

System vs. OS Virtualization

TABLE OF CONTENTS

Environment Setup - Host System Configuration	2
System Virtualization Setup	
QEMU Setup	3
Sysbench Setup on QEMU VM	5
Going In-Depth of QEMU	6
Operating System (OS) Virtualization Setup	
Docker Setup	7
Sysbench Setup in Docker Container	9
Sysbench Experiment	
CPU Test in QEMU VM	10
CPU Test in Docker Container	12
FILE-IO Test in QEMU VM	14
FILE-IO Test in Docker Container	16
Experiment Result Analysis	
CPU Test - QEMU VM vs. Docker Container	19
FILEIO Test - QEMU VM vs. Docker Container	20
Findings and Conclusion of CPU and FILEIO Tests	21
Experiment Shell Scripts	
Shell Scripts for CPU and FILE-IO Test	22
System Performance Tools and Analysis	
CPU Utilization	24
- Table of Comparison Between CPU Test Cases	28
- Findings and Conclusion	29
IO Utilization	31
- Table of Comparison Between FILEIO Test Case	35
- Findings and Conclusion	
Automation	36
Vagrant File	37
Dockerfile	
Resources	38

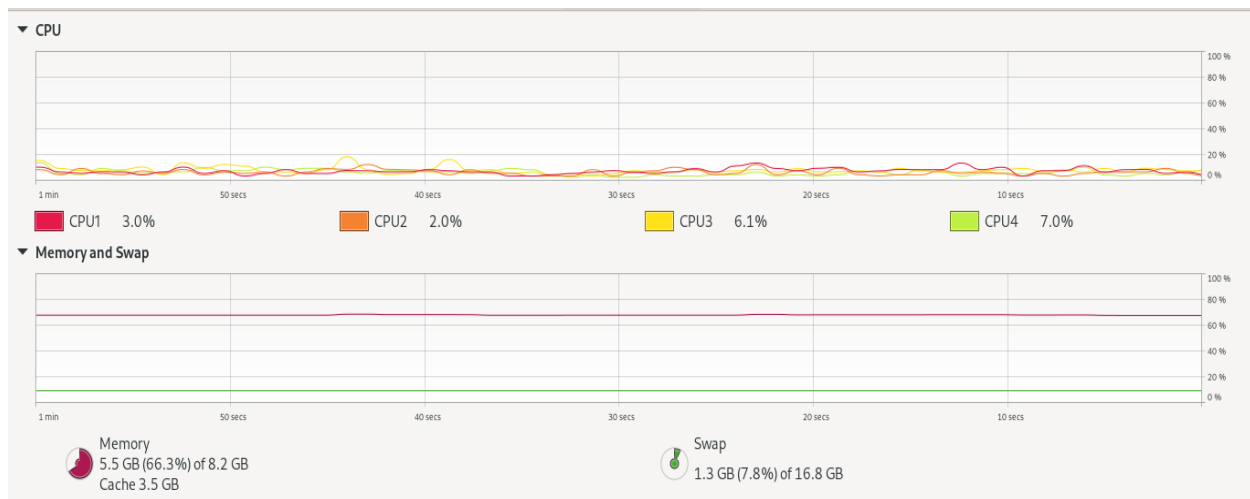
Environment Setup



Host System Configuration Details -

The host system 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 - Linux fedora 5.19.15-201.fc36.x86_64

Detailed Host-System Configuration Screenshot -



Device	Directory	Type	Total	Available	Used	
 /dev/sda7	/	ext4	121.0 GB	81.9 GB	32.9 GB	<div><div></div></div> 28%
 /dev/sda5	/boot/efi	vfat	524.0 MB	509.3 MB	14.7 MB	<div><div></div></div> 2%

QEMU Ubuntu Virtual Machine Configuration Details -

The Ubuntu VM is created with QEMU by allocating 10 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 (Fedora Linux) using following command -

\$ sudo dnf install qemu -y

Command Output -

```
[pranav@fedora temp]$ sudo dnf install qemu
[sudo] password for pranav:
Last metadata expiration check: 0:55:26 ago on Sun 16 Oct 2022 04:05:34 PM PDT.
Dependencies resolved.
=====
Package                                Architecture      Version            Repository
=====
Installing:
qemu                                    x86_64            2:6.2.0-15.fc36    updates
Installing dependencies:
SLiOZ                                  noarch            20210217-4.git33a7322d.fc36    fedora
edk2-aarch64                           noarch            20220826gitba0e0e4c6a17-1.fc36    updates
edk2-arm                               noarch            20220826gitba0e0e4c6a17-1.fc36    updates
openbios                              noarch            1:20200725-4.git7f28286.fc36    fedora
qemu-system-aarch64                    x86_64            2:6.2.0-15.fc36    updates
qemu-system-aarch64-core               x86_64            2:6.2.0-15.fc36    updates
qemu-system-alpha                     x86_64            2:6.2.0-15.fc36    updates
qemu-system-alpha-core                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-arm                       x86_64            2:6.2.0-15.fc36    updates
qemu-system-arm-core                   x86_64            2:6.2.0-15.fc36    updates
qemu-system-avr                       x86_64            2:6.2.0-15.fc36    updates
qemu-system-avr-core                   x86_64            2:6.2.0-15.fc36    updates
qemu-system-cris                      x86_64            2:6.2.0-15.fc36    updates
qemu-system-cris-core                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-m68k                      x86_64            2:6.2.0-15.fc36    updates
qemu-system-m68k-core                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-microblaze                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-microblaze-core           x86_64            2:6.2.0-15.fc36    updates
qemu-system-mips                      x86_64            2:6.2.0-15.fc36    updates
qemu-system-mips-core                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-nios2                     x86_64            2:6.2.0-15.fc36    updates
qemu-system-nios2-core                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-or1k                      x86_64            2:6.2.0-15.fc36    updates
qemu-system-or1k-core                 x86_64            2:6.2.0-15.fc36    updates
qemu-system-ppc                       x86_64            2:6.2.0-15.fc36    updates
qemu-system-ppc-core                  x86_64            2:6.2.0-15.fc36    updates
qemu-system-riscv                     x86_64            2:6.2.0-15.fc36    updates
qemu-system-riscv-core                 x86_64            2:6.2.0-15.fc36    updates
```

2. Download ubuntu server image using following command -

\$ wget <https://releases.ubuntu.com/focal/ubuntu-20.04.5-live-server-amd64.iso>

Command Output -

```
[pranav@fedora temp]$ wget https://releases.ubuntu.com/focal/ubuntu-20.04.5-live-server-amd64.iso
--2022-10-16 17:06:33-- http://wget/
Resolving wget (wget)... failed: Name or service not known.
wget: unable to resolve host address 'wget'
--2022-10-16 17:06:37-- https://releases.ubuntu.com/focal/ubuntu-20.04.5-live-server-amd64.iso
Resolving releases.ubuntu.com (releases.ubuntu.com)... 185.125.190.37, 91.189.91.124, 91.189.91.123, ...
Connecting to releases.ubuntu.com (releases.ubuntu.com)|185.125.190.37|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1406533632 (1.3G) [application/x-iso9660-image]
Saving to: 'ubuntu-20.04.5-live-server-amd64.iso'

ubuntu-20.04.5-live-server-amd64.iso      4%[==>] 58.52M  5.94MB/s
```

