Assignment 1– COEN 241 (Cloud Computing)

Submitted by: Arpita Verma
W1632653

System vs. OS Virtualization Report

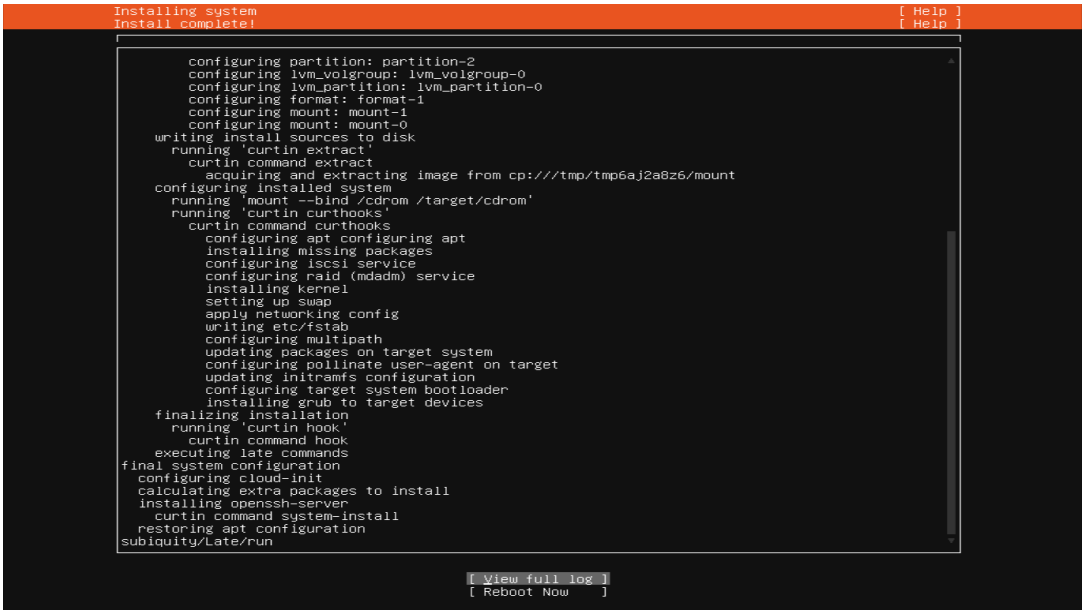
Table of Contents

Environment Setup

Host System Configuration Details -

The host system used used to perform the execution of experiment has following configuration -

- CPU - x86_64 Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz with 4 cores
- Memory - 8 GB RAM
- Disk Space -
    - 500 MB - EFI System
    - 8 GB - Linux Swap
    - 115 GB - Linux Filesystem
- OS - Ubuntu 20.04.5 LTS x86_64



QEMU Ubuntu Virtual Machine Configuration Details -

The Ubuntu VM is created with QEMU by allocating 20 GB of disk space (qcow2 file format), 2 GB of RAM, and 2 CPU cores.

Docker Ubuntu Container Configuration Details -

The Ubuntu container in docker also allocated the same configuration as Ubuntu VM having 2 GB of RAM and 2 CPU cores.

System Virtualization Setup

QEMU Setup -

1. Install the QEMU on host

```
arpita@system:~$ sudo apt-get install qemu
[sudo] password for arpita:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gconf2 libflashrom1 libftdi1-2 libllvm15
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  qemu
0 upgraded, 1 newly installed, 0 to remove and 27 not upgraded.
Need to get 15.3 kB of archives.
After this operation, 139 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 qemu amd64 1:6.2+dfsg-2ubuntu6.6 [15.3 kB]
Fetched 15.3 kB in 0s (35.5 kB/s)
Selecting previously unselected package qemu.
(Reading database ... 239820 files and directories currently installed.)
Preparing to unpack .../qemu_1%3a6.2+dfsg-2ubuntu6.6_amd64.deb ...
Unpacking qemu (1:6.2+dfsg-2ubuntu6.6) ...
Setting up qemu (1:6.2+dfsg-2ubuntu6.6) ...
arpita@system:~$
```

2. Download ubuntu server image using following command -
   $ wget https://releases.ubuntu.com/focal/ubuntu-20.04.5-live-server-amd64.iso

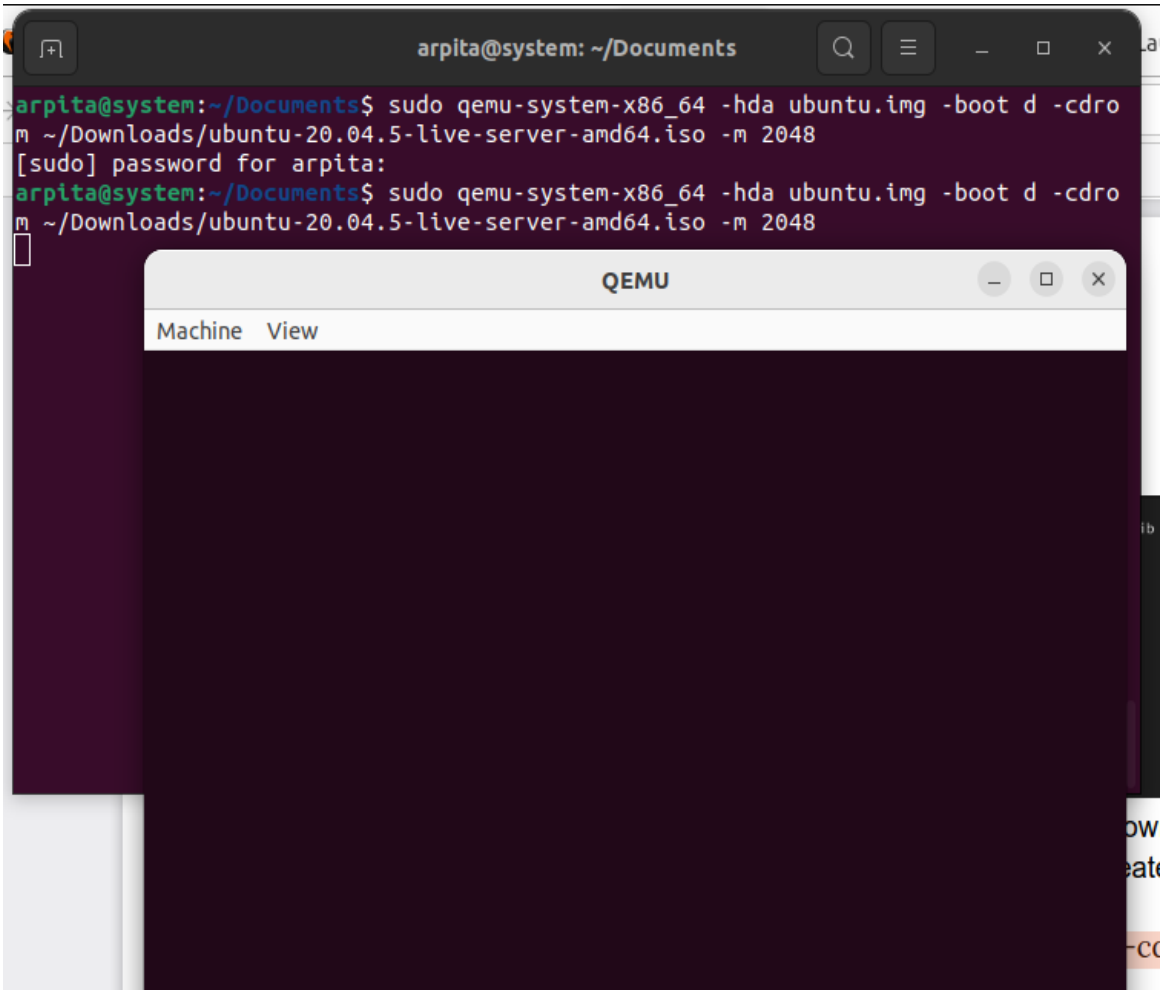
```
arpita-server login: arpita
Password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-125-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed 01 Feb 2023 12:19:56 AM UTC

  System load:           0.9
  Usage of /:            41.0% of 9.75GB
  Memory usage:          9%
  Swap usage:            0%
  Processes:             110
  Users logged in:       0
  IPv4 address for ens3: 10.0.2.15
  IPv6 address for ens3: fec0::5054:ff:fe12:3456

0 updates can be applied immediately.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

3. Create QEMU image of ubuntu in qcow2 file format using following command -
   $ sudo qemu-img create ubuntu.img 20G -f qcow2

