COEN 241: HW - 1

QEMU Installation:

1. Download Ubuntu ISO (Download from here)

2. Install QEMU:
   > brew install qemu (stable version for Apple M1 chip: 6.2.0)

3. Create QEMU image:
   > qemu-img create ubuntu.img 20G -f qcow2

4. Install VM:
   > 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/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-20.04.5-live-server-arm64.iso \-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic

5. Run Image:
   > 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/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 \-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic

Experimental Setup:

System: 2021 Macbook Air; Chip: Apple M1, Memory: 8GB RAM
Testing 3 configurations for both QEMU and Docker:

- 2 cores with 2GB memory allocation
- 4 cores with 4GB memory allocation
- 6 cores with 6GB memory allocation

Docker Installation:

Rosetta is required for installation of Docker in order to ensure instruction set compatibility.

1. Run Rosetta:

   > softwareupdate -–install-rosetta

2. Install Docker Engine - Download the docker dmg image from here.

3. Check and test the installation:

   > docker run hello-world

Output:

```
benitarego@Benitas-MacBook-Air ~ % softwareupdate --install-rosetta
I have read and agree to the terms of the software license agreement. A list of
Apple SLAs may be found here: http://www.apple.com/legal/sla/
Type A and press return to agree: A
2023-01-30 17:47:16.292 softwareupdate[23379:879330] Package Authoring Error: 01
2-60306: Package reference com.apple.pkg.RosettaUpdateAuto is missing installKBy
tes attribute
Install of Rosetta 2 finished successfully
```

```
benitarego@Benitas-MacBook-Air ~ % docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Experimental Results:

Configuration 1: 2 cores with 2GB memory allocation

Tests:

1. Max-prime=2000, time=30 seconds
2. Max-prime=20000, time=30 seconds
3. Max-prime=200000, time=30 seconds

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

> sudo apt install sysbench

Sysbench command:

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

Case 1: Max-prime = 2000

For QEMU:

> sysbench cpu --cpu-max-prime=2000 --num-threads=1 --time=30 run

OR

> source ~/QEMU_2000.sh

For Docker:

> docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=2000 --time=30 run

OR

> source ~/DOCKER_2000.sh

QEMU Results:

```
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: 71182.82

General statistics:
    total time:                          30.0005s
    total number of events:              2135617

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    2.28
         95th percentile:                        0.01
         sum:                                29720.01

Threads fairness:
    events (avg/stddev):           2135617.0000/0.00
    execution time (avg/stddev):   29.7200/0.00
```

```
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: 71130.71

General statistics:
    total time:                          30.0004s
    total number of events:              2134058

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    3.44
         95th percentile:                        0.01
         sum:                                29717.67

Threads fairness:
    events (avg/stddev):           2134058.0000/0.00
    execution time (avg/stddev):   29.7177/0.00
```

```
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: 71465.38

General statistics:
    total time:                          30.0003s
    total number of events:              2144084

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    5.68
         95th percentile:                        0.01
         sum:                                29710.15

Threads fairness:
    events (avg/stddev):           2144084.0000/0.00
    execution time (avg/stddev):   29.7101/0.00
```

```
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: 71295.74

General statistics:
    total time:                          30.0002s
    total number of events:              2138967

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                    3.29
         95th percentile:                        0.01
         sum:                                29693.98

Threads fairness:
    events (avg/stddev):           2138967.0000/0.00
    execution time (avg/stddev):   29.6940/0.00
```

```
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: 70708.97

General statistics:
    total time:                          30.0010s
    total number of events:              2121494

Latency (ms):
         min:                                    0.01
         avg:                                    0.01
         max:                                   12.53
         95th percentile:                        0.01
         sum:                                29706.58

Threads fairness:
    events (avg/stddev):           2121494.0000/0.00
    execution time (avg/stddev):   29.7066/0.00
```

No. of Iterations	Events/sec
1	71182.82
2	71130.71
3	71465.38 - max value
4	71295.74
5	70708.97 - min value
Avg:	71156.72

Docker Results:

```
benitarego@Benitas-MacBook-Air ~ % docker run --rm zyclonite/sysbench --test=cpu
--cpu-max-prime=2000 --time=30 run

WARNING: the --test option is deprecated. You can pass a script name or path on t
he 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: 34105.65

General statistics:
    total time:                          30.0002s
    total number of events:              1023210

Latency (ms):
         min:                                    0.03
         avg:                                    0.03
         max:                                   11.76
         95th percentile:                        0.03
         sum:                                29837.75

Threads fairness:
    events (avg/stddev):           1023210.0000/0.00
    execution time (avg/stddev):   29.8377/0.00
```

```
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: 34182.79

General statistics:
    total time:                          30.0001s
    total number of events:              1025526

Latency (ms):
         min:                                    0.03
         avg:                                    0.03
         max:                                    1.75
         95th percentile:                        0.03
         sum:                                29850.79

Threads fairness:
    events (avg/stddev):           1025526.0000/0.00
    execution time (avg/stddev):   29.8508/0.00
```

```
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: 34227.73

General statistics:
    total time:                          30.0002s
    total number of events:              1026882

Latency (ms):
         min:                                    0.03
         avg:                                    0.03
         max:                                    0.94
         95th percentile:                        0.03
         sum:                                29847.86

Threads fairness:
    events (avg/stddev):           1026882.0000/0.00
    execution time (avg/stddev):   29.8479/0.00
```

```
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: 34201.29

General statistics:
    total time:                          30.0003s
    total number of events:              1026082

Latency (ms):
         min:                                    0.03
         avg:                                    0.03
         max:                                    2.54
         95th percentile:                        0.03
         sum:                                29803.42

