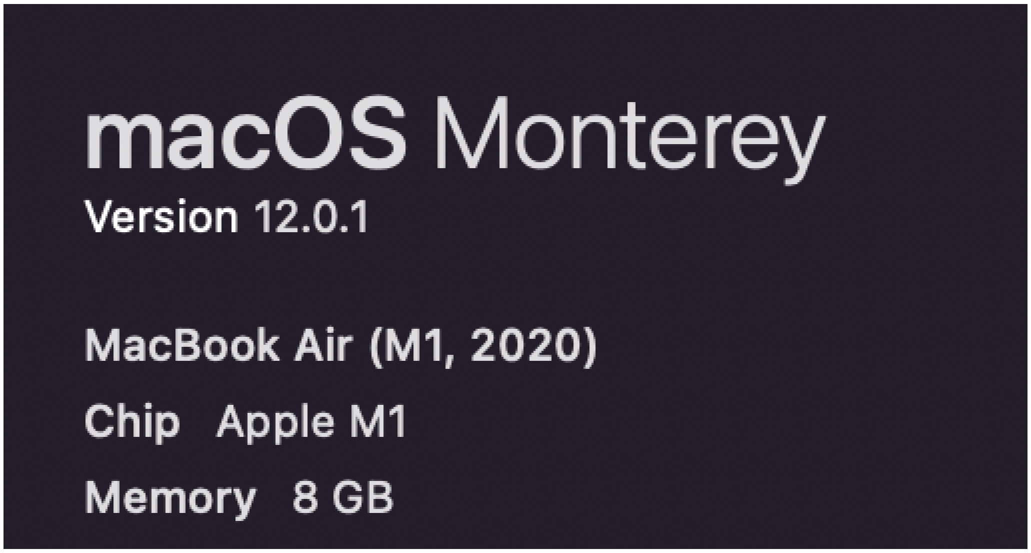
COEN 241: Cloud Computing
System vs OS Virtualization

Janan Gandhi (W1607782)

Table of contents

I. System configuration

The experiments for the virtual QEMU machine and the docker container have been run on Mac M1 with Apple Silicon Chip with 8GB RAM.



macOS Monterey
Version 12.0.1

MacBook Air (M1, 2020)
Chip Apple M1
Memory 8 GB

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 M1 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 \

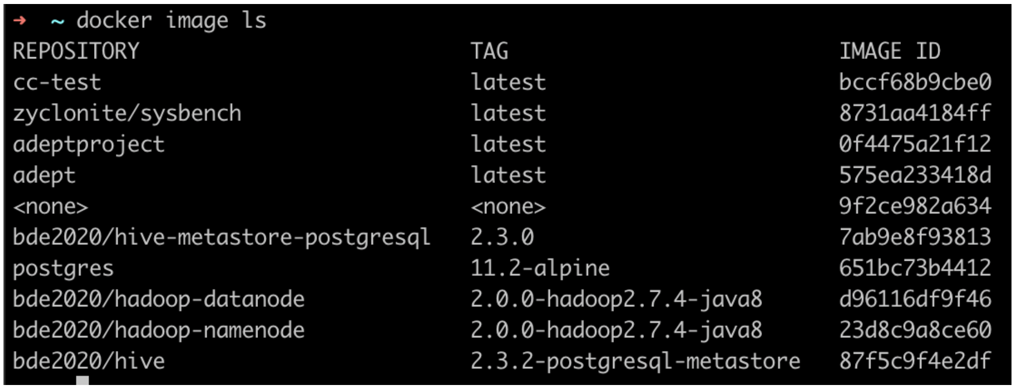
```
        -M virt,highmem=off  -drive file=/opt/homebrew/Cellar/qemu/6.2.0_1/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
        -drive file=ovmf_vars.fd,if=pflash,format=raw serial telnet::4444,server,nowait \
        -drive if=none,file=disk.qcow2,format=qcow2,id=hd0 \
        -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-arm64.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

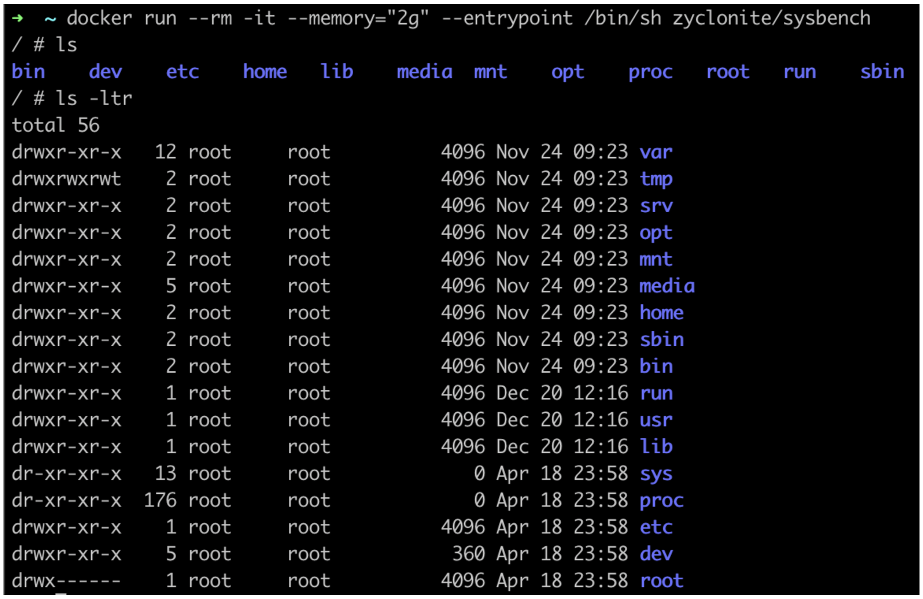
As I was running into issues while trying to install native Docker onto my Apple Silicon MAC, I had to install Docker Desktop for Apple Silicon from the official Docker website - https://docs.docker.com/desktop/mac/apple-silicon/. After installation, I pulled the zyclonite/sysbench image using the docker 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:



```
➜  ~ docker image ls
REPOSITORY                         TAG                          IMAGE ID
cc-test                            latest                       bccf68b9cbe0
zyclonite/sysbench                 latest                       8731aa4184ff
adeptproject                       latest                       0f4475a21f12
adept                              latest                       575ea233418d
<none>                             <none>                       9f2ce982a634
bde2020/hive-metastore-postgresql  2.3.0                        7ab9e8f93813
postgres                           11.2-alpine                  651bc73b4412
bde2020/hadoop-datanode            2.0.0-hadoop2.7.4-java8      d96116df9f46
bde2020/hadoop-namenode            2.0.0-hadoop2.7.4-java8      23d8c9a8ce60
bde2020/hive                       2.3.2-postgresql-metastore   87f5c9f4e2df
```

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



```
➜  ~ docker run --rm -it --memory="2g" --entrypoint /bin/sh zyclonite/sysbench
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin
/ # ls -ltr
total 56
drwxr-xr-x   12 root     root          4096 Nov 24 09:23 var
drwxrwxrwt    2 root     root          4096 Nov 24 09:23 tmp
drwxr-xr-x    2 root     root          4096 Nov 24 09:23 srv
drwxr-xr-x    2 root     root          4096 Nov 24 09:23 opt
drwxr-xr-x    2 root     root          4096 Nov 24 09:23 mnt
drwxr-xr-x    5 root     root          4096 Nov 24 09:23 media
drwxr-xr-x    2 root     root          4096 Nov 24 09:23 home
drwxr-xr-x    2 root     root          4096 Nov 24 09:23 sbin
drwxr-xr-x    2 root     root          4096 Nov 24 09:23 bin
drwxr-xr-x    1 root     root          4096 Dec 20 12:16 run
drwxr-xr-x    1 root     root          4096 Dec 20 12:16 usr
drwxr-xr-x    1 root     root          4096 Dec 20 12:16 lib
dr-xr-xr-x   13 root     root             0 Apr 18 23:58 sys
dr-xr-xr-x  176 root     root             0 Apr 18 23:58 proc
drwxr-xr-x    1 root     root          4096 Apr 18 23:58 etc
drwxr-xr-x    5 root     root           360 Apr 18 23:58 dev
drwx------    1 root     root          4096 Apr 18 23:58 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 - - all 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.

