Zhihao Lin

HW1 Report

Host Environment Configurations:
CPU: 2.6 GHz 6-Core Intel Core i7
Memory: 16 GB
OS: macOS Monterey
Model: MacBook Pro

1) Steps to enable QEMU VM

Enter following commands in terminal.

1. Install qemu
   ```
   $ brew install qemu
   ```

2. Create an QEMU image with 10G memory
   ```
   $ qemu-img create ubuntu.img 10G -f qcow2
   ```

3. Install VM
   ```
   $ qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom ./ubuntu-20.04.4-live-server-amd64.iso -m 2046
   -boot strict=on
   ```

4. Run VM (1 CPU & 2G Memory)
   ```
   $ qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -boot strict=on
   ```

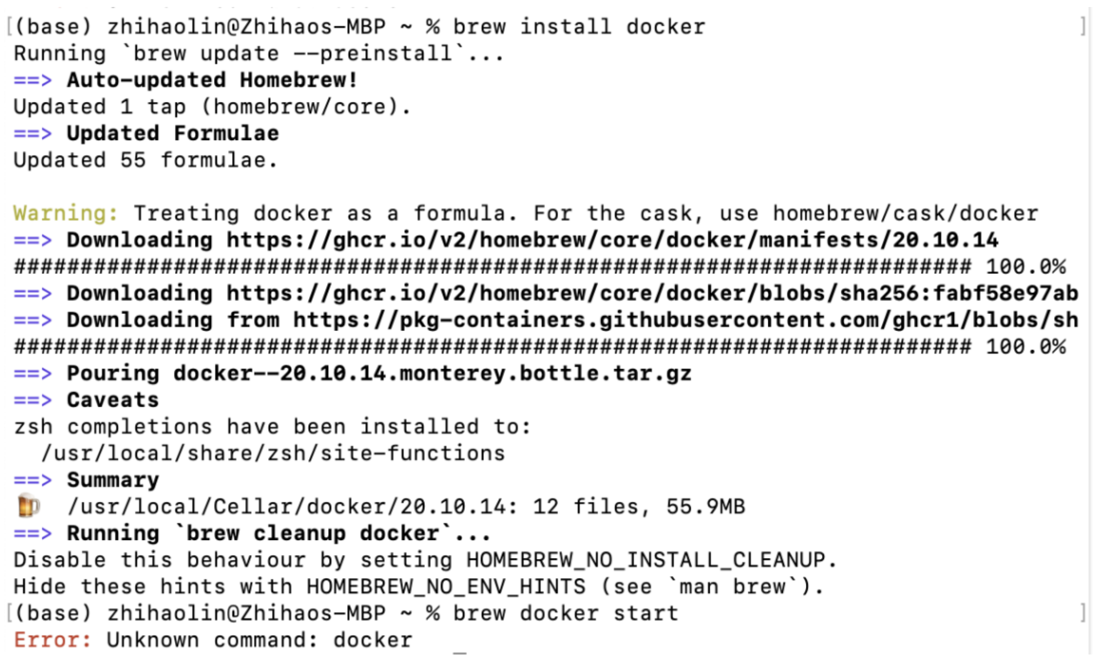
Additional config when run VM:

a. VM with 2 CPU & 2G Memory & accel:
   ```
   $ qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -smp 2 --accel tcg -boot strict=on
   ```

b. VM with 2 CPU & 4G Memory & accel:
   ```
   $ qemu-system-x86_64 -hda ubuntu.img -boot d -m 4G -smp 2 --accel tcg -boot strict=on
   ```

