Santa Clara University

COEN 241: Cloud Computing

HW 1: System vs OS Virtualization

Ankita Kumar (W1620637)

HOST ENVIRONMENT CONFIGURATION:

Detailed system configurations (CPU, Mem, etc...) on which all the experiments have been performed.

Hardware Overview:

Model Name:	MacBook Air
Model Identifier:	MacBookAir10,1
Chip:	Apple M1
Total Number of Cores:	8 (4 performance and 4 efficiency)
Memory:	8 GB
System Firmware Version:	6723.101.4
OS Loader Version:	6723.101.4
Activation Lock Status:	Enabled

QEMU INSTALLATION ON M1 & IT'S CONFIGURATION:

Below are the main steps to install QEMU manually:

a. Install Xcode command line tools.

```
xcode-select --install
```

b. Install Homebrew arm64

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> /Users/$(logname)/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"
```

c. Install necessary packages for building

```
brew install libffi gettext glib pkg-config autoconf automake pixman ninja
```

d. Clone Qemu

```
git clone https://github.com/qemu/qemu
```

e. Change directory to Qemu repository

```
cd qemu
```

f. Checkout to commit dated June 03, 2021 v6.0.0

```
git checkout 3c93dfa42c394fdd55684f2fbf24cf2f39b97d47
```

g. Apply patch series v8 by Alexander Graf

```
curl https://patchwork.kernel.org/series/485309/mbox/ | git am
```

h. Building Qemu installer

