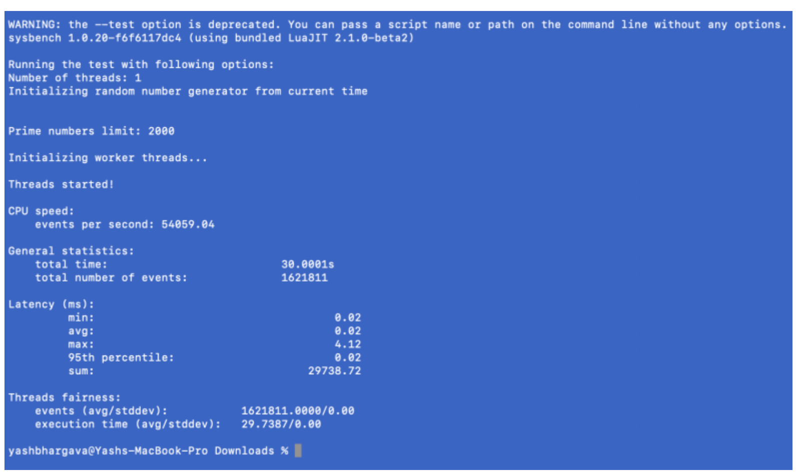
```
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

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: 51260.45

General statistics:
    total time:                          30.0002s
    total number of events:              1537856

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    2.35
         95th percentile:                        0.02
         sum:                                29638.50

Threads fairness:
    events (avg/stddev):           1537856.0000/0.00
    execution time (avg/stddev):   29.6385/0.00
```

Iteration 3



```
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: 54004.94

General statistics:
    total time:                          30.0017s
    total number of events:              1620272

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    2.62
         95th percentile:                        0.02
         sum:                                29754.50

Threads fairness:
    events (avg/stddev):           1620272.0000/0.00
    execution time (avg/stddev):   29.7545/0.00

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)
```

Iteration 4



```
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: 54059.04

General statistics:
    total time:                          30.0001s
    total number of events:              1621811

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    4.12
         95th percentile:                        0.02
         sum:                                29738.72

Threads fairness:
    events (avg/stddev):           1621811.0000/0.00
    execution time (avg/stddev):   29.7387/0.00

yashbhargava@Yashs-MacBook-Pro Downloads %
```

Iteration 5

The findings are reported in a table as follows:

Serial Number of Iteration	Number of events per second
1	56090.13
2	53644.16
3	51260.45
4	54004.96
5	54059.04

Some useful observations:

- Maximum events/second reading recorded - 56090.13
- Minimum events/second reading recorded – 51260.45
- Average events/second – 53811.74

Conclusion – We note that according to our experiment, QEMU VM is faster than Docker.

TEST CASE 2: QEMU AND DOCKER:

For our test case 2, we increase the max-prime argument value to 20,000, which is 10 times the first test case value and we keep the time parameter value constant to maintain consistency between results. Now we will check if the increase in this parameter value impacts our system performance.

QEMU:

The execution of the test case sysbench command is as follows:

```
yash@yash:~$ sysbench cpu --cpu-max-prime=20000 --time=30 run
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: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  4283.83

General statistics:
    total time:                          30.0001s
    total number of events:              128518

Latency (ms):
         min:                                    0.22
         avg:                                    0.23
         max:                                    7.01
         95th percentile:                        0.24
         sum:                                29978.36

Threads fairness:
    events (avg/stddev):           128518.0000/0.00
    execution time (avg/stddev):   29.9784/0.00

yash@yash:~$
```

Iteration 1

Upon running the command for 5 times, the findings are as follows:

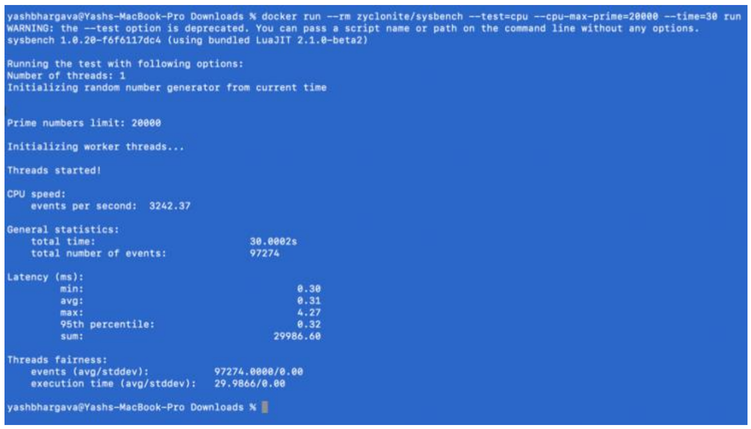
Serial Number of iteration	Events/Second
1	4283.83
2	4259.81
3	4258.24
4	4293.26
5	4296.07

Some useful observations:

- Maximum number of events per second recorded – 4296.07
- Minimum number of events per second recorded – 4258.24
- Average number of events per second recorded – 4278.24

DOCKER:

Docker execution of the test case sysbench command is as follows:



Iteration 1

Upon running the command for 5 times, the findings are as follows:

Serial Number of Iteration	Number of Events/Second
1	3242.67
2	3250.74
3	3249.12
4	3239.95
5	3084.89

Some useful observations:

- Maximum number of events per second recorded – 3250.74
- Minimum number of events per second recorded – 3084.89
- Average number of events per second – 3213.47

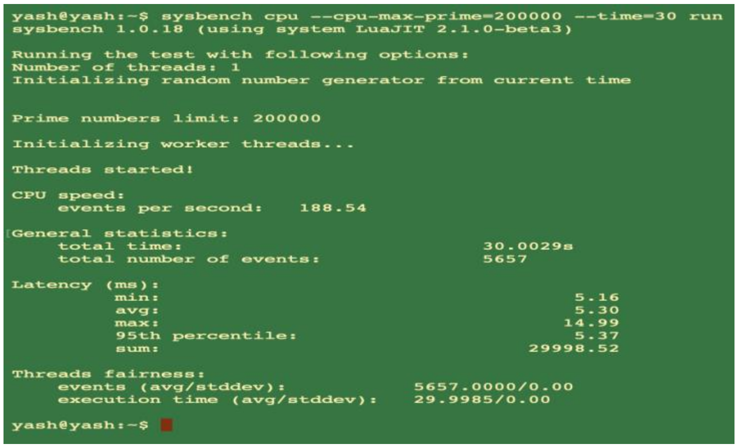
Conclusion: We see that as we increase the max-prime argument value, we see the number of events per second decrease for our current experimental configuration. QEMU still remains faster than Docker desktop on the MacBook M1 chip.

Test Case 3: QEMU AND DOCKER

For our test case 3, we increase the cpu-max-prime argument value to 200,000 while keeping the time parameter constant for result consistency. Now, we again run the test case sysbench commands for both QEMU VM and docker.

