COEN 241 HW 1
System vs OS Virtualization

Ruopu He

W1650128

Detailed configurations of your experimental setup

CPU: Apple M2
Memory: 8 GB
OS: macOS Monterey version 12.6
Disk space: 256 GB

main steps to enable a QEMU VM

Install QEMU by brew:

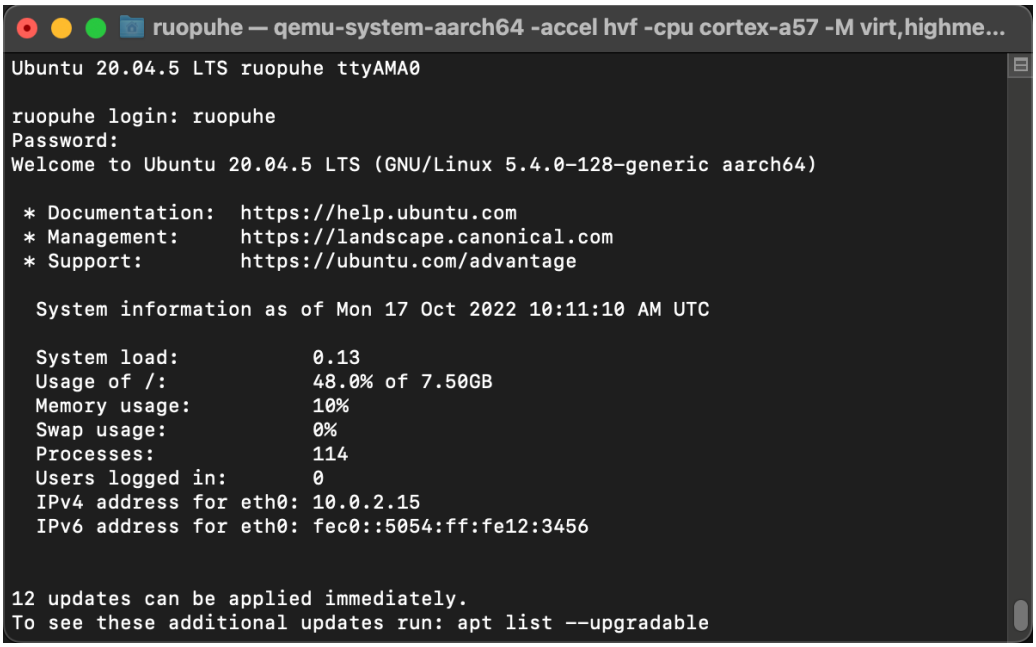
```
brew install qemu
```

Create a hard disk:

```
qemu-img create -f qcow2 disk.qcow2 10G
```

Run QEMU with following commands to install Ubuntu:

```
qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048 -smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/7.1.0/share/qemu/edk2-aarch64-
code.fd,if=pflash,format=raw,readonly=on \
-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/ruopuhe/Downloads/ubuntu-20.04.5-live-server-arm64.iso \
-device usb-ehci -device usb-kbd -device usb-mouse -usb \
-nographic
```



Main steps to enable the Docker container

Install docker desktop:

```
brew install --cask docker
```

Start docker container:

```
open /Applications/Docker.app
```

Installed the docker image zyclonite/sysbench:

```
docker pull zyclonite/sysbench
```

Run sysbench and get my own image in docker:

```
docker run --rm -it --entrypoint /bin/sh zyclonite/sysbench
```

```
ruopuhe@Ruopus-MacBook-Pro ~ % docker run --rm -it --entrypoint /bin/sh zyclonite/sysbench
/ # touch Dockerfile
/ # vi Dockerfile
/ # exit
ruopuhe@Ruopus-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE                 COMMAND     CREATED             STATUS             PORTS
8f5904aa1281   zyclonite/sysbench    "/bin/sh"   40 minutes ago      Up 40 minutes
471a0f0cba6b   zyclonite/sysbench    "/bin/sh"   About an hour ago   Up About an hour
ruopuhe@Ruopus-MacBook-Pro ~ % docker images
REPOSITORY                  TAG       IMAGE ID       CREATED         SIZE
docker101tutorial           latest    c47ee4f83ac0   5 hours ago     27.4MB
ruopuhe/docker101tutorial   latest    c47ee4f83ac0   5 hours ago     27.4MB
alpine/git                  latest    8bfbb50cd816   10 days ago     43.4MB
ubuntu                      20.04     f12f227aa3fd   12 days ago     65.6MB
zyclonite/sysbench          latest    8731aa4184ff   10 months ago   9.19MB
```

```
dcoker commit 8f5904aa1281 my_sysbench
```

```
ruopuhe@Ruopus-MacBook-Pro ~ % docker commit 8f5904aa1281 my_sysbench
sha256:da60f980ef8c75480261730c79012b3172ef9de462046944cec1cefa36fda951
ruopuhe@Ruopus-MacBook-Pro ~ % docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
my_sysbench                 latest    da60f980ef8c   18 seconds ago   9.19MB
docker101tutorial           latest    c47ee4f83ac0   5 hours ago      27.4MB
ruopuhe/docker101tutorial   latest    c47ee4f83ac0   5 hours ago      27.4MB
alpine/git                  latest    8bfbb50cd816   10 days ago      43.4MB
ubuntu                      20.04     f12f227aa3fd   12 days ago      65.6MB
zyclonite/sysbench          latest    8731aa4184ff   10 months ago    9.19MB
ruopuhe@Ruopus-MacBook-Pro ~ % docker history my_sysbench
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
da60f980ef8c   2 minutes ago                                                   31B
8731aa4184ff   10 months ago   /bin/sh -c #(nop) CMD [ "--help" ]              3.86MB    FROM docker.io/library/alpine:3.15
<missing>      10 months ago   /bin/sh -c #(nop) ENTRYPOINT [ "sysbench" ]     0B
<missing>      10 months ago   |1 version=1.0.20-r0 /bin/sh -c apk add --no…   0B
<missing>      10 months ago   /bin/sh -c #(nop) ARG version=1.0.20-r0          0B
<missing>      10 months ago   /bin/sh -c #(nop) LABEL description "Sysbenc…   0B
<missing>      10 months ago   /bin/sh -c #(nop) LABEL version "1.0.20"        0B
<missing>      10 months ago   /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      10 months ago   /bin/sh -c #(nop) ADD file:df538113122843069…   5.33MB
```

useful docker operations:

Check running docker containers:

```
docker ps
```

Check local docker images:

```
docker images
```

Commit a Docker image:

```
docker commit <container ID><images_name>
```

Shell scripts for running the experiment:

CPU-test.sh

```
sysbench --test=cpu --cpu-max-prime=100000 --time=30 run
sysbench --test=cpu --cpu-max-prime=300000 --time=30 run
sysbench --test=cpu --cpu-max-prime=500000 --time=30 run
```

FileIO-test.sh

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=rndrw --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup

sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=seqwr --max-time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup

sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=rndwr --max-time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```

Screenshots of the experiments

CPU test:

Run different prime numbers on both Docker and QEMU. All three tests result are shown as screen snapshots.

