

COEN 241: Homework 1

Name: Juhi Checker
SCU ID: 1605378

Detailed configurations of your experimental setup

My System Setup:

System: Macbook Pro(15 inch, 2016)
Processor: 2.7 GHz Quad-core Intel Core i7
Memory: 16Gb 2133 MHz LPDDR3

QEMU experimental setup

I have decided to go with 3 experimental setup for QEMU. I have made use of two flags to setup the below setup. First is “-m” for memory and “-smp” for number of cores:

1. 3GB memory and 1 core
2. 6GB memory and 2 cores
3. 9GB memory and 4 cores

Docker experimental setup

In order to keep the testing configuration same as QEMU I have 3 experimental setup for Docker as well. I have used the 2 flags to set the memory and number of cores being used. These are “--memory” and “--cpuset-cpus” respectively.

1. 3GB memory and 1 core
2. 6GB memory and 2 cores
3. 9GB memory and 4 cores

Steps taken while installing QEMU

1. Downloading the ISO Ubuntu image 20.04 server from the link given in the assignment.
2. Installing QEMU by using this command
 - a. brew install qemu
3. Creating the QEMU image by using this command. Since I had less space on my disk I gave only 10 GB space and kept my image name as ubuntu.img
 - a. qemu-img create -f raw -o size=10G ubuntu.img
4. Installing ISO file into the ubuntu.img
 - a. qemu-system-x86_64 -boot d -cdrom /Users/juhichecker/Downloads/ubuntu-20.04.5-live-server-amd64.iso -m 2048M -hda ubuntu.img
 - b. You would be asked to enter username and password for the system so please enter them and I went with all the default settings to setup the system.
5. To run QEMU image and start run the following command
 - a. sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 6G -smp 2 -boot strict=on
 - b. Enter laptop password
 - c. Wait for the setup of QEMU window. After entering your username and password you will see a screen below.

```

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:           1167.49
  writes/s:          778.26
  fsyncs/s:          2508.58

Throughput:
  read, MiB/s:      18.24
  written, MiB/s:   12.16

General statistics:
  total time:        25.1245s
  total number of events: 111433

Latency (ms):
  min:                0.01
  avg:                0.88
  max:                17.24
  95th percentile:    2.76
  sum:               98300.14

Threads fairness:
  events (avg/stddev): 27858.2500/385.11
  execution time (avg/stddev): 24.5750/0.00

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-beta0)

Removing test files...
jchecker@jchecker:~/cloud$ [ 3149.049529] watchdog: BUG: soft lockup - CPU#1 stuck for 21s! [swapper/1:0]
jchecker@jchecker:~/cloud$ jchecker@jchecker:~/cloud$ jchecker@jchecker:~/cloud$ jchecker@jchecker:~/cloud$ jchecker@jchecker:~/cloud$ jchecker@jchecker:~/cloud$
```

Steps taken while installing Docker

1. In order to install Docker, I used the link provided in the assignment for my computer.
2. After installing Docker, I used this command to check if there were any images existing in the docker.
 - a. docker images
3. Pulling ubuntu image. I downloaded Ubuntu:focal image since it supported apt-get commands and allowed me to download sysbench. I did the following using the below command.
 - a. docker pull ubuntu:focal

```

(base) Juhis-MacBook-Pro:~ juhichecker$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
hw1            latest        f6a19e22fafc  47 hours ago  134MB
trial           latest        890524d89d37  47 hours ago  72.8MB
ubuntu          kinetic-20220922 2793294e6a90  11 days ago   70.2MB
amd64/ubuntu    latest        216c552ea5ba  11 days ago   77.8MB
ubuntu          latest        216c552ea5ba  11 days ago   77.8MB
ubuntu          focal         817578334b4d  11 days ago   72.8MB
(base) Juhis-MacBook-Pro:~ juhichecker$
```

4. Creating your own image using ubuntu:focal as base image.
 - a. The following command runs the image in the background and keeps it running using the "-d" flag
 - i. docker run -d ubuntu:focal
 - b. To check if your image was pulled run the following command
 - i. docker images

```

brave_chaum
(base) Juhis-MacBook-Pro:~ juhichecker$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hw1            latest   f6a19e22fafc  2 days ago   134MB
trial          latest   890524d89d37  2 days ago   72.8MB
ubuntu          Kinetic-20220922 2793294e6a90  11 days ago  70.2MB
amd64/ubuntu    latest   216c552ea5ba  11 days ago  77.8MB
ubuntu          latest   216c552ea5ba  11 days ago  77.8MB
ubuntu          focal    817578334b4d  11 days ago  72.8MB
(base) Juhis-MacBook-Pro:~ juhichecker$ 

```

- c. Run the following command to check which images are currently running. You should see the ubuntu:focal
- docker ps

```

...emu-system-x86 ~ sudo      ~ -- bash      ~ -- bash      ... docker run -it hw1 +
[base] Juhis-MacBook-Pro:~ juhichecker$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
1b9a80acabac      brave_chaum        "bash"            16 hours ago      Up 16 hours
brave_chaum
(base) Juhis-MacBook-Pro:~ juhichecker$ 

```

- d. Enter the bash shell to install sysbench by running this command with the eeb4a being the container id. Output of the below command will take you to the root of the container
- docker exec -it eeb4a
- e. Install sysbench in it by following the steps bellow
- apt-get update
 - apt-get install sudo
 - This step because sudo is not already installed in the image
 - sudo apt-get install sysbench
 - sysbench --version

```

juhichecker — root@1b9a80acabac:/cloud — docker run -it hw1 — 100x47
...emu-system-x86 ~ sudo      ~ -- bash      ~ -- bash      ... docker run -it hw1 +
[root@1b9a80acabac:/cloud# sysbench --version
sysbench 1.0.18
root@1b9a80acabac:/cloud# 

```

5. To create a new image with the following updates I followed the below commands
- To stop the container
 - docker container stop eeb4
 - Committing all the changes to a new image, named “hw1”
 - docker commit eeb4 hw1
 - To check if the image was created
 - docker images
 - To run the new image. You might also run the command “sysbench --version” to check if the changes are saved in the new image.
 - docker run -it hw1

Experiments

Experiments are conducted in the following manner for all the experimental setup:

- CPU mode (using two flags “—cpu-max-prime” and “--time”)
 - cpu-max-prime = 1500 and time = 25
 - cpu-max-prime = 15000 and time = 25

c. $-\text{cpu_max_prime} = 150000$ and time = 25

2. FileIO mode

- a. Sequential write (SEQWR)
- b. Combined random Read/Write (RNDRW)

Configuration 1: Running on 3GB memory and 1 core

QEMU

```
juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 3G -smp 1 -boot strict=on
```

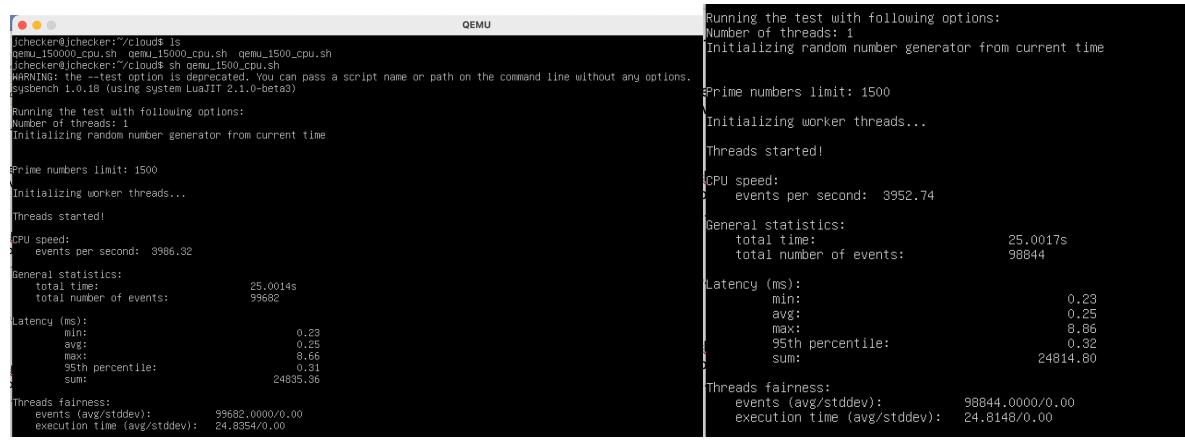


```
[juhichecker@Juhis-MacBook-Pro:~] juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 3G -smp 1 -boot strict=on --accel hvf
>Password:
[base] Juhis-MacBook-Pro:~ juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 3G -smp 2 -boot strict=on --accel hvf
[base] Juhis-MacBook-Pro:~ juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 3G -smp 2 -boot strict=on --accel hax
Failed to open the hax module
Assertion failed: (f != -1), function qemu_set_cloexec, file oslib-posix.c, line 251.
Abort trap: 6
[base] Juhis-MacBook-Pro:~ juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 3G -smp 2 -boot strict=on
You have mail in /var/mail/juhichecker
[base] Juhis-MacBook-Pro:~ juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 3G -smp 1 -boot strict=on
>Password:
[base] Juhis-MacBook-Pro:~ juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 6G -smp 2 -boot strict=on
>Password:
You have mail in /var/mail/juhichecker
[base] Juhis-MacBook-Pro:~ juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 9G -smp 4 -boot strict=on
>Password:
[base] Juhis-MacBook-Pro:~ juhichecker$
```

a. CPU Mode

- i. $-\text{cpu_max_prime} = 1500$ and time = 25

First, I start with a small limit of maximum prime value of 1500 for a time limit of 25 sec. I run this case for 5 iterations using a shell script.



```
[juhichecker@Juhis-MacBook-Pro:~] juhichecker$ ./clouds ls
qemu_i5000.cpu.sh  qemu_i5000_cpu.sh  qemu_i5000_cpu.sh
[juhichecker@Juhis-MacBook-Pro:~] juhichecker$ ./clouds sh qemu_i5000.cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta0)

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

Prime numbers limit: 1500
Threads started!

CPU speed:
events per second: 3952.74

General statistics:
total time: 25.0017s
total number of events: 98844

Latency (ms):
min: 0.23
avg: 0.25
max: 8.86
95th percentile: 0.32
sum: 24814.80

Threads fairness:
events (avg/stddev): 98844.0000/0.00
execution time (avg/stddev): 24.8148/0.00

[juhichecker@Juhis-MacBook-Pro:~] juhichecker$ ./clouds ls
qemu_i5000.cpu.sh  qemu_i5000_cpu.sh  qemu_i5000_cpu.sh
[juhichecker@Juhis-MacBook-Pro:~] juhichecker$ ./clouds sh qemu_i5000.cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta0)

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

Prime numbers limit: 1500
Threads started!

CPU speed:
events per second: 3906.32

General statistics:
total time: 25.0014s
total number of events: 99682

Latency (ms):
min: 0.23
avg: 0.25
max: 8.66
95th percentile: 0.31
sum: 24895.36

Threads fairness:
events (avg/stddev): 99682.0000/0.00
execution time (avg/stddev): 24.8354/0.00
```

