

# HW 1 COEN 241

Nolita Rego W1628657

## QEMU Installation and making QEMU disk image:

1. Download ISO

Ubuntu guest VM is needed which requires server ISO image. We can use the following link:  
ARM (Apple Silicon): Ubuntu 20.04 Server for ARM

2. Use the Following commands for installing QEMU on Macbook Air M1 Chip

```
brew install qemu
```

3. To create QEMU image, use the following command:

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

4. Install VM by running the following:

```
qemu-system-aarch64 \-accel hvf \-cpu cortex-a57 \-M virt,highmem=off \-m 2G \-smp 2 \-drive  
file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,form  
at=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
```

Follow the instructions given while installing VM to get it installed successfully.

5. To run the image, use the following:

```
qemu-system-aarch64 \-accel hvf \-cpu cortex-a57 \-M virt,highmem=off \-m 2G \-smp 2 \-drive  
file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,form  
at=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 Configuration:

Macbook Air M1 Chip (2021), 8GB Memory

Testing 3 configurations:

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

## Docker Installation:

1. Install Rosetta 2 for docker.  
`softwareupdate-install-rosetta`
2. Install docker engine, download docker dmg image [here](#).
3. Check if docker is installed correctly by running the command:  
`docker run hello-world`

```

nolitarego@Nolitas-MacBook-Air ~ % softwareupdate --install-rosetta
usage: softwareupdate <cmd> [<args> ...]

** Manage Updates:
  -l | --list           List all appropriate update labels (options: --no-scan, --product-types)
  -d | --download        Download Only
  -i | --install         Install
    <label> ...       specific updates
    -a | --all           All appropriate updates
    -R | --restart        Automatically restart (or shut down) if required to complete installation.
    -r | --recommended   Only recommended updates
    --os-only            Only OS updates
    --safari-only        Only Safari updates
    --stdinpass          Password to authenticate as an owner. Apple Silicon only.
    --user                Local username to authenticate as an owner. Apple Silicon only.
  --list-full-installers List the available macOS Installers
  --fetch-full-installer Install the latest recommended macOS Installer
  --full-installer-version The version of macOS to install. Ex: --full-installer-version 10.15
  --install-rosetta     Install Rosetta 2
  --background          Trigger a background scan and update operation

** Other Tools:
  --dump-state          Log the internal state of the SU daemon to /var/log/install.log
  --evaluate-products   Evaluate a list of product keys specified by the --products option
  --history              Show the install history. By default, only displays updates installed by softwareupdate.

** Options:
  --no-scan              Do not scan when listing or installing updates (use available updates previously scanned)
  --product-types <type> Limit a scan to a particular product type only - ignoring all others
    Ex: --product-types macOS || --product-types macOS,Safari
  --products             A comma-separated (no spaces) list of product keys to operate on.
  --force                Force an operation to complete. Use with --background to trigger a background scan regardless of "Automatically check" pref
  --agree-to-license     Agree to the software license agreement without user interaction.

  --verbose              Enable verbose output
  --help                 Print this help

nolitarego@Nolitas-MacBook-Air ~ % docker run hello-world
|
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/
nolitarego@Nolitas-MacBook-Air ~ %

```

## Experiments: QEMU and Docker Results

### **Configuration 1: 2GB RAM and 2 cores**

Tests:

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

Install sysbench before running the below command:

Brew install sysbench

Sysbench command:

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

### QEMU Results:

- Test 1:

```

WARNING: the --test option is deprecated. You can pass
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: 117629.79

General statistics:
total time: 30.0033s
total number of events: 3531502

Latency (ms):
min: 0.01
avg: 0.01
max: 5.08
95th percentile: 0.01
sum: 29710.69

Threads fairness:
events (avg/stddev): 3531502.0000/0.00
execution time (avg/stddev): 29.7107/0.00

```

Iteration 1

```

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: 115037.16

General statistics:
total time: 30.0113s
total number of events: 3458529

Latency (ms):
min: 0.01
avg: 0.01
max: 132.41
95th percentile: 0.01
sum: 29714.81

Threads fairness:
events (avg/stddev): 3458529.0000/0.00
execution time (avg/stddev): 29.7148/0.00

```

Iteration 2

```

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: 115160.19

General statistics:
total time: 30.0204s
total number of events: 3459362

Latency (ms):
min: 0.01
avg: 0.01
max: 16.17
95th percentile: 0.01
sum: 29501.63

Threads fairness:
events (avg/stddev): 3459362.0000/0.00
execution time (avg/stddev): 29.5016/0.00

```

Iteration 3