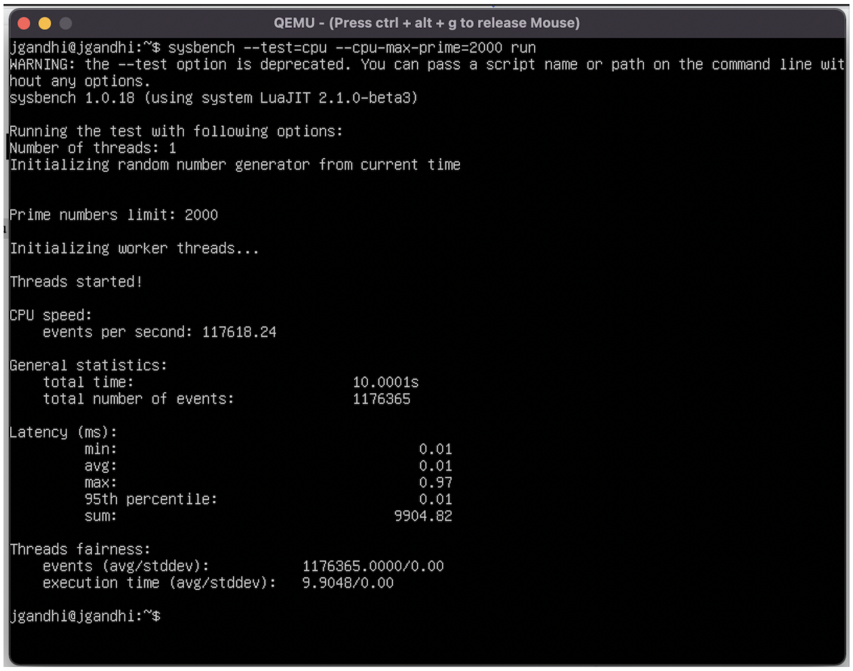
IV. Proof of experiment

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

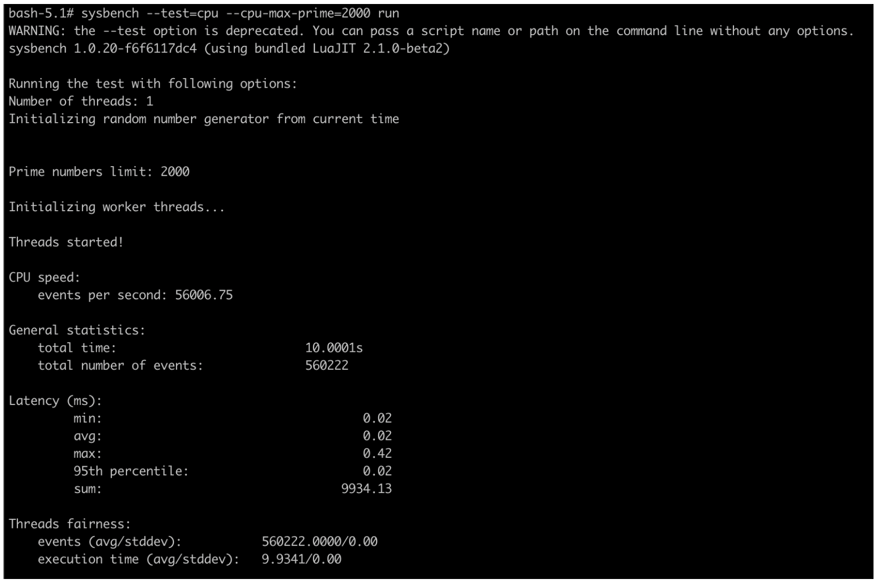
Test 1: sysbench --test=cpu --cpu-max-prime=2000 run

QEMU



```
jgandhi@jgandhi:~$ 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 wit
hout 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: 117618.24

General statistics:
    total time:                          10.0001s
    total number of events:              1176365

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    0.97
         95th percentile:                        0.01
         sum:                                 9904.82

Threads fairness:
    events (avg/stddev):           1176365.0000/0.00
    execution time (avg/stddev):   9.9048/0.00

jgandhi@jgandhi:~$
```

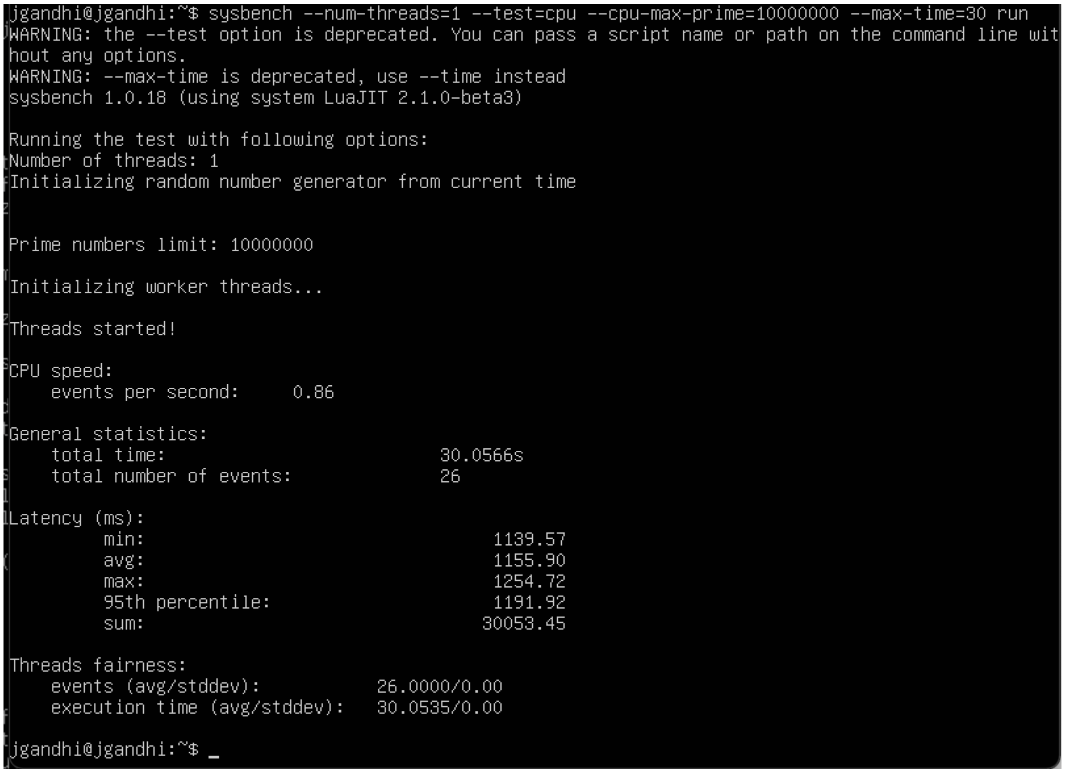
Docker



```
bash-5.1# 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: 56006.75

General statistics:
    total time:                          10.0001s
    total number of events:              560222

Latency (ms):
         min:                                    0.02
         avg:                                    0.02
         max:                                    0.42
         95th percentile:                        0.02
         sum:                                 9934.13

Threads fairness:
    events (avg/stddev):           560222.0000/0.00
    execution time (avg/stddev):   9.9341/0.00
```

