HW1 Report

Name: Jichen Dou
Student ID: W1628888

Host Environment Configurations

CPU: Apple M1
Memory: 8GB
OS: macOS Big Sur
Model: Macbook Pro

Steps to enable QEMU VM

1. Install qemu

Brew install qemu

2. QEMU installation on M1

(1) Install Xcode command line tools:
Xcode-select –install

(2) Configure the environment variables in .bash_profile and add brew path in .bash_profile. Installing necessary packages for building:

```
(base) dudu@dududeMacBook-Pro ~ % vim ~/.bash_profile
(base) dudu@dududeMacBook-Pro ~ % brew install libffi gettext glib pkg-config autoconf auto
make pixman ninja
```

(3) Install qemu:
Sudo brew install qemu
I tried many ways to install qemu on my computer. And I found that when I want to create a virtual machine on qemu, I had to use qemu version above 6.2.0. And these versions can supply hvf accel.

(4) Configure ubuntu server on QEMU and start:

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

And rename ubuntu-20.04.5-live-server-arm64.iso as ubuntu-lts.iso.

```
(base) dudu@dududeMacBook-Pro build % qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048
-smp 4 \
-drive
file=/usr/local/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-lts.iso \
-device usb-ehci -device usb-kbd -device usb-mouse -usb \
-nographic
```

Enable a Docker Container

I installed Docker Desktop. I utilized zyclonite/sysbench as docker image. Some important operations are:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

```
$ docker rm 305297d7a235 ff0a5c3750b9
305297d7a235
ff0a5c3750b9
```

--cpuset-cpus specify number of CPU when run

--memory specify number of memory when run

Experiments

Three different scenarios for each virtualization tech

1. Case 1: cpu-max-prime = 10000
2. Case 2: cpu-max-prime = 20000
3. Case 3: cpu-max-prime = 80000

For CPU test:

1. sysbench --test=cpu --cpu-max-prime=10000 run
2. sysbench --test=cpu --cpu-max-prime=20000 run
3. sysbench --test=cpu --cpu-max-prime=80000 run

For fileIO test:

For this test, I want to use different thread to compare different tech.

1. one thread:

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

2. Four threads:

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

For these experiments, I want to use above shell commands and run them on each virtualization technologies. Then I will compare their running time and their latency to get a result which one is better.

CPU Test Performance

Case 1

QEMU:

Test run	Prime num	Total time
1	10000	10.0000s
2	10000	10.0000s
3	10000	10.0000s
4	10000	10.0000s
5	10000	10.0001s
Min	Max	Avg
10.0000s	10.0001s	10.0000s

Docker:

Test run	Prime num	Total time
1	10000	10.0002s
2	10000	10.0001s
3	10000	10.0001s
4	10000	10.0002s
5	10000	10.0002s
Min	Max	Avg
10.0001s	10.0002s	10.0001s

Case 2

QEMU:

Test run	Prime num	Total time
1	20000	10.002s
2	20000	10.0008s
3	20000	10.0028s
4	20000	10.002s
5	20000	10.0008s
Min	Max	Avg
10.0008s	10.0028s	10.0016s

Docker:

Test run	Prime num	Total time
1	20000	10.0004s
2	20000	10.0002s
3	20000	10.0002s
4	20000	10.0004s
5	20000	10.0004s
Min	Max	Avg
10.0002s	10.0004s	10.0003s

Case 3

QEMU:

Test run	Prime num	Total time
1	80000	10.0078s
2	80000	10.0126s
3	80000	10.0035s
4	80000	10.004s
5	80000	10.0147s
Min	Max	Avg
10.004s	10.0147s	10.0085s

Docker:

Test run	Prime num	Total time
1	80000	10.0007s
2	80000	10.0001s
3	80000	10.0011s
4	80000	10.0014s
5	80000	10.0010s
Min	Max	Avg
10.0001s	10.0014s	10.0008s

FileIO Test Performance

Case1

QEMU:

Test run	Thread	Write	Read	avg
1	1	74.39	111.58	0.03
2	1	75.32	115.98	0.04
3	1	78.55	112.47	0.04
4	1	75.14	112.8	0.04
5	1	75.44	113.69	0.03

Docker:

Test run	Thread	Write	Read	avg
1	1	66.64	99.96	0.04
2	1	80.08	120.13	0.03
3	1	89.21	133.81	0.04
4	1	84.97	127.45	0.03
5	1	80.30	120.45	0.03

Case2

QEMU:

Test run	Thread	Write	Read	avg
1	4	233.48	350.22	0.03
2	4	238.6	357.9	0.05
3	4	237.97	356.96	0.04
4	4	240.74	356.96	0.04
5	4	236.88	354.29	0.03

Docker:

Test run	Thread	Write	Read	avg
1	4	202.80	304.20	0.05
2	4	272.93	409.40	0.05
3	4	338.37	507.56	0.03
4	4	458.46	687.69	0.02
5	4	476.97	715.46	0.02

Analysis

1. CPU performance of Docker is much better than QEMU.
2. FileIO performance of Docker is much better than QEMU.
3. For FileIO, Increasing the number of threads can improve the FileIO performance.
4. There is a small difference between test cases for CPU.
5. And for docker, I think that if we did not change our test data files, Docker can save some their previous files. It means that as our test cases ran, the performance improved better.