```
arpita@system:~$ sudo qemu-img create ubuntu.img 20G -f qcow2
Formatting 'ubuntu.img', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=21474836480 lazy_refcounts=off refcount_bits=
16
```

4. After creating image, install the the Ubuntu VM using iso downloaded in 2nd step i.e. ubuntu-20.04.5-live-server-amd64.iso in the disk image created in 3rd step i.e. ubuntu.qcow2 using following command -

   $ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ubuntu-20.04.5-live-server-amd64.iso -m 2048



In the above command the flags/option used have following meanings -

- -hda : use file as a hard disk
- -boot : this option specifies the boot order, for x86 architecture these drive letters are - a, b (for floppy drives), c (first hard-disk), d (first CD-ROM), n-p (Etherboot from network adapter 1-4). Hard-disk boot is the default option.
- -cdrom : this option specifies the .iso file to be used as a base for image we are creating
- -m : this option sets guest OS's startup RAM to specified value i.e. 2048 MB

Sysbench Setup on QEMU VM -

1. Once VM is launched after finishing the above QEMU setup steps, we are now ready to install sysbench on the Ubuntu VM using following command -

$ curl -s
https://packagecloud.io/install/repositories/akopytov/sysbench/script.deb.sh
| sudo bash
$ apt -y install sysbench

2. Check the version of sysbench installed using following command -
   $ sysbench –
   version
   Command Output
   -

```
arpita@arpita-server:~$ sysbench --version
sysbench 1.0.20
arpita@arpita-server:~$
```

Going In-Depth of QEMU

We can launch the Ubuntu OS from above created ubuntu.img by providing a lot of options. For example we can provide options to give it memory, cpu, network, block device, accelerators etc.

Following are some of the examples of launching Ubuntu VM using extra options -

1. We can start Ubuntu VM by using -accel option to accelerate the machine using paravirtualized hypervisors such as kvm, xen, tcg (default) inside them. For example below command uses ‘kvm’ accelerator to start -

   $ sudo qemu-system-x86_64 -hda ubuntu1.qcow2 -accel kvm -boot c -m 2048

2. We can use -smp option to mention the number of cores the Guest OS can use and -cpu to provide an option for choosing from supported CPUs by QEMU. For example, below command uses 2 cores and for cpu type ‘host’ option which emulates host OS’s cpu -

   $ sudo qemu-system-x86_64 -hda ubuntu1.qcow2 -accel kvm -boot c -m 2048 -smp 2 -cpu host

3. We can use the -drive option to set the drive for guest os to use. For example, below command sets the ubuntu.img drive and uses virt-io interface -

   $ sudo qemu-system-x86_64 -drive file=ubuntu1.img,if=virtio -accel kvm -boot c -m 2048 -smp 2 -cpu host

4. We can use the '-netdev user' option to configure the user mode host network backend which requires no administrator privilege to run. Below is command uses the '-netdev user' -

   $ sudo qemu-system-x86_64 -drive file=ubuntu1.img,if=virtio -accel kvm -boot c -m 2048 -smp 2 -cpu host -netdev user,id=net0

5. Combinedly, below is the more advanced command using various options and their values to launch the guest operating system -

```
$ sudo qemu-system-x86_64 -accel kvm \
        -cpu host \
        -m 2048 \
        -smp 2 \
        -hda ubuntu.img \
        -boot c \
        -device virtio-net,netdev=vm \
        -netdev  user,id=vm,hostfwd=tcp:127.0.0.1:9003-:22
```

Operating System (OS) Virtualization Setup

Docker Setup -

1. Install the Docker on host

```
arpita@system:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

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

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

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

arpita@system:~$
arpita@system:~$ docker --version
Docker version 20.10.23, build 7155243
arpita@system:~$
```

Sysbench Setup in Docker Container -

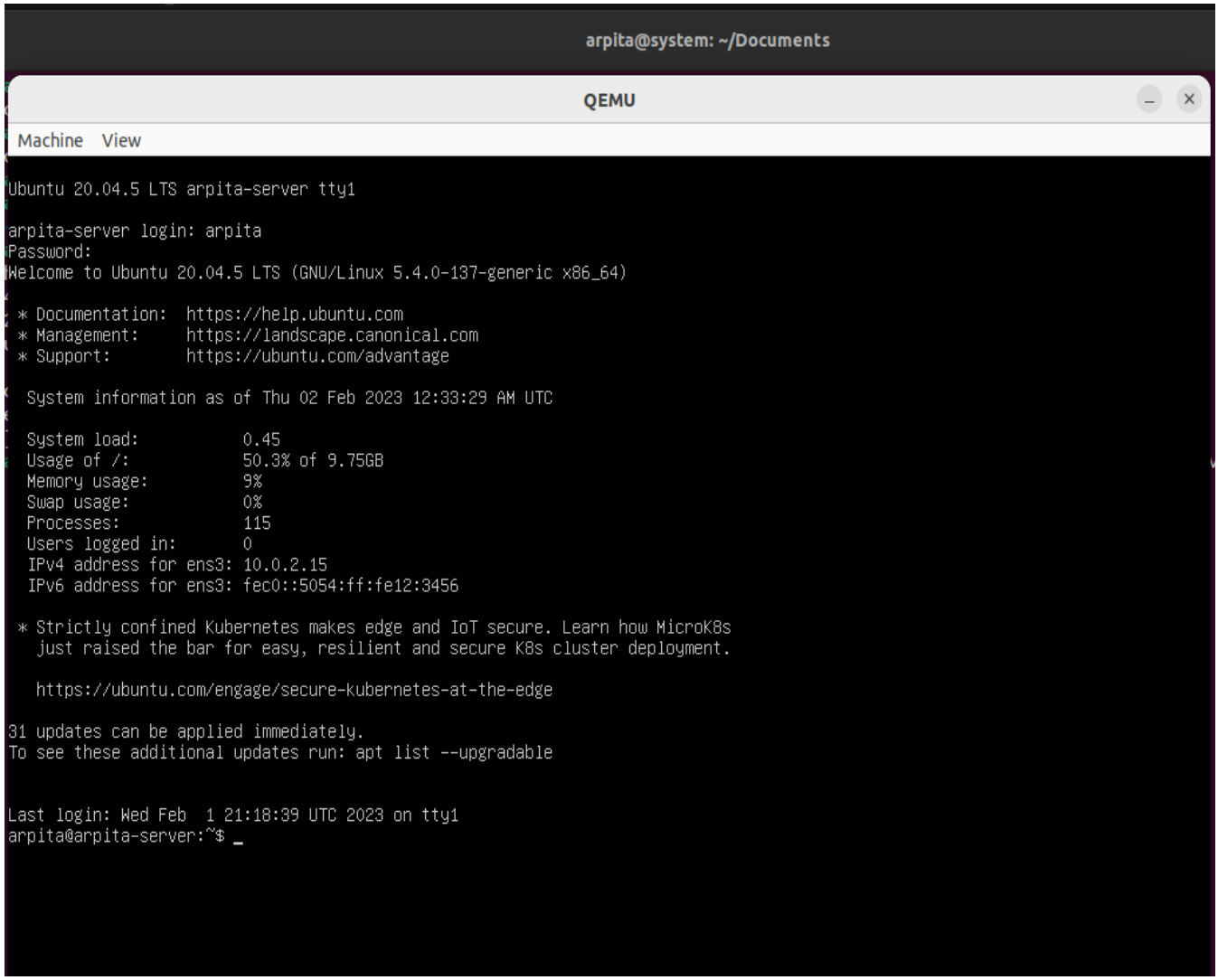
1. Now, since the ubuntu container is running, we can now install the sysbench in container using following commands –

