COEN 241: Cloud Computing System vs OS Virtualization

## Xiao He (W1649820)

Table of contents

1. [System Configuration 2](#_TOC_250008)
2. [Steps to enable QEMU 3](#_TOC_250007)
3. [Steps to enable Docker container 4](#_TOC_250006)
4. [Proof of experiment 6](#_TOC_250005)
5. [Performance measurement 12](#_TOC_250004)
6. [Performance Analysis 16](#_TOC_250003)
7. [Shell scripts for test execution 17](#_TOC_250002)
8. [CPU Utilization 18](#_TOC_250001)
9. [Dockerfile for VM 20](#_TOC_250000)

# System configuration

The experiments for the virtual QEMU machine and the docker container have been run on Mac M2 with Apple Silicon Chip with 8GB RAM.

Graphical user interface, text, application

Description automatically generated

* 1. QEMU

The ubuntu-server iso has been mounted on a QEMU image having 10G hard disk space and 2GB RAM and 1 CPU.

* 1. Docker container

The docker container is created with similar specifications so as to ensure that the test environment is as similar as possible. The memory limit on the container is 2G.

# Steps to enable QEMU

As I am using MAC with an M2 chip, I installed QEMU using homebrew. After installing QEMU I created an image with 10GB of hard disk space. I downloaded the ubuntu server iso and then installed the by loading it as a cdrom. Below are the commands for these steps:

* brew install qemu
* qemu-img create -f qcow2 disk.qcow2 10G
* qemu-system-aarch64 -accel hvf -m 2048 -cpu cortex-a57 \

-M virt,highmem=off -drive file=/opt/homebrew/Cellar/qemu/7.1/share/qemu/edk2-aarch64- code.fd,if=pflash,format=raw,readonly=on \

-drive file=ovmf\_vars.fd,if=pflash,format=raw serial telnet::4444,server,nowait \

-drive if=none,file=disk.qcow2,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

/Users/janangandhi/Documents/Courses/CC/ubuntu-20.04.4-live-server-ar m64.iso \

-device usb-ehci -device usb-kbd -device usb-mouse -usb \

-monitor stdio

Following are the important parameters in the above QEMU command:

1. -m - It is used to assign RAM to the virtual machine. I have assigned a RAM of 2G to the machine.
2. -drive - It is used to attach the qemu-image we created in the previous step.
3. -cdrom - It is used to provide the ubuntu server iso image as a cdrom for ubuntu installation.
4. -accel - It is used to enable an accelerator
5. -cpu - It is used to select amongst different cpu models available

# Steps to enable Docker container

As I was running into issues while trying to install native Docker onto my Apple Silicon MAC, I had to install Docker Desktop for Apple Silicon from the official Docker website - <https://docs.docker.com/desktop/mac/apple-silicon/>. After installation, I pulled the zyclonite/sysbench image using the docker pull command. It stored a copy of the image from the docker hub to my local. I was able to check this image using the below command:Text

Description automatically generated

It displays all the copies of the images that are present on the local machine. Post that I started an instance of that image using the command:

docker run --rm -it --memory="2g" --entrypoint /bin/sh zyclonite/sysbench

Graphical user interface, application, table

Description automatically generated

The command launches an instance of the image using the memory limit of 2GB in interactive mode. I was then able to perform sysbench tests on this instance. Other important docker commands include:

* docker ps - It is used to see all containers that are currently running. You can all the - - all parameter to see all exited containers as well.
* docker rm - it is used to remove/clean up containers.
* docker network ls - to list all the networks created for the docker containers to allow communication between different containers.

# Proof of experiment

I have executed three tests each for CPU and File IO on both the docker container and QEMU. Each test was executed 5 times to get the average results. The tests are as follows:

#### CPU tests

Test 1: sysbench --test=cpu --cpu-max-prime=2000 run QEMU

Text

Description automatically generated

Docker

Text

Description automatically generated

Test 2: sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run QEMU

Text

Description automatically generated

Docker

Text, table

Description automatically generated

Test 3: sysbench --num-threads=8 --test=cpu --cpu-max-prime=10000 run QEMUText

Description automatically generated

Docker

Text

Description automatically generated with medium confidence

#### File IO tests

Test 1:

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

Text

Description automatically generated

Docker

Table

Description automatically generated

Test 2:

sysbench --num-threads=8 --test=fileio --file-total-size=1G --file-test-mode=seqrd run QEMU

Text

Description automatically generated

Docker

Table

Description automatically generated with medium confidence

Test 3:

sysbench --num-threads=8 --test=fileio --file-total-size=1G --file-test-mode=seqwr run QEMUText

Description automatically generated

Docker

Table

Description automatically generated with medium confidence

# Performance measurement

I have used **Events per second** as the standard unit of measurement across all the below tests. The tests for CPU are designed to monitor system performance with varied CPU max prime, the number of threads as well as the max time. The parameters for all CPU tests are as follows:

#### CPU tests

|  |  |  |  |
| --- | --- | --- | --- |
|  | CPU max prime | Number of Threads | Max Time |
| Test 1 | 2000 | 1 | 0 |
| Test 2 | 10000000 | 1 | 30 |
| Test 3 | 10000 | 8 | 0 |

The result of each test rounded off to two decimal places with 5 runs in both Docker and QEMU are as follows:

**Test 1 results**: sysbench --test=cpu --cpu-max-prime=2000 run

|  |  |  |
| --- | --- | --- |
|  | QEMU | Docker |
| Average | 112974.25 | 55210.10 |
| Min | 111913.93 | 53268.05 |
| Max | 115140.45 | 56219.67 |
| Std | 1264.34 | 1167.08 |

**Test 2 results**: sysbench --test=cpu --cpu-max-prime=10000000 --max-time=30 run

|  |  |  |
| --- | --- | --- |
|  | QEMU | Docker |
| Average | 0.842 | 0.85 |
| Min | 0.81 | 0.82 |
| Max | 0.85 | 0.87 |
| Std | 0.01 | 0.01 |

**Test 3 results**: sysbench --num-threads=8 --test=cpu --cpu-max-prime=10000 run

|  |  |  |
| --- | --- | --- |
|  | QEMU | Docker |
| Average | 10634.71 | 27152.73 |
| Min | 10499.60 | 26922.64 |
| Max | 10736.79 | 27532.89 |
| Std | 95.65 | 237.50 |

#### File IO tests

The tests for file IO are designed to monitor the system performance with varied file size, number of threads and modes. The parameters of these tests are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | File size | Number of Threads | Mode |
| Test 1 | 3G | 16 | rndrw |
| Test 2 | 2G | 8 | seqrd |
| Test 3 | 2G | 8 | seqwr |

For each File IO test we need to ensure that the system does not go to cache for the file as that would taint the test results. For each test we run three stages : prepare, run and cleanup. The result of each test rounded off to two decimal places with 5 runs in both Docker and QEMU are as follows:

**Test 1 results**: sysbench --num-threads=16 --test=fileio --file-total-size=3G

--file-test-mode=rndrw

|  |  |  |
| --- | --- | --- |
|  | QEMU | Docker |
| Average | 48839.02 | 34194.45 |
| Min | 40756.71 | 20056.79 |
| Max | 52989.26 | 43222.96 |
| Std | 4784.59 | 8548.62 |

**Test 2 results**: sysbench --num-threads=8 --test=fileio --file-total-size=2G

--file-test-mode=seqrd

|  |  |  |
| --- | --- | --- |
|  | QEMU | Docker |
| Average | 41347.68 | 38156.18 |
| Min | 35455.98 | 12355.28 |
| Max | 48135.42 | 60261.24 |
| Std | 4635.56 | 21426.84 |

**Test 3 results**: sysbench --num-threads=8 --test=fileio --file-total-size=2G

--file-test-mode=seqwr

|  |  |  |
| --- | --- | --- |
|  | QEMU | Docker |
| Average | 52914.12 | 36616.48 |
| Min | 48114.52 | 29624.89 |
| Max | 55605.25 | 39888.78 |
| Std | 3495.21 | 4115.24 |

# Performance Analysis

We can infer the below things from the results of all tests that we have collected:

1. **QEMU is faster than Docker desktop on Mac M2 for CPU tests -** Containers are expected to have better performance than Virtual Machines because they only provide OS virtualization and don’t interact directly with the hardware. However, because the test machine is a Mac M1 the docker performance is relatively worse than QEMU for all CPU tests. From the results we can see that the events per seconds for docker are much less that the events per seconds in QEMU.
2. **Increasing the number of threads in the test helps to increase performance of the system -** For all the tests where the system had more more threads, the performance of the system is much better. However, just increasing the number of threads does not help. The performance also depends on how many cores are present in the machine.
3. **Higher the cpu max prime, lower is the events per seconds -** The number of events that the system performs is directly related to the complexity of the event. From the tests we can see that for higher max prime parameters, the events per seconds fall down drastically since it is more complicated to calculate.
4. **For file tests sequential writes perform much better than random writes -** As observed in the file IO test results, the system performs better for sequential writes. This is because the seek time for sequential write and read is reduced as compared to random read and writes where the head of the disk has to rotate more.

# Shell scripts for test execution

### cpu-test.sh

**Text

Description automatically generated**

### file-test.sh

Text

Description automatically generated

# CPU Utilization

CPU usage has been measured on host OS using the activity monitor. Below is the CPU usage on the host machine:

When sysbench is not running on docker:

Table

Description automatically generated

When cpu sysbench test is running on the docker:

Table

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | Without sysbench | With sysbench |
| User level CPU usage% | 8.46% | 8.51% |
| System(kernel) level CPU usage% | 8.43% | 21.22% |
| Idle % | 81.49% | 72.4% |

We can see from the above table that when we run sysbench, the kernel level CPU usage jumps from 8.43% to 21.22%. This is because the container is using the kernel of the host machine. Also, if we see, the usage of CPU by the qemu process which is used by docker to run the container increases to 105%.

# Dockerfile for VM

Text

Description automatically generated