HW#1 Report

Host Environment Configurations:
CPU(processor): 2.9 GHz Dual-Core Intel Core i5
Memory: 16GB
OS: macOS Monterey
Model: MacBook Pro

1)Steps to enable QEMU VM
Enter following commands in terminal.

1. Install qemu
   brew install qemu

```
==> Pouring pixman--0.40.0.monterey.bottle.tar.gz
🍺  /usr/local/Cellar/pixman/0.40.0: 11 files, 1.3MB
==> Installing qemu dependency: snappy
==> Pouring snappy--1.1.9.monterey.bottle.tar.gz
🍺  /usr/local/Cellar/snappy/1.1.9: 18 files, 147.6KB
==> Installing qemu dependency: vde
==> Pouring vde--2.3.2_1.monterey.bottle.tar.gz
🍺  /usr/local/Cellar/vde/2.3.2_1: 73 files, 1.4MB
==> Installing qemu dependency: lz4
==> Pouring lz4--1.9.4.monterey.bottle.tar.gz
🍺  /usr/local/Cellar/lz4/1.9.4: 22 files, 685.2KB
==> Installing qemu dependency: xz
==> Pouring xz--5.2.7.monterey.bottle.tar.gz
🍺  /usr/local/Cellar/xz/5.2.7: 95 files, 1.4MB
==> Installing qemu dependency: zstd
==> Pouring zstd--1.5.2.monterey.bottle.3.tar.gz
🍺  /usr/local/Cellar/zstd/1.5.2: 31 files, 2.4MB
==> Installing qemu
==> Pouring qemu--7.1.0.monterey.bottle.2.tar.gz
🍺  /usr/local/Cellar/qemu/7.1.0: 163 files, 499.8MB
==> Running `brew cleanup qemu`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
wy9@Wenjies-MacBook-Pro ~ %
```

2. Create an QEMU image with 10G memory
   qemu-img create ubnutu.img 10G -f qcwo2
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(1CPU & 2G Memory)
   qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -boot strict=on

2)Steps to enable a docker container
First, I installed Docker Desktop, then I ran “docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=100 --time=20 run” (installed the sysbench). Shown as a following screen shot:

HW#1 Report

```
wy9@Wenjies-MacBook-Pro ~ % docker run --rm zyclonite/sysbench --test=cpu --cpu-max-prime=100 --time=20 run
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...

Threads started!

WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any opt
ions.
CPU speed:
    events per second: 300923.28

General statistics:
    total time:                          20.0001s
    total number of events:              6019125

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                   36.95
         95th percentile:                        0.00
         sum:                                18928.34

Threads fairness:
    events (avg/stddev):           6019125.0000/0.00
    execution time (avg/stddev):   18.9283/0.00
```

Some important operations for docker are:

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

```
wy9@Wenjies-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE                 COMMAND     CREATED        STATUS        PORTS     NAMES
fdf9ce79adce   zyclonite/sysbench    "/bin/sh"   17 hours ago   Up 17 hours             fervent_wu
455c50e53ca0   zyclonite/sysbench    "/bin/sh"   19 hours ago   Up 19 hours             compassionate_sutherland
072d23e5c412   zyclonite/sysbench    "/bin/sh"   22 hours ago   Up 22 hours             vigilant_blackwell
7c4a6fb2ff9e   zyclonite/sysbench    "/bin/sh"   22 hours ago   Up 22 hours             silly_keldysh
wy9@Wenjies-MacBook-Pro ~ %
```

2. Transfer file from local to container for test automation:
   sudo docker cp <file on local> <container id>:<file location at container>
3. --cpuset-cpus flags to specify number of CPU when run
4. --memory flags to specify number of memory when run
5. docker run is to run a container from an image
6. docker start is to run an existing container
7. docker exec is to enter a running container
8. docker stop is to stop a running container
9. docker ps is to list all running containers