```
root@acf04f6f0574:/# sysbench --version
sysbench 1.0.20
root@acf04f6f0574:/#
```

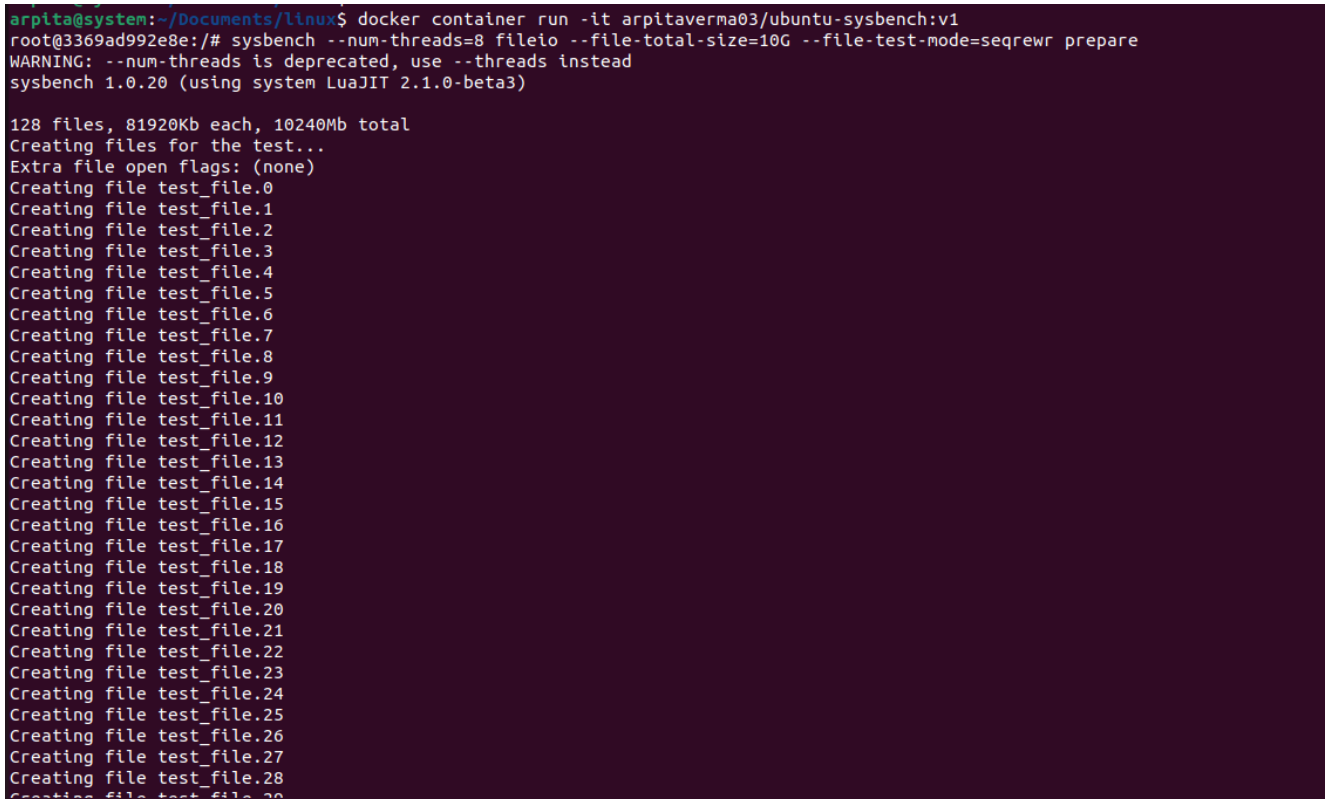
We need ensure the RAM, CPU allocation and the version of sysbench is same in both System Virtualization (QEMU) and OS Virtualization (Docker) setup.

Proof of Experiment

1. **QEMU Running Environment –**



2. Docker Running Environment -



Sysbench Experiment

CPU Test -

1. CPU Test in QEMU VM -

a. Test 1:

$ sysbench cpu --threads=1 --cpu-max-prime=35000 --time=10 run

```
arpita@arpita-server:~$ sysbench cpu --threads=1 --cpu-max-prime=35000 --time=10 run
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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


Prime numbers limit: 35000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   160.30

General statistics:
    total time:                          10.0040s
    total number of events:              1604

Latency (ms):
         min:                                    6.09
         avg:                                    6.23
         max:                                   25.46
         95th percentile:                        6.43
         sum:                                10000.67

Threads fairness:
    events (avg/stddev):           1604.0000/0.00
    execution time (avg/stddev):   10.0007/0.00

arpita@arpita-server:~$ _
```

b. Test2

$ sysbench cpu --threads=2 --cpu-max-prime=350000 --time=30 run

```
arpita@arpita-server:~$ sysbench cpu --threads=2 --cpu-max-prime=350000 --time=30 run
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 2
Initializing random number generator from current time


Prime numbers limit: 350000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:    12.31

General statistics:
    total time:                          30.0570s
    total number of events:              370

Latency (ms):
         min:                                  160.23
         avg:                                  162.44
         max:                                  229.38
         95th percentile:                      164.45
         sum:                                60101.87

Threads fairness:
    events (avg/stddev):           185.0000/0.00
    execution time (avg/stddev):   30.0509/0.00

arpita@arpita-server:~$ _
```

c. Test 3
$ sysbench cpu --threads=16 --cpu-max-prime=6000000 --time=50 run

```
arpita@arpita-server:~$ sysbench cpu --threads=16 --cpu-max-prime=6000000 --time=50 run
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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


Prime numbers limit: 6000000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:     0.23

General statistics:
    total time:                          70.2802s
    total number of events:              16

Latency (ms):
         min:                                70142.43
         avg:                                70208.33
         max:                                70248.76
         95th percentile:                    69758.52
         sum:                              1123333.33

Threads fairness:
    events (avg/stddev):           1.0000/0.00
    execution time (avg/stddev):   70.2083/0.03

arpita@arpita-server:~$
```

2. CPU Test in Docker Container -

a. Test 1:

$ sysbench cpu --threads=1 --cpu-max-prime=35000 --time=10 run

```
root@e2a69ce7da10:/# sysbench cpu --threads=1 --cpu-max-prime=35000 --time=10 run
sysbench 1.0.20 (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: 35000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   161.72

General statistics:
    total time:                          10.0029s
    total number of events:              1618

Latency (ms):
         min:                                    6.06
         avg:                                    6.18
         max:                                    8.82
         95th percentile:                        6.43
         sum:                                10000.68

Threads fairness:
    events (avg/stddev):           1618.0000/0.00
    execution time (avg/stddev):   10.0007/0.00

root@e2a69ce7da10:/#
```

b. Test 2

$ sysbench cpu --threads=2 --cpu-max-prime=350000 --time=30 run

```
root@e2a69ce7da10:/# sysbench cpu --threads=2 --cpu-max-prime=350000 --time=30 run
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 2
Initializing random number generator from current time


Prime numbers limit: 350000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:    12.42

General statistics:
    total time:                          30.0985s
    total number of events:              374

Latency (ms):
         min:                                  156.76
         avg:                                  160.75
         max:                                  171.34
         95th percentile:                      161.51
         sum:                                60119.63

