COEN 241

SYSTEM VS OS VIRTUALIZATION

JAYASRI RAMAKRISHNAN

INTRODUCTION **2**

ENVIRONMENT **3**

QEMU **3**

SETUP - INSTALLATION 4

CONFIGURATIONS AND ARGUMENTS: 5

Default 5

Exploring different arguments for qemu 5

DOCKER **9**

SETUP - INSTALLATION 9

ENABLING A DOCKER CONTAINER 10

SYSBENCH - DOCKER **11**

CPU Tests: 11

Test 1: 11

Test 2: 12

Test 3: 12

User vs Kernel Space Comparison: 13

FileIO Tests 13

Test 1: 13

Test 2: 14

Test 3: 14

Disk Utilization 15

SYSBENCH - QEMU - UBUNTU **15**

CPU Tests: 15

Test 1: 15

Test 2: 16

Test 3: 16

User vs Kernel Space Comparison: 17

FileIO Tests 18

Test 1: 18

Test 2: 19

Test 3: 19
Disk Utilization 19
INFERENCE: **21**
VAGRANT **22**
DOCKERFILE **23**

INTRODUCTION

This report will cover the setup, configuration, operations, benchmarking, comparison and comprehensive results of comparison of two virtualization techniques

- System Virtualization with QEMU
- OS Virtualization via Docker

ENVIRONMENT

The environment for this assignment is use of ***personal computer*** with the following configurations

QEMU

QEMU is an open source machine emulator and a virtualizer. It's a hypervisor and emulates the machine's processor through dynamic binary translation and provides a set of different hardware and device models for the machine, enabling it to run a variety of guest operating systems. For this experiment, we will be using Ubuntu as the guest OS.

SETUP - INSTALLATION

The below were the steps followed to install QEMU as the hypervisor and have Ubuntu as Guest OS on top of it.

1. Downloaded the .iso file for ubuntu version 20.04_03
2. Installed homebrew , the package manager which will aid in the installation of QEMU for MAC with intel chip

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

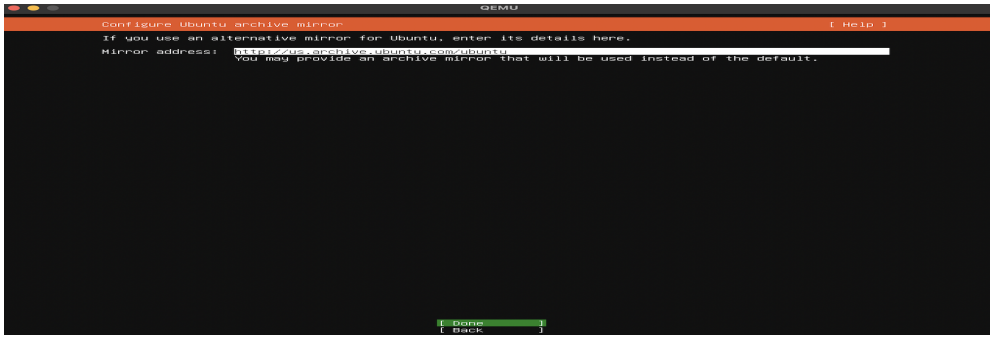
3. Then, installed QEMU using

```
brew install qemu
```

4. Then setup the ubuntu os on qemu using

```
 Jayasri:~ vigneshthirunavukkarasu$ qemu-system-x86_64 -hda ubuntu.img -boot d -cdrom
/Users/vigneshthirunavukkarasu/Downloads/ubuntu-20.04.3-live-server-amd64.iso -m 1536
```

5. Couple of screenshots to show the progress of installation



CONFIGURATIONS AND ARGUMENTS:

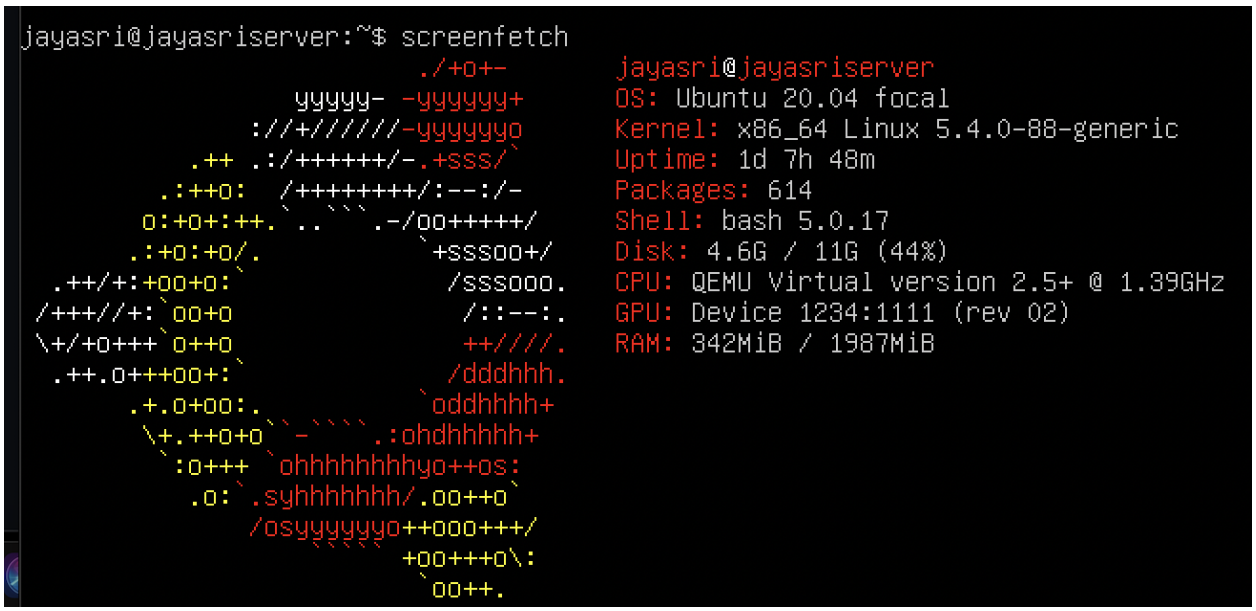
QEMU can be booted with different arguments.