Docker:

Run sysbench: `docker run --rm -it --entrypoint /bin/sh my_sysbench`

Test 1: sysbench --test=cpu --cpu-max-prime=100000 --time=30 run

ruopuhe — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh my_sysbench — 113×36

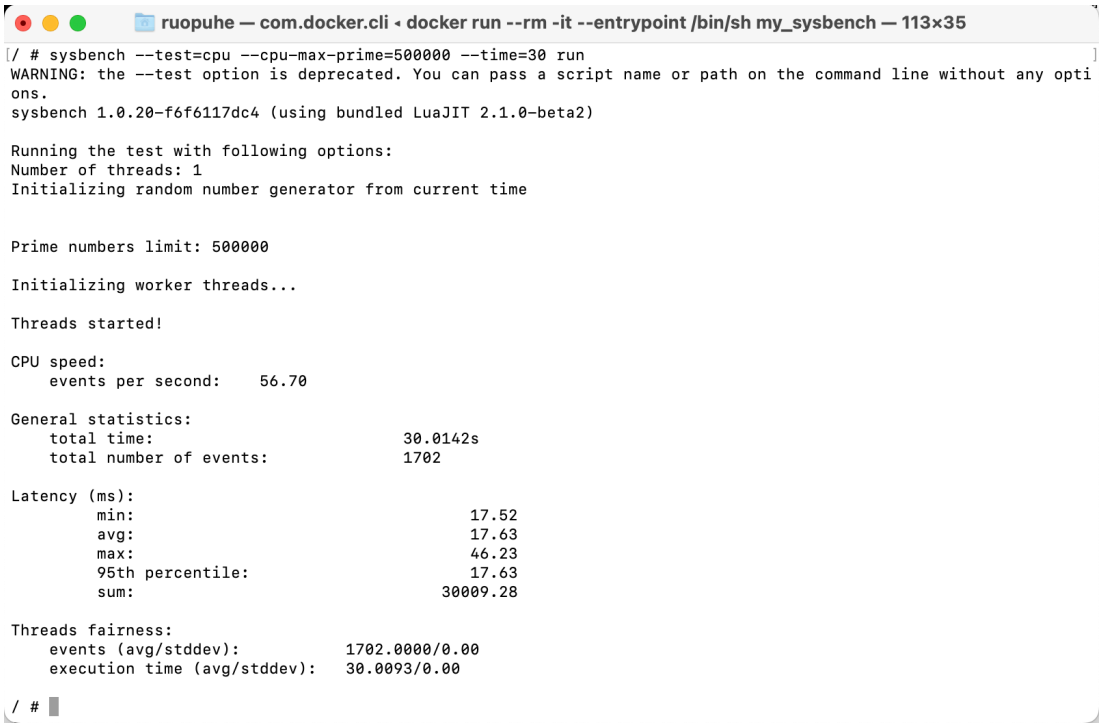
```
ruopuhe@Ruopus-MacBook-Pro ~ % docker run --rm -it --entrypoint /bin/sh my_sysbench
/ # sysbench --test=cpu --cpu-max-prime=100000 --time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any opti
ons.
sysbench 1.0.20-f6f6117dc4 (using 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: 100000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   465.35

General statistics:
    total time:                          30.0004s
    total number of events:              13961

Latency (ms):
         min:                                    2.12
         avg:                                    2.15
         max:                                   11.12
         95th percentile:                        2.22
         sum:                                29989.71

Threads fairness:
    events (avg/stddev):           13961.0000/0.00
    execution time (avg/stddev):   29.9897/0.00

/ #
```

Test 2: sysbench --test=cpu --cpu-max-prime=300000 --time=30 run

ruopuhe — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh my_sysbench — 113×35

```
/ # sysbench --test=cpu --cpu-max-prime=300000 --time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any opti
ons.
sysbench 1.0.20-f6f6117dc4 (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: 300000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   110.83

General statistics:
    total time:                          30.0007s
    total number of events:              3325

Latency (ms):
         min:                                    8.89
         avg:                                    9.02
         max:                                   36.11
         95th percentile:                        9.22
         sum:                                29984.93

Threads fairness:
    events (avg/stddev):           3325.0000/0.00
    execution time (avg/stddev):   29.9849/0.00

/ #
```

Test 3: sysbench --test=cpu --cpu-max-prime=500000 --time=30 run

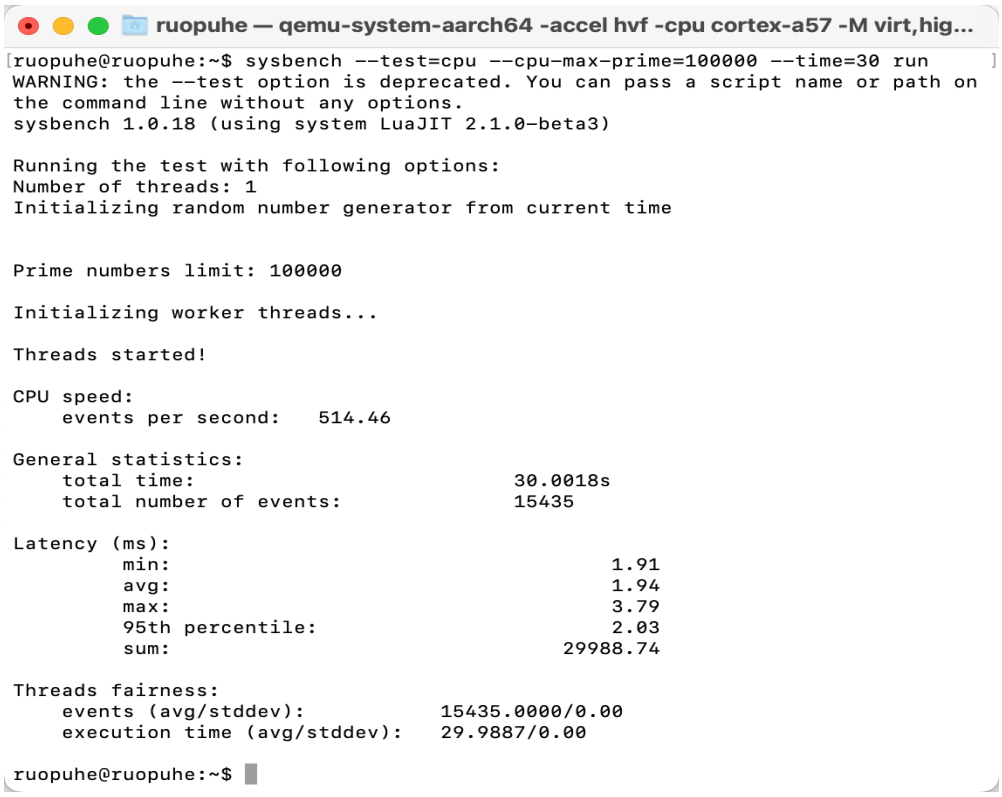


QEMU:

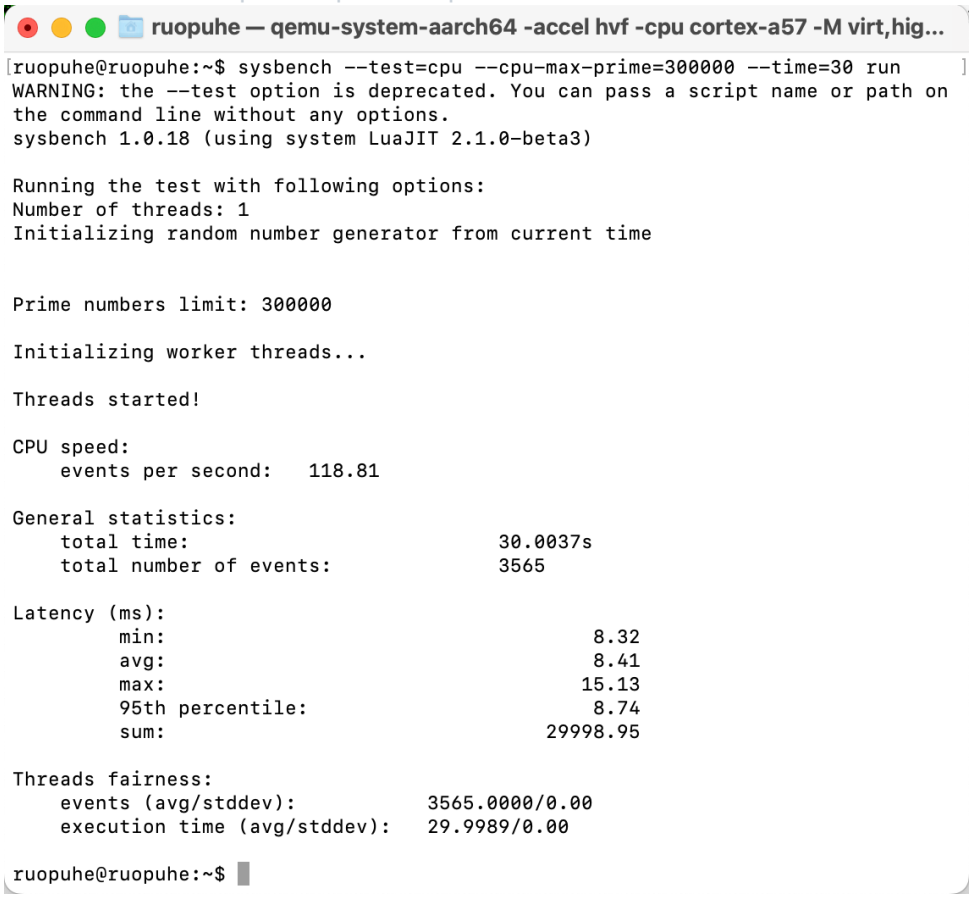
Run QEMU with 2 Cores, 2G memory:

```
qemu-system-aarch64 \
-accel hvf -cpu cortex-a57 -M virt,highmem=off -m 2048 -smp 2 \
-drive file=/opt/homebrew/Cellar/qemu/7.1.0/share/qemu/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
-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 \
-device usb-ehci -device usb-kbd -device usb-mouse -usb \
-nographic
```

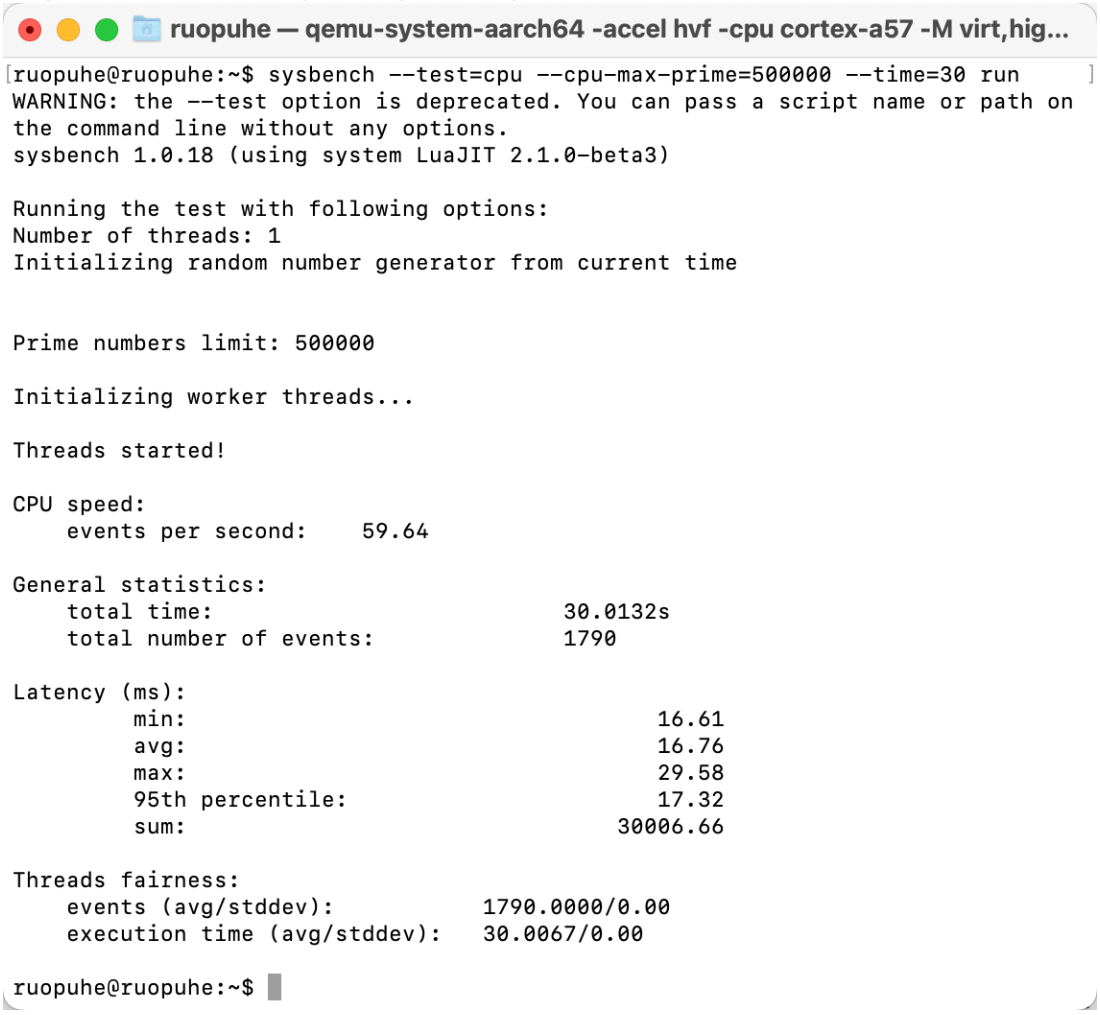
Test 1: `sysbench --test=cpu --cpu-max-prime=100000 --time=30 run`