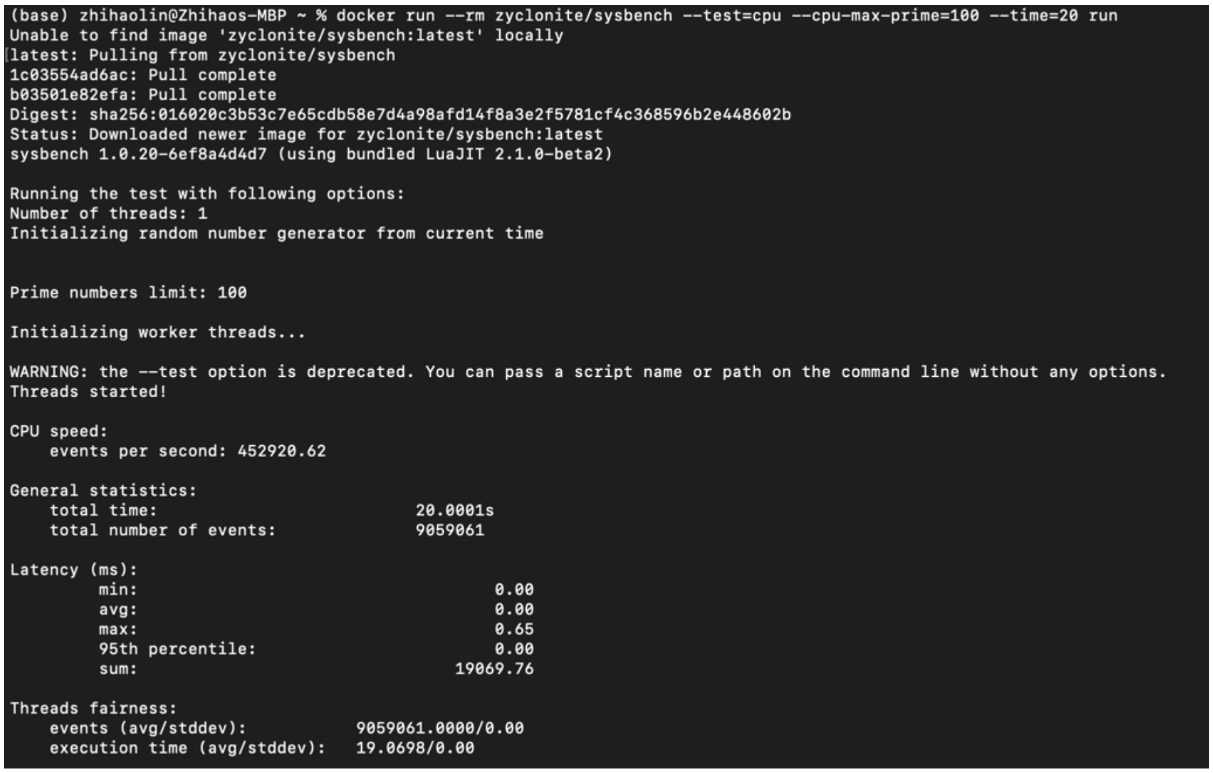
2) Steps to enable a Docker container

First, I tried to install Docker native using brew, but it didn't work when I tried to run as following screenshot. It also stated on Piazza that I need Docker-Machine + VirtualBox to run on mac. So I installed Docker Desktop for Mac environment.



```
[(base) zhihaolin@Zhihaos-MBP ~ % brew install docker                    ]
Running `brew update --preinstall`...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> Updated Formulae
Updated 55 formulae.

Warning: Treating docker as a formula. For the cask, use homebrew/cask/docker
==> Downloading https://ghcr.io/v2/homebrew/core/docker/manifests/20.10.14
######################################################################## 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/docker/blobs/sha256:fabf58e97ab
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sh
######################################################################## 100.0%
==> Pouring docker--20.10.14.monterey.bottle.tar.gz
==> Caveats
zsh completions have been installed to:
  /usr/local/share/zsh/site-functions
==> Summary
🍺  /usr/local/Cellar/docker/20.10.14: 12 files, 55.9MB
==> Running `brew cleanup docker`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
[(base) zhihaolin@Zhihaos-MBP ~ % brew docker start                     ]
Error: Unknown command: docker
```

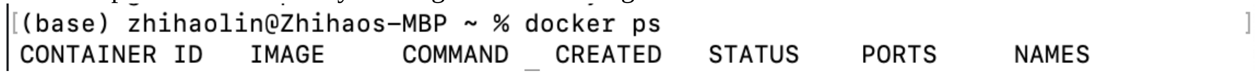
After I have installed Docker Desktop, I ran "$ docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=100 --time=20 run" which installed the sysbench first if it had not been installed. Then it executed the CPU test. Shown as following screen shot:



```
(base) zhihaolin@Zhihaos-MBP ~ % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=100 --time=20 run
Unable to find image 'zyclonite/sysbench:latest' locally
[latest: Pulling from zyclonite/sysbench
1c03554ad6ac: Pull complete
b03501e82efa: Pull complete
Digest: sha256:016020c3b53c7e65cdb58e7d4a98afd14f8a3e2f5781cf4c368596b2e448602b
Status: Downloaded newer image for zyclonite/sysbench:latest
sysbench 1.0.20-6ef8a4d4d7 (using bundled LuaJIT 2.1.0-beta2)

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


Prime numbers limit: 100

Initializing worker threads...

WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
Threads started!

CPU speed:
    events per second: 452920.62

General statistics:
    total time:                          20.0001s
    total number of events:              9059061

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.65
         95th percentile:                        0.00
         sum:                                19069.76

Threads fairness:
    events (avg/stddev):           9059061.0000/0.00
    execution time (avg/stddev):   19.0698/0.00
```

Some important operations for docker are:

1. $ docker ps #Check currently running containers and get their information:

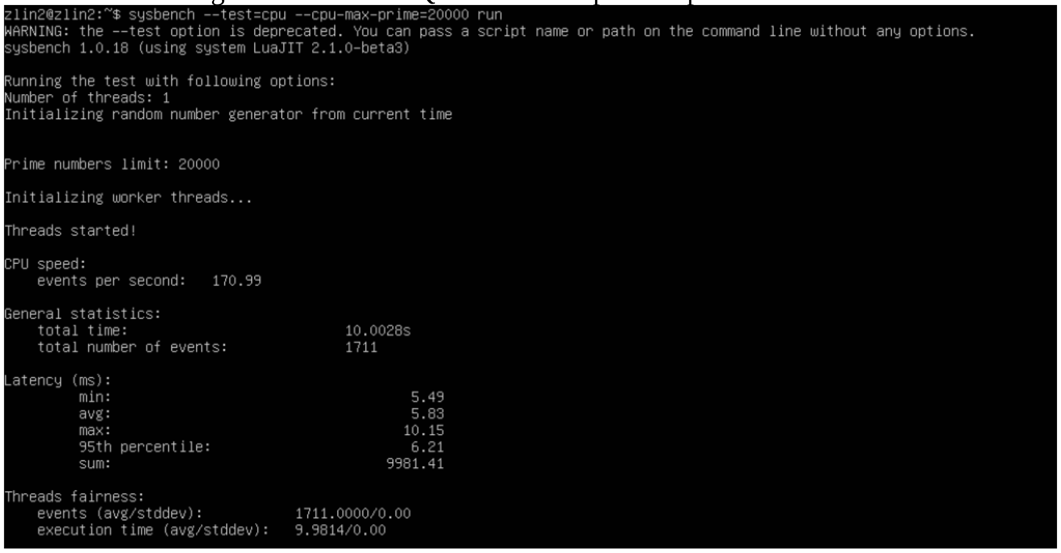


```
(base) zhihaolin@Zhihaos-MBP ~ % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

2. $ docker rm #Delete container
3. Transfer file from local to container for test automation:
   $ sudo docker cp <file on local> <container id>:<file location at container>
4. --cpuset-cpus flags to specify number of CPU when run
5. --memory flags to specify number of memory when run

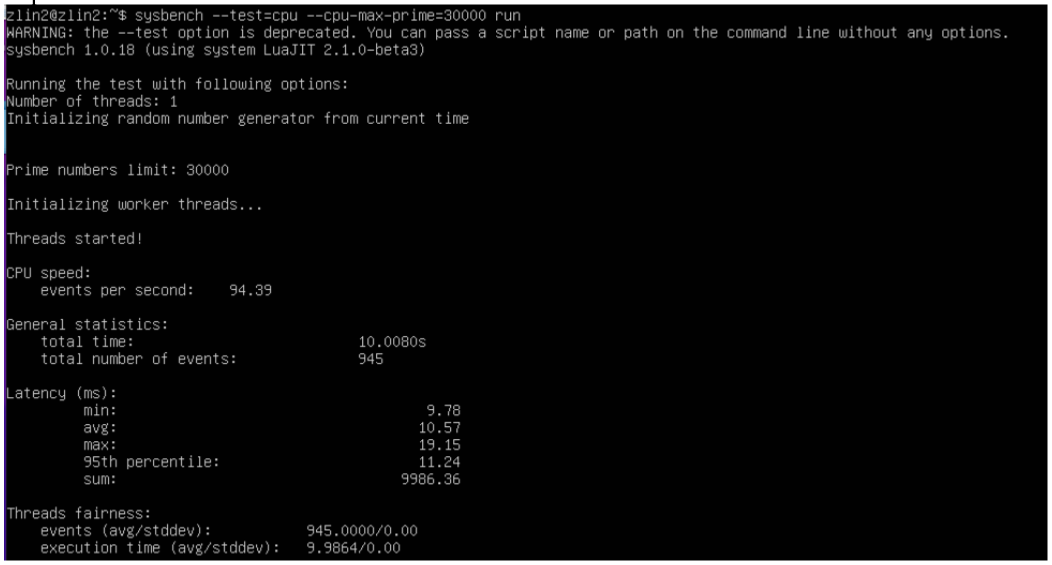
3) Proof of experiment

First I tried running the CPU test on QEMU with --cpu-max-prime=20000 the result is shown as following:



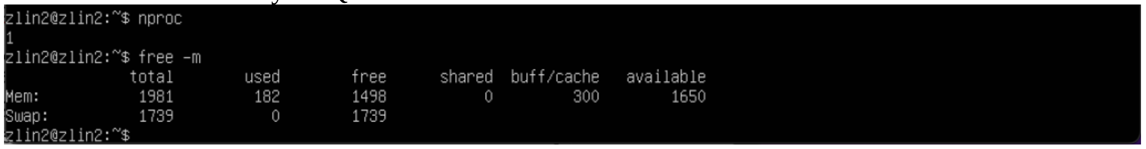
```
zlin2@zlin2:~$ sysbench --test=cpu --cpu-max-prime=20000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

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:   170.99

General statistics:
    total time:                          10.0028s
    total number of events:              1711

Latency (ms):
         min:                                    5.49
         avg:                                    5.83
         max:                                   10.15
         95th percentile:                        6.21
         sum:                                 9981.41

Threads fairness:
    events (avg/stddev):           1711.0000/0.00
    execution time (avg/stddev):   9.9814/0.00
```

Then I tried with --cpu-max-prime=30000, which produced the same total time=10s. I also tried --cpu-max-prime=50000 which also had total time=10s. So I used -time=8 flag and events per second for all my CPU experiments.



