

COEN -241 Cloud Computing

HW1: System Vs OS Virtualization Report

**Submitted by: Sumit Agrawal
(W1631657)**

Host System Configuration:

1	Chip	Apple M1 (Apple Silicon)
2	CPU	8 (4 performance and 4 efficiency)
3	Memory	8 GB
4	Free disk space	110GB
5	Operating System	Mac OS Monterey

We will be using Three different configuration for QEMU and Docker, it is explained in the report further.

QEMU Installation and creation of QEMU image

In this section, we will walk through the steps of installing QEMU and creating a QEMU image. This image would be used for all our experiments.

1. Downloading Ubuntu Guest Virtual Machine

We will first download the required ISO image based on our system configuration. The system's architecture is:

```
Last login: Tue Jan 24 09:34:31 on console  
[base] sumit@Sumits-MacBook-Air ~ % arch  
arm64
```

Therefore, we will be using Arm64v8 architecture. To download a ISO file check the below link.

ARM (Apple Silicon) - [Ubuntu 20.04 Server for ARM](#)

Create one folder to store the ubuntu image and a ISO file

```
(base) sumit@Sumits-MacBook-Air ~ % cd Desktop/Winter2023/Cloud\ computing/qemu  
(base) sumit@Sumits-MacBook-Air qemu % ls  
ubuntu-20.04.5-live-server-arm64.iso
```

2. Install QEMU

Run the following command in the terminal. homebrew must be installed before running this command.

```
brew install qemu
```

```
(base) sumit@Sumits-MacBook-Air ~ % brew install qemu
Running `brew update --auto-update'...
==> Auto-updated Homebrew!
Updated 2 taps (homebrew/core and homebrew/services).
==> New Formulae
ancient          copa           liblerc          okta-awscli      prs
aws-sam-cli      dstack         libsa1s          pari-nflistdata  scriptisto
check-jsonschema grayskull     libmddc         pipdeptree       xcdiff
clang-build-analyzer kwok          libmless        plz-cli          zsh-autopair

You have 2 outdated formulae installed.
You can upgrade them with brew upgrade
or list them with brew outdated.

==> Fetching dependencies for qemu: capstone, pcre2, gettext, glib, ca-certificates, gmp, bdw-gc, m4, libtoo
l, libunistring, pkg-config, readline, guile, libidn2, libtasn1, nettle, pil-kit, libevent, libnghttp2, unbo
und, gnutls, jpeg-turbo, libpng, libslirp, libssh, libusb, lzo, ncurses, pixman, snappy, vde, lz4, xz and zs
td
==> Fetching capstone
==> Downloading https://ghcr.io/v2/homebrew/core/capstone/manifests/4.8.2
#####
==> Downloading https://ghcr.io/v2/homebrew/core/capstone/blobs/sha256:53b534ac8a71300d7e005cb7552a2778ff6d0
e3bc4aa64
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:53b534ac8a71300d7e005cb
7552a2778
#####
==> Fetching pcre2
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/manifests/10.42
#####
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/blobs/sha256:23ce93cf86bd4816b7d039efa0a5d68c751bce3f
552a8cbf4
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:23ce93cf86bd4816b7d039e
```

If homebrew is not installed, then check below link or use this command to install it.
[\(https://brew.sh/\)](https://brew.sh/):

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

3. Check QEMU version

We have to make sure that the QEMU version should be 6.2.0+ which is the minimum supported by Apple Silicon.

```
qemu-system-x86_64 --version
```

We have a QEMU 7.2.0 version.

```
[base) sumit@Sumits-MacBook-Air ~ % qemu-system-x86_64 --version
QEMU emulator version 7.2.0
Copyright (c) 2003-2022 Fabrice Bellard and the QEMU Project developers
(base) sumit@Sumits-MacBook-Air ~ %
```

4. Create QEMU image

Then we'll make a QEMU image. This image will be created in the same folder where the Ubuntu guest virtual machine ISO file was copied. In the terminal, type the following command:

```
qemu-img create ubuntu.img 10G -f qcow2
```

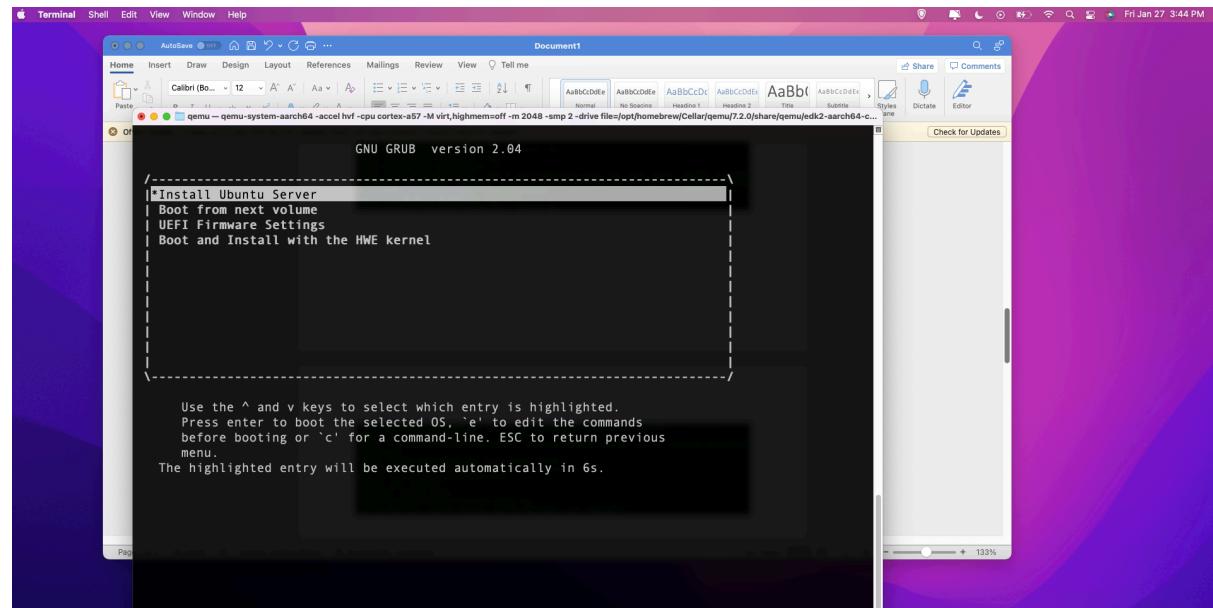
```
(base) sumit@Sumits-MacBook-Air qemu % ls
ubuntu-20.04.5-live-server-arm64.iso
(base) sumit@Sumits-MacBook-Air qemu % /opt/homebrew/bin/qemu-img create ubuntu.img 10G -f qcow2
Formatting 'ubuntu.img', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=
10737418240 lazy_refcounts=off refcount_bits=16
(base) sumit@Sumits-MacBook-Air qemu % ls
ubuntu-20.04.5-live-server-arm64.iso      ubuntu.img
```

5. Install Ubuntu VM

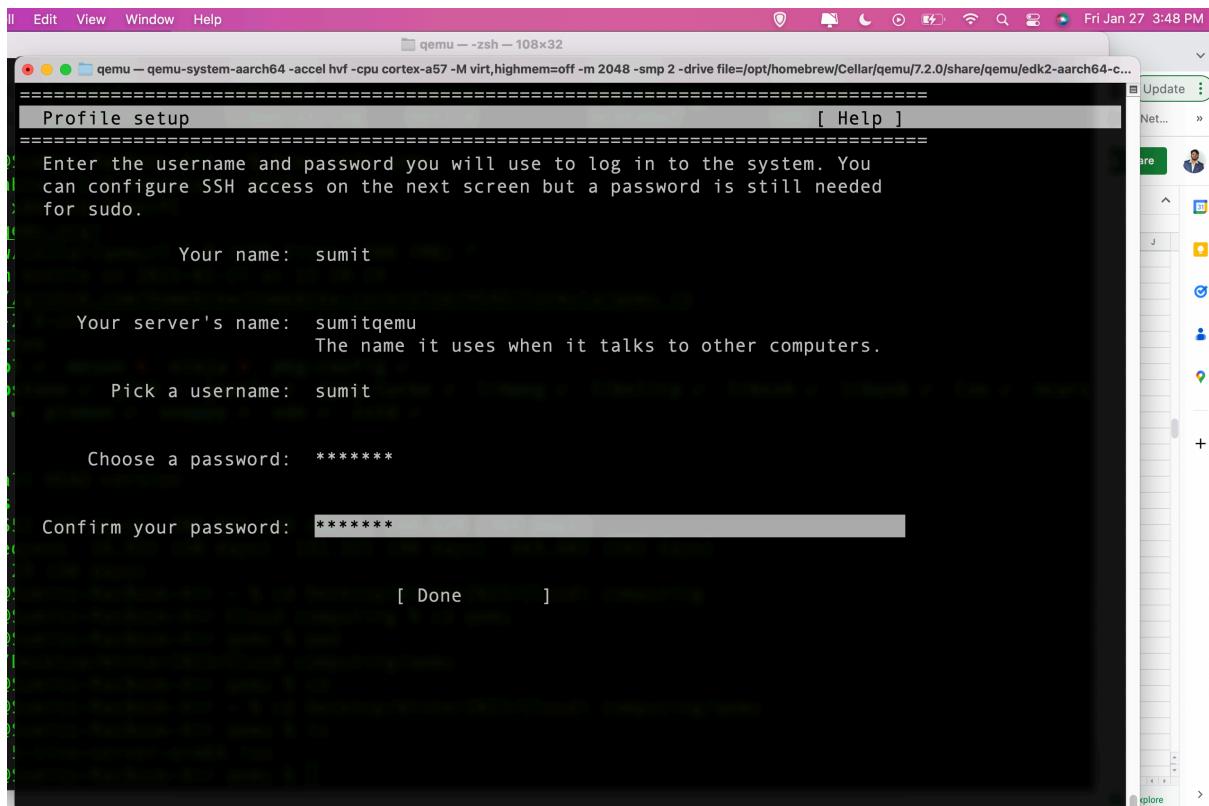
Run the below command to install Ubuntu VM:

```
/opt/homebrew/bin/qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048 -smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-
code.fd,if=pflash,format=raw,readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="dummyserial" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-cdrom ubuntu-20.04.5-live-server-arm64.iso \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic
```

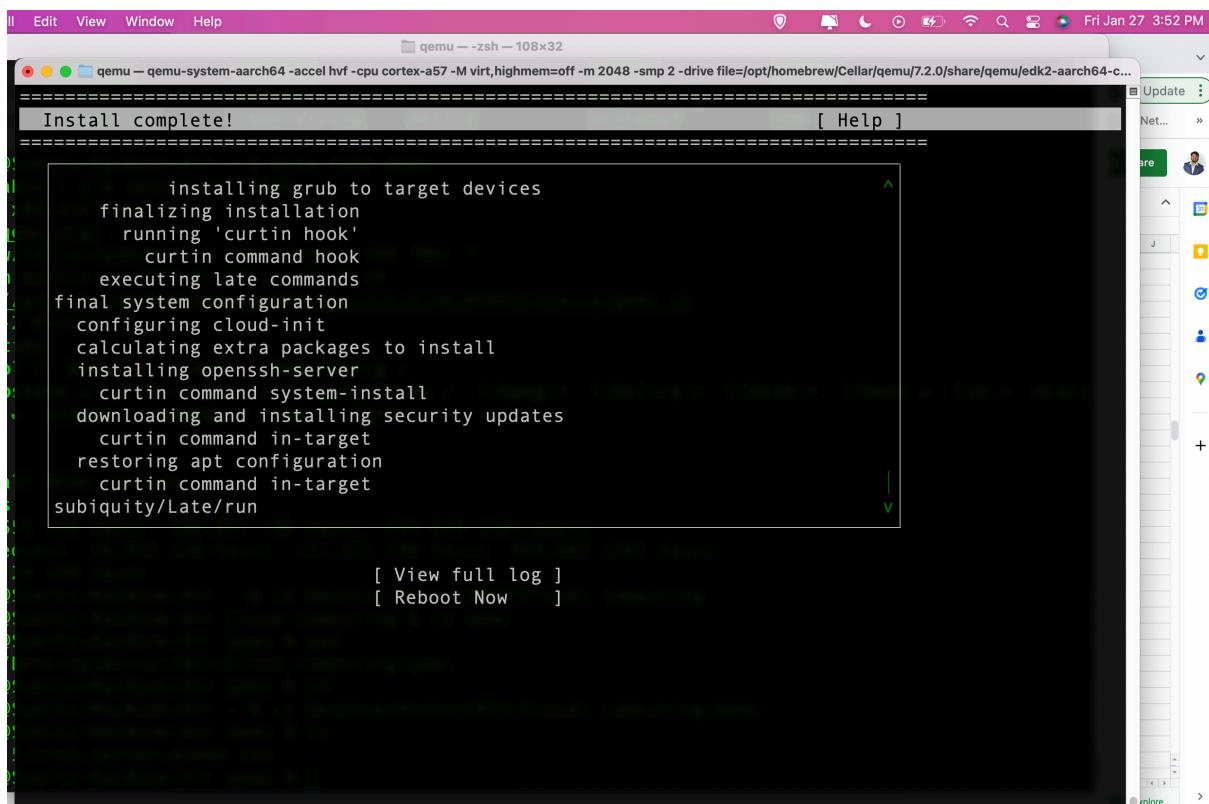
```
(base) sumit@Sumits-MacBook-Air qemu % /opt/homebrew/bin/qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048 -smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,
readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="dummyserial" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-cdrom ubuntu-20.04.5-live-server-arm64.iso \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic
```



While installing it, you must provide a server name, username, and password which will be used to login.



Then Select reboot option.