3) Proof of the experiment

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

HW#1 Report

```
alexyang@alex999:~$ 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:   118.56

General statistics:
    total time:                          10.0065s
    total number of events:              1187

Latency (ms):
         min:                                    7.49
         avg:                                    8.38
         max:                                   79.02
         95th percentile:                       10.09
         sum:                                 9948.10

Threads fairness:
    events (avg/stddev):           1187.0000/0.00
    execution time (avg/stddev):   9.9481/0.00
```

Then I tried with --cpu-max-prime=30000, which produced the same total time ≈ 10s.

```
alexyang@alex999:~$ 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:    72.74

General statistics:
    total time:                          10.0144s
    total number of events:              729

Latency (ms):
         min:                                   13.02
         avg:                                   13.68
         max:                                   18.05
         95th percentile:                       14.46
         sum:                                 9972.13

Threads fairness:
    events (avg/stddev):           729.0000/0.00
    execution time (avg/stddev):   9.9721/0.00
```

HW#1 Report

DOCKER running environment

```
wy9@Wenjies-MacBook-Pro ~ % docker run --rm -it -m="4g" --cpuset-cpus="0-1" --entrypoint /bin/sh zyclonite/sysbench
/ # ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
wy9@Wenjies-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE                 COMMAND     CREATED          STATUS
 PORTS     NAMES
f3f0c1e5e805   zyclonite/sysbench    "/bin/sh"   14 seconds ago   Up 13 seconds
           condescending_payne
3640c7a2b219   zyclonite/sysbench    "/bin/sh"   12 minutes ago   Up 11 minutes
           gifted_heyrovsky
fdf9ce79adce   zyclonite/sysbench    "/bin/sh"   27 hours ago     Up 27 hours
           fervent_wu
455c50e53ca0   zyclonite/sysbench    "/bin/sh"   29 hours ago     Up 29 hours
           compassionate_sutherland
072d23e5c412   zyclonite/sysbench    "/bin/sh"   32 hours ago     Up 32 hours
           vigilant_blackwell
7c4a6fb2ff9e   zyclonite/sysbench    "/bin/sh"   32 hours ago     Up 32 hours
           silly_keldysh
```

QEMU running environment

```
Last login: Thu Oct 20 02:34:23 UTC 2022 on tty1


alexyang@alex999:~$
alexyang@alex999:~$
alexyang@alex999:~$
alexyang@alex999:~$
alexyang@alex999:~$
alexyang@alex999:~$ uname -a
Linux alex999 5.4.0-128-generic #144-Ubuntu SMP Tue Sep 20 11:00:04 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
alexyang@alex999:~$
```

HW#1 Report

Shell scripts for running the experiment(QEMU)

CPU test

```
CPU test 1 - 20000
******* No. 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: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   119.76

General statistics:
    total time:                          8.0024s
    total number of events:              959

Latency (ms):
         min:                                    7.50
         avg:                                    8.31
         max:                                   24.47
         95th percentile:                        9.22
         sum:                                 7965.57

Threads fairness:
    events (avg/stddev):           959.0000/0.00
    execution time (avg/stddev):   7.9656/0.00

******* No. 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: 20000

Initializing worker threads...

Threads started!

^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Ju
^X Exit         ^R Read File    ^\ Replace      ^U Paste Text   ^T To
```

HW#1 Report

fileIO test

```
FileIO testcase 1, Threads=16, size=3G
***** No. 1
WARNING: --num-threads is deprecated, use --threads instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
Creating file test_file.24
Creating file test_file.25
Creating file test_file.26
Creating file test_file.27
Creating file test_file.28
Creating file test_file.29
Creating file test_file.30
Creating file test_file.31
Creating file test_file.32
Creating file test_file.33
Creating file test_file.34
Creating file test_file.35
Creating file test_file.36
Creating file test_file.37
```

HW#1 Report

Shell scripts for running the experiment(Docker)

