

COEN 241 – Fall 2021

System vs. OS Virtualization

Abhay Tamilselvan (W1606871)

TABLE OF CONTENTS

DETAILED CONFIGURATIONS OF EXPERIMENTAL SETUP	2
STEPS TO ENABLE QEMU VM	3
STEPS TO ENABLE DOCKER CONTAINER	4
PROOF OF EXPERIMENT	5
SHELL SCRIPT	11
PERFORMANCE TOOLS AND ANALYSIS	12
CPU UTILIZATION	15
I/O UTILIZATION	18
VAGRANT FILE	21
DOCKER FILE	22

Detailed configurations of experimental setup



The experiments for the VM machine and the docker will be conducted on a MacOS with Intel processor having 16GB RAM and a quad core processor.

The Ubuntu Virtual Machine will be mounted on a QEMU image with 10GB hard disk space. The number of CPUs used is 1 and the RAM size is 2 GB.

The docker container will also have memory limit of 2 GB RAM so that the test environment is similar.

The tests has been conducted for both Ubuntu VM and container under 3 different scenarios -

CPU Tests

1. RAM = 2GB, threads = 1, cpu-max-prime = 20000
2. RAM = 2GB, threads = 1, cpu-max-prime = 5000000, max-time = 30s
3. RAM = 2GB, threads = 16, cpu-max-prime = 50000

File IO Tests

1. RAM = 2GB, threads = 16, file-total-size = 3GB, file-test-mode = RNDRW
2. RAM = 2GB, threads = 4, file-total-size = 2GB, file-test-mode = SEQRD
3. RAM = 2GB, threads = 8, file-total-size = 3GB, file-test-mode = SEQWR

Steps to enable a QEMU VM

1. As I am using a Mac with Intel chip, qemu can be installed using the command below.

```
brew install qemu
```

2. After installation, a QEMU image with a hard disk space of 10GB can be created using

```
sudo qemu-img create ubuntu.img 10G
```

3. After creating the image, we have to download the ubuntu ISO file (ubuntu-20.04.3-live-server-amd64) and then install the VM using the following command

```
sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ./  
ubuntu-20.04.3-live-server-amd64 -m 2G
```

QEMU arguments used in creating the VM

1. `-m` argument is used to assign the startup RAM size to the guest OS. I have used this to assign a RAM size of 1536MB.
2. `-boot` is used to boot the VM. `a`, `b` is for floppy disk, `c` is for hard disk and `d` is for cdrom. Hard disk boot is the default. We have used `d` here and considered the ubuntu ISO file as a cdrom.
3. `-hda` is used to use file as a hard disk image
4. `-cdrom` to use file as a cdrom image

Other useful arguments of QEMU have been tested on the terminal

1. `-smp` argument is used to simulate a SMP (shared memory processing) system with `n` CPUs. I created a guest OS with 4 CPUs using the below command.

```
sudo qemu-system-x86_64 -hda ubuntu.img -m 2G -smp 4
```
2. `-accel` argument is used to enable an accelerator. Different accelerators like kvm, xen, etc can be used. The default accelerator is tcg.
3. `-machine` can be used to select the emulated machine by name.
4. `-cpu` is used to choose among the different cpu models available.

Main steps to enable a Docker container

After installing docker, the following commands should be entered in the terminal to enable the docker image.

```
docker pull csminpp/ubuntu-sysbench  
docker run -it --memory="2g" csminpp/ubuntu-sysbench /bin/bash
```

`docker pull` is used to pull a docker image from the docker hub and store it on the local device.

`docker run` is used to run the docker image.

`-it` flag attaches an interactive tty in the container.

Now, the bash terminal of the docker container is opened and we can run our experiments on it based on our requirements.

Other docker commands that are frequently used to manage docker images and containers are as follows:

`docker images` - This command will display all the docker images on our local machine.

`docker ps` - This command will display all the docker containers that are currently running.

`docker rm` - This command is used to clean up the containers once we are done using it.

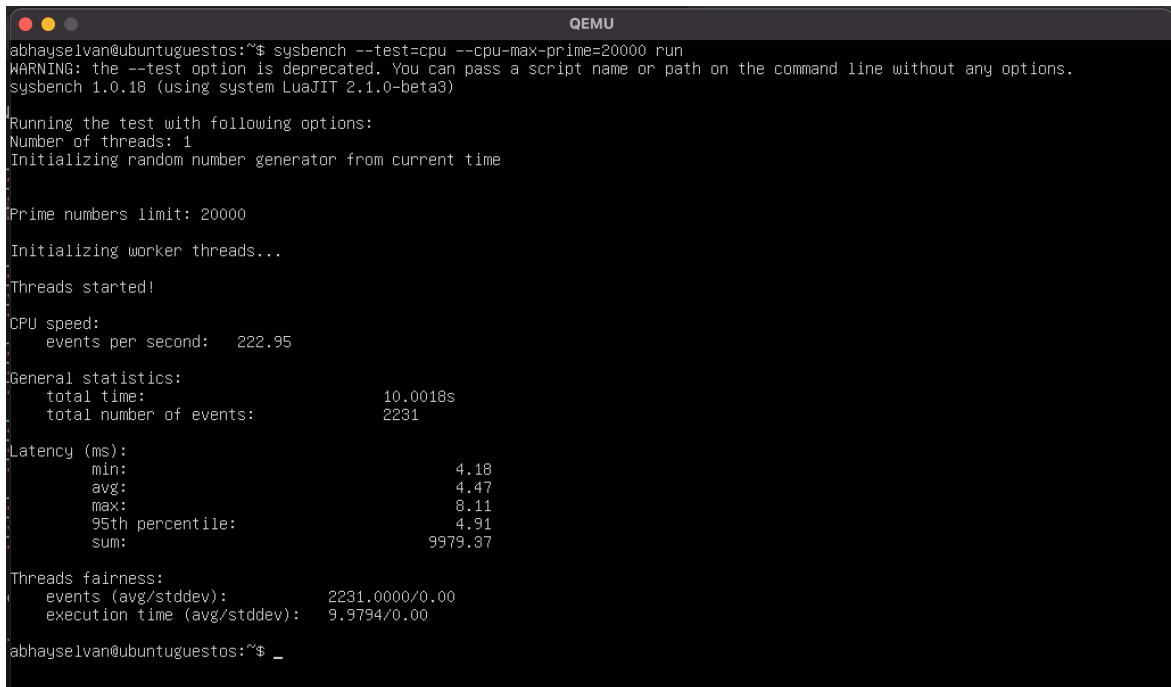
`docker container prune` - This command will remove all stopped containers.