```
mkdir build && cd build
../configure --target-list=aarch64-softmmu
make -j8
```

i. Install Qemu

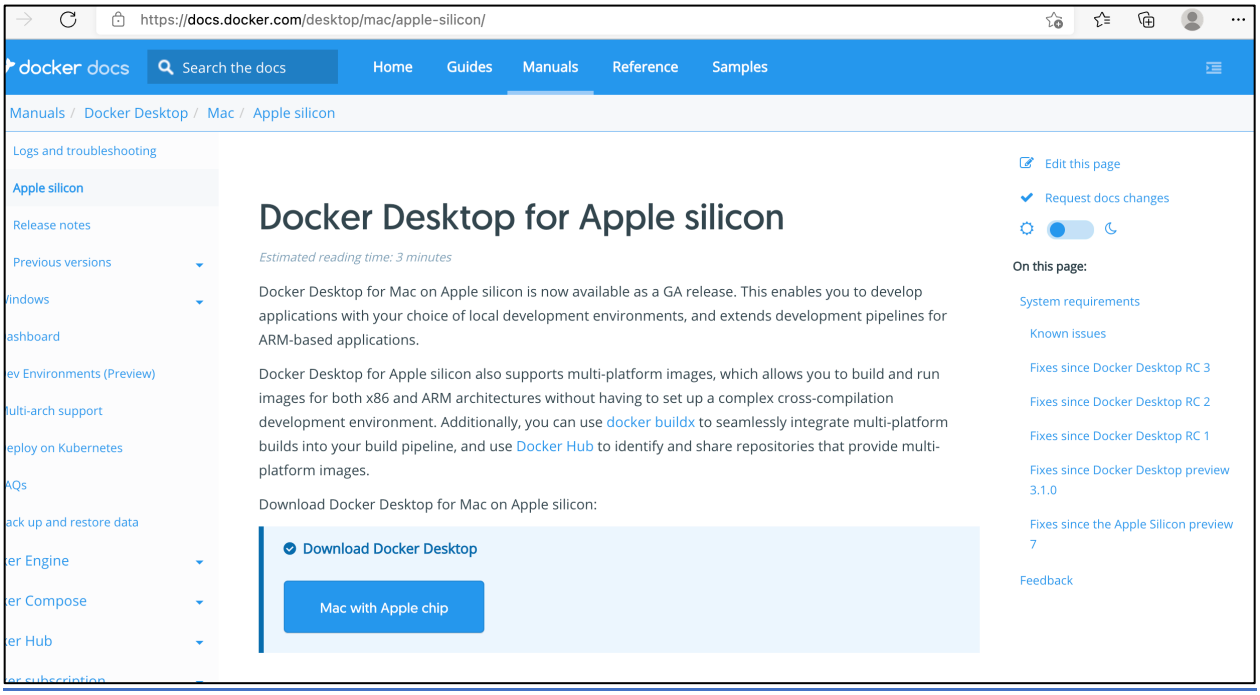
```
sudo make install
```

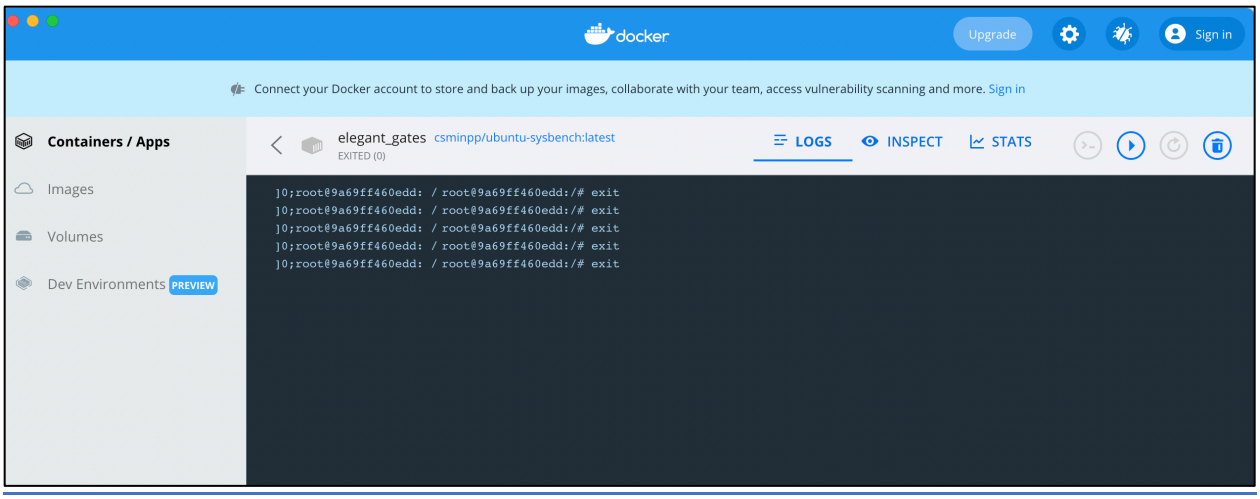
DOCKER INSTALLATION ON M1 & IT'S CONFIGURATION:

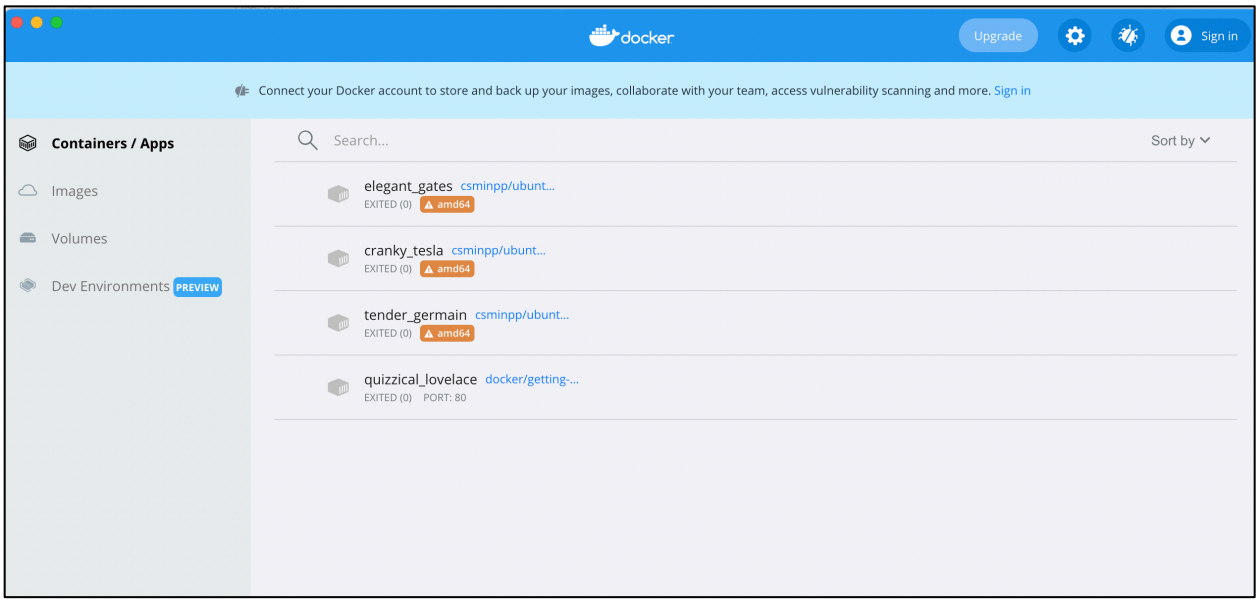
Installing Docker was straight forward through this link.

Docker Desktop for Apple silicon | Docker Documentation

Further directly using the CLI to execute the scripts.




docker
Upgrade
Sign in
Connect your Docker account to store and back up your images, collaborate with your team, access vulnerability scanning and more. Sign in
Containers / Apps
Images
Volumes
Dev Environments PREVIEW
elegant_gates csminpp/ubuntu-sysbench:latest
EXITED (0)
LOGS
INSPECT
STATS
]0;root@9a69ff460edd: / root@9a69ff460edd:/# exit
]0;root@9a69ff460edd: / root@9a69ff460edd:/# exit
]0;root@9a69ff460edd: / root@9a69ff460edd:/# exit
]0;root@9a69ff460edd: / root@9a69ff460edd:/# exit
]0;root@9a69ff460edd: / root@9a69ff460edd:/# exit


docker
Upgrade
Sign in