Test 2: sysbench --test=cpu --cpu-max-prime=300000 --time=30 run



Test 3: `sysbench --test=cpu --cpu-max-prime=500000 --time=30 run`

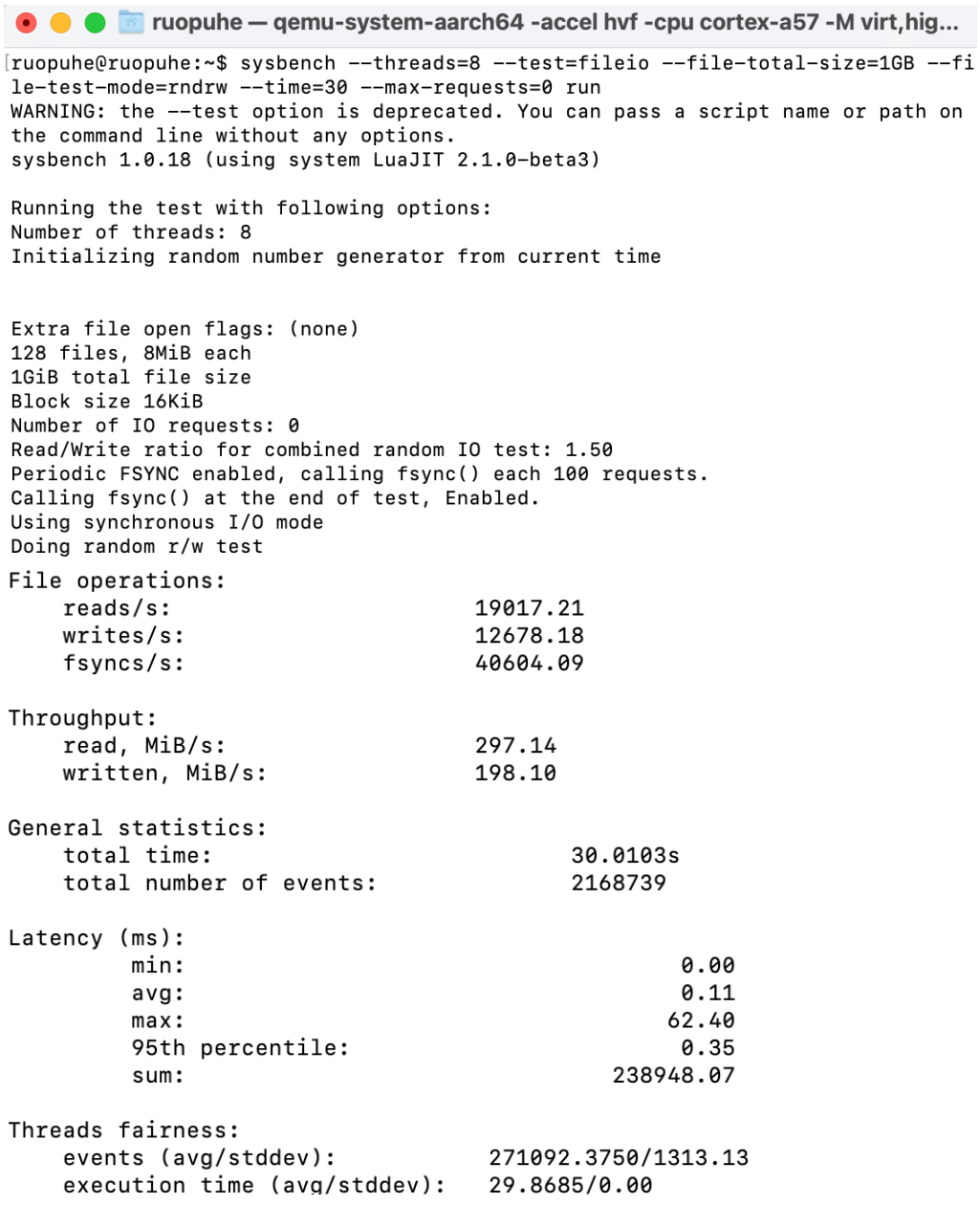


FileIO test:

QEMU:

Test 1: Random read and write

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=rndrw --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```



Test 2: Sequential write

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-mode=seqwr --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```

```
ruopuhe@ruopuhe:~$ sysbench --threads=8 --test=fileio --file-total-size=1GB --fi
le-test-mode=seqwr --time=30 --max-requests=0 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: 8
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 8MiB each
1GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!

File operations:
    reads/s:                      0.00
    writes/s:                     41191.43
    fsyncs/s:                     52755.78

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               643.62

General statistics:
    total time:                          30.0105s
    total number of events:              2818435

Latency (ms):
         min:                                    0.00
         avg:                                    0.08
         max:                                  195.72
         95th percentile:                        0.32
         sum:                               237571.97

Threads fairness:
    events (avg/stddev):           352304.3750/1832.16
    execution time (avg/stddev):   29.6965/0.01
```

Test 3: Random write

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=rndwr --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```