Threads fairness:
    events (avg/stddev):           187.0000/0.00
    execution time (avg/stddev):   30.0598/0.03

root@e2a69ce7da10:/#
```

c. Test 3

$ sysbench cpu --threads=16 --cpu-max-prime=600000 --time=50 run

```
root@e2a69ce7da10:/# sysbench cpu --threads=16 --cpu-max-prime=6000000 --time=50 run
sysbench 1.0.20 (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: 6000000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:     0.35

General statistics:
    total time:                          92.1729s
    total number of events:              32

Latency (ms):
         min:                                    44880.58
         avg:                                    45828.28
         max:                                    47132.58
         95th percentile:                        46941.21
         sum:                                  1466505.00

Threads fairness:
    events (avg/stddev):           2.0000/0.00
    execution time (avg/stddev):   91.6566/0.39
```

FILEIO Test

1. FILE-IO Test in QEMU VM -

Test 1:
$ sysbench --num-threads=4 fileio --file-total-size=3G --file-test mode=rndwr prepare
$ sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr run
$ sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr cleanup

```
arpita@arpita-server:~$ sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr prepare
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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
```

```
arpita@arpita-server:~$ sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr run
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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 write test
Initializing worker threads...

Threads started!
```

```
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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 write test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     948.19
    fsyncs/s:                     1257.64

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               14.82

General statistics:
    total time:                          10.1222s
    total number of events:              21821

Latency (ms):
         min:                                    0.01
         avg:                                    1.83
         max:                                  138.91
         95th percentile:                        5.67
         sum:                                39961.67

Threads fairness:
    events (avg/stddev):           5455.2500/69.32
    execution time (avg/stddev):   9.9904/0.00

arpita@arpita-server:~$
```

Test 2

$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test mode=rndwr prepare

$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=rndwr run

$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=rndwr cleanup

```
arpita@arpita-server:~$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr prepare
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

128 files, 40960Kb each, 5120Mb 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
```

```
arpita@arpita-server:~$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr run
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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


Extra file open flags: (none)
128 files, 40MiB each
5GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential rewrite test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     1226.81
    fsyncs/s:                     1757.24

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               19.17

General statistics:
    total time:                          10.3499s
    total number of events:              28843

Latency (ms):
         min:                                    0.01
         avg:                                    5.55
         max:                                  436.23
         95th percentile:                       28.16
         sum:                                160068.03

Threads fairness:
    events (avg/stddev):           1802.6875/167.09
    execution time (avg/stddev):   10.0043/0.01
```

Test 3:

$ sysbench --num-threads=8 fileio --file-total-size=10G --file-test mode=rndwr prepare
$ sysbench --num-threads=8 fileio --file-total-size=10G --file-test-mode=rndwr run
$ sysbench --num-threads=8 fileio --file-total-size=10G --file-test-mode=rndwr cleanup

```
arpita@arpita-server:~$ sysbench --num-threads=8 fileio --file-total-size=10G --file-test-mode=rndrw prepare
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

128 files, 81920Kb each, 10240Mb 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
_
```

```
arpita@arpita-server:~$ sysbench --num-threads=8 fileio --file-total-size=10G --file-test-mode=rndrw run
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

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


Extra file open flags: (none)
128 files, 80MiB each
10GiB 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:                      507.06
    writes/s:                     337.85
    fsyncs/s:                     1180.30

Throughput:
    read, MiB/s:                  7.92
    written, MiB/s:               5.28

General statistics:
    total time:                          10.1918s
    total number of events:              19621

Latency (ms):
         min:                                    0.00
         avg:                                    4.08
         max:                                  508.44
         95th percentile:                       10.27
         sum:                                79966.28

Threads fairness:
    events (avg/stddev):           2452.6250/129.66
    execution time (avg/stddev):   9.9958/0.00
```

2. **FILE-IO Test in Docker Container -**

Test 1:

$ sysbench --threads=4 fileio --file-total-size=3G --file-test-mode=rndwr prepare
$ sysbench --threads=4 fileio --file-total-size=3G --file-test-mode=rndwr run
$ sysbench --threads=4 fileio --file-total-size=3G --file-test-mode=rndwr cleanup Run

```
root@e2a69ce7da10:/# sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr prepare
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (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
```

```
root@e2a69ce7da10:/# sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr run
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (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, 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 write test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     3350.70
    fsyncs/s:                     4332.38

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               52.35

General statistics:
    total time:                          10.0253s
    total number of events:              76532

Latency (ms):
         min:                                    0.01
         avg:                                    0.52
         max:                                  136.83
         95th percentile:                        1.55
         sum:                                39944.22

Threads fairness:
    events (avg/stddev):           19133.0000/72.24
    execution time (avg/stddev):   9.9861/0.00
```

Test 2:

$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr prepare
$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr run
$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr cleanup Run

```
root@e2a69ce7da10:/# sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr prepare
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

128 files, 40960Kb each, 5120Mb 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
root@e2a69ce7da10:/# sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr run
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (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, 40MiB each
5GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential rewrite test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     5798.03
    fsyncs/s:                     7613.59

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               90.59

General statistics:
    total time:                          10.0702s
    total number of events:              133039
```

```
Latency (ms):
         min:                                    0.01
         avg:                                    1.20
         max:                                  256.44
         95th percentile:                        5.47
         sum:                                160124.36

Threads fairness:
    events (avg/stddev):           8314.9375/275.23
    execution time (avg/stddev):   10.0078/0.00

root@e2a69ce7da10:/#
```

Test 3:

$ sysbench --threads=8 fileio --file-total-size=10G --file-test-mode=rndrw prepare
$ sysbench --threads=8 fileio --file-total-size=10G --file-test-mode=rndrw run
$ sysbench --threads=8 fileio --file-total-size=10G --file-test-mode=rndrw cleanup Run

```
root@e2a69ce7da10:/# sysbench --num-threads=8 fileio --file-total-size=10G --file-test-mode=rndrw prepare
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

128 files, 81920Kb each, 10240Mb 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
```

```
root@e2a69ce7da10:/# sysbench --num-threads=8 fileio --file-total-size=10G --file-test-mode=rndrw run
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.20 (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, 80MiB each
10GiB 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:                      2844.66
    writes/s:                     1895.94
    fsyncs/s:                     6167.37

Throughput:
    read, MiB/s:                  44.45
    written, MiB/s:               29.62

General statistics:
    total time:                          10.0386s
    total number of events:              108502

Latency (ms):
         min:                                    0.00
         avg:                                    0.74
         max:                                  190.27
         95th percentile:                        2.14
         sum:                                79924.06