Connect your Docker account to store and back up your images, collaborate with your team, access vulnerability scanning and more. Sign in
Containers / Apps
Images
Volumes
Dev Environments PREVIEW
Search...
Sort by
elegant_gates csminpp/ubunt...
EXITED (0) amd64
cranky_tesla csminpp/ubunt...
EXITED (0) amd64
tender_germain csminpp/ubunt...
EXITED (0) amd64
quizzical_lovelace docker/getting-...
EXITED (0) PORT: 80

QEMU VM ON M1 & IT'S CONFIGURATION:

Create Ubuntu Server using QEMU on Silicon based Apple Macs. First run, close manually after installation aka first reboot.

1. Install ISO library using below CLI:

   curl https://cdimage.ubuntu.com/releases/20.04/release/ubuntu-20.04.2-live-server-arm64.iso -o ubuntu-lts.iso

2. Create Virtual Hardisk on VM of size 25 GB.

   qemu-img create -f qcow2 virtual-disk.qcow2 25G

3. Creating File for UEFI variables.

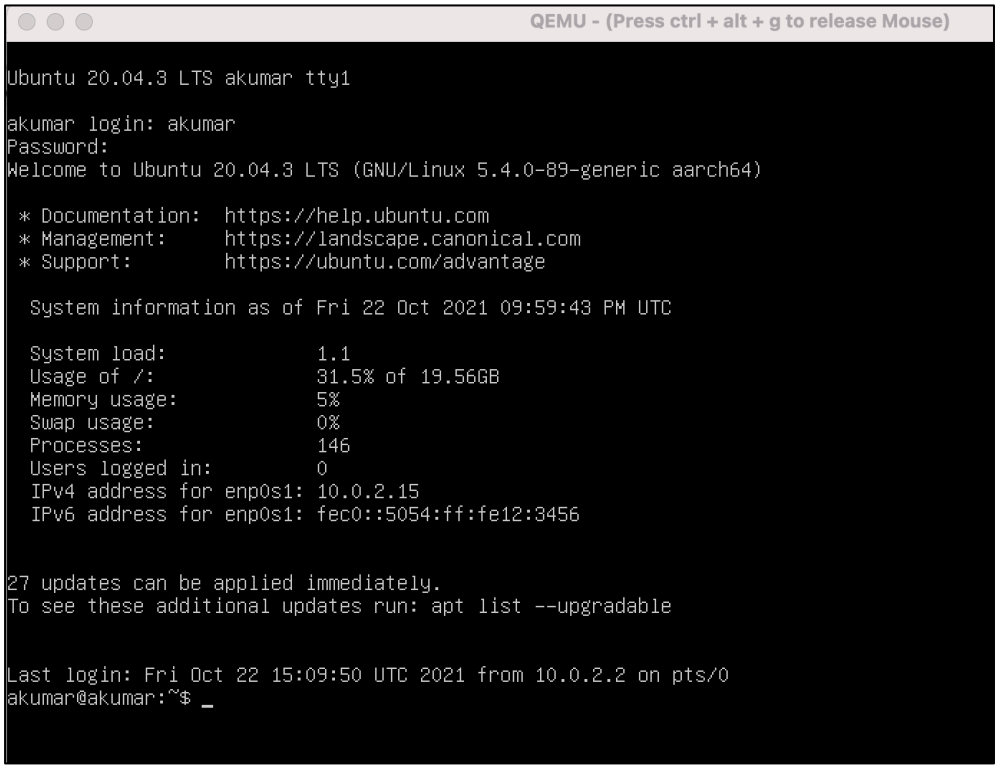
   dd if=/dev/zero conv=sync bs=1m count=64 of=ovmf_vars.fd

4. *Run the below CLI argument to install VM with these configs.*