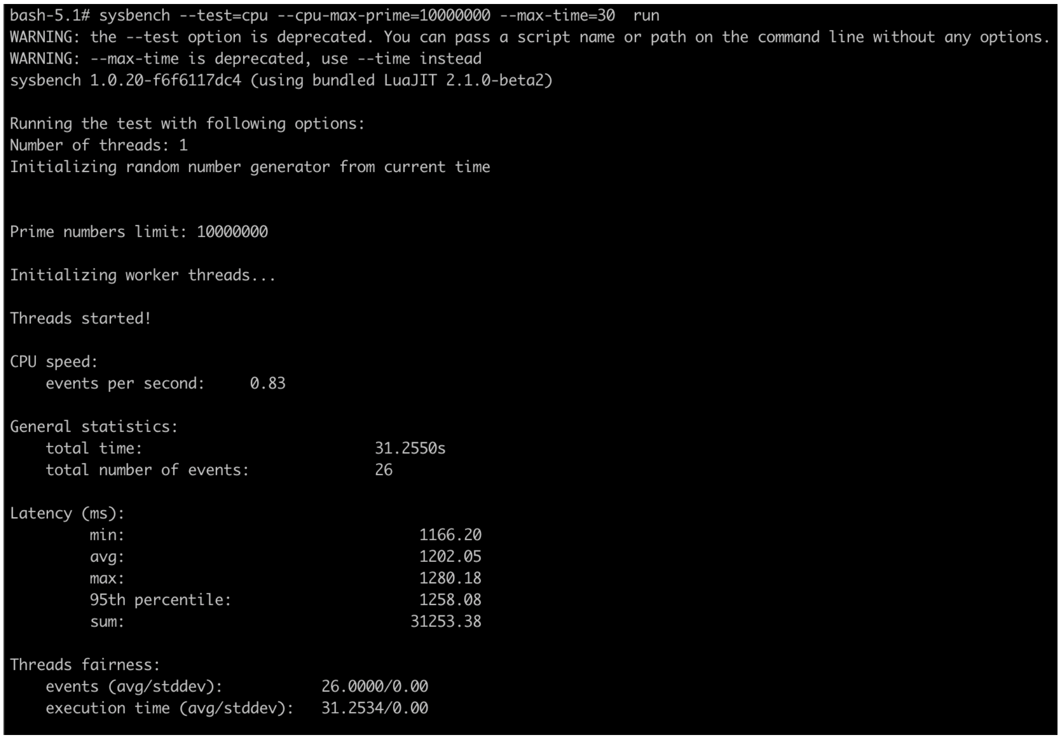
Test 2: sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run

QEMU



```
jgandhi@jgandhi:~$ sysbench --num-threads=1 --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 wit
hout 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:     0.86

General statistics:
    total time:                          30.0566s
    total number of events:              26

Latency (ms):
         min:                                 1139.57
         avg:                                 1155.90
         max:                                 1254.72
         95th percentile:                     1191.92
         sum:                                30053.45

Threads fairness:
    events (avg/stddev):           26.0000/0.00
    execution time (avg/stddev):   30.0535/0.00

jgandhi@jgandhi:~$ _
```

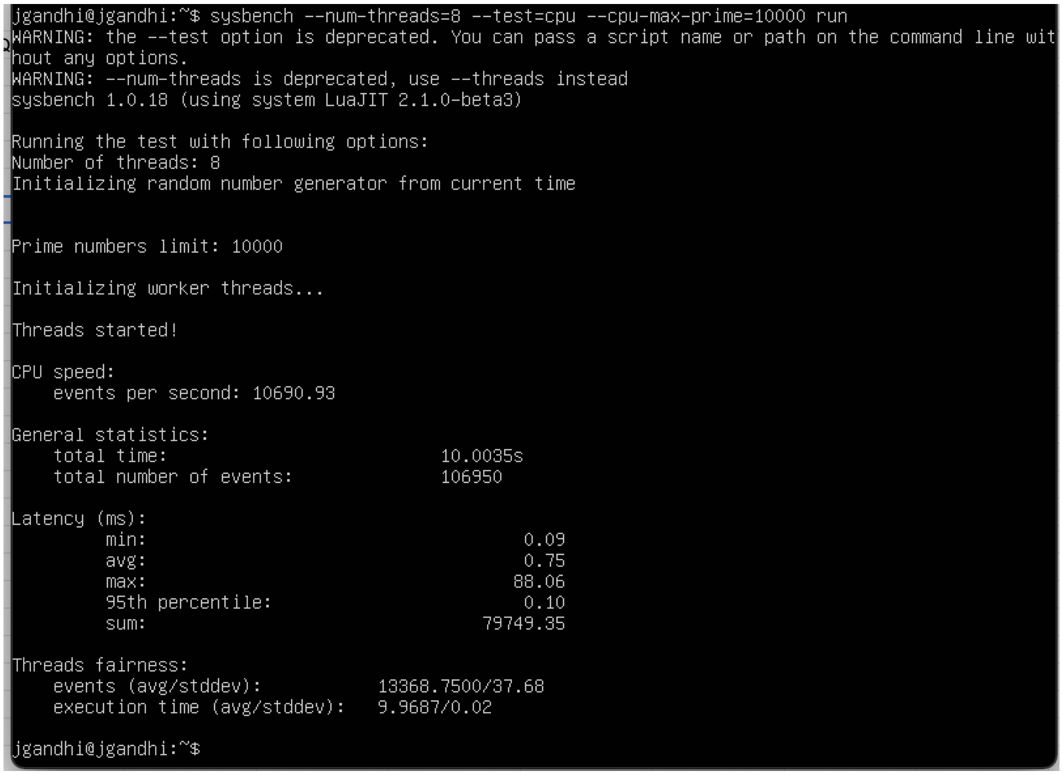
Docker



```
bash-5.1# 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 1.0.20-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:     0.83

General statistics:
    total time:                          31.2550s
    total number of events:              26

Latency (ms):
         min:                                 1166.20
         avg:                                 1202.05
         max:                                 1280.18
         95th percentile:                     1258.08
         sum:                                31253.38

Threads fairness:
    events (avg/stddev):           26.0000/0.00
    execution time (avg/stddev):   31.2534/0.00
```

