

# COEN 241 HW 1: System vs OS Virtualization

Xiongsheng Yi

## Host Machine

My Windows desktop:

Processor: AMD Ryzen 9 5900 12-Core Processor, 3.00 GHz

RAM: 32GB

System type: 64-bit operating system, x64-based processor

Windows edition: Windows 11 Home

Storage: 700GB free of 936GB

### System > About

XYr10  
Alienware Aurora Ryzen Edition

Rename this PC

i

Device specifications

Copy ^

Device name	XYr10	
Processor	AMD Ryzen 9 5900 12-Core Processor	3.00 GHz
Installed RAM	32.0 GB	

System type

64-bit operating system, x64-based processor

Pen and touch

No pen or touch input is available for this display

Related links

[Domain or workgroup](#)

[System protection](#)

[Advanced system settings](#)

Windows specifications

Copy ^

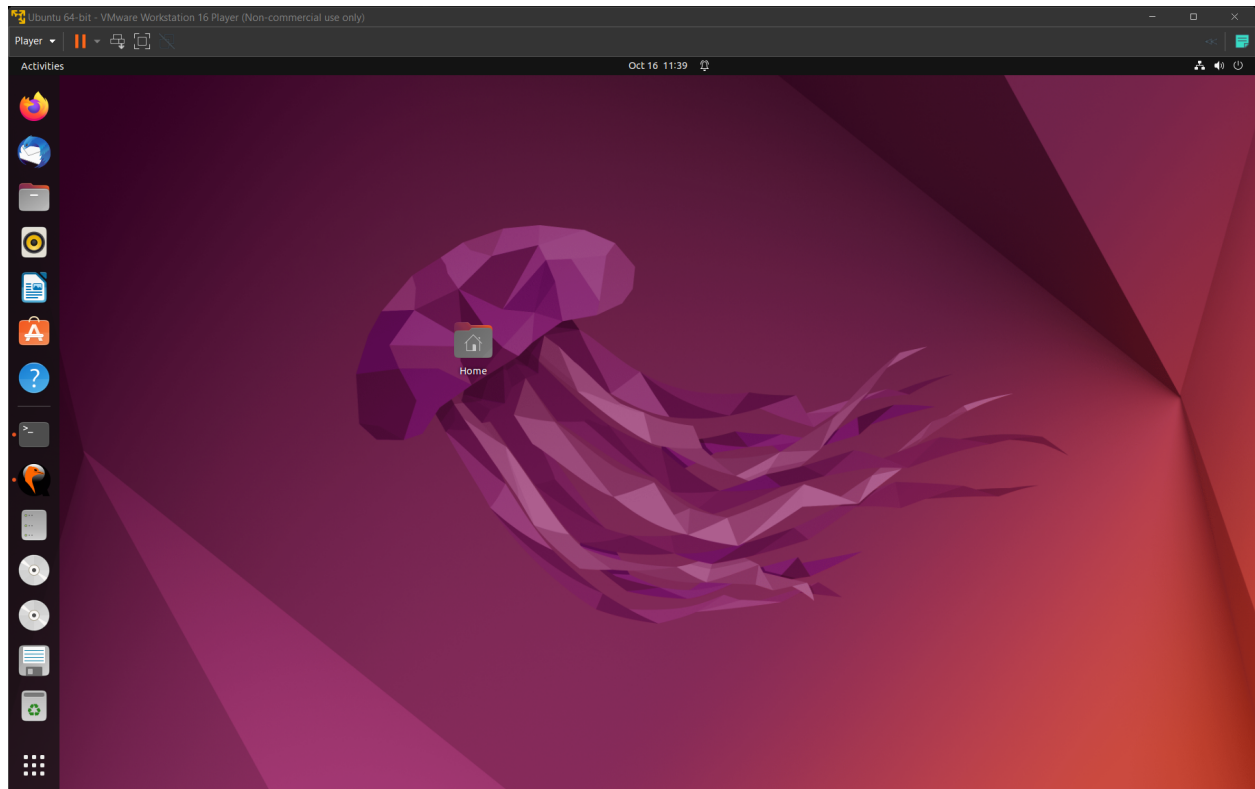
Edition	Windows 11 Home
Version	22H2
Installed on	10/11/2022
OS build	22621.674
Experience	Windows Feature Experience Pack 1000.22634.1000.0

[Microsoft Services Agreement](#)

[Microsoft Software License Terms](#)

## Linux Environment Setup:

Since I was using Windows, I used VMware to create a Linux environment. I downloaded and installed VMware for Windows computer, and created a new Linux OS with the latest Ubuntu 22.04.



Reference Links:

<https://www.vmware.com/products/workstation-player.html>

<https://ubuntu.com/download/desktop>

## QEMU Setup

1. Download the given Ubuntu server ISO image to local.

ubuntu-20.04.5-live-server-amd64.iso is downloaded. (For Linux)

2. Open the terminal and Install QEMU.

```
$ sudo apt-get install git
```

```
$ sudo apt-get install qemu
```

```
$ sudo apt-get install qemu-utils
```

```
$ sudo apt-get install qemu-system
```