Proof of Experiment

I have conducted three experiments for CPU test and three experiments for File IO test on both the virtual machine and the container. For each experiment, I have run the test 5 times each. The screenshots of each experiment is posted below.

CPU TESTS

Test 1: sysbench --test=cpu --cpu-max-prime=20000 run



```
QEMU
abhayselvan@ubuntuguestos:~$ sysbench --test=cpu --cpu-max-prime=20000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 20000

Initializing worker threads...

Threads started!

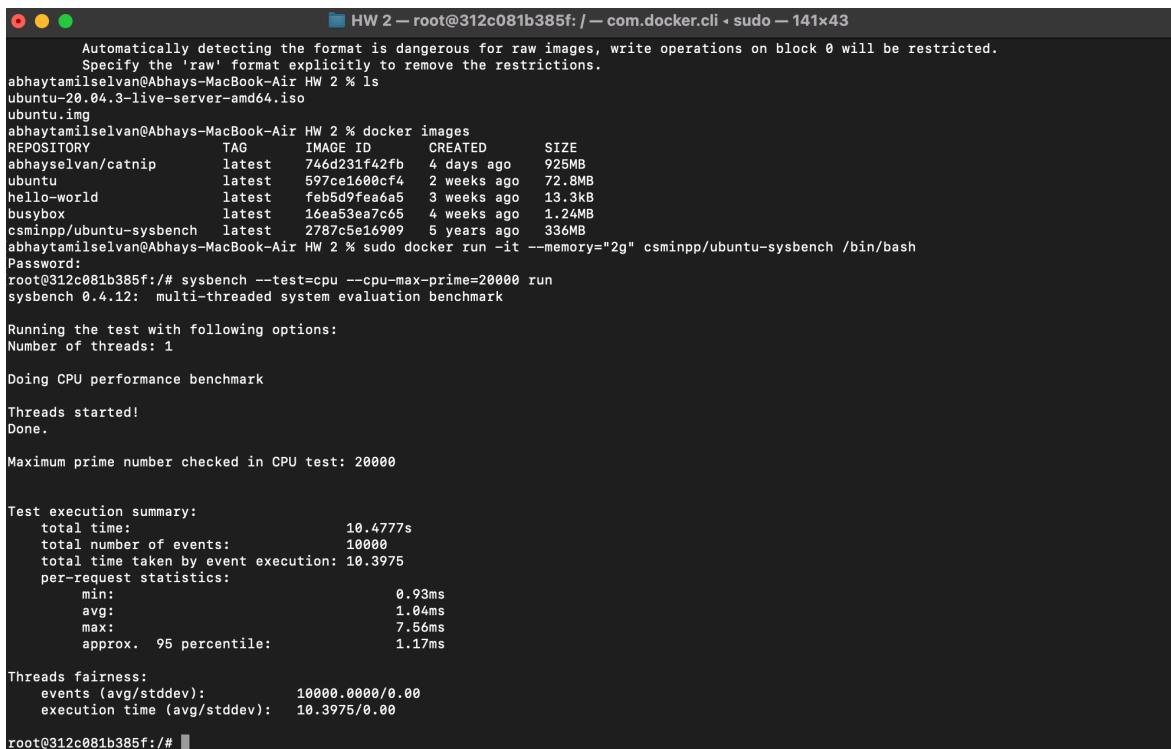
CPU speed:
events per second: 222.95

General statistics:
total time: 10.0018s
total number of events: 2231

Latency (ms):
min: 4.18
avg: 4.47
max: 8.11
95th percentile: 4.91
sum: 9979.37

Threads fairness:
events (avg/stddev): 2231.0000/0.00
execution time (avg/stddev): 9.9794/0.00

abhayselvan@ubuntuguestos:~$ _
```



```
HW 2 – root@312c081b385f:/ – com.docker.cli - sudo – 141x43
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
abhaytamilselvan@Abhays-MacBook-Air HW 2 % ls
ubuntu-20.04.3-live-server-amd64.iso
ubuntu.img
abhaytamilselvan@Abhays-MacBook-Air HW 2 % docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
abhayselvan/catnip    latest   746d231f42fb  4 days ago   925MB
ubuntu              latest   597ce1600cf4  2 weeks ago  72.8MB
hello-world         latest   feb5d9fead65  3 weeks ago  13.3KB
busybox             latest   16ea53ea7cc65 4 weeks ago  1.24MB
csmminpp/ubuntu-sysbench latest   2787c5e16989 5 years ago  336MB
abhaytamilselvan@Abhays-MacBook-Air HW 2 % sudo docker run -it --memory="2g" csmminpp/ubuntu-sysbench /bin/bash
Password:
root@312c081b385f:/# sysbench --test=cpu --cpu-max-prime=20000 run
sysbench 0.4.12: multi-threaded system evaluation benchmark

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

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000

Test execution summary:
total time: 10.4777s
total number of events: 10000
total time taken by event execution: 10.3975
per-request statistics:
min: 0.93ms
avg: 1.04ms
max: 7.56ms
approx. 95 percentile: 1.17ms

Threads fairness:
events (avg/stddev): 10000.0000/0.00
execution time (avg/stddev): 10.3975/0.00

root@312c081b385f:/#
```