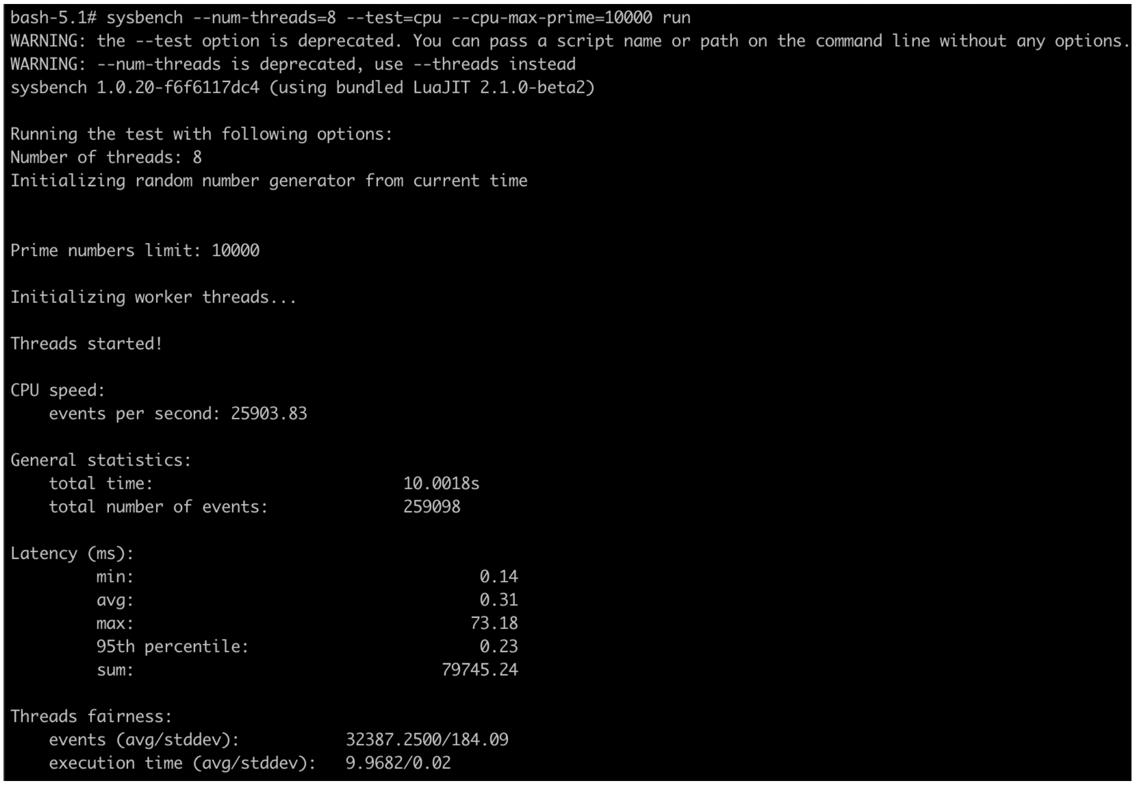
Test 3: sysbench --num-threads=8 --test=cpu --cpu-max-prime=10000 run

QEMU



```
jgandhi@jgandhi:~$ 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 wit
hout 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: 8
Initializing random number generator from current time


Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 10690.93

General statistics:
    total time:                          10.0035s
    total number of events:              106950

Latency (ms):
         min:                                    0.09
         avg:                                    0.75
         max:                                   88.06
         95th percentile:                        0.10
         sum:                                79749.35

Threads fairness:
    events (avg/stddev):           13368.7500/37.68
    execution time (avg/stddev):   9.9687/0.02

jgandhi@jgandhi:~$
```

Docker



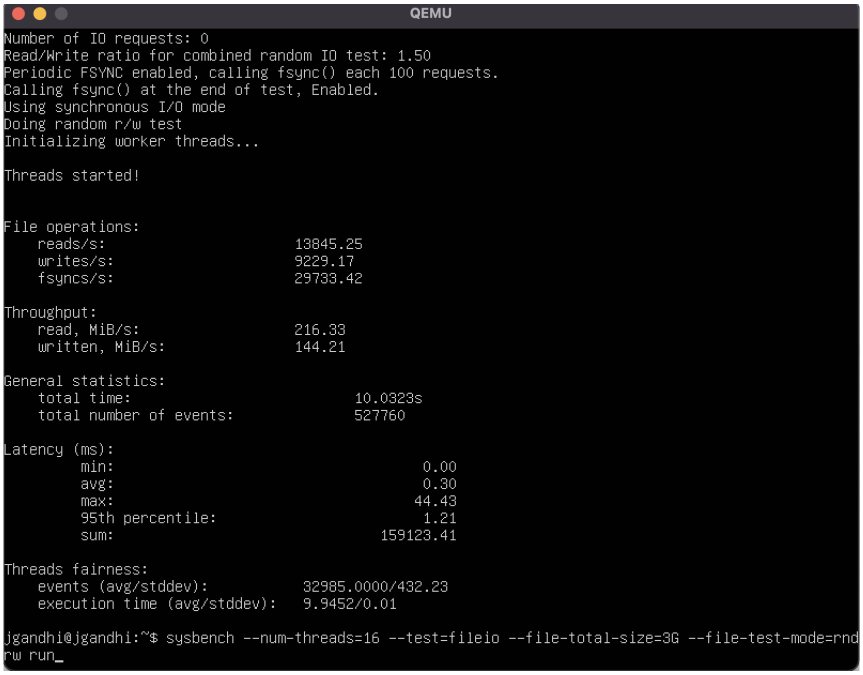
```
bash-5.1# sysbench --num-threads=8 --test=cpu --cpu-max-prime=10000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

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


Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 25903.83

General statistics:
    total time:                          10.0018s
    total number of events:              259098

Latency (ms):
         min:                                    0.14
         avg:                                    0.31
         max:                                   73.18
         95th percentile:                        0.23
         sum:                                79745.24

Threads fairness:
    events (avg/stddev):           32387.2500/184.09
    execution time (avg/stddev):   9.9682/0.02
```

File IO tests

Test 1:

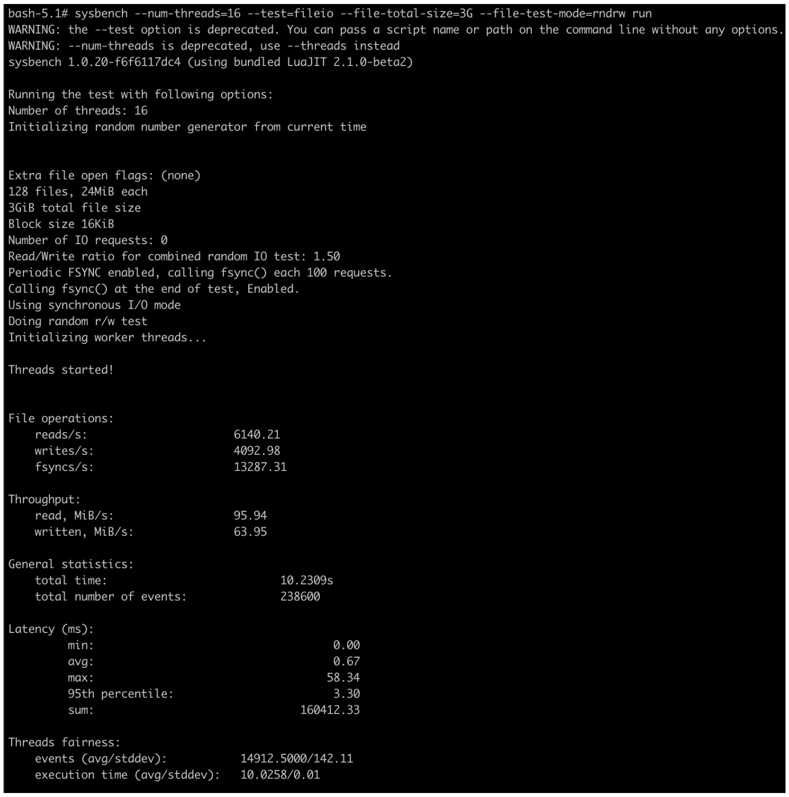
sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw run

QEMU



```
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:                      13845.25
    writes/s:                     9229.17
    fsyncs/s:                     29733.42

Throughput:
    read, MiB/s:                  216.33
    written, MiB/s:               144.21

General statistics:
    total time:                          10.0323s
    total number of events:              527760

Latency (ms):
         min:                                    0.00
         avg:                                    0.30
         max:                                   44.43
         95th percentile:                        1.21
         sum:                               159123.41

Threads fairness:
    events (avg/stddev):           32985.0000/432.23
    execution time (avg/stddev):   9.9452/0.01

jgandhi@jgandhi:~$ sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rnd
rw run_
```