Iteration 1

Iteration 2

<pre> Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1500 Initializing worker threads... Threads started! CPU speed: events per second: 3925.62 General statistics: total time: 25.0017s total number of events: 98167 Latency (ms): min: 0.23 avg: 0.25 max: 9.70 95th percentile: 0.32 sum: 24801.51 Threads fairness: events (avg/stddev): 98167.0000/0.00 execution time (avg/stddev): 24.8016/0.00 </pre>	<pre> Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1500 Initializing worker threads... Threads started! CPU speed: events per second: 4008.48 General statistics: total time: 25.0016s total number of events: 100243 Latency (ms): min: 0.23 avg: 0.25 max: 10.16 95th percentile: 0.31 sum: 24811.73 Threads fairness: events (avg/stddev): 100243.0000/0.00 execution time (avg/stddev): 24.8117/0.00 </pre>	<pre> Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1500 Initializing worker threads... Threads started! CPU speed: events per second: 3999.50 General statistics: total time: 25.0015s total number of events: 100014 Latency (ms): min: 0.23 avg: 0.25 max: 7.44 95th percentile: 0.31 sum: 24816.83 Threads fairness: events (avg/stddev): 100014.0000/0.00 execution time (avg/stddev): 24.8168/0.00 jchecker@jchecker:~/cloud\$ </pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Iteration 3

Iteration 4

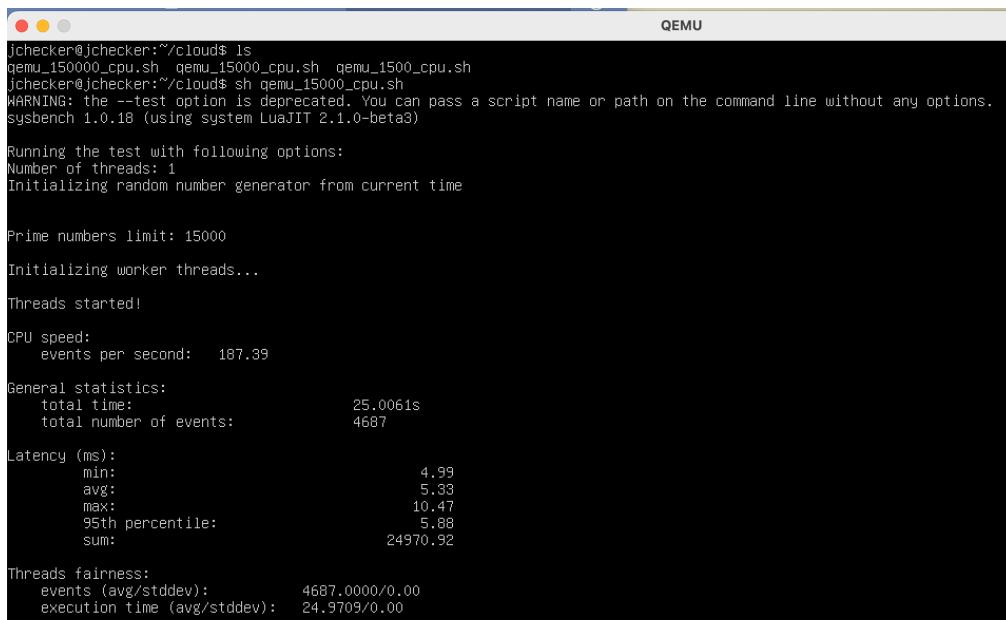
Iteration 5

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 0.23 Max = 8.66 Avg = 0.25	3986
2	Min = 0.23 Max = 8.86 Avg = 0.25	3952.74
3	Min = 0.23 Max = 9.70 Avg = 0.25	3999.5
4	Min = 0.23 Max = 10.16 Avg = 0.25	4008.48
5	Min = 0.23 Max = 7.44 Avg = 0.25	3925.62

ii. $-cpu\max-prime = 15000$ and $time = 25$

I start with a small limit of maximum prime value of 15000 which is 10 times the previous one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.



```
jchecker@jchecker:~/cloud$ ls
qemu_150000_cpu.sh  qemu_15000_cpu.sh  qemu_1500_cpu.sh
jchecker@jchecker:~/cloud$ sh qemu_15000_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuajIT 2.1.0-beta3)

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

Prime numbers limit: 15000
Initializing worker threads...

Threads started!

CPU speed:
events per second: 187.39

General statistics:
total time: 25.0061s
total number of events: 4687

Latency (ms):
min: 4.99
avg: 5.33
max: 10.47
95th percentile: 5.88
sum: 24970.92

Threads fairness:
events (avg/stddev): 4687.0000/0.00
execution time (avg/stddev): 24.9709/0.00
```

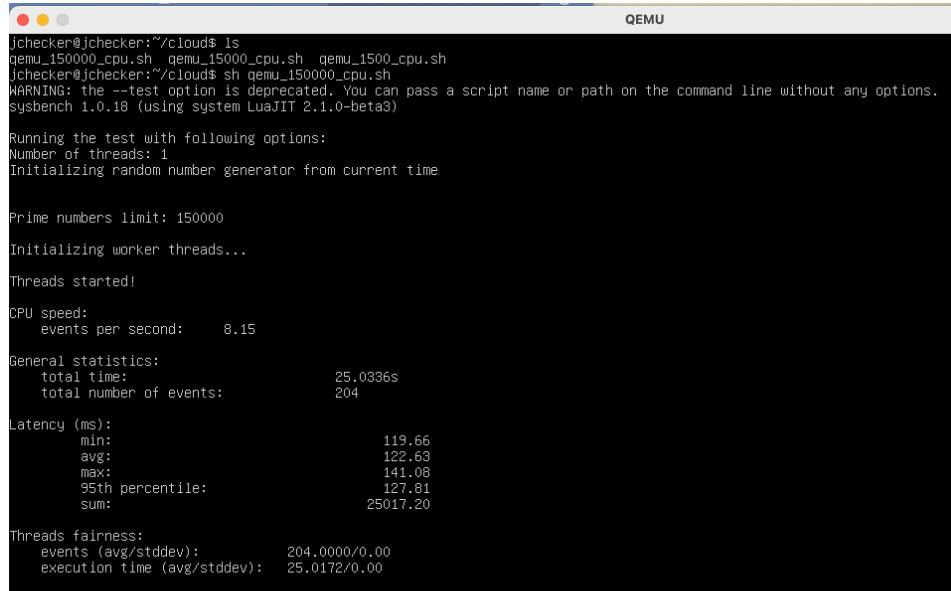
Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 4.99 Max = 10.47 Avg = 5.33	187.79
2	Min = 5.02 Max = 9.65 Avg = 5.36	186.16
3	Min = 4.99 Max = 10.50 Avg = 5.33	186.14
4	Min = 4.99 Max = 17.76 Avg = 5.34	187
5	Min = 5.03 Max = 13.93 Avg = 5.36	187.27

iii. $-\text{cpu-max-prime} = 150000$ and $\text{time} = 25$

I start with a small limit of maximum prime value of 150000 which is 100 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.



```
jchecker@jchecker:~/cloud$ ls
qemu_150000_cpu.sh  qemu_15000_cpu.sh  qemu_1500_cpu.sh
jchecker@jchecker:~/cloud$ sh qemu_150000_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 2.1.0 (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: 150000
Initializing worker threads...
Threads started!
CPU speed:
  events per second:     8.15
General statistics:
  total time:           25.0336s
  total number of events: 204
Latency (ms):
  min:                  119.66
  avg:                 122.63
  max:                  141.08
  95th percentile:      127.81
  sum:                 25017.20
Threads fairness:
  events (avg/stddev): 204.0000/0.00
  execution time (avg/stddev): 25.0172/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 119.66 Max = 141.08 Avg = 122.63	8.15
2	Min = 120.10 Max = 265.79 Avg = 129.26	8.04
3	Min = 119.29 Max = 380.16 Avg = 133.98	8.04
4	Min = 119.46 Max = 157.34 Avg = 123.59	7.44
5	Min = 119.79 Max = 154.15 Avg = 124.33	7.22

b. Fileio

```
QEMU - (Press ctrl + alt + g to release Mouse)
jchecker@jchecker:~/cloud$ ls
qemu_150000_cpu.sh  qemu_15000_cpu.sh  qemu_1500_cpu.sh  qemu_rndrw_fileio.sh  qemu_seqwr_fileio.sh
jchecker@jchecker:~/cloud$ cat qemu_seqwr_fileio.sh
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 prepare
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 run
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 cleanup
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 prepare
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 run
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 cleanup
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 prepare
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 run
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 cleanup
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 prepare
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 run
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 cleanup
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 prepare
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 run
subbench --test=fileio --file-total-size=3G --time=25 --file-test-mode=seqwr --num-threads=4 cleanup
jchecker@jchecker:~/cloud$ sh qemu_seqwr_fileio.sh && sh qemu_rndrw_fileio.sh
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
subbench 1.0.18 (using system LuajIT 2.1.0-beta3)

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
```

i. SEQWR

First, I run sequential write fileio mode for 5 iterations using a shell script.