Test 2: sysbench --test=cpu --cpu-max-prime=5000000 -max-time=30 run

```
QEMU
total number of events: 2095

Latency (ms):
min: 4.20
avg: 4.76
max: 7.07
95th percentile: 5.57
sum: 9966.06

Threads fairness:
events (avg/stddev): 2095.0000/0.00
execution time (avg/stddev): 9.9661/0.00

abbhayselvan@ubuntuguestos:~$ sysbench --test=cpu --cpu-max-prime=5000000 --max-time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --max-time is deprecated, use --time instead
sysbench 1.0.18 (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: 5000000
Initializing worker threads...

Threads started!

CPU speed:
events per second: 0.10

General statistics:
total time: 38.3083s
total number of events: 4

Latency (ms):
min: 9269.80
avg: 9573.31
max: 9919.36
95th percentile: 9977.52
sum: 38293.25

Threads fairness:
events (avg/stddev): 4.0000/0.00
execution time (avg/stddev): 38.2933/0.00
```

```
HW 2 — root@312c081b385f:/ — com.docker.cli • sudo — 141×43

Compiled-in tests:
fileio - File I/O test
cpu - CPU performance test
memory - Memory functions speed test
threads - Threads subsystem performance test
mutex - Mutex performance test
oltp - OLTP test

Commands: prepare run cleanup help version

See 'sysbench --test=<name> help' for a list of options for each test.

[root@312c081b385f:/# sysbench --test=cpu --cpu-max-prime=5000000 --max-time=30 run
sysbench 0.4.12: multi-threaded system evaluation benchmark

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

Doing CPU performance benchmark

Threads started!
Time limit exceeded, exiting...
Done.

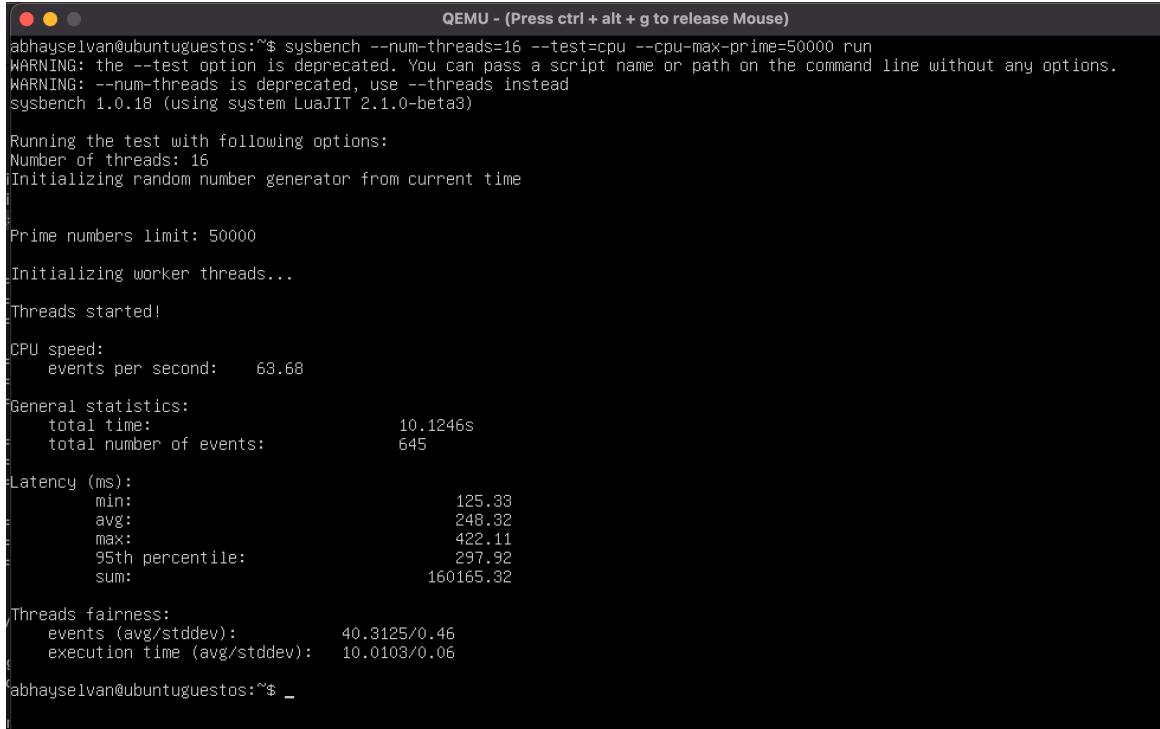
Maximum prime number checked in CPU test: 5000000

Test execution summary:
total time: 30.8546s
total number of events: 14
total time taken by event execution: 30.8537
per-request statistics:
min: 2160.53ms
avg: 2203.84ms
max: 2265.20ms
approx. 95 percentile: 2264.77ms

Threads fairness:
events (avg/stddev): 14.0000/0.00
execution time (avg/stddev): 30.8537/0.00

[root@312c081b385f:/# ]
```

```
Test 3: sysbench --num-threads=16 --test=cpu --cpu-max-prime=50000 run
```



```
QEMU - (Press ctrl + alt + g to release Mouse)
abhayselvan@ubuntuguestos:~$ sysbench --num-threads=16 --test=cpu --cpu-max-prime=50000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 50000

Initializing worker threads...

Threads started!

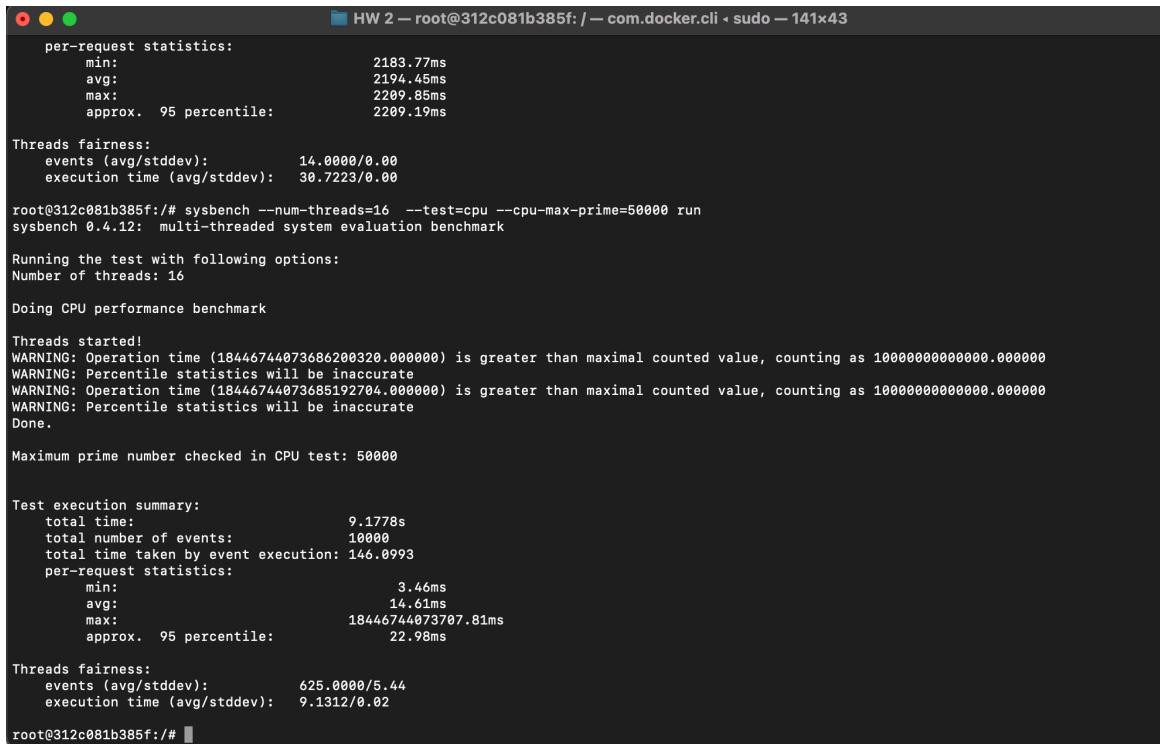
CPU speed:
events per second: 63.68

General statistics:
total time: 10.1246s
total number of events: 645

Latency (ms):
min: 125.33
avg: 248.32
max: 422.11
95th percentile: 297.92
sum: 160165.32

Threads fairness:
events (avg/stddev): 40.3125/0.46
execution time (avg/stddev): 10.0103/0.06

abhayselvan@ubuntuguestos:~$ _
```



```
HW 2 — root@312c081b385f:/ — com.docker.cli • sudo — 141x43
per-request statistics:
min: 2183.77ms
avg: 2194.45ms
max: 2209.85ms
approx. 95 percentile: 2209.19ms

Threads fairness:
events (avg/stddev): 14.0000/0.00
execution time (avg/stddev): 30.7223/0.00

root@312c081b385f:/# sysbench --num-threads=16 --test=cpu --cpu-max-prime=50000 run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Doing CPU performance benchmark

Threads started!
WARNING: Operation time (18446744073686200320.00000) is greater than maximal counted value, counting as 10000000000000.00000
WARNING: Percentile statistics will be inaccurate
WARNING: Operation time (184467440736865192704.00000) is greater than maximal counted value, counting as 10000000000000.00000
WARNING: Percentile statistics will be inaccurate
Done.

Maximum prime number checked in CPU test: 50000

Test execution summary:
total time: 9.1778s
total number of events: 10000
total time taken by event execution: 146.0993
per-request statistics:
min: 3.46ms
avg: 14.61ms
max: 18446744073707.81ms
approx. 95 percentile: 22.98ms

Threads fairness:
events (avg/stddev): 625.0000/5.44
execution time (avg/stddev): 9.1312/0.02

root@312c081b385f:/#
```