```
qemu-system-aarch64 \
  -machine virt,accel=hvf,highmem=off \
  -cpu cortex-a72 -smp 4 -m 4G \
  -device virtio-gpu-pci \
  -device virtio-keyboard-pci \
  -drive "format=raw,file=edk2-aarch64-code.fd,if=pflash,readonly=on" \
  -drive "format=raw,file=ovmf_vars.fd,if=pflash" \
  -drive "format=qcow2,file=virtual-disk.qcow2" \
  -cdrom ubuntu-lts.iso
```

5. The above commands will download Ubuntu Image file. Complete installation manually. Run the above same command line arguments again without -cdrom parameter.
   Second run for ubuntu server:

```
qemu-system-aarch64 \
  -machine virt,accel=hvf,highmem=off \
  -cpu cortex-a72 -smp 4 -m 4G \
  -device virtio-gpu-pci \
  -device virtio-keyboard-pci \
  -drive "format=raw,file=edk2-aarch64-code.fd,if=pflash,readonly=on" \
  -drive "format=raw,file=ovmf_vars.fd,if=pflash" \
  -drive "format=qcow2,file=virtual-disk.qcow2"
```



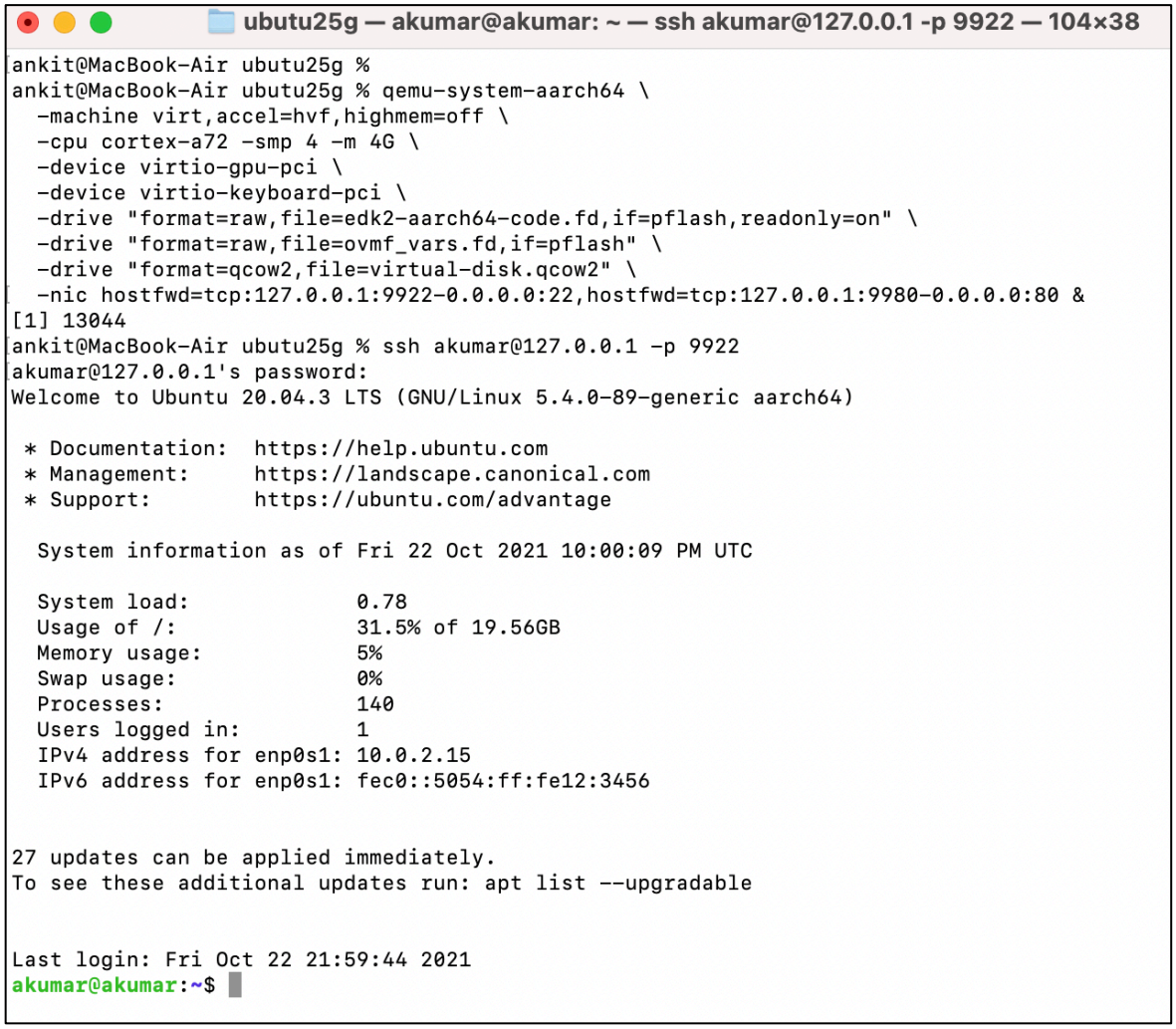
CONNECT UBUNTU VIA SSH AND SERVE PORT 80 AT LOCALHOST.

1. Start using server:

```
qemu-system-aarch64 \
  -machine virt,accel=hvf,highmem=off \
  -cpu cortex-a72 -smp 4 -m 4G \
  -device virtio-gpu-pci \
  -device virtio-keyboard-pci \
  -drive "format=raw,file=edk2-aarch64-code.fd,if=pflash,readonly=on" \
  -drive "format=raw,file=ovmf_vars.fd,if=pflash" \
  -drive "format=qcow2,file=virtual-disk.qcow2" \
  -nic hostfwd=tcp:127.0.0.1:9922-0.0.0.0:22,hostfwd=tcp:127.0.0.1:9980-0.0.0.0:80 &
```

2. Login via SSH using –

   ssh akumar@127.0.0.1 -p 9922



3. Now we can directly run execute the shell scripts at local host by downloading Sysbench using below argument:

```
$ sudo apt-get install sysbench
```

Test different arguments of QEMU :

1. ***Emulated Machine***

   -M <machine-type>
   -machine <machine-type>`[,<property>[=<value>][,..]]`

   Machine

   - `-machine type=q35,accel=kvm` - Modern chipset (PCIe, AHCI, ...) and hardware virtualization acceleration
   - `-object rng-random,id=rng0,filename=/dev/urandom -device virtio-rng-pci,rng=rng0` - Pass-through for host random number generator. Accelerates startup of e.g. Debian VMs because of missing entropy.

2. ***Processor Type***

   -cpu <model>[,<FEATURE>][...]

3. ***Processor Topology***

   smp <n>[,cores=<ncores>][,threads=<nthreads>][,sockets=<nsocks>][,maxcpus=<maxcpus>]
   Hypervisor and guest operating system limits on processor topology apply.

   Processor

   - `-cpu <CPU>` - Specify a processor architecture to emulate. To see a list of supported architectures, run: **`qemu-system-x86_64 -cpu ?`**
   - `-cpu host` - (Recommended) Emulate the host processor.
   - `-smp <NUMBER>` - Specify the number of cores the guest is permitted to use. The number can be higher than the available cores on the host system. Use `-smp $(nproc)` to use all currently available cores.

4. ***Memory Size***

   -m <megs>

   RAM

   - `-m MEMORY` - Specify the amount of memory (default: 128 MB). For instance: **`-m 256M`** (*M* stands for Megabyte, *G* for Gigabyte).

5. ***Generic Drive:***
   -drive <option>[,<option>[,<option>[,...]]]

   Hard drive

   - `-hda IMAGE.img` - Set a virtual hard drive and use the specified image file for it.
   - `-drive` - Advanced configuration of a virtual hard drive:

6. *Optical Drive:*

- `-cdrom IMAGE.iso` - Set a virtual CDROM drive and use the specified image file for it.
- `-cdrom /dev/cdrom` - Set a virtual CDROM drive and use the host drive for it.
- `-drive` - Advanced configuration of a virtual CDROM drive:

CPU PERFORMANCE TESTING:

- We are testing CPU performance using sysbench tool in both Ubuntu VM and Docker Container for below CPU parameters: ***cpu-max-prime=(10000 .. 80000)th. & THREAD =1***
- Each Sysbench measurement is repeated at least 5 times for each test case

Automation for FILE IO performance test

Automation for CPU performance test is done using Shell script program.

Executable File | 18 lines (17 sloc) | 325 Bytes

```
1  #!/bin/bash
2  #Loop
3  exec 3>&1 4>&2
4  trap 'exec 2>&4 1>&3' 0 1 2 3
5  exec 1>Docker_log.output 2>&1
6  START=1
7  END=5
8  cpm_arr=(10000 20000 30000 40000 50000 60000 70000 80000)
9  for i in "${cpm_arr[@]}"
10 do
11 echo "CMP Value:" $i
12 for j in $(eval echo "{$START..$END}")
13 do
14 echo "Loop" $j
15 sysbench --test=cpu --cpu-max-prime=$i run
16 done
17 done
18
```

TEST CASE 1:

- **For a single thread**
  sysbench --test=cpu --cpu-max-prime=10000 run

```
ankit@MacBook-Air COEN241-HW1 % cat Docker_log_CPU.output
CMP Value: 10000
Loop 1
WARNING: the --test option is deprecated. You can pass a
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

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


Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 13672861.69

General statistics:
    total time:                          10.0000s
    total number of events:              136733881

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.06
         95th percentile:                        0.00
         sum:                                 3789.48

Threads fairness:
    events (avg/stddev):           136733881.0000/0.00
    execution time (avg/stddev):   3.7895/0.00
```

```
Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  1000.41

General statistics:
    total time:                          10.0014s
    total number of events:              10014

Latency (ms):
         min:                                    0.46
         avg:                                    0.99
         max:                                    2.55
         95th percentile:                        1.04
         sum:                                 9938.85

Threads fairness:
    events (avg/stddev):           10014.0000/0.00
    execution time (avg/stddev):   9.9389/0.00