We can try logging in with the following command (we must remove the cdrom argument). It will prompt you for your username and password:

```
/opt/homebrew/bin/qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048 -smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-
code.fd,if=pflash,format=raw,readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="dummyserial" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic
```

```
(base) sumit@Sumits-MacBook-Air qemu % /opt/homebrew/bin/qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048 -smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="dummyserial" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic
```

```
[ OK ] Finished Update UTMP about System Runlevel Changes.
[!-] ci-info: no authorized SSH keys fingerprints found for user sumit.
<14>Jan 28 00:01:51 cloud-init: #####-BEGIN SSH HOST KEY FINGERPRINTS-#####
<14>Jan 28 00:01:51 cloud-init: 1024 SHA256:1VFJkCbM127wun2DSY7xTlYI3sb3IcDsx4dUJnxab5g root@sumitqemu (DSA)
<14>Jan 28 00:01:51 cloud-init: 256 SHA256:OMLbYFUam/s3mCJH+bCTPyriXTStzOKX++y4du7onBs root@sumitqemu (ECDSA)
<14>Jan 28 00:01:51 cloud-init: 256 SHA256:jeLRdbHwz31QMuOfBZFlepOF6yLbXXdUEgdmoRjnerU root@sumitqemu (ED25519)
<14>Jan 28 00:01:51 cloud-init: 3072 SHA256:1gwCY1Id6uPrUKIT+FBrmf1eqVrxDGJeKdx2QeqCOZU root@sumitqemu (RSA)
<14>Jan 28 00:01:51 cloud-init: ----END SSH HOST KEY FINGERPRINTS----#
<14>Jan 28 00:01:51 cloud-init: #####-BEGIN SSH HOST KEY KEYS-#####
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAIBmlzdHAyNTYAAABBBC3MlwYfoXeV00wIuSnE7KU9rnM5xOfHY+j+qxnaAjgnvyb
I6b+Myu5bhvcHfuY0TSZ16yC06gmq5XuFxwMqIsWU= root@sumitqemu
ssh-ed25519 AAAAC3NzaC1ZDI1NTE5AAAIAHxeP6SyIo8Zw3Vbm5PQx48HrpCy2H805bQYBhQ+Fquh root@sumitqemu
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAhgQDdUnKRHTrYmQ8YP02ENULrKCIyJVRjGvYQo+ypMeeph7xNx75xMS8ojK6h/1LATnCs+vx07j0VtZvyQ
9G6jZwYd1y0MptzT/2DaApizGlpwIpq7jmfej5VoOuyrWkWMSVJxCg/BwW10eoSjfjEFBKHZiTgjW8Du691Qm2jXm5jKK/w3JKBH6cg+5DTjv+xIjf
0kr7Fkiqfw09PHgPwL5THkIBbvAdBG5RQJZSGp3tAN5lnCMJAMbFAlcZBkVEcXFns5J4jqecIaBbn1xZTooej3MK2V0SwK0Ej1wFc929LKAhJhlj9wyLZW
8eZ/upfi3y4M59777WQwpqvTU9S83v6pgD8/aBnpT04sAcUe+10d51iWyWMm1NR99QEwMDkpX6MMsLT+E3YWqUohbYwLwZZgr+GkWdeV5PUcQo+/pwRGQU
YjsZGCUV2nPwiLeyffRnbisXkCR+ASwzSYCVV2q8LGnTtOyqZ8JmCoIRy3YgDThF4Xp5E8QoXBII+EU= root@sumitqemu
----END SSH HOST KEY KEYS----#
[ 12.565199] cloud-init[1454]: Cloud-init v. 22.2-0ubuntu1~20.04.3 running 'modules:final' at Sat, 28 Jan 2023 00:01:51 +0000. Up 12.49 seconds.
[ 12.565607] cloud-init[1454]: Cloud-init v. 22.2-0ubuntu1~20.04.3 finished at Sat, 28 Jan 2023 00:01:51 +0000. Data source DataSourceNone. Up 12.56 seconds
[ 12.566078] cloud-init[1454]: 2023-01-28 00:01:51,466 - cc_final_message.py[WARNIN]: Used fallback datasource
[ OK ] Finished Execute cloud user/final scripts.
[ OK ] Reached target Cloud-init target.

Ubuntu 20.04.5 LTS sumitqemu ttyAMA0

sumitqemu login: sumit
```

Below is the explanation of arguments used in the command.

-accel	Activates an accelerator. Kvm, xen, hax, hvf, nvmm, whpx, or tcg could be available depending on the target architecture.
-m	This symbol represents memory. We are currently using 2G as the argument value, which means that we are allocating 2GB RAM memory to our VM.
-smp	Denotes the number of cores. We have currently set the argument value to 2, indicating that we have allocated two cores to our VM.

-cdrom	Use file as CD-ROM image (you cannot use -hdc and -cdrom at the same time). You can use the host CD-ROM by using /dev/cdrom as filename.
-device	Mainly to add a device driver where prop=value sets the driver properties.

When running for different configurations, we would update the -m and -smp values.

Docker Installation, Building Docker Image, and Enabling Docker Containers

1. Installation

Now we'll walk through the Docker installation process, which is used to create and manage containers, which are then used to run and manage various images.

Documentation can be found at <https://docs.docker.com/desktop/mac/apple-silicon/>

We will use the above link to install Docker desktop, which includes the Docker cli and Docker engine by default. Run the following commands in terminal to see if Docker is properly installed:

```
docker pull hello-world
docker images
docker run hello-world
```

```
(base) sumit@Sumits-MacBook-Air ~ % docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
7050e35b49f5: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
(base) sumit@Sumits-MacBook-Air ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
arm64v8/ubuntu  20.04    c9eb527b091d  19 hours ago  65.7MB
hello-world     latest    46331d942d63  10 months ago  9.14kB
(base) sumit@Sumits-MacBook-Air ~ % docker run hello-world

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

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

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

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

2. Basic docker image creation and running docker image:

We will make a docker image from dockerfile to run experiments. Below is basic step to create a docker image:

Terminal 1: docker images sudo docker run -it --entrypoint "bin/bash/ arm64v8/ubuntu:20.04 apt update apt install <package>	Terminal 2: docker ps docker commit <container_id> <image_name> docker images docker history <image_name>
--	---

Terminal 1:

```
sumit — root@eb768e50ff89: / — com.docker.cli - docker run -it --entrypoint /bin/bash arm64v8/ubuntu:20.04 — 88x37
(base) sumit@Sumits-MacBook-Air ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
arm64v8/ubuntu  20.04   c9eb527b091d  19 hours ago  65.7MB
hello-world     latest   46331d942d63  10 months ago  9.14kB
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --entrypoint "/bin/bash" arm64v8/ubuntu:20.04
root@eb768e50ff89:/# █
```

Terminal 2:

```
...t@eb768e50ff89: / — com.docker.cli - docker run -it --entrypoint /bin/bash arm64v8/ubuntu:20.04 ...
Last login: Wed Feb 1 22:27:51 on ttys003
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS      NAMES
eb768e50ff89   arm64v8/ubuntu:20.04   "/bin/bash"   2 minutes ago   Up 2 minutes   friendly_rosalind
2faa3547a85d   arm64v8/ubuntu:20.04   "/bin/bash"   8 hours ago    Up 8 hours   confident_bohr
(base) sumit@Sumits-MacBook-Air ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
arm64v8/ubuntu  20.04   c9eb527b091d  19 hours ago  65.7MB
hello-world     latest   46331d942d63  10 months ago  9.14kB
(base) sumit@Sumits-MacBook-Air ~ % docker commit eb768e50ff89 new_docker_image
sha256:4ffcb1d0913356678ba70c2aec1b96c43988a05ed1797e5621356566e9fec776
(base) sumit@Sumits-MacBook-Air ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
new_docker_image  latest   4ffcb1d09133  5 seconds ago  65.7MB
arm64v8/ubuntu  20.04   c9eb527b091d  19 hours ago  65.7MB
hello-world     latest   46331d942d63  10 months ago  9.14kB
(base) sumit@Sumits-MacBook-Air ~ %
```

Following are descriptions of the docker commands:

docker pull image-name:tag	To pull a docker image
docker images	To list out all docker images
docker run image-name	To run a docker image
docker run [container] image [command]	To run a container
docker ps -a	To list out all containers; running as well as exited.
docker commit <container_id> <image_name>	To commit a container's file changes or settings into a new image
docker history <image_name>	To show history of an image

docker inspect container id	To check all details about containers
docker system prune -a	To delete all docker images
Docker exec -it containerID /bin/bash	To go inside the running container

3. Sysbench Installation.

We must make sure that we have a same version of sysbench in QEMU and in Docker. To install use the below command

For Docker, we are using the sysbench library package:
Link: <https://github.com/akopytov/sysbench>

```
apt update;
apt install sysbench;
```

Test Conditions: To conduct the experiment with the VM, we will consider three test conditions showed in the table below. To ensure consistency between results, the same test conditions will be used for Docker as well as for QEMU

Cores	Memory allocation
2	2 GB
3	3 GB
6	3 GB

Proof of experiment - Reports and Findings

In this section, we will look at test cases that are specifically related to sysbench CPU and File I/O commands. We will also test different QEMU VM configurations to see if we get different results by changing the VM configurations.

1. CPU Testing

We will use the following three test cases to evaluate CPU performance between QEMU and Docker. For our testing, we will use the sysbench command and the test cases listed below:

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

1. max-prime = 1000 and time = 30 seconds
2. max-prime = 10,000 and time = 30 seconds
3. max-prime = 100,000 and time = 30 seconds

2. FILEIO Testing

For File I/O testing we will be using random read/write (rndrw). Also, for our testing purposes, we will use the following sysbench commands, and changing the values for different configuration.

```
sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndwr prepare  
sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw run  
sysbench --num-threads=16 --test=fileio --file-total-size=3G --time=30 --file-test-mode=rndrw cleanup
```

3. Results

1. Configuration 1: 2 GB RAM with 2 Cores

1.1 CPU Testing

To conduct these experiments, I wrote three bash scripts, each with a different max-prime value and five iterations:

```
[● ● ● ● qemu — qemu-system-aarch64 -accel hvf -cpu cortex-a57 -M virt,highmem=  
[sumit@sumitqemu:~$ chmod 777 100000_cpu.sh  
[sumit@sumitqemu:~$ ls -lrt  
total 12  
-rwxrwxrwx 1 sumit sumit 300 Feb 2 06:53 1000_cpu.sh  
-rwxrwxrwx 1 sumit sumit 305 Feb 2 07:07 10000_cpu.sh  
-rwxrwxrwx 1 sumit sumit 310 Feb 2 07:09 100000_cpu.sh  
sumit@sumitqemu:~$
```

QEMU Results, Configuration: 2 GB 2 cores for max-prime = 1000 and time = 30 seconds

Iteration	Screenshot
1	<pre>sumit@sumitqemu:~/ \$./1000_cpu.sh sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1000 Initializing worker threads... Threads started! CPU speed: events per second: 207070.30 General statistics: total time: 30.0009s total number of events: 6221583 Latency (ms): min: 0.00 avg: 0.00 max: 4.78 95th percentile: 0.00 sum: 29206.13 Threads fairness: events (avg/stddev): 6221583.0000/0.00 execution time (avg/stddev): 29.2061/0.00</pre>
2	<pre>sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1000 Initializing worker threads... Threads started! CPU speed: events per second: 206991.10 General statistics: total time: 30.0011s total number of events: 6211693 Latency (ms): min: 0.00 avg: 0.00 max: 2.80 95th percentile: 0.00 sum: 29202.54 Threads fairness: events (avg/stddev): 6211693.0000/0.00 execution time (avg/stddev): 29.2025/0.00</pre>

3

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

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 205115.20

General statistics:
    total time:                      30.0012s
    total number of events:           6154335

Latency (ms):
    min:                            0.00
    avg:                            0.00
    max:                            9.31
    95th percentile:                0.00
    sum:                           29201.47

Threads fairness:
    events (avg/stddev):          6154335.0000/0.00
    execution time (avg/stddev):   29.2015/0.00
```

4

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

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 208193.94

General statistics:
    total time:                      30.0012s
    total number of events:           6246737

Latency (ms):
    min:                            0.00
    avg:                            0.00
    max:                            10.80
    95th percentile:                0.00
    sum:                           29206.46

Threads fairness:
    events (avg/stddev):          6246737.0000/0.00
    execution time (avg/stddev):   29.2065/0.00
```

5

```

sysbench 1.0.18 (using system LuajIT 2.1.0-beta3)

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 208309.07

General statistics:
    total time: 30.0006s
    total number of events: 6249904

Latency (ms):
    min: 0.00
    avg: 0.00
    max: 2.79
    95th percentile: 0.00
    sum: 29216.73

