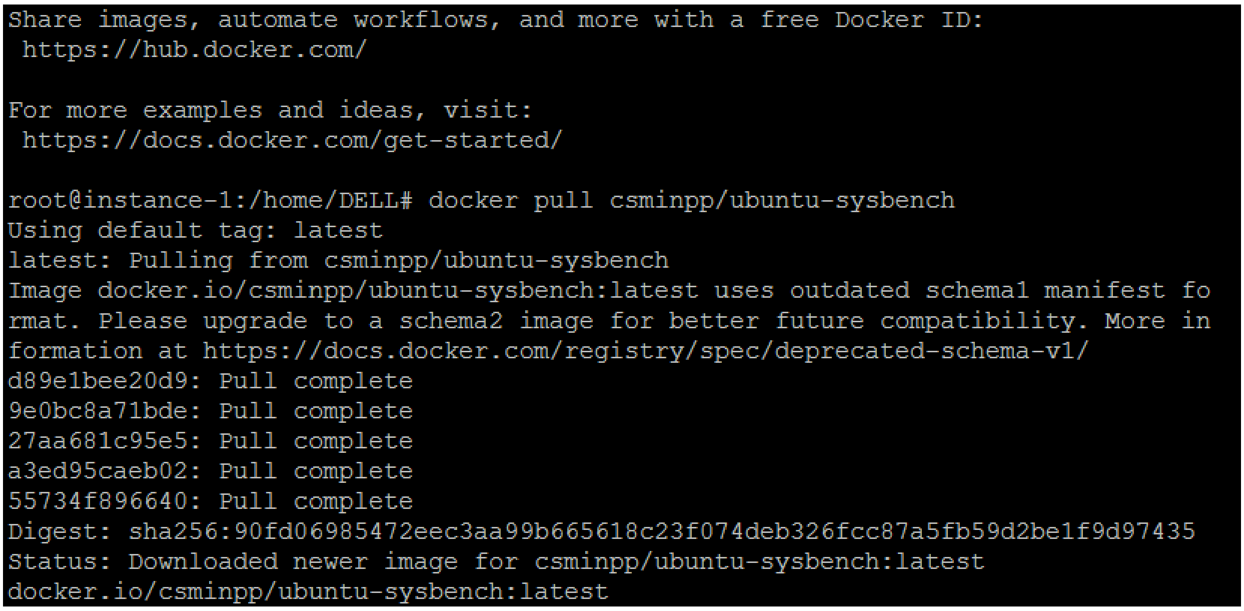
CS580K: Adv. Topics In Cloud Computing

Mini Project-1

Name : Prathamesh N Lonkar
Bnumber: B00811727

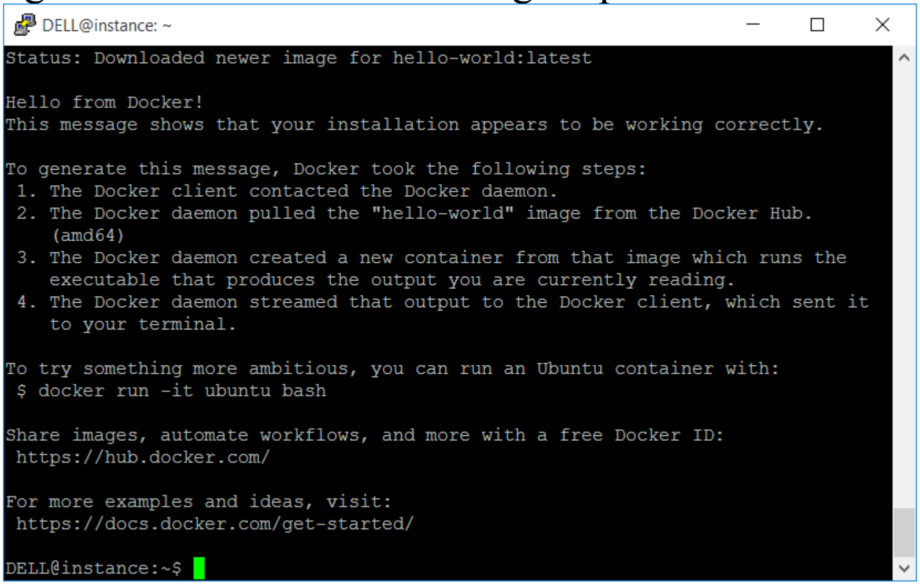
Performance Measurement In Docker: CPU utilization and fileio test.

- The Docker container is created using the steps given in the project-1 document and the Docker manual using the pre-installed docker image csminpp/ubuntu-sysbench.
- All the tests were conducted using the Sysbench Benchmark and the iostat tool of the sysstat library, where the kernel level reading were displayed by the sysbench results and the user level reading were displayed by the iostat log which I created.
- The following Image show that proper installation of the docker container was done.