```
QEMU
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 write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:                      0.00
  writes/s:                     2200.52
  fsyncs/s:                      2836.43

Throughput:
  read, MiB/s:                  0.00
  written, MiB/s:                34.38

General statistics:
  total time:                   25.0820s
  total number of events:        125840

Latency (ms):
  min:                           0.11
  avg:                           0.78
  max:                           72.98
  95th percentile:              3.82
  sum:                          97791.23

Threads fairness:
  events (avg/stddev):          81460.0000/85.82
  execution time (avg/stddev):   24.4478/0.03
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput (read (Mib/s) = 0.0)
1	Min = 0.11 Max = 72.98 Avg = 0.78	Written (Mib/s) = 34.38
2	Min = 0.11 Max = 39.53 Avg = 0.77	Written (Mib/s) = 34.44
3	Min = 0.11 Max = 43.27 Avg = 0.78	Written (Mib/s) = 31.98
4	Min = 0.11 Max = 84.71 Avg = 0.78	Written (Mib/s) = 34.8
5	Min = 0.11 Max = 37.02 Avg = 0.76	Written (Mib/s) = 35.67

ii. RNDRW

I run combined random read write fileio mode for 5 iterations using a shell script.

```

Running the test with following options:
Number of threads: 4
Initializing random number generator from current time

Extra file open flags: (none)
128 files, 24MiB each
36iB 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:          1071.28
  writes/s:         714.12
  fsyncs/s:        2304.88

Throughput:
  read, MiB/s:      16.74
  written, MiB/s:   11.16

General statistics:
  total time:           25.0887s
  total number of events: 102123

Latency (ms):
  min:                  0.02
  avg:                  0.96
  max:                 18.98
  95th percentile:      3.13
  sum:                98293.91

Threads fairness:
  events (avg/stddev): 25530.7500/118.67
  execution time (avg/stddev): 24.5735/0.02

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput
1	Min = 0.02 Max = 18.98 Avg = 0.96	Read (Mib/s) = 16.74 Written (Mib/s) = 11.36
2	Min = 0.02 Max = 62.36 Avg = 0.98	Read (Mib/s) = 16.87 Written (Mib/s) = 11.25
3	Min = 0.02 Max = 21.85 Avg = 0.96	Read (Mib/s) = 16.87 Written (Mib/s) = 10.34
4	Min = 0.02 Max = 23.65 Avg = 0.95	Read (Mib/s) = 16.47 Written (Mib/s) = 11.4
5	Min = 0.02 Max = 19.56 Avg = 0.95	Read (Mib/s) = 17.23 Written (Mib/s) = 1133

Docker

juhichecker\$ docker run -it --memory=3G --cpuset-cpus=0 hw1

a. CPU Mode

i. --cpu-max-prime = 1500 and time = 25

I start with a small limit of maximum prime value of 1500 for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9483.70
General statistics:
total time: 25.0002s
total number of events: 237114
Latency (ms):
min: 0.02
avg: 0.09
max: 6.64
95th percentile: 0.14
sum: 20522.58
Threads fairness:
events (avg/stddev): 237114.0000/0.00
execution time (avg/stddev): 20.5226/0.00

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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9875.45
General statistics:
total time: 25.0001s
total number of events: 246904
Latency (ms):
min: 0.06
avg: 0.08
max: 4.22
95th percentile: 0.14
sum: 20505.88
Threads fairness:
events (avg/stddev): 246904.0000/0.00
execution time (avg/stddev): 20.5059/0.00

```

Iteration 1 Iteration 2

```

juhichecker --root@d17c79977b85:/-- docker run -it --mem... juhichecker --root@d17c79977b85:/-- docker run -it --mem... juhichecker --root@d17c79977b85:/-- docker run -it --mem...
...6 ~ sudo ...-bash ...us=0 hw1 ...-bash + ...6 ~ sudo ...-bash ...us=0 hw1 ...-bash + ...6 ~ sudo ...-bash ...us=0 hw1 ...-bash + ...

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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9624.04
General statistics:
total time: 25.00001s
total number of events: 240619
Latency (ms):
min: 0.02
avg: 0.09
max: 13.93
95th percentile: 0.14
sum: 20518.60
Threads fairness:
events (avg/stddev): 240619.0000/0.00
execution time (avg/stddev): 28.5186/0.00

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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9988.69
General statistics:
total time: 25.00001s
total number of events: 249736
Latency (ms):
min: 0.02
avg: 0.08
max: 11.61
95th percentile: 0.14
sum: 20495.32
Threads fairness:
events (avg/stddev): 249736.0000/0.00
execution time (avg/stddev): 28.4953/0.00

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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9972.04
General statistics:
total time: 25.00001s
total number of events: 249319
Latency (ms):
min: 0.02
avg: 0.08
max: 0.64
95th percentile: 0.14
sum: 20478.71
Threads fairness:
events (avg/stddev): 249319.0000/0.00
execution time (avg/stddev): 28.4787/0.00

```

Iteration 1

Iteration 2

Iteration 3

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 0.02 Max = 6.64 Avg = 0.09	9483.70
2	Min = 0.04 Max = 0.64 Avg = 0.08	9875.45
3	Min = 0.06 Max = 11.61 Avg = 0.08	9624.04
4	Min = 0.06 Max = 4.22 Avg = 0.08	9988.0
5	Min = 0.02 Max = 13.93 Avg = 0.09	9972.04

ii. $-cpu-max-prime = 15000$ and $time = 25$

I start with a small limit of maximum prime value of 15000 which is 10 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

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

Prime numbers limit: 15000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 557.38

General statistics:
total time: 25.0010s
total number of events: 13936

Latency (ms):
min: 1.66
avg: 1.78
max: 3.21
95th percentile: 1.96
sum: 24739.39

Threads fairness:
events (avg/stddev): 13936.0000/0.00
execution time (avg/stddev): 24.7394/0.00

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 1.66 Max = 12.69 Avg = 1.77	557.3
2	Min = 1.66 Max = 3.21 Avg = 1.778	567.45
3	Min = 1.66 Max = 6.51 Avg = 1.77	524.9
4	Min = 1.66 Max = 3.77 Avg = 1.77	558.31
5	Min = 1.66 Max = 3.14 Avg = 1.77	553.78

iii. $-cpu-max-prime = 150000$ and $time = 25$

I start with a small limit of maximum prime value of 150000 which is 100 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```
juhichecker — root@d17c79977b85: / — docker run -it --mem...
```

...6 * sudo	~ — -bash	...us=0 hw1	...— -bash	+
-------------	-----------	-------------	------------	---

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

Prime numbers limit: 150000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 23.88

General statistics:
total time: 25.0393s
total number of events: 598

Latency (ms):
min: 40.39
avg: 41.85
max: 55.25
95th percentile: 43.39
sum: 25026.11

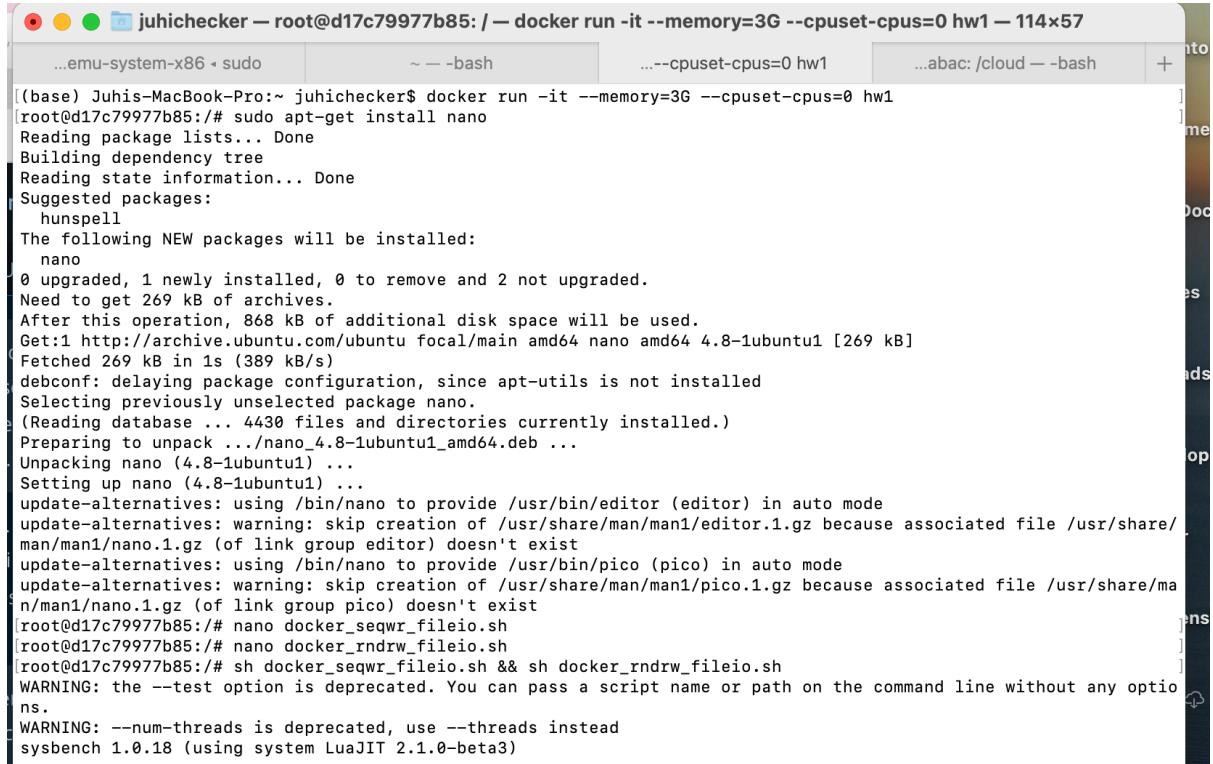
Threads fairness:
events (avg/stddev): 598.0000/0.00
execution time (avg/stddev): 25.0261/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 40.39 Max = 55.25 Avg = 41.05	23.88
2	Min = 40.89 Max = 55.99 Avg = 41.96	23.97
3	Min = 40.9 Max = 51.12 Avg = 41.92	29.35
4	Min = 40.88 Max = 49.0 Avg = 41.78	23.78
5	Min = 40.66 Max = 50.67 Avg = 41.9	22.85

b. FileIO



```
[(base) Juhis-MacBook-Pro:~ juhichecker$ docker run -it --memory=3G --cpuset-cpus=0 hw1 -- 114x57
...emu-system-x86 < sudo      ~ -- bash      ...cpuset-cpus=0 hw1      ...abac: /cloud -- bash      +
[juhichecker ~] juhichecker$ docker run -it --memory=3G --cpuset-cpus=0 hw1
[root@d17c79977b85:/# sudo apt-get install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  hunspell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 269 kB of archives.
After this operation, 868 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 nano amd64 4.8-1ubuntu1 [269 kB]
Fetched 269 kB in 1s (389 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 4430 files and directories currently installed.)
Preparing to unpack .../nano_4.8-1ubuntu1_amd64.deb ...
Unpacking nano (4.8-1ubuntu1) ...
Setting up nano (4.8-1ubuntu1) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group editor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group pico) doesn't exist
[root@d17c79977b85:/# nano docker_seqwr_fileio.sh
[root@d17c79977b85:/# nano docker_rndrw_fileio.sh
[root@d17c79977b85:/# sh docker_seqwr_fileio.sh && sh docker_rndrw_fileio.sh
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)
```

i. SEQWR

I run sequential write fileio mode for 5 iterations using a shell script.