Threads fairness:
    events (avg/stddev): 6249904.0000/0.00
    execution time (avg/stddev): 29.2167/0.00

```

Detailed report:

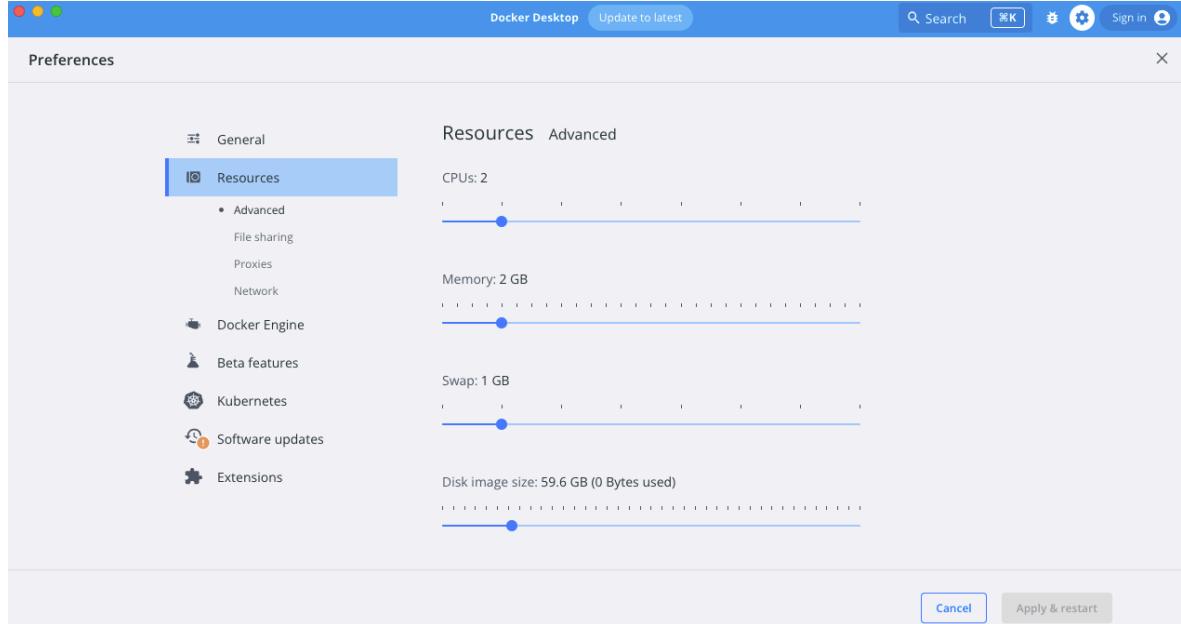
Iteration No	Events/second
1	207070.30
2	206991.10
3	205115.20
4	208193.94
5	208309.07

Observations:

Average events per second	207135.922
Minimum Number of events per second recorded	205115.20
Maximum Number of events per second recorded	208309.07

Docker Results, Configuration: 2 GB 2 cores for max-prime = 1000 and time = 30 seconds

Setting CPU count and memory in docker desktop:



Dockerfile, sysbench command, building docker image and running it:

```
(base) sumit@Sumits-MacBook-Air ~ % cat 1000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 1000_cpu.sh /
ENTRYPOINT ["../1000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % cat 1000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=1000 --time=30 run
(sysbench) sumit@Sumits-MacBook-Air ~ %
```

```
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 1000cpudocker1 . -f 1000_CPU_Dockerfile
[+] Building 0.2s (9/9) FINISHED
=> [internal] load build definition from 1000_CPU_Dockerfile
=> => transferring dockerfile: 176B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
=> [internal] load build context
=> => transferring context: 33B
=> CACHED [2/4] RUN apt-get update
=> CACHED [3/4] RUN apt-get install -y sysbench
=> CACHED [4/4] COPY 1000_cpu.sh /
=> exporting to image
=> => exporting layers
=> => writing image sha256:06d58584b4ba6dd79d5bd5e9b85d7336d97032bb59b54c3d88d6b6fc1b2afale
=> => naming to docker.io/library/1000cpudocker1

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="2g" --cpuset-cpus="0-1" -d 1000cpudocker1
e41453ffea3cd9df0fa6fb28c6cbced5d191736e2528c1be6e0a6cca658793b
(base) sumit@Sumits-MacBook-Air ~ % docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
<none>            <none>   276b5558b578  13 minutes ago  116MB
1000cpudocker1     latest   06d58584b4ba  13 minutes ago  116MB
1000cpudocker      latest   7ad29dcc6c7a  26 minutes ago  116MB
new_docker_image    latest   4ffcb1d09133  2 hours ago   65.7MB
arm64v8/ubuntu     20.04   c9eb527b091d  21 hours ago   65.7MB
hello-world         latest   46331d942d63  10 months ago  9.14kB
(base) sumit@Sumits-MacBook-Air ~ %
```

```
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f jolly_wu > 2_2cpuresult.log
(base) sumit@Sumits-MacBook-Air ~ % cat 2_2cpuresult.log
sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)
```

Docker inspect shows us the memory and cores set for running container:

```
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
e41453ffea3cd9df0fa6fb28c6cbced5d191736e2528c1be6e0a6cca658793b
(base) sumit@Sumits-MacBook-Air ~ % docker inspect e41453ffea3cd9df0fa6fb28c6cbced5d191736e2528c1be6e0a6cca658793b
[
  {
    "Id": "e41453ffea3cd9df0fa6fb28c6cbced5d191736e2528c1be6e0a6cca658793b",
    "Created": "2023-02-02T08:33:10.559151418Z",
    "Path": "./1000_cpu.sh",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 4052,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-02-02T08:33:10.97802771Z",
      "FinishedAt": "2023-02-02T08:33:10.97802771Z"
    },
    "Image": "sha256:06d58584b4ba6dd79d5bd5e9b85d7336d97032bb59b54c3d88d6b6fc1b2afale",
    "Config": {
      "ExposedPorts": {},
      "Env": [
        "TERM=xterm"
      ],
      "Labels": {}
    }
  }
]
```

```
          "CpuShares": 0,
          "Memory": 2147483648,
          "NanoCpus": 0,
          "CgroupParent": "",
          "BlkioWeight": 0,
          "BlkioWeightDevice": [],
          "BlkioDeviceReadBps": null,
          "BlkioDeviceWriteBps": null,
          "BlkioDeviceReadIOPS": null,
          "BlkioDeviceWriteIOPS": null,
          "CpuPeriod": 0,
          "CpuQuota": 0,
          "CpuRealtimePeriod": 0,
          "CpuRealtimeRuntime": 0,
          "CpusetCPUs": "0-1",
          "CpusetMems": ""}
```

Iteration	Screenshot
1	<pre>(base) sumit@Sumits-MacBook-Air ~ % docker logs -f jolly_wu > 2_2cpurest.log (base) sumit@Sumits-MacBook-Air ~ % cat 2_2cpurest.log sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1000 Initializing worker threads... Threads started! CPU speed: events per second: 205997.60 General statistics: total time: 30.0008s total number of events: 6180477 Latency (ms): min: 0.00 avg: 0.00 max: 0.90 95th percentile: 0.00 sum: 29208.14 Threads fairness: events (avg/stddev): 6180477.0000/0.00 execution time (avg/stddev): 29.2081/0.00</pre>
2	<pre>sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 1000 Initializing worker threads... Threads started! CPU speed: events per second: 175928.15 General statistics: total time: 30.0025s total number of events: 5278897 Latency (ms): min: 0.00 avg: 0.01 max: 44.58 95th percentile: 0.01 sum: 29069.15 Threads fairness: events (avg/stddev): 5278897.0000/0.00 execution time (avg/stddev): 29.0691/0.00</pre>

3

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

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 204527.44

General statistics:
  total time:          30.0004s
  total number of events: 6136114

Latency (ms):
  min:                0.00
  avg:                0.00
  max:                7.93
  95th percentile:    0.00
  sum:               29192.77

Threads fairness:
  events (avg/stddev):   6136114.0000/0.00
  execution time (avg/stddev): 29.1928/0.00
```

4

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

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 206178.88

General statistics:
  total time:          30.0008s
  total number of events: 6186018

Latency (ms):
  min:                0.00
  avg:                0.00
  max:                9.81
  95th percentile:    0.00
  sum:               29196.54

Threads fairness:
  events (avg/stddev):   6186018.0000/0.00
  execution time (avg/stddev): 29.1965/0.00
```

5

```

sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 203682.07

General statistics:
total time: 30.0007s
total number of events: 6110880

Latency (ms):
min: 0.00
avg: 0.00
max: 2.71
95th percentile: 0.00
sum: 29188.54

Threads fairness:
events (avg/stddev): 6110880.0000/0.00
execution time (avg/stddev): 29.1885/0.00

```

Detailed report:

Iteration No	Events per second
1	205997.60
2	175928.15
3	204527.44
4	206178.88
5	203682.07

Observations:

Average events per second	199262.828
Minimum Number of events per second recorded	175928.15
Maximum Number of events per second recorded	206178.88

Conclusion: We can conclude that QEMU is faster than Docker as per this experiment.

QEMU Results for 2 GB 2 cores for max-prime = 10000 and time = 30 seconds

	<pre>sumit@sumitqemu:~/10000_cpu.sh sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 10000 Initializing worker threads... Threads started! CPU speed: events per second: 6690.72 General statistics: total time: 30.0004s total number of events: 200761 Latency (ms): min: 0.15 avg: 0.15 max: 1.93 95th percentile: 0.16 sum: 29936.58 Threads fairness: events (avg/stddev): 200761.0000/0.00 execution time (avg/stddev): 29.9366/0.00</pre>
2	<pre>sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 10000 Initializing worker threads... Threads started! CPU speed: events per second: 6676.86 General statistics: total time: 30.0012s total number of events: 200333 Latency (ms): min: 0.15 avg: 0.15 max: 3.80 95th percentile: 0.16 sum: 29951.39 Threads fairness: events (avg/stddev): 200333.0000/0.00 execution time (avg/stddev): 29.9514/0.00</pre>

3

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6715.79

General statistics:
  total time: 30.0004s
  total number of events: 201488

Latency (ms):
  min: 0.15
  avg: 0.15
  max: 1.50
  95th percentile: 0.16
  sum: 29928.71

Threads fairness:
  events (avg/stddev): 201488.0000/0.00
  execution time (avg/stddev): 29.9287/0.00
```

4

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6667.16

General statistics:
  total time: 30.0013s
  total number of events: 200138

Latency (ms):
  min: 0.15
  avg: 0.15
  max: 4.49
  95th percentile: 0.16
  sum: 29894.93

Threads fairness:
  events (avg/stddev): 200138.0000/0.00
  execution time (avg/stddev): 29.8949/0.00
```

5

```

sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 6686.84

General statistics:
total time: 30.0010s
total number of events: 200633

Latency (ms):
min: 0.15
avg: 0.15
max: 1.03
95th percentile: 0.16
sum: 29889.49

Threads fairness:
events (avg/stddev): 200633.0000/0.00
execution time (avg/stddev): 29.8895/0.00

```

Detailed report:

Iteration No	Events per second
1	6690.72
2	6676.86
3	6715.79
4	6667.16
5	6686.84

Observations:

Average events per second	6687.474
Minimum Number of events per second recorded	6667.16
Maximum Number of events per second recorded	6715.79

Docker results for 2 GB 2 cores for max-prime = 10000 and time = 30 seconds

Dockerfile, sysbench command, building docker image and running it:

```
(base) sumit@Sumits-MacBook-Air ~ % cat 10000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=10000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ % cat 10000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 10000_cpu.sh /
ENTRYPOINT ["./10000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ %

(base) sumit@Sumits-MacBook-Air ~ % docker build -t 10000cpudocker . -f 10000_CPU_Dockerfile
[+] Building 0.1s (9/9) FINISHED
=> [internal] load build definition from 10000_CPU_Dockerfile
=> => transferring dockerfile: 179B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
=> [internal] load build context
=> => transferring context: 358B
=> CACHED [2/4] RUN apt-get update
=> CACHED [3/4] RUN apt-get install -y sysbench
=> [4/4] COPY 10000_cpu.sh /
=> exporting to image
=> => exporting layers
=> => writing image sha256:9a9867686617e7f4a6ad822a8511e6d7565b3ccbfcaf0139dff26c78120c26f1
=> => naming to docker.io/library/10000cpudocker

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="2g" --cpuset-cpus="0-1" -d 10000cpudocker
4356ef4715d9607b8bbe647d75b479957a0a49255def39d18fadcb2474f3b609
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4356ef4715d9 10000cpudocker "./10000_cpu.sh" 8 seconds ago Up 7 seconds crazy_bartik
(base) sumit@Sumits-MacBook-Air ~ %

(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4356ef4715d9 10000cpudocker "./10000_cpu.sh" 8 seconds ago Up 7 seconds crazy_bartik
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f crazy_bartik > 2_2_10000_cpuresult.log
(base) sumit@Sumits-MacBook-Air ~ % cat 2_2_10000_cpuresult.log
sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)
```

Iteration	Screenshot
-----------	------------

1

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6624.25

General statistics:
  total time: 30.0008s
  total number of events: 198743

Latency (ms):
  min: 0.15
  avg: 0.15
  max: 4.92
  95th percentile: 0.16
  sum: 29939.66

Threads fairness:
  events (avg/stddev): 198743.0000/0.00
  execution time (avg/stddev): 29.9397/0.00
```

2

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6548.58

General statistics:
  total time: 30.0009s
  total number of events: 196479

Latency (ms):
  min: 0.15
  avg: 0.15
  max: 12.62
  95th percentile: 0.16
  sum: 29931.06

Threads fairness:
  events (avg/stddev): 196479.0000/0.00
  execution time (avg/stddev): 29.9311/0.00
```

3

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6651.05

General statistics:
  total time: 30.0008s
  total number of events: 199594

Latency (ms):
  min: 0.15
  avg: 0.15
  max: 2.90
  95th percentile: 0.16
  sum: 29932.66

Threads fairness:
  events (avg/stddev): 199594.0000/0.00
  execution time (avg/stddev): 29.9327/0.00
```

4

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6684.13

General statistics:
  total time: 30.0003s
  total number of events: 200542

Latency (ms):
  min: 0.15
  avg: 0.15
  max: 1.11
  95th percentile: 0.16
  sum: 29956.44

Threads fairness:
  events (avg/stddev): 200542.0000/0.00
  execution time (avg/stddev): 29.9564/0.00
```

5