FILEIO TESTS

Test 1: sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw run

```
● ● ● QEMU - (Press ctrl + alt + g to release Mouse)
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:          759.91
  writes/s:         506.32
  fsyncs/s:        1811.54

Throughput:
  read, MiB/s:      11.87
  written, MiB/s:   7.91

General statistics:
  total time:       10.3838s
  total number of events: 29920

Latency (ms):
  min:              0.01
  avg:              5.31
  max:              62.60
  95th percentile: 18.61
  sum:             158933.26

Threads fairness:
  events (avg/stddev):    1870.0000/45.22
  execution time (avg/stddev): 9.9833/0.01
```

```
● ● ● HW 2 – root@7fefb9deb3a4:/ – com.docker.cli - docker run -it --memory=2g csmnpp/ubuntu-sysbench /bin/bash – 141x43
root@7fefb9deb3a4:/# sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
root@7fefb9deb3a4:/# sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 24Mb each
3Gb total file size
Block size 16Kb
Number of random requests for random IO: 10000
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
Threads started!
Done.

Operations performed: 6060 Read, 4031 Write, 12801 Other = 22892 Total
Read 94.688Mb Written 62.984Mb Total transferred 157.67Mb (120.49Mb/sec)
7711.56 Requests/sec executed

Test execution summary:
  total time:           1.3086s
  total number of events: 10091
  total time taken by event execution: 3.5494
  per-request statistics:
    min:                 0.02ms
    avg:                 0.35ms
    max:                 8.95ms
    approx. 95 percentile: 1.28ms

Threads fairness:
  events (avg/stddev):    630.6875/47.84
  execution time (avg/stddev): 0.2218/0.01

root@7fefb9deb3a4:/#
```

Test 2: sysbench --num-threads=4 --test=fileio --file-total-size=2G --file-test-mode=seqrd run

```
QEMU - (Press ctrl + alt + g to release Mouse)
abhayselvan@ubuntuguestos:~$ sysbench --test=fileio --num-threads=4 --file-total-size=2G --file-test-mode=seqrd run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line 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: 4
Initializing random number generator from current time

Extra file open flags: (none)
128 files, 16MiB each
2GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential read test
Initializing worker threads...

Threads started!

File operations:
  reads/s:           8246.99
  writes/s:          0.00
  fsyncs/s:          0.00

Throughput:
  read, MiB/s:      128.86
  written, MiB/s:   0.00

General statistics:
  total time:        10.0027s
  total number of events: 82519

Latency (ms):
  min:               0.01
  avg:               0.30
  max:               46.00
  95th percentile:  0.50
  sum:              24513.32

Threads fairness:
  events (avg/stddev): 20629.7500/201.06
  execution time (avg/stddev): 6.1283/0.04

abhayselvan@ubuntuguestos:~$
```

```
HW 2 – root@7fefb9deb3a4:/ – com.docker.cli - docker run -it --memory=2g csmnpp/ubuntu-sysbench /bin/bash – 141x43

Removing test files...
root@7fefb9deb3a4:/# sysbench --num-threads=4 --test=fileio --file-total-size=2G --file-test-mode=seqrd prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark

128 files, 16384Kb each, 2048Mb total
Creating files for the test...
root@7fefb9deb3a4:/# sysbench --num-threads=4 --test=fileio --file-total-size=2G --file-test-mode=seqrd run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4

Extra file open flags: 0
128 files, 16Mb each
2Gb total file size
Block size 16Kb
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential read test
Threads started!
Done.

Operations performed: 131072 Read, 0 Write, 0 Other = 131072 Total
Read 2Gb Written 0b Total transferred 2Gb (543.49Mb/sec)
34783.33 Requests/sec executed

Test execution summary:
  total time:           3.7682s
  total number of events: 131072
  total time taken by event execution: 13.7845
  per-request statistics:
    min:                 0.02ms
    avg:                 0.11ms
    max:                864.18ms
    approx. 95 percentile: 0.24ms

Threads fairness:
  events (avg/stddev): 32768.0000/2257.53
  execution time (avg/stddev): 3.4461/0.01

root@7fefb9deb3a4:/#
```