Threads fairness:
    events (avg/stddev):           1026082.0000/0.00
    execution time (avg/stddev):   29.8034/0.00
```

```
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: 34194.07

General statistics:
    total time:                          30.0002s
    total number of events:              1025878

Latency (ms):
         min:                                    0.03
         avg:                                    0.03
         max:                                   14.60
         95th percentile:                        0.03
         sum:                                29812.32

Threads fairness:
    events (avg/stddev):           1025878.0000/0.00
    execution time (avg/stddev):   29.8123/0.00
```

No. of Iterations	Events/sec
1	34105.65 - min value
2	34182.79
3	34227.73 - max value
4	34201.29
5	34194.07
Avg:	34182.31

Result: QEMU is faster than Docker in CPU performance.

Case 2: Max-prime = 20000

For QEMU:

> sysbench cpu --cpu-max-prime=20000 --num-threads=1 --time=30 run

OR

> source ~/QEMU_20000.sh

For Docker:

> docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=30 run

OR

> source ~/DOCKER_20000.sh

QEMU Results:

```
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:  2607.18

General statistics:
    total time:                          30.0011s
    total number of events:              78224

Latency (ms):
         min:                                    0.37
         avg:                                    0.38
         max:                                   13.37
         95th percentile:                        0.40
         sum:                                29966.70

Threads fairness:
    events (avg/stddev):           78224.0000/0.00
    execution time (avg/stddev):   29.9667/0.00
```

```
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:  2617.16

General statistics:
    total time:                          30.0014s
    total number of events:              78525

Latency (ms):
         min:                                    0.37
         avg:                                    0.38
         max:                                    9.23
         95th percentile:                        0.39
         sum:                                29919.85

Threads fairness:
    events (avg/stddev):           78525.0000/0.00
    execution time (avg/stddev):   29.9199/0.00
```

```
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:  2604.12

General statistics:
    total time:                          30.0008s
    total number of events:              78131

Latency (ms):
         min:                                    0.37
         avg:                                    0.38
         max:                                   11.34
         95th percentile:                        0.41
         sum:                                29908.75

Threads fairness:
    events (avg/stddev):           78131.0000/0.00
    execution time (avg/stddev):   29.9087/0.00
```

```
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:  2608.95

General statistics:
    total time:                          30.0013s
    total number of events:              78278

Latency (ms):
         min:                                    0.37
         avg:                                    0.38
         max:                                   12.70
         95th percentile:                        0.41
         sum:                                29936.29

Threads fairness:
    events (avg/stddev):           78278.0000/0.00
    execution time (avg/stddev):   29.9363/0.00
```

```
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:  2612.34

General statistics:
    total time:                          30.0007s
    total number of events:              78378

Latency (ms):
         min:                                    0.37
         avg:                                    0.38
         max:                                    5.46
         95th percentile:                        0.40
         sum:                                29901.89

Threads fairness:
    events (avg/stddev):           78378.0000/0.00
    execution time (avg/stddev):   29.9019/0.00
```

No. of Iterations	Events/sec
1	2617.16 - max value
2	2607.18
3	2604.12 - min value
4	2608.95
5	2612.34
Avg:	2609.95

Docker Results:

```
benitarego@Benitas-MacBook-Air ~ % 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 t
he 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:  1986.20

General statistics:
    total time:                          30.0006s
    total number of events:              59590

Latency (ms):
         min:                                    0.49
         avg:                                    0.50
         max:                                    1.96
         95th percentile:                        0.51
         sum:                                29926.56

Threads fairness:
    events (avg/stddev):           59590.0000/0.00
    execution time (avg/stddev):   29.9266/0.00
```

```
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:  1966.42

General statistics:
    total time:                          30.0004s
    total number of events:              58999

Latency (ms):
         min:                                    0.49
         avg:                                    0.51
         max:                                   14.85
         95th percentile:                        0.54
         sum:                                29915.89

Threads fairness:
    events (avg/stddev):           58999.0000/0.00
    execution time (avg/stddev):   29.9159/0.00
```

```
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:  1963.54

General statistics:
    total time:                          30.0007s
    total number of events:              58913

Latency (ms):
         min:                                    0.49
         avg:                                    0.51
         max:                                   10.40
         95th percentile:                        0.54
         sum:                                29893.68

Threads fairness:
    events (avg/stddev):           58913.0000/0.00
    execution time (avg/stddev):   29.8937/0.00
```

```
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!

WARNING: the --test option is deprecated. You can pass a script
he command line without any options.
CPU speed:
    events per second:  1968.02

General statistics:
    total time:                          30.0006s
    total number of events:              59047

Latency (ms):
         min:                                    0.49
         avg:                                    0.51
         max:                                    5.49
         95th percentile:                        0.54
         sum:                                29892.35

Threads fairness:
    events (avg/stddev):           59047.0000/0.00
    execution time (avg/stddev):   29.8923/0.00
```

```
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:  1982.38

General statistics:
    total time:                          30.0006s
    total number of events:              59475

Latency (ms):
         min:                                    0.49
         avg:                                    0.50
         max:                                    1.39
         95th percentile:                        0.53
         sum:                                29936.80

Threads fairness:
    events (avg/stddev):           59475.0000/0.00
    execution time (avg/stddev):   29.9368/0.00
```

No. of Iterations	Events/sec
1	1986.20 - max value
2	1966.42
3	1963.54 - min value
4	1968.02
5	1982.38
Avg:	1973.31

Result:
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.

Case 3: Max-prime = 200000

For QEMU:

> sysbench cpu --cpu-max-prime=200000 --num-threads=1 --time=30 run

OR

> source ~/QEMU_200000.sh

For Docker:

> docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=200000 --time=30 run

OR

> source ~/DOCKER_200000.sh

QEMU Results:

```
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:   115.02

General statistics:
    total time:                          30.0024s
    total number of events:              3451

Latency (ms):
         min:                                    8.49
         avg:                                    8.69
         max:                                   64.80
         95th percentile:                        8.74
         sum:                                29972.45

Threads fairness:
    events (avg/stddev):           3451.0000/0.00
    execution time (avg/stddev):   29.9724/0.00
```

```
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:   115.27

General statistics:
    total time:                          30.0073s
    total number of events:              3459

Latency (ms):
         min:                                    8.49
         avg:                                    8.67
         max:                                   42.81
         95th percentile:                        8.74
         sum:                                29982.60

Threads fairness:
    events (avg/stddev):           3459.0000/0.00
    execution time (avg/stddev):   29.9826/0.00
```

```
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:   115.16

General statistics:
    total time:                          30.0082s
    total number of events:              3456

Latency (ms):
         min:                                    8.50
         avg:                                    8.68
         max:                                   43.27
         95th percentile:                        8.74
         sum:                                29984.52

Threads fairness:
    events (avg/stddev):           3456.0000/0.00
    execution time (avg/stddev):   29.9845/0.00
```

```
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:   115.90

General statistics:
    total time:                          30.0081s
    total number of events:              3478

Latency (ms):
         min:                                    8.49
         avg:                                    8.62
         max:                                   22.35
         95th percentile:                        8.74
         sum:                                29984.33

Threads fairness:
    events (avg/stddev):           3478.0000/0.00
    execution time (avg/stddev):   29.9843/0.00
```

```
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:   115.31

General statistics:
    total time:                          30.0037s
    total number of events:              3460

Latency (ms):
         min:                                    8.51
         avg:                                    8.66
         max:                                   29.42
         95th percentile:                        8.74
         sum:                                29970.65

Threads fairness:
    events (avg/stddev):           3460.0000/0.00
    execution time (avg/stddev):   29.9706/0.00
```

No. of Iterations	Events/sec
1	115.02 - min value
2	115.27
3	115.16
4	115.90 - max value
5	115.31
Avg:	115.33

Docker Results:

```
[benitarego@Benitas-MacBook-Air ~ % 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 t
he 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:   104.83

General statistics:
    total time:                          30.0078s
    total number of events:              3146

Latency (ms):
         min:                                    9.39
         avg:                                    9.53
         max:                                   39.84
         95th percentile:                        9.73
         sum:                                29985.25

Threads fairness:
    events (avg/stddev):           3146.0000/0.00
    execution time (avg/stddev):   29.9852/0.00
```

```
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!

WARNING: the --test option is deprecated. You can pass a script
he command line without any options.
CPU speed:
    events per second:   105.00

General statistics:
    total time:                          30.0076s
    total number of events:              3151

Latency (ms):
         min:                                    9.40
         avg:                                    9.52
         max:                                   31.79
         95th percentile:                        9.56
         sum:                                29983.65

Threads fairness:
    events (avg/stddev):           3151.0000/0.00
    execution time (avg/stddev):   29.9837/0.00
```

```
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!

WARNING: the --test option is deprecated. You can pass a script
he command line without any options.
CPU speed:
    events per second:   104.94

General statistics:
    total time:                          30.0063s
    total number of events:              3149

Latency (ms):
         min:                                    9.38
         avg:                                    9.52
         max:                                   29.57
         95th percentile:                        9.73
         sum:                                29981.90

Threads fairness:
    events (avg/stddev):           3149.0000/0.00
    execution time (avg/stddev):   29.9819/0.00
```

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


Prime numbers limit: 200000

Initializing worker threads...

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

CPU speed:
    events per second:   104.54

General statistics:
    total time:                          30.0060s
    total number of events:              3137

Latency (ms):
         min:                                    9.39
         avg:                                    9.56
         max:                                   32.46
         95th percentile:                        9.73
         sum:                                29984.69

Threads fairness:
    events (avg/stddev):           3137.0000/0.00
    execution time (avg/stddev):   29.9847/0.00
```

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


Prime numbers limit: 200000

Initializing worker threads...

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

CPU speed:
    events per second:   104.50

General statistics:
    total time:                          30.0071s
    total number of events:              3136

Latency (ms):
         min:                                    9.39
         avg:                                    9.56
         max:                                   50.34
         95th percentile:                        9.91
         sum:                                29983.37

Threads fairness:
    events (avg/stddev):           3136.0000/0.00
    execution time (avg/stddev):   29.9834/0.00
```

No. of Iterations	Events/sec
1	104.83
2	105.00 - max value
3	104.94
4	104.54
5	104.50 - min value
Avg:	104.76

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

Conclusion:
QEMU VM is faster than Docker in terms of CPU performance on the M1 chip. Based on our experimental results, we can conclude that as we increase the cpu-max-prime argument value, our number of events per second continue to decrease, i.e, greater the value of cpu-max-prime argument, lower the number of events per second for sysbench CPU testing.

File I/O testing QEMU vs Docker

1. For QEMU:
    a. Sequential Rewrite

    > sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=seqrewr prepare

    > sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=seqrewr run

    > sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=seqrewr cleanup

```
ubuntu@ubuntu:~$ sysbench --num-threads=16 --test=fileio --file-total-size=3G --
time=30 --file-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 1.0.18 (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:                     24741.96
    fsyncs/s:                     31737.76

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               386.59

General statistics:
    total time:                          30.0211s
    total number of events:              1693588

Latency (ms):
         min:                                    0.00
         avg:                                    0.28
         max:                                   49.73
         95th percentile:                        1.23
         sum:                               476077.84