3. Create QEMU image of ubuntu in qcow2 file format using following command -

\$ qemu-img create -f qcow2 ubuntu.img 10G

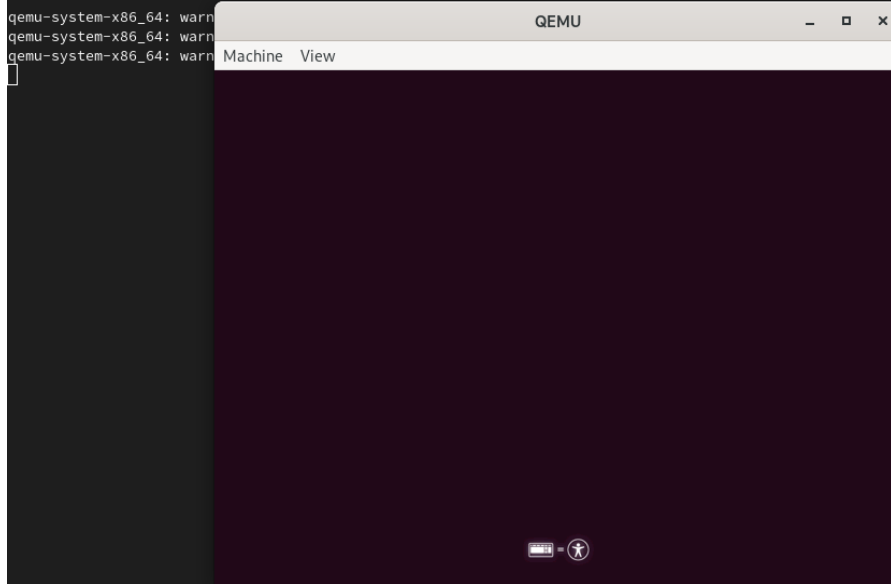
Command Output -

```
[pranav@fedora temp]$ qemu-img create -f qcow2 ubuntu.img 10G
Formatting 'ubuntu.img', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=10737418240 lazy_refcounts=off refcount_bits=16
[pranav@fedora temp]$ qemu-img info ubuntu.img
image: ubuntu.img
file format: qcow2
virtual size: 10 GiB (10737418240 bytes)
disk size: 196 KiB
cluster_size: 65536
Format specific information:
  compat: 1.1
  compression type: zlib
  lazy refcounts: false
  refcount bits: 16
  corrupt: false
  extended l2: false
```

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

Command Output -

```
[pranav@fedora temp]$ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ubuntu-20.04.5-live-server-amd64.iso -m 2048
[sudo] password for pranav:
qemu-system-x86_64: warning: AT-SPI: Could not obtain desktop path or name
```



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

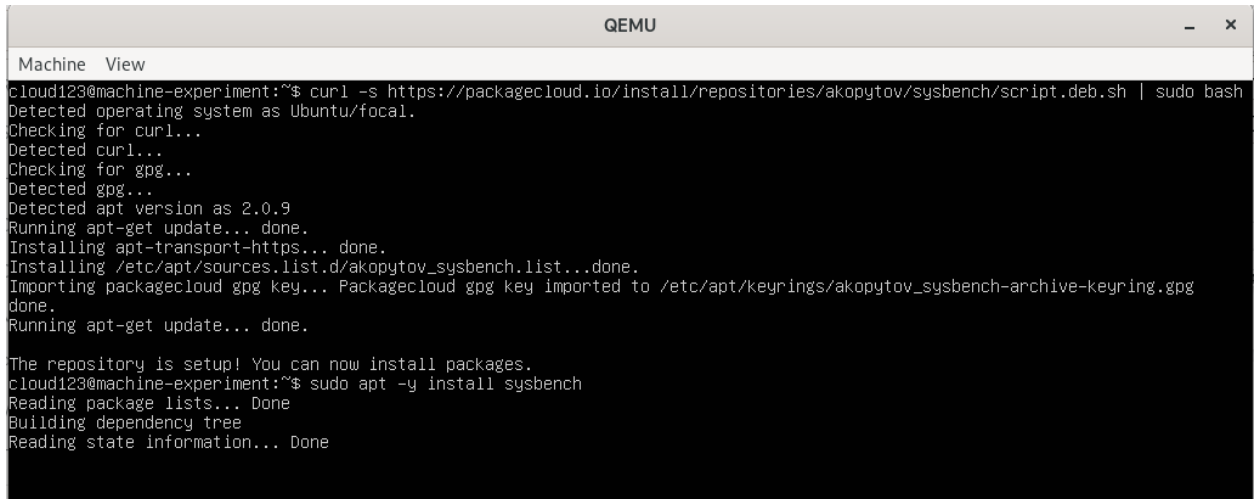
- hda : use file as a hard disk
- boot : this option specifies the boot order, for x86 architecture these drive letters are - a, b (for floppy drives), c (first hard-disk), d (first CD-ROM), n-p (Etherboot from network adapter 1-4). Hard-disk boot is the default option.
- cdrom : this option specifies 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
```

Command Output -

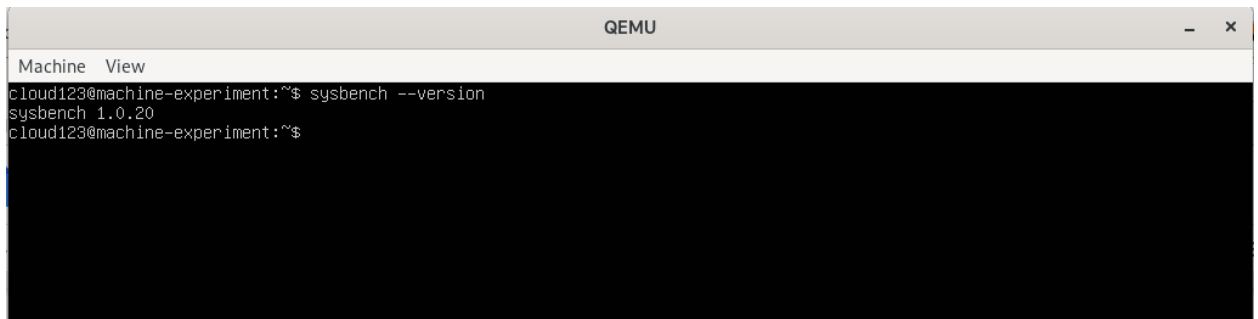


```
QEMU  
Machine View  
cloud123@machine-experiment:~$ curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.deb.sh | sudo bash  
Detected operating system as Ubuntu/focal.  
Checking for curl...  
Detected curl...  
Checking for gpg...  
Detected gpg...  
Detected apt version as 2.0.9  
Running apt-get update... done.  
Installing apt-transport-https... done.  
Installing /etc/apt/sources.list.d/akopytov_sysbench.list...done.  
Importing packagecloud gpg key... Packagecloud gpg key imported to /etc/apt/keyrings/akopytov_sysbench-archive-keyring.gpg  
done.  
Running apt-get update... done.  
  