3. Create a QEMU image. (10G disk space, image format of QEMU copy-on-write v2)

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

4. Then start the VM installation.

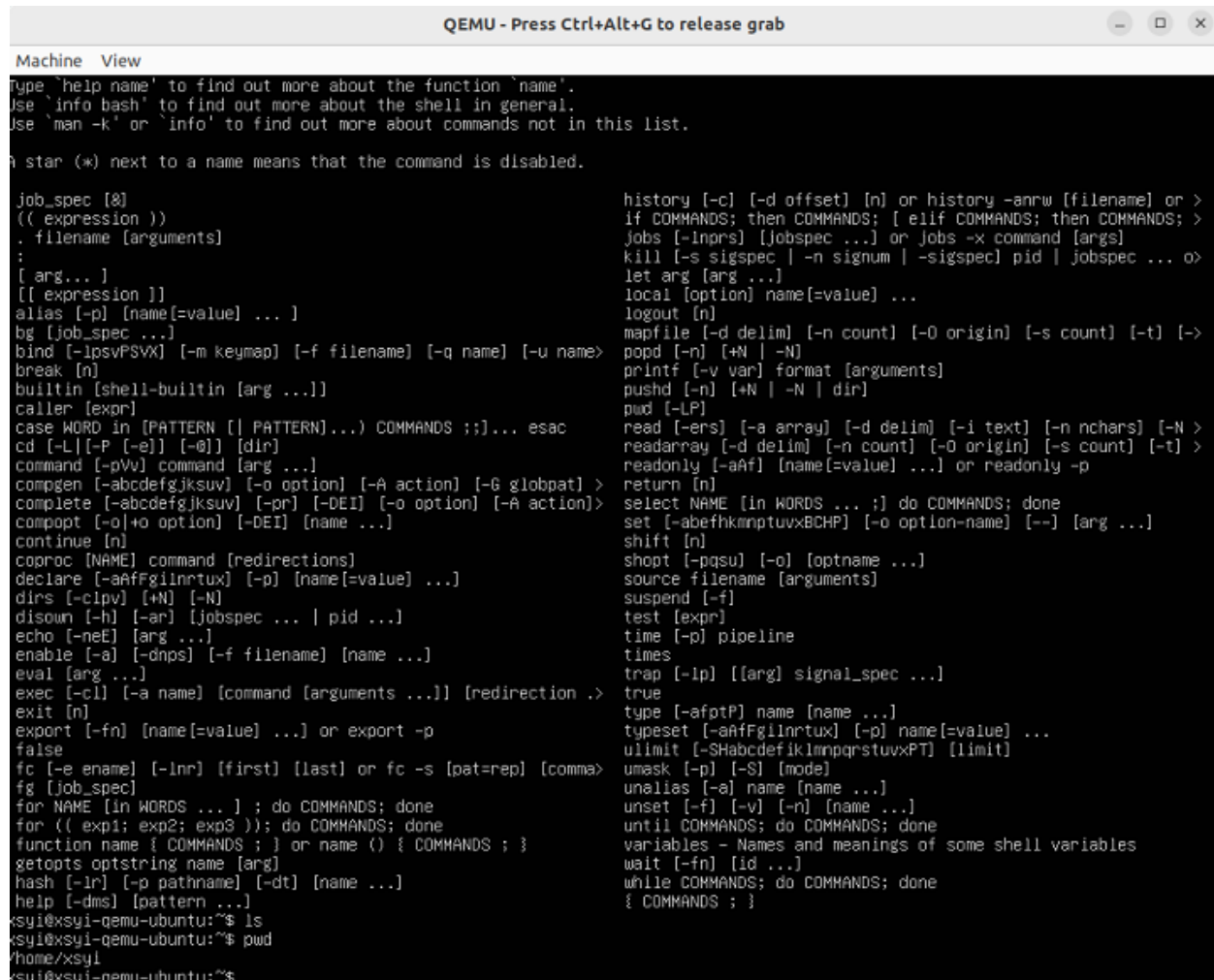
```
$ sudo qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom
```

```
./ubuntu-20.04.5-live-server-amd64.iso -m 2046 -boot strict=on
```

5. After the installation is done, boot the VM from the image ubuntu.img created.

```
$ qemu-system-x86_64 -hda ubuntu.img -boot d -m 2046 -boot
strict=on
```

6. Then login to the QEMU with username and password. You will see a new QEMU window.



```
Machine View
type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f filename] [-q name] [-u name]
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN] [PATTERN]...) COMMANDS ;;)... esac
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abcdefgjkshuv] [-o option] [-A action] [-G globpat]
complete [-abcdefgjkshuv] [-pr] [-DEI] [-o option] [-A action]
compopt [-o|+o option] [-DEI] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAffgIlntux] [-p] [name[=value] ...]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid ...]
echo [-neE] [arg ...]
enable [-a] [-dnps] [-f filename] [name ...]
eval [arg ...]
exec [-cl] [-a name] [command [arguments ...]] [redirection ...]
exit [n]
export [-fn] [name[=value] ...] or export -p
false
fc [-e ename] [-lnr] [first] [last] or fc -s [pat=rep] [comma]
fg [job_spec]
for NAME [in WORDS ... ] ; do COMMANDS; done
for (( exp1; exp2; exp3 )); do COMMANDS; done
function name { COMMANDS ; } or name () { COMMANDS ; }
getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt] [name ...]
help [-dms] [pattern ...]
xsyi@xsyi-qemu-ubuntu:~$ ls
xsyi@xsyi-qemu-ubuntu:~$ pwd
/home/xsyi
xsyi@xsyi-qemu-ubuntu:~$
```