```

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: 110559.19

General statistics:
total time: 30.0076s
total number of events: 3318943

Latency (ms):
min: 0.01
avg: 0.01
max: 36.13
95th percentile: 0.01
sum: 29494.88

Threads fairness:
events (avg/stddev): 3318943.0000/0.00
execution time (avg/stddev): 29.4949/0.00

```

Iteration 4

```

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: 112912.98

General statistics:
total time: 30.0164s
total number of events: 3390114

Latency (ms):
min: 0.01
avg: 0.01
max: 24.70
95th percentile: 0.01
sum: 29660.42

Threads fairness:
events (avg/stddev): 3390114.0000/0.00
execution time (avg/stddev): 29.6604/0.00

```

### Iteration 5

Iteration no.	Events per second
1	117629.79 - max
2	115037.16
3	115160.19
4	110559.19 - min
5	112912.98
Average	114259.86

- Test 2:

```

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: 4301.90

General statistics:
total time: 30.0070s
total number of events: 129154

Latency (ms):
min: 0.23
avg: 0.23
max: 7.74
95th percentile: 0.24
sum: 29968.78

Threads fairness:
events (avg/stddev): 129154.0000/0.00
execution time (avg/stddev): 29.9688/0.00

```

Iteration 1

```

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: 4319.10

General statistics:
total time: 30.0002s
total number of events: 129581

Latency (ms):
min: 0.23
avg: 0.23
max: 2.00
95th percentile: 0.24
sum: 29949.83

Threads fairness:
events (avg/stddev): 129581.0000/0.00
execution time (avg/stddev): 29.9498/0.00

```

Iteration 2

```

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: 4326.48

General statistics:
total time: 30.0005s
total number of events: 129821

Latency (ms):
min: 0.23
avg: 0.23
max: 9.91
95th percentile: 0.24
sum: 29973.12

Threads fairness:
events (avg/stddev): 129821.0000/0.00
execution time (avg/stddev): 29.9731/0.00

```

Iteration 3

```

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: 4331.81

General statistics:
total time: 30.0013s
total number of events: 129987

Latency (ms):
min: 0.23
avg: 0.23
max: 0.84
95th percentile: 0.24
sum: 29970.90

Threads fairness:
events (avg/stddev): 129987.0000/0.00
execution time (avg/stddev): 29.9709/0.00

```

Iteration 4

```

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: 4322.40

General statistics:
total time: 30.0004s
total number of events: 129683

Latency (ms):
min: 0.23
avg: 0.23
max: 1.10
95th percentile: 0.24
sum: 29968.28

Threads fairness:
events (avg/stddev): 129683.0000/0.00
execution time (avg/stddev): 29.9683/0.00

```

### Iteration 5

Iteration no.	Events per second
1	4301.90 - min
2	4319.10
3	4326.8
4	4331.81 - max
5	4322.40
Average	4320.33

- Test 3:

```

WARNING: the --test option is deprecated. You can pass a
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 200000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 189.83

General statistics:
total time: 30.0041s
total number of events: 5698

Latency (ms):
min: 5.18
avg: 5.26
max: 7.48
95th percentile: 5.37
sum: 29957.30

Threads fairness:
events (avg/stddev): 5698.0000/0.00
execution time (avg/stddev): 29.9573/0.00

```

Iteration 1

```

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: 190.20

General statistics:
total time: 30.0087s
total number of events: 5710

Latency (ms):
min: 5.19
avg: 5.25
max: 7.94
95th percentile: 5.28
sum: 29968.41

Threads fairness:
events (avg/stddev): 5710.0000/0.00
execution time (avg/stddev): 29.9684/0.00

```

Iteration 2

```

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: 190.30

General statistics:
total time: 30.0040s
total number of events: 5712

Latency (ms):
min: 5.18
avg: 5.25
max: 8.63
95th percentile: 5.37
sum: 29973.66

Threads fairness:
events (avg/stddev): 5712.0000/0.00
execution time (avg/stddev): 29.9737/0.00

```

Iteration 3

```

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: 190.59

General statistics:
total time: 30.0005s
total number of events: 5718

Latency (ms):
min: 5.18
avg: 5.24
max: 16.02
95th percentile: 5.28
sum: 29972.36

Threads fairness:
events (avg/stddev): 5718.0000/0.00
execution time (avg/stddev): 29.9724/0.00

```

Iteration 4

```

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: 190.00

General statistics:
  total time:          30.0056s
  total number of events: 5703

Latency (ms):
  min:                5.19
  avg:                5.26
  max:                9.22
  95th percentile:    5.37
  sum:               29974.24

Threads fairness:
  events (avg/stddev): 5703.0000/0.00
  execution time (avg/stddev): 29.9742/0.00

```

### Iteration 5

<b>Iteration no.</b>	<b>Events per second</b>
1	189.90 - min
2	190.20
3	190.30
4	190.59 - max
5	190.00
Average	190.20

### Docker Results:

- Test 1:

```

WARNING: the --test option is deprecated. You can pass
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: 55705.48

General statistics:
total time: 30.0061s
total number of events: 1671610

Latency (ms):
min: 0.02
avg: 0.02
max: 6.59
95th percentile: 0.02
sum: 29847.15

Threads fairness:
events (avg/stddev): 1671610.0000/0.00
execution time (avg/stddev): 29.8471/0.00

```

Iteration 1

```

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: 53339.24

General statistics:
total time: 30.0035s
total number of events: 1600418

Latency (ms):
min: 0.02
avg: 0.02
max: 647.49
95th percentile: 0.02
sum: 29799.64

Threads fairness:
events (avg/stddev): 1600418.0000/0.00
execution time (avg/stddev): 29.7996/0.00

sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

```

Iteration 2

```

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!

WARNING: the --test option is deprecated. You can pass
CPU speed:
events per second: 53770.33

General statistics:
total time: 30.0019s
total number of events: 1613486

Latency (ms):
min: 0.02
avg: 0.02
max: 75.10
95th percentile: 0.02
sum: 29750.40

Threads fairness:
events (avg/stddev): 1613486.0000/0.00
execution time (avg/stddev): 29.7504/0.00

```

Iteration 3

```

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: 54454.38

General statistics:
total time: 30.0011s
total number of events: 1633769

Latency (ms):
min: 0.02
avg: 0.02
max: 19.96
95th percentile: 0.02
sum: 29839.23

Threads fairness:
events (avg/stddev): 1633769.0000/0.00
execution time (avg/stddev): 29.8392/0.00

sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

```

Iteration 4

```

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!

WARNING: the --test option is deprecated. You can pass
CPU speed:
    events per second: 55029.96

General statistics:
    total time:          30.0011s
    total number of events: 1651037

Latency (ms):
    min:                0.02
    avg:                0.02
    max:                3.30
    95th percentile:    0.02
    sum:               29848.91

Threads fairness:
    events (avg/stddev): 1651037.0000/0.00
    execution time (avg/stddev): 29.8489/0.00

```

### Iteration 5

<b>Iteration no.</b>	<b>Events per second</b>
1	55705.48 - max
2	53339.24 - min
3	53770.33
4	54454.38
5	55029.96
Average	54459.88

- Test 2:

```

WARNING: the --test option is deprecated. You can pass
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-1

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: 3203.72
General statistics:
  total time: 30.0012s
  total number of events: 96123
Latency (ms):
  min: 0.30
  avg: 0.31
  max: 9.90
  95th percentile: 0.33
  sum: 29927.79
Threads fairness:
  events (avg/stddev): 96123.0000/0.00
  execution time (avg/stddev): 29.9278/0.00

```

Iteration 1

```

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: 3190.15
General statistics:
  total time: 30.0207s
  total number of events: 95901
Latency (ms):
  min: 0.30
  avg: 0.31
  max: 8.36
  95th percentile: 0.33
  sum: 29967.28
Threads fairness:
  events (avg/stddev): 95901.0000/0.00
  execution time (avg/stddev): 29.9673/0.00
WARNING: the --test option is deprecated. You can pass
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-

```

Iteration 2

```

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: 3207.99
General statistics:
  total time: 30.0007s
  total number of events: 96245
Latency (ms):
  min: 0.30
  avg: 0.31
  max: 4.64
  95th percentile: 0.32
  sum: 29963.17
Threads fairness:
  events (avg/stddev): 96245.0000/0.00
  execution time (avg/stddev): 29.9632/0.00

```

WARNING: the --test option is deprecated. You can pass  
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-

Iteration 3

```

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: 3197.21
General statistics:
  total time: 30.0010s
  total number of events: 95924
Latency (ms):
  min: 0.30
  avg: 0.31
  max: 1.89
  95th percentile: 0.32
  sum: 29963.23
Threads fairness:
  events (avg/stddev): 95924.0000/0.00
  execution time (avg/stddev): 29.9632/0.00

```

sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

Iteration 4

```

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

Prime numbers limit: 20000

Initializing worker threads...

WARNING: the --test option is deprecated. You can pass
Threads started!