```

sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 6638.55

General statistics:
total time: 30.0008s
total number of events: 199169

Latency (ms):
min: 0.15
avg: 0.15
max: 13.29
95th percentile: 0.16
sum: 29940.62

Threads fairness:
events (avg/stddev): 199169.0000/0.00
execution time (avg/stddev): 29.9406/0.00

```

Detailed report:

Iteration No	Events per second
1	6624.25
2	6548.58
3	6651.05
4	6684.13
5	6638.55

Observations:

Average events per second	6629.312
Minimum Number of events per second recorded	6548.58
Maximum Number of events per second recorded	6684.13

Conclusion: As we can see, increasing the `cpu-max-prime` argument value reduces the no. of events/second for our current experimental configuration. On this MacBook chip, QEMU is faster than Docker desktop.

QEMU Results for 2 GB 2 cores for max-prime = 100000 and time = 30 seconds

Iteration	Screenshot
1	<pre> sumit@sumitqemu:~/100000_cpu.sh sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 100000 Initializing worker threads... Threads started! CPU speed: events per second: 291.96 General statistics: total time: 30.0021s total number of events: 8762 Latency (ms): min: 3.35 avg: 3.42 max: 6.75 95th percentile: 3.49 sum: 29949.29 Threads fairness: events (avg/stddev): 8762.0000/0.00 execution time (avg/stddev): 29.9493/0.00 </pre>
2	<pre> sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 100000 Initializing worker threads... Threads started! CPU speed: events per second: 286.00 General statistics: total time: 30.0019s total number of events: 8581 Latency (ms): min: 3.34 avg: 3.49 max: 16.51 95th percentile: 3.55 sum: 29934.45 Threads fairness: events (avg/stddev): 8581.0000/0.00 execution time (avg/stddev): 29.9345/0.00 </pre>

3

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 292.42

General statistics:
  total time: 30.0032s
  total number of events: 8775

Latency (ms):
  min: 3.35
  avg: 3.42
  max: 8.03
  95th percentile: 3.43
  sum: 29970.93

Threads fairness:
  events (avg/stddev): 8775.0000/0.00
  execution time (avg/stddev): 29.9709/0.00
```

4

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 290.90

General statistics:
  total time: 30.0048s
  total number of events: 8729

Latency (ms):
  min: 3.34
  avg: 3.43
  max: 14.01
  95th percentile: 3.49
  sum: 29969.36

Threads fairness:
  events (avg/stddev): 8729.0000/0.00
  execution time (avg/stddev): 29.9694/0.00
```

5

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 291.67

General statistics:
    total time: 30.0027s
    total number of events: 8754

Latency (ms):
    min: 3.33
    avg: 3.42
    max: 11.50
    95th percentile: 3.49
    sum: 29948.29

Threads fairness:
    events (avg/stddev): 8754.0000/0.00
    execution time (avg/stddev): 29.9483/0.00
```

Detailed report:

Iteration No	Events per second
1	291.96
2	286.00
3	292.42
4	290.90
5	291.67

Observations:

Average events per second	290.59
Minimum Number of events per second recorded	286.00
Maximum Number of events per second recorded	292.42

Docker results for 2 GB 2 cores for max-prime = 100000 and time = 30 seconds

Dockerfile, sysbench command, building docker image and running it:

```
(base) sumit@Sumits-MacBook-Air ~ % cat 100000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=100000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ % cat 100000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 100000_cpu.sh /
ENTRYPOINT ["/100000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 100000cpudocker . -f 100000_CPU_Dockerfile
[+] Building 0.3s (9/9) FINISHED
--> [internal] load build definition from 100000_CPU_Dockerfile
--> => transferring dockerfile: 184B
--> [internal] load .dockerrigignore
--> => transferring context: 2B
--> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
--> [internal] load build context
--> => transferring context: 364B
--> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
--> CACHED [2/4] RUN apt-get update
--> CACHED [3/4] RUN apt-get install -y sysbench
--> [4/4] COPY 100000_cpu.sh /
--> exporting to image
--> => exporting layers
--> => writing image sha256:3e9a4bf1e4e1192c9aaa2e466d45ca8b783385d45296a2dd9e3f71b656a0ba32
--> => naming to docker.io/library/100000cpudocker

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="2g" --cpuset-cpus="0-1" -d 100000cpudocker
6c1cid294e5d72a21d7110e5508e89d99b7cc84cae1ff7c47bf14dc97a312c0
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
6c1cid294e5d 100000cpudocker "/100000_cpu.sh" 5 seconds ago Up 4 seconds frosty_jang
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f frosty_jang > 2_2_100000_cpurest.log
```

Iteration	Screenshot
1	<pre>(base) sumit@Sumits-MacBook-Air ~ % cat 2_2_100000_cpurest.log sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3) Running the test with following options: Number of threads: 1 Initializing random number generator from current time Prime numbers limit: 100000 Initializing worker threads... Threads started! CPU speed: events per second: 290.32 General statistics: total time: 30.0026s total number of events: 8711 Latency (ms): min: 3.37 avg: 3.44 max: 14.49 95th percentile: 3.49 sum: 29965.81 Threads fairness: events (avg/stddev): 8711.0000/0.00 execution time (avg/stddev): 29.9658/0.00</pre>

2

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 290.33

General statistics:
  total time: 30.0013s
  total number of events: 8711

Latency (ms):
  min: 3.37
  avg: 3.44
  max: 5.10
  95th percentile: 3.49
  sum: 29969.05

Threads fairness:
  events (avg/stddev): 8711.0000/0.00
  execution time (avg/stddev): 29.9691/0.00
```

3

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 288.77

General statistics:
  total time: 30.0040s
  total number of events: 8665

Latency (ms):
  min: 3.37
  avg: 3.46
  max: 15.23
  95th percentile: 3.49
  sum: 29947.63

Threads fairness:
  events (avg/stddev): 8665.0000/0.00
  execution time (avg/stddev): 29.9476/0.00
```

4

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 289.91

General statistics:
  total time: 30.0007s
  total number of events: 8698

Latency (ms):
  min: 3.37
  avg: 3.44
  max: 18.62
  95th percentile: 3.49
  sum: 29951.10

Threads fairness:
  events (avg/stddev): 8698.0000/0.00
  execution time (avg/stddev): 29.9511/0.00
```

5

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 290.64

General statistics:
  total time: 30.0061s
  total number of events: 8722

Latency (ms):
  min: 3.38
  avg: 3.44
  max: 7.97
  95th percentile: 3.49
  sum: 29970.61

Threads fairness:
  events (avg/stddev): 8722.0000/0.00
  execution time (avg/stddev): 29.9706/0.00
```

Detailed report:

Iteration No	Events per second
1	290.32
2	290.33
3	288.77
4	289.91
5	290.64

Observations:

Average events per second	289.994
Minimum Number of events per second recorded	288.77
Maximum Number of events per second recorded	290.64

Conclusion: As we can see, increasing the cpu-max-prime argument value reduces the no. of events/second for our current experimental configuration. QEMU is slightly faster than the Docker but the overall result is very close.

1.2 FILE I/O Testing

QEMU Results for 2 GB 2 cores for Combined random read/write (rndrw)

```
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:        13965.48
  fsyncs/s:       17977.94

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   218.21

General statistics:
  total time:           10.0165s
  total number of events: 318971

Latency (ms):
  min:                  0.00
  avg:                  0.25
  max:                 35.76
  95th percentile:     0.77
  sum:                79654.91

Threads fairness:
  events (avg/stddev): 39871.3750/429.67
  execution time (avg/stddev): 9.9569/0.00

sumit@sumitqemu:~$ sysbench --num-threads=8 --test=fileio --file-total-size=2G --file-test-mode=rndwr cleanup
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

Removing test files...
```

Iteration	Results
1	reads/s: 0.00 writes/s: 13965.48 fsyncs/s: 17977.94
2	reads/s: 0.00 writes/s: 15485.85 fsyncs/s: 19915.19
3	reads/s: 0.00 writes/s: 13610.93 fsyncs/s: 17516.29
4	reads/s: 0.00 writes/s: 12708.62 fsyncs/s: 16357.61
5	reads/s: 0.00 writes/s: 14108.83 fsyncs/s: 18155.79

Docker Results for 2 GB 2 cores for Combined random read/write (rndrw)

```
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:        13146.47
  fsyncs/s:        16925.64

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   205.41

General statistics:
  total time:           10.0243s
  total number of events: 300464

Latency (ms):
  min:                  0.00
  avg:                  0.27
  max:                 15.76
  95th percentile:       0.78
  sum:                79767.97

Threads fairness:
  events (avg/stddev): 37558.0000/708.09
  execution time (avg/stddev): 9.9710/0.00

root@3906cdb8946c:/#
```

Iteration	Results
1	reads/s: 0.00 writes/s: 13146.47 fsyncs/s: 16925.64
2	reads/s: 0.00 writes/s: 13246.90 fsyncs/s: 17053.29
3	reads/s: 0.00 writes/s: 11504.98 fsyncs/s: 14820.90
4	reads/s: 0.00 writes/s: 12673.63 fsyncs/s: 16312.28
5	reads/s: 0.00 writes/s: 13016.04 fsyncs/s: 16755.58

Conclusion: As we see, QEMU is faster than the docker desktop

2. Configuration 2: 3 GB RAM with 3 Cores

```
(base) sumit@Sumits-MacBook-Air:~ % /opt/homebrew/bin/qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem-off -m 3072 -smp 3 \
-drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="dummyserial" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic
```

2.1 CPU Testing

QEMU Results for 3 GB 3 cores for max-prime = 1000 and time = 30 seconds

Screenshot

```
sumit@sumitqemu:~/1000_cpu.sh
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 340901.54

General statistics:
    total time:          30.0009s
    total number of events: 10228006

Latency (ms):
    min:                  0.00
    avg:                  0.00
    max:                  4.78
    95th percentile:     0.00
    sum:                 29207.12

Threads fairness:
    events (avg/stddev): 10228006.0000/0.00
    execution time (avg/stddev): 29.2071/0.00
```

Detailed report:

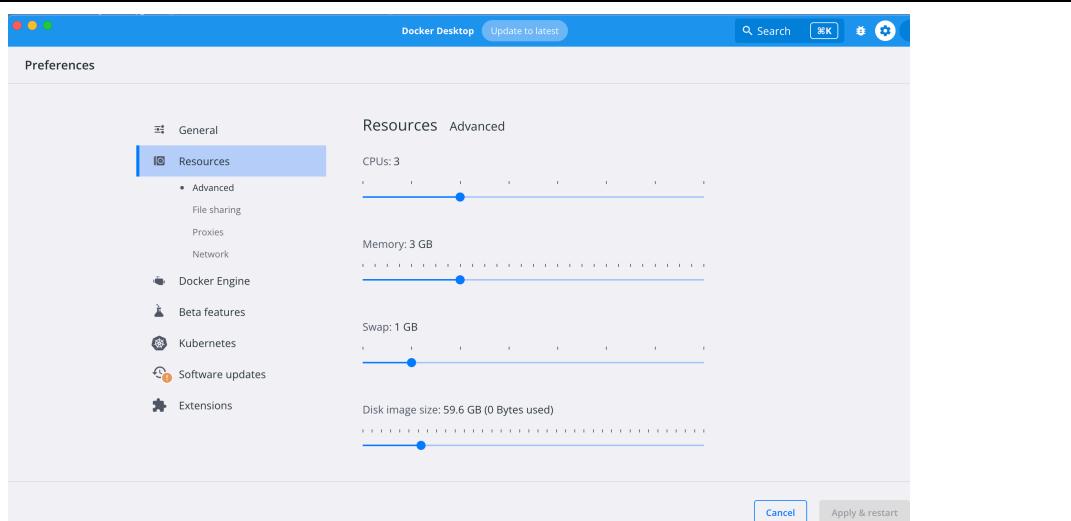
Iteration No	Events per second
1	340901.54
2	336953.57
3	333870.77

4	334280.58
5	333166.36

Observations:

Average events per second	334203.858
Minimum Number of events per second recorded	333166.36
Maximum Number of events per second recorded	340901.54

Docker results for 3 GB 3 cores for max-prime = 1000 and time = 30 seconds



Docker Desktop Preferences window showing resource allocation:

- CPU: 3
- Memory: 3 GB
- Swap: 1 GB

Disk image size: 59.6 GB (0 Bytes used)

Sumit's terminal output showing Docker build and run commands:

```

(base) sumit@Sumits-MacBook-Air ~ % cat 1000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 1000_cpu.sh /
ENTRYPOINT ["/1000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % cat 1000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=1000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 1000cpudocker_3gb . -f 1000_CPU_Dockerfile
[+] Building 0.2s (9/9) FINISHED
=> [internal] load build definition from 1000_CPU_Dockerfile          0.1s
=> => transferring dockerfile: 176B                                0.1s
=> [internal] load .dockignore                                     0.1s
=> => transferring context: 2B                                    0.1s
=> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04      0.0s
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04                      0.0s
=> [internal] load build context                                    0.0s
=> => transferring context: 352B                                 0.0s
=> CACHED [2/4] RUN apt-get update                                0.0s
=> CACHED [3/4] RUN apt-get install -y sysbench                  0.0s
=> CACHED [4/4] COPY 1000_cpu.sh /                               0.0s
=> exporting to image                                         0.0s
=> => exporting layers                                       0.0s
=> => writing image sha256:06d58584b4ba6dd79d5bd5e9b85d7336d97032bb59b54c3d88d6b6fc1b2afa1e 0.0s
=> => naming to docker.io/library/1000cpudocker_3gb           0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-2" -d 1000cpudocker_3gb
lecb2bd7a0eb7a1e949c7fd3b67ea463803031991b0775df669752197efcbc1c9

```

```
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1ecb2bd7a0eb 1000cpudocker_3gb "./1000_cpu.sh" About a minute ago Up About a minute
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f sweet_shockley > 3_3_cpu1000.log
(base) sumit@Sumits-MacBook-Air ~ % cat 3_3_cpu1000.log
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)
```