```
zlin2@zlin2:~$ sysbench --test=cpu --cpu-max-prime=30000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
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: 30000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:    94.39

General statistics:
    total time:                          10.0080s
    total number of events:              945

Latency (ms):
         min:                                    9.78
         avg:                                   10.57
         max:                                   19.15
         95th percentile:                       11.24
         sum:                                 9986.36

Threads fairness:
    events (avg/stddev):           945.0000/0.00
    execution time (avg/stddev):   9.9864/0.00
```

Check CPU and memory on QEMU:

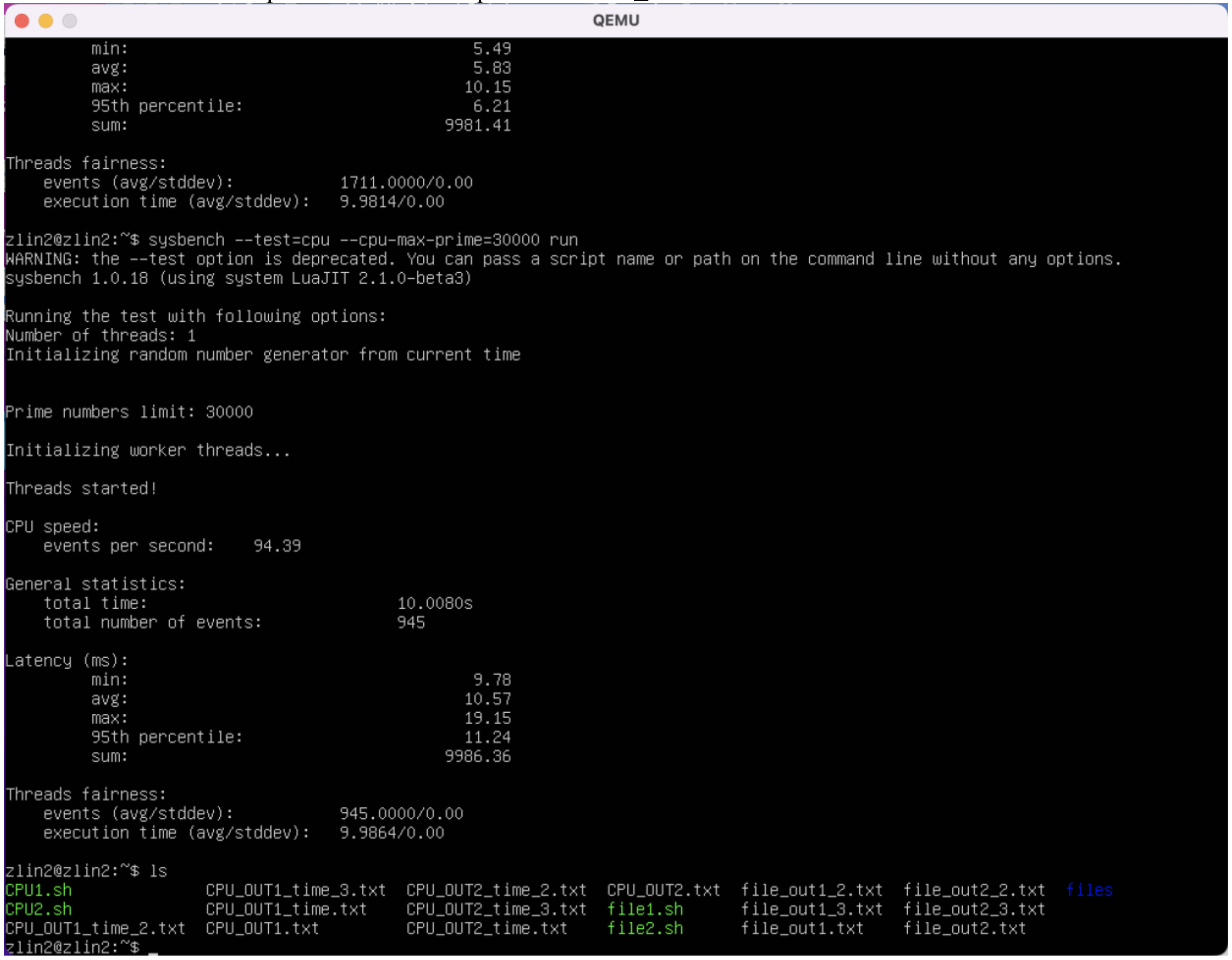