CPU speed:
  events per second: 3194.72

General statistics:
  total time:          30.0029s
  total number of events: 95854

Latency (ms):
  min:                0.30
  avg:                0.31
  max:                3.43
  95th percentile:    0.32
  sum:               29956.37

Threads fairness:
  events (avg/stddev):   95854.0000/0.00
  execution time (avg/stddev): 29.9564/0.00

```

### Iteration 5

<b>Iteration no.</b>	<b>Events per second</b>
1	3203.72
2	3190.15 - min
3	3207.99 - max
4	3197.21
5	3194.72
Average	3198.75

- Test 3:

<p>Running the test with following options:  Number of threads: 1  Initializing random number generator from current time</p> <p>Prime numbers limit: 200000  Initializing worker threads...  Threads started!  [  WARNING: the --test option is deprecated. You can pass a script  CPU speed:  events per second: 171.87</p> <p>General statistics:  total time: 30.0043s  total number of events: 5157</p> <p>Latency (ms):  min: 5.72  avg: 5.81  max: 15.25  95th percentile: 5.99  sum: 29967.81</p> <p>Threads fairness:  events (avg/stddev): 5157.0000/0.00  execution time (avg/stddev): 29.9678/0.00</p>	<p>Running the test with following options:  Number of threads: 1  Initializing random number generator from current time</p> <p>Prime numbers limit: 200000  Initializing worker threads...  Threads started!</p> <p>CPU speed:  events per second: 170.92</p> <p>General statistics:  total time: 30.0060s  total number of events: 5129</p> <p>Latency (ms):  min: 5.72  avg: 5.85  max: 13.94  95th percentile: 6.09  sum: 29982.56</p> <p>Threads fairness:  events (avg/stddev): 5129.0000/0.00  execution time (avg/stddev): 29.9826/0.00</p>
--	--

Iteration 1

Iteration 2

<p>Running the test with following options:  Number of threads: 1  Initializing random number generator from current time</p> <p>Prime numbers limit: 200000  Initializing worker threads...  WARNING: the --test option is deprecated. You can pass a script  Threads started!</p> <p>CPU speed:  events per second: 170.08</p> <p>General statistics:  total time: 30.0068s  total number of events: 5104</p> <p>Latency (ms):  min: 5.72  avg: 5.87  max: 16.41  95th percentile: 6.09  sum: 29981.24</p> <p>Threads fairness:  events (avg/stddev): 5104.0000/0.00  execution time (avg/stddev): 29.9812/0.00</p>	<p>Running the test with following options:  Number of threads: 1  Initializing random number generator from current time</p> <p>Prime numbers limit: 200000  Initializing worker threads...  WARNING: the --test option is deprecated. You can pass a script  Threads started!</p> <p>CPU speed:  events per second: 168.58</p> <p>General statistics:  total time: 30.0064s  total number of events: 5059</p> <p>Latency (ms):  min: 5.72  avg: 5.93  max: 18.57  95th percentile: 6.21  sum: 29978.28</p> <p>Threads fairness:  events (avg/stddev): 5059.0000/0.00  execution time (avg/stddev): 29.9783/0.00</p>
---	---

Iteration 3

Iteration 4

```

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: 167.92

General statistics:
  total time:          30.0058s
  total number of events: 5039

Latency (ms):
  min:                 5.73
  avg:                 5.95
  max:                13.19
  95th percentile:    6.09
  sum:               29963.09

Threads fairness:
  events (avg/stddev):   5039.0000/0.00
  execution time (avg/stddev): 29.9631/0.00

```

### Iteration 5

Iteration no.	Events per second
1	171.87 - max
2	170.92
3	170.08
4	168.58
5	167.92 - min
Average	169.87

### Conclusion:

Even after using more resources for both QEMU and Docker, there is very slight deviation in the performance. Also, when we increase max-prime value, events per seconds decreases.

### File I/O Testing QEMU Vs. Docker:

For file I/O testing, 2 mods are supported by sysbench:

- Sequential Rewrite (seqrewr)
- Combined random read/write (rndrw)

file size =3 GB

Performance is tested using experimental setups.

Two modes are used which sysbench supports, which are: We will keep the file size constant at 3GB and will test the performance against our experimental setups accordingly.

### **QEMU execution:**

1. Sequential Rewrite

```

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:        18410.73
    fsyncs/s:       23633.23

Throughput:
  read, MiB/s:      0.00
  written, MiB/s: 287.67

General statistics:
  total time:      30.0458s
  total number of events: 1261733

Latency (ms):
  min:              0.00
  avg:             0.38
  max:            469.79
  95th percentile: 1.47
  sum:           477690.18

Threads fairness:
  events (avg/stddev): 78858.3125/1462.36
  execution time (avg/stddev): 29.8556/0.07

```

Iteration no.	Output
1	reads/s = 0 writes/s = 18410.73 fsyncs/s = 23633.23 events/s = 21028.88
2	reads/s = 0 writes/s = 44361.77 fsyncs/s = 56848.83 events/s = 101252.66
3	reads/s = 0 writes/s = 44132.92 fsyncs/s = 56554.27 events/s = 100665.5
4	reads/s = 0 writes/s = 42941.03 fsyncs/s = 55031.80 events/s = 97985.86
5	reads/s = 0 writes/s = 40869.42 fsyncs/s = 52379.79

	events/s = 93235.5
--	--------------------

## 2. Random read Write (rndrw)