QEMU:

The execution of test case sysbench command is shown as follows:



```
yash@yash:~$ sysbench cpu --cpu-max-prime=200000 --time=30 run
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: 200000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   188.54

General statistics:
    total time:                          30.0029s
    total number of events:              5657

Latency (ms):
         min:                                    5.16
         avg:                                    5.30
         max:                                   14.99
         95th percentile:                        5.37
         sum:                                29998.52

Threads fairness:
    events (avg/stddev):           5657.0000/0.00
    execution time (avg/stddev):   29.9985/0.00

yash@yash:~$
```

Iteration 1

Upon running the command 5 times, the findings are as follows:

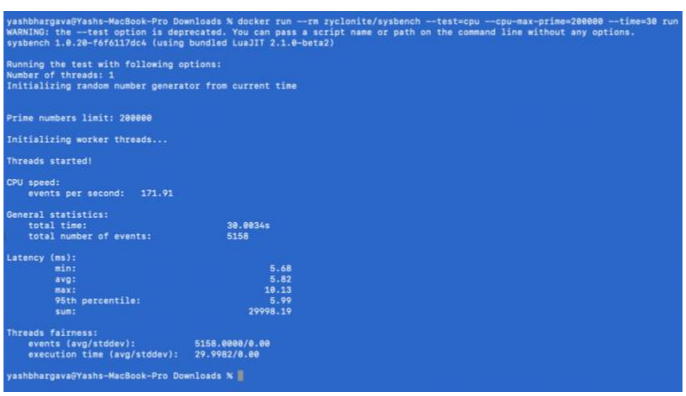
Serial Number of Iteration	Number of Events per Second
1	188.54
2	188.28
3	183.25
4	189.02
5	188.56

Some useful observations:

- Maximum number of events per second recorded – 189.02
- Minimum number of events per second recorded – 183.25
- Average number of events per second – 187.53

<u>DOCKER:</u>

The test case sysbench command execution for docker is as follows:



```
yashbhargava@Yashs-MacBook-Pro Downloads % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=200000 --time=30 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: 200000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   171.91

General statistics:
    total time:                          30.0034s
    total number of events:              5158

Latency (ms):
         min:                                    5.68
         avg:                                    5.82
         max:                                   10.13
         95th percentile:                        5.99
         sum:                                29998.19

Threads fairness:
    events (avg/stddev):           5158.0000/0.00
    execution time (avg/stddev):   29.9982/0.00

yashbhargava@Yashs-MacBook-Pro Downloads %
```

Iteration 1

Upon running the command 5 times, the findings are as follows:

Serial Number of Iteration	Number of Events per Second
1	171.91
2	172.47
3	172.49
4	172.24
5	167.53

Some useful observations:

- Maximum number of events per second recorded – 172.49
- Minimum number of events per second recorded – 167.53
- Average number of events per second – 171.32

Conclusion: We note that the number of events further decrease as we continue to increase the cpu-max-prime argument value. QEMU VM continues to be faster than Docker desktop for M1 MacBooks.

Based on our findings, we can safely conclude that as we increase the cpu-max-prime argument value, our number of events per second continue to decrease, as in, greater the value of cpu-max-prime argument, lower the number of events per second for sysbench CPU testing.

b. FILE I/O TESTING (QEMU AND DOCKER):

For File I/O testing, I will use two modes which sysbench supports, which are:

i. Sequential Rewrite (seqrewr)

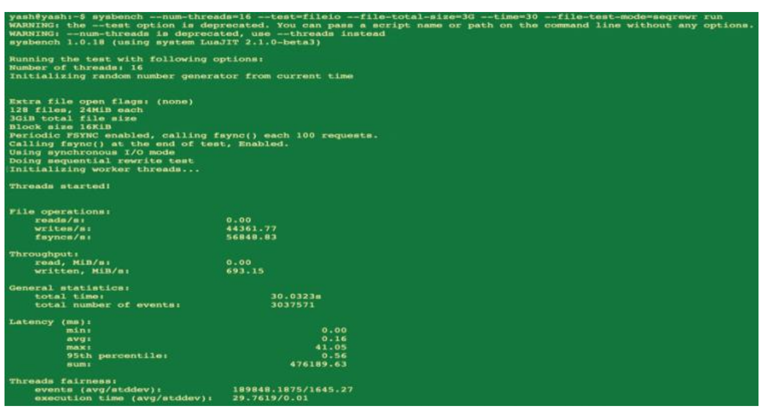
ii. Combined random read/write (rndrw)

We will keep the file size constant at 3GB and will test the performance against our experimental setups accordingly.

QEMU EXECUTION:

i. Sequential Rewrite

The execution for QEMU sequential rewrite command is shown as follows:



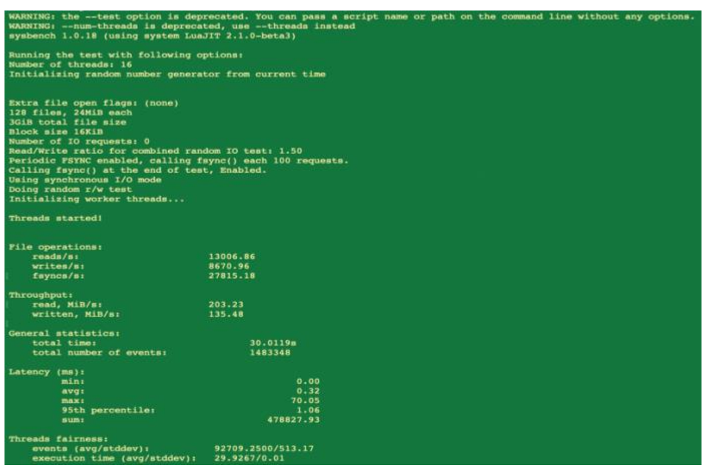
Iteration 1

After executing the command 5 times, We get the following results:

Serial number of Iteration	Results
1	reads/s = 0 writes/sec = 44361.77 fsyncs/sec = 56848.83 events/second = 101252.66
2	reads/s = 0 writes/s=44132.92 fsyncs/sec = 56554.27 events/second = 100665.5
3	reads/s = 0 writes/s = 42941.03 fsyncs/sec = 55031.80 events/second = 97985.86
4	reads/s = 0 writes/s = 40869.42 fsyncs/s = 52379.79 events/second = 93235.5
5	reads/s = 0 writes/s = 38115.98 fsyncs/s = 48856.20 events/second = 86943.5

ii. **<u>Combined Random Read Write (rndrw)</u>**

The execution of the rndrw command is shown below as follows:



```
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 1.0.18 (using system LuaJIT 2.1.0-beta3)

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


Extra file open flags: (none)
128 files, 24MiB each
3GiB 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:                      13006.86
    writes/s:                     8670.96
    fsyncs/s:                     27815.18

Throughput:
    read, MiB/s:                  203.23
    written, MiB/s:               135.48

General statistics:
    total time:                          30.0119s
    total number of events:              1483348

Latency (ms):
         min:                                    0.00
         avg:                                    0.32
         max:                                   70.05
         95th percentile:                        1.06
         sum:                               478827.93

Threads fairness:
    events (avg/stddev):           92709.2500/513.17
    execution time (avg/stddev):   29.9267/0.01
```

Iteration 1

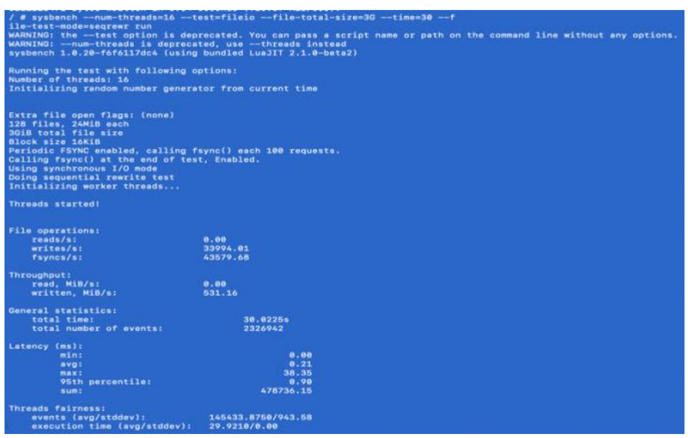
Upon running the command 5 times, we get the following results:

Serial Number of Iteration	Results
1	reads/s = 13006.86 writes/s = 8670.96 fsyncs/s = 27815.18 events/second = 49444.93
2	reads/s = 12925.26 writes/s = 8616.51 fsyncs/s = 27637.79 events/second = 49145.3
3	reads/s = 10645.87 writes/s = 7096.89 fsyncs/s =22777.00 events/second = 40479.13
4	reads/s = 11293.72 writes/s = 7529.37 fsyncs/s = 24157.81 events/seconds = 42936.03
5	reads/s = 12491.06 writes/s = 8327.07 fsyncs/s = 26713.69 events/seconds = 47486.16

DOCKER EXECUTION:

The docker execution is given as follows:

i. **Sequential Read Write**



Iteration 1

Running the command for 5 times, we get the following readings:

Serial number of Iteration	Results
1	reads/s = 0 writes/sec = 33944.01 fsyncs/sec = 43579.68 events/second = 77564.73
2	reads/s = 0 writes/s=35435.34 fsyncs/sec = 42342.32 events/second = 75343.24
3	reads/s = 0 writes/s = 33958.73 fsyncs/sec = 43684.54 events/second = 77846.34
4	reads/s = 0 writes/s = 32091.93 fsyncs/s = 41321.54 events/second = 73654.18
5	reads/s = 0 writes/s = 34546.54 fsyncs/s = 44323.20 events/second = 78234.32

ii. Combined Random Read Write

The execution for combined random read write command is as follows:

Iteration 1

Upon running the command 5 times, we have the following results:

Serial number of Iteration	Results
1	reads/s = 13950.71 writes/sec = 9300.24 fsyncs/sec = 29826.54 events/second = 53042.66
2	reads/s = 13155.49 writes/s=8769.99 fsyncs/sec = 28129.15 events/second = 50019.5
3	reads/s = 14353.95 writes/s = 9569.02 fsyncs/sec = 30688.31 events/second = 54574.26
4	reads/s = 14674.66 writes/s = 9782.83 fsyncs/s = 31370.93 events/second = 55796.33
5	reads/s = 14488.90 writes/s = 9659.14 fsyncs/s = 30975.53 events/second = 55094.1

Conclusion: We note that for sequential rewrite operations, QEMU is faster than docker desktop but in combined read write, docker is indeed faster than QEMU.

CONFIGURATION 2: 4GB RAM 4 CORES

Now, we change our system configuration and run the same tests again, to verify whether we see any performance changes. Our next configuration is 4GB ram and 4 CPU cores for both our QEMU VM and Docker.

a. CPU TESTING

i. QEMU CPU TESTING

TEST CASE 1: 2000

The execution command and the results of test case 1 are shown as below:

```
yash@yash:~$ ./QEMU_2000_CPU.sh
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: 113926.26

General statistics:
    total time:                          30.0001s
    total number of events:              3417869

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    6.30
         95th percentile:                        0.01
         sum:                                29596.91

Threads fairness:
    events (avg/stddev):           3417869.0000/0.00
    execution time (avg/stddev):   29.5969/0.00
```

Iteration 1

Upon running the command for 5 times, the results are as follows:

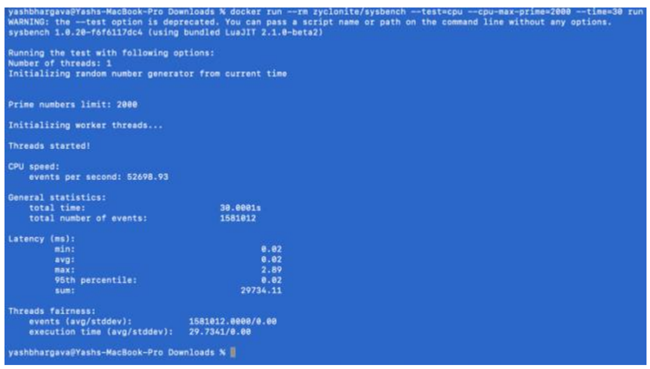
Serial Number of Iteration	Number of events per second
1	113926.26
2	113201.06
3	113018.63
4	114854.19
5	113129.79

Some useful observations:

- Maximum number of events/second = 114854.19
- Minimum number of events/second = 113018.63
- Average number of events/second = 113625.98

<u>DOCKER CASE 1 TESTING:</u>

The test case execution for docker is shown as below:



Iteration 1

Upon running the command 5 times, we get the following results:

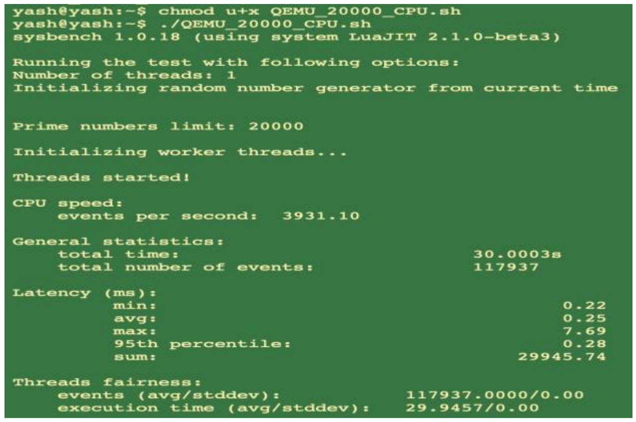
Serial Number of Iteration	Number of events per second
1	52698.93
2	52132.09
3	51893.90
4	56403.96
5	56134.25

Some useful observations:

- Maximum number of events/second = 56403.96
- Minimum number of events/second = 51893.90
- Average number of events = 53852.62

<u>QEMU TEST CASE 2:</u>

As done for the first experimental configuration, we increase the cpu-max-prime argument value to 20000 and run the test command. The execution for QEMU VM command is as follows:



```
yash@yash:~$ chmod u+x QEMU_20000_CPU.sh
yash@yash:~$ ./QEMU_20000_CPU.sh
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: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  3931.10

General statistics:
    total time:                          30.0003s
    total number of events:              117937

Latency (ms):
         min:                                    0.22
         avg:                                    0.25
         max:                                    7.69
         95th percentile:                        0.28
         sum:                                29945.74

Threads fairness:
    events (avg/stddev):           117937.0000/0.00
    execution time (avg/stddev):   29.9457/0.00
```

Iteration 1

Upon running the command 5 times, we get the following result:

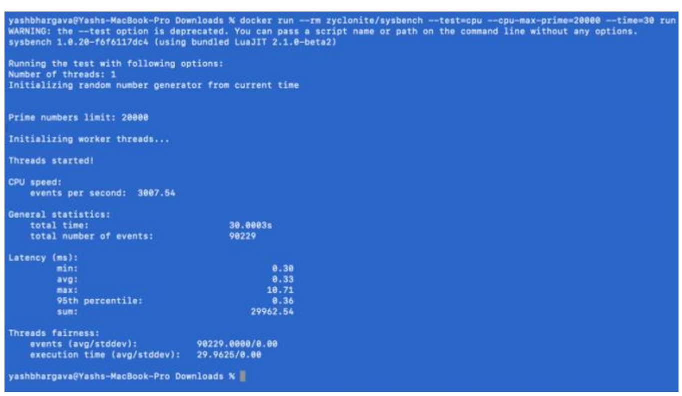
Serial Number of iteration	Events/Second
1	3931.10
2	3933.90
3	4281.01
4	4051.55
5	3972.59

Some useful observations:

- Maximum number of events/second = 4281.01
- Minimum number of events/second = 4051.55
- Average number of events/second = 4034.03