Docker

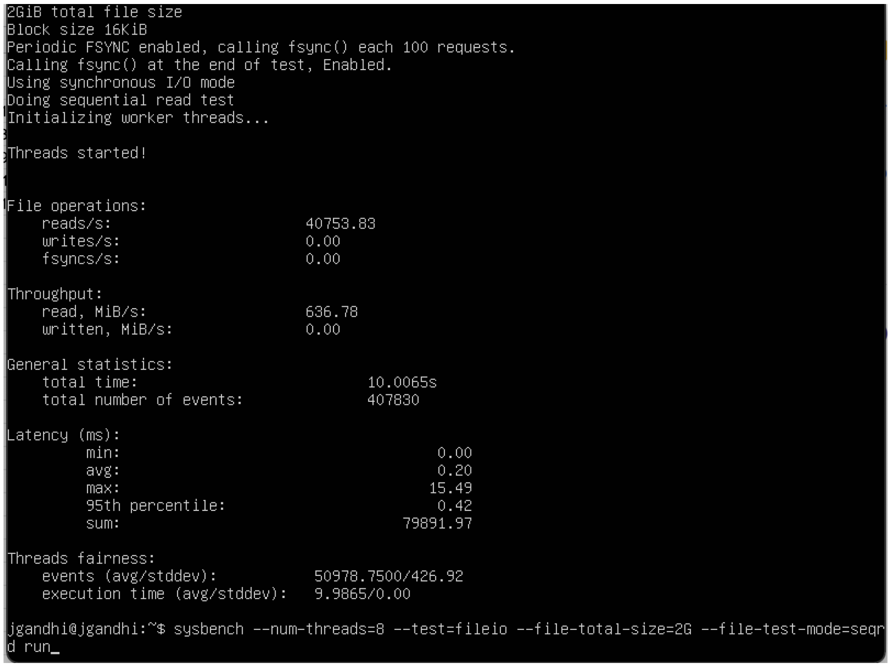


```
bash-5.1# sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (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:                      6140.21
    writes/s:                     4092.98
    fsyncs/s:                     13287.31

Throughput:
    read, MiB/s:                  95.94
    written, MiB/s:               63.95

General statistics:
    total time:                          10.2309s
    total number of events:              238600

Latency (ms):
         min:                                    0.00
         avg:                                    0.67
         max:                                   58.34
         95th percentile:                        3.30
         sum:                               160412.33

Threads fairness:
    events (avg/stddev):           14912.5000/142.11
    execution time (avg/stddev):   10.0258/0.01
```

Test 2:

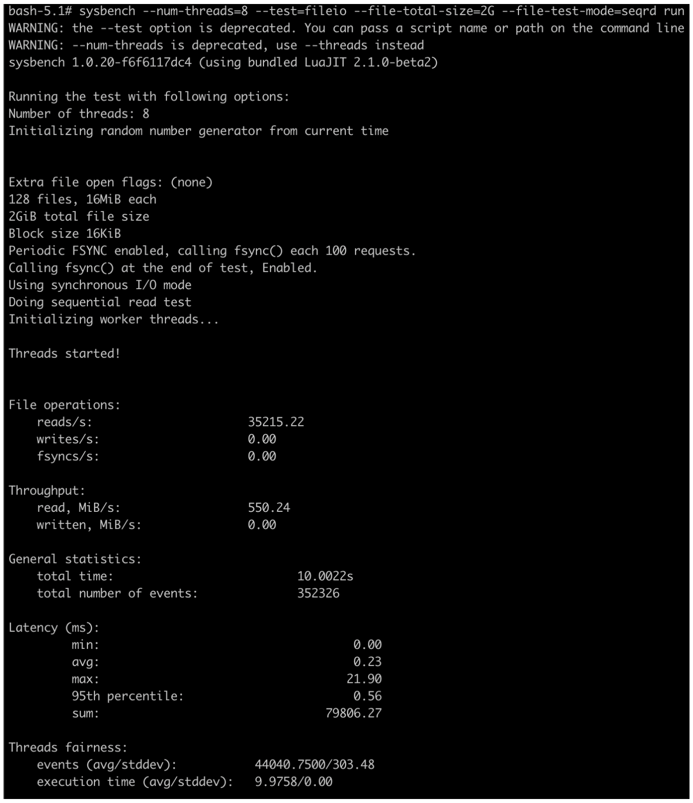
sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode=seqrd run

QEMU



```
2GiB 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:                      40753.83
    writes/s:                     0.00
    fsyncs/s:                     0.00

Throughput:
    read, MiB/s:                  636.78
    written, MiB/s:               0.00

General statistics:
    total time:                          10.0065s
    total number of events:              407830

Latency (ms):
         min:                                    0.00
         avg:                                    0.20
         max:                                   15.49
         95th percentile:                        0.42
         sum:                                79891.97

Threads fairness:
    events (avg/stddev):           50978.7500/426.92
    execution time (avg/stddev):   9.9865/0.00

jgandhi@jgandhi:~$ sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode=seqr
d run_
```

Docker

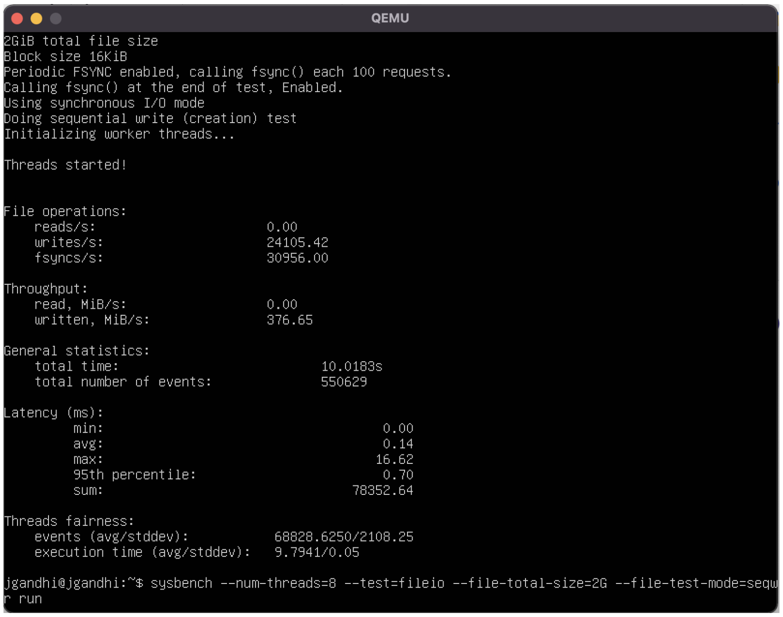


```
bash-5.1# sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode=seqrd run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

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


Extra file open flags: (none)
128 files, 16MiB each
2GiB 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:                      35215.22
    writes/s:                     0.00
    fsyncs/s:                     0.00

Throughput:
    read, MiB/s:                  550.24
    written, MiB/s:               0.00

General statistics:
    total time:                          10.0022s
    total number of events:              352326

Latency (ms):
         min:                                    0.00
         avg:                                    0.23
         max:                                   21.90
         95th percentile:                        0.56
         sum:                                79806.27

Threads fairness:
    events (avg/stddev):           44040.7500/303.48
    execution time (avg/stddev):   9.9758/0.00
```

Test 3:

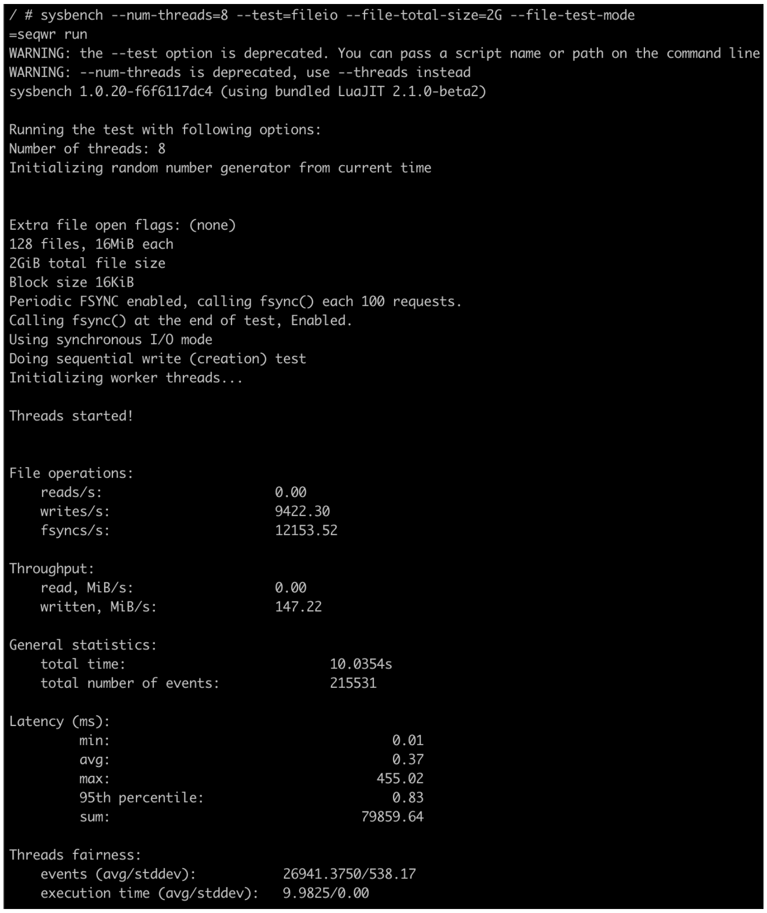
sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode=seqwr run

QEMU



```
2GiB 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:                     24105.42
    fsyncs/s:                     30956.00

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               376.65

General statistics:
    total time:                          10.0183s
    total number of events:              550629

Latency (ms):
         min:                                    0.00
         avg:                                    0.14
         max:                                   16.62
         95th percentile:                        0.70
         sum:                                78352.64

Threads fairness:
    events (avg/stddev):           68828.6250/2108.25
    execution time (avg/stddev):   9.7941/0.05

jgandhi@jgandhi:~$ sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode=seqw
r run
```

Docker



```
/ # sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode
=seqwr 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


Extra file open flags: (none)
128 files, 16MiB each
2GiB 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:                     9422.30
    fsyncs/s:                     12153.52

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               147.22

General statistics:
    total time:                          10.0354s
    total number of events:              215531

Latency (ms):
         min:                                    0.01
         avg:                                    0.37
         max:                                  455.02
         95th percentile:                        0.83
         sum:                                79859.64

Threads fairness:
    events (avg/stddev):           26941.3750/538.17
    execution time (avg/stddev):   9.9825/0.00
```

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	0.846	0.82
Min	0.84	0.81
Max	0.86	0.83
Std	0.01	0.01

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

	QEMU	Docker
Average	10636.61	27152.60
Min	10499.78	26922.66
Max	10736.77	27532.95
Std	95.69	237.49

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	48838.11	34193.33
Min	40755.23	20052.39
Max	52990.52	43217.14
Std	4785.09	8547.09

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

	QEMU	Docker
Average	41348.84	38150.88
Min	35451.91	12353.32
Max	48131.12	60266.11
Std	4634.94	21427.31

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

	QEMU	Docker
Average	52916.56	36614.33
Min	48110.52	29622.55
Max	55602.72	39886.12
Std	3496.41	4110.01

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 M1 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 that the events per seconds in QEMU.

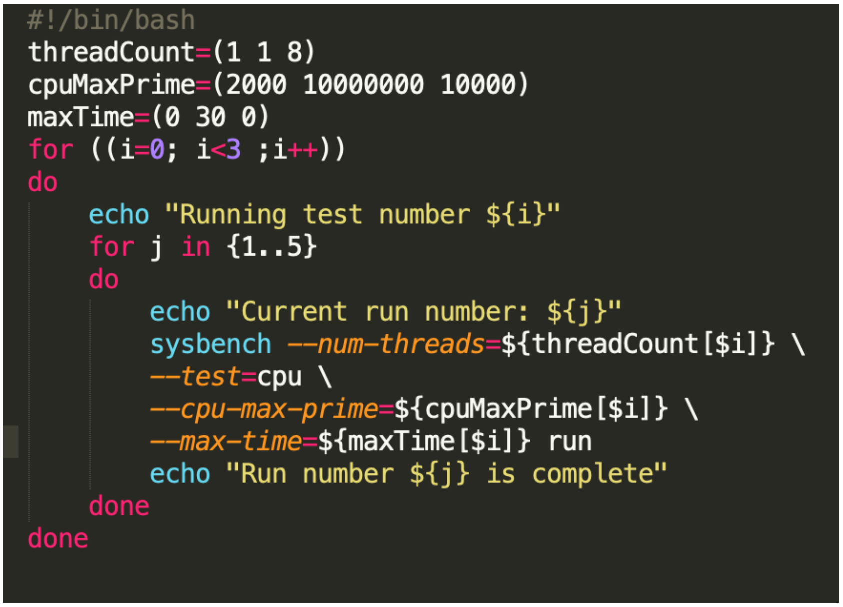
2. **Increasing the number of threads in the test helps to increase performance of the system -** For all the tests where the system had more more threads, the performance of the system is much better. However, just increasing the number of threads does not help. The performance also depends on how many 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.