```

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:           11594.62
  writes/s:          7729.41
  fsyncs/s:          24799.34

Throughput:
  read, MiB/s:       181.17
  written, MiB/s:    120.77

General statistics:
  total time:        30.0235s
  total number of events: 1323432

Latency (ms):
  min:                0.00
  avg:                0.36
  max:                84.02
  95th percentile:   1.42
  sum:               478877.93

Threads fairness:
  events (avg/stddev): 82714.5000/459.07
  execution time (avg/stddev): 29.9299/0.00

```

Iteration no.	Output
1	reads/s = 11594.62 writes/s = 7729.41 fsyncs/s = 24799.34 events/s = 22057.2
2	reads/s = 12925.26 writes/s = 8616.51 fsyncs/s = 27637.79 events/s = 49145.3
3	reads/s = 10645.87 writes/s = 7096.89 fsyncs/s = 22777.00 events/s = 40479.13
4	reads/s = 11293.72 writes/s = 7529.37

	fsyncs/s = 24157.81 events/s = 42946.03
5	reads/s = 12491.06 writes/s = 8327.07 fsyncs/s = 26713.69 events/s = 47486.16

### Docker execution:

#### 1. Sequential Rewrite

```

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
20GiB 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:        33994.01
  fsyncs/s:        43579.68

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   631.16

General statistics:
  total time:       39.0225s
  total number of events: 2326942

Latency (ms):
  min:              0.00
  avg:              0.21
  max:              38.35
  95th percentile:  0.98
  sum:             478736.18

Threads fairness:
  events (avg/stddev): 146433.8780/943.68
  execution time (avg/stddev): 29.9210/0.00

```

Iteration no.	Output
1	reads/s = 0 writes/s = 33944.01 fsyncs/s = 43579.68 events/s = 77564.73
2	reads/s = 0 writes/s = 35435.34 fsyncs/s = 42342.32 events/s = 75343.24
3	reads/s = 0 writes/s = 33958.73 fsyncs/s = 43684.54 events/s = 77846.34
4	reads/s = 0 writes/s = 32091.93 fsyncs/s = 41321.54

	events/s = 73654.18
5	reads/s = 0 writes/s = 34546.54 fsyncs/s = 44323.20 events/s = 78234.32

## 2. Random read Write

```

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 TueJIT 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:           13950.71
  writes/s:          9300.24
  fsyncs/s:          29826.54

Throughput:
  read, MiB/s:      217.98
  written, MiB/s:   146.32

General statistics:
  total time:        30.0184s
  total number of events: 1691280

Latency (ms):
  min:                0.00
  avg:                0.30
  max:                35.62
  95th percentile:    0.98
  sum:               478965.26

Threads fairness:
  events (avg/stddev): 99458.0000/844.16
  execution time (avg/stddev): 29.9383/0.00

```

Iteration no.	Output
1	reads/s = 13950.71 writes/s = 9300.24 fsyncs/s = 29826.54 events/s = 53024.66
2	reads/s = 13155.49 writes/s = 8769.99 fsyncs/s = 28129.15 events/s = 50019.5
3	reads/s = 14353.95 writes/s = 9569.02 fsyncs/s = 30688.31 events/s = 54574.26
4	reads/s = 14674.66 writes/s = 9782.83 fsyncs/s = 31370.93 events/s = 55796.33
5	reads/s = 14488.90

	writes/s = 9659.14 fsyncs/s = 30975.53 events/s = 55094.1
--	---

Conclusion:

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

**Configuration 2: 4GB RAM and 4 cores**

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
```

Iteration no.	Events per second
1	3931.10
2	3933.90
3	4281.01 - max
4	4051.55 - min
5	3972.59
Average	4034.03

```

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

Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 3931.10

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

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

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

```

Iteration no.	Events per second
1	3931.10
2	3933.90
3	4281.01 - max
4	4051.55 - min
5	3972.59
Average	4034.03

```

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

Prime numbers limit: 200000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 188.73

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

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

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

```

Iteration no.	Events per second
1	188.73 - max
2	188.33
3	185.49 - min
4	189.86
5	189.83
Average	188.44

Docker 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: 52698.93  
  
General statistics:  
    total time: 30.00010  
    total number of events: 1581012  
  
Latency (ms):  
    min: 0.02  
    avg: 0.02  
    max: 2.89  
    95th percentile: 0.02  
    sum: 29734.11  
  