<u>TEST CASE 2 DOCKER EXECUTION:</u>

The docker execution is shown as below:



```
yashbhargava@Yashs-MacBook-Pro Downloads % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=30 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: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  3007.54

General statistics:
    total time:                          30.0003s
    total number of events:              90229

Latency (ms):
         min:                                    0.30
         avg:                                    0.33
         max:                                   10.71
         95th percentile:                        0.36
         sum:                                29962.54

Threads fairness:
    events (avg/stddev):           90229.0000/0.00
    execution time (avg/stddev):   29.9625/0.00

yashbhargava@Yashs-MacBook-Pro Downloads %
```

Iteration 1

Upon running the command for 5 times, we get the following results:

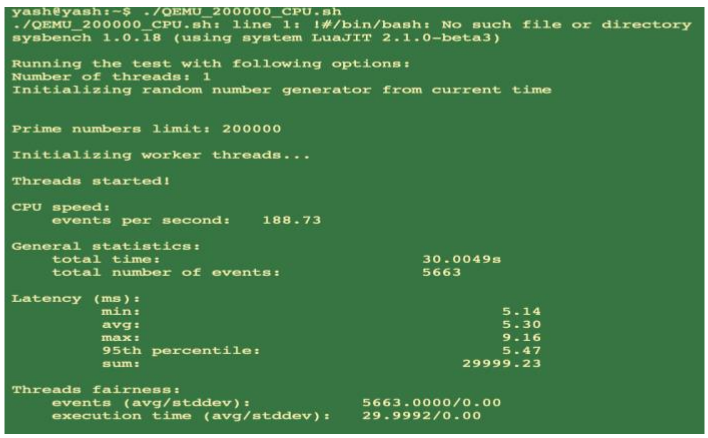
Serial Number of iteration	Events/Second
1	3007.54
2	3247.12
3	3181.35
4	3164.53
5	2917.72

Some useful observations:

- Maximum number of events/second = 3247.12
- Minimum number of events/second = 2917.72
- Average number of events/second = 3103.65

TEST CASE 3: QEMU

The execution of test case 3 in QEMU VM is shown as below:



```
yash@yash:~$ ./QEMU_200000_CPU.sh
./QEMU_200000_CPU.sh: line 1: !#/bin/bash: No such file or directory
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: 200000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   188.73

General statistics:
    total time:                          30.0049s
    total number of events:              5663

Latency (ms):
         min:                                    5.14
         avg:                                    5.30
         max:                                    9.16
         95th percentile:                        5.47
         sum:                                29999.23

Threads fairness:
    events (avg/stddev):           5663.0000/0.00
    execution time (avg/stddev):   29.9992/0.00
```

Iteration 1

Upon running the command for 5 times, we get the following result:

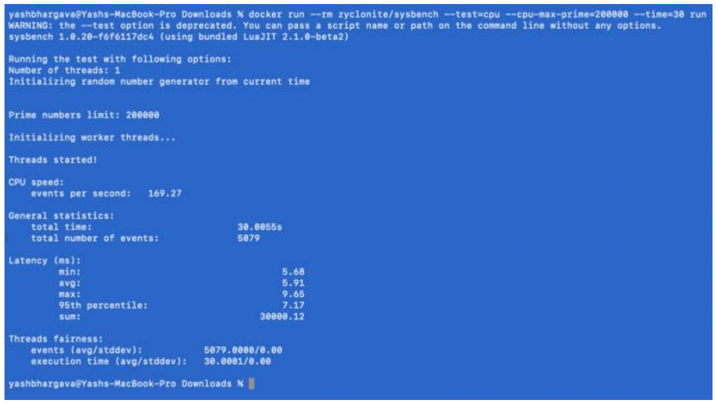
Serial Number of iteration	Events/Second
1	188.73
2	188.33
3	185.49
4	189.86
5	189.83

Some useful observations:

- Maximum number of events/second – 188.73
- Minimum number of events/second – 185.49
- Average number of events/second – 188.44

TEST CASE 3: DOCKER EXECUTION

The docker execution is shown as below:


```
yashbhargava@Yashs-MacBook-Pro Downloads % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=200000 --time=30 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: 200000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   169.27

General statistics:
    total time:                          30.0055s
    total number of events:              5079

Latency (ms):
         min:                                    5.68
         avg:                                    5.91
         max:                                    9.65
         95th percentile:                        7.17
         sum:                                30000.12

Threads fairness:
    events (avg/stddev):           5079.0000/0.00
    execution time (avg/stddev):   30.0001/0.00

yashbhargava@Yashs-MacBook-Pro Downloads %
```

Iteration 1

Upon running the command 5 times, we get the following results:

Serial Number of iteration	Events/Second
1	169.27
2	172.51
3	172.34
4	171.67
5	173.24

Some useful observations:

- Maximum number of events/second = 173.24
- Minimum number of events/second = 169.27
- Average number of events/second = 171.80

<u>CONCLUSION – We still see that QEMU remains faster than docker desktop in terms of CPU Performance. We note that there is no significant change in performance if we increase the system configuration.</u>

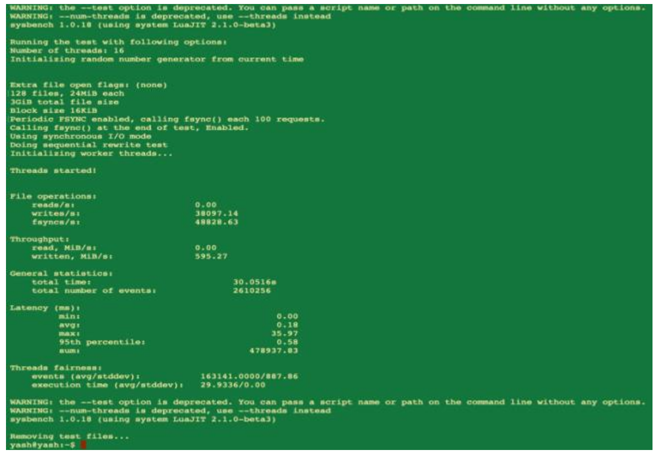
b. <u>FILE I/O TESTING</u>

i. <u>QEMU FILE I/O</u>

For QEMU, we keep the same file io test cases as above.

a. <u>Sequential Rewrite (QEMU)</u>

The execution for QEMU VM is shown below:



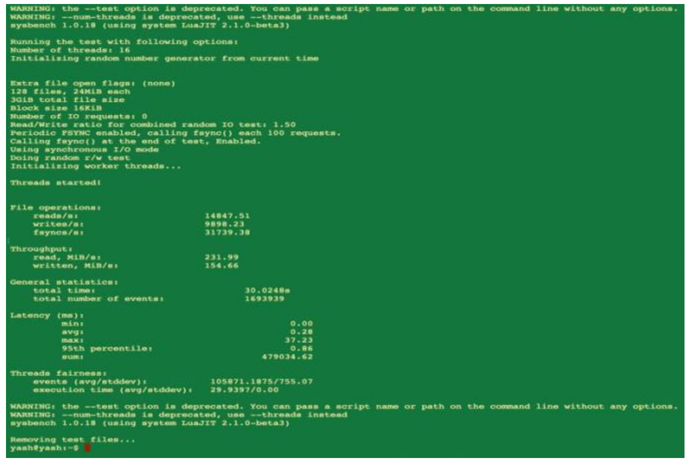
Iteration 1

Upon running the command for 5 times, we get the following results:

Serial number of Iteration	Results
1	reads/s = 0 writes/sec = 38097.14 fsyncs/sec = 48828.63 events/second = 87008.56
2	reads/s = 0 writes/s=39530.62 fsyncs/sec = 50664.21 events/second = 90199.3
3	reads/s = 0 writes/s = 35886.20 fsyncs/sec = 46000.81 events/second = 81888.26
4	reads/s = 0 writes/s = 33329.60 fsyncs/s = 42727.80 events/second = 76088.93
5	reads/s = 0 writes/s = 33704.58 fsyncs/s = 43206.91 events/second = 76908.83

b. <u>Combined Random Read Write (QEMU VM)</u>

The execution for combined random read write for QEMU VM is shown below:



Iteration 1

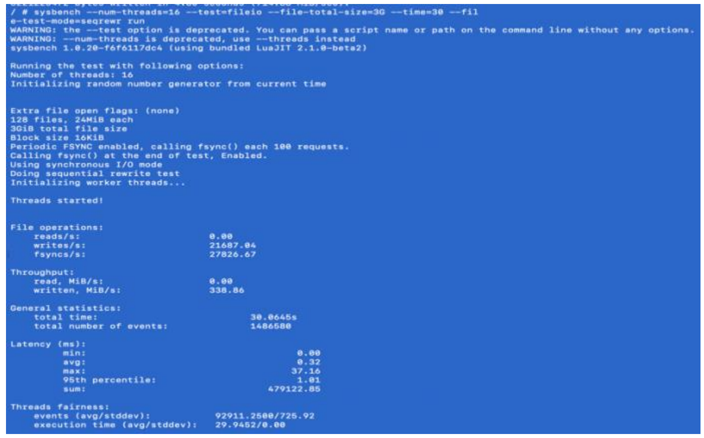
Upon running the command 5 times, we get the following result:

Serial Number of Iteration	Results
1	reads/s = 14487.51 writes/s = 9898.23 fsyncs/s = 31739.38 events/second = 56464.43
2	reads/s = 13910 writes/s = 9273 fsyncs/s = 29740 events/second = 52915.76
3	reads/s = 14165.54 writes/s = 9443.58 fsyncs/s =30284.24 events/second = 53880.36
4	reads/s = 12349.31 writes/s = 8232.87 fsyncs/s = 26409.53 events/seconds = 46994.56
5	reads/s = 12001.06 writes/s = 8000.63 fsyncs/s = 25669.69 events/seconds =45638.56

ii. <u>FILE I/O (DOCKER) EXECUTIONS</u>

a. <u>Sequential Rewrite</u>

The execution for sequential read write in docker is shown below



```
/ # sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --fil
e-test-mode=seqrewr 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, 24MiB each
3GiB 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 rewrite test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     21687.04
    fsyncs/s:                     27826.67

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               338.86

General statistics:
    total time:                          30.0645s
    total number of events:              1486580

Latency (ms):
         min:                                    0.00
         avg:                                    0.32
         max:                                   37.16
         95th percentile:                        1.01
         sum:                               479122.85

Threads fairness:
    events (avg/stddev):           92911.2500/725.92
    execution time (avg/stddev):   29.9452/0.00
```

Iteration 1

Upon running the command 5 times, we get the following results:

Serial number of Iteration	Results
1	reads/s = 0 writes/sec = 21687.04 fsyncs/sec = 27826.67 events/second = 49552.66
2	reads/s = 0 writes/s= 22690.64 fsyncs/sec = 29019.49 events/second = 51806.5
3	reads/s = 0 writes/s = 25688.43 fsyncs/sec = 32946.24 events/second = 58659.23
4	reads/s = 0 writes/s = 22382.72 fsyncs/s = 28716.33 events/second = 51146.33
5	reads/s = 0 writes/s = 21880.69 fsyncs/s =28071.27 events/second = 49996.26

b. <u>Combined Random Read Write (Docker)</u>

The execution for combined read write for docker is shown below:

Iteration 1

Upon running the command 5 times, we get the following result:

Serial number of Iteration	Results
1	reads/s = 13850.71 writes/sec = 9320.24 fsyncs/sec = 28826.54 events/second = 53032.66
2	reads/s = 14155.49 writes/s=8754.94 fsyncs/sec = 27129.15 events/second = 50045.5
3	reads/s = 14251.83 writes/s = 9545.02 fsyncs/sec = 30457.31 events/second = 54375.26
4	reads/s = 14324.66 writes/s = 9231.83 fsyncs/s = 31432.93 events/second = 55987.33
5	reads/s = 14231.90 writes/s = 9658.14 fsyncs/s = 30324.53 events/second = 55432.1

CONCLUSION – When compared to earlier experiment configuration (2GB RAM 2 Cores) we see a little decrease in file i/o in both QEMU VM and Docker Desktop.

TEST CONFIGURATION 3 (6GB RAM 6 CORES)

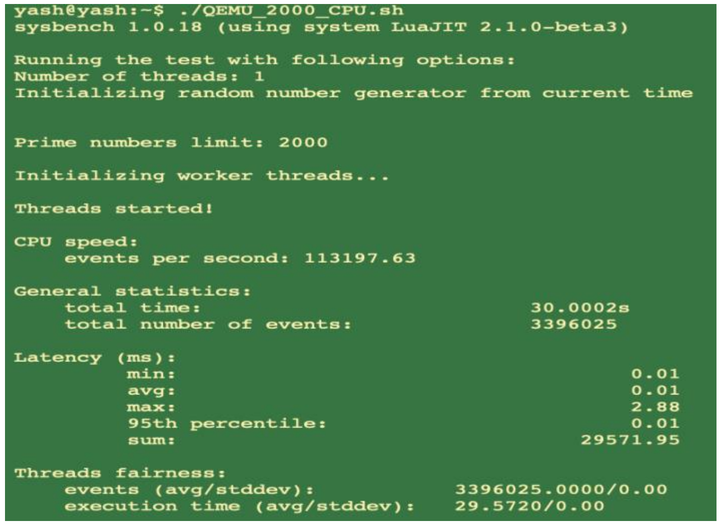
Now, we construct a QEMU VM with 6GB ram and 6 cores and we also limit docker resource utilization to 6 cores and 6GB RAM. Now, we test the same commands to verify whether we see an increase In performance.

a. CPU TESTING

i. QEMU AND DOCKER EXECUTIONS

TEST CASE 1 EXECUTION (QEMU)

The execution for first test case is shown below:



```
yash@yash:~$ ./QEMU_2000_CPU.sh
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: 113197.63

General statistics:
    total time:                          30.0002s
    total number of events:              3396025

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    2.88
         95th percentile:                        0.01
         sum:                                29571.95

Threads fairness:
    events (avg/stddev):           3396025.0000/0.00
    execution time (avg/stddev):   29.5720/0.00
```

Iteration 1

Upon running the command 5 times, we get the following results:

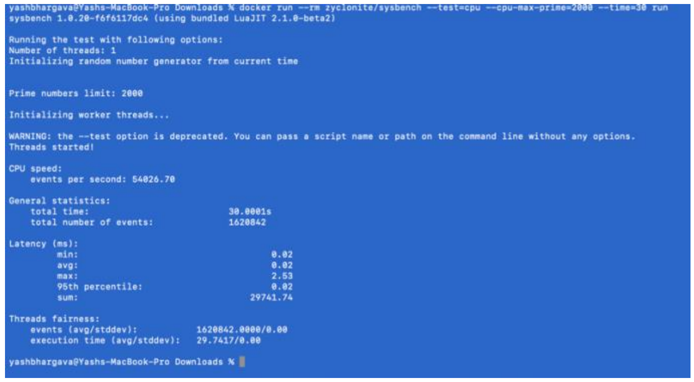
Serial Number of Iteration	Number of events per second
1	113197.63
2	114251.46
3	113318.53
4	114254.59
5	113529.89