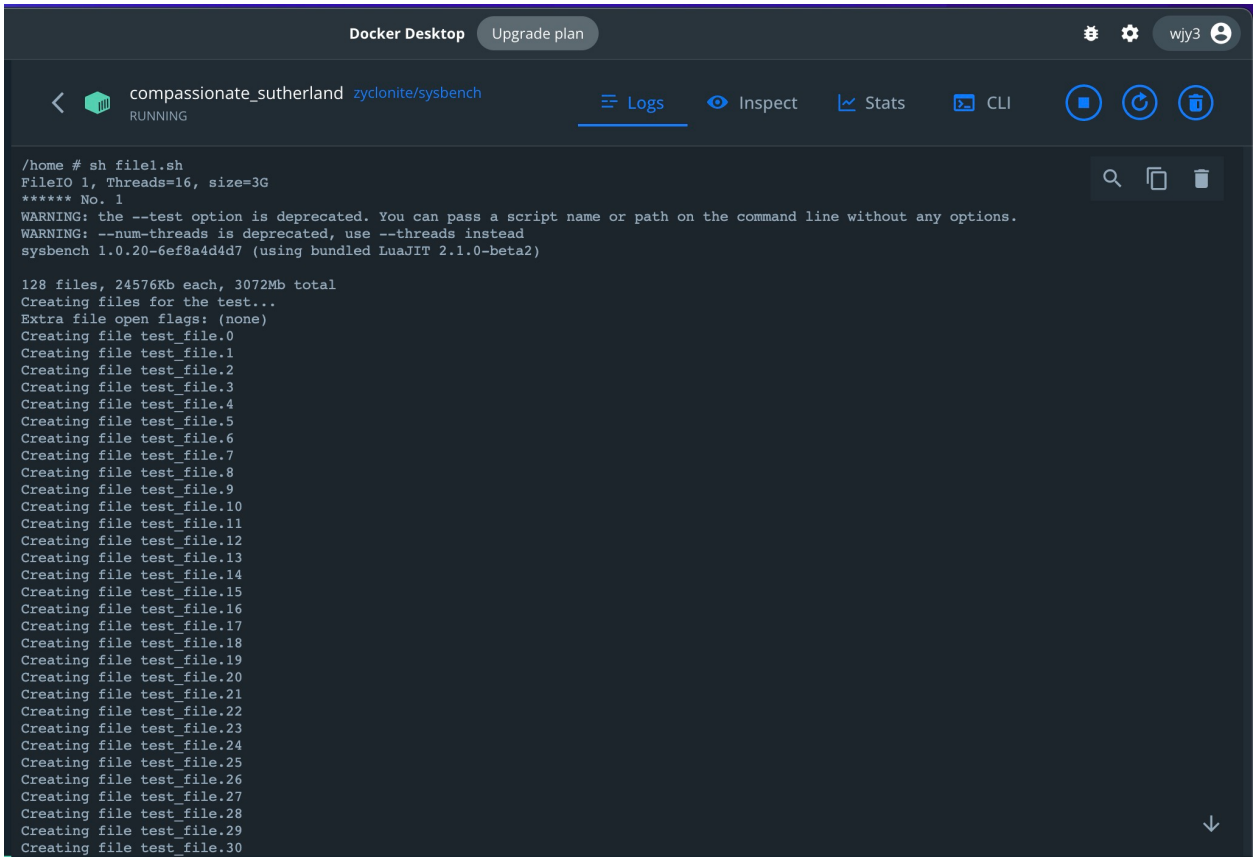
CPU test

```
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:   338.26

General statistics:
    total time:                          8.0008s
    total number of events:              2707

Latency (ms):
         min:                                    2.82
         avg:                                    2.95
         max:                                   12.15
         95th percentile:                        3.13
         sum:                                 7997.40

Threads fairness:
    events (avg/stddev):           2707.0000/0.00
    execution time (avg/stddev):   7.9974/0.00

******* No.5
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: 20000

Initializing worker threads...

Threads started!

WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
CPU speed:
    events per second:   336.88

General statistics:
    total time:                          8.0007s
    total number of events:              2696

Latency (ms):
         min:                                    2.86
         avg:                                    2.97
         max:                                    9.28
         95th percentile:                        3.19
         sum:                                 7996.75

Threads fairness:
    events (avg/stddev):           2696.0000/0.00
    execution time (avg/stddev):   7.9967/0.00

wy9@Wenjies-MacBook-Pro testcase1 %
```