The repository is setup! You can now install packages.  
cloud123@machine-experiment:~$ sudo apt -y install sysbench  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done
```

2. Check the version of sysbench installed using following command -

```
$ sysbench --version
```

Command Output -



```
QEMU  
Machine View  
cloud123@machine-experiment:~$ sysbench --version  
sysbench 1.0.20  
cloud123@machine-experiment:~$
```

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 ubuntu1.img \
    -boot c \
    -device virtio-net,netdev=vm \
    -netdev user,id=vm,hostfwd=tcp:127.0.0.1:9001-:22
```

Operating System (OS) Virtualization Setup

Docker Setup -

1. Install the Docker on host (Fedora Linux) using following commands -

```
$ sudo dnf -y install dnf-plugins-core
```

```
$ sudo dnf config-manager \
    --add-repo \
```

```
https://download.docker.com/linux/fedora/docker-ce.repo
```

```
$ sudo dnf install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

2. The above command installs the components required for docker and creates a group called docker. Now, to start the docker use following command -

```
$ sudo systemctl start docker
```

3. To verify installation is proper, run a hello world container -

```
$ sudo docker run hello-world
```

Command Output -

```
[pranav@fedora hw1]$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:18a657d0cc1c7d0678a3fbea8b7eb4918bba25968d3e1b0adebfa71caddbc346
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/
```

4. Now, pull the ubuntu:latest image from docker hub using following command -

```
$ docker pull ubuntu:latest
```

Command Output -

```
[pranav@fedora hw1]$ docker pull ubuntu:latest
latest: Pulling from library/ubuntu
cf92e523b49e: Pull complete
Digest: sha256:35fb073f9e56eb84041b0745cb714eff0f7b225ea9e024f703cab56aaa5c7720
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

5. Run the docker container with ubuntu:latest base image using following command -

```
$ docker container run -it --memory="2g" --cpus="2"
pranav2306/sysbench-ubuntu:version1 /bin/bash
```

Command Output -

```
[pranav@fedora cloud-computing-course]$ docker container run -it --memory="2g" --cpus="2" pranav2306/sysbench-ubuntu:version1 /bin/bash
root@44cfe525fbb4:/#
```

6. Check and update the Ubuntu OS running inside the container using the following command -

```
$ apt install update
```

7. Check the cpu and memory the container is using using following command -
\$ docker stats 44cfe525fbb4

Command Output:

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
44cfe525fbb4	busy_pasteur	0.00%	1.918MiB / 2GiB	0.09%	4.35kB / 0B	1.78MB / 0B	1

Sysbench Setup in Docker Container -

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

```
$ 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 the following command -

```
$ sysbench --version
```

Command Output -

```
root@44cfe525fbb4:/# sysbench --version  
sysbench 1.0.20
```

3. After setting up sysbench on docker, we need to create an image out of this container to use it in future test cases. Create image using following command -

```
$ docker commit 44cfe525fbb4 panu2306/sysbench-ubuntu:version1
```

4. Now, lets check the created image and its history using following commands -

```
$ docker image ls  
$ docker image history 30dd295e0a5b
```

Command Output -

```
[pranav@fedora cloud-computing-course]$ docker image ls  
REPOSITORY          TAG          IMAGE ID        CREATED        SIZE  
pranav2306/sysbench-ubuntu  version1    30dd295e0a5b   3 days ago    139MB  
[pranav@fedora cloud-computing-course]$ docker image history 30dd295e0a5b  
IMAGE          CREATED        CREATED BY          SIZE      COMMENT  
30dd295e0a5b   3 days ago    /bin/bash           61.6MB  
<missing>      13 days ago   /bin/sh -c #(nop)  CMD ["bash"]      0B  
<missing>      13 days ago   /bin/sh -c #(nop)  ADD file:6cd2e13356aa5339c...  77.8MB  
[pranav@fedora cloud-computing-course]$
```

NOTE - Make sure 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 -

```
[pranav@fedora:~/Documents/temp] — sudo qemu-system-x86_64 -cdrom ubuntu-20.04.5-live-server-amd64.iso -accel kvm -cpu host -m 2048 -smp 2 -hda u...  
[pranav@fedora ~]$  
-device virtio-net  
[sudo] password pranav:  
qemu-system-x86_64: warning: /etc/libvirt/qemu.conf:17: warn: qemu.conf:17: smp: invalid value: 2048, ignored.  
[pranav@fedora ~]$  
cloud-compute-1 login: cloudy  
password:  
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-128-generic x86_64)  
  
qemu-system-x86_64 * Documentation:  https://help.ubuntu.com  
qemu-system-x86_64 * Management:    https://landscape.canonical.com  
qemu-system-x86_64 * Support:      https://ubuntu.com/advantage  
  
System information disabled due to load higher than 2.0  
  
12 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
New release '22.04.1 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Oct 17 18:50:48 UTC 2022 on tty1  
cloudy@cloud-compute-1:~$
```

2. Docker Running Environment -

```
root@8a9aeeecb3bb:/

[pranav@fedora cloud-computing-course]$ docker container run -it --memory="2g" --cpus="2" pranav2306/sysbench-ubuntu:version1 /bin/bash
root@8a9aeeecb3bb:/# sysbench --version
sysbench 1.0.20
root@8a9aeeecb3bb:/#
```

Sysbench Experiment

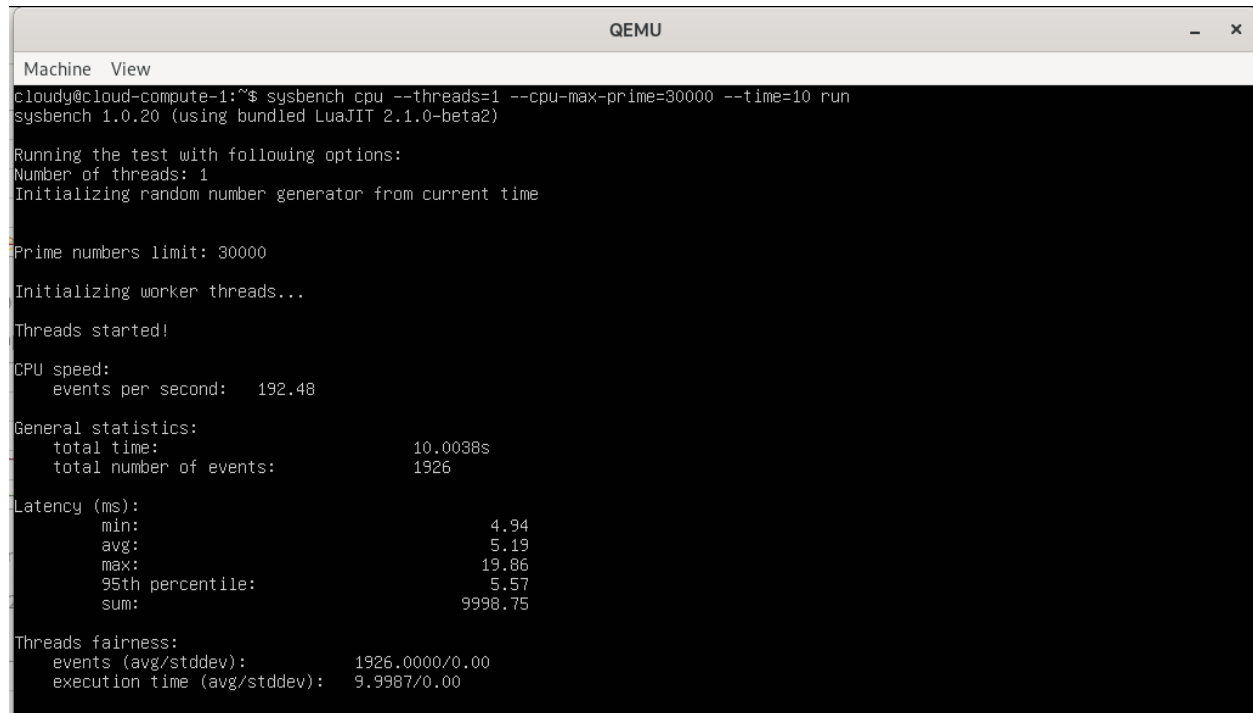
CPU Test -

1. CPU Test in QEMU VM -

a. Test 1:

```
$ sysbench cpu --threads=1 --cpu-max-prime=30000 --time=10 run
```

Command O/P Screenshot -



```
QEMU
Machine View
cloudy@cloud-compute-1:~$ sysbench cpu --threads=1 --cpu-max-prime=30000 --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: 30000

Initializing worker threads...

Threads started!