```

juhichecker — root@d17c79977b85: / — docker run -it --memo...
...6 ← sudo      ~ — -bash      ...us=0 hw1      ...— -bash      +
Running the test with following options:
Number of threads: 4
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 write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:          0.00
  writes/s:        2813.70
  fsyncs/s:        3617.76

Throughput:
  read, MiB/s:     0.00
  written, MiB/s:  43.96

General statistics:
  total time:       25.0900s
  total number of events: 160863

Latency (ms):
  min:              0.01
  avg:             0.48
  max:            117.46
  95th percentile: 1.34
  sum:           77224.24

Threads fairness:
  events (avg/stddev):   40215.7500/677.74
  execution time (avg/stddev): 19.3061/0.03

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput (read (Mib/s) = 0.0)
1	Min = 0.01 Max = 170.32 Avg = 0.62	Written (Mib/s) = 43.96
2	Min = 0.01 Max = 379.22 Avg = 0.97	Written (Mib/s) = 39.05
3	Min = 0.01 Max = 84.11 Avg = 0.55	Written (Mib/s) = 37.45
4	Min = 0.01 Max = 117.46 Avg = 0.48	Written (Mib/s) = 36.89
5	Min = 0.01 Max = 127.63 Avg = 0.58	Written (Mib/s) = 39.35

ii. RNDRW

I run combined random read write fileio mode for 5 iterations using a shell script.

```

juhichecker — root@d17c79977b85: / — docker run -it --memo...
...6 ✘ sudo ~ — -bash ...us=0 hw1 ...— -bash + 

Running the test with following options:
Number of threads: 4
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:          1401.28
  writes/s:         934.12
  fsyncs/s:        3005.77

Throughput:
  read, MiB/s:      21.89
  written, MiB/s:   14.60

General statistics:
  total time:           25.0905s
  total number of events: 133509

Latency (ms):
  min:                 0.01
  avg:                 0.66
  max:                21.49
  95th percentile:     1.86
  sum:               88380.04

Threads fairness:
  events (avg/stddev): 33377.2500/119.86
  execution time (avg/stddev): 22.0950/0.02

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput
1	Min = 0.01 Max = 21.49 Avg = 0.66	Read (Mib/s) = 21.89 Written (Mib/s) = 14.60
2	Min = 0.01 Max = 186.20 Avg = 0.72	Read (Mib/s) = 28.53 Written (Mib/s) = 13.21
3	Min = 0.01 Max = 154.36 Avg = 0.70	Read (Mib/s) = 21.45 Written (Mib/s) = 14.34
4	Min = 0.01 Max = 45.43 Avg = 0.68	Read (Mib/s) = 21.78 Written (Mib/s) = 14.24
5	Min = 0.01 Max = 42.49 Avg = 0.68	Read (Mib/s) = 21.34 Written (Mib/s) = 14.34

Conclusion:

1. In CPU mode test it is observed that as the number of max prime value increases the latency of QEMU and Docker also increases. However the latency of QEMU is higher than that of Docker for almost all the cases.
2. In File IO mode test it is observed that for sequential write operation latency of QEMU was almost similar to that of Docker. In combined random read write operation the latency for docker was less than that of QEMU.

Configuration 2: Running on 6GB memory and 2 core

QEMU

```
juhichecker$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 6G -smp 2 -boot strict=on
```

a. CPU Mode

i. --cpu-max-prime = 1500 and time = 25

First, I start with a small limit of maximum prime value of 1500 for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```
jchecker@jchecker:~/cloud/ && ls
qemu_150000_cpu.sh  qemu_15000_cpu.sh  qemu_1500_cpu.sh
jchecker@jchecker:~/cloud$ sh qemu_1500_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path
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: 1500
Initializing worker threads...
Threads started!
CPU speed:
    events per second: 3917.98

General statistics:
    total time:          25.0016s
    total number of events: 97979

Latency (ms):
    min:                  0.28
    avg:                  0.25
    max:                  2.16
    95th percentile:     0.32
    sum:                 24560.92

Threads fairness:
    events (avg/stddev): 97979.0000/0.00
    execution time (avg/stddev): 24.5609/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 0.23 Max = 2.16 Avg = 0.25	3917.98
2	Min = 0.23 Max = 3.10 Avg = 0.25	3889.99
3	Min = 0.23 Max = 4.11 Avg = 0.25	3908.13
4	Min = 0.23 Max = 3.91 Avg = 0.25	3899.13
5	Min = 0.23 Max = 3.51 Avg = 0.25	3945.14

ii. `-cpu-max-prime = 15000 and time = 25`

I start with a small limit of maximum prime value of 15000 which is 10 times the previous one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```
jchecker@jchecker:~/cloud$ ls
qemu_150000_cpu.sh  qemu_15000_cpu.sh  qemu_1500_cpu.sh
jchecker@jchecker:~/cloud$ sh qemu_15000_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path
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: 15000
Initializing worker threads...

Threads started!

CPU speed:
    events per second: 185.10

General statistics:
    total time:          25.0023s
    total number of events: 4629

Latency (ms):
    min:                  5.06
    avg:                  5.39
    max:                  8.48
    95th percentile:      5.88
    sum:                 24946.11

Threads fairness:
    events (avg/stddev): 4629.0000/0.00
    execution time (avg/stddev): 24.9461/0.00
```

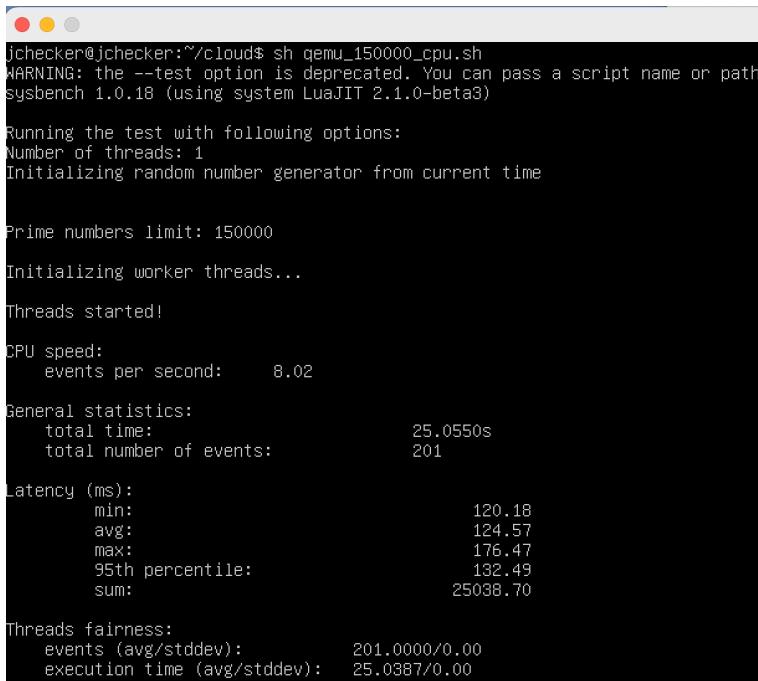
Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 5.06 Max = 8.48 Avg = 5.39	185.1
2	Min = 4.99 Max = 9.53 Avg = 5.33	186.34
3	Min = 4.99 Max = 10.31 Avg = 5.37	187.19
4	Min = 5.01 Max = 9.92 Avg = 5.34	185.34
5	Min = 5.02 Max = 10.28 Avg = 5.35	189.23

iii. `--cpu-max-prime = 150000 and time = 25`

I start with a small limit of maximum prime value of 150000 which is 100 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.



```
jchecker@jchecker:~/cloud$ sh qemu_150000_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path
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: 150000
Initializing worker threads...

Threads started!

CPU speed:
    events per second: 8.02

General statistics:
    total time:          25.0550s
    total number of events: 201

Latency (ms):
    min:                120.18
    avg:                124.57
    max:                176.47
    95th percentile:   132.49
    sum:                25038.70

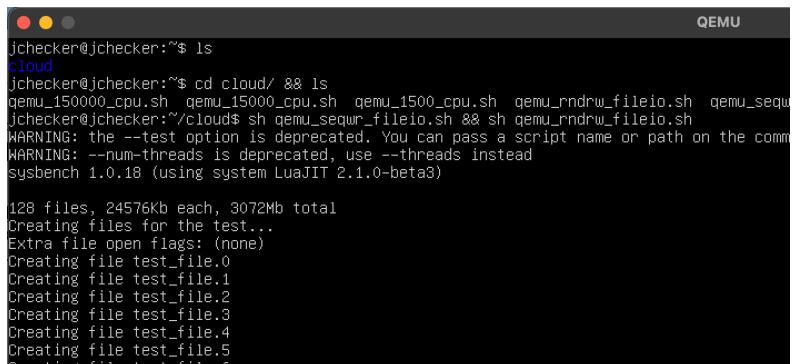
Threads fairness:
    events (avg/stddev): 201.0000/0.00
    execution time (avg/stddev): 25.0387/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 120.18 Max = 176.47 Avg = 124.12	8.02
2	Min = 119.9 Max = 174.1 Avg = 124.66	7.99
3	Min = 120.4 Max = 175.8 Avg = 125.1	8.01
4	Min = 119.9 Max = 152.2 Avg = 123.9	7.78
5	Min = 120.5 Max = 170.0 Avg = 125.4	8.06

b. FileIO

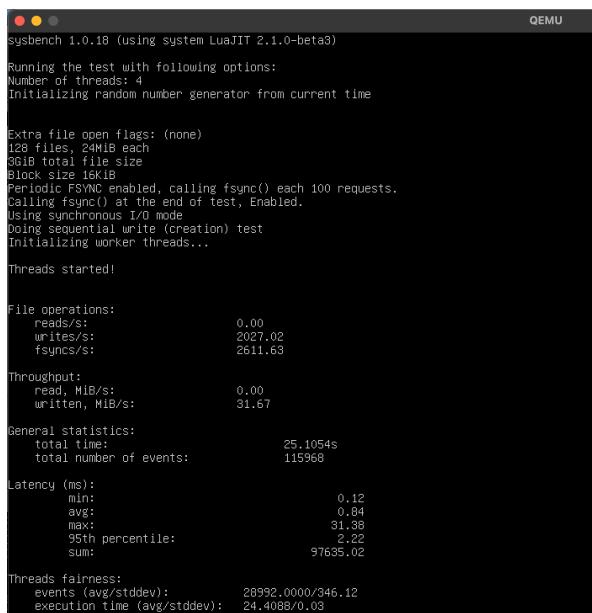