HW#1 Report

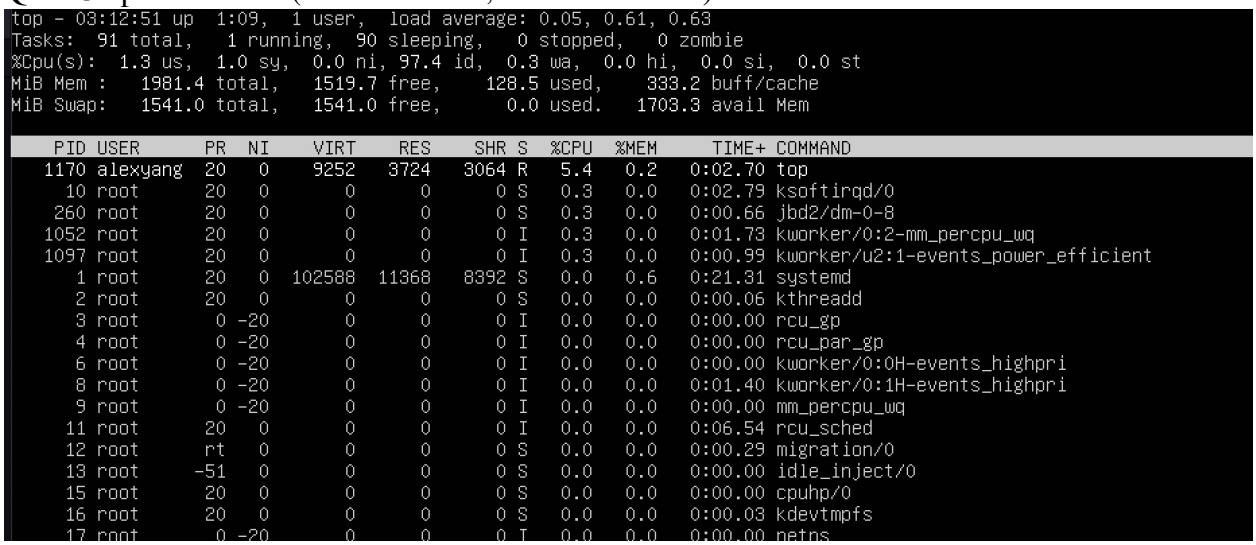
fileIO test

```
/home # ls
file1.sh
/home # sh file1.sh
FileIO 1, Threads=16, size=3G
****** No. 1
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 1.0.20-6ef8a4d4d7 (using bundled LuaJIT 2.1.0-beta2)

128 files, 24576Kb each, 3072Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
```