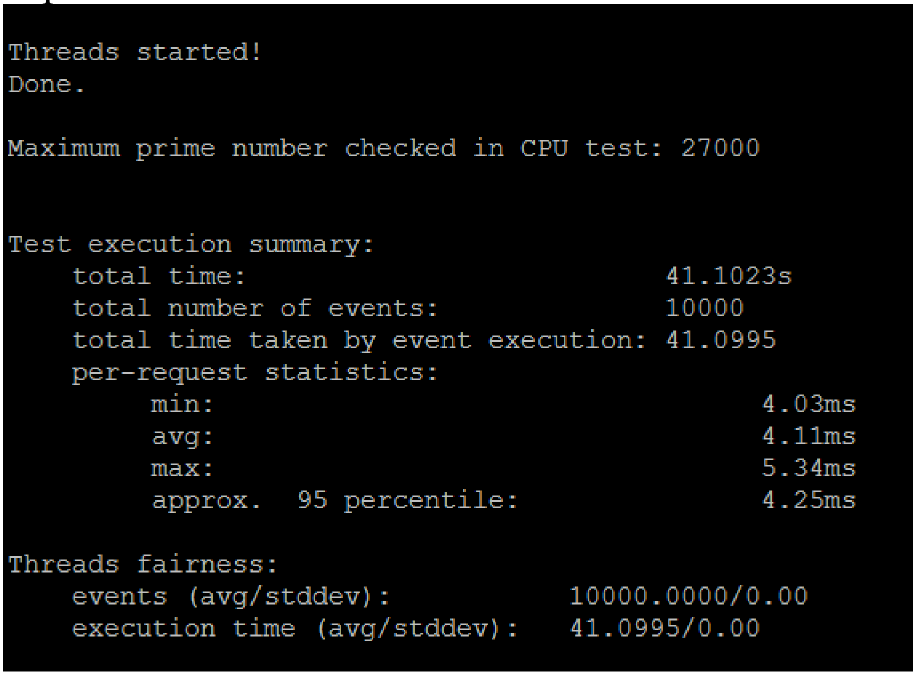
```
Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

root@instance-1:/home/DELL# docker pull csminpp/ubuntu-sysbench
Using default tag: latest
latest: Pulling from csminpp/ubuntu-sysbench
Image docker.io/csminpp/ubuntu-sysbench:latest uses outdated schema1 manifest fo
rmat. Please upgrade to a schema2 image for better future compatibility. More in
formation at https://docs.docker.com/registry/spec/deprecated-schema-v1/
d89e1bee20d9: Pull complete
9e0bc8a71bde: Pull complete
27aa681c95e5: Pull complete
a3ed95caeb02: Pull complete
55734f896640: Pull complete
Digest: sha256:90fd06985472eec3aa99b665618c23f074deb326fcc87a5fb59d2be1f9d97435
Status: Downloaded newer image for csminpp/ubuntu-sysbench:latest
docker.io/csminpp/ubuntu-sysbench:latest
```

- After the installation was completed I run the basic ‘Hello World’ program in the container to check if the container was up and running and as seen in the following output it was a sucess.



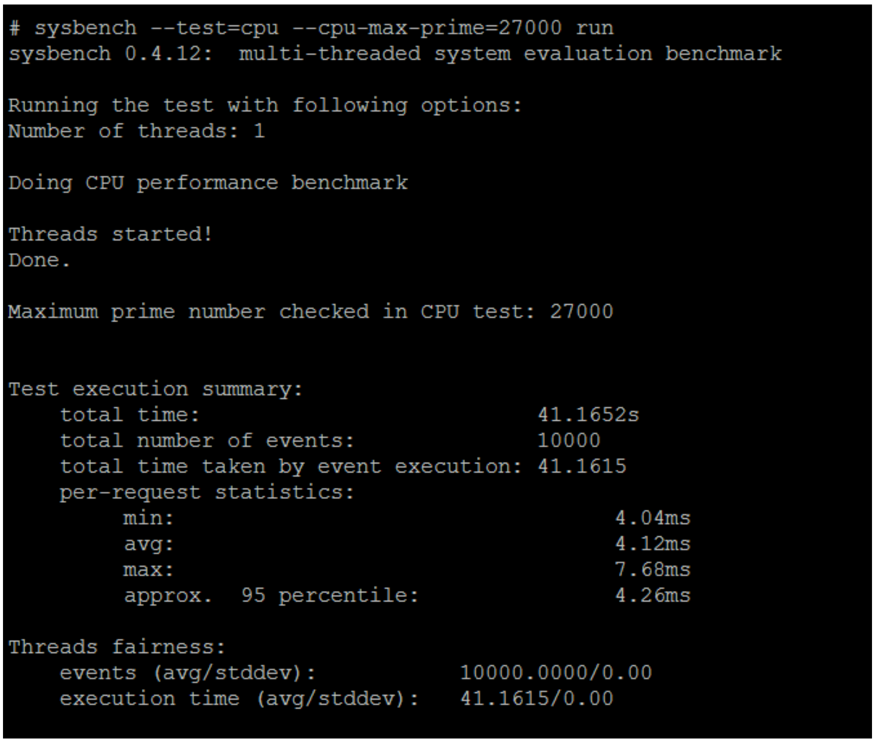
```
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

DELL@instance:~$
```

- After all the installations were complete I started the docker container and performed a sysbench on a cpu intensive operation which is -cpu-max-prime- where after several iterations I decided on the amount of prime number to be checked which is 27000 as it takes more than 30s finish.
- The output of the first test were as follows:

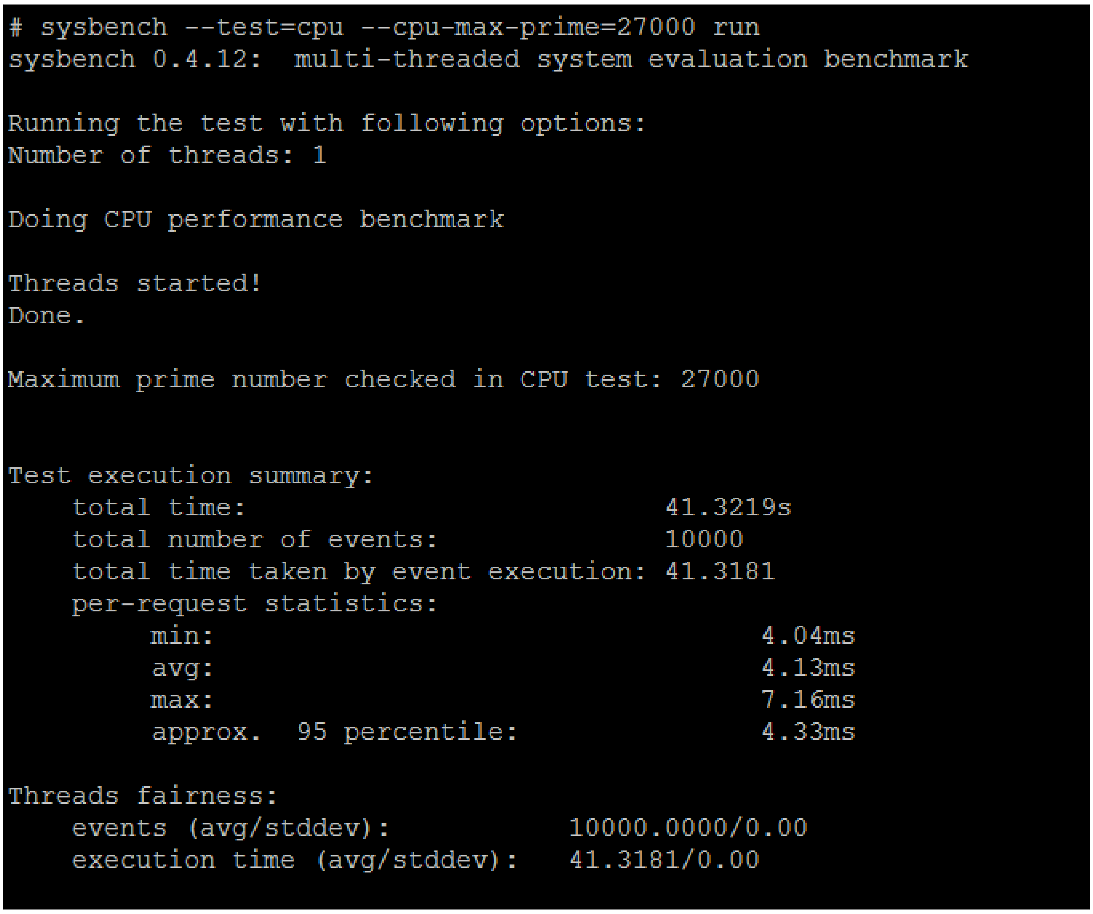


```
Threads started!
Done.

Maximum prime number checked in CPU test: 27000


Test execution summary:
    total time:                          41.1023s
    total number of events:              10000
    total time taken by event execution: 41.0995
    per-request statistics:
         min:                                  4.03ms
         avg:                                  4.11ms
         max:                                  5.34ms
         approx.  95 percentile:               4.25ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   41.0995/0.00
```

- I also performed the iostat test simultaneously and the log file are attached in the folder itself.(check for cpu.txt in iostat_test_report folder).
- The sysbench performance test were conducted additional two time and the results are as follows:

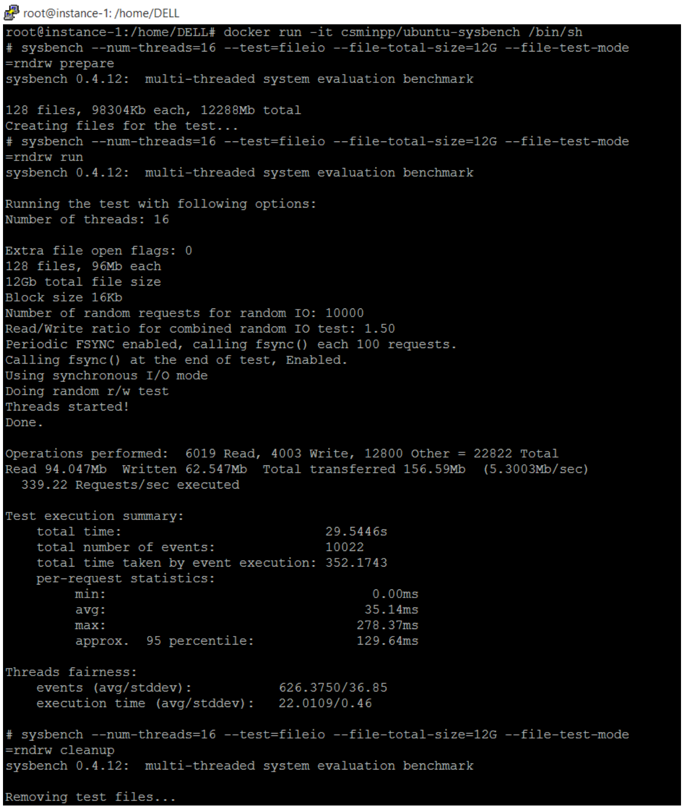


```
# sysbench --test=cpu --cpu-max-prime=27000 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 27000


Test execution summary:
    total time:                          41.1652s
    total number of events:              10000
    total time taken by event execution: 41.1615
    per-request statistics:
         min:                                  4.04ms
         avg:                                  4.12ms
         max:                                  7.68ms
         approx.  95 percentile:               4.26ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   41.1615/0.00
```