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

```
(base) sumit@Sumits-MacBook-Air ~ % docker inspect 1ecb2bd7a0eb
[
  {
    "Id": "1ecb2bd7a0eb7a1e949c7fd3b67ea463803031991b0775df669752197efcb1c9",
    "Created": "2023-02-02T20:50:05.070517011Z",
    "Path": "./1000_cpu.sh",
    "Args": [],
    "State": {
      "Status": "exited",
      "Running": false,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 0,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-02-02T20:50:05.550347803Z",
      "FinishedAt": "2023-02-02T20:52:35.686180969Z"
    },
    "Image": "sha256:06d58584b4ba6dd79d5bd5e9b85d7336d97032bb59b54c3d88d6b6fc1b2afa1e",
    "Config": {
      "Labels": {},
      "Env": [
        "TERM=xterm-256color"
      ],
      "WorkingDir": "/root",
      "Entrypoint": [
        "sh"
      ],
      "Cmd": [
        "-c",
        "cd /root; ./1000_cpu.sh"
      ],
      "Labels": {}
    }
  }
]
```

```
  ],
  "Isolation": "",
  "CpuShares": 0,
  "Memory": 3221225472,
  "NanoCpus": 0,
  "CgroupParent": "",
  "BlkioWeight": 0,
  "BlkioWeightDevice": [],
  "BlkioDeviceReadBps": null,
  "BlkioDeviceWriteBps": null,
  "BlkioDeviceReadIops": null,
  "BlkioDeviceWriteIops": null,
  "CpuPeriod": 0,
  "CpuQuota": 0,
  "CpuRealtimePeriod": 0,
  "CpuRealtimeRuntime": 0,
  "CpusetCpus": "0-2",
  "CpusetMems": "",
  "Devices": [],
  "DeviceCgroupRules": null,
  "DeviceRequests": null,
```

Screenshot

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

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 331730.09

General statistics:
  total time:          30.0008s
  total number of events: 9952546

Latency (ms):
  min:                0.00
  avg:                0.00
  max:               11.36
  95th percentile:    0.00
  sum:              29201.06

Threads fairness:
  events (avg/stddev): 9952546.0000/0.00
  execution time (avg/stddev): 29.2011/0.00
```

Detailed report:

Iteration No	Events per second
1	331730.09
2	335519.61
3	332949.22
4	331667.16
5	333345.60

Observations:

Average events per second	333042.336
Minimum Number of events per second recorded	331667.16
Maximum Number of events per second recorded	335519.61

QEMU Results for 3 GB 3 cores for max-prime = 10000 and time = 30 seconds

Screenshot

```
sumit@sumitqemu:~$ ./10000_cpu.sh
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 10994.20

General statistics:
  total time:          30.0019s
  total number of events: 329913

Latency (ms):
  min:                 0.09
  avg:                 0.09
  max:                 0.90
  95th percentile:    0.10
  sum:                29955.90

Threads fairness:
  events (avg/stddev): 329913.0000/0.00
  execution time (avg/stddev): 29.9559/0.00
```

Detailed report:

Iteration No	Events per second
1	10994.20
2	10931.65
3	10983.04
4	10895.02
5	10911.60

Observations:

Average events per second	10943.102
Minimum Number of events per second recorded	10895.02
Maximum Number of events per second recorded	10994.20

Docker Results for 3 GB 3 cores for max-prime = 10000 and time = 30 seconds

```
(base) sumit@Sumits-MacBook-Air ~ % cat 10000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 10000_cpu.sh /
ENTRYPOINT [ "./10000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % cat 10000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=10000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 10000cpudocker_3gb . -f 10000_CPU_Dockerfile
[+] Building 0.2s (9/9) FINISHED
--> [internal] load build definition from 10000_CPU_Dockerfile
--> => transferring dockerfile: 179B
--> [internal] load .dockerrigore
--> => transferring context: 2B
--> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
--> [internal] load build context
--> => transferring context: 358B
--> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
--> CACHED [2/4] RUN apt-get update
--> CACHED [3/4] RUN apt-get install -y sysbench
--> CACHED [4/4] COPY 10000_cpu.sh /
--> exporting to image
--> => exporting layers
--> => writing image sha256:9a98676861e7f4a6ad822a8511e6d7565b3ccbfcaf0139dff26c78120c26f1
--> => naming to docker.io/library/10000cpudocker_3gb
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-2" -d 10000cpudocker_3gb
aeff7d9a7cb7e145f8cc8daa27571f781344fa6346cc976cfa99364a8b1ca669
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
aeff7d9a7cb7e 10000cpudocker_3gb "/10000_cpu.sh" 11 seconds ago Up 9 seconds adoring_payne
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f adoring_payne > 3_3_cpu10000.log
(base) sumit@Sumits-MacBook-Air ~ % cat 3_3_cpu10000.log
sysbench 1.0.18 (using system LuajIT 2.1.0-beta3)

Running the test with following options:
```

Screenshot

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

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 10945.48

General statistics:
total time: 30.0005s
total number of events: 328381

Latency (ms):
min: 0.09
avg: 0.09
max: 6.07
95th percentile: 0.10
sum: 29964.14

Threads fairness:
events (avg/stddev): 328381.0000/0.00
execution time (avg/stddev): 29.9641/0.00
```

Detailed report:

Iteration No	Events per second
1	10945.48
2	10942.04
3	10981.41
4	10868.70
5	10959.65

Observations:

Average events per second	10939.456
Minimum Number of events per second recorded	10868.70
Maximum Number of events per second recorded	10981.41

QEMU Results for 3 GB 3 cores for max-prime = 100000 and time = 30 seconds

Screenshot

```
sumit@sumitqemu:~$ ./100000_cpu.sh
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second:  477.50

General statistics:
  total time:          30.0022s
  total number of events: 14327

Latency (ms):
  min:                 2.04
  avg:                 2.09
  max:                 4.14
  95th percentile:    2.18
  sum:                29960.40

Threads fairness:
  events (avg/stddev): 14327.0000/0.00
  execution time (avg/stddev): 29.9604/0.00
```

Detailed report:

Iteration No	Events per second
1	477.50
2	475.11
3	475.29
4	472.13
5	475.37

Observations:

Average events per second	475.08
Minimum Number of events per second recorded	472.13
Maximum Number of events per second recorded	477.50

Docker Results for 3 GB 3 cores for max-prime = 100000 and time = 30 seconds

```
(base) sumit@Sumits-MacBook-Air ~ % cat 100000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 100000_cpu.sh /
ENTRYPOINT ["./100000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % cat 100000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=100000 --time=30 run
(sysbench cpu --threads=1 --cpu-max-prime=100000 --time=30 run)
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 100000cpudocker_3gb . -f 100000_CPU_Dockerfile
[+] Building 0.1s (9/9) FINISHED
=> [internal] load build definition from 100000_CPU_Dockerfile
=> => transferring dockerfile: 184B
=> [internal] load .dockerrignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
=> [internal] load build context
=> => transferring context: 364B
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
=> CACHED [2/4] RUN apt-get update
=> CACHED [3/4] RUN apt-get install -y sysbench
=> CACHED [4/4] COPY 100000_cpu.sh /
=> exporting to image
=> => exporting layers
=> => writing image sha256:3e9a4bf1e4e1192c9aaa2e466d45ca8b783385d45296a2dd9e3f71b656a0ba32
=> => naming to docker.io/library/100000cpudocker_3gb

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-2" -d 100000cpudocker_3gb
9898aaaa26d8a9875395192bef104905d7/b8aa09ab18bbb962fe8777789257
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9898aaaa26d8 100000cpudocker_3gb "./100000_cpu.sh" 3 seconds ago Up 3 seconds
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f cranky>khayyam > 3_3_cpu100000.log
Error: No such container: cranky
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f cranky_khayyam > 3_3_cpu100000.log
```

Screenshot

```
(base) sumit@Sumits-MacBook-Air ~ % cat 3_3_cpu100000.log
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 473.65

General statistics:
    total time: 30.0011s
    total number of events: 14211

Latency (ms):
    min: 2.05
    avg: 2.11
    max: 3.90
    95th percentile: 2.18
    sum: 29968.05

Threads fairness:
    events (avg/stddev): 14211.0000/0.00
    execution time (avg/stddev): 29.9680/0.00
```

Detailed report:

Iteration No	Events per second
1	473.65
2	473.98
3	476.43
4	468.36
5	475.49

observations:

Average events per second	473.582
Minimum Number of events per second recorded	468.36
Maximum Number of events per second recorded	476.43

Conclusion: In terms of CPU performance, QEMU continues to outperform Docker Desktop. We notice that increasing the system configuration has no major effect on performance.

2.2 FILE I/O Testing

QEMU Results for 3 GB 3 cores for Combined random read/write (rndrw)

```
SGID 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:         14329.82
  fsyncs/s:         18409.85

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   223.90

General statistics:
  total time:       30.0344s
  total number of events: 981297

Latency (ms):
  min:              0.00
  avg:              0.49
  max:              69.30
  95th percentile:  1.52
  sum:              479067.75

Threads fairness:
  events (avg/stddev):    61331.0625/477.57
  execution time (avg/stddev): 29.9417/0.00
```

Iteration	Results
1	reads/s: 0.00 writes/s: 14329.82 fsyncs/s: 18409.85
2	reads/s: 0.00 writes/s: 14139.47 fsyncs/s: 18165.80

3	reads/s: 0.00 writes/s: 14532.74 fsyncs/s: 18668.68
4	reads/s: 0.00 writes/s: 13995.19 fsyncs/s: 17850.40
5	reads/s: 0.00 writes/s: 14317.42 fsyncs/s: 18228.44

Docker Results for 3 GB 3 cores for Combined random read/write (rndrw)

```
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:        13675.22
  fsyncs/s:        17570.21

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   213.68

General statistics:
  total time:       30.0458s
  total number of events: 936785

Latency (ms):
  min:                0.00
  avg:                0.51
  max:              100.68
  95th percentile:    1.58
  sum:            479062.16

Threads fairness:
  events (avg/stddev): 58549.0625/617.58
  execution time (avg/stddev): 29.9414/0.00
```

Iteration	Results
1	reads/s: 0.00 writes/s: 13675.22 fsyncs/s: 17570.21
2	reads/s: 0.00 writes/s: 14518.05 fsyncs/s: 18650.54

3	reads/s: 0.00 writes/s: 13791.48 fsyncs/s: 17717.81
4	reads/s: 0.00 writes/s: 12034.29 fsyncs/s: 15468.43
5	reads/s: 0.00 writes/s: 10943.55 fsyncs/s: 14073.20

Conclusion: When compared to previous experiment configurations, we can observe that result are very similar.

3. Configuration 3 : 3 GB RAM with 6 Cores

3.1 CPU Testing

QEMU results for 3 GB 6 cores for max-prime = 1000 and time = 30 seconds

Screenshot

```
(base) sumit@Sumits-MacBook-Air ~ % cd Desktop/Winter2023/Cloud\ computing/qemu
(base) sumit@Sumits-MacBook-Air qemu % /opt/homebrew/bin/qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 3072 -smp 6 \
-drive file=/opt/homebrew/Cellar/qemu/7.2.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
-drive if=none,file=ubuntu.img,format=qcow2,id=hd0 \
-device virtio-blk-device,drive=hd0,serial="dummyserial" \
-device virtio-net-device,netdev=net0 \
-netdev user,id=net0 \
-vga none -device ramfb \
-device usb-ehci -device usb-kbd -device usb-mouse -usb -nographic

[sumit@sumitqemu:~$ ./1000_cpu.sh
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 206756.81

General statistics:
total time: 30.0017s
total number of events: 6204108

Latency (ms):
min: 0.00
avg: 0.00
max: 13.93
95th percentile: 0.00
sum: 29161.22

Threads fairness:
events (avg/stddev): 6204108.0000/0.00
execution time (avg/stddev): 29.1612/0.00
```

Detailed report:

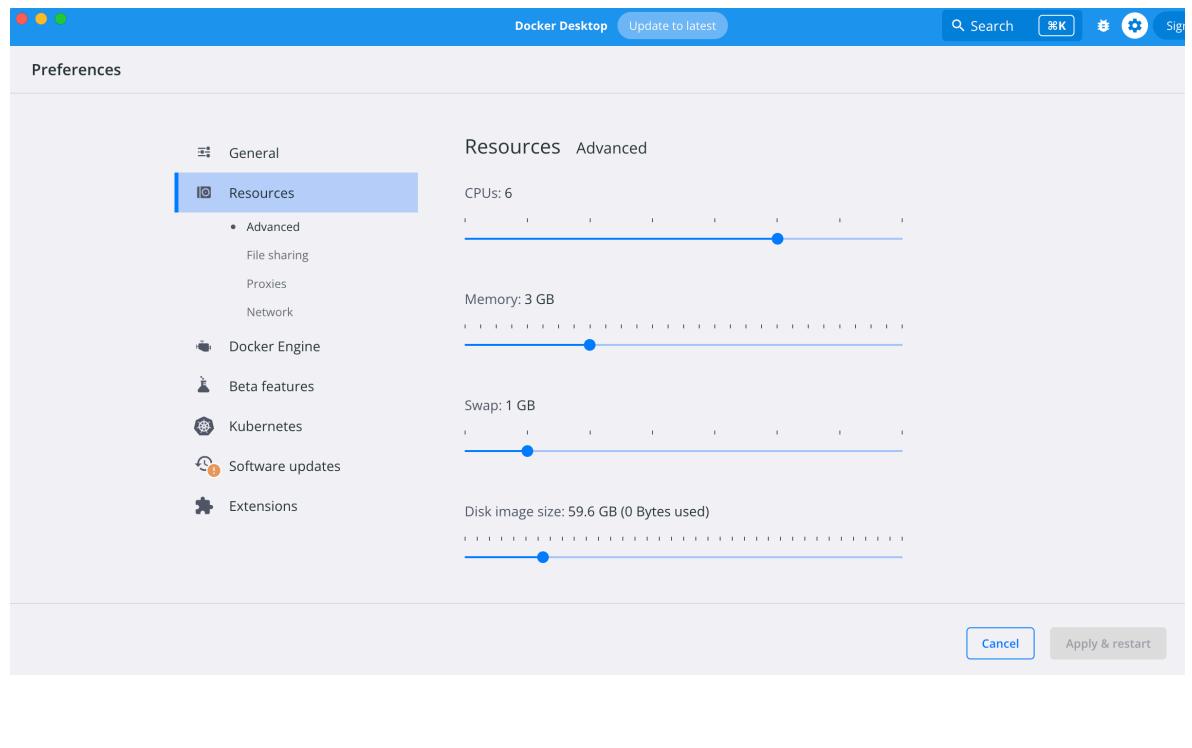
Iteration No	Events per second
1	206756.81
2	202822.32
3	204940.54
4	206212.28
5	206690.08