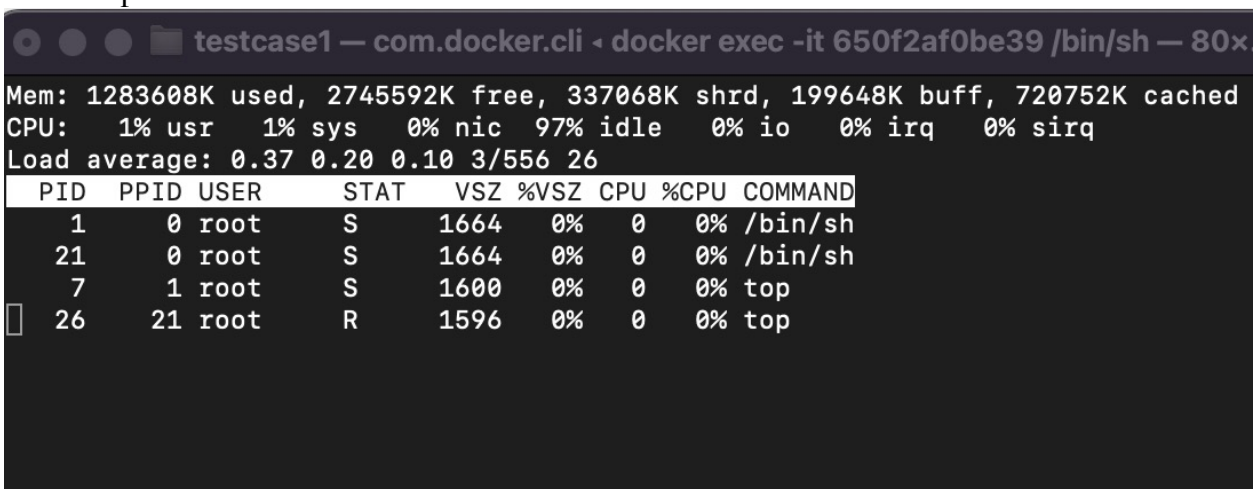


HW#1 Report

For CPU utilization and I/O disk utilization, I used “top” commands to access the information:
Shown as the 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
QEMU cpu utilization(user-level: 1.3; kernel-level:1.0):