```
jchecker@jchecker:~$ ls
cloud
jchecker@jchecker:~$ cd cloud/ && ls
qemu_150000_cpu.sh  qemu_15000_cpu.sh  qemu_1500_cpu.sh  qemu_rndrw_fileio.sh  qemu_seqw
jchecker@jchecker:~/cloud$ sh qemu_seqwr_fileio.sh && sh qemu_rndrw_fileio.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line instead.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
Creating file test_file.24
Creating file test_file.25
Creating file test_file.26
Creating file test_file.27
Creating file test_file.28
Creating file test_file.29
Creating file test_file.30
Creating file test_file.31
Creating file test_file.32
Creating file test_file.33
Creating file test_file.34
Creating file test_file.35
Creating file test_file.36
Creating file test_file.37
Creating file test_file.38
Creating file test_file.39
Creating file test_file.40
Creating file test_file.41
Creating file test_file.42
Creating file test_file.43
Creating file test_file.44
Creating file test_file.45
Creating file test_file.46
Creating file test_file.47
Creating file test_file.48
Creating file test_file.49
Creating file test_file.50
Creating file test_file.51
Creating file test_file.52
Creating file test_file.53
Creating file test_file.54
Creating file test_file.55
Creating file test_file.56
Creating file test_file.57
Creating file test_file.58
Creating file test_file.59
Creating file test_file.60
Creating file test_file.61
Creating file test_file.62
Creating file test_file.63
Creating file test_file.64
Creating file test_file.65
Creating file test_file.66
Creating file test_file.67
Creating file test_file.68
Creating file test_file.69
Creating file test_file.70
Creating file test_file.71
Creating file test_file.72
Creating file test_file.73
Creating file test_file.74
Creating file test_file.75
Creating file test_file.76
Creating file test_file.77
Creating file test_file.78
Creating file test_file.79
Creating file test_file.80
Creating file test_file.81
Creating file test_file.82
Creating file test_file.83
Creating file test_file.84
Creating file test_file.85
Creating file test_file.86
Creating file test_file.87
Creating file test_file.88
Creating file test_file.89
Creating file test_file.90
Creating file test_file.91
Creating file test_file.92
Creating file test_file.93
Creating file test_file.94
Creating file test_file.95
Creating file test_file.96
Creating file test_file.97
Creating file test_file.98
Creating file test_file.99
Creating file test_file.100
Creating file test_file.101
Creating file test_file.102
Creating file test_file.103
Creating file test_file.104
Creating file test_file.105
Creating file test_file.106
Creating file test_file.107
Creating file test_file.108
Creating file test_file.109
Creating file test_file.110
Creating file test_file.111
Creating file test_file.112
Creating file test_file.113
Creating file test_file.114
Creating file test_file.115
Creating file test_file.116
Creating file test_file.117
Creating file test_file.118
Creating file test_file.119
Creating file test_file.120
Creating file test_file.121
Creating file test_file.122
Creating file test_file.123
Creating file test_file.124
Creating file test_file.125
Creating file test_file.126
Creating file test_file.127
```

i. SEQWR

First, I run sequential write fileio mode for 5 iterations using a shell script.



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

Running the test with following options:
Number of threads: 4
Initializing random number generator from current time

Extra file open flags: (none)
128 files, 24MB each
36GB total file size
Block size: 16KB
Fsync mode: FSync enabled, calling fsync() at the end of test, Enabled.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:          0.00
  writes/s:         2027.02
  fsyncs/s:        2611.63

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   31.67

General statistics:
  total time:           25.1054s
  total number of events: 115968

Latency (ms):
  min:                 0.12
  avg:                 0.84
  max:                91.38
  95th percentile:    2.22
  sum:               97695.02

Threads fairness:
  events (avg/stddev): 28992.0000/346.12
  execution time (avg/stddev): 24.4088/0.03
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput (read (Mib/s) = 0.0)
1	Min = 0.12 Max = 31.38 Avg = 0.84	Written (Mib/s) = 31.67
2	Min = 0.17 Max = 83.3 Avg = 0.85	Written (Mib/s) = 31.05
3	Min = 0.12 Max = 78.16 Avg = 0.9	Written (Mib/s) = 35.45
4	Min = 0.12 Max = 70.38 Avg = 0.85	Written (Mib/s) = 29.35
5	Min = 0.12 Max = 35.1 Avg = 0.84	Written (Mib/s) = 31.35

ii. RNDRW

I run combined random read write fileio mode for 5 iterations using a shell script.

```
QEMU
Running the test with following options:
Number of threads: 4
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:                  1201.27
    writes/s:                 800.91
    fsyncs/s:                 2579.62

Throughput:
    read, MiB/s:              18.77
    written, MiB/s:            12.51

General statistics:
    total time:                25.1169s
    total number of events:     114595

Latency (ms):
    min:                      0.02
    avg:                      0.86
    max:                      20.46
    95th percentile:           2.76
    sum:                      98225.95

Threads fairness:
    events (avg/stddev):      28648.7500/568.12
    execution time (avg/stddev): 24.5565/0.00
```

Iteration 1

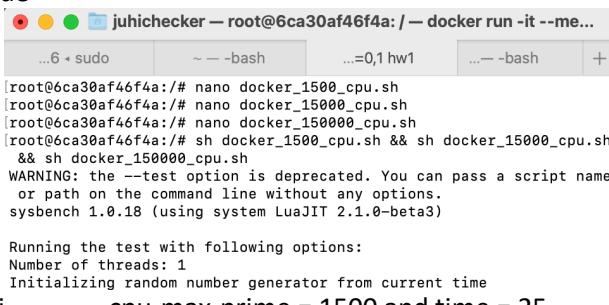
The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput
1	Min = 0.02 Max = 20.4 Avg = 0.86	Read (Mib/s) = 18.77 Written (Mib/s) = 12.51
2	Min = 0.02 Max = 26.3 Avg = 0.84	Read (Mib/s) = 19.00 Written (Mib/s) = 12.34
3	Min = 0.02 Max = 80.6 Avg = 0.86	Read (Mib/s) = 18.23 Written (Mib/s) = 12.89
4	Min = 0.02 Max = 17.2 Avg = 0.85	Read (Mib/s) = 19.23 Written (Mib/s) = 13.02
5	Min = 0.02 Max = 20.19 Avg = 0.84	Read (Mib/s) = 19.34 Written (Mib/s) = 12.99

Docker

juhichecker\$ docker run -it --memory=6G --cpuset-cpus=0,1 hw1

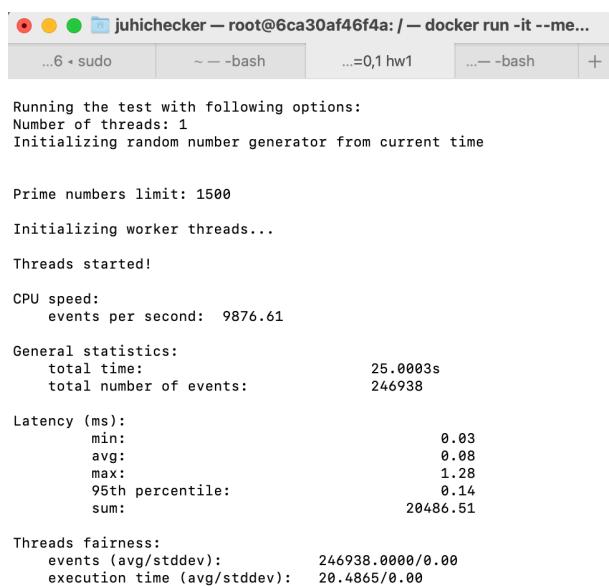
a. CPU Mode



```
[root@6ca30af46f4a:/# nano docker_1500_cpu.sh
[root@6ca30af46f4a:/# nano docker_15000_cpu.sh
[root@6ca30af46f4a:/# nano docker_150000_cpu.sh
[root@6ca30af46f4a:/# sh docker_1500_cpu.sh && sh docker_15000_cpu.sh
&& sh docker_150000_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name
or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time
i.      -cpu-max-prime = 1500 and time = 25
```

First, I start with a small limit of maximum prime value of 1500 for a time limit of 25 sec. I run this case for 5 iterations using a shell script.



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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9876.61

General statistics:
total time: 25.0003s
total number of events: 246938

Latency (ms):
min: 0.03
avg: 0.08
max: 1.28
95th percentile: 0.14
sum: 20486.51

Threads fairness:
events (avg/stddev): 246938.0000/0.00
execution time (avg/stddev): 20.4865/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 0.03 Max = 1.28 Avg = 0.08	9876.34
2	Min = 0.02 Max = 1.45 Avg = 0.08	9983.01
3	Min = 0.05 Max = 7.39 Avg = 0.08	9934.31
4	Min = 0.06 Max = 0.66 Avg = 0.08	9913.90
5	Min = 0.03 Max = 1.06 Avg = 0.08	98913.4

ii. -cpu-max-prime = 15000 and time = 25

I start with a small limit of maximum prime value of 15000 which is 10 times the previous one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

juhichecker — root@6ca30af46f4a: / — docker run -it --me...
...6 < sudo      ~ — -bash      ...=0,1 hw1      ...— -bash      +
Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Prime numbers limit: 15000
Initializing worker threads...
Threads started!
CPU speed:
events per second: 558.41

General statistics:
total time: 25.0015s
total number of events: 13962

Latency (ms):
min: 1.66
avg: 1.77
max: 3.46
95th percentile: 1.93
sum: 24738.10

Threads fairness:
events (avg/stddev): 13962.0000/0.00
execution time (avg/stddev): 24.7381/0.00

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 1.66 Max = 3.46 Avg = 1.77	558.41
2	Min = 1.62 Max = 3.08 Avg = 1.77	557.34
3	Min = 1.66 Max = 3.32 Avg = 1.78	560.32
4	Min = 1.66 Max = 3.13 Avg = 1.77	555.24
5	Min = 1.66 Max = 3.63 Avg = 1.77	576.34

iii. `--cpu-max-prime = 150000 and time = 25`

I start with a small limit of maximum prime value of 150000 which is 100 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

juhichecker - root@6ca30af46f4a:/ - docker run -it --me...
...6 < sudo      ~ -- -bash      ...=0,1 hw1      ...— -bash      +
Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Prime numbers limit: 150000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 23.87

General statistics:
    total time:          25.0114s
    total number of events: 597

Latency (ms):
    min:                  40.91
    avg:                  41.87
    max:                  52.65
    95th percentile:     44.17
    sum:                 24997.61

Threads fairness:
    events (avg/stddev): 597.0000/0.00
    execution time (avg/stddev): 24.9976/0.00

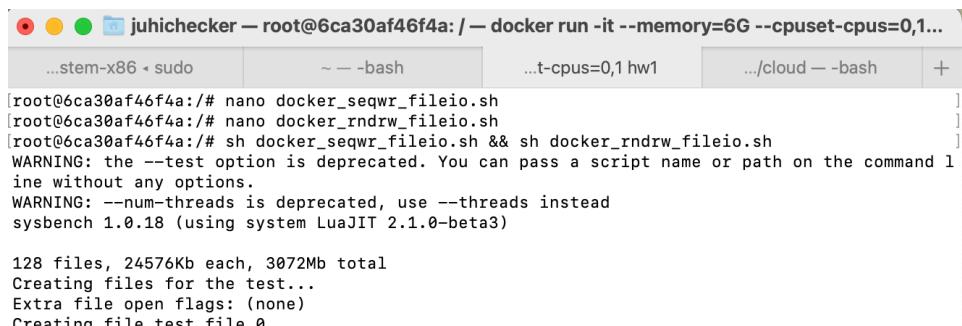
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 40.91 Max = 52.65 Avg = 41.87	23.87
2	Min = 41.0 Max = 50.3 Avg = 42.83	22.34
3	Min = 41.01 Max = 51.47 Avg = 42.63	22.50
4	Min = 40.98 Max = 53.86 Avg = 42.49	23.45
5	Min = 41.0 Max = 57.4 Avg = 42.69	22.34

b. FileIO

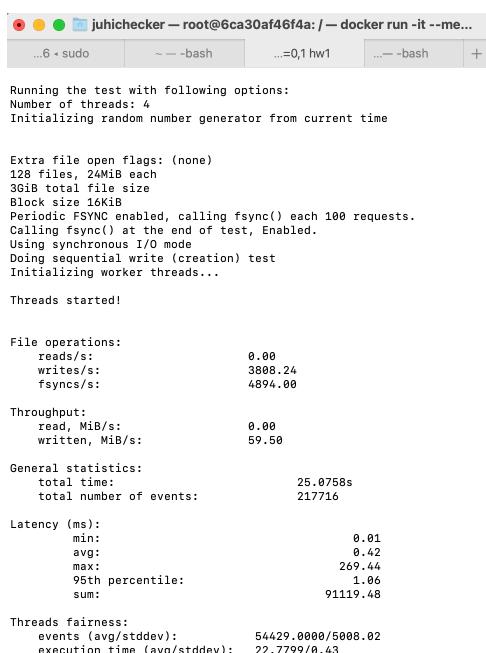