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

```
QEMU - (Press ctrl + alt + g to release Mouse)
abhayselvan@ubuntuguestos:~$ sysbench --num-threads=8 --test=fileio --file-total-size=3G --file-test-mode=seqwr run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 8
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:         1819.11
  fsyncs/s:         2428.48

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   28.42

General statistics:
  total time:        10.22223s
  total number of events: 42409

Latency (ms):
  min:                0.08
  avg:                1.85
  max:                90.66
  95th percentile:    7.70
  sum:               78333.57

Threads fairness:
  events (avg/stddev):   5301.1250/123.96
  execution time (avg/stddev): 9.7917/0.04

abhayselvan@ubuntuguestos:~$
```

```
HW 2 — root@7fefb9deb3a4:/ — com.docker.cli • docker run -it --memory=2g csmnpp/ubuntu-sysbench /bin/bash — 141x43

FATAL: asynchronous I/O mode is unsupported on this platform.
root@7fefb9deb3a4:/# sysbench --num-threads=8 --test=fileio --file-total-size=3G --file-test-mode=seqwr prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
root@7fefb9deb3a4:/# sysbench --num-threads=8 --test=fileio --file-total-size=3G --file-test-mode=seqwr run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 8

Extra file open flags: 0
128 files, 24Mb each
3Gb total file size
Block size 16Kb
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
Threads started!
Done.

Operations performed: 0 Read, 196608 Write, 128 Other = 196736 Total
Read 9b Written 3Gb Total transferred 3Gb (496.54Mb/sec)
31778.78 Requests/sec executed

Test execution summary:
  total time:          6.1868s
  total number of events: 196608
  total time taken by event execution: 31.2968
  per-request statistics:
    min:                 0.03ms
    avg:                 0.16ms
    max:                 159.16ms
    approx. 95 percentile: 0.09ms

Threads fairness:
  events (avg/stddev): 24576.0000/646.87
  execution time (avg/stddev): 3.9121/0.07

root@7fefb9deb3a4:/#
```

Shell Scripts

cputests.sh

```
#!/bin/bash
#
primes=("20000" "5000000" "50000")
times=("0" "30" "0")
threads=("1" "1" "16")
testruns=5
testcases=3

for ((i=0; i<$testcases;i++));
do
    for (( j=1; j <=$testruns; j++ ))
    do
        sysbench --num-threads=${threads[$i]} --test(cpu) --cpu-max-prime=${primes[$i]} --max-time=${times[$i]} run
    done
done
```

fileiotests.sh

```
#!/bin/bash
#
commands=("prepare" "run" "cleanup")
modes=("rndrw" "seqrd" "seqwr")
filesizes=("3G" "2G" "3G")
threads=("16" "4" "8")
testruns=5
testcases=3
teststages=3

for ((i=0; i<$testcases;i++));
do
    for (( j=0; j <=$testruns; j++ ))
    do
        for (( k=0; k<$teststages; k++ ))
        do
            sysbench --num-threads=${threads[$i]} --test(fileio) --file-total-size=${filesizes[$i]} --file-test-mode=${modes[$i]} ${commands[$k]}
        done
    done
done
```

Performance tools and analysis

Events per second has been chosen as unit of measurement for both CPU tests and File IO tests because two different versions of sysbench have been used for docker and QEMU VM respectively.

CPU Tests

For CPU tests, I have altered three values to generate comprehensive test results - CPU-max-prime value, max-time and num-threads.

cpu-max-prime=20000

	Average	Min	Max	Std
Docker	954.88	952.3	957.5	1.739
QEMU VM	215.71	209.26	222.95	5.256

cpu-max-prime=5000000 -max-time=30

	Average	Min	Max	Std
Docker	0.455	0.453	0.456	0.009
QEMU VM	0.104	0.102	0.105	0.013

cpu-max-prime=50000 num-threads=16

	Average	Min	Max	Std
Docker	1039.26	947.66	1085.39	55.68
QEMU VM	63.27	61.75	64.03	0.815

File IO Tests

While conducting File IO tests, it is important to use the flag `--file-extra-flags=direct` to avoid caching of files in the memory. We should also allocate `--file-total-size` greater than that of the total memory just to make sure that caching doesn't happen despite our best efforts.

```
num-threads=16 --file-total-size=3G --file-test-mode=rndrw
```

	Average	Min	Max	Std
Docker	8057.95	7711.56	8270.26	192.12
QEMU VM	2878.5	2756.63	2946.41	74.29

```
num-threads=4 --file-total-size=2G --file-test-mode=seqrd
```

	Average	Min	Max	Std
Docker	40517.66	34783.33	45787.02	3767.68
QEMU VM	10011.33	9208.67	10526.14	464.69

```
num-threads=8 --file-total-size=3G --file-test-mode=seqwr
```

	Average	Min	Max	Std
Docker	32044.89	31556.37	32474.51	462.03
QEMU VM	4019.25	3686.14	4212.16	203.13

Inference:

- Docker container is faster than QEMU VM in all three CPU tests
- Docker container is faster than QEMU VM in all three File IO tests
- Events per second decreases when max prime value increases for both docker container and QEMU VM.
- Increasing the number of threads results in increase in performance of the docker container for CPU tests but the QEMU VM's performance doesn't increase correspondingly.
- The sequential file IO operations have better performance compared to random file IO operations

Analysis:

Containers faster than VMs

Containers have better performance than Virtual Machines which is expected because they only provide OS virtualization and don't interact directly with the hardware. Even installation of containers is much faster than Virtual Machines making it easier to deploy and share across platforms. This is because a guest OS needs to be installed for a VM whereas the host OS itself is used by containers.