Observations:

Average events per second	205484.406
Minimum Number of events per second recorded	202822.32
Maximum Number of events per second recorded	206756.81

Docker Results for 3 GB 6 cores for max-prime = 1000 and time = 30 seconds

Screenshot



```
(base) sumit@Sumits-MacBook-Air ~ % cat 1000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 1000_cpu.sh /
ENTRYPOINT ["/1000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % cat 1000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=1000 --time=30 run
(sysbench cpu --threads=1 --cpu-max-prime=1000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ %
```

```
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 1000cpudocker_6 -f 1000_CPU_Dockerfile
[+] Building 0.2s (9/9) FINISHED
=> [internal] load build definition from 1000_CPU_Dockerfile
=> [internal] transfering dockerfile: 176B
=> [internal] load .dockerignore
=> [internal] transfering context: 2B
=> [internal] load index.json for docker.io/arm64v8/ubuntu:20.04
=> [internal] load build context
=> [internal] transfering context: 352B
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
=> CACHED [2/4] RUN apt-get update
=> CACHED [3/4] RUN apt-get install -y sysbench
=> CACHED [4/4] COPY 1000_cpu.sh /
=> exporting to image
=> exporting layers
=> writing image sha256:06d58584b2ba6dd79d5bd5e9b85d7336d97032bb59b54c3d88d6b6fc1b2afale
=> naming to docker.io/library/1000cpudocker_6

use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-5" -d 1000_cpudocker_6
Unable to find image '1000_cpudocker_6:latest' locally
docker: Error response from daemon: pull access denied for 1000_cpudocker_6, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-5" -d 1000cpudocker_6
247f97799d0910395b4ee520afee10d9a5ed1078a5e841f580fb5b76649e
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
247f97799d09        "1000_cpu.sh"      "4 seconds ago"    Up 3 seconds          blissful_blackburn
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f blissful_blackburn > 3_6_cpus1000.log
```

```
((base) sumit@Sumits-MacBook-Air ~ % cat 3_6_cpus1000.log
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 1000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 204987.20

General statistics:
total time: 30.00006s
total number of events: 6150021

Latency (ms):
min: 0.00
avg: 0.00
max: 0.99
95th percentile: 0.00
sum: 29184.18

Threads fairness:
events (avg/stddev): 6150021.0000/0.00
execution time (avg/stddev): 29.1842/0.00
```

Detailed report:

Iteration No	Events per second
1	204987.20
2	205311.25
3	203469.36
4	203823.34
5	204282.94

observations:

Average events per second	204374.818
Minimum Number of events per second recorded	203469.36
Maximum Number of events per second recorded	204987.20

QEMU Results for 3 GB 6 cores for max-prime = 10000 and time = 30 seconds

Screenshot

```
[sumit@sumitqemu:~$ ./10000_cpu.sh
sysbench 1.0.18 (using system LuAJIT 2.1.0-beta3)

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 6633.91

General statistics:
  total time:          30.0005s
  total number of events: 199027

Latency (ms):
  min:                0.15
  avg:                0.15
  max:                7.10
  95th percentile:    0.17
  sum:               29897.77

Threads fairness:
  events (avg/stddev): 199027.0000/0.00
  execution time (avg/stddev): 29.8978/0.00
```

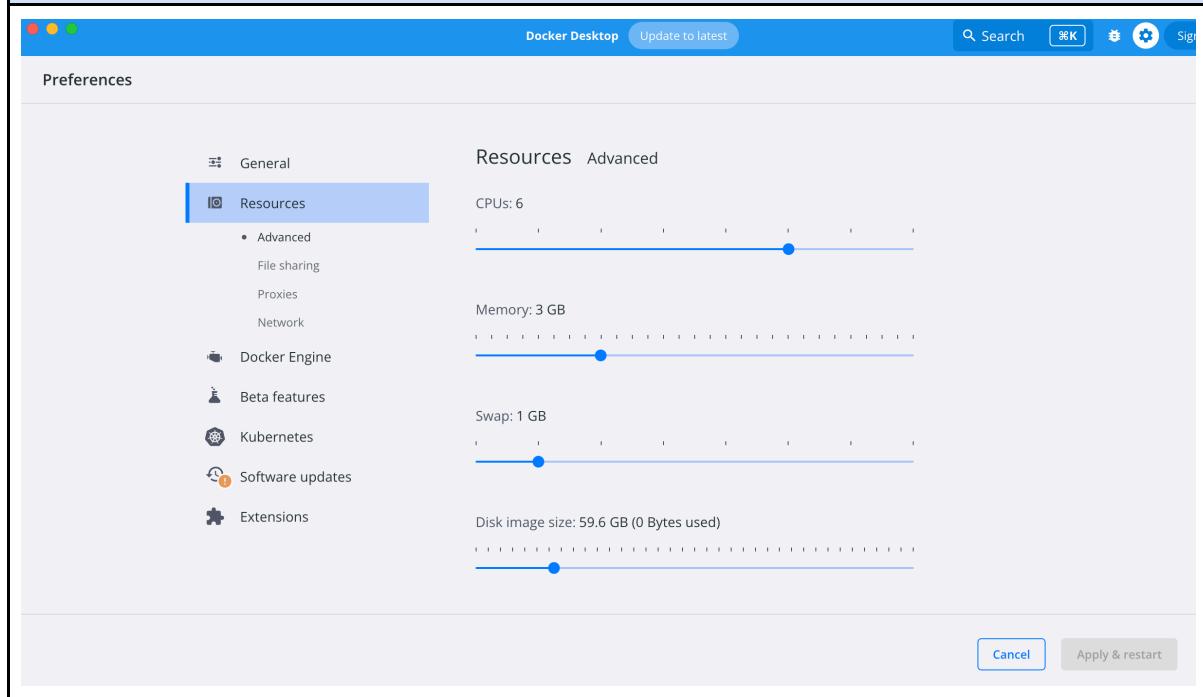
Detailed report:

Iteration No	Events per second
1	6633.91
2	6666.30
3	6678.28
4	6702.64
5	6662.45

Observations:

Average events per second	6668.716
Minimum Number of events per second recorded	6633.91
Maximum Number of events per second recorded	6702.64

Docker results for 3 GB 6 cores for max-prime = 10000 and time = 30 seconds



```

(base) sumit@Sumits-MacBook-Air ~ % cat 10000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=10000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 10000cpudocker_6 . -f 10000_CPU_Dockerfile
[+] Building 0.2s (9/9) FINISHED
=> [internal] load build definition from 10000_CPU_Dockerfile
=> => transferring dockerfile: 179B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
=> [internal] load build context
=> => transferring context: 358B
=> CACHED [2/4] RUN apt-get update
=> CACHED [3/4] RUN apt-get install -y sysbench
=> CACHED [4/4] COPY 10000_cpu.sh /
=> exporting to image
=> => exporting layers
=> => writing image sha256:9a9867686617e7f4a6ad822a8511e6d7565b3ccbfcaf0139dff26c78120c26f1
=> => naming to docker.io/library/10000cpudocker_6

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-5" -d 10000cpudocker_6
99f5ed63cc5f96ec05b1d7dfe9d0bad9a8b7f63b7a62a54b980f9520bc0e091
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
99f5ed63cc5f 10000cpudocker_6 "/10000_cpu.sh" 4 seconds ago Up 3 seconds quirky_cannon

```

```

(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
99f5ed63cc5f 10000cpudocker_6 "/10000_cpu.sh" 4 seconds ago Up 3 seconds quirky_cannon
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f quirky_cannon > 3_6_cpu10000.log
(base) sumit@Sumits-MacBook-Air ~ % cat 3_6_cpu10000.log
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:

```

Screenshot

```

sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 6488.84

General statistics:
total time: 30.0010s
total number of events: 194684

Latency (ms):
min: 0.15
avg: 0.15
max: 20.59
95th percentile: 0.17
sum: 29934.68

Threads fairness:
events (avg/stddev): 194684.0000/0.00
execution time (avg/stddev): 29.9347/0.00

```

Detailed report:

Iteration No	Events per second
1	6488.84
2	6477.29
3	6627.75
4	6630.42
5	6617.29

Observations:

Average events per second	6568.318
Minimum Number of events per second recorded	6477.29
Maximum Number of events per second recorded	6630.42

QEMU Results for 3 GB 6 cores for max-prime = 100000 and time = 30 seconds

Screenshot

```
[sumit@sumitqemu:~$ ./100000_cpu.sh
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
events per second: 289.31

General statistics:
total time: 30.0033s
total number of events: 8681

Latency (ms):
min: 3.36
avg: 3.45
max: 11.42
95th percentile: 3.55
sum: 29974.80

Threads fairness:
events (avg/stddev): 8681.0000/0.00
execution time (avg/stddev): 29.9748/0.00
```

Detailed report:

Iteration No	Events per second
1	289.31
2	289.84
3	290.23
4	290.01
5	291.77

Observations:

Average events per second	290.232
Minimum Number of events per second recorded	291.77
Maximum Number of events per second recorded	289.31

Docker results for 3 GB 6 cores for max-prime = 100000 and time = 30 seconds

```
(base) sumit@Sumits-MacBook-Air ~ % cat 100000_CPU_Dockerfile
FROM arm64v8/ubuntu:20.04
RUN apt-get update
RUN apt-get install -y sysbench
COPY 100000_cpu.sh /
ENTRYPOINT ["./100000_cpu.sh"]
(base) sumit@Sumits-MacBook-Air ~ % cat 100000_cpu.sh
#!/bin/bash
sysbench cpu --threads=1 --cpu-max-prime=100000 --time=30 run
(base) sumit@Sumits-MacBook-Air ~ % docker build -t 100000cpudocker_6 . -f 100000_CPU_Dockerfile
[+] Building 0.3s (9/9) FINISHED
=> [internal] load build definition from 100000_CPU_Dockerfile
=> => transferring dockerfile: 184B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/arm64v8/ubuntu:20.04
=> [internal] load build context
=> => transferring context: 364B
=> [1/4] FROM docker.io/arm64v8/ubuntu:20.04
=> CACHED [2/4] RUN apt-get update
=> CACHED [3/4] RUN apt-get install -y sysbench
=> CACHED [4/4] COPY 100000_cpu.sh /
=> exporting to image
=> => exporting layers
=> => writing image sha256:3e9a4bf1e4e1192c9aaa2e466d45ca8b783385d45296a2dd9e3f71b656a0ba32
=> => naming to docker.io/library/100000cpudocker_6

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-5" -d 100000cpudocker_6
fd2fd93397333cfdd08f7910ac14ba8322993a88cc5d47c8d51cbdd344fd13e
(base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
fd2fd933973 100000cpudocker_6 "./100000_cpu.sh" 3 seconds ago Up 2 seconds nervous_galileo
(base) sumit@Sumits-MacBook-Air ~ % docker logs -f nervous_galileo > 3_6_cpus10000.log
```

Screenshot

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

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

Prime numbers limit: 100000

Initializing worker threads...

Threads started!

CPU speed:
  events per second: 289.17

General statistics:
  total time: 30.0015s
  total number of events: 8676

Latency (ms):
  min: 3.37
  avg: 3.46
  max: 7.17
  95th percentile: 3.49
  sum: 29983.72

Threads fairness:
  events (avg/stddev): 8676.0000/0.00
  execution time (avg/stddev): 29.9837/0.00
```

Detailed report:

Iteration No	Events per second
1	289.17
2	286.87
3	283.76
4	288.26
5	288.80

Observations:

Average events per second	287.372
Minimum Number of events per second recorded	283.76
Maximum Number of events per second recorded	289.17

Conclusion: Even after changing the configuration and increasing the resource, there is no major difference in performance for Docker and QEMU. QEMU is performing slightly better than Docker.

3.2 FILE I/O Testing

QEMU Results for 3 GB 6 cores for Combined random read/write (rndrw)

```
QEMU 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:         9992.15
  fsyncs/s:        12854.78

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   156.13

General statistics:
  total time:       30.0628s
  total number of events: 684813

Latency (ms):
  min:              0.00
  avg:              0.70
  max:              67.59
  95th percentile:  1.96
  sum:             479361.69

Threads fairness:
  events (avg/stddev): 42800.8125/437.75
  execution time (avg/stddev): 29.9601/0.00
```

Iteration	Results
1	reads/s: 0.00 writes/s: 9992.15 fsyncs/s: 12854.78
2	reads/s: 0.00 writes/s: 9658.93 fsyncs/s: 12429.61
3	reads/s: 0.00 writes/s: 8924.33 fsyncs/s: 11487.49
4	reads/s: 0.00 writes/s: 9860.38 fsyncs/s: 12686.89
5	reads/s: 0.00 writes/s: 9674.05 fsyncs/s: 12448.82

Docker Results for 3 GB 6 cores for Combined random read/write (rndrw)