```
[root@6ca30af46f4a:/] nano docker_seqwr_fileio.sh
[root@6ca30af46f4a:/] nano docker_rndrw_fileio.sh
[root@6ca30af46f4a:/] sh docker_seqwr_fileio.sh && sh docker_rndrw_fileio.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
```

i. SEQWR

First, I run sequential write fileio mode for 5 iterations using a shell script.



```
Running the test with following options:
Number of threads: 4
Initializing random number generator from current time

Extra file open flags: (none)
128 files, 24MiB each
36GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling sync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:          0.00
  writes/s:        3808.24
  fsyncs/s:       4894.00

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   59.50

General statistics:
  total time:           25.0758s
  total number of events: 217716

Latency (ms):
  min:                  0.01
  avg:                  0.42
  max:                269.44
  95th percentile:     1.06
  sum:            91119.48

Threads fairness:
  events (avg/stddev): 54429.0000/5008.02
  execution time (avg/stddev): 22.7799/0.43
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput (read (Mib/s) = 0.0)
1	Min = 0.01 Max = 269.44 Avg = 0.42	Written (Mib/s) = 59.89
2	Min = 0.01 Max = 347.3 Avg = 0.42	Written (Mib/s) = 58.67
3	Min = 0.01 Max = 465.7 Avg = 0.43	Written (Mib/s) = 59.03
4	Min = 0.01 Max = 354.94 Avg = 0.47	Written (Mib/s) = 58.32
5	Min = 0.01 Max = 363.1 Avg = 0.47	Written (Mib/s) = 58.88

ii. RNDRW

I run combined random read write fileio mode for 5 iterations using a shell script.

```

juhichecker - root@6ca30af46f4a:/ -- docker run -it --me...
...6 + sudo      --- -bash    ...=0,1 hw1    ...- -bash    +
Running the test with following options:
Number of threads: 4
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:          1399.60
  writes/s:         933.00
  fsyncs/s:        3005.99

Throughput:
  read, MiB/s:      21.87
  written, MiB/s:   14.58

General statistics:
  total time:       25.0777s
  total number of events: 133376

Latency (ms):
  min:              0.01
  avg:              0.72
  max:             350.91
  95th percentile:  1.96
  sum:            95465.25

Threads fairness:
  events (avg/stddev): 33344.0000/1231.72
  execution time (avg/stddev): 23.8663/0.05

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput
1	Min = 0.01 Max = 350.9 Avg = 0.72	Read (Mib/s) = 21.87 Written (Mib/s) = 14.58
2	Min = 0.00 Max = 255.52 Avg = 0.67	Read (Mib/s) = 22.34 Written (Mib/s) = 15.01
3	Min = 0.0 Max = 434.43 Avg = 0.70	Read (Mib/s) = 23.21 Written (Mib/s) = 14.99
4	Min = 0.01 Max = 112.39 Avg = 0.66	Read (Mib/s) = 21.34 Written (Mib/s) = 15.34
5	Min = 0.01 Max = 203.63 Avg = 0.64	Read (Mib/s) = 24.45 Written (Mib/s) = 15.34

Conclusion:

1. In CPU mode test it is observed that as the number of max prime value increases the latency of QEMU and Docker also increases. It is observed that on smaller tasks and very huge tasks QEMU's latency is better than Docker, whereas on average tasks Docker latency is slightly better than QEMU.
2. In File IO mode test it is observed that for sequential write operation latency of QEMU was less than that of Docker. In combined random read write operation the latency for docker was less than that of QEMU.
3. There was no significant changes seen on doubling the memory and number of cores.

Configuration 3: Running on 9GB memory and 4 core

QEMU

juhichecker\$ sudo qemu-system-x86_64 -hda ubuntu.img -boot c -m 9G -smp 4 -boot strict=on

a. CPU Mode

i. --cpu-max-prime = 1500 and time = 25

First, I start with a small limit of maximum prime value of 1500 for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```
jchecker@jchecker:~$ cd cloud/ && ls
qemu_150000_cpu.sh qemu_15000_cpu.sh qemu_1500_cpu.sh
jchecker@jchecker:~/cloud$ sh qemu_1500_cpu.sh && sh qemu_15000_cpu.sh && sh
WARNING: the --test option is deprecated. You can pass a script name or path
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: 1500
Initializing worker threads...
Threads started!
CPU speed:
    events per second: 3948.63

General statistics:
    total time: 25.0018s
    total number of events: 98746

Latency (ms):
    min: 0.23
    avg: 0.25
    max: 1.03
    95th percentile: 0.31
    sum: 24514.47

Threads fairness:
    events (avg/stddev): 98746.0000/0.00
    execution time (avg/stddev): 24.5145/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 0.23 Max = 1.03 Avg = 0.25	3948.63
2	Min = 0.23 Max = 2.55 Avg = 0.25	3984.34
3	Min = 0.23 Max = 2.52 Avg = 0.25	4004.24
4	Min = 0.23 Max = 2.46 Avg = 0.25	3986.345
5	Min = 0.23 Max = 2.92 Avg = 0.25	3899.34

ii. $-\text{cpu-max-prime} = 15000$ and $\text{time} = 25$

I start with a small limit of maximum prime value of 15000 which is 10 times the previous one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

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

Prime numbers limit: 15000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 185.91

General statistics:
  total time: 25.0048s
  total number of events: 4650

Latency (ms):
  min: 5.01
  avg: 5.36
  max: 9.00
  95th percentile: 5.77
  sum: 24940.00

Threads fairness:
  events (avg/stddev): 4650.0000/0.00
  execution time (avg/stddev): 24.9400/0.00

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 5.01 Max = 9.0 Avg = 5.36	185.91
2	Min = 5.03 Max = 9.14 Avg = 5.36	187.34
3	Min = 5.01 Max = 8.77 Avg = 5.33	188.34
4	Min = 4.99 Max = 8.557 Avg = 5.32	187.0
5	Min = 4.9 Max = 10.01 Avg = 5.33	189.9

iii. **--cpu-max-prime = 150000 and time = 25**

I start with a small limit of maximum prime value of 150000 which is 100 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

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

Prime numbers limit: 150000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 8.09

General statistics:
total time: 25.0969s
total number of events: 203

Latency (ms):
min: 120.40
avg: 123.52
max: 154.01
95th percentile: 130.13
sum: 25075.23

Threads fairness:
events (avg/stddev): 203.0000/0.00
execution time (avg/stddev): 25.0752/0.00

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 120.99 Max = 169.73 Avg = 16.14	8.09
2	Min = 120.4 Max = 154.13 Avg = 123.52	8.04
3	Min = 120.84 Max = 153.91 Avg = 124.28	8.11
4	Min = 119.75 Max = 142.67 Avg = 123.75	7.92
5	Min = 120.25 Max = 156.26 Avg = 123.82	8.07

b. FileIO Mode

```

jchecker@jchecker:~/cloud$ sh qemu_seqwr_fileio.sh && sh qemu_rndrw_fileio.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line instead.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5

```

i. SEQWR

First, I run sequential write fileio mode for 5 iterations using a shell script.

```

sysbench 1.0.10 (using system LuaJIT 2.1.0-beta3)
Running the test with following options:
Number of threads: 4
Initializing random number generator from current time

Extra file open flags: (none)
128 files, 24MB each
36GB total file size
File size: 16KB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling sync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:          0.00
  writes/s:        1868.19
  fsync/s:         2409.21

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   29.19

General statistics:
  total time:       25.1255s
  total number of events: 106980

Latency (ms):
  min:                 0.13
  avg:                0.92
  max:                29.58
  95th percentile:    1.52
  sum:               98029.67

Threads fairness:
  events (avg/stddev): 26745.0000/141.34
  execution time (avg/stddev): 24.5074/0.00

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput (read (Mib/s) = 0.0)
1	Min = 0.13 Max = 29.58 Avg = 0.92	Written (Mib/s) = 29.19
2	Min = 0.68 Max = 51.11 Avg = 0.95	Written (Mib/s) = 30.21
3	Min = 0.12 Max = 25.3 Avg = 0.91	Written (Mib/s) = 29.99
4	Min = 0.13 Max = 26.45 Avg = 0.9	Written (Mib/s) = 30.38
5	Min = 0.12 Max = 32.68 Avg = 0.93	Written (Mib/s) = 29.91

ii. RNDRW

I run combined random read write fileio mode for 5 iterations using a shell script.

```

Running the test with following options:
Number of threads: 4
Initializing 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: 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:          1141.03
  writes/s:         760.76
  fsyncs/s:        2452.71

Throughput:
  read, MiB/s:      17.83
  written, MiB/s:   11.89

General statistics:
  total time:       25.1282s
  total number of events: 108935

Latency (ms):
  min:                 0.01
  avg:                 0.90
  max:                18.83
  95th percentile:    2.81
  sum:               98253.01

Threads fairness:
  events (avg/stddev): 27233.7500/105.14
  execution time (avg/stddev): 24.5633/0.00