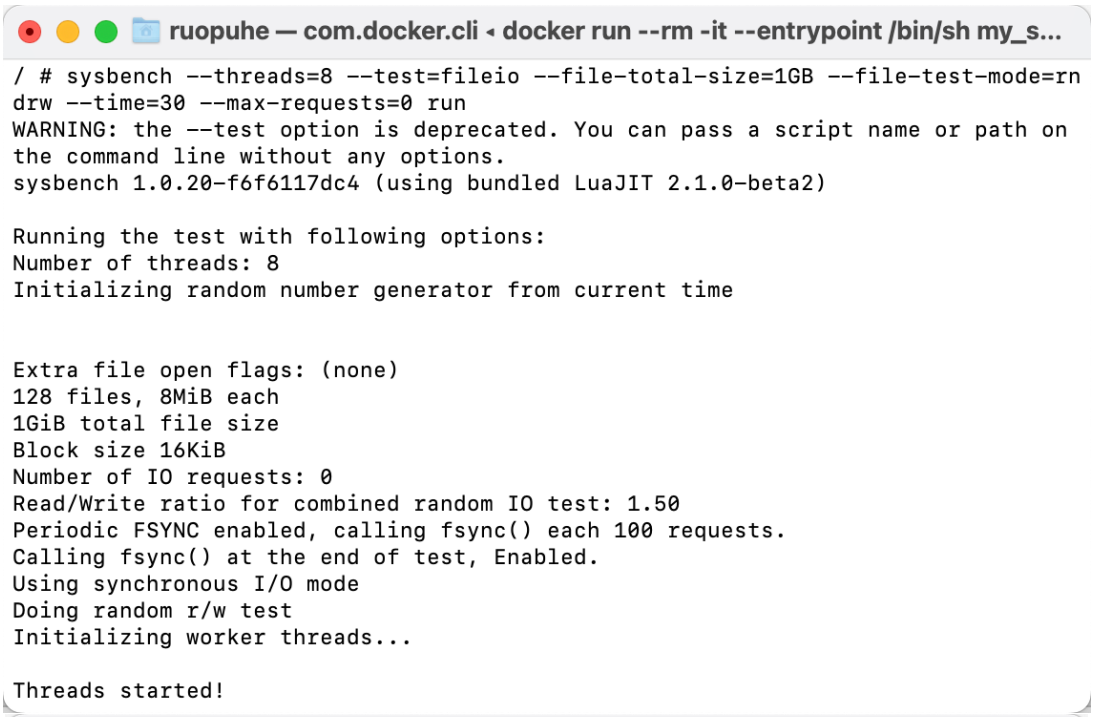
```
ruopuhe@ruopuhe:~$ sysbench --threads=8 --test=fileio --file-total-size=1GB --fi
le-test-mode=rndwr --time=30 --max-requests=0 run
WARNING: the --test option is deprecated. You can pass a script name or path on
the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 8
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 write test
Initializing worker threads...

Threads started!

File operations:
    reads/s:                      0.00
    writes/s:                     22689.64
    fsyncs/s:                     29072.96

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               354.53

General statistics:
    total time:                          30.0132s
    total number of events:              1552563

Latency (ms):
         min:                                    0.00
         avg:                                    0.15
         max:                                  190.13
         95th percentile:                        0.46
         sum:                               239390.46

Threads fairness:
    events (avg/stddev):           194070.3750/1529.31
    execution time (avg/stddev):   29.9238/0.00
```

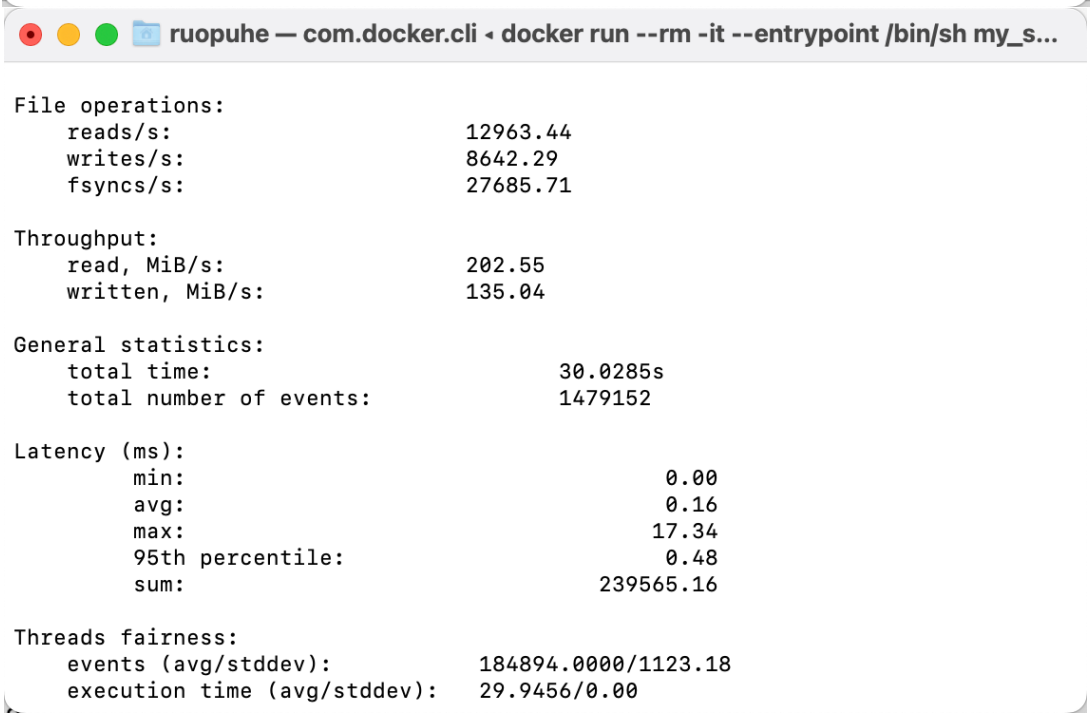
Docker:

Run sysbench with this command: `docker run --rm -it --entrypoint /bin/sh my_sysbench`

Test 1: Random read and write

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-mode=rndrw --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```





Test 2: Sequential write

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=seqwr --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```

```
ruopuhe — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh my_s...
/ # sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-mode=se
qwr --time=30 --max-requests=0 run
WARNING: the --test option is deprecated. You can pass a script name or path on
the command line without any options.
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 8
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 8MiB each
1GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!
```

```
ruopuhe — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh my_s...
File operations:
    reads/s:                      0.00
    writes/s:                     21546.07
    fsyncs/s:                     27611.14

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               336.66

General statistics:
    total time:                          30.0328s
    total number of events:              1475330

Latency (ms):
         min:                                    0.00
         avg:                                    0.16
         max:                                  525.97
         95th percentile:                        0.44
         sum:                               239606.48

Threads fairness:
    events (avg/stddev):           184416.2500/1100.38
    execution time (avg/stddev):   29.9508/0.00
```