```
zlin2@zlin2:~$ nproc
1
zlin2@zlin2:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           1981         182        1498           0         300        1650
Swap:          1739           0        1739
zlin2@zlin2:~$
```

3.1) How to collect performance data

I used sysbench to get some user-level information for CPU test, and to get latency and I/O throughput of fileio test.

All the experiments were automated by using shell scripts and the sysbench output were stored into .txt files for better access as shown in following screenshot. Example command: $ ./CPU1.sh > CPU_OUT1.txt which will run "CPU1.sh" script and store the output into "CPU_OUT1.txt" file



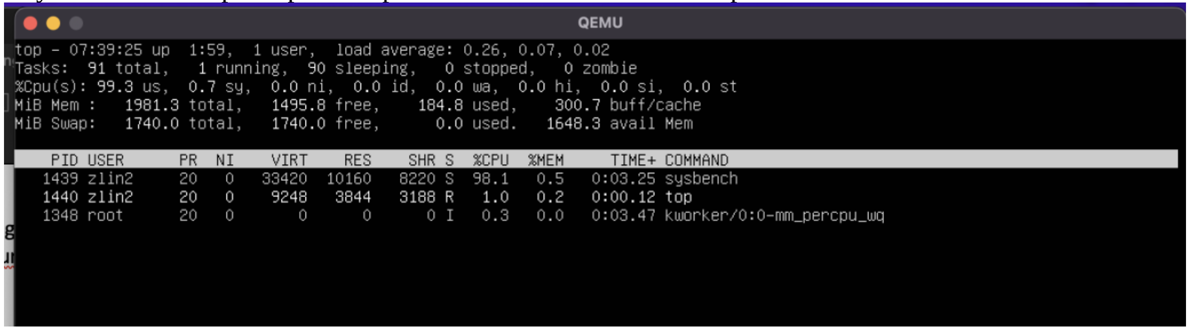
QEMU

```
         min:                                    5.49
         avg:                                    5.83
         max:                                   10.15
         95th percentile:                        6.21
         sum:                                 9981.41

Threads fairness:
    events (avg/stddev):           1711.0000/0.00
    execution time (avg/stddev):   9.9814/0.00

zlin2@zlin2:~$ sysbench --test=cpu --cpu-max-prime=30000 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
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: 30000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:    94.39

General statistics:
    total time:                          10.0080s
    total number of events:              945

Latency (ms):
         min:                                    9.78
         avg:                                   10.57
         max:                                   19.15
         95th percentile:                       11.24
         sum:                                 9986.36

Threads fairness:
    events (avg/stddev):           945.0000/0.00
    execution time (avg/stddev):   9.9864/0.00

zlin2@zlin2:~$ ls
CPU1.sh                CPU_OUT1_time_3.txt  CPU_OUT2_time_2.txt  CPU_OUT2.txt  file_out1_2.txt  file_out2_2.txt  files
CPU2.sh                CPU_OUT1_time.txt    CPU_OUT2_time_3.txt  file1.sh      file_out1_3.txt  file_out2_3.txt
CPU_OUT1_time_2.txt    CPU_OUT1.txt         CPU_OUT2_time.txt    file2.sh      file_out1.txt    file_out2.txt
zlin2@zlin2:~$ _
```

All output files are uploaded inside "output" directory on GitHub.

For CPU utilization and I/O disk utilization, I used “top” commands to access the information:

The following is a screenshot of CPU utilization during the test on QEMU with CPU=1 & Memory=2G.
The command I ran:
$ sysbench --test=cpu --cpu-max-prime=30000 -time=8 run & top -i