CPU speed:
  events per second:   192.48

General statistics:
  total time:          10.0038s
  total number of events: 1926

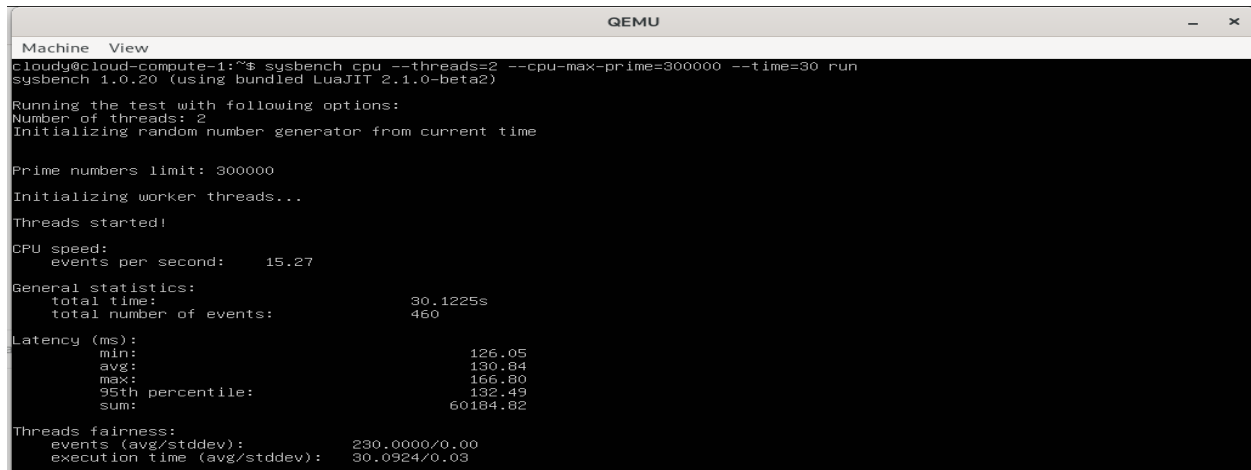
Latency (ms):
  min:                 4.94
  avg:                 5.19
  max:                 19.86
  95th percentile:    5.57
  sum:                 9998.75

Threads fairness:
  events (avg/stddev): 1926.0000/0.00
  execution time (avg/stddev): 9.9987/0.00
```

b. Test 2:

```
$ sysbench cpu --threads=2 --cpu-max-prime=300000 --time=30 run
```

Command O/P Screenshot -



```
Machine View
cloudy@cloud-compute-1:~$ sysbench cpu --threads=2 --cpu-max-prime=300000 --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: 300000
Initializing worker threads...
Threads started!

CPU speed:
  events per second:    15.27

General statistics:
  total time:            30.1225s
  total number of events: 460

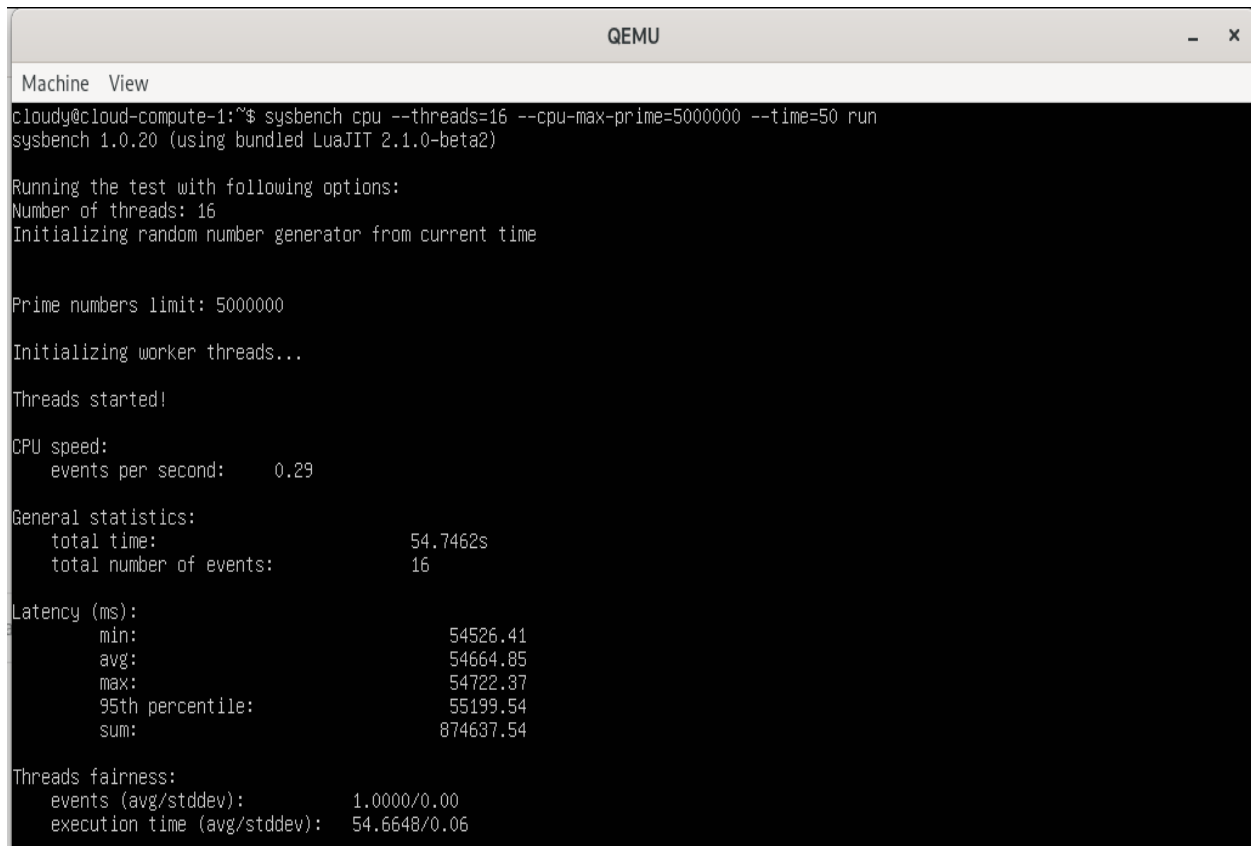
Latency (ms):
  min:                   126.05
  avg:                   130.84
  max:                   166.80
  95th percentile:      132.49
  sum:                   60184.82

Threads fairness:
  events (avg/stddev):   230.0000/0.00
  execution time (avg/stddev): 30.0924/0.03
```

c. Test 3:

```
$ sysbench cpu --threads=16 --cpu-max-prime=5000000 --time=50 run
```

Command O/P Screenshot -



```
Machine View
cloudy@cloud-compute-1:~$ sysbench cpu --threads=16 --cpu-max-prime=5000000 --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: 5000000
Initializing worker threads...
Threads started!