Some useful observations:

- Maximum number of events/second = 114254.59
- Minimum number of events/second = 113197.63
- Average number of events/second = 113710.42

TEST CASE 1 EXECUTION (DOCKER)

The test case 1 execution for docker is shown as below:



```
yashbhargava@Yashs-MacBook-Pro Downloads % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=2000 --time=30 run
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...

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

CPU speed:
    events per second: 54026.70

General statistics:
    total time:                          30.0001s
    total number of events:              1620842

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    2.53
         95th percentile:                        0.02
         sum:                                29741.74

Threads fairness:
    events (avg/stddev):           1620842.0000/0.00
    execution time (avg/stddev):   29.7417/0.00

yashbhargava@Yashs-MacBook-Pro Downloads %
```

Iteration 1

Upon running the command 5 times, we get the following results:

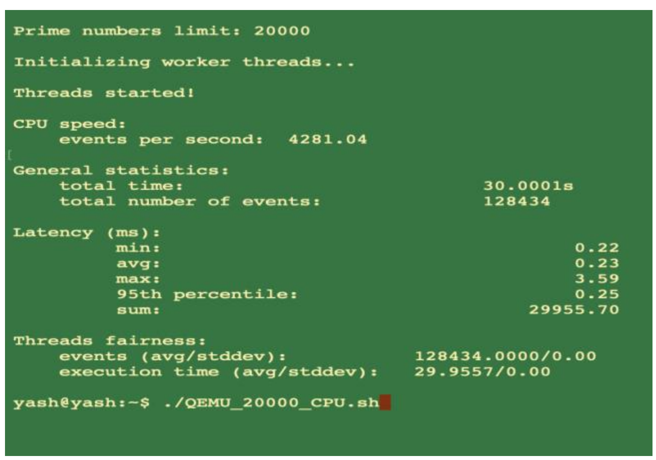
Serial Number of Iteration	Number of events per second
1	54026.70
2	52534.09
3	52843.95
4	53403.88
5	56479.38

Some useful observations:

- Maximum number of events/second = 56479.38
- Minimum number of events/second = 52534.09
- Average number of events/second = 53857.6

TEST CASE 2 (QEMU VM)

The execution of test case 2 in QEMU VM is shown below:



```
Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  4281.04

General statistics:
    total time:                          30.0001s
    total number of events:              128434

Latency (ms):
         min:                                    0.22
         avg:                                    0.23
         max:                                    3.59
         95th percentile:                        0.25
         sum:                                29955.70

Threads fairness:
    events (avg/stddev):           128434.0000/0.00
    execution time (avg/stddev):   29.9557/0.00

yash@yash:~$ ./QEMU_20000_CPU.sh
```

Iteration 1

Upon running the command 5 times, we get the following result:

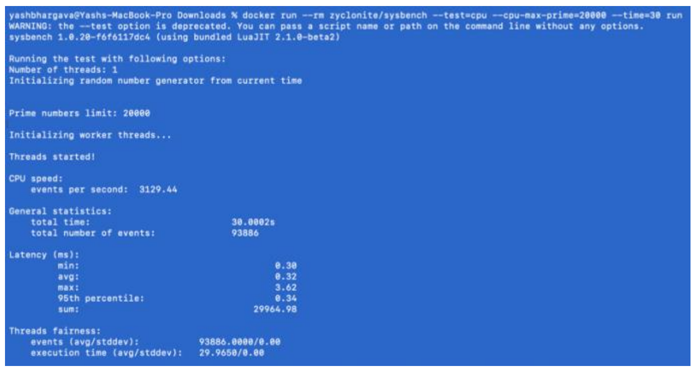
Serial Number of iteration	Events/Second
1	4281.04
2	3945.40
3	4284.01
4	4054.45
5	3942.69

Some useful observations:

- Maximum number of events/second = 4284.01
- Minimum number of events/second = 3942.69
- Average number of events/second = 4101.51

<u>TEST CASE 2 (DOCKER)</u>

The test case 2 execution for docker is shown as below:



```
yashbhargava@Yashs-MacBook-Pro Downloads % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=30 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: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  3129.44

General statistics:
    total time:                          30.0002s
    total number of events:              93886

Latency (ms):
         min:                                    0.30
         avg:                                    0.32
         max:                                    3.62
         95th percentile:                        0.34
         sum:                                29964.98

Threads fairness:
    events (avg/stddev):           93886.0000/0.00
    execution time (avg/stddev):   29.9650/0.00
```

Iteration 1

Upon running the command 5 times, we get the following results:

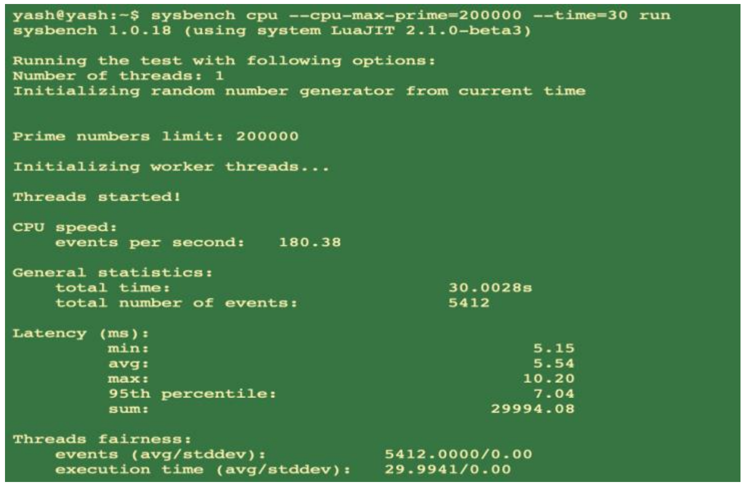
Serial Number of iteration	Events/Second
1	3129.44
2	3147.42
3	3141.35
4	3264.53
5	3024.72

Some usual observations:

- Maximum number of events/second = 3264.53
- Minimum Number of events/second = 3024.72
- Average events/second = 3141.49

<u>TEST CASE 3 (QEMU) EXECUTION:</u>

The test case 3 execution for QEMU VM is shown below:



```
yash@yash:~$ sysbench cpu --cpu-max-prime=200000 --time=30 run
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: 200000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   180.38

General statistics:
    total time:                          30.0028s
    total number of events:              5412

Latency (ms):
         min:                                    5.15
         avg:                                    5.54
         max:                                   10.20
         95th percentile:                        7.04
         sum:                                29994.08

Threads fairness:
    events (avg/stddev):           5412.0000/0.00
    execution time (avg/stddev):   29.9941/0.00
```

Iteration 1

Upon running the command for 5 times, we get the following results:

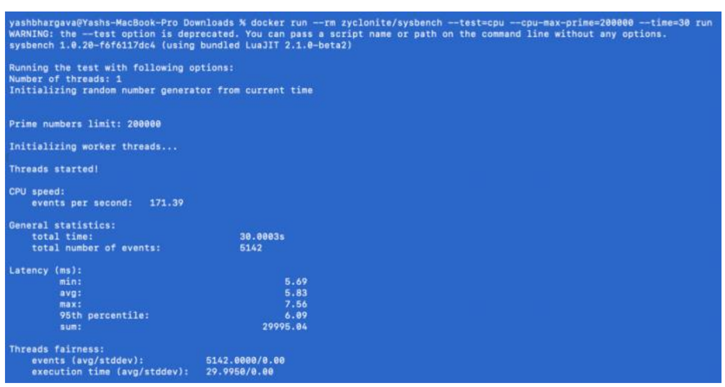
Serial Number of iteration	Events/Second
1	180.38
2	182.33
3	184.49
4	180.86
5	181.83