```
top - 07:39:25 up  1:59,  1 user,  load average: 0.26, 0.07, 0.02
Tasks:  91 total,   1 running,  90 sleeping,   0 stopped,   0 zombie
%Cpu(s): 99.3 us,  0.7 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1981.3 total,   1495.8 free,    184.8 used,    300.7 buff/cache
MiB Swap:   1740.0 total,   1740.0 free,      0.0 used.   1648.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1439 zlin2     20   0   33420  10160   8220 S  98.1   0.5   0:03.25 sysbench
   1440 zlin2     20   0    9248   3844   3188 R   1.0   0.2   0:00.12 top
   1348 root      20   0       0      0      0 I   0.3   0.0   0:03.47 kworker/0:0-mm_percpu_wq
```

4) Three different scenarios for each virtualization technology

I conducted three scenarios by specifying different number of CPU and memory. The three scenarios are listed below:

1. Scenario 1: CPU=1 & Memory=2G
2. Scenario 2: CPU=2 & Memory=2G
3. Scenario 3: CPU=2 & Memory=4G

Detail commands about how I specifying those scenarios when running the experiments.

QEMU:
-smp specify number of CPU
-m specify number of Memory

1. qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -boot strict=on
2. qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -smp 2 --accel tcg -boot strict=on
3. qemu-system-x86_64 -hda ubuntu.img -boot d -m 4G -smp 2 --accel tcg -boot strict=on

Docker:
--cpuset-cpus specify number of CPU
--memory or -m specify number of memory

For CPU test:
Prime=20000
1. docker run --rm -m="2g" --cpuset-cpus="0" zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=8 run
2. docker run --rm -m="2g" --cpuset-cpus="0-1" zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=8 run
3. docker run --rm -m="4g" --cpuset-cpus="0-1" zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=8 run

Prime=30000
1. docker run --rm -m="2g" --cpuset-cpus="0" zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=8 run
2. docker run --rm -m="2g" --cpuset-cpus="0-1" zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=8 run