CPU speed:
  events per second:    0.29

General statistics:
  total time:            54.7462s
  total number of events: 16

Latency (ms):
  min:                   54526.41
  avg:                   54664.85
  max:                   54722.37
  95th percentile:      55199.54
  sum:                   874637.54

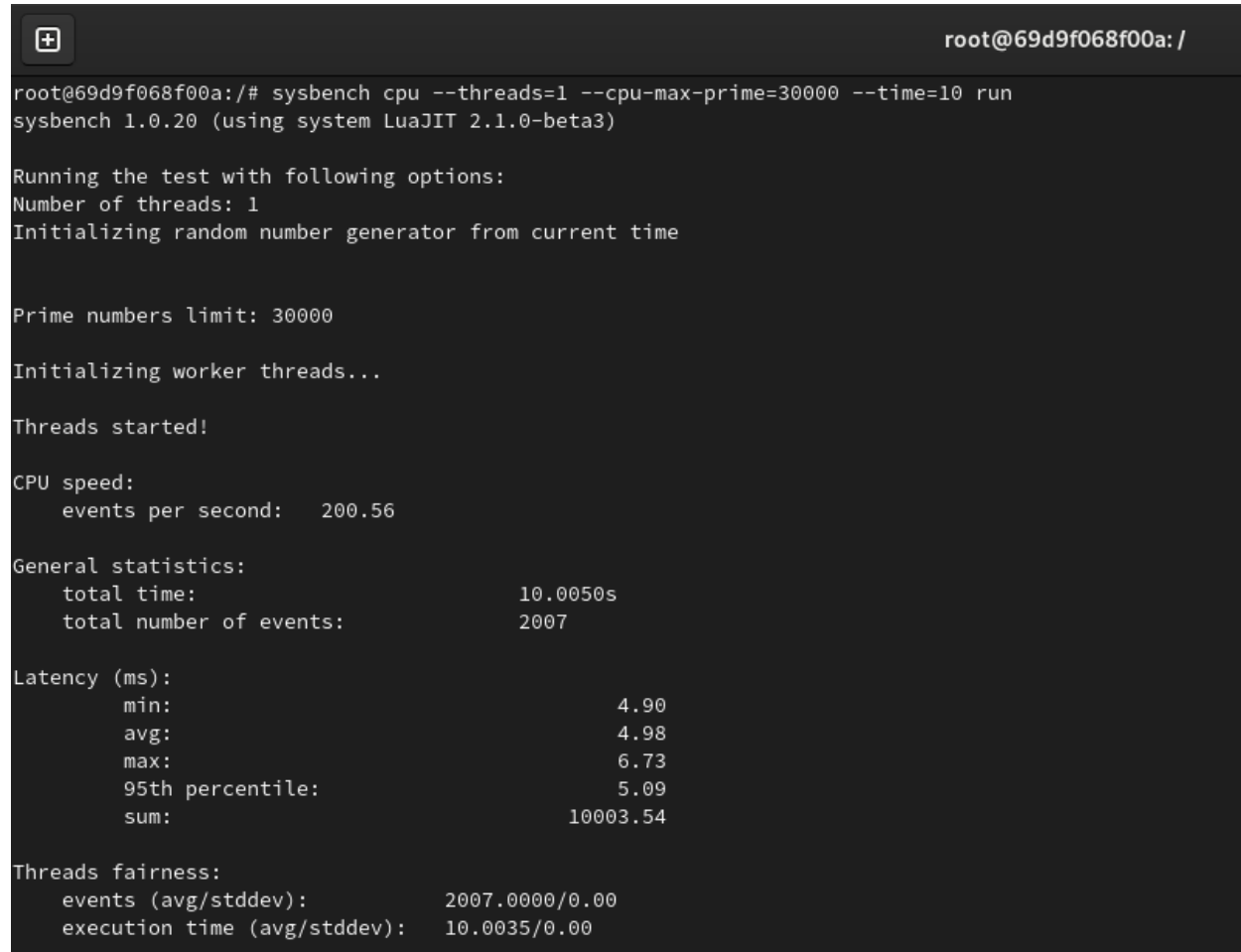
Threads fairness:
  events (avg/stddev):   1.0000/0.00
  execution time (avg/stddev): 54.6648/0.06
```

2. CPU Test in Docker Container -

a. Test 1:

```
$ sysbench cpu --threads=1 --cpu-max-prime=30000 --time=10 run
```

Command O/P Screenshot -



```
root@69d9f068f00a: /  
root@69d9f068f00a:/# sysbench cpu --threads=1 --cpu-max-prime=30000 --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: 30000  
  
Initializing worker threads...  
  
Threads started!  
  
CPU speed:  
  events per second:   200.56  
  
General statistics:  
   total time:                   10.0050s  
   total number of events:       2007  
  
Latency (ms):  
   min:                          4.90  
   avg:                          4.98  
   max:                          6.73  
   95th percentile:             5.09  
   sum:                          10003.54  
  
Threads fairness:  
   events (avg/stddev):       2007.0000/0.00  
   execution time (avg/stddev): 10.0035/0.00
```

b. Test 2:

```
$ sysbench cpu --threads=2 --cpu-max-prime=300000 --time=30 run
```

Command O/P Screenshot -

```
root@69d9f068f00a: /  
root@69d9f068f00a:/# sysbench cpu --threads=2 --cpu-max-prime=300000 --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: 300000  
Initializing worker threads...  
Threads started!  
  
CPU speed:  
  events per second:    15.41  
  
General statistics:  
  total time:                30.1053s  
  total number of events:    464  
  
Latency (ms):  
  min:                        127.87  
  avg:                        129.73  
  max:                        164.67  
  95th percentile:          132.49  
  sum:                        60195.50  
  
Threads fairness:  
  events (avg/stddev):       232.0000/0.00  
  execution time (avg/stddev): 30.0978/0.01
```

c. Test 3:

```
$ sysbench cpu --threads=16 --cpu-max-prime=5000000 --time=50 run
```

Command O/P Screenshot -

```
root@69d9f068f00a: /  
root@69d9f068f00a:/# sysbench cpu --threads=16 --cpu-max-prime=5000000 --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: 5000000  
Initializing worker threads...  
Threads started!  
  
CPU speed:  
  events per second:      0.23  
  
General statistics:  
  total time:                69.9404s  
  total number of events:    16  
  
Latency (ms):  
  min:                        55624.08  
  avg:                        66333.14  
  max:                        69899.46  
  95th percentile:          69758.52  
  sum:                        1061330.27  
  
Threads fairness:  
  events (avg/stddev):       1.0000/0.00  
  execution time (avg/stddev): 66.3331/4.29
```

FILEIO Test -

1. FILE-IO Test in QEMU VM -

a. Test 1:

```
$ sysbench --num-threads=4 fileio --file-total-size=2G --file-test-mode=rndwr  
prepare  
$ sysbench --num-threads=4 fileio --file-total-size=2G --file-test-mode=rndwr run  
$ $ sysbench --num-threads=4 fileio --file-total-size=2G --file-test-mode=rndwr  
cleanup
```

Run Command O/P Screenshot -

```
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, 16MiB each  
2GiB 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:         912.74  
  fsyncs/s:        1206.58  
  
Throughput:  
  read, MiB/s:      0.00  
  written, MiB/s:   14.26  
  
General statistics:  
  total time:       10.1869s  
  total number of events: 21082  
  
Latency (ms):  
  min:              0.00  
  avg:              1.89  
  max:             135.70  
  95th percentile: 5.99  
  sum:             39946.46  
  
Threads fairness:  
  events (avg/stddev): 5270.5000/407.55  
  execution time (avg/stddev): 9.9866/0.00  
cloudy@cloud-compute-1:~$
```

b. 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 Command O/P Screenshot -

```

cloudy@cloud-compute-1:~$ 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:         2315.72
  fsyncs/s:         3154.30

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:    36.18

General statistics:
  total time:        10.3618s
  total number of events: 54643

Latency (ms):
  min:               0.01
  avg:                2.93
  max:               190.80
  95th percentile:   9.22
  sum:               159908.26

Threads fairness:
  events (avg/stddev): 3415.1875/163.00
  execution time (avg/stddev): 9.9943/0.00

cloudy@cloud-compute-1:~$

```

c. Test 3:

```

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

```

Run Command O/P Screenshot -

```

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, 32MiB each
4GiB 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:          618.72
  writes/s:         412.32
  fsyncs/s:         1420.18

Throughput:
  read, MiB/s:      9.67
  written, MiB/s:    6.44

General statistics:
  total time:        10.1818s
  total number of events: 23939

Latency (ms):
  min:               0.00
  avg:                3.34
  max:               65.25
  95th percentile:   9.22
  sum:               79928.41