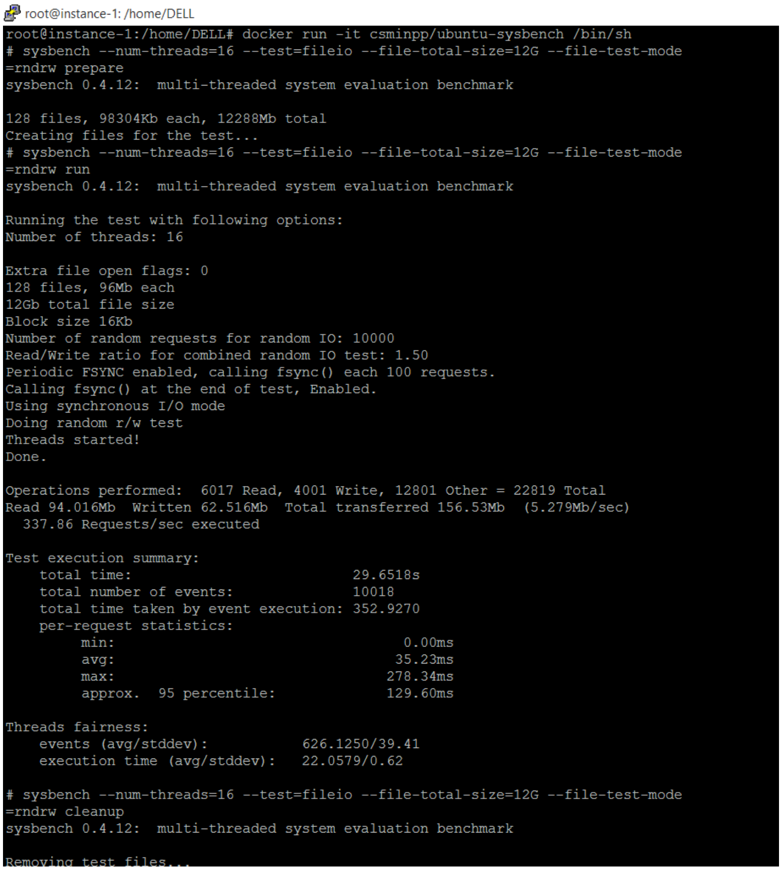


```
# sysbench --test=cpu --cpu-max-prime=27000 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 27000


Test execution summary:
    total time:                          41.3219s
    total number of events:              10000
    total time taken by event execution: 41.3181
    per-request statistics:
         min:                                  4.04ms
         avg:                                  4.13ms
         max:                                  7.16ms
         approx.  95 percentile:               4.33ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   41.3181/0.00
```

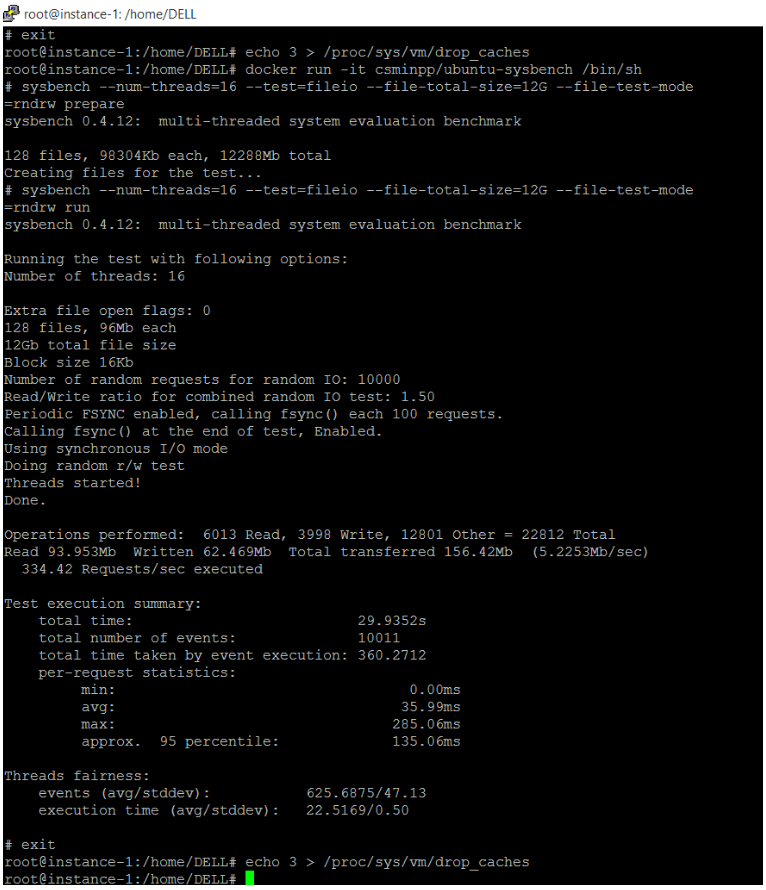
- The iostat log for the above above two iterations are attached in the file.
- I also conducted the Sysbench performance test for Fileio intensive operation, with 12G of files as it gives exact 30s to compute the test the results are as follows:



```
root@instance-1:/home/DELL# docker run -it csminpp/ubuntu-sysbench /bin/sh
# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw prepare
sysbench 0.4.12:  multi-threaded system evaluation benchmark

128 files, 98304Kb each, 12288Mb total
Creating files for the test...
# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 96Mb each
12Gb total file size
Block size 16Kb
Number of random requests for random IO: 10000
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 r/w test
Threads started!
Done.

Operations performed:  6019 Read, 4003 Write, 12800 Other = 22822 Total
Read 94.047Mb  Written 62.547Mb  Total transferred 156.59Mb  (5.3003Mb/sec)
  339.22 Requests/sec executed

Test execution summary:
    total time:                          29.5446s
    total number of events:              10022
    total time taken by event execution: 352.1743
    per-request statistics:
         min:                                  0.00ms
         avg:                                 35.14ms
         max:                                278.37ms
         approx.  95 percentile:             129.64ms

Threads fairness:
    events (avg/stddev):           626.3750/36.85
    execution time (avg/stddev):   22.0109/0.46

# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw cleanup
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Removing test files...
```