Some useful observations:

- Maximum events/second = 184.49
- Minimum events/second = 180.38
- Average events/second = 181.97

TEST CASE 3 DOCKER EXECUTION:

The execution for test case 3 for docker is as shown below:



```
yashbhargava@Yashs-MacBook-Pro Downloads % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=200000 --time=30 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: 200000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   171.39

General statistics:
    total time:                          30.0003s
    total number of events:              5142

Latency (ms):
         min:                                    5.69
         avg:                                    5.83
         max:                                    7.56
         95th percentile:                        6.09
         sum:                                29995.04

Threads fairness:
    events (avg/stddev):           5142.0000/0.00
    execution time (avg/stddev):   29.9950/0.00
```

Iteration 1

Upon running the command 5 times, we get the following results:

Serial Number of iteration	Events/Second
1	171.39
2	172.47
3	171.34
4	172.67
5	174.24

Some useful observations:

- Maximum events/second = 174.24
- Minimum events/second = 171.34
- Average events/second = 172.42

CONCLUSION: Even after increasing our resources, we see no significant changes in the cpu command performance for both QEMU and Docker desktop. There are only slight variations.

ii. **FILE I/O Testing**

a. **QEMU FILE I/O TESTING**

i. **Sequential Rewrite**

The sequential rewrite testing for QEMU is shown below.

```
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: 16
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 24MiB each
3GiB 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 rewrite test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     25473.58
    fsyncs/s:                     32673.72

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               398.02

General statistics:
    total time:                          30.0533s
    total number of events:              1745550

Latency (ms):
         min:                                    0.00
         avg:                                    0.27
         max:                                   39.67
         95th percentile:                        0.81
         sum:                               479121.05

Threads fairness:
    events (avg/stddev):           109096.8750/702.37
    execution time (avg/stddev):   29.9451/0.00
```

Iteration 1

Upon running the command for 5 times, we get the following results:

Serial number of Iteration	Results
1	reads/s = 0 writes/sec = 25473.58 fsyncs/sec = 32673.72 events/second = 58185
2	reads/s = 0 writes/s=27687.54 fsyncs/sec = 33764.65 events/second = 59124.32
3	reads/s = 0 writes/s = 24534.32 fsyncs/sec = 31986.87 events/second = 56126.76
4	reads/s = 0 writes/s = 25765.32 fsyncs/s = 32675.83 events/second = 58234.21
5	reads/s = 0 writes/s = 28675.77 fsyncs/s = 35687.32 events/second = 59543.65

ii. **Combined Random Read and Write**

The execution for combined random read and write is as follows:

```
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: 16
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 24MiB each
3GiB 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:                      9938.58
    writes/s:                     6625.83
    fsyncs/s:                     21268.99

Throughput:
    read, MiB/s:                  155.29
    written, MiB/s:               103.53

General statistics:
    total time:                          30.0701s
    total number of events:              1135621

Latency (ms):
         min:                                    0.00
         avg:                                    0.42
         max:                                   37.10
         95th percentile:                        1.44
         sum:                               479198.63

Threads fairness:
    events (avg/stddev):           70976.3125/665.20
    execution time (avg/stddev):   29.9499/0.00
```

Iteration 1

Upon running the command 5 times, we get the following result:

Serial Number of Iteration	Results
1	reads/s = 9938.58 writes/s = 6625.83 fsyncs/s = 21268.98 events/second = 37854.03
2	reads/s = 9845.43 writes/s = 6541.32 fsyncs/s = 20267.32 events/second = 35643.76
3	reads/s = 9876.73 writes/s = 6543.21 fsyncs/s =21124.43 events/second = 35541.28
4	reads/s = 9765.53 writes/s = 7021.32 fsyncs/s = 30832.11 events/seconds = 39878.32
5	reads/s = 9763.23 writes/s = 6312.63 fsyncs/s = 24569.69 events/seconds =34532.56

b. DOCKER FILE I/O TESTING

a. Sequential Rewrite

The sequential rewrite execution for docker is shown below:

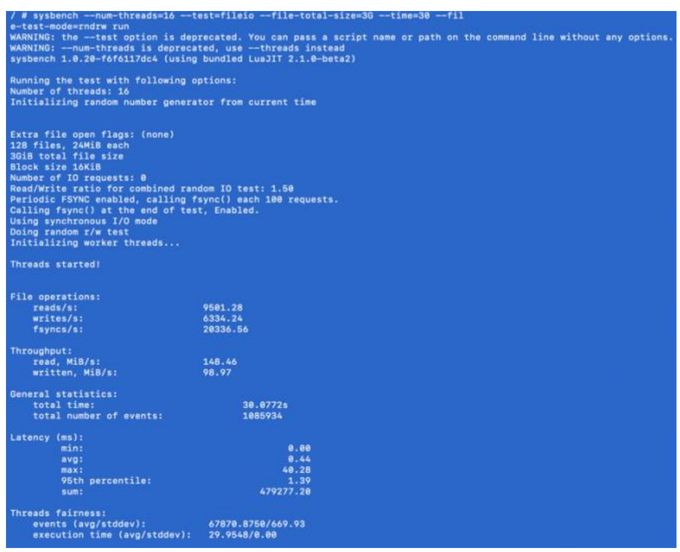
Iteration 1

Upon running the commands for 5 times, we get the following results:

Serial number of Iteration	Results
1	reads/s = 0 writes/sec = 13778.77 fsyncs/sec = 17701.47 events/second = 31506.16
2	reads/s = 0 writes/s= 13543.64 fsyncs/sec = 17634.49 events/second = 30234.5
3	reads/s = 0 writes/s = 12456.32 fsyncs/sec = 16323.87 events/second = 29876.36
4	reads/s = 0 writes/s = 14328.87 fsyncs/s = 18785.62 events/second = 32324.32
5	reads/s = 0 writes/s = 13534.11 fsyncs/s =17431.89 events/second = 31232.21

b. <u>Combined Random Read and Write (Docker)</u>

The execution for docker is shown as below:


```
/ # sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --fil
e-test-mode=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, 24MiB each
3GiB 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:                      9501.28
    writes/s:                     6334.24
    fsyncs/s:                     20336.56

Throughput:
    read, MiB/s:                  148.46
    written, MiB/s:               98.97

General statistics:
    total time:                          30.0772s
    total number of events:              1085934

Latency (ms):
         min:                                    0.00
         avg:                                    0.44
         max:                                   40.28
         95th percentile:                        1.39
         sum:                               479277.20

Threads fairness:
    events (avg/stddev):           67870.8750/669.93
    execution time (avg/stddev):   29.9548/0.00
```

Iteration 1

Upon running the command 5 times, we get the following results:

Serial number of Iteration	Results
1	reads/s = 9501.28 writes/sec = 6334.24 fsyncs/sec = 20336.56 events/second = 36197.8
2	reads/s = 9323.32 writes/s=6213.41 fsyncs/sec = 20223.12 events/second = 36097.87
3	reads/s = 9674.98 writes/s = 6984.73 fsyncs/sec = 20989.89 events/second = 37019.21
4	reads/s = 9594.32 writes/s = 6434.43 fsyncs/s = 20214.43 events/second = 35788.33
5	reads/s = 9287.90 writes/s = 6021.14 fsyncs/s = 18673.53 events/second = 32764.12

CONCLUSION: We see that the file I/O performance actually decreases if we increase resource allocation according to the experiments conducted above. Docker performs similar to QEMU although most times QEMU outperforms docker.