```
((base) sumit@Sumits-MacBook-Air ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
100000cpudocker    latest   3e9a4bf1e4e1  14 hours ago  116MB
100000cpudocker_3gb  latest   3e9a4bf1e4e1  14 hours ago  116MB
100000cpudocker_6    latest   3e9a4bf1e4e1  14 hours ago  116MB
100000cpudocker_6    latest   9a9867686617  15 hours ago  116MB
100000cpudocker_6    latest   9a9867686617  15 hours ago  116MB
<none>          <none>   cb211e545529  15 hours ago  116MB
<none>          <none>   276b5558b578  15 hours ago  116MB
1000cpudocker1     latest   06d58584b4ba  15 hours ago  116MB
1000cpudocker_3gb  latest   06d58584b4ba  15 hours ago  116MB
1000cpudocker_6    latest   06d58584b4ba  15 hours ago  116MB
1000cpudocker     latest   7ad29dcc6c7a  16 hours ago  116MB
new_docker_image    latest   4ffcb1d09133  17 hours ago  65.7MB
arm64v8/ubuntu     20.04   c9eb527b091d  36 hours ago  65.7MB
hello-world        latest   463310942d63  10 months ago  9.14KB
((base) sumit@Sumits-MacBook-Air ~ % docker run -it --memory="3g" --cpuset-cpus="0-5" -d arm64v8/ubuntu:20.04
b82ceef4c50d5c9b27f38f9c982cb7636428547ed242f77036c01e6256405768
((base) sumit@Sumits-MacBook-Air ~ % docker ps
CONTAINER ID IMAGE           COMMAND      CREATED      STATUS      PORTS      NAMES
b82ceef4c50d  arm64v8/ubuntu:20.04 "/bin/bash"  7 seconds ago Up 6 seconds      vigorous_newton
((base) sumit@Sumits-MacBook-Air ~ % docker inspect b82ceef4c50d
{
  "Id": "b82ceef4c50d5c9b27f38f9c982cb7636428547ed242f77036c01e6256405768",
  "Created": "2023-02-02T23:48:30.906904511Z",
  "Path": "/bin/bash",
  "Args": [],
  "State": {
    "Status": "running",
    "Running": true,
    "Paused": false,
    "Restarting": false,
    "OOMKilled": false,
    "Dead": false,
    "Pid": 3129,
    "ExitCode": 0,
    "Error": "",
    "StartedAt": "2023-02-02T23:48:31.376306886Z",
    "FinishedAt": "2000-01-01T00:00:00Z"
  },
  "Image": "sha256:c9eb527b091d53464371291e6470289f0b4f9aab431da2da2ed35e43149bab2b",
  "Isolation": "",
  "CpuShares": 0,
  "Memory": 3221225472,
  "NanoCpus": 0,
  "CgroupParent": "",
  "BlkioWeight": 0,
  "BlkioWeightDevice": [],
  "BlkioDeviceReadBps": null,
  "BlkioDeviceWriteBps": null,
  "BlkioDeviceReadIOps": null,
  "BlkioDeviceWriteIOps": null,
  "CpuPeriod": 0,
  "CpuQuota": 0,
  "CpuRealtimePeriod": 0,
  "CpuRealtimeRuntime": 0,
  "CpusetCpus": "0-5",
  "CpusetMems": "",
  "Devices": [],
  "DeviceCgroupRules": null,
  "DeviceRequests": null,
  "KernelMemory": 0
}
```

```

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:        7418.32
  fsyncs/s:       9560.44

Throughput:
  read, MiB/s:      0.00
  written, MiB/s:   115.91

General statistics:
  total time:        30.1268s
  total number of events: 509490

Latency (ms):
  min:                0.00
  avg:                0.94
  max:              104.91
  95th percentile:    2.76
  sum:            479404.83

Threads fairness:
  events (avg/stddev): 31843.1250/516.16
  execution time (avg/stddev): 29.9628/0.00

```

Iteration	Results
1	reads/s: 0.00 writes/s: 7418.32 fsyncs/s: 9560.44
2	reads/s: 0.00 writes/s: 7408.39 fsyncs/s: 9548.02
3	reads/s: 0.00 writes/s: 7220.23 fsyncs/s: 9309.59
4	reads/s: 0.00 writes/s: 7310.45 fsyncs/s: 9421.63
5	reads/s: 0.00 writes/s: 6752.18 fsyncs/s: 8708.85

Conclusion: If we compare the results with the 3GB and 3 core settings then we can conclude the File I/O operations performed well.

Performance Analysis

In this section, we have listed out the disk and CPU utilization for both QEMU and docker desktop.

1. QEMU Disk Utilization

2 GB RAM 2 Cores	Random Read Write	written, MiB/s = 769.38
3 GB RAM 3 Cores	Random Read Write	written, MiB/s = 767.96
3 GB RAM 6 Cores	Random Read Write	written, MiB/s = 789.35

Processes: 357 total, 4 running, 353 sleeping, 2857 threads																	17:11:04	
Load Avg: 2.63, 2.28, 2.26 CPU usage: 5.46% user, 24.10% sys, 70.42% idle SharedLibs: 165M resident, 37M data, 7104K linkedit.																		
MemRegions: 686424 total, 821M resident, 75M private, 710M shared. PhysMem: 7550M used (1712M wired), 81M unused.																		
VH: 192T vszie, 37800 framework vszie, 54428764(2584) swapins, 57556253(1008) swapouts. Networks: packets: 10533276/10G in, 2757486/2313M out.																		
Disks: 39836055/1772G read, 22147039/1281G written.																		
PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID	FAULTS	COW
43059	qemu-system-	97.1	15:16:07	10/1	1	32	589M	0B	5834M-	43059	36159	running	*[1]	0.00000	0.00000	501	18504053+	415
0	kernel_task	67.9	08:02:00	492/8	0	0	13M+	0B	0B	0	0	running	*[0]	0.00000	0.00000	0	38195	0
36153	Terminal	27.3	01:24:57	9	3	351+	158M+	18M+	33M-	36153	1	sleeping	*[765]	0.17460	0.00000	501	1387585+	763
361	WindowServer	5.8	03:53:12	22	6	3440+	1045M-	0B-	290M+	361	1	sleeping	*[1]	0.00000	0.35132	88	22781708+	222641
48113	top	4.4	08:02:04	1/1	0	33+	6193K	0B	1184K+	48113	48049	running	*[1]	0.00000	0.00000	0	9023+	62
918	Google Chrom	2.7	01:52:30	43	1	2868	478M-	0B	297M+	918	1	sleeping	*[0]12467]	0.00000	0.00000	501	47218838+	134289
31829	ScreenTimeAg	1.8	03:05:44	5	4	162+	19M+	0B	8432K-	31829	1	sleeping	*[13209]	0.00000	0.00000	501	899472+	201
859	suggestd	1.7	02:52:34	8	7	577+	30M+	0B	17M-	859	1	sleeping	*[1441]	0.00000	0.00000	501	1286641+	338
52121	Google Chrom	1.7	03:51:34	18	1	223	113M+	0B	93M+	918	918	sleeping	*[4]	0.00000	0.00000	501	2562533+	729
63789	PerfPowerSer	1.4	03:17:37	5	3	4435+	14M+	0B	3504K-	63789	1	sleeping	*[1948]	0.05584	0.00000	0	808751+	463
935	Google Chrom	1.2	23:17:39	12	1	196	61M-	0B	23M+	918	918	sleeping	*[3]	0.00000	0.00000	501	7390280+	664
17798	storagekitd	1.1	00:33:80	7	3	90+	33M+	0B	28M-	17798	1	sleeping	*[47]	0.00000	0.00000	0	314274+	97
687	fontd	1.1	00:51:79	5	4	206+	15M+	0B	7728K-	687	1	sleeping	*[34808]	0.00000	0.00000	501	756187+	74
934	Google Chrom	1.1	76:39:91	24	5	543	885M	0B	360M+	918	918	sleeping	*[5]	0.00000	0.00000	501	16354611+	921
464	searchpartyd	0.9	02:02:90	5	3	104+	7361K+	0B	3184K-	464	1	sleeping	*[15148]	0.00000	0.89543	0	687887+	195
377	airportd	0.6	22:31:61	11	9	1633	20M	0B	14M+	377	1	sleeping	*[5975]446	0.00000	0.00000	0	1626124	144
334	locationd	0.6	06:35:53	7	4	196+	9153K	0B	4720K+	334	1	sleeping	*[0]109619+]	0.00000	0.55837	205	1329692+	163
314	mds	0.6	13:37:11	10	7	386+	104M+	0B	92M+	314	1	sleeping	*[1]	4.33934	0.18287	0	4878854+	940
14894	bluetoothd	0.5	04:16:72	18	4	274	9857K	0B	4112K+	14894	1	sleeping	*[1]	1.60623	0.00000	0	787619+	140
566	mds_stores	0.5	25:44:63	8	6	118	77M+	0B	61M-	566	1	sleeping	*[1]	0.00000	0.45798	0	8313100+	2667
483	softwareupda	0.3	02:35:96	10	6	226+	83M	0B	69M+	483	1	sleeping	*[542]	0.00000	0.14661	200	1565871	3504
1	launchd	0.3	15:32:37	4	3	2256+	18M	0B	6240K+	1	0	sleeping	*[0]	0.00000	0.15387	0	2761764+	28135+
727	Logi Bolt	0.3	12:47:84	100	1	427	1603M	0B	1593M+	727	1	sleeping	*[1]	0.00000	0.00000	501	5374770+	1955
292	fseventsds	0.3	18:39:17	19	1	308	5937K	0B	1888K+	292	1	sleeping	*[1]	0.01532	0.00000	0	1832569+	58
453	mDNSResponde	0.3	04:49:34	4	2	164	6033K	0B	2800K+	453	1	sleeping	*[1]	0.00000	0.00000	65	585621+	123
732	sharingd	0.3	04:32:34	5	1	282	29M	0B	22M+	732	1	sleeping	*[1]	0.00000	0.00000	501	1518118+	321
592	distnoted	0.3	02:05:56	2	1	298	3153K	0B	896K+	592	1	sleeping	*[1]	0.00000	0.29424	501	344687	45
536	com.avg.acti	0.2	04:41:64	1	0	18	3137K	0B	1600K	536	1	sleeping	*[1]	0.00000	0.00000	0	445268+	60
47001	advertisers	0.2	00:00:00	2	1	47	3550K	0B	1760K	47001	1	sleeping	*[1]	0.17375	0.00000	501	10361	0

2. QEMU CPU Utilization

Percentage of CPU used – 2.2%

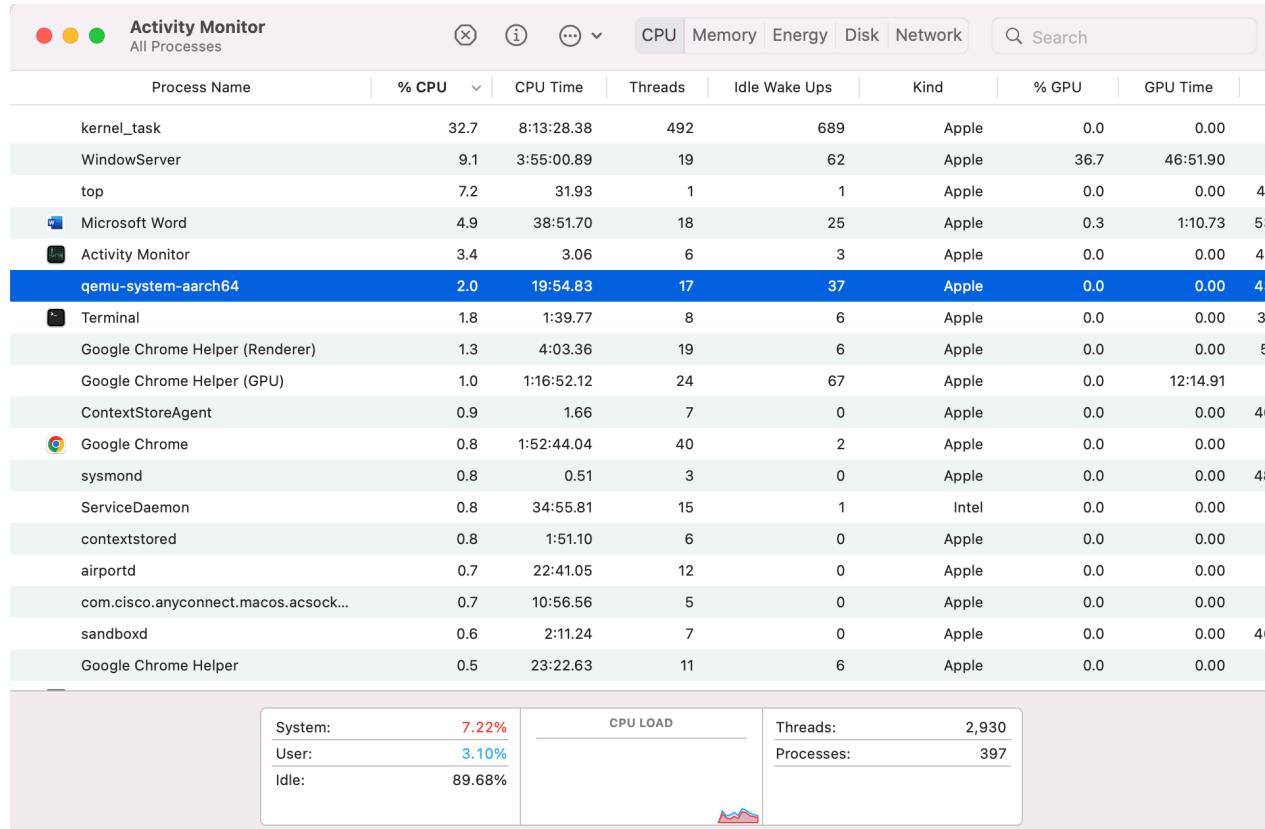
System – 10.16%

Idle – 87.95%

Kernel Usage – user = 1.88%

```
Processes: 381 total, 2 running, 379 sleeping, 2894 threads
Load Avg: 2.29, 2.19, 2.22 CPU usage: 1.88% user, 10.16% sys, 87.95% idle SharedLibs: 186M resident, 60M data, 10M linkedit.
MemRegions: 697487 total, 1367M resident, 71M private, 944M shared. PhysMem: 7552M used (1661M wired), 80M unused.
VH: 283T vsize, 3780M framework vsize, 54697347(176) swapins, 57824441(0) swapouts. Networks: packets: 10585137/10G in, 2761958/2314M out.
Disks: 39984905/1780G read, 22184038/1286G written.                                         17:22:18