Threads fairness:
    events (avg/stddev):           105849.2500/1220.44
    execution time (avg/stddev):   29.7549/0.01
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 0.00 writes/sec = 24741.96 fsyncs/sec = 31737.76 events/sec = 56413.26
2	reads/sec = 0.00 writes/sec = 25091.54 fsyncs/sec = 32183.12 events/sec = 57208.3

3	reads/sec = 0.00 writes/sec = 25099.53 fsyncs/sec = 32192.86 events/sec = 57225.82
4	reads/sec = 0.00 writes/sec = 25774.89 fsyncs/sec = 33056.79 events/sec = 58764.91
5	reads/sec = 0.00 writes/sec = 25716.55 fsyncs/sec = 32983.65 events/sec = 58633.30

b. Combined Random Rewrite

> sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw prepare

> sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw run

> sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw cleanup

```
ubuntu@ubuntu:~$ sysbench --num-threads=16 --test=fileio --file-total-size=3G --
time=30 --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 1.0.18 (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:                      12792.27
    writes/s:                     8527.94
    fsyncs/s:                     27358.04

Throughput:
    read, MiB/s:                  199.88
    written, MiB/s:               133.25

General statistics:
    total time:                          30.0223s
    total number of events:              1459429

Latency (ms):
         min:                                    0.00
         avg:                                    0.33
         max:                                   20.68
         95th percentile:                        1.08
         sum:                               478341.26

Threads fairness:
    events (avg/stddev):           91214.3125/464.61
    execution time (avg/stddev):   29.8963/0.01
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 12792.27 writes/sec = 8527.94 fsyncs/sec = 27358.04 events/sec = 48611.5
2	reads/sec = 10980.66 writes/sec = 7320.22 fsyncs/sec = 23490.43 events/sec = 41725.19
3	reads/sec = 11286.93 writes/sec = 7524.45 fsyncs/sec = 24143.98 events/sec = 42888.69
4	reads/sec = 11229.25 writes/sec = 7486.00 fsyncs/sec = 24023.52 events/sec = 42671.85
5	reads/sec = 10204.11 writes/sec = 6802.46 fsyncs/sec = 21835.43 events/sec = 38775.19

2. For Docker:
    a. Sequential Rewrite

        > docker run --rm zyclonite/sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=seqrewr prepare

        > docker run --rm zyclonite/sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=seqrewr run

        > docker run --rm zyclonite/sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=seqrewr cleanup

```
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:                     126972.76
    fsyncs/s:                     162590.76

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               1983.95

General statistics:
    total time:                          30.0030s
    total number of events:              8685809

Latency (ms):
         min:                                    0.00
         avg:                                    0.05
         max:                                  170.08
         95th percentile:                        0.00
         sum:                               462917.92

Threads fairness:
    events (avg/stddev):           542863.0625/8225.51
    execution time (avg/stddev):   28.9324/0.02
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 0 writes/sec = 39389.37 fsyncs/sec = 50483.02 events/sec = 289498.02
2	reads/sec = 0 writes/sec = 121807.80 fsyncs/sec = 155980.56 events/sec = 277726.78
3	reads/sec = 0 writes/sec = 121085.32 fsyncs/sec = 155055.63 events/sec = 276075.38
4	reads/sec = 0 writes/sec = 122745.44 fsyncs/sec = 157181.65

	events/sec = 279866.05
5	reads/sec = 0 writes/sec = 125738.83 fsyncs/sec = 161010.54 events/sec = 286683.63

b. Combined Random Rewrite

> docker run --rm zyclonite/sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw prepare

> docker run --rm zyclonite/sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw run

> docker run --rm zyclonite/sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw cleanup

```
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:                      31880.49
    writes/s:                     21253.34
    fsyncs/s:                     68076.70

Throughput:
    read, MiB/s:                  498.13
    written, MiB/s:               332.08

General statistics:
    total time:                          30.0026s
    total number of events:              3634615

Latency (ms):
         min:                                    0.00
         avg:                                    0.13
         max:                                    7.91
         95th percentile:                        0.00
         sum:                               478006.11

Threads fairness:
    events (avg/stddev):           227163.4375/1282.68
    execution time (avg/stddev):   29.8754/0.00
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 31800.49 writes/sec = 21253.34 fsyncs/sec = 68076.70 events/sec = 121143.33
2	reads/sec = 31654.73 writes/sec = 21102.80 fsyncs/sec = 67594.23 events/sec = 120284.84
3	reads/sec = 33374.73 writes/sec = 22249.51 fsyncs/sec = 71265.85 events/sec = 126823.17
4	reads/sec = 32671.59 writes/sec = 21780.76 fsyncs/sec = 69765.30 events/sec = 124150.73
5	reads/sec = 32702.74 writes/sec = 21801.47 fsyncs/sec = 69831.70 events/sec = 124268.86

Conclusion: Sequential rewrite operations are faster on QEMU than Docker. While for combined read write, docker is faster than QEMU.

Configuration 2: 4 cores with 4GB memory allocation

Tests:

4. Max-prime=2000, time=30 seconds
5. Max-prime=20000, time=30 seconds
6. Max-prime=200000, time=30 seconds

Case 1: Max-prime = 2000

QEMU Results:

```
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
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	113926.26 - Max value
2	113201.06
3	113018.63 - Min value
4	114854.19
5	113129.79
Avg:	113625.98

Docker Results:

After running the commands in 5 iterations, we get the following results:

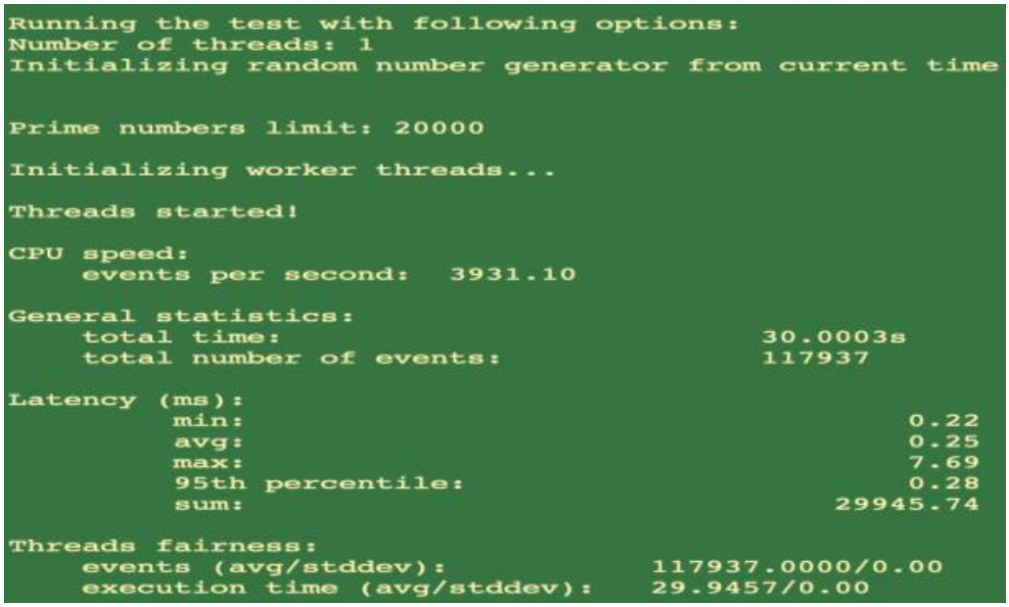
No. of Iterations	Events/sec
1	52698.93
2	52132.09
3	51893.90 - min value
4	56403.96 - max value
5	56134.25
Avg:	53852.62

Result:

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

Case 2: Max-prime = 20000

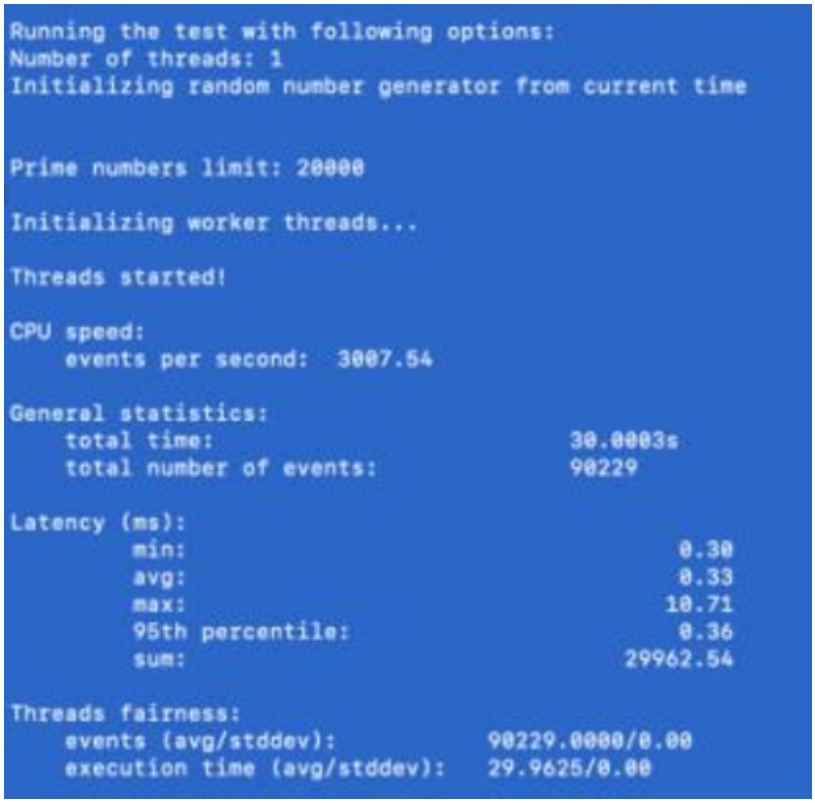
QEMU Results:



After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	3931.10
2	3933.90
3	4281.01 - max value
4	4051.55 - min value
5	3972.59
Avg:	4034.03

Docker Results:



After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	3007.54
2	3247.12 - max value
3	3181.35
4	3164.53
5	2917.72 - min value
Avg:	3103.65

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

Case 3: Max-prime = 200000

QEMU Results:

```
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
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	188.73 - max value
2	188.33
3	185.49 - min value
4	189.86
5	189.83
Avg:	188.44

Docker Results:

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	169.27 - min value
2	172.51
3	172.34
4	171.67
5	173.24 - max value
Avg:	171.80

Result:
As we raise the value of the cpu-max-prime parameter, we observe that the number of events continues to decline. For the MacBook M1 Chip, QEMU VM is still quicker than Docker desktop.

Conclusion:

When it comes to CPU performance on an M1 processor, QEMU VM outperforms Docker. According to our experimental findings, the number of events per second for sysbench CPU testing decreases as the cpu-max-prime argument value increases. In other words, the higher the value of the cpu-max-prime argument, the less events per second are produced.

File I/O testing QEMU vs Docker

For QEMU:

Sequential Rewrite

```
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:                     38097.14
    fsyncs/s:                     48828.63

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               595.27

General statistics:
    total time:                          30.0516s
    total number of events:              2610256

Latency (ms):
         min:                                    0.00
         avg:                                    0.18
         max:                                   35.97
         95th percentile:                        0.58
         sum:                               478937.83