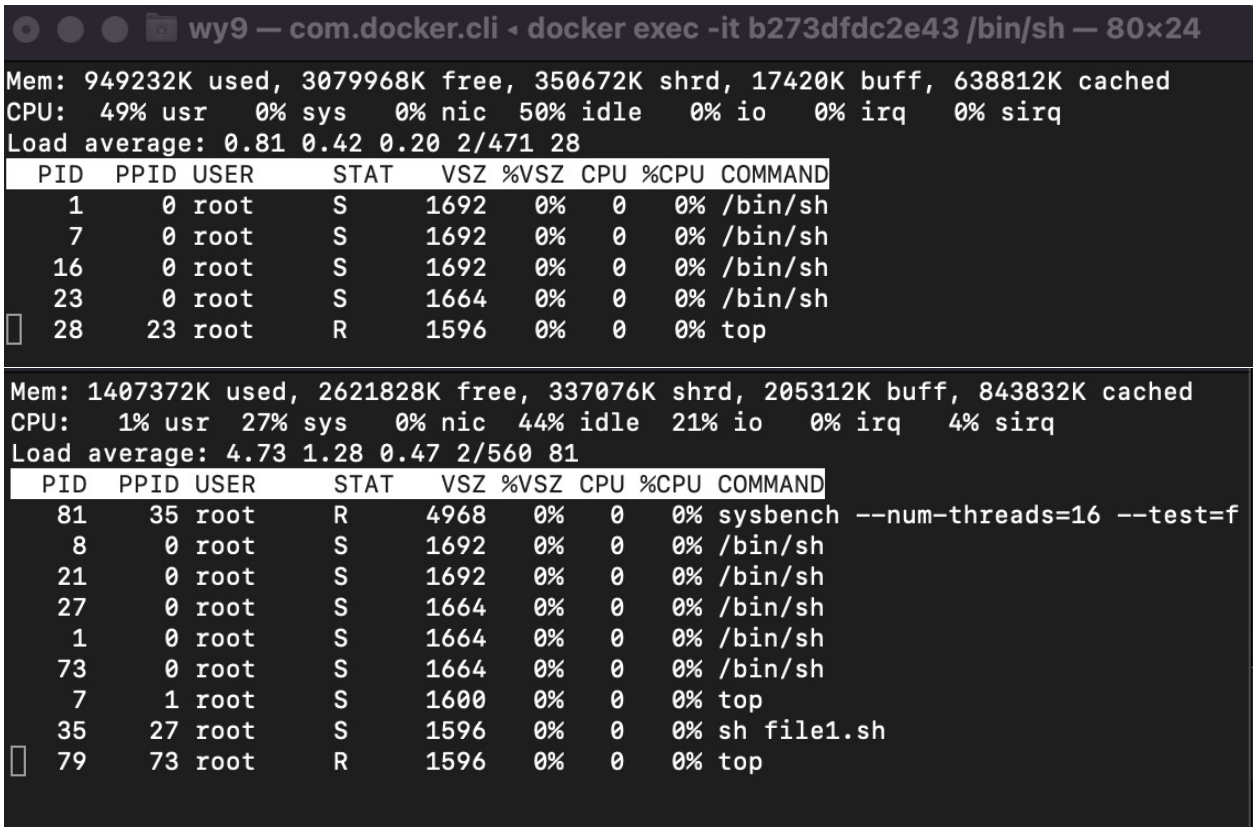


Docker cpu utilization:

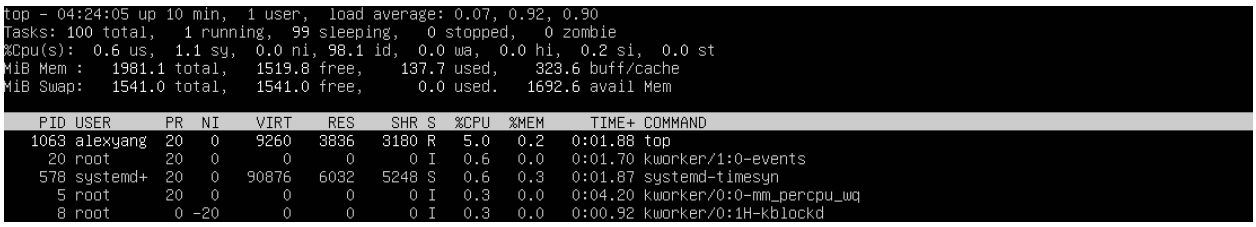


HW#1 Report

Fiel IO:
IO's disk utilization is shown below:
Docker:



QEMU:



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 by fileio test.

Using shell scripts and the sysbench output were sotred into .txt files. All the experiments was shown as following screenshot. For example: ./CPU1.sh > CPU1.txt which will run "CPU1.sh" script anr store the output into "CPU1.txt" file.

HW#1 Report

```
alexyang@alex999:~$ 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:    72.74

General statistics:
    total time:                          10.0144s
    total number of events:              729

Latency (ms):
         min:                                    13.02
         avg:                                    13.68
         max:                                    18.05
         95th percentile:                        14.46
         sum:                                  9972.13

Threads fairness:
    events (avg/stddev):           729.0000/0.00
    execution time (avg/stddev):   9.9721/0.00

alexyang@alex999:~$ [ 1506.429720] watchdog: BUG: soft lockup - CPU#0 stuck for 124s! [dpkg-preconfigu:1431]
[ 2937.381781] blk_update_request: I/O error, dev fd0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 0
[ 4548.176529] blk_update_request: I/O error, dev fd0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 0

alexyang@alex999:~$ ls
CPU1_2.txt  CPU1.sh   CPU2_2.txt  CPU2.sh   file1_2.txt  file1.sh   file2_2.txt  file2.sh
CPU1_3.txt  CPU1.txt  CPU2_3.txt  CPU2.txt  file1_3.txt  file1.txt  file2_3.txt  file2.txt
```

4)Three different scenarios for each virtualization technology

There are three scenarios by specifying three different number of CPU and memory.