Threads fairness:
    events (avg/stddev):           13562.7500/136.61
    execution time (avg/stddev):   9.9905/0.00
```

Experiment Result Analysis

CPU Test - QEMU VM vs. Docker Container -

To perform CPU Test, I have taken following three main parameters into consideration -

1. --threads: number of threads to be used to perform test
2. --cpu-max-prime: the maximum number up to which numbers to be tested if they are prime
3. –time: maximum time process can take to finish

By altering above parameters, following three CPU Tests are performed and the result of them in various measures are shown below in respective test tables -

Test 1:

$ sysbench cpu –threads=1 –cpu-max-prime=35000 –time=10 run

	min	**average**	**max**	**no. of events**
QEMU VM	**6.09**	**6.23**	**25.46**	**1604**
Docker	**6.06**	**6.18**	**8.82**	**1618**

Test 2:

$ sysbench cpu –threads=2 –cpu-max-prime=350000 –time=30 run

	min	**average**	**max**	**no. of events**
QEMU VM	**160.23**	**162.44**	**229.38**	**370**
Docker	**156.76**	**160.75**	**171.34**	**374**

Test 3:

$ sysbench cpu –threads=16 –cpu-max-prime=6000000 –time=50 run

	min	**average**	**max**	**no. of events**
QEMU VM	**70142.43**	**70208.33**	**70248.76**	**16**
Docker	**44880.58**	**45828.28**	**47132.58**	**32**

FILEIO Test - QEMU VM vs. Docker Container

To perform FILE-IO Test, I have taken following three main parameters into consideration-

1. --threads: number of threads to be used to perform test
2. --file-total-size: total size of file/files to be created
3. --file-test-mode: mode of of file test, there are five modes of file-io -
    a. rndrd
    b. rndrw
    c. rndwr
    d. seqrd
    e. seqrewr
    f. seqwr

By altering above parameters, following three File-IO Tests are performed and the result of them in various measures are shown below in respective test tables -

Test 1:

$ sysbench --num-threads=4 fileio --file-total-size=3G --file-test-mode=rndwr run

	min	average	max	no. of events
QEMU VM	**0.01**	**1.83**	**138.91**	**21821**
Docker	**0.01**	**0.52**	**136.83**	**76532**

Test 2:

$ sysbench --num-threads=16 fileio --file-total-size=5G --file-test-mode=seqrewr run

	min	average	max	no. of events
QEMU VM	**0.01**	**5.55**	**436.23**	**28843**
Docker	**0.01**	**1.20**	**256.44**	**133039**

Test 3:

$ sysbench --num-threads=8 fileio --file-total-size=4G --file-test-mode=rndrw run

	min	average	max	no. of events
QEMU VM	**0**	**4.08**	**508.44**	**19621**
Docker	**0**	**0.74**	**190.27**	**108502**

Findings and Conclusion of CPU and FILEIO Tests-

The above analysis has been based on three different cases for each CPU test and

File-IO test. Moreover, each test is run five times to see if the result of each test is consistent with the other case. Following are some of the findings which can be drawn from the results mentioned -

Experiment Shell Scripts

Shell Scripts for CPU and FILE-IO Test -

1. **Shell Script for CPU Tests -**

$ ./cpu_test.sh

```
#! /bin/bash

echo "Hello World, you are in $0, and going to test cpu!"

PRIMES_END=("35000" "350000" "6000000")
MAX_TIME=("10" "30" "50")
NUM_THREADS=("1" "2" "16")
TOTAL_RUNS=5
TOTAL_CASES=3

for ((i=0; i<$TOTAL_CASES;i++))
do
echo "Starting ${i+1}st Test Case"
for (( j=1; j <= $TOTAL_RUNS; j++ ))
do
        echo "Running ${j}st run of Test Case ${i+1}"
        sysbench cpu --threads=${NUM_THREADS[$i]} --cpu-max-
prime=${PRIMES_END[$i]} --time=${MAX_TIME[$i]} run
        echo "Completed ${j}st run of Test Case ${i+1}"
done
echo "Completed ${i}st Test Case"
done
```

2. Shell Script for FileIO Tests -

$ ./fileio_test.sh

```
#! /bin/bash

echo "Hello World, you are in $0, and going to test fileio"

NUM_THREADS=("4" "16" "8")
TOTAL_FILE_SIZES=("2G" "5G" "4G")
FILE_MODE=("rndwr" "seqrewr" "rndrw")
TOTAL_RUNS=5
```