Threads fairness:  
    events (avg/stddev): 1581012.0000/0.00  
    execution time (avg/stddev): 29.7341/0.00
```

Iteration no.	Events per second
1	52698.93
2	52132.09
3	51893.90 - min
4	56403.96 - max
5	56134.25
Average	53852.62

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

**Prime numbers limit: 20000**

**Initializing worker threads...**

**Threads started!**

**CPU speed:**  
events per second: 3007.54

**General statistics:**

total time:	39.0003s
total number of events:	98229

**Latency (ms):**

min:	0.30
avg:	0.33
max:	10.71
95th percentile:	0.36
sum:	29962.54

**Threads fairness:**

events (avg/stddev):	98229.0000/0.00
execution time (avg/stddev):	29.9625/0.00

Iteration no.	Events per second
1	3007.54
2	3247.12 - max
3	3181.35
4	3164.53
5	2917.72 - min
Average	3103.65

```

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

Prime numbers limit: 200000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 169.27

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

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

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

```

Iteration no.	Events per second
1	169.27
2	172.51
3	172.34
4	171.67
5	173.24
Average	171.80

#### Conclusion:

QEMU is faster than docker. But, change in configuration does not have significant effect on performance.

#### File I/O Testing QEMU Vs. Docker:

#### QEMU execution:

## 1. Sequential Rewrite

```

Running the test with following options:
Number of threads: 16
Initialising 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 fync() each 100 requests.
Calling feync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential rewrite test
Initialising worker threads...

Threads started!

File operations:
  reads/s:          0.00
  writes/s:        38097.14
  fyncs/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

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 1.0.10 (using system LuaJIT 2.1.0-beta3)

```

Iteration no.	Output
1	reads/s = 0 writes/s = 38097.14 fsyncs/s = 48828.63 events/s = 87008.56
2	reads/s = 0 writes/s = 39530.62 fsyncs/s = 50644.21 events/s = 90199.3
3	reads/s = 0 writes/s = 35866.20 fsyncs/s = 46000.81 events/s = 81888.26
4	reads/s = 0 writes/s = 33329.60 fsyncs/s = 42727.80 events/s = 76088.93
5	reads/s = 0 writes/s = 33704.58 fsyncs/s = 43206.91 events/s = 76908.83

## 2. Random Read Write

```

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

Extra file open flags: (none)
128 files, 24MB each
3GB total file size
Block size 16KB
Number of IO requests: 6
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fync() each 100 requests.
Calling fynce() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initialising worker threads...

Threads started!

File operations:
  reads/s:           14887.51
  writes/s:          9898.23
  fsyncs/s:          31798.38

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

General statistics:
  total time:        39.0248s
  total number of events: 1693939

Latency (ms):
  min:               0.00
  avg:               0.28
  max:               37.23
  95th percentile:   0.84
  sum:              479034.62

Threads fairness:
  events (avg/stddev): 105871.1075/755.07
  execution time (avg/stddev): 29.9397/0.00

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 1.0.10 (using system LIBRARY 2.1.0-beta3)

```

Iteration no.	Output
1	reads/s = 14887.51 writes/s = 9898.23 fsyncs/s = 31798.38 events/s = 56464.43
2	reads/s = 13910 writes/s = 9273 fsyncs/s = 29740 events/s = 52915.76
3	reads/s = 14165.54 writes/s = 9443.58 fsyncs/s = 30284.24 events/s = 53880.36
4	reads/s = 12349.31 writes/s = 8232.87 fsyncs/s = 26409.53 events/s = 46994.56
5	reads/s = 12001.06 writes/s = 8000.63 fsyncs/s = 25669.69 events/s = 45638.56

### Docker Execution:

1. 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: 1484580

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

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

```

Iteration no.	Output
1	reads/s = 0 writes/s = 21687.04 fsyncs/s = 27826.67 events/s = 87008.56
2	reads/s = 0 writes/s = 22690.64 fsyncs/s = 29019.49 events/s = 51806.5
3	reads/s = 0 writes/s = 25688.43 fsyncs/s = 32946.24 events/s = 58659.23
4	reads/s = 0 writes/s = 22382.72 fsyncs/s = 28716.33 events/s = 51146.33

5	reads/s = 0 writes/s = 21880.69 fsyncs/s = 28071.27 events/s = 49996.26
---	--

```
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
30GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/write ratio for combined random IO test: 1.00
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:          13669.27
  writes/s:         9106.99
  fsyncs/s:        29204.47

Throughput:
  read, MiB/s:      213.43
  written, MiB/s:   142.26

General statistics:
  total time:       39.03900
  total number of events: 1669140

Latency (ms):
  min:              0.00
  avg:              0.31
  max:              38.66
  99th percentile:  0.92
  sum:             470019.26

Threads fairness:
  events (avg/stddev): 97444.2500/836.67
  execution time (avg/stddev): 39.9387/0.00

// A sysbench --num-threads=16 --test=fileio --file-total-size=30 --time=30 --fil
e-test-mode=rndrw cleanup
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-fef6a17dc4 (using Bundled LuaJIT 2.1.0-beta2)
```