```

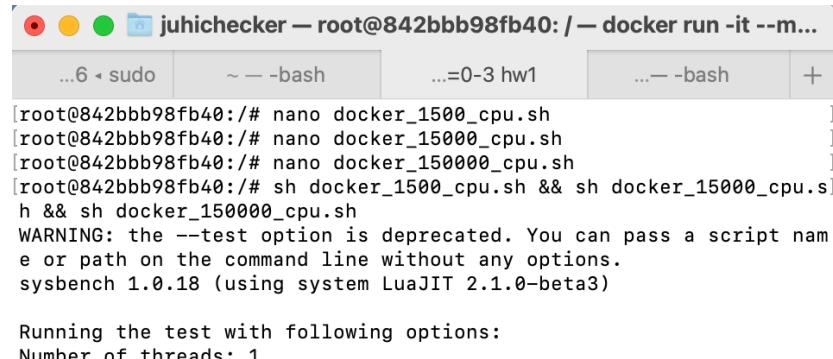
Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput
1	Min = 0.01 Max = 119.4 Avg = 0.09	Read (Mib/s) = 17.83 Written (Mib/s) = 11.09
2	Min = 0.01 Max = 18.82 Avg = 0.9	Read (Mib/s) = 18.34 Written (Mib/s) = 10.99
3	Min = 0.01 Max = 35.72 Avg = 0.94	Read (Mib/s) = 18.3 Written (Mib/s) = 11.23
4	Min = 0.0 Max = 17.2 Avg = 0.9	Read (Mib/s) = 17.23 Written (Mib/s) = 12.34
5	Min = 0.01 Max = 17.24 Avg = 0.08	Read (Mib/s) = 18.98 Written (Mib/s) = 11.63

Docker

```
juhichecker$ docker run -it --memory=9G --cpuset-cpus=0-3 hw1
```



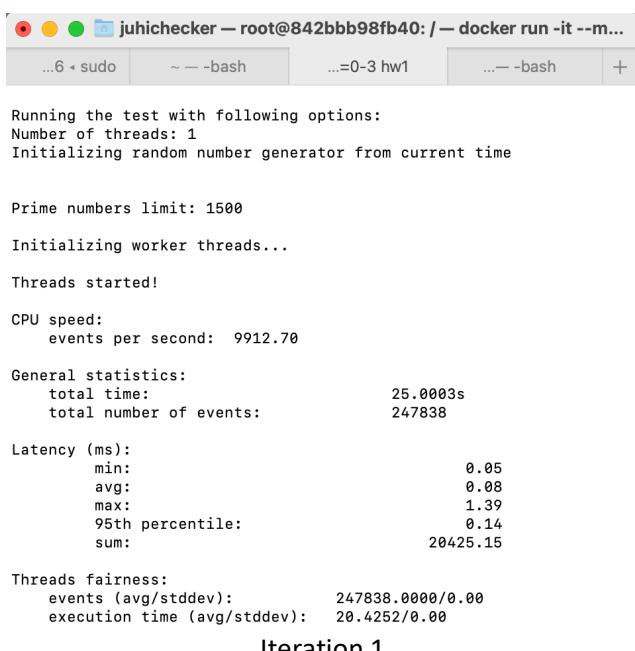
```
[root@842bbb98fb40:/] nano docker_1500_cpu.sh
[root@842bbb98fb40:/] nano docker_15000_cpu.sh
[root@842bbb98fb40:/] nano docker_150000_cpu.sh
[root@842bbb98fb40:/] sh docker_1500_cpu.sh && sh docker_15000_cpu.sh && sh docker_150000_cpu.sh
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
```

a. CPU Mode

- i. --cpu-max-prime = 1500 and time = 25

First, I start with a small limit of maximum prime value of 1500 for a time limit of 25 sec. I run this case for 5 iterations using a shell script.



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

Prime numbers limit: 1500
Initializing worker threads...
Threads started!
CPU speed:
events per second: 9912.70

General statistics:
total time: 25.0003s
total number of events: 247838

Latency (ms):
min: 0.05
avg: 0.08
max: 1.39
95th percentile: 0.14
sum: 20425.15

Threads fairness:
events (avg/stddev): 247838.0000/0.00
execution time (avg/stddev): 20.4252/0.00
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 0.04 Max = 1.01 Avg = 0.08	9921.78
2	Min = 0.04 Max = 0.82 Avg = 0.08	9934.13
3	Min = 0.01 Max = 2.53 Avg = 0.09	9957.56

4	Min = 0.05 Max = 1.39 Avg = 0.08	9744.24
5	Min = 0.06 Max = 0.73 Avg = 0.08	9667.8

ii. $-\text{cpu-max-prime} = 15000$ and $\text{time} = 25$

I start with a small limit of maximum prime value of 15000 which is 10 times the previous one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```

juhichecker — root@842bbb98fb40: / — docker run -it --m...
...6 < sudo ~ — -bash ...=0-3 hw1 ...— -bash + 

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

Prime numbers limit: 15000
Initializing worker threads...

Threads started!

CPU speed:
events per second: 557.48

General statistics:
total time: 25.0018s
total number of events: 13939

Latency (ms):
min: 1.66
avg: 1.77
max: 3.56
95th percentile: 1.93
sum: 24740.25

Threads fairness:
events (avg/stddev): 13939.0000/0.00
execution time (avg/stddev): 24.7403/0.00

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 1.66 Max = 4.99 Avg = 1.80	557.48
2	Min = 1.66 Max = 4.36 Avg = 1.79	546.7
3	Min = 1.66 Max = 4.67 Avg = 1.78	589.42
4	Min = 1.66 Max = 3.56 Avg = 1.77	588.24
5	Min = 1.66 Max = 14.15 Avg = 1.78	555.3

iii. $-cpu-max-prime = 150000$ and $time = 25$

I start with a small limit of maximum prime value of 150000 which is 100 times the first one for a time limit of 25 sec. I run this case for 5 iterations using a shell script.

```
juhichecker -- root@842bbb98fb40: /— docker run -it --m...
...6 * sudo ~ — -bash ...=0-3 hw1 ...— -bash + 

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

Prime numbers limit: 150000
Initializing worker threads...
Threads started!
CPU speed:
events per second: 23.76

General statistics:
total time: 25.0366s
total number of events: 595

Latency (ms):
min: 40.98
avg: 42.06
max: 51.45
95th percentile: 44.17
sum: 25023.39

Threads fairness:
events (avg/stddev): 595.0000/0.00
execution time (avg/stddev): 25.0234/0.00
Iteration 1
```

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Events/sec
1	Min = 41.02 Max = 51.82 Avg = 41.83	23.81
2	Min = 48.99 Max = 59.50 Avg = 41.98	22.45
3	Min = 41.02 Max = 50.81 Avg = 41.89	25.24
4	Min = 40.98 Max = 51.45 Avg = 42.96	23.23
5	Min = 41.00 Max = 50.89 Avg = 41.98	23.45

b. FileIO Mode

```
[root@842bbb98fb40:/]# docker run -it --memory=9G --cpuset-cpus=0-3 hw1
[(base) Juhis-MacBook-Pro:~ juhichecker$ docker run -it --memory=9G --cpuset-cpus=0-3 hw1
[root@842bbb98fb40:/]# sudo apt-get install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  hunspell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 269 kB of archives.
After this operation, 868 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 nano amd64 4.8-1ubuntu1 [269 kB]
Fetched 269 kB in 1s (426 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 4430 files and directories currently installed.)
Preparing to unpack .../nano_4.8-1ubuntu1_amd64.deb ...
Unpacking nano (4.8-1ubuntu1) ...
Setting up nano (4.8-1ubuntu1) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/nano.1.gz
i.  SEQWR
```

First, I run sequential write fileio mode for 5 iterations using a shell script.

```
[root@842bbb98fb40:/]# juhichecker --root@842bbb98fb40:/ -- docker run -it --memory=9G --cpuset-cpus=0-3 hw1
[(base) Juhis-MacBook-Pro:~ juhichecker$ Running the test with following options:
Number of threads: 4
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 write (creation) test
Initializing worker threads...

Threads started!

File operations:
  reads/s:          0.00
  writes/s:        3711.44
  fsyncs/s:        4768.34

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   57.99

General statistics:
  total time:           25.0831s
  total number of events: 212200

Latency (ms):
  min:                  0.01
  avg:                  0.45
  max:                246.99
  95th percentile:     0.87
  sum:            95218.40

Threads fairness:
  events (avg/stddev): 53050.0000/1314.68
  execution time (avg/stddev): 23.8046/0.04
```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput (read (Mib/s) = 0.0)
1	Min = 0.01 Max = 246.99 Avg = 0.45	Written (Mib/s) = 57.99
2	Min = 0.01 Max = 280.4 Avg = 0.49	Written (Mib/s) = 55.78
3	Min = 0.01 Max = 321.78 Avg = 0.47	Written (Mib/s) = 54.67
4	Min = 0.01 Max = 263.55 Avg = 0.46	Written (Mib/s) = 55.67
5	Min = 0.01 Max = 269.98 Avg = 0.46	Written (Mib/s) = 58.9

ii. RNDRW

I run combined random read write fileio mode for 5 iterations using a shell script.

```

root@842bbb98fb40:/# docker run -it --mem-reservation=10G --mem-swap-reservation=10G juhichecker
...6 sudo ~ -- bash ...=0-3 hw1 ...-- bash +
Running the test with following options:
Number of threads: 4
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:          1509.01
  writes/s:         1005.94
  fsyncs/s:         3235.33

Throughput:
  read, MiB/s:      23.58
  written, MiB/s:   15.72

General statistics:
  total time:        25.0884s
  total number of events: 143762

Latency (ms):
  min:                0.01
  avg:                0.67
  max:               475.55
  95th percentile:    1.96
  sum:              96718.17

Threads fairness:
  events (avg/stddev): 35940.5000/270.10
  execution time (avg/stddev): 24.1795/0.01

```

Iteration 1

The table below are the latency scores after the 5 iterations:

Iterations	Latency	Throughput
1	Min = 0.01 Max = 475.55 Avg = 0.67	Read (Mib/s) = 23.58 Written (Mib/s) = 15.72
2	Min = 0.01 Max = 296.68 Avg = 0.67	Read (Mib/s) = 22.45 Written (Mib/s) = 16.79
3	Min = 0.01 Max = 204.43 Avg = 0.70	Read (Mib/s) = 22.6 Written (Mib/s) = 15.8
4	Min = 0.01 Max = 593.98 Avg = 0.68	Read (Mib/s) = 24.33 Written (Mib/s) = 12.34
5	Min = 0.01 Max = 150.76 Avg = 0.66	Read (Mib/s) = 23.66 Written (Mib/s) = 11.2

Conclusion:

1. In CPU mode test it is observed that as the number of max prime value increases the latency of QEMU and Docker also increases. However the latency of QEMU is higher than that of Docker for almost all the cases.
2. In File IO mode test it is observed that for sequential write operation latency of QEMU was almost similar to that of Docker. In combined random read write operation the latency for docker was less than that of QEMU.
3. The results are similar to first configuration results.