Loop 2
WARNING: the --test option is deprecated. You can pass a script
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
```

Docker container QEMU VM

For QEMU

Test run	Prime numbers limit	Total time (In Sec)	Min (In ms)	Avg (in ms)	Max (in ms)	CPU Spped (Events /Sec)
1	10000	10.0000s	0	0	0.06	13,672,861.69
2	10000	10.0000s	0	0	0.04	13,695,972.75
3	10000	10.0000s	0	0	0.06	13,744,287.78
4	10000	10.0000s	0	0	0.05	13,743,972.53
5	10000	10.0001s	0	0	0.05	13,741,889.82

For Docker

A	B	C	D	E	F	G
Test run	CMP	Total time (In Sec)	Min (In ms)	Avg (in ms)	Max (in ms)	CPU Spped (Events /Sec)
1	10000	10.0014	0.46	0.99	2.55	1000.41
2	10000	10.001	0.46	1	5.48	994.79
3	10000	10.001	0.46	0.98	3.67	1018.5
4	10000	10.001	0.46	1	4.06	997.51
5	10000	10.0005	0.46	1.01	25.76	987.69

TEST CASE 2 :

- **For a single thread**
  sysbench --test=cpu --cpu-max-prime=20000 run

```
CMP Value: 20000
Loop 1
WARNING: the --test option is deprecated. You can pass
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

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


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 13741889.82

General statistics:
    total time:                          10.0001s
    total number of events:              137424821

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.05
         95th percentile:                        0.00
         sum:                                 3775.06

Threads fairness:
    events (avg/stddev):           137424821.0000/0.00
    execution time (avg/stddev):   3.7751/0.00
```

```
CMP Value: 20000
Loop 1
WARNING: the --test option is deprecated. You can pass a s
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   388.38

General statistics:
    total time:                          10.0020s
    total number of events:              3886

Latency (ms):
         min:                                    1.19
         avg:                                    2.56
         max:                                    5.32
         95th percentile:                        2.66
         sum:                                 9963.47

Threads fairness:
    events (avg/stddev):           3886.0000/0.00
    execution time (avg/stddev):   9.9635/0.00
```

Docker container QEMU VM

For Docker

Test run	Prime numbers lim	Total time (In Sec	Min (In n	Avg (in m	Max (in ms)	CPU Spped (Events /Se
1	20000	10.0001s	0	0	0.05	13,741,889.82
2	20000	10.0001s	0	0	0.05	13,736,544.34
3	20000	10.0001s	0	0	0.06	13,744,655.94
4	20000	10.0001s	0	0	0.05	13,742,488.02
5	20000	10.0001s	0	0	0.04	13,572,744.02

For QEMU

Test run	CMP	Total time (In Sec)	Min (In n	Avg (in ms)	Max (in m	CPU Spped (Events /Sec
1	20000	10.002	1.19	2.56	5.32	388.38
2	20000	10.0008	1.19	2.53	8.43	393.93
3	20000	10.0028	1.19	2.57	4.96	386.65
4	20000	10.002	1.19	2.57	4.06	387.64
5	20000	10.0008	1.19	2.56	8.04	388.34

TEST CASE 3 :

- **For a single thread**
  sysbench --test=cpu --cpu-max-prime=80000 run

```
CMP Value: 80000
Loop 1
WARNING: the --test option is deprecated. You can pass a
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

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


Prime numbers limit: 80000

Initializing worker threads...

Threads started!

CPU speed:
    events per second: 13413099.56

General statistics:
    total time:                          10.0001s
    total number of events:              134136495

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.06
         95th percentile:                        0.00
         sum:                                 3794.00

Threads fairness:
    events (avg/stddev):           134136495.0000/0.00
    execution time (avg/stddev):   3.7940/0.00
```

Docker container

```
CMP Value: 80000
Loop 1
WARNING: the --test option is deprecated. You can pass a
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: 80000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:    56.93

General statistics:
    total time:                          10.0078s
    total number of events:              570

Latency (ms):
         min:                                   11.96
         avg:                                   17.54
         max:                                   22.76
         95th percentile:                       17.95
         sum:                                 9997.29

Threads fairness:
    events (avg/stddev):           570.0000/0.00
    execution time (avg/stddev):   9.9973/0.00
```

QEMU VM

For Docker

Test run	CMP	Total time (In Sec)	Min (In n	Avg (in ms)	Max (in m	CPU Spped (Events /Sec
1	80000	10.0078	11.96	17.54	22.76	56.93
2	80000	10.0126	8.5	17.7	27.83	56.41
3	80000	10.0035	17.35	17.62	21.19	56.66
4	80000	10.004	8.21	1738	20.41	57.46
5	80000	10.0147	11.53	17.71	23.45	56.4

For QEMU

Test run	CMP	Total time (In Sec)	Min (In n	Avg (in ms)	Max (in m	CPU Spped (Events /Sec
1	80000	10.0078	11.96	17.54	22.76	56.93
2	80000	10.0126	8.5	17.7	27.83	56.41
3	80000	10.0035	17.35	17.62	21.19	56.66
4	80000	10.004	8.21	1738	20.41	57.46
5	80000	10.0147	11.53	17.71	23.45	56.4

CPU SPEED PERFORMANCE COMPARISION BETWEEEN DOCKER AND QEMU VM

FILE IO PERFORMANCE TESTING:

- We are testing File IO performance using sysbench tool in both Ubuntu VM and Docker Container for below CPU parameters:

```
sysbench --num-threads=$j --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare
sysbench --num-threads=$j --test=fileio --file-total-size=3G --file-test-mode=rndrw run
sysbench --num-threads=$j --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup
```

- Each Sysbench measurement is repeated at least 5 times for each test case

Automation for FILE IO performance test

- Automation for FILE IO performance test is done using Shell script program. And the logs are saved into a File named: FileIO_log_Docker.output .