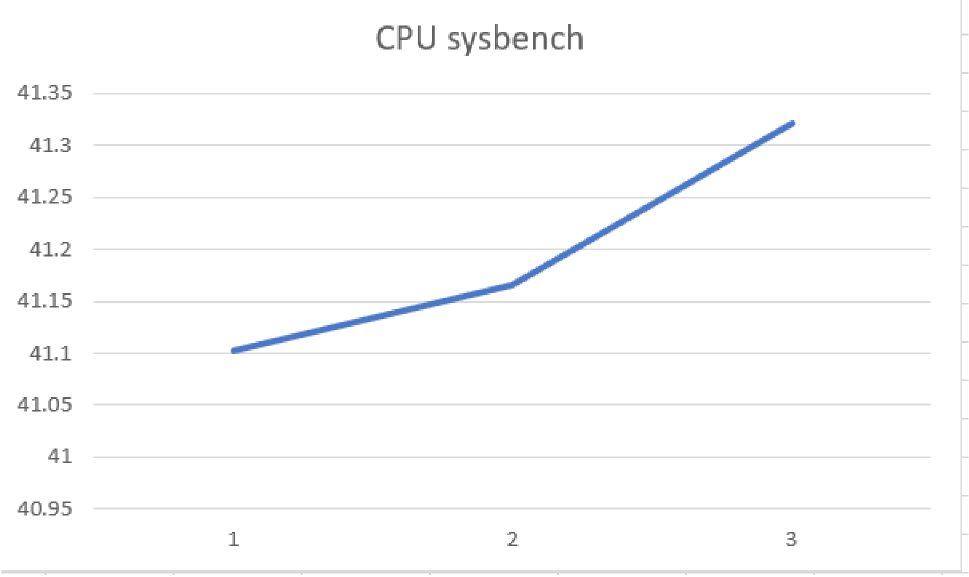


```
root@instance-1:/home/DELL# docker run -it csminpp/ubuntu-sysbench /bin/sh
# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw prepare
sysbench 0.4.12:  multi-threaded system evaluation benchmark

128 files, 98304Kb each, 12288Mb total
Creating files for the test...
# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 96Mb each
12Gb total file size
Block size 16Kb
Number of random requests for random IO: 10000
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 r/w test
Threads started!
Done.

Operations performed:  6017 Read, 4001 Write, 12801 Other = 22819 Total
Read 94.016Mb  Written 62.516Mb  Total transferred 156.53Mb  (5.279Mb/sec)
  337.86 Requests/sec executed

Test execution summary:
    total time:                          29.6518s
    total number of events:              10018
    total time taken by event execution: 352.9270
    per-request statistics:
         min:                                  0.00ms
         avg:                                 35.23ms
         max:                                278.34ms
         approx.  95 percentile:             129.60ms

Threads fairness:
    events (avg/stddev):           626.1250/39.41
    execution time (avg/stddev):   22.0579/0.62

# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw cleanup
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Removing test files...
```



```
# exit
root@instance-1:/home/DELL# echo 3 > /proc/sys/vm/drop_caches
root@instance-1:/home/DELL# docker run -it csminpp/ubuntu-sysbench /bin/sh
# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw prepare
sysbench 0.4.12:  multi-threaded system evaluation benchmark

128 files, 98304Kb each, 12288Mb total
Creating files for the test...
# sysbench --num-threads=16 --test=fileio --file-total-size=12G --file-test-mode
=rndrw run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 16

Extra file open flags: 0
128 files, 96Mb each
12Gb total file size
Block size 16Kb
Number of random requests for random IO: 10000
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 r/w test
Threads started!
Done.

Operations performed:  6013 Read, 3998 Write, 12801 Other = 22812 Total
Read 93.953Mb  Written 62.469Mb  Total transferred 156.42Mb  (5.2253Mb/sec)
  334.42 Requests/sec executed

Test execution summary:
    total time:                          29.9352s
    total number of events:              10011
    total time taken by event execution: 360.2712
    per-request statistics:
         min:                                  0.00ms
         avg:                                 35.99ms
         max:                                285.06ms
         approx.  95 percentile:             135.06ms

Threads fairness:
    events (avg/stddev):           625.6875/47.13
    execution time (avg/stddev):   22.5169/0.50

# exit
root@instance-1:/home/DELL# echo 3 > /proc/sys/vm/drop_caches
root@instance-1:/home/DELL# 
```

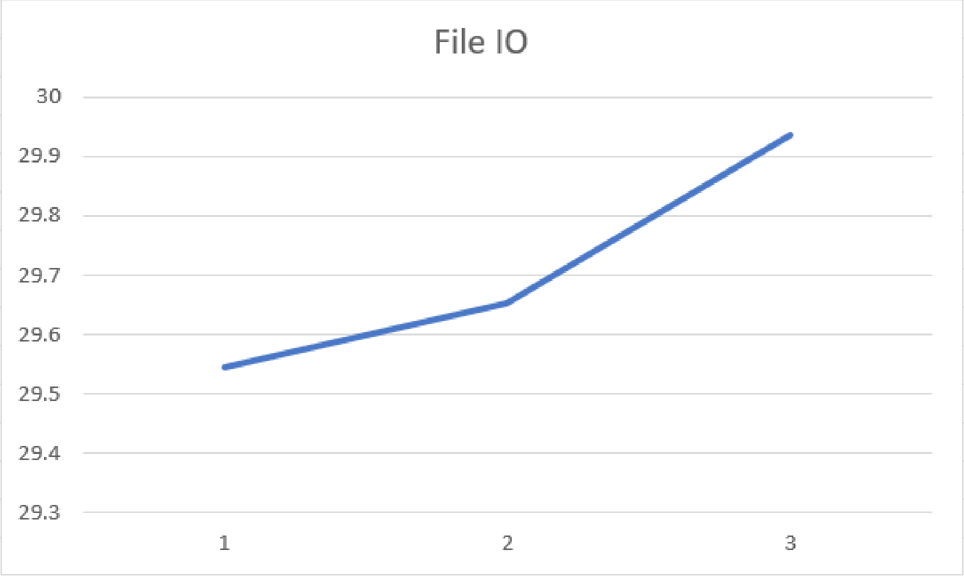
- The sysbench benchmark time taken to run the cpu intensive operation is plotted into a line graph and shown below.