5. PERFORMANCE ANALYSIS

a. QEMU DISK UTILIZATION

A. 2GB RAM 2 CORES

i. **Sequential Rewrite**
read, MiB/s = 0
written, MiB/s = 693.15

ii. **Combined Random Read Write**
read, MiB/s = 203.23
written, MiB/s = 135.48

B. 4GB RAM 4 CORES

i. **Sequential Rewrite**
read, MiB/s = 0
written, MiB/s = 595.27

ii. **Combined Random Read Write**
read, MiB/s = 231.99
written, MiB/s = 154.66

C. 6GB RAM 6 Cores

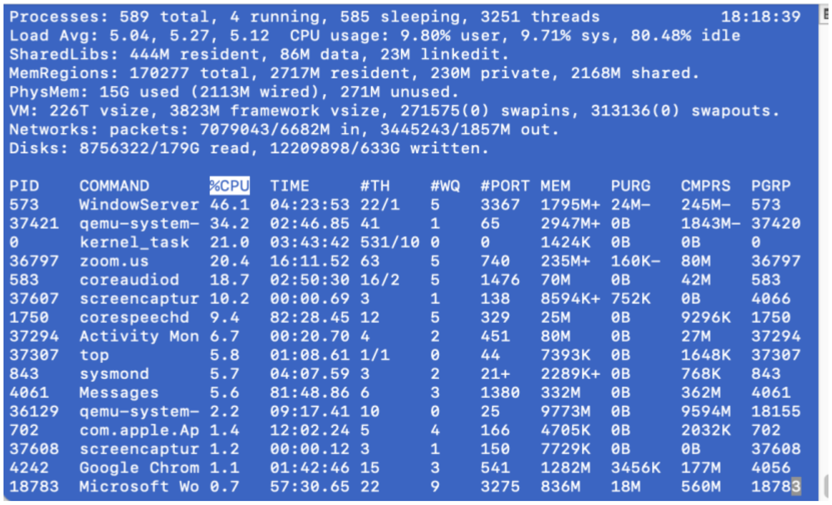
i. **Sequential Rewrite**
read, MiB/s = 0
written, MiB/s = 398.02

ii. **Combined Random Read Write**

read, MiB/s = 155.29
written, MiB/s = 103.53

b. QEMU CPU UTILIZATION

Google C...	11.8	19:19.38	24	115	Apple	0.0	0.00	18913	yashbhargava
Messages	6.4	1:25:36.87	6	132	Apple	0.0	0.00	4061	yashbhargava
qemu-sy...	3.2	11:58.50	14	96	Apple	0.0	0.00	36129	yashbhargava



```
Processes: 589 total, 4 running, 585 sleeping, 3251 threads          18:18:39
Load Avg: 5.04, 5.27, 5.12  CPU usage: 9.80% user, 9.71% sys, 80.48% idle
SharedLibs: 444M resident, 86M data, 23M linkedit.
MemRegions: 170277 total, 2717M resident, 230M private, 2168M shared.
PhysMem: 15G used (2113M wired), 271M unused.
VM: 226T vsize, 3823M framework vsize, 271575(0) swapins, 313136(0) swapouts.
Networks: packets: 7079043/6682M in, 3445243/1857M out.
Disks: 8756322/179G read, 12209898/633G written.

PID    COMMAND      %CPU TIME     #TH    #WQ  #PORT MEM    PURG   CMPRS  PGRP
573    WindowServer 46.1 04:23:53 22/1   5    3367  1795M+ 24M-   245M-  573
37421  qemu-system- 34.2 02:46.85 41     1    65    2947M+ 0B     1843M- 37420
0      kernel_task  21.0 03:43:42 531/10 0    0     1424K  0B     0B     0
36797  zoom.us      20.4 16:11.52 63     5    740   235M+  160K-  80M    36797
583    coreaudiod   18.7 02:50:30 16/2   5    1476  70M    0B     42M    583
37607  screencaptur 10.2 00:00.69 3      1    138   8594K+ 752K   0B     4066
1750   corespeechd  9.4  82:28.45 12     5    329   25M    0B     9296K  1750
37294  Activity Mon 6.7  00:20.70 4      2    451   80M    0B     27M    37294
37307  top          5.8  01:08.61 1/1    0    44    7393K  0B     1648K  37307
843    sysmond      5.7  04:07.59 3      2    21+   2289K+ 0B     768K   843
4061   Messages     5.6  81:48.86 6      3    1380  332M   0B     362M   4061
36129  qemu-system- 2.2  09:17.41 10     0    25    9773M  0B     9594M  18155
702    com.apple.Ap 1.4  12:02.24 5      4    166   4705K  0B     2032K  702
37608  screencaptur 1.2  00:00.12 3      1    150   7729K  0B     0B     37608
4242   Google Chrom 1.1  01:42:46 15     3    541   1282M  3456K  177M   4056
18783  Microsoft Wo 0.7  57:30.65 22     9    3275  836M   18M    560M   18783
```

Percentage of CPU used – 34.2%
Kernel Usage – user = 9.71% , System – 9.71%, Idle – 80.48%

c. Docker Disk Utilization

A. 2GB RAM 2 CORES

i. **Sequential Rewrite**
read, MiB/s = 0
written, MiB/s = 531.16

ii. **Combined Random Read Write**
read, MiB/s = 217.98
written, MiB/s = 145.32

B. 4GB RAM 4 CORES

i. **Sequential Rewrite**
read, MiB/s = 0
written, MiB/s = 338.86

ii. **Combined Random Read Write**
read, MiB/s = 213.43
written, MiB/s = 142.28

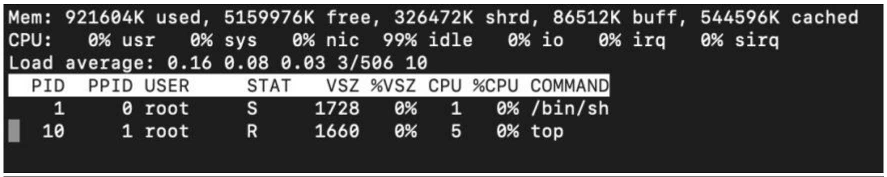
C. 6GB RAM 6 CORES

i. **Sequential Rewrite**
read, MiB/s = 0
written, MiB/s = 215.29

ii. **<u>Combined Random Read Write</u>**

read, MiB/s = 148.46
written, MiB/s = 98.97

d. **<u>Docker CPU Utilization</u>**



```
Mem: 921604K used, 5159976K free, 326472K shrd, 86512K buff, 544596K cached
CPU:   0% usr   0% sys   0% nic  99% idle   0% io   0% irq   0% sirq
Load average: 0.16 0.08 0.03 3/506 10
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
    1     0 root     S     1728   0%   1   0% /bin/sh
   10     1 root     R     1660   0%   5   0% top
```

Docker D...	0.2	3:03.96	9	0	Apple	0.0	3.29	18207	yashbhargava
Docker D...	0.0	4:45.15	27	1	Apple	0.0	0.00	18202	yashbhargava

CPU Time used – 4hrs 45mins
Kernel used – 921604Kb => 115.2 MB

6. **<u>Github Repository Information</u>**

Account Name – yash-bhargava18
Repository Name – COEN241
Folder which contains the assignment – Homework 1
Link to Repository - https://github.com/yash-bhargava18/COEN241
commit id - ca518f4b0c2ea39d2c5dc4f6966aa58b3eb97685