```
1  #!/bin/bash
2  #Loop
3  exec 3>&1 4>&2
4  trap 'exec 2>&4 1>&3' 0 1 2 3
5  exec 1>FileIO_log_Docker.output 2>&1
6  for j in {1..4}
7  do
8  for i in {1..4}
9  do
10 sysbench --num-threads=$j --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare
11 sysbench --num-threads=$j --test=fileio --file-total-size=3G --file-test-mode=rndrw run
12 sysbench --num-threads=$j --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup
13
14 done
15 done
16
```

TEST CASE 1:

For Thread =1

*sysbench --num-**threads=1** --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare*
*sysbench --num-**threads=1** --test=fileio --file-total-size=3G --file-test-mode=rndrw run*
*sysbench --num-**threads=1** --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup*

```
Number of threads: 1
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 24MiB each
3GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      10193.91
    writes/s:                     6795.91
    fsyncs/s:                     21758.75

Throughput:
    read, MiB/s:                  159.28
    written, MiB/s:               106.19

General statistics:
    total time:                          10.0002s
    total number of events:              387380

Latency (ms):
         min:                                    0.00
         avg:                                    0.03
         max:                                    5.69
         95th percentile:                        0.00
         sum:                                 9969.13

Threads fairness:
    events (avg/stddev):           387380.0000/0.00
    execution time (avg/stddev):   9.9691/0.00
```

```
Number of threads: 1
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 8MiB each
1GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      7141.04
    writes/s:                     4760.69
    fsyncs/s:                     15242.02

Throughput:
    read, MiB/s:                  111.58
    written, MiB/s:               74.39

General statistics:
    total time:                          10.0063s
    total number of events:              271498

Latency (ms):
         min:                                    0.00
         avg:                                    0.04
         max:                                    3.13
         95th percentile:                        0.10
         sum:                                 9931.79
```

Docker Container

Test run	Operation	FileSize	Threads	Throughput. - Read(MiB/Sec)	Throughput. written, MiB/s
1	r/w test	1G	1	159.28	106.19
2	r/w test	1G	1	141.64	94.42
3	r/w test	1G	1	137.83	91.89
4	r/w test	1G	1	134.22	89.48

QEMU VM

Test ru	Operatic	FileSiz	Thread	Throughput. - Read(MiB/Sec)	Throughput. written, MiB
1	r/w test	1G	1	111.58	74.39
2	r/w test	1G	1	115.98	77.32
3	r/w test	1G	1	111.49	74.33
4	r/w test	1G	1	112.7	75.14
5	r/w test	1G	1	113.16	75.44

TEST CASE 2:

For Thread =4

```
sysbench --num-threads=4 --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare
sysbench --num-threads=4 --test=fileio --file-total-size=3G --file-test-mode=rndrw run
sysbench --num-threads=4 --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup
```

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


Extra file open flags: (none)
128 files, 24MiB each
3GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      22845.65
    writes/s:                     15230.23
    fsyncs/s:                     48784.10

Throughput:
    read, MiB/s:                  356.96
    written, MiB/s:               237.97

General statistics:
    total time:                          10.0016s
    total number of events:              868257

Latency (ms):
         min:                                    0.00
         avg:                                    0.05
         max:                                    9.56
         95th percentile:                        0.00
         sum:                                39915.74

Threads fairness:
    events (avg/stddev):           217064.2500/509.32
    execution time (avg/stddev):   9.9789/0.00
```

```
Number of threads: 4
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 8MiB each
1GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      10571.50
    writes/s:                     7047.66
    fsyncs/s:                     22594.35

Throughput:
    read, MiB/s:                  165.18
    written, MiB/s:               110.12

General statistics:
    total time:                          10.0170s
    total number of events:              402327

Latency (ms):
         min:                                    0.00
         avg:                                    0.10
         max:                                   12.69
         95th percentile:                        0.26
         sum:                                39847.18
```