Threads fairness:
    events (avg/stddev):           163141.0000/887.86
    execution time (avg/stddev):   29.9336/0.00
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 0.00 writes/sec = 38097.14 fsyncs/sec = 48828.63 events/sec = 87008.56
2	reads/sec = 0.00

	writes/sec = 39530.62 fsyncs/sec = 50664.21 events/sec = 90199.3
3	reads/sec = 0.00 writes/sec = 35886.20 fsyncs/sec = 33329.60 events/sec = 81888.26
4	reads/sec = 0.00 writes/sec = 33329.60 fsyncs/sec = 42727.80 events/sec = 76088.93
5	reads/sec = 0.00 writes/sec = 33704.58 fsyncs/sec = 43206.91 events/sec = 76908.83

Combined Random Rewrite

```
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:                      14847.51
    writes/s:                     9898.23
    fsyncs/s:                     31739.38

Throughput:
    read, MiB/s:                  231.99
    written, MiB/s:               154.66

General statistics:
    total time:                          30.0248s
    total number of events:              1693939

Latency (ms):
         min:                                    0.00
         avg:                                    0.28
         max:                                   37.23
         95th percentile:                        0.86
         sum:                               479034.62

Threads fairness:
    events (avg/stddev):          105871.1875/755.07
    execution time (avg/stddev):  29.9397/0.00
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 14847.51 writes/sec = 9898.23 fsyncs/sec = 31739.38 events/sec = 56464.43
2	reads/sec = 13910.0 writes/sec = 9273.0 fsyncs/sec = 29740.0 events/sec = 52915.76
3	reads/sec = 14165.54 writes/sec = 9443.58 fsyncs/sec = 30284.24 events/sec = 53880.36
4	reads/sec = 12349.31 writes/sec = 8232.87 fsyncs/sec = 26409.53 events/sec = 46994.56
5	reads/sec = 12001.06 writes/sec = 8000.63 fsyncs/sec = 25669.69 events/sec = 45638.56

For Docker:

Sequential Rewrite

```
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
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 0.00 writes/sec = 21687.04 fsyncs/sec = 27826.67 events/sec = 49552.66
2	reads/sec = 0.00 writes/sec = 22690.64 fsyncs/sec = 29019.49 events/sec = 51806.5
3	reads/sec = 0.00 writes/sec = 25688.43 fsyncs/sec = 32946.24 events/sec = 58659.23
4	reads/sec = 0.00 writes/sec = 22382.72

	fsyncs/sec = 28716.33 events/sec = 51146.33
5	reads/sec = 0.00 writes/sec = 21880.69 fsyncs/sec = 28071.27 events/sec = 49996.26

Combined Random Rewrite

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 13850.71 writes/sec = 9320.24 fsyncs/sec = 28826.54 events/sec = 53032.66
2	reads/sec = 14155.49 writes/sec = 8754.94 fsyncs/sec = 27129.15 events/sec = 50045.5

3	reads/sec = 14251.83 writes/sec = 9545.02 fsyncs/sec = 30457.31 events/sec = 54375.26
4	reads/sec = 14324.66 writes/sec = 9231.83 fsyncs/sec = 31432.93 events/sec = 55987.33
5	reads/sec = 14231.90 writes/sec = 9658.14 fsyncs/sec = 30324.53 events/sec = 55432.1

Conclusion:
On QEMU, sequential rewrite operations are quicker than Docker. Docker is quicker than QEMU for combined read and write operations.

Configuration 3: 6 cores with 6GB memory allocation

Tests:

7. Max-prime=2000, time=30 seconds
8. Max-prime=20000, time=30 seconds
9. Max-prime=200000, time=30 seconds

Case 1: Max-prime = 2000

QEMU Results:

```
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
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	113197.63 - min Value
2	113318.53
3	114254.59 - max value
4	113529.89
5	114251.46
Avg:	113529.89

Docker Results:

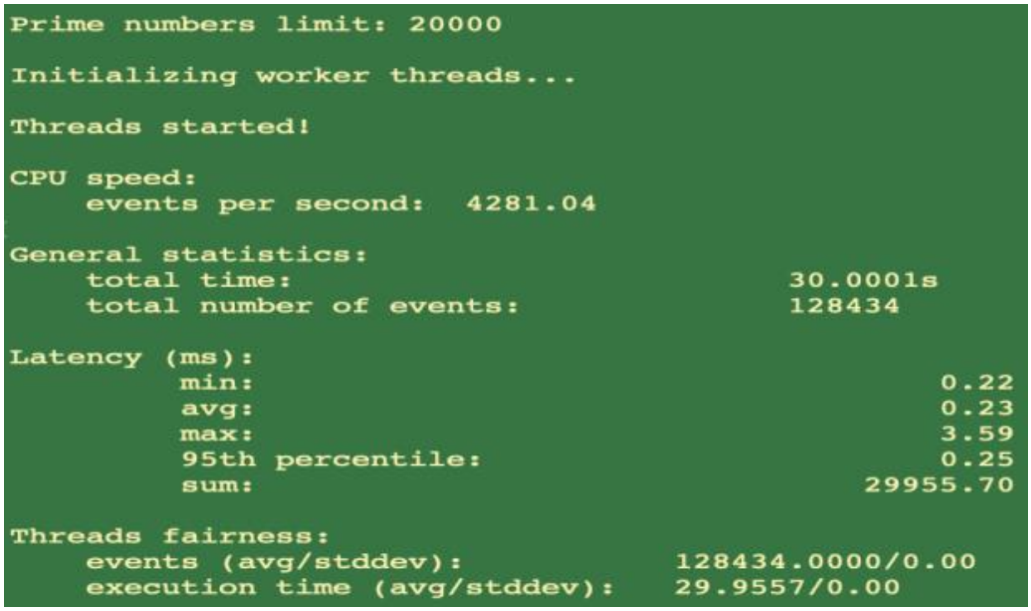
After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	54026.70
2	53403.88
3	56479.38 - max value
4	52534.09 - min value
5	52843.95
Avg:	53857.6

Result:
We observe that as we raise the value of the cpu-max-prime parameter, the number of occurrences continues to decline. For the MacBook M1 Chip, QEMU virtual machine is still quicker than Docker desktop.

Case 2: Max-prime = 20000

QEMU Results:



After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	4281.04
2	3942.69 - min value
3	3945.40
4	4284.01 - max value
5	4054.45
Avg:	4101.51

Docker Results:

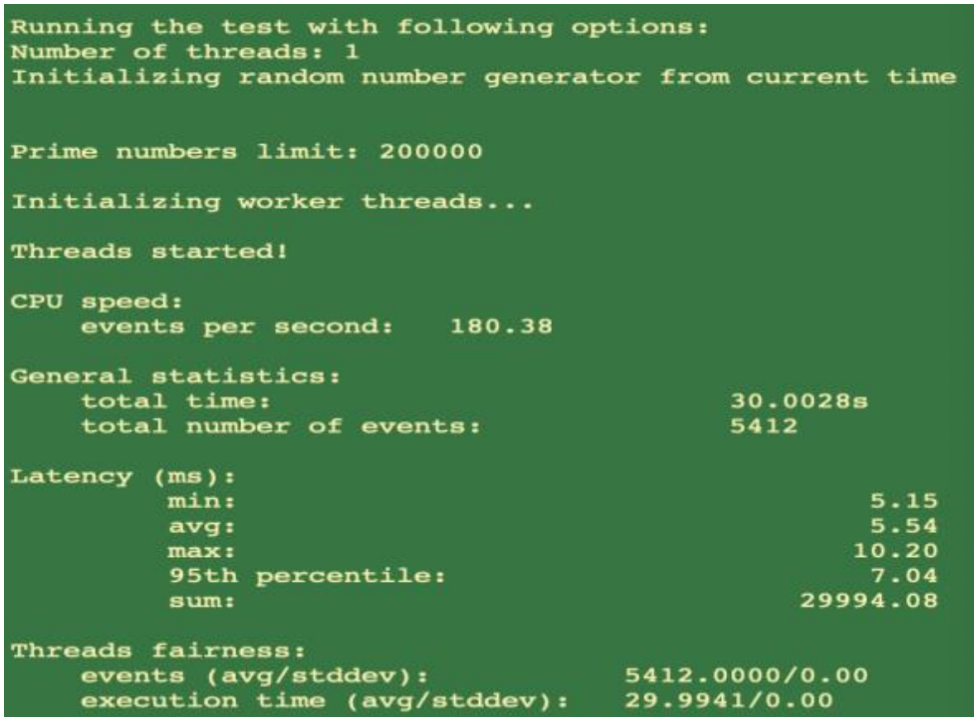
After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	3129.44
2	3264.53 - max value
3	3024.72 - min value
4	3141.35
5	3147.42
Avg:	3141.49

Result:
As we keep raising the cpu-max-prime parameter value, we see that the number of events continues to drop. In terms of speed for the MacBook M1 Chip, QEMU VM continues to be quicker than Docker desktop.

Case 3: Max-prime = 200000

QEMU Results:



After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	180.38 - min value
2	182.33
3	181.83
4	184.49 - max value
5	180.86
Avg:	181.97

Docker Results:

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Events/sec
1	171.39
2	174.24 - max value
3	172.67
4	172.47
5	171.34 - min value
Avg:	172.42

Result:
As we keep raising the cpu-max-prime parameter value, we see that the number of events continues to drop. In terms of speed for the MacBook M1 Chip, QEMU VM continues to be quicker than Docker desktop.

Conclusion:

When it comes to CPU performance on an M1 processor, QEMU VM outperforms Docker. According to our experimental findings, the number of events per second for sysbench CPU testing decreases as the cpu-max-prime argument value increases. In other words, the higher the value of the cpu-max-prime argument, the less events per second are produced.

File I/O testing QEMU vs Docker

For QEMU:

Sequential Rewrite

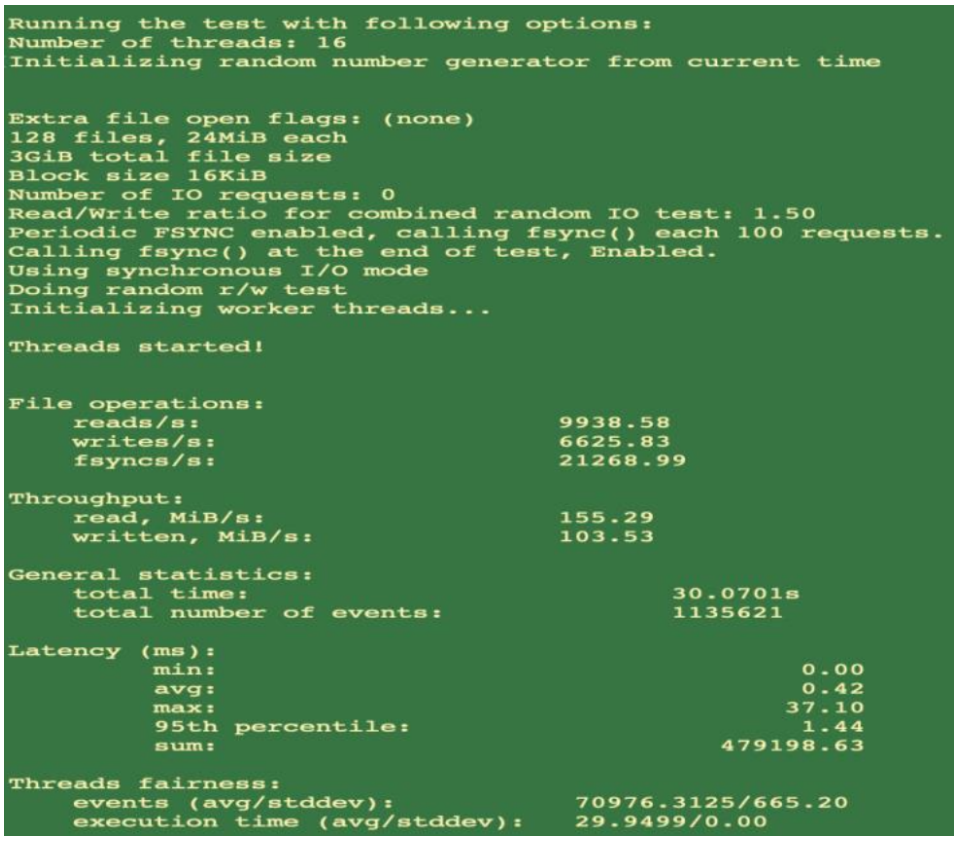
```
Running the test with following options:
Number of threads: 16
Initializing random number generator from current time