3. docker run --rm -m="4g" --cpuset-cpus="0-1" zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=8 run

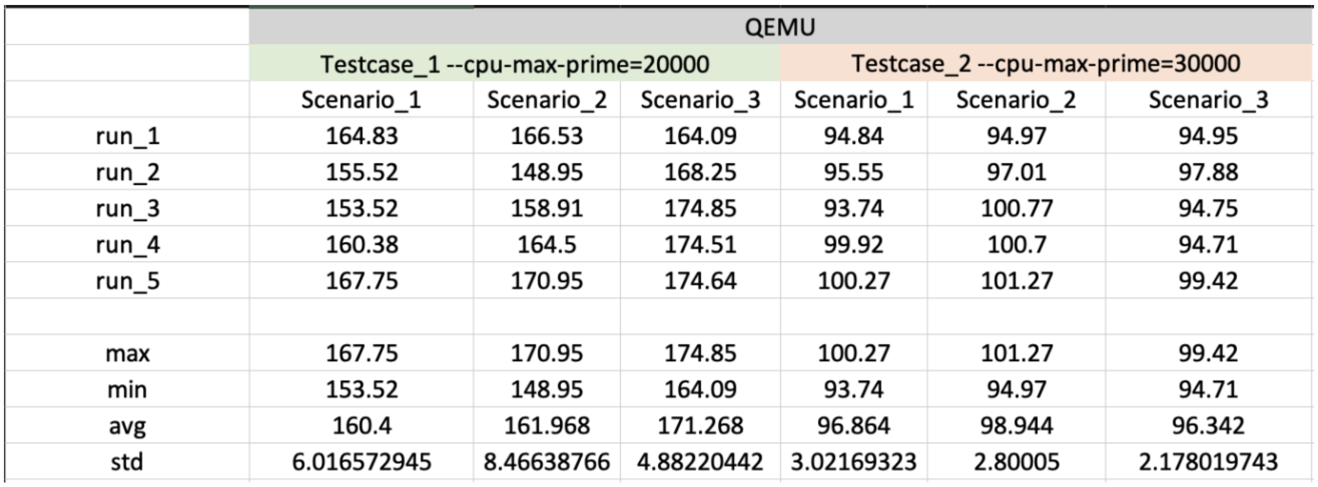
For fileIO test:
1. Docker run --rm -it -m="2g" -- cpuset-cpus="0" --entrypoint /bin/sh zyclonite/sysbench
2. Docker run --rm -it -m="2g" -- cpuset-cpus="0-1" --entrypoint /bin/sh zyclonite/sysbench
3. Docker run --rm -it -m="4g" -- cpuset-cpus="0-1" --entrypoint /bin/sh zyclonite/sysbench

5) Presentation and analysis of the performance data

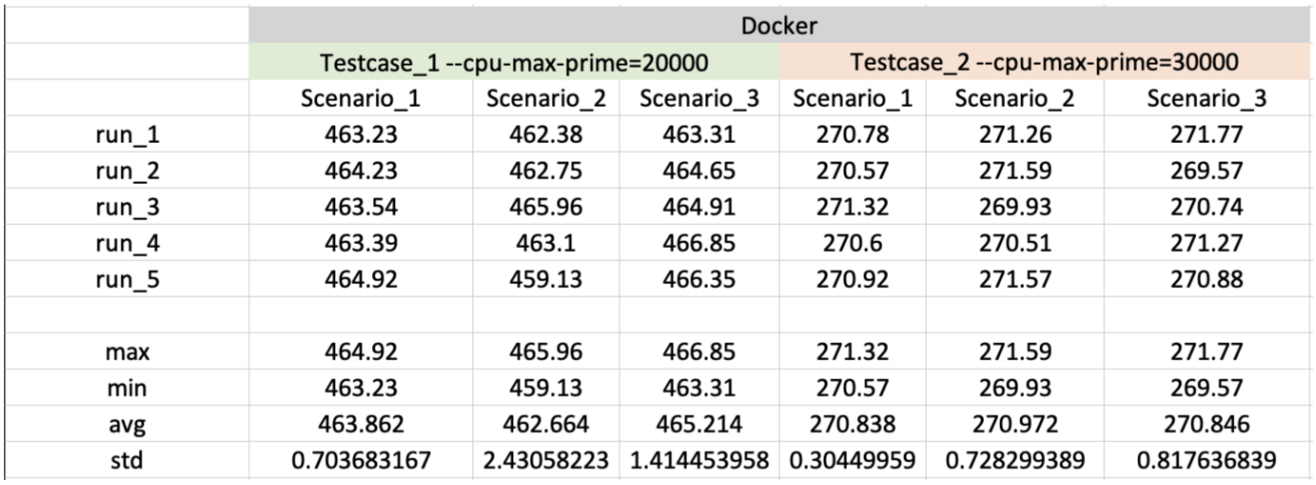
All data are extracted using the average, you can see details with min, max and std in the excel file.

The data is present in the order of three system scenarios:
- Scenario 1: CPU=1 & Memory=2G
- Scenario 2: CPU=2 & Memory=2G
- Scenario 3: CPU=2 & Memory=4G

CPU tests: (unit: events/second)

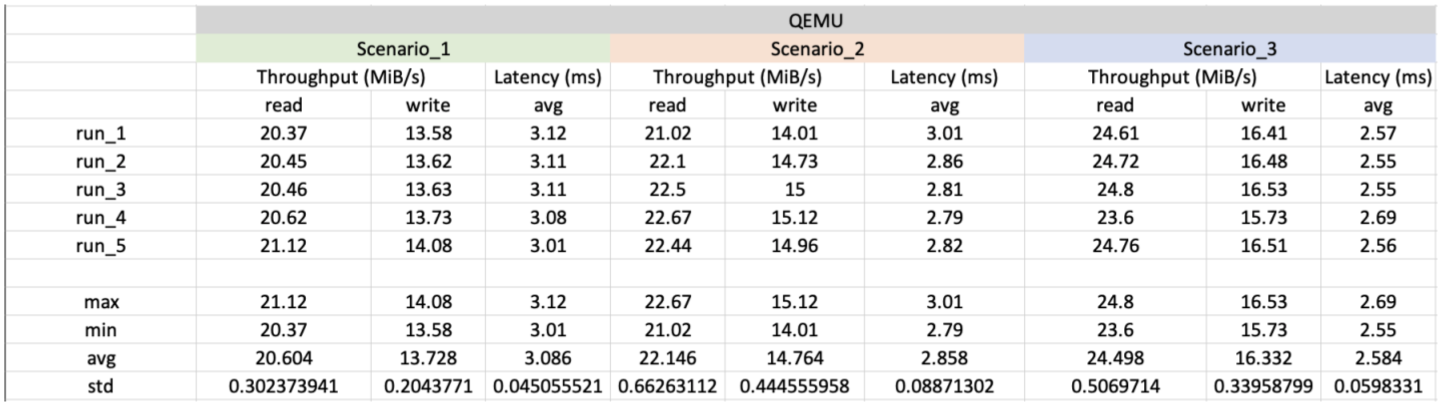


	QEMU					
	Testcase_1 --cpu-max-prime=20000			Testcase_2 --cpu-max-prime=30000		
	Scenario_1	Scenario_2	Scenario_3	Scenario_1	Scenario_2	Scenario_3
run_1	164.83	166.53	164.09	94.84	94.97	94.95
run_2	155.52	148.95	168.25	95.55	97.01	97.88
run_3	153.52	158.91	174.85	93.74	100.77	94.75
run_4	160.38	164.5	174.51	99.92	100.7	94.71
run_5	167.75	170.95	174.64	100.27	101.27	99.42
max	167.75	170.95	174.85	100.27	101.27	99.42
min	153.52	148.95	164.09	93.74	94.97	94.71
avg	160.4	161.968	171.268	96.864	98.944	96.342
std	6.016572945	8.46638766	4.88220442	3.02169323	2.80005	2.178019743

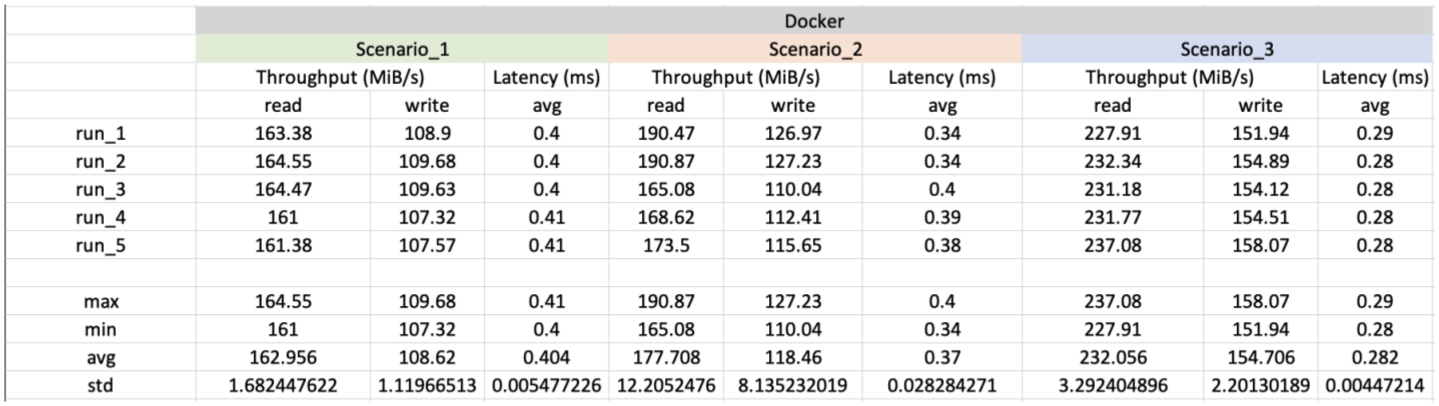


	Docker					
	Testcase_1 --cpu-max-prime=20000			Testcase_2 --cpu-max-prime=30000		
	Scenario_1	Scenario_2	Scenario_3	Scenario_1	Scenario_2	Scenario_3
run_1	463.23	462.38	463.31	270.78	271.26	271.77
run_2	464.23	462.75	464.65	270.57	271.59	269.57
run_3	463.54	465.96	464.91	271.32	269.93	270.74
run_4	463.39	463.1	466.85	270.6	270.51	271.27
run_5	464.92	459.13	466.35	270.92	271.57	270.88
max	464.92	465.96	466.85	271.32	271.59	271.77
min	463.23	459.13	463.31	270.57	269.93	269.57
avg	463.862	462.664	465.214	270.838	270.972	270.846
std	0.703683167	2.43058223	1.414453958	0.30449959	0.728299389	0.817636839