Increasing number of threads increases performance

When we increase the number of threads, more instructions are processed by the CPU concurrently. But this doesn't mean that randomly increasing threads would result in a better performance. It depends on the number of cores that a machine has and the instructions themselves being non-sequential.

Sequential I/O faster than random I/O

The sequential I/O is faster than random I/O because of the way disk drives work. To access a file, the disk head needs to seek the location of the file. If a file is stored sequentially, then time is saved on the seek operation while the read/write operation take the same time for both random and sequential operations.

Events per second decreases with complexity of task to be performed

An event is the task that the system needs to perform. So, if it's a very simple task, then it is completed quickly. So more numbers of events can be completed per second if it's an easy task. Thus, it can be easily inferred that events per second decreases with increase in the complexity of a task.

CPU Utilization

CPU usage has been measured on the docker container as well as the host OS during idle time and sysbench cpu benchmark.

Container

We need to create a new terminal for the same docker container so that we can test the CPU usage while the sysbench tests are running. We can do that by getting the container id of the running container and using the following command:

```
docker exec -it <container id> /bin/bash
```

top command has been used to check the CPU usage on the docker container.

In %cpu(s),

us - stands for user level CPU usage

sy - stands for kernel level CPU usage

id - stands for idle CPU usage (% of available CPU that is idle)

The PIDs of the processes and their usage are also displayed below.

When docker is idle

```
...e: / – com.docker.cli - docker run -it csmnpp/ubuntu-sysbench /bin/bash ...aafb38e: / – com.docker.cli - docker exec -it 92a0faafb38e /bin/bash +  
top - 00:26:52 up 17:09, 0 users, load average: 0.25, 0.32, 0.46  
Tasks: 3 total, 1 running, 2 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 0.3 us, 1.3 sy, 0.0 ni, 98.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
KiB Mem: 2035400 total, 936140 used, 1099260 free, 42924 buffers  
KiB Swap: 1048572 total, 76488 used, 972084 free. 483276 cached Mem  
  
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND  
1 root 20 0 18184 2452 1964 S 0.0 0.1 0:00.13 bash  
18 root 20 0 18184 620 128 S 0.0 0.0 0:00.05 bash  
276 root 20 0 19860 2344 2028 R 0.0 0.1 0:00.08 top
```

When docker is running sysbench cpu benchmark test

```
...e: / – com.docker.cli - docker run -it csmnpp/ubuntu-sysbench /bin/bash ...aafb38e: / – com.docker.cli - docker exec -it 92a0faafb38e /bin/bash +  
top - 00:27:10 up 17:09, 0 users, load average: 3.18, 0.96, 0.66  
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 99.7 us, 0.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
KiB Mem: 2035400 total, 936636 used, 1098764 free, 42940 buffers  
KiB Swap: 1048572 total, 76488 used, 972084 free. 483280 cached Mem  
  
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND  
280 root 20 0 20172 1624 1336 S 397.5 0.1 0:49.43 sysbench  
1 root 20 0 18184 2472 1984 S 0.0 0.1 0:00.13 bash  
18 root 20 0 18184 620 128 S 0.0 0.0 0:00.05 bash  
276 root 20 0 19860 2344 2028 R 0.0 0.1 0:00.08 top
```

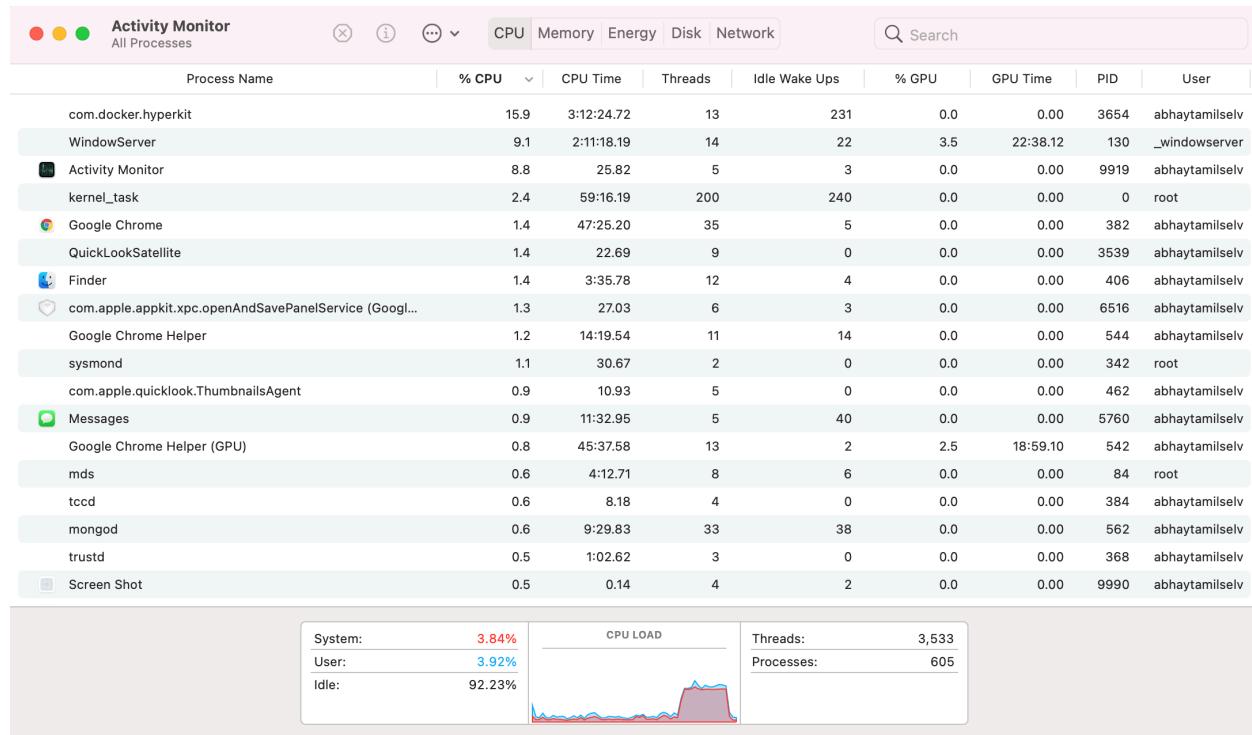
	Sysbench not running	Sysbench CPU test running
User level CPU usage %	0.3	99.7
Kernel level CPU usage %	1.3	0.3
Idle CPU %	98.4	0.0

As can be seen from the table above, when sys bench CPU benchmark test is being run, the user level CPU usage is close to 100%. The kernel level is barely used.