- Here we can see that the total time taken to run the benchmark is increasing exponentially.
- The table showing the sysbench results is shown below.

Cpu Intensive operation	**Total Time**
Test 1	41.1023s
Test 2	41.1652s
Test 3	41.3219s
Avg:	41.1964s

- The sysbench benchmark time taken to run the Fileio intensive operation is plotted into a line graph and shown below.



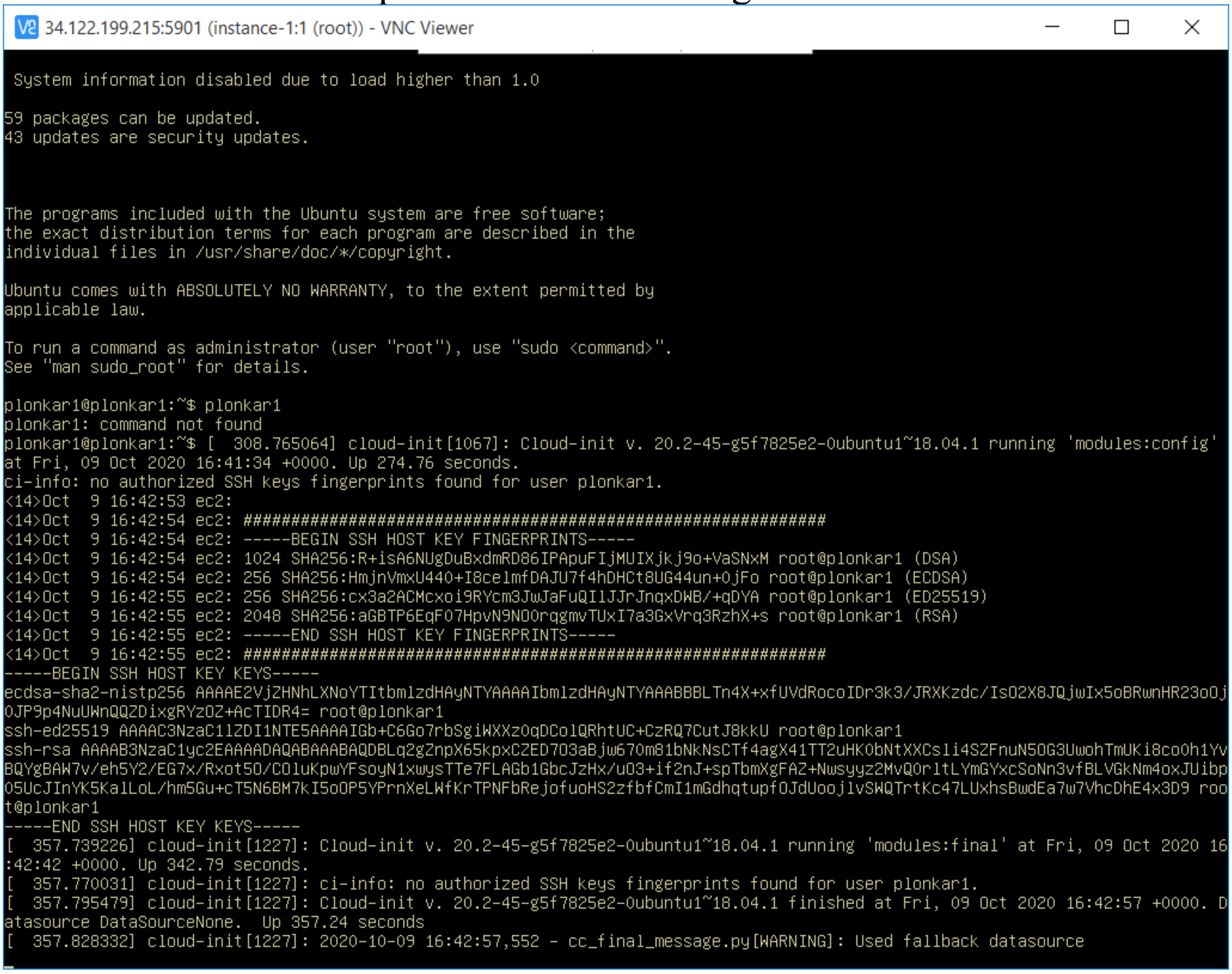
- Here too we can see that the total time taken to run the benchmark is increasing exponentially.
- The table showing the sysbench results is shown below.

File I/O Intensive Operation	**Total Time**
Test 1	29.5446s
Test 2	29.6518s
Test 3	29.9352s
Avg:	29.7105s

- The Throughput during the cpu intensive task was nearly negligible as seen in the cpu.txt file attached in the folder, this is because in cpu intensive task we do not write or read from a file.
- The Disk Utilization on the other hand was increased as the sysbench test was started. The disk utilization was nearly 50% during this test.
- The Throughput during the File I/O task was seen to be increased as the sysbench test was started as It was reading and writing to the files.
- The Disk Utilisation on the other hand was also greater as there were nearly 12 Gb of files prepared on the disk and operations were been performed on them.
- There was at least 80% latency occurring during the File I/O operation as It could not write at the same time it read hence it had to wait for the process to finish writing.