QEMU version:

```
xsyi@xsyi-qemu-ubuntu:~$ qemu-system-x86_64 -version
QEMU emulator version 4.2.1 (Debian 1:4.2-3ubuntu6.23)
Copyright (c) 2003-2019 Fabrice Bellard and the QEMU Project developers
```

7. Install the sysbench in the QEMU.

```
$ sudo apt update
```

```
$ sudo apt install sysbench
```

```
QEMU - Press Ctrl+Alt+G to release grab

Machine View
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  liblua5.1-2 liblua5.1-common libmysqlclient21 libpq5 mysql-common
The following NEW packages will be installed:
  liblua5.1-2 liblua5.1-common libmysqlclient21 libpq5 mysql-common sysbench
0 upgraded, 6 newly installed, 0 to remove and 12 not upgraded.
Need to get 1827 kB of archives.
After this operation, 9267 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 liblua5.1-common all 2.1.0~beta3+dfsg-5.1build1 [44.3 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 liblua5.1-2 amd64 2.1.0~beta3+dfsg-5.1build1 [228 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal/main amd64 mysql-common all 5.8+1.0.5ubuntu2 [7496 B]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libmysqlclient21 amd64 8.0.30-0ubuntu0.20.04.2 [1323 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpq5 amd64 12.12-0ubuntu0.20.04.1 [117 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 sysbench amd64 1.0.18+ds-1 [107 kB]
Fetched 1827 kB in 7s (272 kB/s)
Selecting previously unselected package liblua5.1-common.
(Reading database ... 69179 files and directories currently installed.)
Preparing to unpack .../0-liblua5.1-common_2.1.0~beta3+dfsg-5.1build1_all.deb ...
Unpacking liblua5.1-common (2.1.0~beta3+dfsg-5.1build1) ...
Selecting previously unselected package liblua5.1-2:amd64.
Preparing to unpack .../1-liblua5.1-2_2.1.0~beta3+dfsg-5.1build1_amd64.deb ...
Unpacking liblua5.1-2:amd64 (2.1.0~beta3+dfsg-5.1build1) ...
Selecting previously unselected package mysql-common.
Preparing to unpack .../2-mysql-common_5.8+1.0.5ubuntu2_all.deb ...
Unpacking mysql-common (5.8+1.0.5ubuntu2) ...
Selecting previously unselected package libmysqlclient21:amd64.
Preparing to unpack .../3-libmysqlclient21_8.0.30-0ubuntu0.20.04.2_amd64.deb ...
Unpacking libmysqlclient21:amd64 (8.0.30-0ubuntu0.20.04.2) ...
Selecting previously unselected package libpq5:amd64.
Preparing to unpack .../4-libpq5_12.12-0ubuntu0.20.04.1_amd64.deb ...
Unpacking libpq5:amd64 (12.12-0ubuntu0.20.04.1) ...
Selecting previously unselected package sysbench.
Preparing to unpack .../5-sysbench_1.0.18+ds-1_amd64.deb ...
Unpacking sysbench (1.0.18+ds-1) ...
Setting up mysql-common (5.8+1.0.5ubuntu2) ...
update-alternatives: using /etc/mysql/my.cnf.fallback to provide /etc/mysql/my.cnf (my.cnf) in auto mode
Setting up libmysqlclient21:amd64 (8.0.30-0ubuntu0.20.04.2) ...
Setting up libpq5:amd64 (12.12-0ubuntu0.20.04.1) ...
Setting up liblua5.1-common (2.1.0~beta3+dfsg-5.1build1) ...
Setting up liblua5.1-2:amd64 (2.1.0~beta3+dfsg-5.1build1) ...
Setting up sysbench (1.0.18+ds-1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
xsyi@xsyi-qemu-ubuntu:~$ _
```

It is ready for testing now.

## QEMU sysbench version

```
xsyi@xsyi-qemu-ubuntu:~$ sysbench --version
sysbench 1.0.18
xsyi@xsyi-qemu-ubuntu:~$ _
```

## Docker container Setup

I used this tutorial to install the docker:

<https://docs.docker.com/engine/install/ubuntu/>

Verify that Docker is installed and running correctly

```
$ sudo systemctl start docker
```

```
$ sudo systemctl status docker
```



\$ docker --version	Display current version of docker.
\$ docker exec	Run a command in a running container.
\$ docker restart	Restart one or more containers.
\$ docker kill	Kill one or more containers.
\$ docker commit	Create a new image from the container image.
\$ docker push	Push an image or repository to a registry.
ctl + p + q	Go back to terminal from running container, but keep container running

## Proof of experiment

Try the first CPU performance comparison

```
$ sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
```

For QEMU:

```
xsyi@xsyi-qemu-ubuntu:~$ sysbench --test=cpu --cpu-max-prime=20000 --max-time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --max-time is deprecated, use --time instead
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:   294.81

General statistics:
  total time:          30.0027s
  total number of events: 8846

Latency (ms):
  min:                 3.18
  avg:                 3.39
  max:                 9.78
  95th percentile:    3.62
  sum:                 29959.83

Threads fairness:
  events (avg/stddev): 8846.0000/0.00
  execution time (avg/stddev): 29.9598/0.00
```