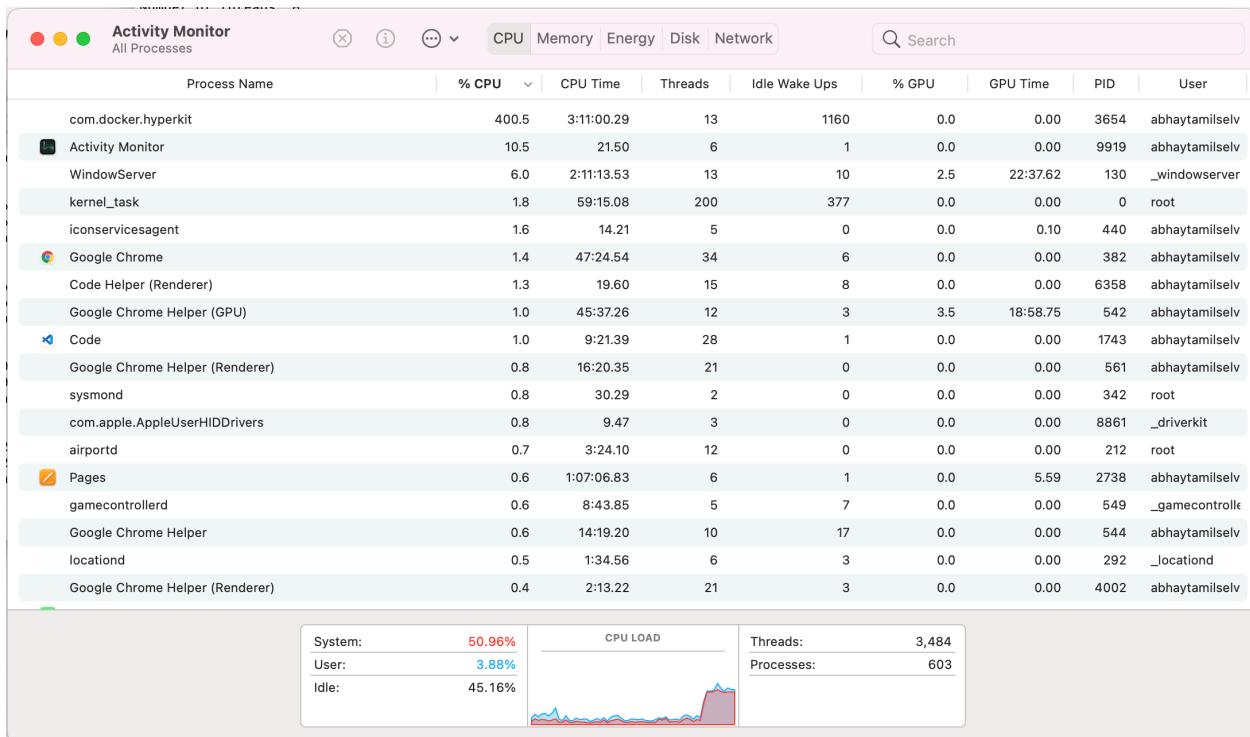
Host OS

When we look at the CPU usage in the host OS, the following CPU usage was observed.

When sysbench is not running on docker



When sysbench is running CPU benchmark test on docker



	Sysbench not running	Sysbench CPU test running
User level CPU usage %	3.92	3.88
Kernel level CPU usage %	3.84	50.96
Idle CPU %	92.23	45.16

In the host OS, the kernel level CPU usage jumps from 3.84% to 50.96% when we run sysbench CPU benchmark test. The docker container when idle has a CPU usage of 15.9% which jumps to 400% when sysbench is run. One difference to note here is that inside the docker, user level CPU usage is accessed for the process whereas in the host OS, the kernel level CPU usage is used. **This shows that the container uses the kernel of the host OS.**

I/O Utilization

I/O utilization of a VM or container can be measured using the `iostat` command. To make use of this command, we need to install `sysstat` which can be done using the command:

```
sudo apt install sysstat
```

After installing `sysstat`, we need to use the command,

```
iostat -dxzm 1
```

The `-d` flag is used to show disk utilization only. `-x` is used to show extended columns, `-z` is used to skip devices with zero metrics. The `1` here denotes that the disk utilization report would be displayed on the command line every 1 second. After running the above commands, we can run the `sysbench` `fileIO` test on a different terminal of the same container to see how disk utilization changes when we run the test.

The screenshots for the disk utilization during the three phases of the `sysbench` `fileio` test are as follows

During prepare stage:

Device	w_await	svctm	%util	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrrq-sz	avgqu-sz	await	r_await		
vda	4.86	0.30	0.28	0.06	1.64	2.07	4.58	0.16	1.58	535.59	0.03	3.71	1.16		
vda	0.92	1.33	1.68	0.00	32.00	0.00	12.00	0.00	0.17	29.33	0.02	0.92	0.00		
vda	1.00	1.38	1.78	0.00	34.65	0.00	12.87	0.00	0.19	29.54	0.02	1.00	0.00		
vda	0.79	1.21	2.98	0.00	69.00	0.00	24.00	0.00	0.36	31.00	0.03	0.79	0.00		
vda	5.10	0.38	16.88	0.00	73.00	0.00	445.00	0.00	400.24	1842.00	2.29	5.10	0.00		
vda	10.20	0.74	44.16	0.00	553.47	7.92	590.10	0.06	635.95	2178.12	6.22	10.15	6.25		