Threads fairness:
  events (avg/stddev): 2992.3750/248.65
  execution time (avg/stddev): 9.9911/0.00

cloudy@cloud-compute-1:~$

```


2. FILE-IO Test in Docker Container -

a. Test 1:

```
$ sysbench --threads=4 fileio --file-total-size=2G --file-test-mode=rndwr prepare
```

```
$ sysbench --threads=4 fileio --file-total-size=2G --file-test-mode=rndwr run
```

```
$ sysbench --threads=4 fileio --file-total-size=2G --file-test-mode=rndwr cleanup
```

Run Command O/P Screenshot-

```
root@69d9f068f00a: /  
2GiB 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:               2110.23  
  fsyncs/s:               2740.11  
  
Throughput:  
  read, MiB/s:            0.00  
  written, MiB/s:         32.97  
  
General statistics:  
  total time:              10.0441s  
  total number of events:  48216  
  
Latency (ms):  
  min:                     0.00  
  avg:                     0.83  
  max:                     221.44  
  95th percentile:        3.30  
  sum:                     39965.67  
  
Threads fairness:  
  events (avg/stddev):     12054.0000/224.07  
  execution time (avg/stddev): 9.9914/0.00
```

b. 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 Command O/P Screenshot-

```
root@69d9f068f00a: /
+
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:        3604.37
      fsyncs/s:        4807.81

Throughput:
      read, MiB/s:      0.00
      written, MiB/s:   56.32

General statistics:
      total time:       10.4298s
      total number of events: 85706

Latency (ms):
      min:              0.01
      avg:              1.88
      max:              562.65
      95th percentile: 7.84
      sum:              161255.32

Threads fairness:
      events (avg/stddev): 5356.6250/256.59
      execution time (avg/stddev): 10.0785/0.09
```

c. Test 3:

```
$ sysbench --threads=8 fileio --file-total-size=4G --file-test-mode=rndrw prepare
```

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

```
$ sysbench --threads=8 fileio --file-total-size=4G --file-test-mode=rndrw cleanup
```

Run Command O/P Screenshot

```
root@69d9f068f00a: /  
4GiB 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:                1641.15  
  writes/s:               1093.57  
  fsyncs/s:               3591.42  
  
Throughput:  
  read, MiB/s:            25.64  
  written, MiB/s:         17.09  
  
General statistics:  
  total time:              10.1549s  
  total number of events:  63234  
  
Latency (ms):  
  min:                     0.00  
  avg:                     1.26  
  max:                     266.85  
  95th percentile:        4.82  
  sum:                     79965.97  
  
Threads fairness:  
  events (avg/stddev):     7904.2500/77.87  
  execution time (avg/stddev): 9.9957/0.00
```

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 upto 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=30000 --time=10 run

	min	average	max	no. of events
QEMU VM	4.94	5.19	19.86	1926
Docker	4.90	4.98	6.73	2007

Test 2:

\$ sysbench cpu --threads=2 --cpu-max-prime=300000 --time=30 run

	min	average	max	no. of events
QEMU VM	126.05	130.84	166.80	460
Docker	127.87	129.73	164.67	464

Test 3:

\$ sysbench cpu --threads=16 --cpu-max-prime=5000000 --time=50 run

	min	average	max	no. of events
QEMU VM	54526.41	54664.84	54722.37	16
Docker	55624.08	66333.14	69899.46	16

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=2G --file-test-mode=rndwr run
```

	min	average	max	no. of events
QEMU VM	0.00	1.89	135.70	21082
Docker	0.00	0.83	221.44	48216

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	2.93	190.80	54643
Docker	0.01	1.88	562.65	85706

Test 3:

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

	min	average	max	no. of events
QEMU VM	0.00	3.34	65.25	23939
Docker	0.00	1.26	266.85	63234

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 -

1. The performance of docker containers is faster than the QEMU VM given that they have been provided with the same resources such as CPU and RAM.
2. In all three test cases of CPU tests, the docker container was found to be faster than the QEMU VM.
3. In all three test cases of File-IO tests, the docker container was found to be faster than the QEMU VM.
4. For CPU Tests, the number of events decreases as the thread numbers increase.
5. For CPU Tests, as the number of threads and time given to perform a case is increased, the latency values, min, max, and the average also increases.
6. For File-IO Tests, the number of events performed in both docker container and QEMU VM is more in sequential order than the random order operation for all read, write, or read-write operations.
7. When the number of threads increases in both, for docker containers the max latency parameter value increases as thread numbers increase while the QEMU VMs same parameter value decreases as thread number increases.

Experiment Shell Scripts

Shell Scripts for CPU and FILE-IO Test -

1. Shell Script for CPU Tests -

\$./cpu_test.sh

```
#!/bin/bash
```

```
echo "Hey, you are in $0, and about to test cpu!"
```

```
PRIMES_UPTO=("30000" "300000" "5000000")
```

```
MAX_TIME=("10" "30" "50")
```

```
THREADS=("1" "2" "16")
```

```
TEST_RUNS=5
```

```
TEST_CASES=3
```

```
for ((i=0; i<$TEST_CASES;i++))
```

```
do
```

```
    echo "*****Starting ${i+1}st Test  
Case*****"
```

```
    for ((j=1; j <= $TEST_RUNS; j++ ))
```

```
    do
```

```
        echo "Running ${j}st run of Test Case ${i+1}"
```

```
        sysbench cpu --threads=${THREADS[$i]}
```

```
--cpu-max-prime=${PRIMES_UPTO[$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 "Hey, you are in $0, and about to test fileio"
```

```
THREADS=("4" "16" "8")
```

```
FILE_TOTAL_SIZES=("2G" "5G" "4G")
```

```
TEST_MODE=("rndwr" "seqrewr" "rndrw")
```

```
TEST_RUNS=5
```

```
TEST_CASES=3
```

```
for ((i=0; i<$TEST_CASES;i++))
```

```
do
```

```
    echo "*****Starting ${i+1}st Test  
Case*****"
```

```
    for ((j=1; j <=$TEST_RUNS; j++ ))
```

```
    do
```

```
        echo "Running ${j}st run of Test Case ${i+1}"
```

```
        sysbench --threads=${THREADS[$i]} fileio
```

```
--file-total-size=${FILE_TOTAL_SIZES[$i]} --file-test-mode=${TEST_MODE[i]}  
prepare
```

```
        sysbench --threads=${THREADS[$i]} fileio
```

```
--file-total-size=${FILE_TOTAL_SIZES[$i]} --file-test-mode=${TEST_MODE[i]} run
```

```
        sysbench --threads=${THREADS[$i]} fileio
```

```
--file-total-size=${FILE_TOTAL_SIZES[$i]} --file-test-mode=${TEST_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 -

pranav@fedora:~/Documents/github/cloud-computing-cours

top - 11:56:01 up 2:33, 1 user, load average: 1.08, 1.10, 1.21

Tasks: 280 total, 1 running, 279 sleeping, 0 stopped, 0 zombie

%Cpu(s): 53.7 us, 1.0 sy, 0.0 ni, 44.7 id, 0.0 wa, 0.4 hi, 0.2 si, 0.0 st

MiB Mem : 7854.3 total, 201.1 free, 4123.7 used, 3529.4 buff/cache

MiB Swap: 16046.0 total, 16042.2 free, 3.8 used. 2856.5 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
13028	root	20	0	3224884	1.2g	51556	S	202.3	15.4	2:26.36	qemu-system-x86
2561	pranav	20	0	790316	74284	52772	S	6.6	0.9	0:55.13	gnome-terminal-
1879	pranav	20	0	4736856	240152	142236	S	6.3	3.0	10:11.56	gnome-shell
12118	pranav	20	0	573916	71396	48464	S	2.0	0.9	0:09.40	Xwayland
2089	pranav	20	0	528204	11852	6232	S	0.7	0.1	0:47.93	ibus-daemon
2209	pranav	20	0	601880	48052	23196	S	0.7	0.6	0:12.91	ibus-extension-
13484	pranav	20	0	226448	4176	3284	R	0.7	0.1	0:00.07	top
1743	pranav	20	0	7392	5180	2440	S	0.3	0.1	0:02.11	dbus-broker
12084	root	20	0	0	0	0	I	0.3	0.0	0:01.46	kworker/1:1-events
1	root	20	0	173396	17148	10280	S	0.0	0.2	0:04.42	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	slub_flushwq
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
10	root	0	-20	0	0	0	I	0.0	0.0	0:01.03	kworker/0:1H-events_highpri
11	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
16	root	20	0	0	0	0	S	0.0	0.0	0:00.12	ksoftirqd/0
17	root	20	0	0	0	0	I	0.0	0.0	0:04.48	rcu_preempt
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	migration/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
22	root	rt	0	0	0	0	S	0.0	0.0	0:00.18	migration/1
23	root	20	0	0	0	0	S	0.0	0.0	0:05.31	ksoftirqd/1
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-events_highpri
26	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2

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