For docker:

```

xsyl@xsyl-VM:~$ docker run -it --cpus=2 -m 2G zyclonite/sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.20-6ef8a4d4d7 (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!

CPU speed:
  events per second: 1969.34

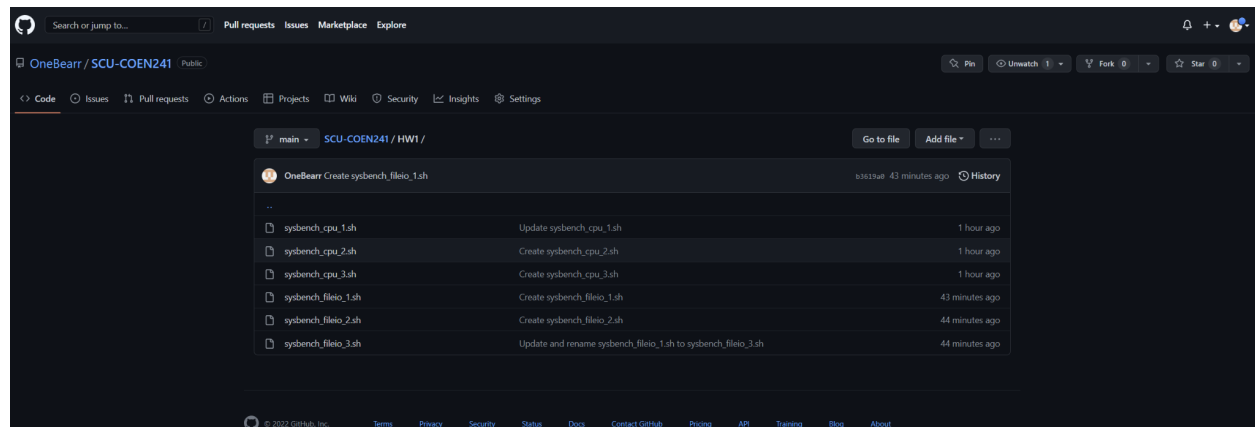
General statistics:
  total time:          30.0000s
  total number of events: 59081

Latency (ms):
  min:                 0.50
  avg:                 0.51
  max:                 2.98
  95th percentile:    0.54
  sum:                 29988.94

Threads fairness:
  events (avg/stddev): 59081.0000/0.00
  execution time (avg/stddev): 29.9889/0.00

```

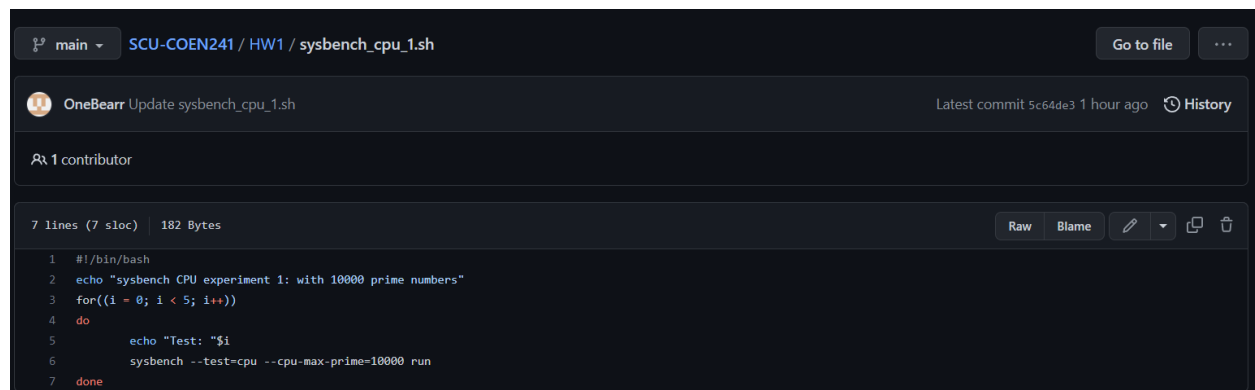
All the test cases shell scripts are saved in the Github repository



For CPU test cases:

I tested “cpu max prime” 10000, 20000, 30000

Each shell script is like: (repeat 5 times)

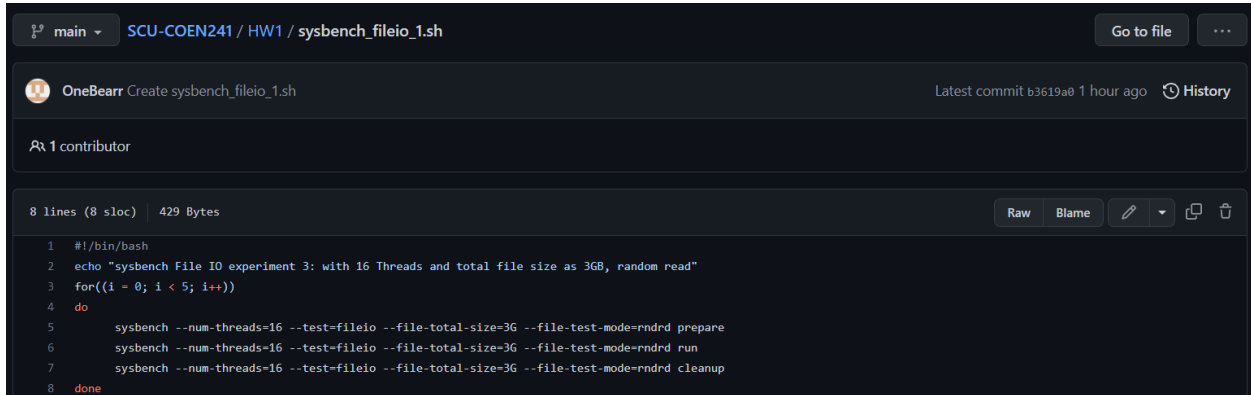


For File I/O test cases:

I tested with 16 Threads and total file size as 3GB,

“random read”, “random write” and “combined random read/write”

Each shell script is like:(repeat 5 times)



```
1  #!/bin/bash
2  echo "sysbench File IO experiment 3: with 16 Threads and total file size as 3GB, random read"
3  for((i = 0; i < 5; i++))
4  do
5      sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrd prepare
6      sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrd run
7      sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrd cleanup
8  done
```

The previous test result will speed up the following test from the cache, which is unfair and inaccurate.