```
TOTAL_CASES=3

for ((i=0; i<$TOTAL_CASES;i++))
do
   echo "*****************************Starting ${i+1}st Test
Case*************************************"
   for (( j=1; j <=$TOTAL_RUNS; j++ ))
   do
        echo "Running ${j}st run of Test Case ${i+1}"
        sysbench --threads=${NUM_THREADS[$i]} fileio --file-total-
size=${TOTAL_FILE_SIZES[$i]} --file-test-mode=${FILE_MODE[i]} prepare
        sysbench --threads=${NUM_THREADS[$i]} fileio --file-total-
size=${TOTAL_FILE_SIZES[$i]} --file-test-mode=${FILE_MODE[i]} run
        sysbench --threads=${NUM_THREADS[$i]} fileio --file-total-
size=${TOTAL_FILE_SIZES[$i]} --file-test-mode=${FILE_MODE[i]} cleanup
        echo "Completed ${j}st run of Test Case ${i+1}"
   done
   echo "*****************************Completed ${i}st Test
Case*************************************"
done
```

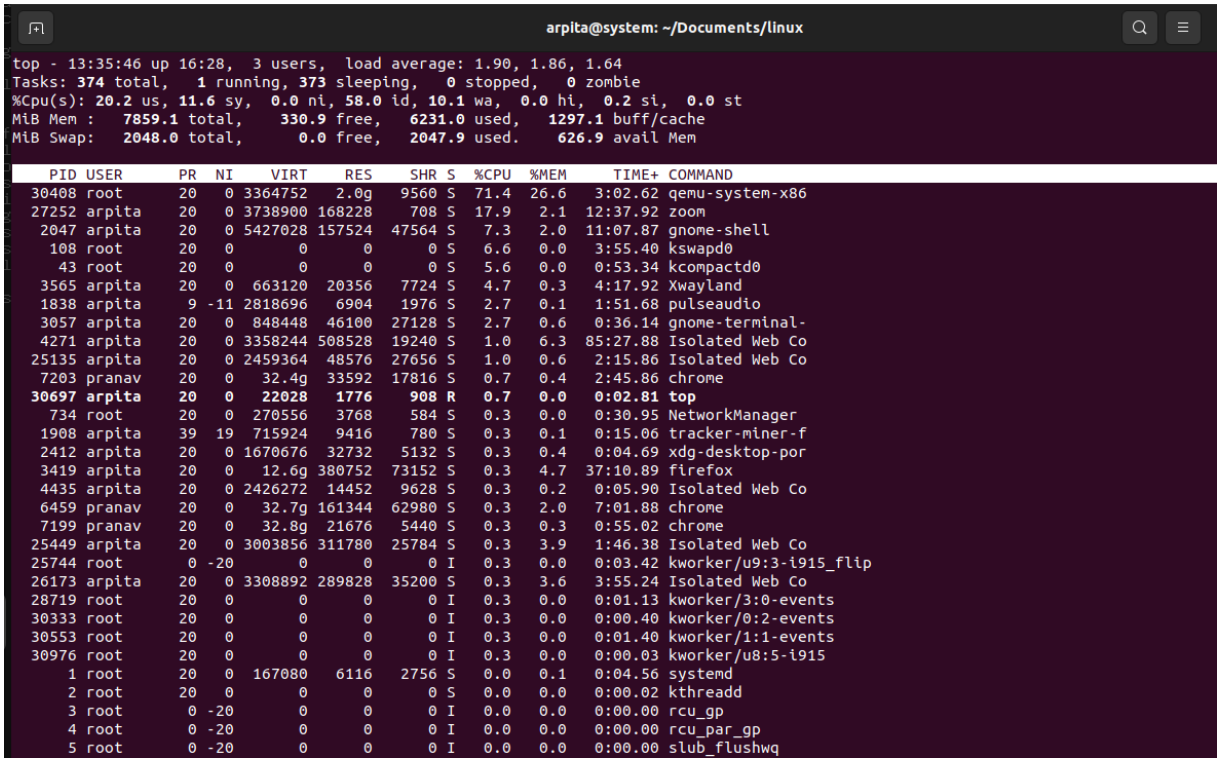
System Performance Tools and Analysis

CPU Utilization -

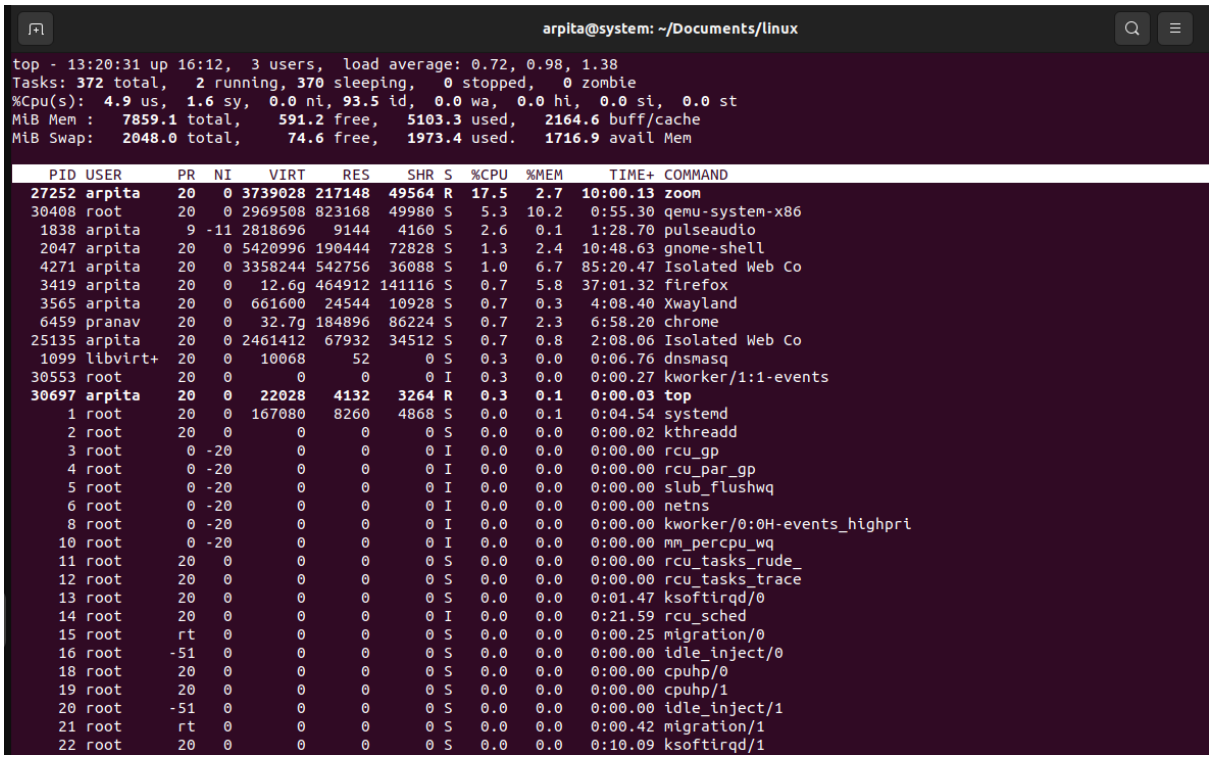
By making use of ‘top’ command in linux we can see the cpu usage in User Mode, System Mode and Idle Mode.

The CPU Utilization is observed in following cases -

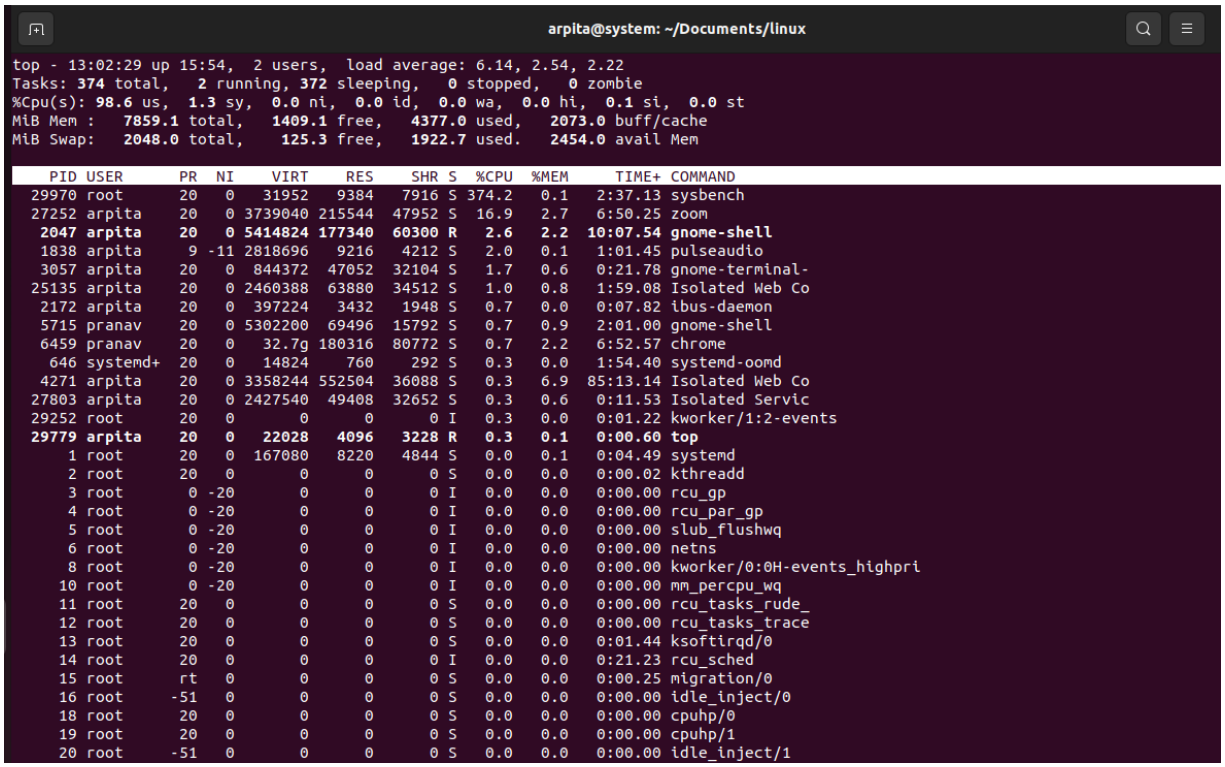
1. QEMU VM Running Sysbench CPU Test Case –



2. QEMU VM Not-Running Sysbench CPU Test Case –



3. Docker Container Running Sysbench CPU Test Case –



4. Docker Container Not-Running Sysbench CPU Test Case -

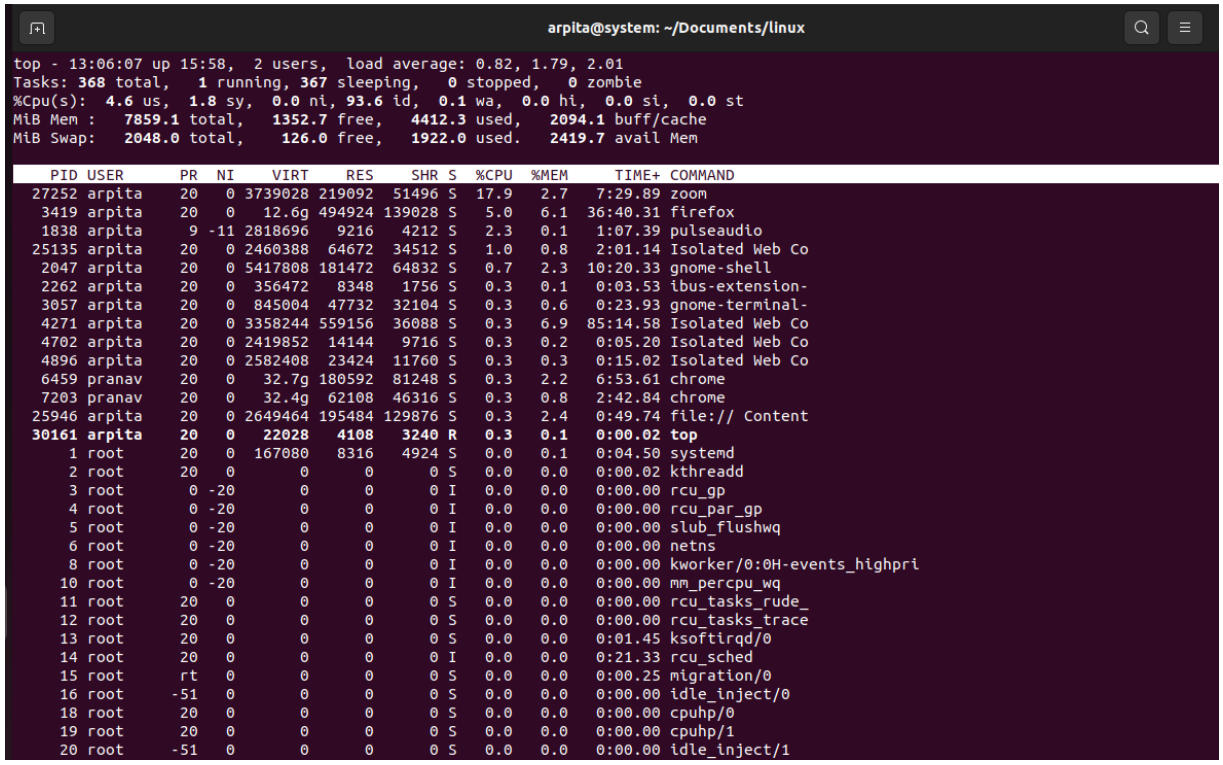


Table of comparison between above four cases -

1. **Docker Container CPU -**

	Sysbench Running	Sysbench Not-Running
User level CPU usage %	**6.6**	**3.4**
Kernel level CPU usage %	**3.2**	**0.9**
Idle CPU %	**28.5**	**95.7**

2. **Host OS CPU Usage - Docker Container -**

	Sysbench Running	Sysbench Not-Running
User level CPU usage %	**98.6**	**4.6**
Kernel level CPU usage %	**1.3**	**1.8**
Idle CPU %	**0.0**	**93.6**

3. **Host OS CPU Usage - QEMU VM -**

	Sysbench Running	Sysbench Not-Running
User level CPU usage %	**20.2**	**4.9**
Kernel level CPU usage %	**11.6**	**1.6**
Idle CPU %	**58.0**	**93.5**

IO Utilization -

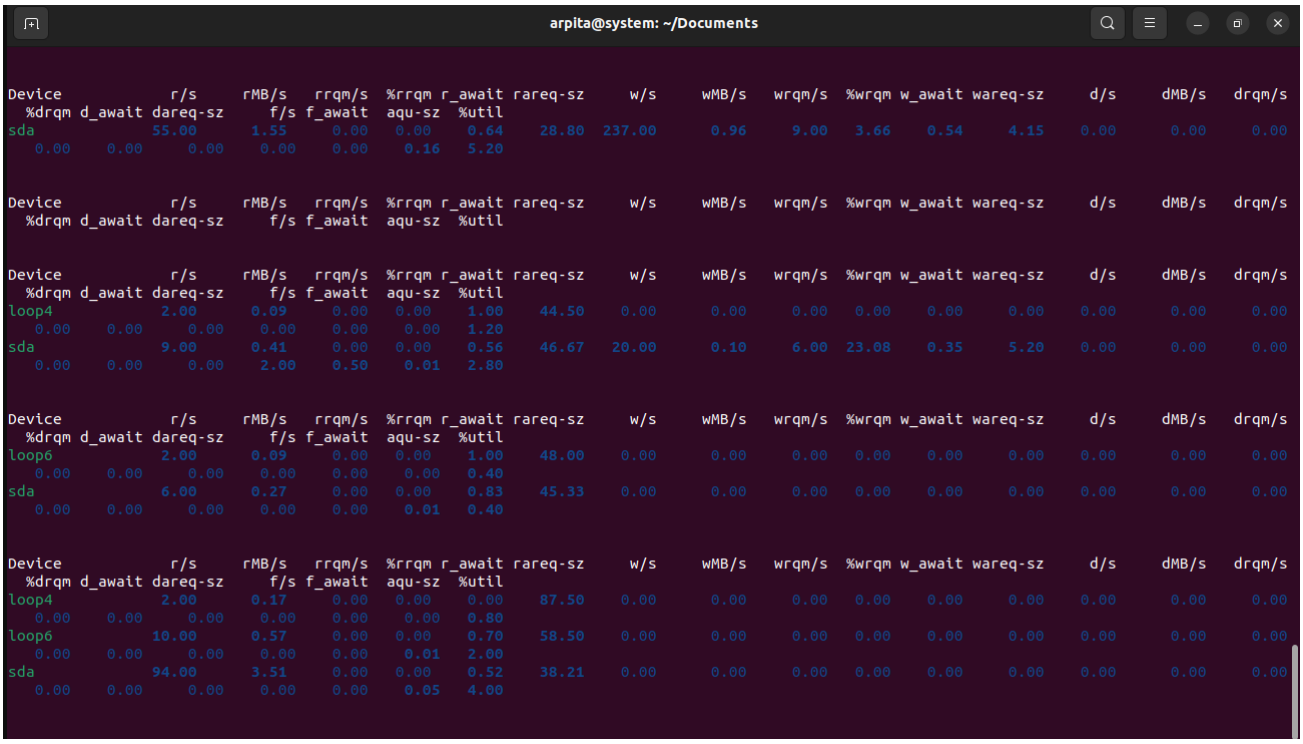
Real-time disk utilization is observed using the `iotop` tool. The installation of iotop requires to run below command in both docker container and qemu vm -

$ sudo apt install systat

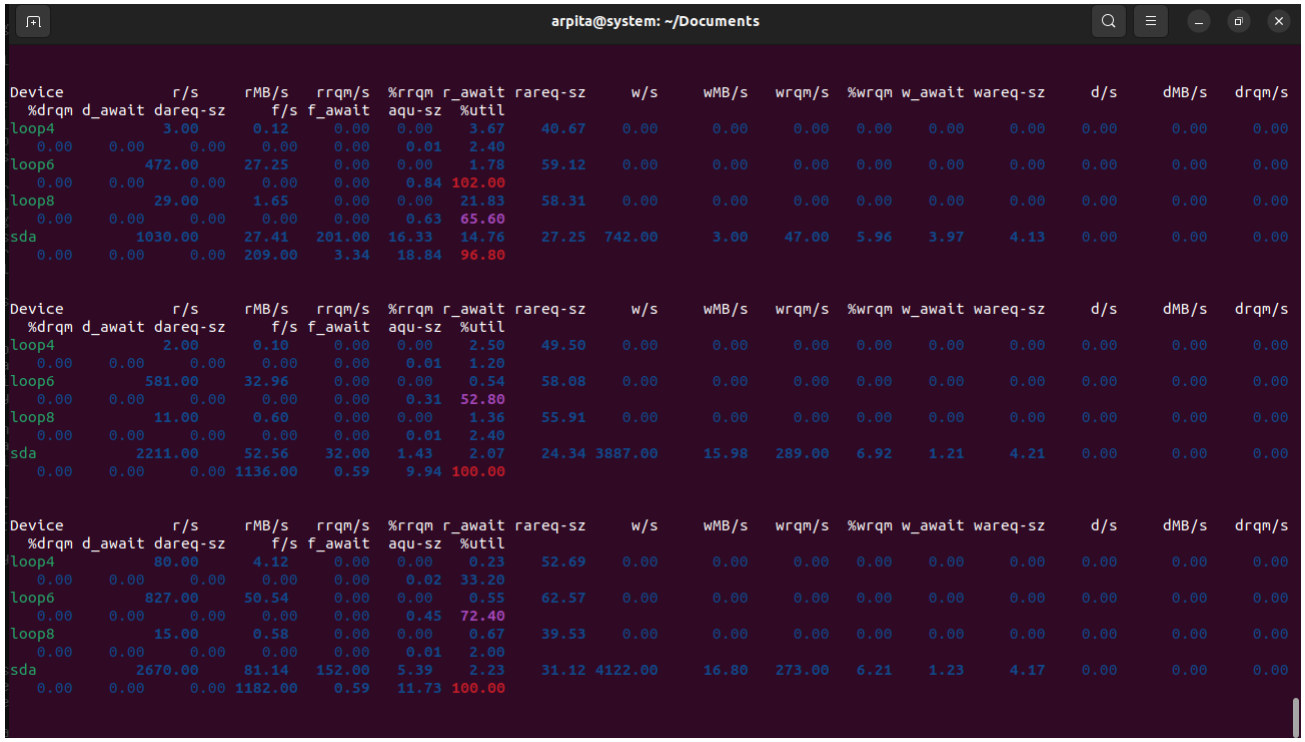
Using this tool, the disk utilization of Host OS while running Docker Container and QEMU VM is observed during fileio test in docker container using following command -

$ sudo iostat -dxzm 1