fileio tests:



	QEMU								
	Scenario_1			Scenario_2			Scenario_3		
	Throughput (MiB/s)		Latency (ms)	Throughput (MiB/s)		Latency (ms)	Throughput (MiB/s)		Latency (ms)
	read	write	avg	read	write	avg	read	write	avg
run_1	20.37	13.58	3.12	21.02	14.01	3.01	24.61	16.41	2.57
run_2	20.45	13.62	3.11	22.1	14.73	2.86	24.72	16.48	2.55
run_3	20.46	13.63	3.11	22.5	15	2.81	24.8	16.53	2.55
run_4	20.62	13.73	3.08	22.67	15.12	2.79	23.6	15.73	2.69
run_5	21.12	14.08	3.01	22.44	14.96	2.82	24.76	16.51	2.56
max	21.12	14.08	3.12	22.67	15.12	3.01	24.8	16.53	2.69
min	20.37	13.58	3.01	21.02	14.01	2.79	23.6	15.73	2.55
avg	20.604	13.728	3.086	22.146	14.764	2.858	24.498	16.332	2.584
std	0.302373941	0.2043771	0.045055521	0.66263112	0.444555958	0.08871302	0.5069714	0.33958799	0.0598331



	Docker								
	Scenario_1			Scenario_2			Scenario_3		
	Throughput (MiB/s)		Latency (ms)	Throughput (MiB/s)		Latency (ms)	Throughput (MiB/s)		Latency (ms)
	read	write	avg	read	write	avg	read	write	avg
run_1	163.38	108.9	0.4	190.47	126.97	0.34	227.91	151.94	0.29
run_2	164.55	109.68	0.4	190.87	127.23	0.34	232.34	154.89	0.28
run_3	164.47	109.63	0.4	165.08	110.04	0.4	231.18	154.12	0.28
run_4	161	107.32	0.41	168.62	112.41	0.39	231.77	154.51	0.28
run_5	161.38	107.57	0.41	173.5	115.65	0.38	237.08	158.07	0.28
max	164.55	109.68	0.41	190.87	127.23	0.4	237.08	158.07	0.29
min	161	107.32	0.4	165.08	110.04	0.34	227.91	151.94	0.28
avg	162.956	108.62	0.404	177.708	118.46	0.37	232.056	154.706	0.282
std	1.682447622	1.11966513	0.005477226	12.2052476	8.135232019	0.028284271	3.292404896	2.20130189	0.00447214

Analysis:

1. CPU performance of Docker is much better than QEMU
2. fileio performance of Docker is much better than QEMU
3. Increase the cpu-max-prime will decrease the CPU performance (events per second) on both Docker and QEMU
4. Increase the number of CPU for the environment shows a lower percentage of CPU utilization
5. Increase the number of CPU and memory can increase the fileio performance, with a lower average latency and higher throughput.