Thus, prepare -> run -> cleanup

## For QEMU:

Installations:

\$ apt update

\$ apt install sysbench

\$ apt install git

\$ apt install vim

Use “git clone https://github.com/OneBarr/SCU-COEN241.git” to pull the repository to the local VM.

Give the files permission to be executed by

\$ chmod 755 file\_name.sh

Run each shell script and save results by

\$ ./file\_name.sh > output\_file\_name.txt

To open and read the output files by

\$ vim output\_file\_name.txt

After running all the test case files,



```

xsysi@xsysi-qemu2:~/SCU-COEN241/HW1$ ls
output_cpu1.txt  output_cpu3.txt  output_fileio2.txt  sysbench_cpu_1.sh  sysbench_cpu_3.sh  sysbench_fileio_2.sh
output_cpu2.txt  output_fileio1.txt  output_fileio3.txt  sysbench_cpu_2.sh  sysbench_fileio_1.sh  sysbench_fileio_3.sh
xsysi@xsysi-qemu2:~/SCU-COEN241/HW1$ _

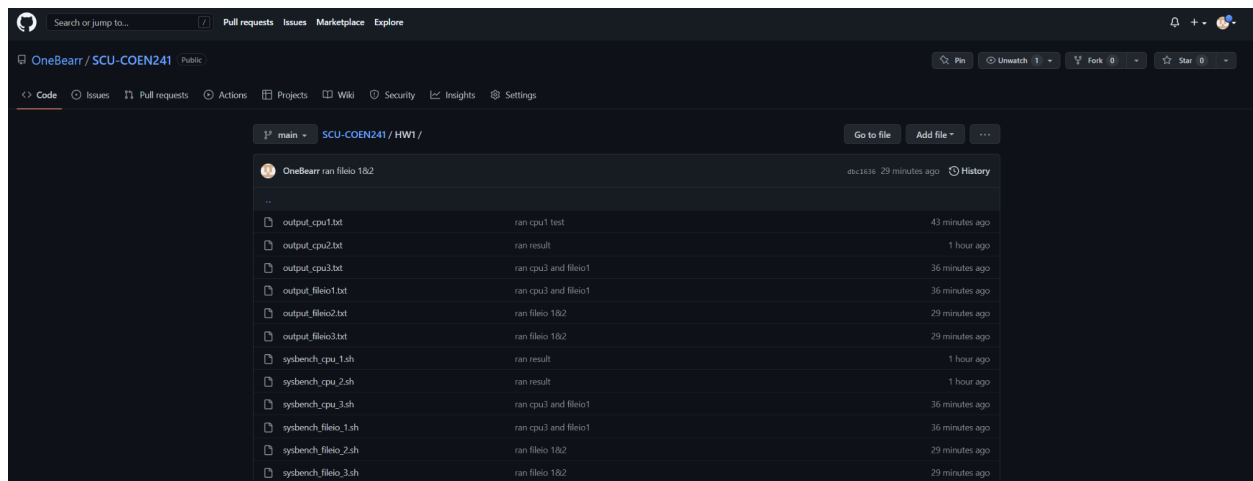
```

After all the Github authorization setup(ssh, etc.), I pushed all the output files to my github,

```
$ git add
```

```
$ git commit
```

```
$ git push origin main
```



Ref: <https://github.com/OneBarr/SCU-COEN241/tree/main/HW1>

For cpu1 with 10000 prime: (Average of 5 tests)

CPU speed:

events per second: 750

General statistics:

total time: 10s

total number of events: 7500

For cpu2 with 20000 prime: (Average of 5 tests)

CPU speed:

events per second: 295

General statistics:

total time: 10s

total number of events: 2950

For cpu3 with 30000 prime: (Average of 5 tests)

CPU speed:

events per second: 172

General statistics:

total time:	10s
total number of events:	1720

### Analysis:

CPU1 has the highest CPU speed with most events per sec, CPU3 is the lowest.

For fileio1 with 16 Threads and total file size as 1GB, random read: (Average of 5 tests)

#### Throughput:

read, MiB/s:	1250
written, MiB/s:	0.00

#### General statistics:

total time:	10s
total number of events:	800000

#### Threads fairness:

events (avg/stddev):	49078.3125/585.55
execution time (avg/stddev):	9.2867/0.17

For fileio2 with 16 Threads and total file size as 1GB, random write: (Average of 5 tests)

#### Throughput:

read, MiB/s:	0.00
written, MiB/s:	22.98

#### General statistics:

total time:	10s
total number of events:	35000

#### Threads fairness:

events (avg/stddev):	2189.3750/272.67
execution time (avg/stddev):	9.9626/0.02

For fileio3 with 16 Threads and total file size as 1GB, combined random read/write: (Average of 5 tests)