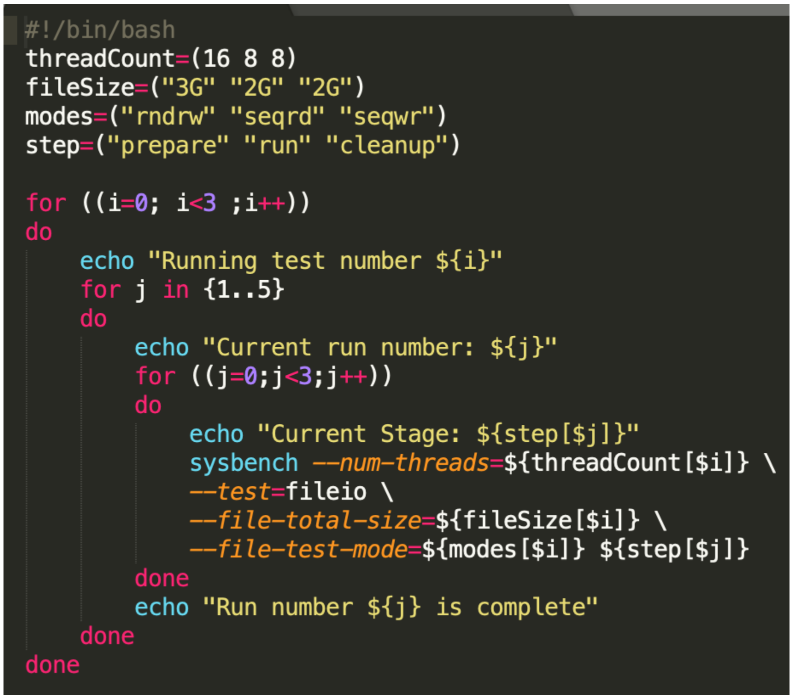
VII. Shell scripts for test execution

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[$i]} \
        --test=cpu \
        --cpu-max-prime=${cpuMaxPrime[$i]} \
        --max-time=${maxTime[$i]} run
        echo "Run number ${j} is complete"
    done
done
```

file-test.sh



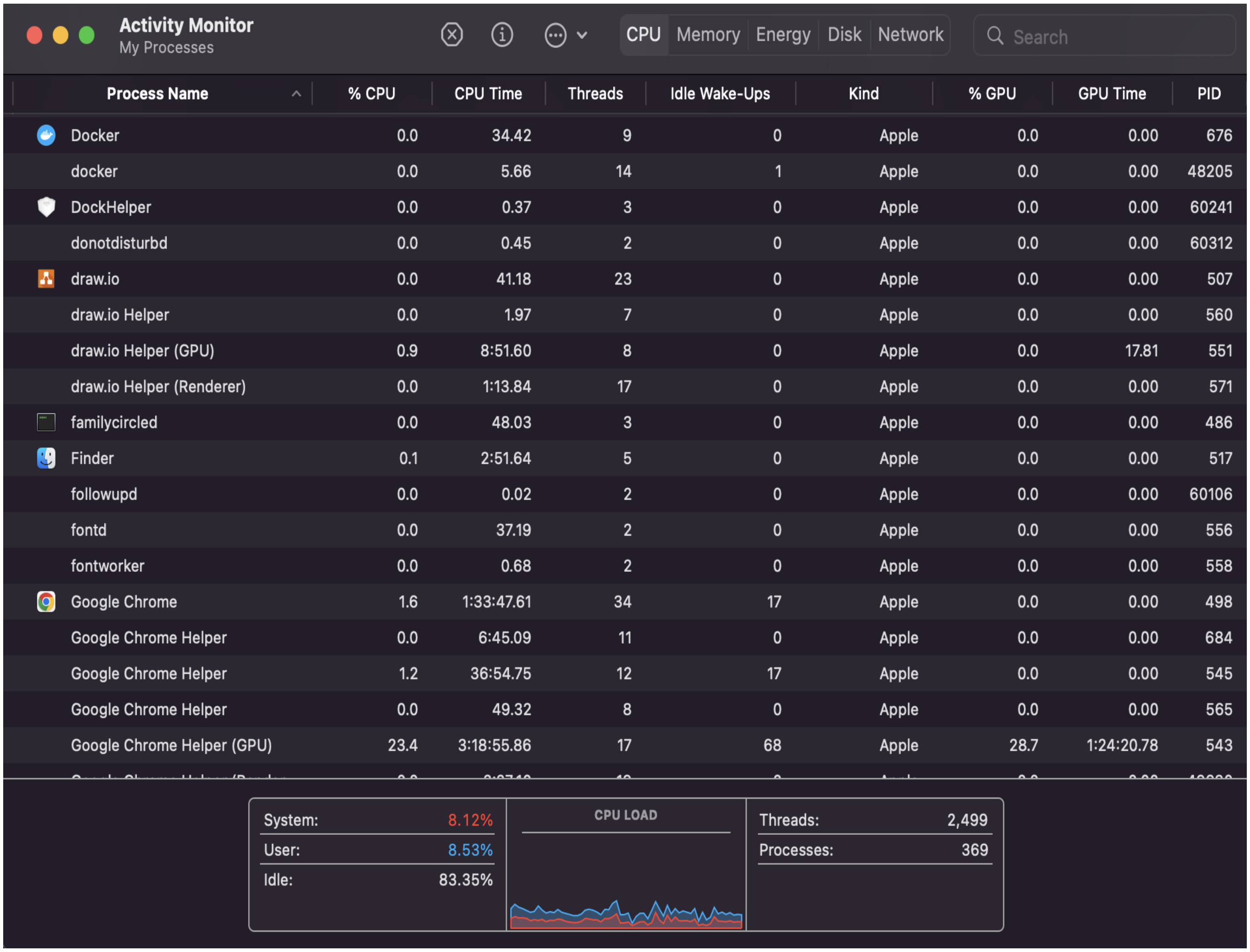
```
#!/bin/bash
threadCount=(16 8 8)
fileSize=("3G" "2G" "2G")
modes=("rndrw" "seqrd" "seqwr")
step=("prepare" "run" "cleanup")