Iteration no.	Output
1	reads/s = 13850.71 writes/s = 9320.24 fsyncs/s = 28826.54 events/s = 53032.66
2	reads/s = 14155.49 writes/s = 8754.94 fsyncs/s = 27129.15 events/s = 50045.5
3	reads/s = 14251.83 writes/s = 9545.02 fsyncs/s = 30457.31 events/s = 54375.26
4	reads/s = 14324.66 writes/s = 9231.83 fsyncs/s = 31432.93 events/s = 55987.33
5	reads/s = 14231.90 writes/s = 9658.14

	fsyncs/s = 30324.53 events/s = 55432.1
--	---

### Conclusion:

As compared with 2GB 2 core config, there is decrease in file i/o for QEMU and Docker.

### **Configuration 2: 6GB RAM and 6 cores**

#### 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
```

Iteration no.	Events per second
1	113197.63 - min
2	114251.46
3	113318.53
4	114254.59 - max
5	113529.89
Average	113710.42

```
Prime numbers limit: 20000
Initializing worker threads...
Threads started!
CPU speed:
  events per second: 4281.04
General statistics:
  total time:          30.0001s
  total number of events: 128434
Latency (ms):
  min:                0.22
  avg:                0.23
  max:                3.59
  95th percentile:    0.25
  sum:               29955.70
Threads fairness:
  events (avg/stddev): 128434.0000/0.00
  execution time (avg/stddev): 29.9557/0.00
```

Iteration no.	Events per second
1	4281.04 - max
2	3954.40
3	4284.01
4	4054.45
5	3942.69 - min
Average	4101.51

```

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

Prime numbers limit: 200000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 180.38

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

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

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

```

Iteration no.	Events per second
1	180.38 - min
2	182.33
3	184.49 - max
4	180.86
5	181.83
Average	181.97

Docker 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...

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

CPU speed:
events per second: 54026.70

General statistics:
total time: 30.00016
total number of events: 1620842

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

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

```

Iteration no.	Events per second
1	54026.70
2	52534.09
3	52843.95 - min
4	53403.88
5	56479.38 - max
Average	53857.6

```
Running the test with following options:  
Number of threads: 1  
Initializing random number generator from current time  
  
Prime numbers limit: 20000  
  
Initializing worker threads...  
  
Threads started!  
  
CPU speed:  
    events per second: 3129.44  
  
General statistics:  
    total time:          30.0002s  
    total number of events: 93886  
  
Latency (ms):  
    min:                0.30  
    avg:                0.32  
    max:                3.62  
    95th percentile:    0.34  
    sum:               29964.98  
  
Threads fairness:  
    events (avg/stddev): 93886.0000/0.00  
    execution time (avg/stddev): 29.9650/0.00
```

Iteration no.	Events per second
1	3129.44
2	3147.42
3	3141.35
4	3264.53 - max
5	3024.72 - min
Average	3141.49

```

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

Prime numbers limit: 200000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 171.39

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

Latency (ms):
min: 5.69
avg: 5.83
max: 7.56
95th percentile: 6.09
sum: 29995.84

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

```

Iteration no.	Events per second
1	171.39
2	172.47
3	171.34 - min
4	172.67
5	174.24 - max
Average	172.42

#### Conclusion:

There is no significant change in CPU performance after adding resources.

#### File I/O Testing QEMU Vs. Docker:

##### **QEMU execution:**

1. 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

```

Iteration no.	Output
1	reads/s = 0 writes/s = 25473.58 fsyncs/s = 32673.72 events/s = 58185
2	reads/s = 0 writes/s=27687.54 fsyncs/s = 33764.65 events/s = 59124.32
3	reads/s = 0 writes/s = 24534.32 fsyncs/s = 31986.87 events/s = 56126.76
4	reads/s = 0 writes/s = 25765.32 fsyncs/s = 32675.83

	events/s = 58234.21
5	reads/s = 0 writes/s = 28675.77 fsyncs/s = 35687.32 events/s = 59543.65

## 2. Random read Write

```

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

Extra file open flags: (none)
128 files, 24MiB each
3GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!