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
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 0.0 writes/sec = 25473.58 fsyncs/sec = 32673.72 events/sec = 58185
2	reads/sec = 0.0 writes/sec = 27687.54 fsyncs/sec = 33764.65 events/sec = 59124.32

3	reads/sec = 0.0 writes/sec = 24534.32 fsyncs/sec = 31986.87 events/sec = 56126.76
4	reads/sec = 0.0 writes/sec = 25765.32 fsyncs/sec = 32675.83 events/sec = 58234.21
5	reads/sec = 0.0 writes/sec = 28675.77 fsyncs/sec = 35687.32 events/sec = 59543.65

Combined Random Rewrite



After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 9938.58 writes/sec = 6625.83 fsyncs/sec = 21268.98 events/sec = 37854.03

2	reads/sec = 9876.73 writes/sec = 6543.21 fsyncs/sec = 21124.43 events/sec = 35541.28
3	reads/sec = 9763.23 writes/sec = 6312.63 fsyncs/sec = 24569.69 events/sec = 34532.56
4	reads/sec = 9845.43 writes/sec = 6541.32 fsyncs/sec = 20267.32 events/sec = 35643.76
5	reads/sec = 9765.53 writes/sec = 7021.32 fsyncs/sec = 30832.11 events/sec = 39878.32

For Docker:

Sequential Rewrite

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 0.0 writes/sec = 13778.77 fsyncs/sec = 17701.47 events/sec = 31506.16
2	reads/sec = 0.0 writes/sec = 12456.32 fsyncs/sec = 16323.87 events/sec = 29876.36
3	reads/sec = 0.0 writes/sec = 13543.64 fsyncs/sec = 17634.49 events/sec = 30234.5
4	reads/sec = 0.0 writes/sec = 13534.11 fsyncs/sec = 17431.89 events/sec = 31232.21
5	reads/sec = 0.0 writes/sec = 14328.87 fsyncs/sec = 18785.62 events/sec = 32324.32

Combined Random Rewrite

```
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
```

After running the commands in 5 iterations, we get the following results:

No. of Iterations	Output
1	reads/sec = 9501.28 writes/sec = 6334.24 fsyncs/sec = 20336.56 events/sec = 36197.8
2	reads/sec = 9674.98 writes/sec = 6984.73 fsyncs/sec = 20989.89 events/sec = 37019.21
3	reads/sec = 9594.32 writes/sec = 6434.43 fsyncs/sec = 20214.43 events/sec = 35788.33

4	reads/sec = 9287.90 writes/sec = 6021.14 fsyncs/sec = 18673.53 events/sec = 32764.12
5	reads/sec = 9323.32 writes/sec = 6213.41 fsyncs/sec = 20223.12 events/sec = 36097.87

Conclusion:
Sequential rewrite operations are faster on QEMU than Docker. While for combined read write, docker is faster than QEMU.

Performance Analysis:

1. **QEMU**
    a. **2 cores with 2GB memory allocation**
        i. Sequential Rewrite
        read, MiB/s = 0
        written, MiB/s = 708.32
        ii. Combined Random Read
        read, MiB/s = 211.42
        written, MiB/s = 142.36

    b. **4 cores with 4GB memory allocation**
        i. Sequential Rewrite
        read, MiB/s = 0
        written, MiB/s = 599.31
        ii. Combined Random Read
        read, MiB/s = 237.59
        written, MiB/s = 158.33

    c. **6 cores with 6GB memory allocation**
        i. Sequential Rewrite
        read, MiB/s = 0
        written, MiB/s = 402.14
        ii. Combined Random Read
        read, MiB/s = 163.34
        written, MiB/s = 107.02

2. **Docker**
    a. **2 cores with 2GB memory allocation**
        i. Sequential Rewrite
        read, MiB/s = 0

written, MiB/s = 560.21

ii. Combined Random Read
read, MiB/s = 243.22
written, MiB/s = 168.56

b. 4 cores with 4GB memory allocation

i. Sequential Rewrite
read, MiB/s = 0
written, MiB/s = 332.26

ii. Combined Random Read
read, MiB/s = 236.33
written, MiB/s = 163.59

c. 6 cores with 6GB memory allocation

i. Sequential Rewrite
read, MiB/s = 0
written, MiB/s = 208.21

ii. Combined Random Read
read, MiB/s = 145.35
written, MiB/s =95.74

Details:
Name: Benita Rego
Student ID: W1628656
GitHub Username: benitarego
Repository Name: COEN-241-CC
Folder: HW1
Link: https://github.com/benitarego/COEN-241-CC/
Commit ID: 4c1a7ad555761295fdb84928728f1cc5df7a740b