```
pranav@fedora:~/Documents/github/cloud-computing-cours
top - 11:57:17 up 2:34, 1 user, load average: 1.32, 1.19, 1.24
Tasks: 282 total, 1 running, 281 sleeping, 0 stopped, 0 zombie
%Cpu(s): 7.1 us, 2.4 sy, 0.0 ni, 89.4 id, 0.0 wa, 0.7 hi, 0.4 si, 0.0 st
MiB Mem : 7854.3 total, 274.3 free, 4090.3 used, 3489.7 buff/cache
MiB Swap: 16046.0 total, 16042.2 free, 3.8 used, 2901.7 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 13028 root        20   0 3224884    1.2g  51352 S   12.2   15.4   3:43.92 qemu-system-x86
   1879 pranav     20   0 4749376   240628 142264 S   11.9    3.0  10:16.98 gnome-shell
   2561 pranav     20   0  790196    74076  52772 S    5.6    0.9   0:56.00 gnome-terminal-
  12118 pranav     20   0  570732    71396  48464 S    5.0    0.9   0:10.07 Xwayland
   2913 pranav     20   0 4130904   510496 202284 S    0.7    6.3  19:52.36 firefox
   9444 root        20   0      0      0      0 I    0.7    0.0   0:02.42 kworker/u8:1-events_unbound
  13484 pranav     20   0 226448    4176   3284 R    0.7    0.1   0:00.40 top
    23 root        20   0      0      0      0 S    0.3    0.0   0:05.36 ksoftirqd/1
   685 systemd+   20   0   17664    8012   7032 S    0.3    0.1   0:14.69 systemd-oomd
  1704 pranav     20   0   23224   14328  10168 S    0.3    0.2   0:04.31 systemd
  2078 root        20   0   261568   29688   7940 S    0.3    0.4   0:08.35 sssd_kcm
  2089 pranav     20   0   528204   11852   6232 S    0.3    0.1   0:48.19 ibus-daemon
  2209 pranav     20   0   601880   48052  23196 S    0.3    0.6   0:12.96 ibus-extension-
  3225 pranav     20   0  2720948 148036  89344 S    0.3    1.8   0:07.51 Isolated Web Co
  4500 pranav     20   0 3269220 436308 131924 S    0.3    5.4  17:05.06 Isolated Web Co
   8598 root        20   0      0      0      0 I    0.3    0.0   0:01.72 kworker/0:0-events
  11017 root        20   0      0      0      0 I    0.3    0.0   0:00.77 kworker/2:3-events
  12084 root        20   0      0      0      0 I    0.3    0.0   0:01.63 kworker/1:1-events
    1 root        20   0  173396   17148  10280 S    0.0    0.2   0:04.44 systemd
    2 root        20   0      0      0      0 S    0.0    0.0   0:00.01 kthreadd
    3 root         0 -20      0      0      0 I    0.0    0.0   0:00.00 rcu_gp
    4 root         0 -20      0      0      0 I    0.0    0.0   0:00.00 rcu_par_gp
    5 root         0 -20      0      0      0 I    0.0    0.0   0:00.00 slub_flushwq
    6 root         0 -20      0      0      0 I    0.0    0.0   0:00.00 netns
    8 root         0 -20      0      0      0 I    0.0    0.0   0:00.00 kworker/0:0H-events_highpri
   10 root         0 -20      0      0      0 I    0.0    0.0   0:01.03 kworker/0:1H-events_highpri
   11 root         0 -20      0      0      0 I    0.0    0.0   0:00.00 mm_percpu_wq
   13 root        20   0      0      0      0 I    0.0    0.0   0:00.00 rcu_tasks_kthread
   14 root        20   0      0      0      0 I    0.0    0.0   0:00.00 rcu_tasks_rude_kthread
   15 root        20   0      0      0      0 I    0.0    0.0   0:00.00 rcu_tasks_trace_kthread
```

3. Docker Container Running Sysbench CPU Test Case -

```
root@69d9f068f00a: /  
top - 18:22:27 up 1:59, 0 users, load average: 4.85, 1.49, 1.07  
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 57.9 us, 2.2 sy, 0.8 ni, 38.3 id, 0.1 wa, 0.5 hi, 0.2 si, 0.0 st  
MiB Mem : 7854.3 total, 210.1 free, 2924.4 used, 4719.8 buff/cache  
MiB Swap: 16046.0 total, 16045.5 free, 0.5 used. 4154.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
307	root	20	0	32492	9400	7932	S	200.3	0.1	0:43.97	sysbench
1	root	20	0	4624	3708	3104	S	0.0	0.0	0:00.20	bash
288	root	20	0	4624	3612	3024	S	0.0	0.0	0:00.03	bash
306	root	20	0	7312	3336	2772	R	0.0	0.0	0:00.01	top

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

```
root@69d9f068f00a: /  
top - 18:26:34 up 2:03, 0 users, load average: 0.76, 1.97, 1.49  
Tasks: 3 total, 1 running, 2 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 2.9 us, 0.6 sy, 0.0 ni, 95.8 id, 0.2 wa, 0.3 hi, 0.2 si, 0.0 st  
MiB Mem : 7854.3 total, 322.5 free, 2931.7 used, 4600.2 buff/cache  
MiB Swap: 16046.0 total, 16045.5 free, 0.5 used. 4119.7 avail Mem  


| PID | USER | PR | NI | VIRT | RES  | SHR  | S | %CPU | %MEM | TIME+   | COMMAND |
|-----|------|----|----|------|------|------|---|------|------|---------|---------|
| 1   | root | 20 | 0  | 4624 | 3708 | 3104 | S | 0.0  | 0.0  | 0:00.20 | bash    |
| 288 | root | 20 | 0  | 4624 | 3612 | 3024 | S | 0.0  | 0.0  | 0:00.03 | bash    |
| 306 | root | 20 | 0  | 7312 | 3336 | 2772 | R | 0.0  | 0.0  | 0:00.07 | top     |


```

Table of comparison between above four cases -

1. Docker Container CPU -

	Sysbench Running	Sysbench Not-Running
User level CPU usage %	57.9	2.9
Kernel level CPU usage %	2.2	0.6
Idle CPU %	38.3	95.8

2. Host OS CPU Usage - Docker Container -

	Sysbench Running	Sysbench Not-Running
User level CPU usage %	57.1	0.2
Kernel level CPU usage %	1.2	0.3
Idle CPU %	40.9	99.3

3. Host OS CPU Usage - QEMU VM -

	Sysbench Running	Sysbench Not-Running
User level CPU usage %	53.7	7.1
Kernel level CPU usage %	1.0	2.4
Idle CPU %	44.7	89.4

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.