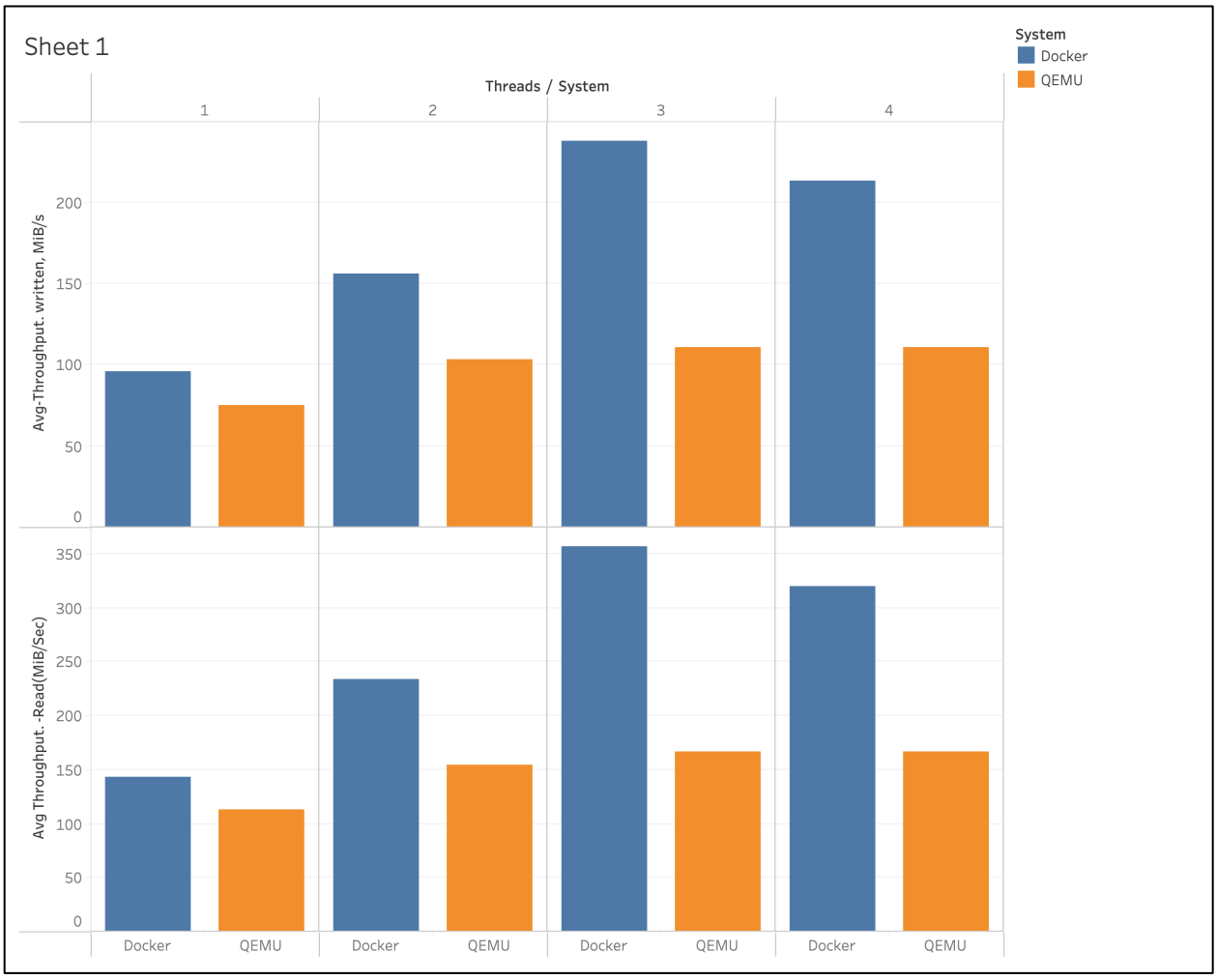
Docker container

Test run	Operation	FileSize	Threads	Throughput. - Read(MiB/Sec)	Throughput. written, MiB
1	r/w test	1G	4	166.43	110.95
2	r/w test	1G	4	166.22	110.82
3	r/w test	1G	4	166.44	110.96
4	r/w test	1G	4	166.44	110.95
5	r/w test	1G	4	165.18	110.12

QEMU VM

Test run	Operation	FileSize	Threads	Throughput. - Read(MiB/Sec)	Throughput. written, MiB/s
1	r/w test	1G	4	350.22	233.48
2	r/w test	1G	4	357.9	238.6
3	r/w test	1G	4	356.96	237.97
4	r/w test	1G	4	361.11	240.74

FILE I/O PERFORMANCE COMAPARISION B/W DOCKER AND QEMU



FILE I/O PERFORMANCE : AVERAGE THROUGHPUT

System	Threads	Avg Throughput. - Read(MiB/Sec)	Avg-Throughput. written, MiB/s
Docker	1	143.24	95.50
Docker	2	234.10	156.06
Docker	3	356.55	237.70
Docker	4	319.39	212.93
QEMU	1	112.982	75.324
QEMU	2	154.73	103.148
QEMU	3	166.338	110.892
QEMU	4	166.142	110.76

CONCLUSION:

1. The CPU performance was found to be MUCH better in Docker.
2. Avg CPU speed decrease with increase in Prime max limit. With Increase in threads performance increases.
3. FILE IO performance for docker is better than QEMU and with increase in thread performance improves
4. Average read and Written throughput increases with threads for both docker and QEMU.