During run stage:

```

root@2e7706c297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=100 --file-test-mode=l
rndrw prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark
128 files, 81920kb each, 10240Mb total
Creating files for the test...
root@2e7706c297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=100 --file-test-mode=l
rndrw prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark
Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 80Mb each
100b total file size
Block size 16kb
Number of random requests for random IO: 10000
Read/Write ratio for combined random IO test: 1.50
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
Threads started!
[]

Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  138.00   0.00  880.00   0.00  889.04 2869.05   5.27   5.84   0.00
               5.84    0.51  44.70
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  102.00   0.00  815.00   0.00  753.62 1893.77   8.17   9.82   0.00
               9.82    0.64  51.90
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  142.00   0.00  1073.00   0.00  1004.00 1917.82   6.28   5.81   0.00
               5.81    0.38  48.66
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  125.00   2.00  961.00   0.01  888.50 1872.56   6.62   6.78   3.50
               6.79    0.48  46.10
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  149.00   0.00  1069.00   0.00  1003.04 1921.64   6.32   5.87   0.00
               5.87    0.38  48.50
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  134.00   0.00  1023.00   0.00  958.04 1901.94   5.87   5.65   0.00
               5.65    0.41  42.00
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  73.00   0.00  519.00   0.00  528.31 2084.75   4.90   9.38   0.00
               9.38    0.51  26.70
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  137.00 3485.00   24.77  16.28  16.84   9.36   1.88   4.25
               0.94    0.13  67.00
Device:          rrqm/s   wrqm/s    r/s    w/s   rMB/s   wMB/s avgqrq-sz avgqu-sz   await r_await
w_await: svctm   Kutil    0.00     0.00  568.00 1307.00   31.08  28.31  15.13  14.79   1.77   4.23
               0.96    0.12  99.90

```

```

root@2e7f86e297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=10G --file-test-mode=1
rndrw prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark
128 files, 81920Kb each, 10240Mb total
Creating files for the test...
root@2e7f86e297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=10G --file-test-mode=1
rndrw run
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 8096 each
100% total file size
Block size: 64Kb
Number of random requests for random IO: 10000
Read/Write ratio for combined random IO test: 1.50
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
Threads started!
Done.

Operations performed: 6019 Read, 4002 Write, 12803 Other = 22824 Total
Read 94.847M Written 62.531M Total transferred 156.88Mb (54.602Mb/sec)
3494.51 Requests/sec executed

Test execution summary:
total time: 2.8676s
total number of events: 10021
total time taken by event execution: 26.8478
per-request statistics:
    min:          0.02ms
    avg:         2.58ms
    max:        16.24ms
    approx. 95 percentile: 6.33ms

Threads fairness:
events (avg/stddev): 626.3125/26.28
execution time (avg/stddev): 1.6154/0.02
root@2e7f86e297fb:#
```

During cleanup:

```

root@2e7786e297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=100 --file-test-mode=1
rndr prepare
sysbench 0.4.12: multi-threaded system evaluation benchmark

128 files, 61920Kb each, 10240Mb total
Creating files for the test...
root@2e7786e297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=100 --file-test-mode=1
rndr cleanup
sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 60Mb each
10Gb total file size
Block size 16kB
Number of random requests for random IO: 10000
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling sync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Threads started!
Done.

Operations performed: 6819 Read, 4002 Write, 12803 Other = 22824 Total
Read 94.047Mb Written 62.531Mb Total transferred 156.58Mb (54.602Mb/sec)
349.51 Requests/sec executed

Test execution summary:
total time: 2.8676s
total number of events: 10021
total time taken by event execution: 25.8470
per-request statistics:
    min: 0.02ms
    avg: 2.38ms
    max: 16.24ms
    approx. 95 percentile: 6.33ms

Threads fairness:
events (avg/stddev): 626.3125/26.20
execution time (avg/stddev): 1.6154/0.02

root@2e7786e297fb:/# sysbench --num-threads=16 --test=fileio --file-total-size=100 --file-test-mode=1
rndr cleanup
sysbench 0.4.12: multi-threaded system evaluation benchmark

Removing test files...
root@2e7786e297fb:/# ^C
root@2e7786e297fb:/#

```

I/O latency is the time taken from request to completion of I/O request. It is measured using the await value in the iostat report. Await displays the average time (in milliseconds) for I/O requests issued to the device to be served.

I/O throughput is the data transfer speed in megabytes per second. It is measured using the wMBs value in the iostat report. wMBs is the number of megabytes written to the device per second.

I/O disk utilization is the percentage of CPU time during which I/O requests were issued to the device. We can measure this using the %util value.

	Prepare stage	Run stage	Idle/cleanup stage
I/O Latency	~6 milliseconds	~6 milliseconds	N/A
I/O Throughput	~900 MBPS	~900MBPS	N/A
I/O Disk Utilization	~50%	~100%	0 - 1%

From the table, the I/O latency for the device is around 6 milliseconds. The I/O throughput is around 900 MBPS. The disk utilization during prepare stage is around half of the total bandwidth available. During run stage, it uses close to 100% of the bandwidth (device saturation occurs this value is close to 100%). During cleanup or idle stage, there is close to zero disk utilization.

Vagrant File for Virtual Box

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-20.04"

  # Provider Settings
  config.vm.provider "virtualbox" do |vb|
    vb.memory = 2048
    vb.cpus = 4
  end
  # Network Settings
  # config.vm.network "forwarded_port", guest: 80, host: 8080
  # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip:
  "127.0.0.1"
  # config.vm.network "private_network", ip: "192.168.33.10"
  # config.vm.network "public_network"

  # Folder Settings
  config.vm.synced_folder ".", "/vagrant_data"

  # Provision Settings
  config.vm.provision "shell", path: "vagrantbootstrap.sh"

end
```

vagrantbootstrap.sh

```
#!/bin/bash
#
CPUTEST_PATH="/vagrant_data/cputests.sh"
FILEIOTEST_PATH="/vagrant_data/fileiotests.sh"

sudo apt-get update
sudo apt-get install -y sysbench
chmod +x /vagrant_data/cputests.sh
chmod +x /vagrant_data/fileiotests.sh

"$CPUTEST_PATH"
"$FILEIOTEST_PATH"
```

Dockerfile

```
FROM csminpp/ubuntu-sysbench

COPY dockerbootstrap.sh /dockerbootstrap.sh
COPY cputests.sh /cputests.sh
COPY fileiotests.sh /fileiotests.sh

RUN chmod +x dockerbootstrap.sh
RUN chmod +x cputests.sh
RUN chmod +x fileiotests.sh

ENTRYPOINT bash dockerbootstrap.sh
```

dockerbootstrap.sh

```
#!/bin/bash
#
CPUTEST_PATH="/cputests.sh"
FILEIOTEST_PATH="/fileiotests.sh"
"$CPUTEST_PATH"
"$FILEIOTEST_PATH"
```