Test Case Commands for fileio –
Normal Host FileIO Screenshot –



Run Stage – Docker



Run Stage QEMU

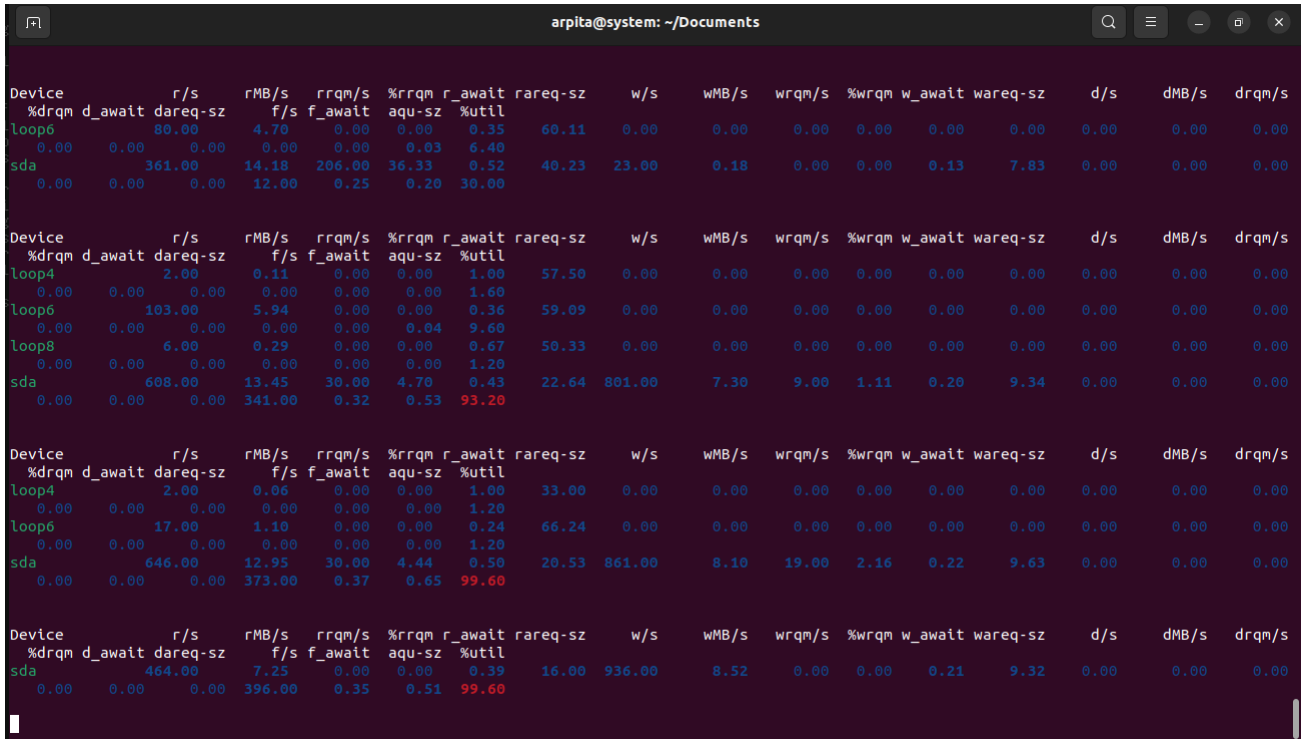


Table of I/O Throughput, I/O Latency, and I/O Disk Utilization Host OS

	I/O Throughput	I/O Latency	I/O Disk Utilization
Run Stage - Docker	0.03	0.00	0.20
Run Stage - QEMU	0.50	0.05	0.20

Findings and Conclusion -

1. The host OS disk utilization while running FILEIO test cases of containers was higher than that of QEMU VM.

2. The I/O Throughput, I/O Latency, and I/O Disk Utilization vary based on which stage of test they are captured. The Prepare Stage has the highest value for all three parameters, then comes the Run Stage, and then the Cleanup Stage.

Automation

Vagrant File -

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.

Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "bento/ubuntu-20.04-live"
```

```
# Disable automatic box update checking. If you disable this, then
# boxes will only be checked for updates when the user runs
# `vagrant box outdated`. This is not recommended.
# config.vm.box_check_update = false

# Create a forwarded port mapping which allows access to a specific port
# within the machine from a port on the host machine. In the example below,
# accessing "localhost:8080" will access port 80 on the guest machine.
# NOTE: This will enable public access to the opened port
# config.vm.network "forwarded_port", guest: 80, host: 8080

# Create a forwarded port mapping which allows access to a specific port
# within the machine from a port on the host machine and only allow access
# via 127.0.0.1 to disable public access
# config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"

# Create a private network, which allows host-only access to the machine
# using a specific IP.
# config.vm.network "private_network", ip: "192.168.33.10"

# Create a public network, which generally matched to bridged network.
# Bridged networks make the machine appear as another physical device on
# your network.
# config.vm.network "public_network"

# Share an additional folder to the guest VM. The first argument is
# the path on the host to the actual folder. The second argument is
# the path on the guest to mount the folder. And the optional third
# argument is a set of non-required options.
# config.vm.synced_folder "../data", "/vagrant_data"

# Provider Settings
config.vm.provider "virtualbox" do |vb|
        vb.memory = 2048
        vb.cpus = 2
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: "vagrant_script.sh"

end
```