#### Throughput:

read, MiB/s:	20
written, MiB/s:	13

#### General statistics:

total time:	10s
total number of events:	50000

#### Threads fairness:

events (avg/stddev):	3038.0625/274.21
----------------------	------------------

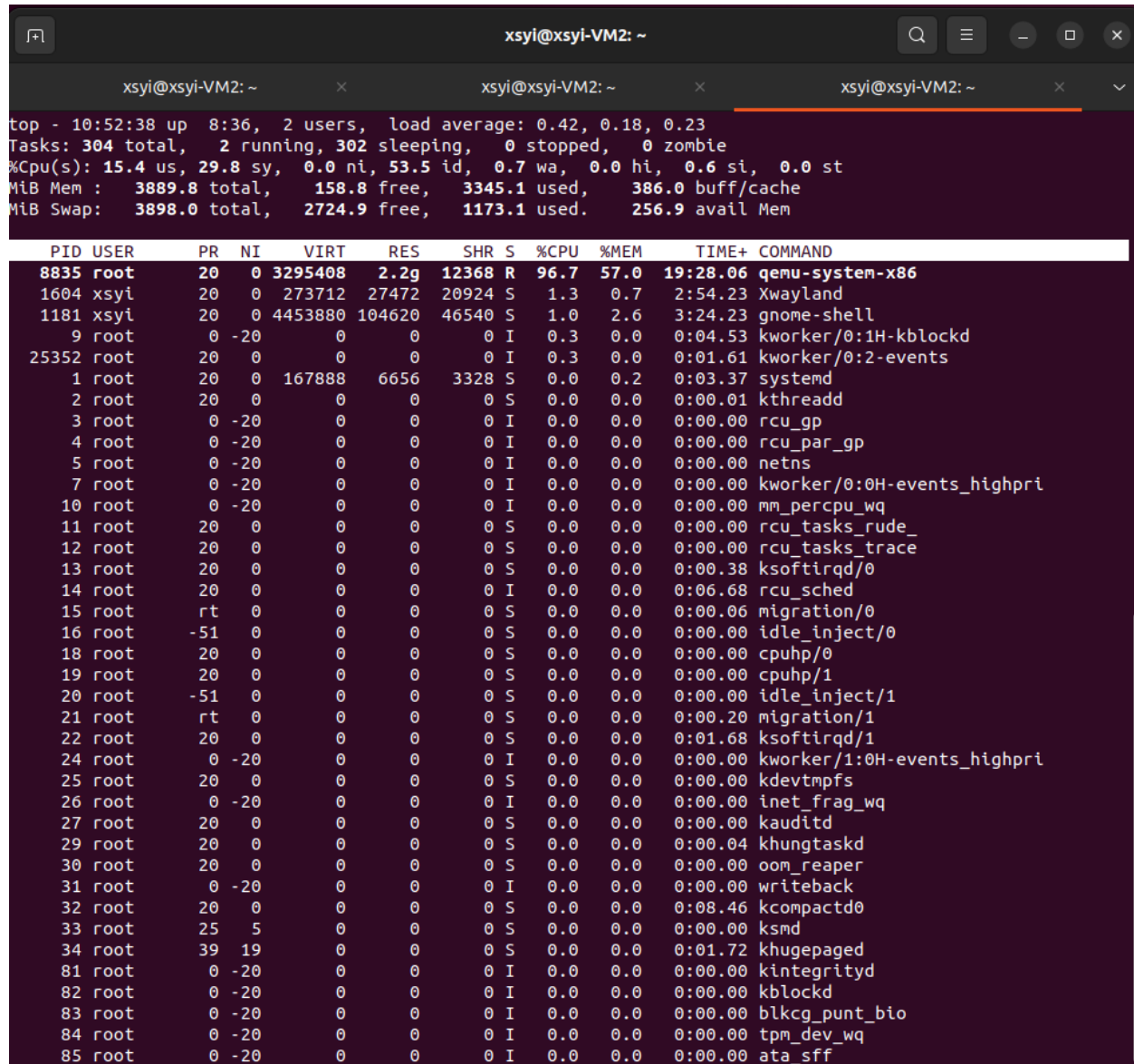
execution time (avg/stddev): 9.9558/0.02

## Analysis:

File read is much faster than file write relatively, the throughput is about 50 times faster.

The CPU utilization when running tests:

About 96-97%



```
top - 10:52:38 up 8:36, 2 users, load average: 0.42, 0.18, 0.23
Tasks: 304 total, 2 running, 302 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.4 us, 29.8 sy, 0.0 ni, 53.5 id, 0.7 wa, 0.0 hi, 0.6 si, 0.0 st
MiB Mem : 3889.8 total, 158.8 free, 3345.1 used, 386.0 buff/cache
MiB Swap: 3898.0 total, 2724.9 free, 1173.1 used. 256.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8835	root	20	0	3295408	2.2g	12368	R	96.7	57.0	19:28.06	qemu-system-x86
1604	xsysi	20	0	273712	27472	20924	S	1.3	0.7	2:54.23	Xwayland
1181	xsysi	20	0	4453880	104620	46540	S	1.0	2.6	3:24.23	gnome-shell
9	root	0	-20	0	0	0	I	0.3	0.0	0:04.53	kworker/0:1H-kblockd
25352	root	20	0	0	0	0	I	0.3	0.0	0:01.61	kworker/0:2-events
1	root	20	0	167888	6656	3328	S	0.0	0.2	0:03.37	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
13	root	20	0	0	0	0	S	0.0	0.0	0:00.38	ksoftirqd/0
14	root	20	0	0	0	0	I	0.0	0.0	0:06.68	rcu_sched
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.06	migration/0
16	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
20	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.20	migration/1
22	root	20	0	0	0	0	S	0.0	0.0	0:01.68	ksoftirqd/1
24	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-events_highpri
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_frag_wq
27	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
29	root	20	0	0	0	0	S	0.0	0.0	0:00.04	khungtaskd
30	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
32	root	20	0	0	0	0	S	0.0	0.0	0:08.46	kcompactd0
33	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
34	root	39	19	0	0	0	S	0.0	0.0	0:01.72	khugepaged
81	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrityd
82	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
83	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	blkcg_punt_bio
84	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	tpm_dev_wq
85	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff

## For Docker:

Docker preparation,

\$ docker start

\$ docker pull ubuntu

\$ docker run -ti ubuntu

Installations:

apt update

apt install sysbench

apt install git

apt install vim

Use “git clone <https://github.com/OneBearr/SCU-COEN241.git>” to pull the repository to the local VM.

Give the files permission to be executed by

\$ chmod +x \*.sh

Run each shell script and save results by

\$ ./file\_name.sh > output\_file\_name.txt

To open and read the output files by

\$ vim output\_file\_name.txt

After running all the test case files,

```
root@eafa0fdcb229:/SCU-COEN241/HW1# ls
output_cpu1.txt      output_cpu2_docker.txt  output_fileio1.txt      output_fileio2_docker.txt  sysbench_cpu_1.sh  sysbench_fileio_1.sh
output_cpu1_docker.txt  output_cpu3.txt        output_fileio1_docker.txt  output_fileio3.txt        sysbench_cpu_2.sh  sysbench_fileio_2.sh
output_cpu2.txt      output_cpu3_docker.txt  output_fileio2.txt      output_fileio3_docker.txt  sysbench_cpu_3.sh  sysbench_fileio_3.sh
```

After all the Github authorization setup(ssh, etc.), I pushed all the output files to my github,

\$ git add

\$ git commit

\$ git push origin main

The screenshot shows a GitHub repository page for 'OneBarr / SCU-COEN241'. The 'main' branch is selected, and the file path is 'SCU-COEN241 / HW1'. A table titled 'OneBarr ran tests' displays the results of various tests. The table has three columns: a file icon, the test name, and the time since the test was run. The tests include 'output\_cpu1.txt', 'output\_cpu1\_docker.txt', 'output\_cpu2.txt', 'output\_cpu2\_docker.txt', 'output\_cpu3.txt', 'output\_cpu3\_docker.txt', 'output\_fileio1.txt', 'output\_fileio1\_docker.txt', 'output\_fileio2.txt', 'output\_fileio2\_docker.txt', 'output\_fileio3.txt', 'output\_fileio3\_docker.txt', 'sysbench\_cpu\_1.sh', 'sysbench\_cpu\_2.sh', 'sysbench\_cpu\_3.sh', 'sysbench\_fileio\_1.sh', 'sysbench\_fileio\_2.sh', and 'sysbench\_fileio\_3.sh'.

File	Test Name	Time
..	..	..
output_cpu1.txt	ran cpu1 test	2 hours ago
output_cpu1_docker.txt	ran tests	20 minutes ago
output_cpu2.txt	ran result	2 hours ago
output_cpu2_docker.txt	ran tests	20 minutes ago
output_cpu3.txt	ran cpu3 and fileio1	2 hours ago
output_cpu3_docker.txt	ran tests	20 minutes ago
output_fileio1.txt	ran cpu3 and fileio1	2 hours ago
output_fileio1_docker.txt	ran tests	20 minutes ago
output_fileio2.txt	ran fileio 1&2	2 hours ago
output_fileio2_docker.txt	ran tests	20 minutes ago
output_fileio3.txt	ran fileio 1&2	2 hours ago
output_fileio3_docker.txt	ran tests	20 minutes ago
sysbench_cpu_1.sh	ran result	2 hours ago
sysbench_cpu_2.sh	ran result	2 hours ago
sysbench_cpu_3.sh	ran cpu3 and fileio1	2 hours ago
sysbench_fileio_1.sh	ran cpu3 and fileio1	2 hours ago
sysbench_fileio_2.sh	ran fileio 1&2	2 hours ago
sysbench_fileio_3.sh	ran fileio 1&2	2 hours ago

Ref: <https://github.com/OneBarr/SCU-COEN241/tree/main/HW1>

For cpu1 with 10000 prime: (Average of 5 tests)

CPU speed:

events per second: 5300

General statistics:

total time: 10s

total number of events: 53000

For cpu2 with 20000 prime: (Average of 5 tests)

CPU speed:

events per second: 2080

General statistics:

total time: 10s

total number of events: 20800

For cpu3 with 30000 prime: (Average of 5 tests)

CPU speed:

events per second: 1200

General statistics:

total time: 10s

total number of events: 12000

## Analysis:

CPU1 has the highest CPU speed with most events per sec, CPU3 is the lowest.

For fileio1 with 16 Threads and total file size as 1GB, random read: (Average of 5 tests)

Throughput:

read, MiB/s:	15000
written, MiB/s:	0.00

General statistics:

total time:	10s
total number of events:	10200000

Threads fairness:

events (avg/stddev):	616737.9375/23978.66
execution time (avg/stddev):	9.3833/0.15

For fileio2 with 16 Threads and total file size as 1GB, random write: (Average of 5 tests)

Throughput:

read, MiB/s:	0.00
written, MiB/s:	315.90

General statistics:

total time:	10s
total number of events:	461015

Threads fairness:

events (avg/stddev):	28813.4375/2757.16
execution time (avg/stddev):	9.9626/0.02

For fileio3 with 16 Threads and total file size as 1GB, combined random read/write: (Average of 5 tests)

Throughput:

read, MiB/s:	424.99
written, MiB/s:	283.33

General statistics:

total time:	10s
total number of events:	1033712

Threads fairness:

events (avg/stddev):	64607.0000/6649.11
execution time (avg/stddev):	9.9558/0.02

### **Analysis:**

File read is much faster than file write relatively, the throughput is about 50 times faster.

The CPU utilization when running tests:

80%-90% varies.