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

Some commands about how I specifying those scenarios when running experiment.
Qemu:

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:

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

HW#1 Report

prime=30000

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

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=3 & Memory=4G

CPU test(events/second):

	QEMU					
	Testcase_1--cpu-max-prime=20000			Testcase_2--cpu-max-prime=30000		
	senario_1	senario_2	senario_3	senario_1	senario_2	senario_3
run_1	119.76	124.77	123.28	70.89	72.36	71.26
run_2	124.55	124.93	112.61	58.4	71.95	70.31
run_3	123.83	124.71	118.66	62.02	71.5	71.18
run_4	123.75	124.48	122.75	65.38	71.72	71.8
run_5	120.81	125.03	122	58.07	71.94	71.26
max	124.55	125.03	123.28	70.89	72.36	71.8
min	119.76	124.48	112.61	58.07	71.5	70.31
avg	122.54	124.784	119.86	62.952	71.894	71.162
std	2.114804956	0.212084889	4.433694847	5.3462669	0.31934308	0.5367681

HW#1 Report

DOCKER(CPU)						
	Testcase_1--cpu-max-prime=20000			Testcase_2--cpu-max-prime=30000		
	senario_1	senario_2	senario_3	senario_1	senario_2	senario_3
run_1	336.48	338.51	323.44	172.35	189.14	183.61
run_2	341.23	310.8	337.82	181.72	198.14	178.28
run_3	339.38	307.02	338.13	186.89	198.11	182.55
run_4	340.56	306.85	340.4	186.67	198.87	197.39
run_5	341.2	334.25	317.15	177.73	197.47	197.95
max	341.23	338.51	340.4	186.89	198.87	197.95
min	336.48	306.85	317.15	172.35	189.14	178.28
avg	339.77	319.486	331.388	181.072	196.346	187.956
std	1.986001007	15.57565825	10.41552543	6.18160335	4.05864879	9.09145093

Fileio test:

QEMU(fileIO)									
	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	7.48	4.99	7.75	7.83	5.21	7.87	9.53	6.35	6.52
run_2	7.58	5.05	8.11	8.13	5.42	7.64	9.39	6.26	6.6
run_3	7.95	5.3	7.74	9.03	6.01	6.87	9.06	6.04	6.85
run_4	8.47	5.65	7.26	7.97	5.31	7.68	10.13	6.76	6.13
run_5	7.56	5.04	8.14	8.05	5.36	7.68	11.56	7.71	5.36
max	8.47	5.65	8.14	9.03	6.01	7.87	11.56	7.71	6.85
min	7.48	4.99	7.26	7.83	5.21	6.87	9.06	6.04	5.36
avg	7.808	5.206	7.8	8.202	5.462	7.548	9.934	6.624	6.292
std	0.412152884	0.275916654	0.356861318	0.47594117	0.31586389	0.38944833	0.988094125	0.660779842	0.58160983

DOCKER(fileIO)									
	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	57.84	38.54	1.13	56.61	37.72	1.14	92.58	61.73	0.71
run_2	78	52	0.84	77.52	51.66	0.84	86.97	57.98	0.75
run_3	61.54	41.01	1.06	80.58	53.71	0.81	95.56	63.72	0.68
run_4	57.05	38.02	1.15	78.53	52.35	0.83	81.12	54.08	0.81
run_5	48.28	32.18	1.36	67.9	45.25	0.96	79.61	53.07	0.82
max	78	52	1.36	80.58	53.71	1.14	95.56	63.72	0.82
min	48.28	32.18	0.84	56.61	37.72	0.81	79.61	53.07	0.68
avg	60.542	40.35	1.108	72.228	48.138	0.916	87.168	58.116	0.754
std	10.90446331	7.273444851	0.187002674	10.0016234	6.67210387	0.13831124	6.954428086	4.643159485	0.06107373

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

HW#1 Report

4. Increase the number of CPU and memory can increase the fileiIO performance, with a lower average latency and higher throughput.

Git Repository Information
Link of Repository:
https://github.com/YCoder099/coen-241