Analysis of Performance Data

QEMU Analysis:

1. CPU utilization of QEMU:

Activity Monitor											Search					
All Processes		% CPU	CPU Time	Threads	Idle Wake-Ups	% GPU	GPU Time	PID	User							
qemu-system-x86_64		102.9	4:22.33	14	0	0.0	0.00	21561	root							
Safari Web Content		56.8	52:11.18	7	672	0.0	0.00	1489	juhichecker							
WindowServer		15.3	4:14:31.28	15	1647	1.0	3:22:52.59	165	windowserver							
Processes: 713 total, 4 running, 709 sleeping, 2960 threads											18:37:25					
Load Avg: 3.17, 3.08, 3.03 CPU usage: 22.86% user, 4.84% sys, 72.28% idle SharedLibs: 566M resident, 89M data, 120M linkededit.																
MemRegions: 290486 total, 5199M resident, 180M private, 3037M shared. PhysMem: 16G used (2854M wired), 45M unused.																
VM: 37T vsize, 3153M framework vsize, 54262370(0) swapins, 58783075(0) swapouts. Networks: packets: 20865768/200 in, 10560197/5189M out.																
Disks: 23380497/5806G read, 11961962/4556 written.																
PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PPGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID
21561	qemu-system-	102.3	04:10.38	12/1	2	196-	2335M-	2480K	239M	21558	21558	running	*#0[19]	0.09143	0.00000	0
1489	com.apple.We	65.5	52:04.12	7/1	1	121	187M	0B	181M	1489	1	running	0[174420]	0.00000	0.00000	501
165	WindowServer	7.8	04:14:30	14	6	5633-	990M-	512K-	230M-	165	1	sleeping	*#0[1]	0.87513	0.23160	88

Percentage of CPU used = 102.9

Kernel Usage

User = 22.86%

System = 4.8%

Idle = 72.28%

2. Disk Utilization

a. 3Gb memory and 1 core

- i. Sequential write
 - 1. Read (MiB/s) = 0.0
 - 2. Written (MiB/s) = 34.654
 - ii. Combined random read write
 - 1. Read (MiB/s) = 16.79
 - 2. Written (MiB/s) = 11.196
- b. 6Gb memory and 2 core
- i. Sequential write
 - 1. Read (MiB/s) = 0.0
 - 2. Written (MiB/s) = 31.14
 - ii. Combined random read write
 - 1. Read (MiB/s) = 18.95
 - 2. Written (MiB/s) = 12.638
- c. 9Gb memory and 4 core
- i. Sequential write
 - 1. Read (MiB/s) = 0.0
 - 2. Written (MiB/s) = 29.008
 - ii. Combined random read write
 - 1. Read (MiB/s) = 17.706
 - 2. Written (MiB/s) = 11.872

Docker Analysis:

1. CPU utilization of Docker:

Activity Monitor											Search			
All Processes		% CPU	CPU Time	Threads	Idle Wake-Ups	% GPU	GPU Time	PID	User					
com.docker.hyperkit		104.9	57:18.09	18	1149	0.0	0.00	79464	juhichecker					
Safari Web Content		70.0	33:11.87	7	21	0.0	0.00	1489	juhichecker					
juhichecker — top — 139x59														
~ -- bash			~ -- top											
Processes: 712 total, 4 running, 708 sleeping, 2944 threads														
Load Avg: 4.47, 3.76, 3.22 CPU usage: 15.58% user, 17.66% sys, 66.74% idle SharedLibs: 568M resident, 89M data, 120M linkedit.														
MemRegions: 289942 total, 4963M resident, 176M private, 2868M shared. PhysMem: 15G used (2794M wired), 625M unused.														
VM: 37T vsize, 3153M framework vsize, 54179235(64) swapins, 58588193(0) swapouts. Networks: packets: 20857972/20G in, 10555849/5188M out.														
Disks: 23250447/579G read, 11914518/453G written.														
PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME %CPU_OTHRS UID
79464	com.docker.h	106.7	57:08.70	18/1	0	43	3936M	0B	2502M-	79447	79453	running	*0[1]	0.00000 0.00000 501
1489	com.apple.We	89.5	33:05.22	7/1	1	120	187M	0B	181M	1489	1	running	0[174418]	0.00000 0.00000 501
165	WindowServer	20.1	04:14:10	15	6	5093	985M+	4480K-	230M	165	1	sleeping	*0[1]	0.03230 0.99813 88
9083	Activity Mon	16.0	02:55.93	5	3	904+	86M+	4528K+	39M-	9083	1	sleeping	*0-[492]	6.54365 0.00000 501
0	kernel_task	7.5	03:25:33	243/8	0	0	697M	0B	0B	0	0	running	*0[0]	0.00000 0.00000 0
20651	top	7.3	00:02.63	1/1	0	28	8296K+	0B	0B	20651	77904	running	*0[1]	0.00000 0.00000 0

Percentage of CPU used = 104.9

Kernel Usage

User = 15.58%

System = 17.66%

Idle = 66.74%

2. Disk Utilization

- a. 3Gb memory and 1 core
 - i. Sequential write
 - 1. Read (MiB/s) = 0.0
 - 2. Written (MiB/s) = 39.05

- ii. Combined random read write
 - 1. Read (MiB/s) = 21.07
 - 2. Written (MiB/s) = 14.04
- b. 6Gb memory and 2 core
 - i. Sequential write
 - 1. Read (MiB/s) = 0.0
 - 2. Written (MiB/s) = 53.96
 - ii. Combined random read write
 - 1. Read (MiB/s) = 25.07
 - 2. Written (MiB/s) = 15.60
- c. 9Gb memory and 4 core
 - i. Sequential write
 - 1. Read (MiB/s) = 0.0
 - 2. Written (MiB/s) = 53.72
 - ii. Combined random read write
 - 1. Read (MiB/s) = 23.96
 - 2. Written (MiB/s) = 15.97

Conclusion for CPU Utilization:

1. It was observed that in QEMU utilised the cpu a bit less than Docker. Moreover, with respect to kernel usage QEMU used more of user space and being idle when compared to Docker whereas, system was utilized more by Docker.

Conclusion for Disk Utilization:

1. It was observed that in QEMU sequential write, write operation's values decreased as the memory size increased along with cores. Further, combined random read write operation values had no significant changes on increasing the memory and core.
2. It was observed that in Docker sequential write, write operation's values increased little and remained constant later as the memory size along with cores. Further, combined random read write operation values had no significant changes on increasing the memory and core like QEMU.
3. Between QEMU and Docker, QEMU performed better than Docker file io operations making QEMU faster than Docker.

Docker File Implementation

I have created a docker image using following commands:

1. vim Dockerfile
2. Add the commands to be run. I have used From, Run and CMD. Please find the file in github
3. docker build -t <containerName>:<Tag> .
4. docker images
5. docker run <ContainerID>

6. Check if sudo was installed
7. The output is attached below.

```

Cloud computing — root@ac1f5b8e0d3d: / — docker run -it 028ef0ea8d40 — 160x57
~ — qemu-system-x86 + sudo ~ -- -bash ... ...9G --cpuset-cpus=0-3 hw1 ...cker run -it 028ef0ea8d4

(base) Juhis-MacBook-Pro:Cloud computing juhichecker$ docker build -t dockerfileexample:1.0 .
Sending build context to Docker daemon 8.448MB
Step 1/4 : FROM ubuntu:focal
--> 817578334b4d
Step 2/4 : RUN apt-get update
--> Using cache
--> 7aae956878f2
Step 3/4 : RUN apt-get -y install sudo
--> Using cache
--> df7c7f1c50c8
Step 4/4 : CMD bash
--> Using cache
--> 028ef0ea8d40
Successfully built 028ef0ea8d40
Successfully tagged dockerfileexample:1.0
(base) Juhis-MacBook-Pro:Cloud computing juhichecker$ docker images
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
dockerfileexample   1.0        028ef0ea8d40    3 minutes ago  113MB
hw1                latest     f6a19e22fafc    3 days ago   134MB
trial               latest     890524d489d37   3 days ago   72.8MB
ubuntu              kinetic-20220922 2793294e6a90    13 days ago  70.2MB
amd64/ubuntu        latest     216c552ea5ba   13 days ago  77.8MB
ubuntu              latest     216c552ea5ba   13 days ago  77.8MB
ubuntu              focal      817578334b4d   13 days ago  72.8MB
(base) Juhis-MacBook-Pro:Cloud computing juhichecker$ docker run 028ef0ea8d40
(base) Juhis-MacBook-Pro:Cloud computing juhichecker$ docker run -it 028ef0ea8d40
root@ac1f5b8e0d3d:/# sudo apt-get install sysbench
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  krb5-locales libai01 libasn1-8-heimdal libgssapi-krb5-2 libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal lib
  libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3 libkrb5support0 libldap-2.4-2 libldap-common libluajit-5.1-2 lib
  libpq5 libroken18-heimdal libss1-2 libss1-2-modules libss1-2-modules-db libsqlite3-0 libssl1.1 libwind0-heimdal mysql
Suggested packages:
  krb5-doc krb5-user libss1-2-modules-gssapi-mit | libss1-2-modules-gssapi-heimdal libss1-2-modules-ldap libss1-2-modules
The following NEW packages will be installed:
  krb5-locales libai01 libasn1-8-heimdal libgssapi-krb5-2 libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal lib
  libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3 libkrb5support0 libldap-2.4-2 libldap-common libluajit-5.1-2 lib
  libpq5 libroken18-heimdal libss1-2 libss1-2-modules libss1-2-modules-db libsqlite3-0 libssl1.1 libwind0-heimdal mysql
0 upgraded, 29 newly installed, 0 to remove and 2 not upgraded.
Need to get 5386 kB of archives.
After this operation, 21.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libsqlite3-0 amd64 3.31.1-4ubuntu0.4 [549 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libss1.1.1 amd64 1.1.1f-1ubuntu2.16 [1321 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 krb5-locales all 1.17-6ubuntu4.1 [11.4 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libkrb5support0 amd64 1.17-6ubuntu4.1 [30.9 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libk5crypto3 amd64 1.17-6ubuntu4.1 [79.9 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libkeyutils1 amd64 1.6-6ubuntu1.1 [10.3 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libkrb5-3 amd64 1.17-6ubuntu4.1 [330 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libgssapi-krb5-2 amd64 1.17-6ubuntu4.1 [121 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/main amd64 libai01 amd64 0.3.112-5 [7184 B]
Get:10 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libroken18-heimdal amd64 7.7.0+dfsg-1ubuntu1.1 [42.0 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libss1-8-heimdal amd64 7.7.0+dfsg-1ubuntu1.1 [181 kB]

```

Shell Scripts:

Account – checkerjuhi21

Repository Name – CloudComputing

Folder Name – HW1

GitHub link – <https://github.com/checkerjuhi21/CloudComputing.git>

Commit ID - f8396eb0d00cc06c571537cec2f4f43740a3f838

Commit id may change after I push this pdf into the github. Please refer to commit id in the text entry that I submit it on canvas.