Test 3: Random write

```
sysbench --threads=8 --test=fileio --file-total-size=1GB prepare
sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-
mode=rndwr --time=30 --max-requests=0 run
sysbench --threads=8 --test=fileio --file-total-size=1GB cleanup
```

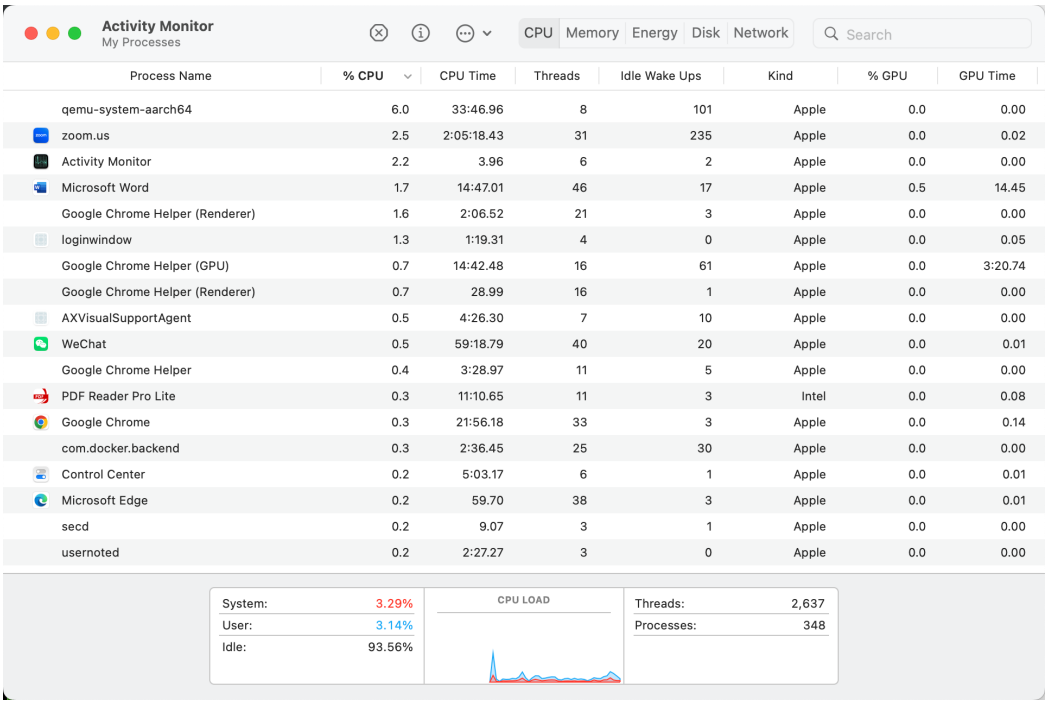
```
ruopuhe — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh my_s...
/ # sysbench --threads=8 --test=fileio --file-total-size=1GB --file-test-mode=rn
dwr --time=30 --max-requests=0 run
WARNING: the --test option is deprecated. You can pass a script name or path on
the command line without any options.
sysbench 1.0.20-f6f6117dc4 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 8
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 write test
Initializing worker threads...

Threads started!
```

```
ruopuhe — com.docker.cli ◂ docker run --rm -it --entrypoint /bin/sh my_s...
File operations:
    reads/s:                      0.00
    writes/s:                     15525.66
    fsyncs/s:                     19903.08

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               242.59

General statistics:
    total time:                          30.0333s
    total number of events:              1063048

Latency (ms):
         min:                                    0.00
         avg:                                    0.23
         max:                                   17.58
         95th percentile:                        0.61
         sum:                               239669.87

Threads fairness:
    events (avg/stddev):           132881.0000/509.37
    execution time (avg/stddev):   29.9587/0.00
```

CPU utilization for user-level and kernel -level:

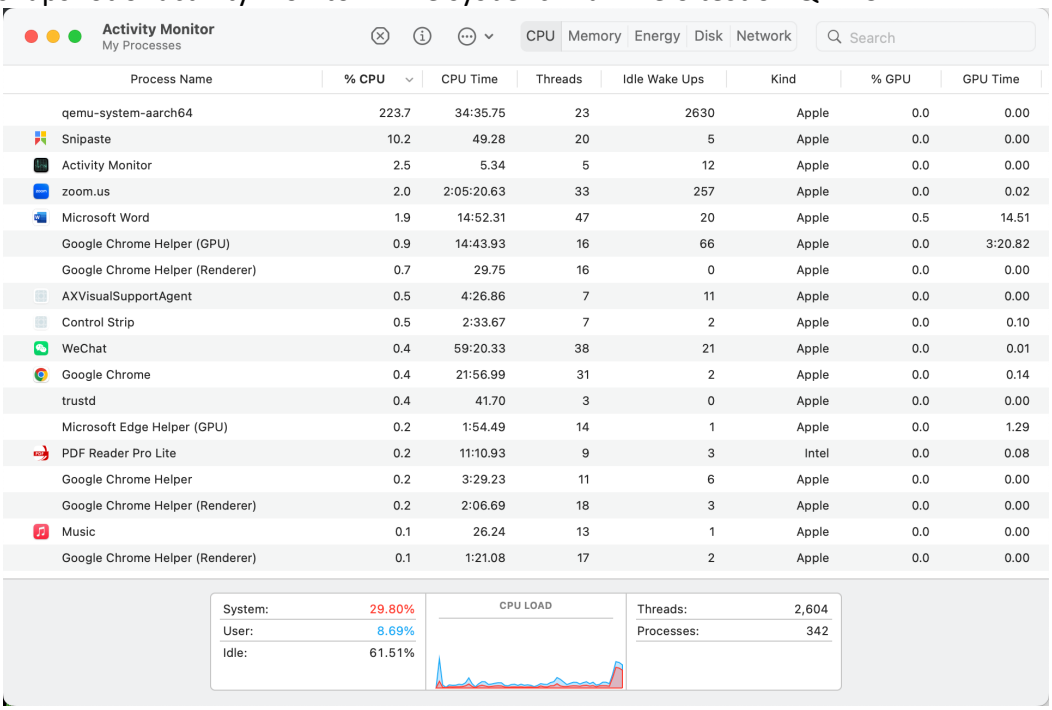
QEMU:

Screen snapshot of activity monitor while QEMU is idle:



Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	Kind	% GPU	GPU Time
qemu-system-aarch64	6.0	33:46.96	8	101	Apple	0.0	0.00
zoom.us	2.5	2:05:18.43	31	235	Apple	0.0	0.02
Activity Monitor	2.2	3.96	6	2	Apple	0.0	0.00
Microsoft Word	1.7	14:47.01	46	17	Apple	0.5	14.45
Google Chrome Helper (Renderer)	1.6	2:06.52	21	3	Apple	0.0	0.00
loginwindow	1.3	1:19.31	4	0	Apple	0.0	0.05
Google Chrome Helper (GPU)	0.7	14:42.48	16	61	Apple	0.0	3:20.74
Google Chrome Helper (Renderer)	0.7	28.99	16	1	Apple	0.0	0.00
AXVisualSupportAgent	0.5	4:26.30	7	10	Apple	0.0	0.00
WeChat	0.5	59:18.79	40	20	Apple	0.0	0.01
Google Chrome Helper	0.4	3:28.97	11	5	Apple	0.0	0.00
PDF Reader Pro Lite	0.3	11:10.65	11	3	Intel	0.0	0.08
Google Chrome	0.3	21:56.18	33	3	Apple	0.0	0.14
com.docker.backend	0.3	2:36.45	25	30	Apple	0.0	0.00
Control Center	0.2	5:03.17	6	1	Apple	0.0	0.01
Microsoft Edge	0.2	59.70	38	3	Apple	0.0	0.01
secd	0.2	9.07	3	1	Apple	0.0	0.00
usernoted	0.2	2:27.27	3	0	Apple	0.0	0.00

Screen snapshot of activity monitor while sysbench run fileio test on QEMU:

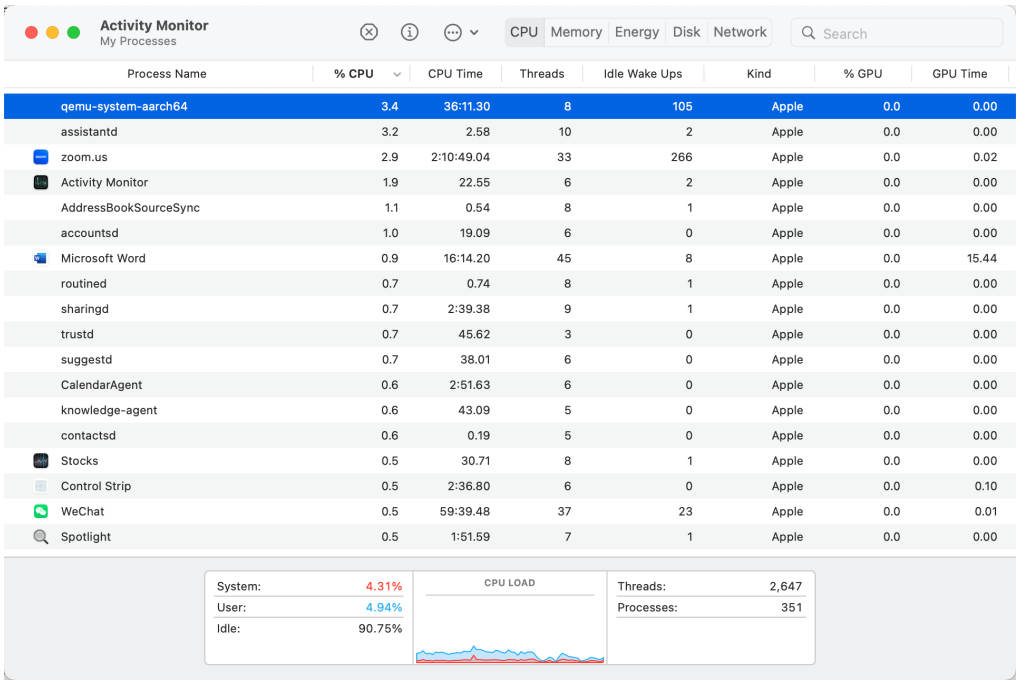


Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	Kind	% GPU	GPU Time
qemu-system-aarch64	223.7	34:35.75	23	2630	Apple	0.0	0.00
Snipaste	10.2	49.28	20	5	Apple	0.0	0.00
Activity Monitor	2.5	5.34	5	12	Apple	0.0	0.00
zoom.us	2.0	2:05:20.63	33	257	Apple	0.0	0.02
Microsoft Word	1.9	14:52.31	47	20	Apple	0.5	14.51
Google Chrome Helper (GPU)	0.9	14:43.93	16	66	Apple	0.0	3:20.82
Google Chrome Helper (Renderer)	0.7	29.75	16	0	Apple	0.0	0.00
AXVisualSupportAgent	0.5	4:26.86	7	11	Apple	0.0	0.00
Control Strip	0.5	2:33.67	7	2	Apple	0.0	0.10
WeChat	0.4	59:20.33	38	21	Apple	0.0	0.01
Google Chrome	0.4	21:56.99	31	2	Apple	0.0	0.14
trustd	0.4	41.70	3	0	Apple	0.0	0.00
Microsoft Edge Helper (GPU)	0.2	1:54.49	14	1	Apple	0.0	1.29
PDF Reader Pro Lite	0.2	11:10.93	9	3	Intel	0.0	0.08
Google Chrome Helper	0.2	3:29.23	11	6	Apple	0.0	0.00
Google Chrome Helper (Renderer)	0.2	2:06.69	18	3	Apple	0.0	0.00
Music	0.1	26.24	13	1	Apple	0.0	0.00
Google Chrome Helper (Renderer)	0.1	1:21.08	17	2	Apple	0.0	0.00