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 sysstat
```

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 -

1. Prepare Stage -

```
$ sysbench --threads=16 fileio --file-total-size=4G --file-test-mode=rndrw prepare
```

Screenshot of Disk IO Utilization, latency, throughput of host os when above command is running in QEMU VM -

Device	f/s	f_await	aqu-sz	%util	r/s	rMB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s	wMB/s	wrqm/s	%wrqm	w_await	wareq-sz	d/s	dMB/s	drqm/s	%drqm	d_await	dareq-sz
sda	4.00	2.75	3.43	81.80	0.00	0.00	0.00	0.00	0.00	0.00	43.00	40.45	4.00	8.51	79.44	963.26	0.00	0.00	0.00	0.00	0.00	0.00
sdb	4.00	2.75	3.14	74.10	0.00	0.00	0.00	0.00	26.50	22.00	38.00	36.79	0.00	0.00	79.58	991.26	0.00	0.00	0.00	0.00	0.00	0.00
sdc	0.00	0.00	0.00	1.10	0.00	0.00	0.00	0.00	0.00	0.00	173.00	0.68	0.00	0.00	0.01	4.00	0.00	0.00	0.00	0.00	0.00	0.00
sdd	6.00	2.67	1.65	51.60	0.00	0.00	0.00	0.00	0.00	0.00	133.00	117.12	1.00	0.75	12.29	981.74	0.00	0.00	0.00	0.00	0.00	0.00
sde	4.00	6.00	3.36	81.10	1.00	0.06	0.00	0.00	6.00	64.00	49.00	43.04	2.00	3.92	67.94	899.35	0.00	0.00	0.00	0.00	0.00	0.00
sde	2.00	0.50	2.76	71.40	1.00	0.12	0.00	0.00	9.00	128.00	54.00	54.88	1.00	1.82	50.87	1040.74	0.00	0.00	0.00	0.00	0.00	0.00
sde	8.00	0.38	2.12	59.30	0.00	0.00	0.00	0.00	0.00	0.00	119.00	103.68	2.00	1.65	17.84	892.20	0.00	0.00	0.00	0.00	0.00	0.00

[illegible]

```
$ sysbench --threads=16 fileio --file-total-size=4G --file-test-mode=rndrw run
```

```
Activities Terminal Oct 17 4:12 PM
pranav@fedora:~/Documents/temp — lsof -xzm 1
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
sda 0.00 0.00 2.00 0.01 0.00 0.00 0.50 4.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.10
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
zram0 0.00 0.00 1.00 0.00 0.00 0.00 4.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.10
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
sda 0.00 0.00 0.00 0.00 0.00 0.00 0.00 2.00 0.03 6.00 75.00 0.50 16.00 0.00 0.00 0.00 0.00 0.00 0.00
2.00 0.00 0.00 0.20
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
Device      r/s      rMB/s    rrqm/s    %rrqm  r_await  rareq-sz    w/s    wMB/s    wrqm/s    %wrqm  w_await  wareq-sz    d/s    dMB/s    drqm/s    %drqm  d_await  dareq-sz
f/s f_await aqu-sz %util
```


Screenshot of Disk IO Utilization, latency, throughput of host os when above command is running in Docker Container -

Activities Terminal Oct 17 4:26 PM pranav@fedora: ~/Documents/temp — iostat -dxzm 1

sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00	96.00	0.44	2.00	2.04	0.42	4.67	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.04	0.40															
Device	r/s	rMB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s	wMB/s	wrqm/s	%wrqm	w_await	wareq-sz	d/s	dMB/s	drqm/s	%drqm	d_await	dareq-sz	
f/s	f_await	aqm-sz	%util																
Device	r/s	rMB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s	wMB/s	wrqm/s	%wrqm	w_await	wareq-sz	d/s	dMB/s	drqm/s	%drqm	d_await	dareq-sz	
f/s	f_await	aqm-sz	%util																
Device	r/s	rMB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s	wMB/s	wrqm/s	%wrqm	w_await	wareq-sz	d/s	dMB/s	drqm/s	%drqm	d_await	dareq-sz	
f/s	f_await	aqm-sz	%util																
sda	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.05	12.00	85.71	0.50	28.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2.00	0.00	0.00	0.20															
Device	r/s	rMB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s	wMB/s	wrqm/s	%wrqm	w_await	wareq-sz	d/s	dMB/s	drqm/s	%drqm	d_await	dareq-sz	
f/s	f_await	aqm-sz	%util																
sda	164.00	6.58	27.00	14.14	0.66	41.10	26.00	0.11	24.00	48.00	1.00	4.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	11.00	0.36	0.14	6.00															
Device	r/s	rMB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s	wMB/s	wrqm/s	%wrqm	w_await	wareq-sz	d/s	dMB/s	drqm/s	%drqm	d_await	dareq-sz	
f/s	f_await	aqm-sz	%util																
sda	2021.00	31.83	0.00	0.00	2.05	16.13	10860.00	46.59	1122.00	9.36	0.84	4.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1563.00	0.53	14.12	101.70															

3. Cleanout Stage -

\$ sysbench --threads=16 fileio --file-total-size=4G --file-test-mode=rndrw cleanup

Screenshot of Disk IO Utilization, latency, throughput of host os when above command is running in QEMU VM -

Table of I/O Throughput, I/O Latency, and I/O Disk Utilization Host OS in case of Docker Container based on above screenshots -

	I/O Throughput	I/O Latency	I/O Disk Utilization
Prepare Stage	43.04	6.00	81.10
Run Stage	0.03	0.00	0.20
Cleanup Stage	0.12	0.56	0.10

Table of I/O Throughput, I/O Latency, and I/O Disk Utilization Host OS in case of QEMU VM based on above screenshots -

	I/O Throughput	I/O Latency	I/O Disk Utilization
Prepare Stage	98.23	8.00	93.20
Run Stage	0.50	0.05	0.20
Cleanup Stage	0.00	0.00	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 -

```
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"

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

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

  # Provision Settings
  config.vm.provision "shell", path: "vagrant_script.sh"
end
```

Dockerfile -

```
FROM pranav2306/sysbench-ubuntu:version1

COPY docker_script.sh /docker_script.sh
COPY cpu_test.sh /cpu_test.sh
COPY fileio_test.sh /fileio_test.sh

RUN chmod +x docker_script.sh
RUN chmod +x cpu_test.sh
RUN chmod +x fileio_test.sh

ENTRYPOINT bash docker_script.sh
```

Resources

- **[MAIN LINK]** The link to the GitHub repository to access the homework - <https://github.com/panu2306/cloud-computing-course/tree/main/homeworks/hw1>
- The link to the docker image of created image in this experiment - <https://hub.docker.com/repository/docker/pranav2306/sysbench-ubuntu/>
Or
simply pull using -
`$ docker pull pranav2306/sysbench-ubuntu:version1`
- The link for Vagrantfile - <https://github.com/panu2306/cloud-computing-course/blob/main/homeworks/hw1/gemu/Vagrantfile>
- The link for Dockerfile - <https://github.com/panu2306/cloud-computing-course/blob/main/homeworks/hw1/docker/Dockerfile>
- The link for CPU Tests shell script - https://github.com/panu2306/cloud-computing-course/blob/main/homeworks/hw1/shell_scripts/cpu_test.sh
- The link for FILEIO Tests shell script - https://github.com/panu2306/cloud-computing-course/blob/main/homeworks/hw1/shell_scripts/fileio_test.sh