```
xsysi@xsysi-VM2: ~  
top - 12:13:43 up 9:57, 4 users, load average: 7.90, 6.51, 3.27  
Tasks: 323 total, 2 running, 321 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 32.1 us, 2.3 sy, 0.0 ni, 65.4 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st  
MiB Mem : 3889.8 total, 1137.5 free, 1854.5 used, 897.8 buff/cache  
MiB Swap: 3898.0 total, 1247.4 free, 2650.6 used, 1738.0 avail Mem  
  
  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND  
 31244 root        20   0 31472  9664  7672  S   80.1   0.2   0:02.41 sysbench  
 1181 xsysi       20   0 4440156 100416 39540  R   2.7   2.5   3:54.91 gnome-shell  
 2552 xsysi       20   0 581700  35248 23744  S   1.3   0.9   0:17.21 gnome-terminal-  
 8835 root        20   0 3295408 753932 9232  S   1.0  18.9  20:53.64 qemu-system-x86  
 30460 root        20   0   0     0     0  I   0.3   0.0   0:00.26 kworker/0:2-events  
    1 root        20   0 168028  8512  4080  S   0.0   0.2   0:04.47 systemd  
    2 root        20   0   0     0     0  S   0.0   0.0   0:00.01 kthreadd  
    3 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 rcu_gp  
    4 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 rcu_par_gp  
    5 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 netns  
    7 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri  
    9 root        20  -20   0     0     0  I   0.0   0.0   0:00.55 kworker/0:1H-events_highpri  
   10 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 mm_percpu_wq  
   11 root        20   0   0     0     0  S   0.0   0.0   0:00.00 rcu_tasks_rude_  
   12 root        20   0   0     0     0  S   0.0   0.0   0:00.00 rcu_tasks_trace  
   13 root        20   0   0     0     0  S   0.0   0.0   0:00.62 ksoftirqd/0  
   14 root        20   0   0     0     0  I   0.0   0.0   0:07.62 rcu_sched  
   15 root        20   0   0     0     0  S   0.0   0.0   0:00.07 migration/0  
   16 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 idle_inject/0  
   18 root        20   0   0     0     0  S   0.0   0.0   0:00.00 cpuhp/0  
   19 root        20   0   0     0     0  S   0.0   0.0   0:00.00 cpuhp/1  
   20 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 idle_inject/1  
   21 root        20   0   0     0     0  S   0.0   0.0   0:00.21 migration/1  
   22 root        20   0   0     0     0  S   0.0   0.0   0:02.34 ksoftirqd/1  
   24 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri  
   25 root        20   0   0     0     0  S   0.0   0.0   0:00.00 kdevtmpfs  
   26 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 inet_frag_wq  
   27 root        20   0   0     0     0  S   0.0   0.0   0:00.00 kauditd  
   29 root        20   0   0     0     0  S   0.0   0.0   0:00.04 khungtaskd  
   30 root        20   0   0     0     0  S   0.0   0.0   0:00.00 oom_reaper  
   31 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 writeback  
   32 root        20   0   0     0     0  S   0.0   0.0   0:00.37 kcompactd0  
   33 root        25   5   0     0     0  S   0.0   0.0   0:00.00 ksmd  
   34 root        39  19   0     0     0  S   0.0   0.0   0:01.88 khugepaged  
   81 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 kintegrityd  
   82 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 kblockd  
   83 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 blkcg_punt_bio  
   84 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 tpm_dev_wq  
   85 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 ata_sff  
   86 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 md  
   87 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 edac-poller  
   88 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 devfreq_wq  
   89 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 watchdogd  
   92 root        20   0   0     0     0  S   0.0   0.0   0:35.21 kswapd0  
   93 root        20   0   0     0     0  S   0.0   0.0   0:00.00 ecryptfs-kthrea  
   95 root        20  -20   0     0     0  I   0.0   0.0   0:00.00 kthrotld  
   96 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 irq/24-pciehp  
   97 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 irq/25-pciehp  
   98 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 irq/26-pciehp  
   99 root       -51   0   0     0     0  S   0.0   0.0   0:00.00 irq/27-pciehp
```

## QEMU vs Docker container Performance Comparisons

Apparently, docker containers are much faster than the QEMU.

For cpu tests,

The events per second is 7-8 times faster.

For fileio tests,

The throughputs are about 12-15 times faster in MiB/s.

Git Repository:

<https://github.com/OneBarr/SCU-COEN241>