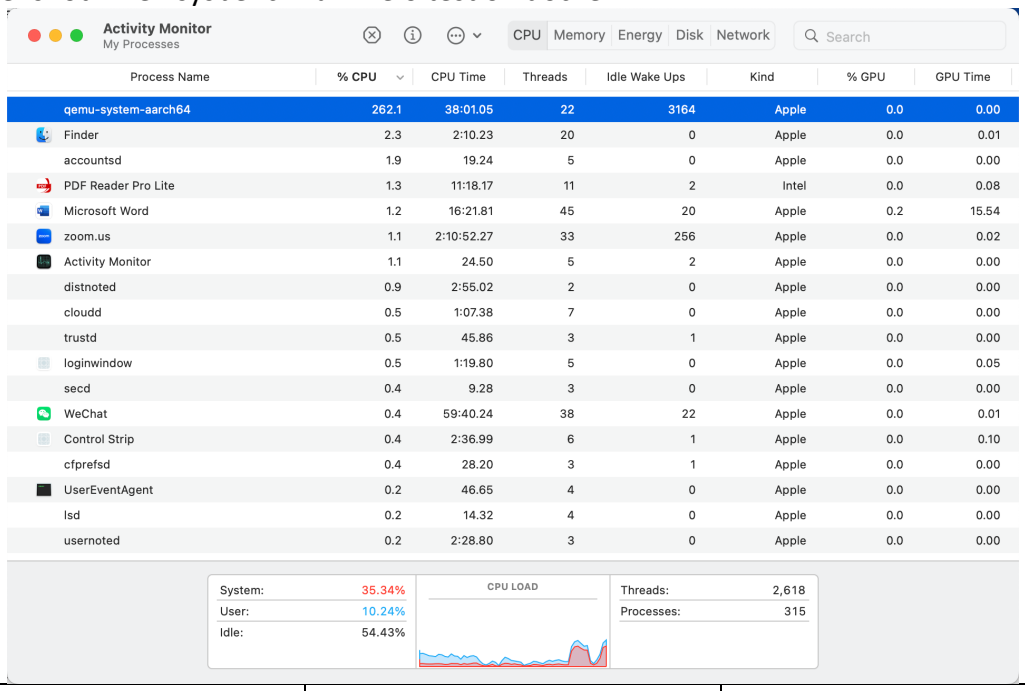
From the above snapshots, we can see the CPU usage increases from 6% to 223.7%

Docker:

The Screenshot of activity monitor while docker is idle:



The Screenshot when sysbench run fileio test on docker:



	Sysbench not running	Sysbench running
User level CPU	3.29%	29.80%
Kernel level CPU	4.31%	35.34%

Presentation and analysis of the performance data:

Manually drop cache in the Host using this command:

```
sync && sudo purge
```

CPU test for Docker:

Test 1: --cpu-max-prime=100000

	avg	min	max	Events per sec
1	2.15	2.12	11.12	465.35
2	2.16	2.12	12.69	461.79
3	2.16	2.13	13.48	463.66
4	2.15	2.12	14.63	464.28
5	2.14	2.12	14.71	466.10

Test 2: --cpu-max-prime=300000

	avg	min	max	Events per sec
1	9.02	8.89	36.11	110.83
2	9.00	8.89	29.97	111.05
3	9.00	8.89	31.44	111.00
4	8.98	8.89	34.06	111.27
5	9.00	8.89	29.99	111.12

Test 3: --cpu-max-prime=500000

	avg	min	max	Events per sec
1	17.56	17.23	46.23	56.70
2	17.72	17.52	50.66	56.42
3	17.81	17.55	49.73	56.14
4	17.72	17.52	33.87	56.42
5	17.74	17.52	51.47	56.34

CPU test for QEMU:

Test 1: --cpu-max-prime=100000

	avg	min	max	Events per sec
1	1.94	1.91	3.79	514.46
2	1.93	1.91	5.26	518.17
3	1.94	1.91	4.81	515.91
4	1.94	1.91	5.23	516.01
5	1.94	1.91	5.50	514.53

Test 2: --cpu-max-prime=300000

	avg	min	max	Events per sec
1	8.41	8.32	15.13	118.81
2	8.40	8.32	15.50	118.99
3	8.41	8.32	16.12	118.86
4	8.38	8.32	16.10	119.27
5	8.37	8.32	16.12	119.45

Test 3: --cpu-max-prime=500000

	avg	min	max	Events per sec
1	16.74	16.61	29.98	59.64
2	16.79	16.61	31.81	59.54
3	16.82	16.61	27.07	59.42
4	16.76	16.61	31.71	59.66
5	16.78	16.61	30.86	59.56

FileIO test for Docker:

Test 1: Random read and write

	avg	min	max	read	write
1	0.16	0.00	17.34	202.55	135.04
2	0.17	0.00	13.50	197.34	131.56
3	0.16	0.00	31.90	199.54	133.03
4	0.17	0.00	34.63	193.76	129.17
5	0.16	0.00	25.10	203.90	135.93

Test 2: Sequential write

	avg	min	max	read	write
1	0.16	0.00	525.97	0.00	336.66
2	0.16	0.00	16.27	0.00	325.03
3	0.17	0.00	27.46	0.00	331.28
4	0.16	0.00	27.79	0.00	333.19
5	0.17	0.00	26.93	0.00	327.92

Test 3: Random write

	avg	min	max	read	write
1	0.23	0.00	17.58	0.00	242.59
2	0.23	0.00	19.02	0.00	234.88
3	0.23	0.00	33.89	0.00	235.55
4	0.23	0.00	18.54	0.00	237.11
5	0.22	0.00	25.53	0.00	253.08

FileIO test for QEMU:

Test 1: Random read and write

	avg	min	max	read	write
1	0.11	0.00	62.40	297.14	198.10
2	0.11	0.00	22.39	293.61	195.74
3	0.09	0.00	16.62	347.45	231.63
4	0.10	0.00	15.28	342.27	228.18
5	0.09	0.00	19.69	355.79	237.19

Test 2: Sequential write

	avg	min	max	read	write
1	0.08	0.00	195.72	0.00	643.62
2	0.08	0.00	24.63	0.00	687.59
3	0.08	0.00	17.88	0.00	686.43
4	0.08	0.00	811.77	0.00	643.26
5	0.08	0.00	18.95	0.00	661.17

Test 3: Random write

	avg	min	max	read	write
1	0.15	0.00	190.13	0.00	354.53
2	0.14	0.00	24.97	0.00	386.13
3	0.15	0.00	33.91	0.00	373.25
4	0.14	0.00	9.74	0.00	381.98
5	0.15	0.00	27.97	0.00	373.70

Observation and Analysis of the performance data:

From the above result data, we can see that QEMU has a better performance rather than docker in CPU tests and FileIO test. The number of events per second for Docker(OS Virtualization) Is lower than QEMU(System Virtualization) in CPU tests. And Docker also has the lower throughput of reading and writing and the higher latency than QEMU in FileIO test. This is pretty unusual because Docker which is OS Virtualization should perform better than QEMU, which is System Virtualization.

I think this is because native Docker is not available in my Apple Silicon laptop, I have to use Docker desktop instead of native Docker. Docker Desktop may add another layer which cause the low performance in my laptop.

Git Repository:
URL: https://github.com/kinghe233/Ruopuhe_COEN241.git