File operations:
  reads/s:          9938.58
  writes/s:         6625.83
  fsyncs/s:         21268.99

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

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

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

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

```

Iteration no.	Output
1	reads/s = 9938.58 writes/s = 6625.83 fsyncs/s = 21268.98 events/s = 37854.03
2	reads/s = 9845.43 writes/s = 6541.32 fsyncs/s = 20267.32

	events/s = 35643.76
3	reads/s = 9876.73 writes/s = 6543.21 fsyncs/s = 21124.43 events/s = 35541.28
4	reads/s = 9765.53 writes/s = 7021.32 fsyncs/s = 30382.11 events/s = 39878.32
5	reads/s = 9763.23 writes/s = 6312.63 fsyncs/s = 24569.69 events/s = 34532.56

#### Docker Execution:

##### 1. Sequential Rewrite

```

Running the test with following options:
Number of threads: 14
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:         13770.77
  fsync/s:          17701.47

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   215.29

General statistics:
  total time:        30.0093s
  total number of events: 945185

Latency (ms):
  min:                0.00
  avg:                0.51
  max:                56.15
  95th percentile:    1.42
  sum:               479365.45

Threads fairness:
  events (avg/stddev): 59074.0026/661.83
  execution time (avg/stddev): 29.9616/0.00

```

Iteration no.	Output
1	reads/s = 0 writes/s = 13778.77 fsyncs/s = 17701.47 events/s = 31506.16
2	reads/s = 0 writes/s = 13543.64 fsyncs/s = 17634.49 events/s = 30234.5
3	reads/s = 0 writes/s = 12456.32 fsyncs/s = 16323.87 events/s = 29876.36
4	reads/s = 0 writes/s = 14328.87 fsyncs/s = 18785.62 events/s = 32324.32
5	reads/s = 0 writes/s = 13534.11 fsyncs/s = 17431.89 events/s = 31232.21

## 2. Random read Write

```

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: 1.00
Periodic FSYNC enabled, calling sync() each 100 requests.
Calling sync() 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:          9561.28
  writes/s:         6334.24
  fsyncs/s:        26336.56

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

General statistics:
  total time:       30.07728
  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

```

<b>Iteration no.</b>	<b>Output</b>
1	reads/s = 9501.28 writes/s = 6334.24 fsyncs/s = 20336.56 events/s = 36197.8
2	reads/s = 9323.32 writes/s = 6213.41 fsyncs/s = 20223.12 events/s = 36097.87
3	reads/s = 9674.98 writes/s = 6984.73 fsyncs/s = 20989.89 events/s = 37019.21
4	reads/s = 9594.32 writes/s = 6434.43 fsyncs/s = 20214.43 events/s = 35788.33
5	reads/s = 9287.90 writes/s = 6021.14 fsyncs/s = 18673.53 events/s = 32764.12

### Conclusion:

Performance of file io decreases if resources are increased. Sometimes QEMU is better than Docker.

### Performance Analysis:

#### **QEMU:**

Disk:

2GB 2cores

##### 1. Sequential rewrite

- a. read,MiB/s=0
- b. written,MiB/s=715.68

##### 2. Random Read Write

- a. read, MiB/s=188.23
- b. written, MiB/s=122.43

4GB 4 cores

##### 1. Sequential rewrite

- a. read,MiB/s=0
- b. written,MiB/s=595.27

##### 2. Random Read Write

- a. read, MiB/s=231.99

b. written, MiB/s=154.66

6GB 6 cores

1. Sequential rewrite

- a. read,MiB/s=0
  - b. written,MiB/s=398.02
2. Random Read Write
- a. read, MiB/s=155.29
  - b. written, MiB/s=103.53

CPU:

Percentage of CPU used – 34.2%

Kernel Usage – user = 9.71% , System = 9.71%, Idle = 80.48%

**Docker:**

Disk:

2GB 2 cores

1. Sequential rewrite

- a. read,MiB/s=0
  - b. written,MiB/s=531.16
2. Random Read Write

- a. read, MiB/s=217.98
- b. written, MiB/s=145.32

4GB 4 cores

1. Sequential rewrite

- a. read,MiB/s=0
  - b. written,MiB/s=338.86
2. Random Read Write

- a. read, MiB/s=213.43
- b. written, MiB/s=142.28

6GB 6 cores

1. Sequential rewrite

- a. read,MiB/s=0
  - b. written,MiB/s=215.29
2. Random Read Write

- a. read, MiB/s=148.46
- b. written, MiB/s=98.97

CPU:

CPU Time used – 4hrs 45mins

Kernel used – 921604Kb => 115.2 MB

**Git Repository:**

Username : nolita26

Repository name: COEN241

Folder: CC\_HW1

Link: <https://github.com/nolita26/COEN241>

Commit ID: aa5f92c2196451a00cbc1225640261861ef125a6