Qemu Installation and Boot Up:

- Qemu was installed using the steps given in the project-1 document.
- It took nearly 1:30 – 2 hrs to install the vm and it took nearly 274.76 seconds to boot up as shown in the image below.



```
 System information disabled due to load higher than 1.0

59 packages can be updated.
43 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

plonkar1@plonkar1:~$ plonkar1
plonkar1: command not found
plonkar1@plonkar1:~$ [  308.765064] cloud-init[1067]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1~18.04.1 running 'modules:config'
at Fri, 09 Oct 2020 16:41:34 +0000. Up 274.76 seconds.
ci-info: no authorized SSH keys fingerprints found for user plonkar1.
<14>Oct  9 16:42:53 ec2:
<14>Oct  9 16:42:54 ec2: #############################################################
<14>Oct  9 16:42:54 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Oct  9 16:42:54 ec2: 1024 SHA256:R+isA6NUgDuBxdmRD86IPApuFIjMUIXjkj9o+VaSNxM root@plonkar1 (DSA)
<14>Oct  9 16:42:54 ec2: 256 SHA256:HmjnVmxU440+I8celmfDAJU7f4hDHCt8UG44un+0jFo root@plonkar1 (ECDSA)
<14>Oct  9 16:42:55 ec2: 256 SHA256:cx3a2ACMcxoi9RYcm3JwJaFuQIlJJrJnqxDWB/+qDYA root@plonkar1 (ED25519)
<14>Oct  9 16:42:55 ec2: 2048 SHA256:aGBTP6EqF07HpvN9NO0rqgmvTUxI7a3GxVrq3RzhX+s root@plonkar1 (RSA)
<14>Oct  9 16:42:55 ec2: -----END SSH HOST KEY FINGERPRINTS-----
<14>Oct  9 16:42:55 ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBLTn4X+xfUVdRocoIDr3k3/JRXKzdc/IsO2X8JQjwIx5oBRwnHR23oOj
0JP9p4NuUWnQQZDixgRYzOZ+AcTIDR4= root@plonkar1
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGb+C6Go7rbSgiWXXzOqDColQRhtUC+CzRQ7CutJ8kkU root@plonkar1
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDBLq2gZnpX65kpxCZED7O3aBjw67Om81bNkNsCTf4agX41TT2uHK0bNtXXCsli4SZFnuN5OG3UwohTmUKi8coOh1Yv
BQYgBAW7v/eh5Y2/EG7x/Rxot5O/COluKpwYFsoyN1xwysTTe7FLAGb1GbcJzHx/uO3+if2nJ+spTbmXgFAZ+Nwsyyz2MvQOrltLYmGYxcSoNn3vfBLVGkNm4oxJUibp
O5UcJInYK5KalLoL/hm5Gu+cT5N6BM7kI5oOP5YPrnXeLWfKrTPNFbRejofuoHS2zfbfCmI1mGdhqtupfOJdUoojlvSWQTrtKc47LUxhsBwdEa7w7VhcDhE4x3D9 roo
t@plonkar1
-----END SSH HOST KEY KEYS-----
[  357.739226] cloud-init[1227]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1~18.04.1 running 'modules:final' at Fri, 09 Oct 2020 16
:42:42 +0000. Up 342.79 seconds.
[  357.770031] cloud-init[1227]: ci-info: no authorized SSH keys fingerprints found for user plonkar1.
[  357.795479] cloud-init[1227]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1~18.04.1 finished at Fri, 09 Oct 2020 16:42:57 +0000. D
atasource DataSourceNone.  Up 357.24 seconds
[  357.828332] cloud-init[1227]: 2020-10-09 16:42:57,552 - cc_final_message.py[WARNING]: Used fallback datasource
_
```

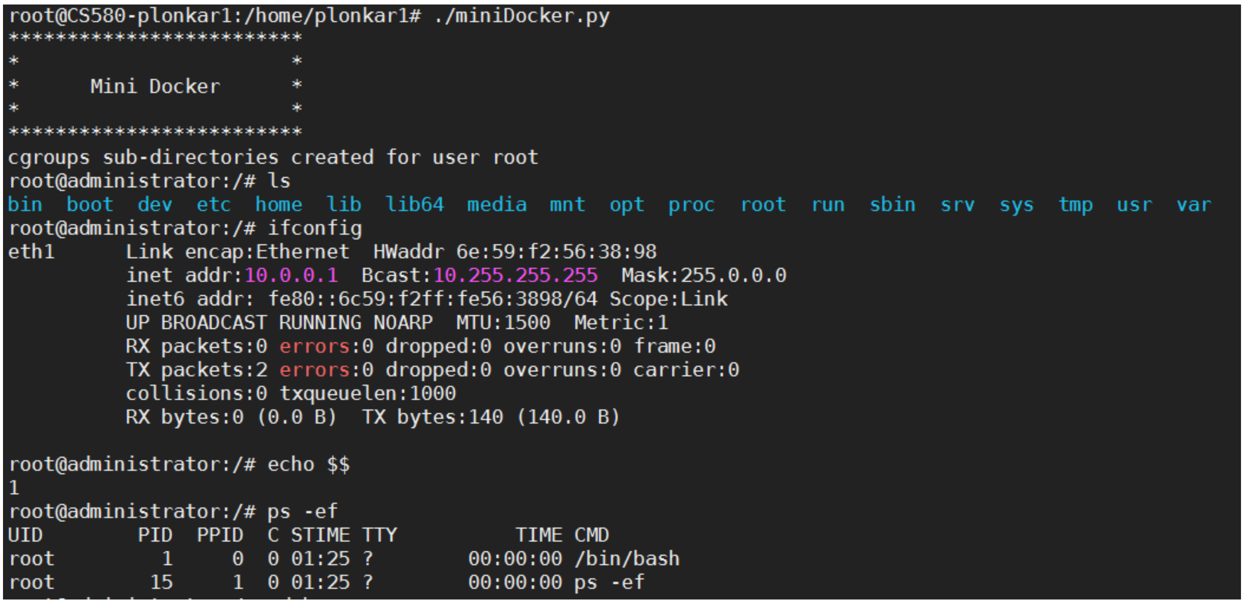
- There are many reasons why Qemu based Vm's are slow, some of them are listed below.