Dockerfile -

```
FROM arpitaverma03/ubuntu-sysbench

COPY docker_script.sh /docker_script.sh
COPY test-cpu.sh /test-cpu.sh
COPY test-fileio.sh /test-fileio.sh

RUN chmod +x docker_script.sh
RUN chmod +x test-cpu.sh
RUN chmod +x test-cpu.sh

ENTRYPOINT bash docker_script.sh
```

Findings and Conclusion -

1. When the sysbench test is running inside a docker container, the user level cpu usage and kernel level cpu usage increases more than when the sysbench is not running. Moreover, the idle time decreases when sysbench is running in a docker container.

2. When the sysbench test is running in a docker container, the host os cpu usage goes up. The kernel level cpu usage goes from 0.3 to 1.2 when docker container starts running sysbench test and also the idle time decreases. This shows that the docker containers use the kernel level CPUs of host os.

3. When the sysbench test is running in a qemu vm, the host OS kernel-level cpu usage doesn’t go up, but user-level goes up. This shows that the qemu vm does not use the kernel level CPUs of host os.

Resources

- The link to the GitHub repository to access the homework – https://github.com/arpitav03/Cloud-Computing-Course-2023

- The link to the docker image of created image in this experiment https://hub.docker.com/repository/docker/arpitaverma03/ubuntu-sysbench/

  Or

  $ docker pull arpitaverma03/sysbench-ubuntu:version1