PID  COMMAND      XCPU TIME #TH #WQ #PORT MEM PURG CMPRS PGRP PPID STATE BOOSTS %CPU_ME %CPU_OHRS UID FAULTS COW
0  kernel_task  70.2 08:10:22 492/9 0 0 12M 0B 0 0 0  running 0[0] 0.00000 0.00000 0 38241 0
48891 top        5.6 00:12:18 1/1 0 36 6017K 0B 1248K+ 48891 48049 running *0[1] 0.00000 0.00000 0 7872+ 63
361 WindowServer 3.1 03:54:24 19 4 3436- 1016M- 0B 298M+ 361 1 sleeping *0[1] 0.00000 0.32001 88 22886454+ 272631
36153 Terminal  2.7 01:34:77 7 1 347 158M+ 0B 34M- 36153 1 sleeping *0[1] 0.00000 0.00000 501 16546093+ 764
599 com.cisco.an 2.5 10:52:96 18 7 124+ 18M+ 0B 4864K- 599 1 sleeping *0[1] 0.00000 0.00000 0 964085+ 93
43059 qemu-system- 2.2 19:50:49 18 1 32 5898M 0B 5846M 43059 36159 sleeping *0[1] 0.00000 0.00000 501 18987256+ 415
45357 Google Chrom 1.4 01:48:31 17 1 214 182M- 0B 81M- 918 918 sleeping *0[3] 0.00000 0.00000 501 3803916+ 793
935 Google Chrom 1.1 23:21:07 12 1 202 61M+ 0B 25M 918 918 sleeping *0[3] 0.00000 0.00000 501 7420962+ 664
52121 Google Chrom 1.0 03:59:75 19 1 225 113M 0B 87M- 918 918 sleeping *0[4] 0.00000 0.00000 501 2585546+ 729
53694 Microsoft Wo 1.0 38:38:76 18 6 2897 724M 0B 509M+ 53694 1 sleeping *0[4936] 0.00000 0.00000 501 2525722 3195
377 airportd   0.9 22:38:56 13 11 1636 21M 0B 12M+ 377 1 sleeping *5975(446) 0.00000 0.00000 0 1631146+ 144
934 Google Chrome 0.9 76:48:64 24 5 545 885M 0B 360M 918 918 sleeping *1[5] 0.00000 0.00000 501 16388416 921
1408 com.avg.Anti 0.8 07:50:89 4 3 32+ 18M+ 0B 5952K- 1408 1 sleeping *0[1] 0.00000 0.00000 0 1150144+ 119
918 Google Chrom 0.8 01:52:40 42 1 2876 479M 0B 272M- 918 1 sleeping *0[12475] 0.00000 0.00000 501 47400139+ 134530
52534 com.docker.b 0.5 02:54:06 26 1 100 33M 0B 24M- 52534 52532 sleeping *0[1] 0.00000 0.00000 501 6814856+ 2837681+
453 mDNSResponde 0.4 04:52:49 4 2 176 6081K 0B 2592K 453 1 sleeping *0[1] 0.00000 0.00000 65 587383 123
49198 mdworker_sha 0.4 00:00:06 3 1 47 2881K+ 0B 960K 49198 1 sleeping *0[1] 0.38717 0.00000 501 993+ 91
727 Logi Bolt  0.3 12:50:09 100 1 428 1603M 0B 1593M- 727 1 sleeping *0[1] 0.00000 0.00000 501 5382363+ 1955
566 mds_stores  0.3 25:46:97 5 3 106 78M 0B 63M- 566 1 sleeping *0[1] 0.00000 0.32244 0 8333852+ 2667
732 sharingd   0.3 04:35:03 5 1 283 29M 0B 19M+ 732 1 sleeping *0[1] 0.00000 0.00000 501 1524421 321
1397 com.avg.Anti 0.3 16:36:33 28 3 70 7105K 0B 4032K+ 1281 1281 sleeping *0[1] 0.00000 0.00000 0 418420 65
45353 Google Chrom 0.2 05:04:40 24 1 425 245M 0B 209M- 918 918 sleeping *0[7] 0.00000 0.00000 501 8213718+ 837
314 mds       0.2 13:38:67 7 4 393 184M 0B 95M+ 314 1 sleeping *0[1] 0.00000 0.06473 0 4892236+ 940
473 symptomsd  0.2 02:37:68 3 2 138 6641K 0B 3168K- 473 1 sleeping *0[1616] 0.00000 0.00000 24 1829178+ 128
31381 Google Chrom 0.2 00:18:68 16 1 137 30M 0B 25M- 918 918 sleeping *0[3] 0.00000 0.00000 501 757256+ 664
1327 com.avg.prox 0.1 09:40:55 36 1 75 42M+ 0B 37M- 1327 1 sleeping *0[1] 0.00000 0.00000 0 4425137+ 173
292 fseventsdf 0.1 10:40:35 14 1 298 5921K 0B 2704K 292 1 sleeping *0[1] 0.02311 0.00000 0 1837844+ 58
1744 com.apple.Am 0.1 05:45:79 4 2 73 3761K 0B 2416K 1744 1 sleeping *0[1] 0.12820 0.00000 0 203543+ 88
254 searchlightd 0.1 06:31:05 3 3 121 6512K 0B 2079K 354 1 sleeping *0[1] 0.11757 0.00000 0 204562 86
```



1. Docker Disk Utilization

2 GB RAM 2 Cores	Random Read Write	written, MiB/s = 286.00
3 GB RAM 3 Cores	Random Read Write	written, MiB/s = 162.36
3 GB RAM 6 Cores	Random Read Write	written, MiB/s = 90.01

2. Docker CPU Utilization

CPU% used – 40.1% when container is running

Idle – 60.23%

System – 14.1%,

Kernel User usage = 5.74%

Processes: 401 total, 2 running, 399 sleeping, 2934 threads																		
Load Avg: 2.67, 2.33, 2.17 CPU usage: 5.74% user, 14.1% sys, 80.23% idle SharedLibs: 111M resident, 28M data, 5344K linkedit.																		
MemRegions: 689252 total, 742M resident, 61M private, 794M shared, PhysMem: 7104M used (1605M wired), 527M unused.																		
VM: 209T vsize, 3780M framework vsize, 55159713(96) swapins, 58302175(0) swapouts Networks: packets: 16642414/10G in, 2779144/2318M out. Disks: 40249181/1811G read, 22242211/1293G written.																		
PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID	STATE	BOOSTS	%CPU_ME	%CPU_OTHRS	UID	FAULTS	COW
0	kernel_task	88.2	08:19:19	492/9	0	0	12M+	0B	0B	0	0	running	0[0]	0.00000	0.00000	0	38327	0
53086	Docker Deskt	40.1	18:35:95	21	1	165	154M-	0B	41M-	52534	52582	sleeping	*[1]	0.00000	0.00000	501	11790651+	510
50144	top	5.7	08:22:26	1/1	0	32-	6513K	0B	1728K	50144	48049	running	*[1]	0.00000	0.00000	0	125176+	64
361	WindowServer	2.9	03:56:14	18	3	3624-	1057M+	4672K+	316M-	361	1	sleeping	*[1]	0.00000	0.29958	88	23077625+	273
338	ServiceDaemon	2.0	35:02:94	18	4	127	66M+	0B	47M-	338	1	sleeping	*[1]	0.00000	0.02035	0	35018918+	213
599	com.cisco.an	1.3	11:04:24	5	2	123	10M	0B	6464K	599	1	sleeping	*[1]	0.00000	0.00000	0	913155	93
34519	gemu-system-	1.2	22:47:24	19	0	34	5726M	0B	5653M-	52534	34499	sleeping	*[1]	0.00000	0.00000	501	22221870+	273
52121	Google Chrom	1.1	04:11:19	19	1	225	113M+	0B	93M-	918	918	sleeping	*[4]	0.00000	0.00000	501	2616752+	729
56133	Terminal	1.1	01:51:40	7	1	345	149M	0B	52M	36153	1	sleeping	*[855]	0.00015	0.00000	501	2301528	837
918	Google Chrom	1.0	01:52:52	42	1	2868	479M+	0B	305M-	918	1	sleeping	*[12477]	0.00000	0.00000	501	47699764+	134
377	airportd	0.8	22:46:31	11	9	1635	20M	0B	14M	377	1	sleeping	*[975][446]	0.00000	0.00000	0	1648144	144
934	Google Chrom	0.8	76:59:86	24	5	543	885M	0B	359M	918	918	sleeping	*[5]	0.00000	0.00000	501	16427943	921
52534	com.docker.b	0.6	02:57:10	26	1	100	34M	0B	26M-	52534	52532	sleeping	*[1]	0.00000	0.00000	501	6932796+	288
935	Google Chrom	0.5	23:26:07	12	1	196	61M	0B	26M-	918	918	sleeping	*[3]	0.09744	0.00000	501	7482579+	664
453	mDNSResponde	0.4	04:55:91	3	1	176	6033K	0B	2960K-	453	1	sleeping	*[1]	0.00000	0.00000	65	590683+	123
292	fseventsds	0.3	10:41:97	15	1	303	5857K	0B	2688K-	292	1	sleeping	*[1]	0.02035	0.00000	0	1845829+	58
727	Log Bolt	0.3	12:53:11	100	1	428	1603M	0B	1594M-	727	1	sleeping	*[1]	0.00000	0.00000	501	5392054+	195
1397	com.avg.Anti	0.3	16:48:29	28	3	72	7153K	0B	3872K-	1281	1281	sleeping	*[1]	0.00000	0.00000	0	421849+	65
732	sharingd	0.3	04:38:12	5	1	282	29M	0B	23M	732	1	sleeping	*[1]	0.00000	0.00000	501	1534247	321
314	mds	0.2	13:40:94	11	8	391-	105M	0B	95M-	314	1	sleeping	*[1]	0.13938	0.00000	0	4918078+	940
536	com.avg.acti	0.1	04:43:93	1	0	18	3137K	64K	1616K	536	1	sleeping	*[1]	0.00000	0.00000	0	450957	60
566	mds_stores	0.1	25:52:11	10	8	119	81M	0B	67M-	566	1	sleeping	*[1]	0.00000	0.13938	0	8378859+	266
14894	bluetoothd	0.1	04:24:25	10	4	274	9761K	0B	3760K	14894	1	sleeping	*[1]	0.04838	0.00000	0	801752	140
334	locationd	0.1	06:40:52	7	4	204	9233K	0B	4912K-	334	1	sleeping	*[0][13244+]	0.00000	0.04838	205	1339557+	163
354	corebrightne	0.1	05:13:63	2	1	121	4481K	0B	2928K	354	1	sleeping	*[1]	0.11112	0.00000	0	1053962+	96
52582	Docker Deskt	0.1	00:59:24	28	1	404	85M	0B	75M-	52534	52534	sleeping	*[639]	0.00000	0.00000	501	1509574+	887
483	softwareupda	0.1	02:38:45	7	4	219	83M	0B	75M-	483	1	sleeping	*[542]	0.00000	0.08757	200	1583397+	350
1744	com.apple.Am	0.1	05:46:83	4	2	73	3761K	0B	2416K	1744	1	sleeping	*[1]	0.09330	0.00000	0	204909	88
1	launchd	0.1	15:36:58	4	3	2288-	18M	0B	8528K-	1	0	sleeping	*[0]	0.00000	0.02852	0	2796871+	282
45996	backupd	0.1	00:00:	35	4	40+	2753K	0B	1504K-	45996	1	sleeping	*[1]	0.00000	0.07890	0	7633+	106

Activity Monitor
All Processes

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	Kind	% GPU	GPU Time	F
kernel_task	51.4	8:16:38.73	492	1696	Apple	0.0	0.00	
WindowServer	15.6	3:55:48.91	21	141	Apple	16.8	47:08.38	
top	6.9	6.16	1	1	Apple	0.0	0.00	50
Docker Desktop Helper (Renderer)	6.1	18:19.52	21	4	Apple	0.0	0.00	53
ServiceDaemon	2.4	35:00.03	18	2	Intel	0.0	0.00	
launchd	2.0	15:35.98	3	1	Apple	0.0	0.00	
com.cisco.anyconnect.macos.acsock...	1.6	11:00.95	8	0	Apple	0.0	0.00	
Activity Monitor	1.5	7.92	5	2	Apple	0.0	0.00	49
screencapture	1.2	0.20	3	0	Apple	0.0	0.00	50
deleted	1.2	6.08	5	0	Apple	0.0	0.00	46
Google Chrome Helper (Renderer)	1.2	4:08.11	19	6	Apple	0.0	0.00	5
Google Chrome Helper (GPU)	1.2	1:16:56.87	24	67	Apple	0.0	12:14.91	
CleanMyMac X Menu	1.1	8:05.76	5	0	Intel	0.0	0.00	
Microsoft Word	1.0	39:23.05	19	20	Apple	0.2	1:11.16	53
Google Chrome	1.0	1:52:48.66	39	4	Apple	0.0	0.00	
airportd	0.7	22:44.03	11	0	Apple	0.0	0.00	
Terminal	0.7	1:46.25	7	4	Apple	0.0	0.00	36
com.avg.Antivirus.EndpointSecurity	0.7	16:39.13	28	1	Apple	0.0	0.00	

System: **22.47%** CPU LOAD Threads: **2,943**
User: **4.38%** .. Processes: **410**

Git Repository Information:

Account name	Sumit8698
Repository name	COEN-241-Cloud-Computing
Folder which contains HW1	HW1
Link to repository	https://github.com/sumit8698/COEN241-Cloud-Computing

Optional Part:

Docker File

```
FROM sumit8698/sysbench_ubuntu_docker

COPY docker-script.sh /docker-script.sh

COPY cpu-script.sh /cpu-script.sh

COPY fileio-script.sh /fileio-script.sh

RUN chmod +x docker-script.sh

RUN chmod +x cpu-script.sh

RUN chmod +x fileio-script.sh

ENTRYPOINT bash docker-script.sh
```

Vagrant File:

```
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/hirsute64"

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

  config.vm.synced_folder "qemu", "/"
  config.vm.provision "shell", path: "vagrant_setup.sh"
end
```

For all the automation scripts, please check the link below.

Link: <https://github.com/sumit8698/COEN241-Cloud-Computing>