1. In the Qemu based Vm we are firstly installing various library to complete the installation of the Vm itself, which takes a heavy toll on the installation process.
2. We are installing the Vm thorough local server on the Google Cloud Platform, hence there is also the network issue involved.
3. The Qemu based vm's require a GUI and a lot of disk space hence it is slower to install.
4. While booting it has to load all the libraries, dependencies and also the GUI to boot hence it takes a while to boot up.

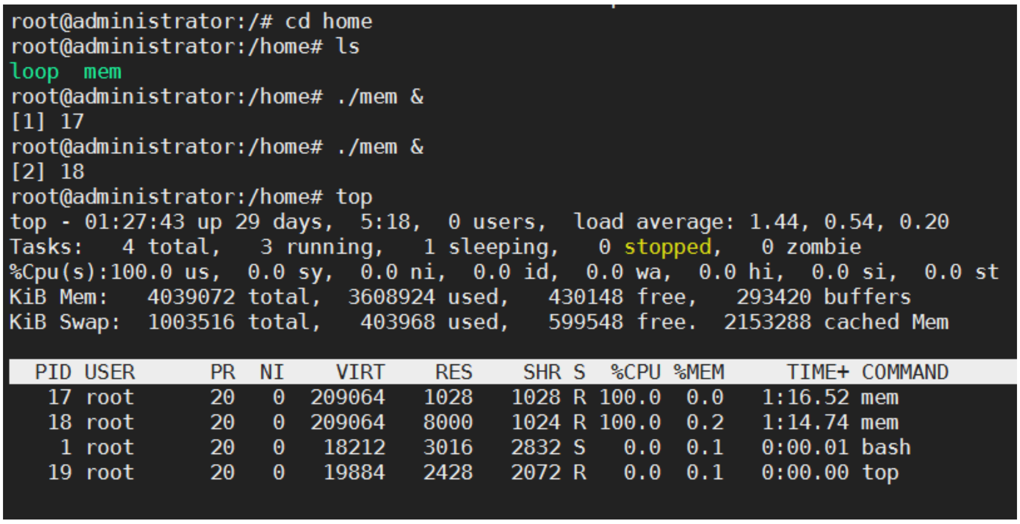
MiniDocker container:

- I have used a partial template given by the Professor i.e. miniDocker_template.py.
- I have created another python script which contains all the dependency functions required by main script minDocker.
- The dependency_function.py is imported in the miniDocker.py script as df and all the function cal are made through this alias.
- In the exe_bash() function I have used the chdir function of the os library to change the current directory of the system to our desired directory i.e. new_root.
- Then I am changing the current root using the chroot function of the os library.
- After performing the above steps we have to now mount our new directory into the system that is done with the system call function of the os library.
- In he above step I tried various mounting methods i.e. use of bind or move etc but when we perform the ps -ef operation we get a prompt to use the -t flag instead of the -bind or -move flag.
- I have basically performed 4 namespaces and one cgroups using the unshare library which was encouraged by the professor.
- For the Cgroup I followed a blog which depicted how to create a cgroup but was unable to do so[4].
- For the network namespace I have followed all the steps given by the professor in his slides[1].
- The only thing I couldn't fix was that after every itteration i have to manually delete the network device, which is done by the folloowing command:

  ```
  ip netns del myns1
  ```
- I have done much research on namespace and cgroups to complete the project apart from the references given.
- The output of the code is shown below, which resembles the desired output:



```
root@CS580-plonkar1:/home/plonkar1# ./miniDocker.py
*************************
*                       *
*      Mini Docker      *
*                       *
*************************
cgroups sub-directories created for user root
root@administrator:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@administrator:/# ifconfig
eth1      Link encap:Ethernet  HWaddr 6e:59:f2:56:38:98
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::6c59:f2ff:fe56:3898/64 Scope:Link
          UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

root@administrator:/# echo $$
1
root@administrator:/# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 01:25 ?        00:00:00 /bin/bash
root        15     1  0 01:25 ?        00:00:00 ps -ef
```



```
root@administrator:/# cd home
root@administrator:/home# ls
loop  mem
root@administrator:/home# ./mem &
[1] 17
root@administrator:/home# ./mem &
[2] 18
root@administrator:/home# top
top - 01:27:43 up 29 days,  5:18,  0 users,  load average: 1.44, 0.54, 0.20
Tasks:   4 total,   3 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   4039072 total,  3608924 used,   430148 free,   293420 buffers
KiB Swap:  1003516 total,   403968 used,   599548 free.  2153288 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   17 root      20   0  209064   1028   1028 R 100.0  0.0   1:16.52 mem
   18 root      20   0  209064   8000   1024 R 100.0  0.2   1:14.74 mem
    1 root      20   0   18212   3016   2832 S   0.0  0.1   0:00.01 bash
   19 root      20   0   19884   2428   2072 R   0.0  0.1   0:00.00 top
```

References

1.)http://www.cs.binghamton.edu/~huilu/slidesfall2020/Lecture_4_5_Containerization.pdf

2.)https://man7.org/linux/man-pages/man2/unshare.2.html

3.)https://pypi.org/project/unshare/

4) https://github.com/francisbouvier/cgroups

5.)https://blogs.igalia.com/dpino/2016/04/10/network-namespaces/