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=${fileSize[$i]} \
            --file-test-mode=${modes[$i]} ${step[$j]}
        done
        echo "Run number ${j} is complete"
    done
done
```

Name: Janan Gandhi Student Id: W1607782

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:



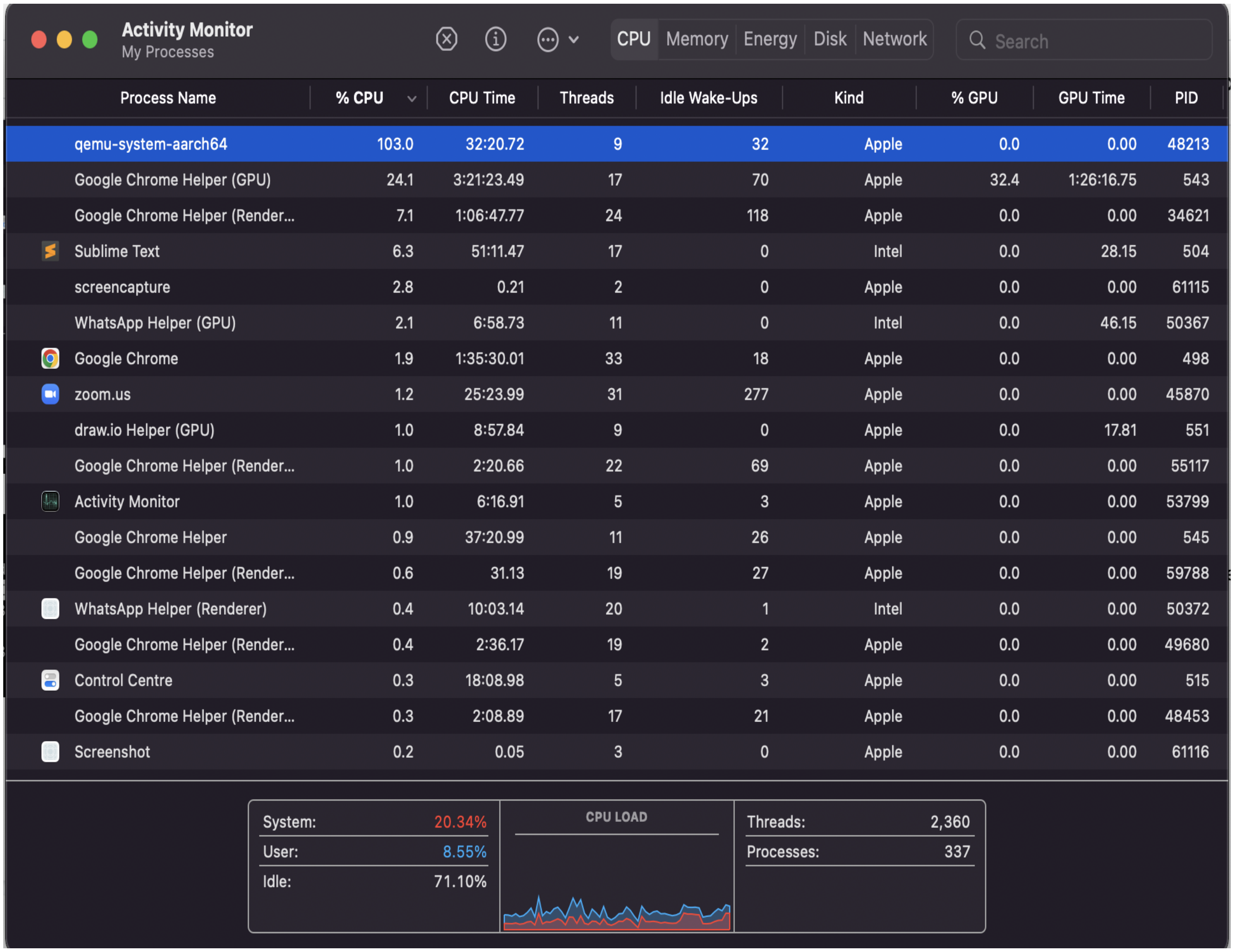
Process Name	% CPU	CPU Time	Threads	Idle Wake-Ups	Kind	% GPU	GPU Time	PID
Docker	0.0	34.42	9	0	Apple	0.0	0.00	676
docker	0.0	5.66	14	1	Apple	0.0	0.00	48205
DockHelper	0.0	0.37	3	0	Apple	0.0	0.00	60241
donotdisturbd	0.0	0.45	2	0	Apple	0.0	0.00	60312
draw.io	0.0	41.18	23	0	Apple	0.0	0.00	507
draw.io Helper	0.0	1.97	7	0	Apple	0.0	0.00	560
draw.io Helper (GPU)	0.9	8:51.60	8	0	Apple	0.0	17.81	551
draw.io Helper (Renderer)	0.0	1:13.84	17	0	Apple	0.0	0.00	571
familycircled	0.0	48.03	3	0	Apple	0.0	0.00	486
Finder	0.1	2:51.64	5	0	Apple	0.0	0.00	517
followupd	0.0	0.02	2	0	Apple	0.0	0.00	60106
fontd	0.0	37.19	2	0	Apple	0.0	0.00	556
fontworker	0.0	0.68	2	0	Apple	0.0	0.00	558
Google Chrome	1.6	1:33:47.61	34	17	Apple	0.0	0.00	498
Google Chrome Helper	0.0	6:45.09	11	0	Apple	0.0	0.00	684
Google Chrome Helper	1.2	36:54.75	12	17	Apple	0.0	0.00	545
Google Chrome Helper	0.0	49.32	8	0	Apple	0.0	0.00	565
Google Chrome Helper (GPU)	23.4	3:18:55.86	17	68	Apple	28.7	1:24:20.78	543

System:	8.12%
User:	8.53%
Idle:	83.35%

Threads:	2,499
Processes:	369

Name: Janan Gandhi Student Id: W1607782

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	103.0	32:20.72	9	32	Apple	0.0	0.00	48213
Google Chrome Helper (GPU)	24.1	3:21:23.49	17	70	Apple	32.4	1:26:16.75	543
Google Chrome Helper (Render...	7.1	1:06:47.77	24	118	Apple	0.0	0.00	34621
Sublime Text	6.3	51:11.47	17	0	Intel	0.0	28.15	504
screencapture	2.8	0.21	2	0	Apple	0.0	0.00	61115
WhatsApp Helper (GPU)	2.1	6:58.73	11	0	Intel	0.0	46.15	50367
Google Chrome	1.9	1:35:30.01	33	18	Apple	0.0	0.00	498
zoom.us	1.2	25:23.99	31	277	Apple	0.0	0.00	45870
draw.io Helper (GPU)	1.0	8:57.84	9	0	Apple	0.0	17.81	551
Google Chrome Helper (Render...	1.0	2:20.66	22	69	Apple	0.0	0.00	55117
Activity Monitor	1.0	6:16.91	5	3	Apple	0.0	0.00	53799
Google Chrome Helper	0.9	37:20.99	11	26	Apple	0.0	0.00	545
Google Chrome Helper (Render...	0.6	31.13	19	27	Apple	0.0	0.00	59788
WhatsApp Helper (Renderer)	0.4	10:03.14	20	1	Intel	0.0	0.00	50372
Google Chrome Helper (Render...	0.4	2:36.17	19	2	Apple	0.0	0.00	49680
Control Centre	0.3	18:08.98	5	3	Apple	0.0	0.00	515
Google Chrome Helper (Render...	0.3	2:08.89	17	21	Apple	0.0	0.00	48453
Screenshot	0.2	0.05	3	0	Apple	0.0	0.00	61116

System:	20.34%
User:	8.55%
Idle:	71.10%

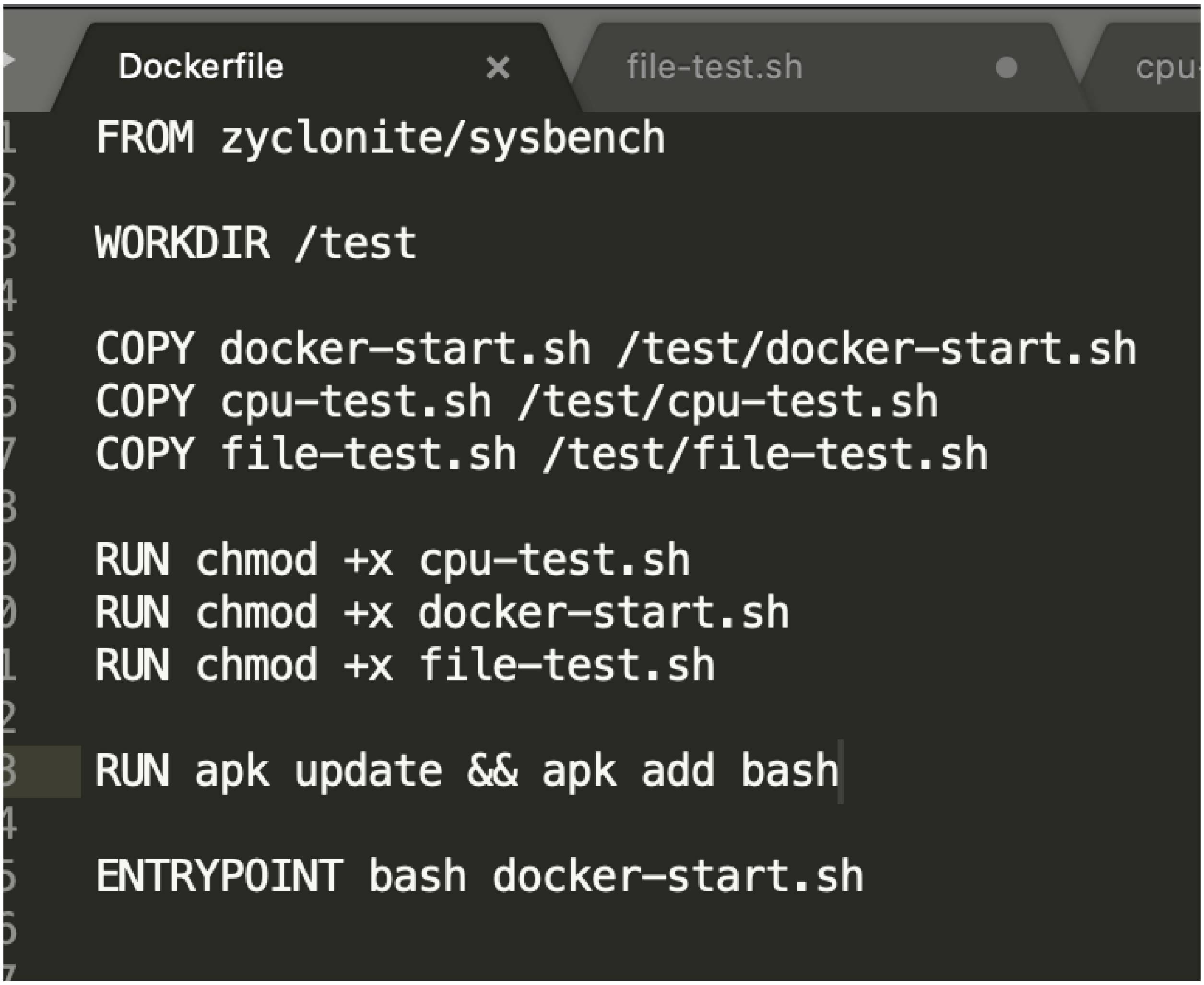
CPU LOAD

Threads:	2,360
Processes:	337

	Without sysbench	With sysbench
User level CPU usage%	8.53%	8.55%
System(kernel) level CPU usage%	8.12%	20.34%
Idle %	83.58%	71.1%

We can see from the above table that when we run sysbench, the kernel level CPU usage jumps from 8.12% to 20.34%. 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 103%.

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
```