Default

When booted with memory of 2G using -m 2G argument without changing CPU, the configurations look like the below.

Command:

```
sudo qemu-system-x86_64 -hda ubuntu.img -boot d -m 2G
```



Exploring different arguments for qemu

Command:

```
Jayasri:~ vigneshthirunavukkarasu$ sudo qemu-system-x86_64  -machine accel=hvf ubuntu.i
mg -boot d --cpu Haswell-v4 -smp 2 -m 4G
```

Explaining each below:

Memory -m:

Memory is set to 4G.

CPU -cpu:

If looked at the screenshot in default section, the CPU is QEMU virtual version 2.5+. This can be changed to use any of the below to improve compute or use the host itself.

```
Jayasri:~ vigneshthirunavukkarasu$ qemu-system-x86_64 -cpu help
Available CPUs:
x86 486                  (alias configured by machine type)
x86 486-v1
x86 Broadwell            (alias configured by machine type)
x86 Broadwell-IBRS       (alias of Broadwell-v3)
x86 Broadwell-noTSX      (alias of Broadwell-v2)
x86 Broadwell-noTSX-IBRS  (alias of Broadwell-v4)
x86 Broadwell-v1         Intel Core Processor (Broadwell)
x86 Broadwell-v2         Intel Core Processor (Broadwell, no TSX)
x86 Broadwell-v3         Intel Core Processor (Broadwell, IBRS)
x86 Broadwell-v4         Intel Core Processor (Broadwell, no TSX, IBRS)
x86 Cascadelake-Server   (alias configured by machine type)
x86 Cascadelake-Server-noTSX  (alias of Cascadelake-Server-v3)
x86 Cascadelake-Server-v1  Intel Xeon Processor (Cascadelake)
x86 Cascadelake-Server-v2  Intel Xeon Processor (Cascadelake) [ARCH_CAPABILITIES]
x86 Cascadelake-Server-v3  Intel Xeon Processor (Cascadelake) [ARCH_CAPABILITIES, no TSX]
x86 Cascadelake-Server-v4  Intel Xeon Processor (Cascadelake) [ARCH_CAPABILITIES, no TSX]
```

The command for using host would be

-cpu host

After changing cpu, the configuration looks like:

The CPU has changed to Intel Core (Haswell)

```
Last login: Fri Oct  8 20:09:05 UTC 2021 on tty1
jayasri@jayasriserver:~$ screenfetch
                           ./+o+-       jayasri@jayasriserver
                   yyyyy- -yyyyyy+      OS: Ubuntu 20.04 focal
                ://+//////-yyyyyyo      Kernel: x86_64 Linux 5.4.0-88-generic
            .++ .:/++++++/-.+sss/`      Uptime: 1m
          .:++o:  /++++++++/:--:/-      Packages: 614
         o:+o+:++.`..`` `.-/oo+++++/    Shell: bash 5.0.17
        .:+o:+o/.          `+sssoo+/    Disk: 4.6G / 12G (40%)
  .++/+:+oo+o:`             /sssooo.   CPU: Intel Core (Haswell, no TSX, IBRS) @ 2x 1.392GHz
 /+++//+:`oo+o               /::--:.   GPU: Device 1234:1111 (rev 02)
 \+/+o+++`o++o               ++////.   RAM: 363MiB / 3936MiB
  .++.o+++oo+:`             /dddhhh.
       .+.o+oo:.            `oddhhhh+
        \+.++o+o``-````.:ohdhhhhh+
         `:o+++ `ohhhhhhhhyo++os:
           .o:`.syhhhhhhh/.oo++o`
               /osyyyyyyo++ooo+++/
                   ````` +oo+++o\:
                          `oo++.
```

Cores: -smp

This argument is used to configure the number of cores.

Before setting the cores using -smp command,

```
jayasri@jayasriserver:~$ lscpu
Architecture: x86_64
CPU op-mode(s):   32-bit, 64-bit
Byte Order: Little Endian
Address sizes: 40 bits physical, 48 bits virtua
CPU(s):   1
On-line CPU(s) list: 0
Thread(s) per core: 1
Core(s) per socket: 1
Socket(s):   1
NUMA node(s):   1
Vendor ID: AuthenticAMD
CPU family: 15
Model: 107
Model name: QEMU Virtual CPU version 2.5+
```
```

After Setting smp to 2, we can see from the below screenshot that the cpus and sockets have increased to 2

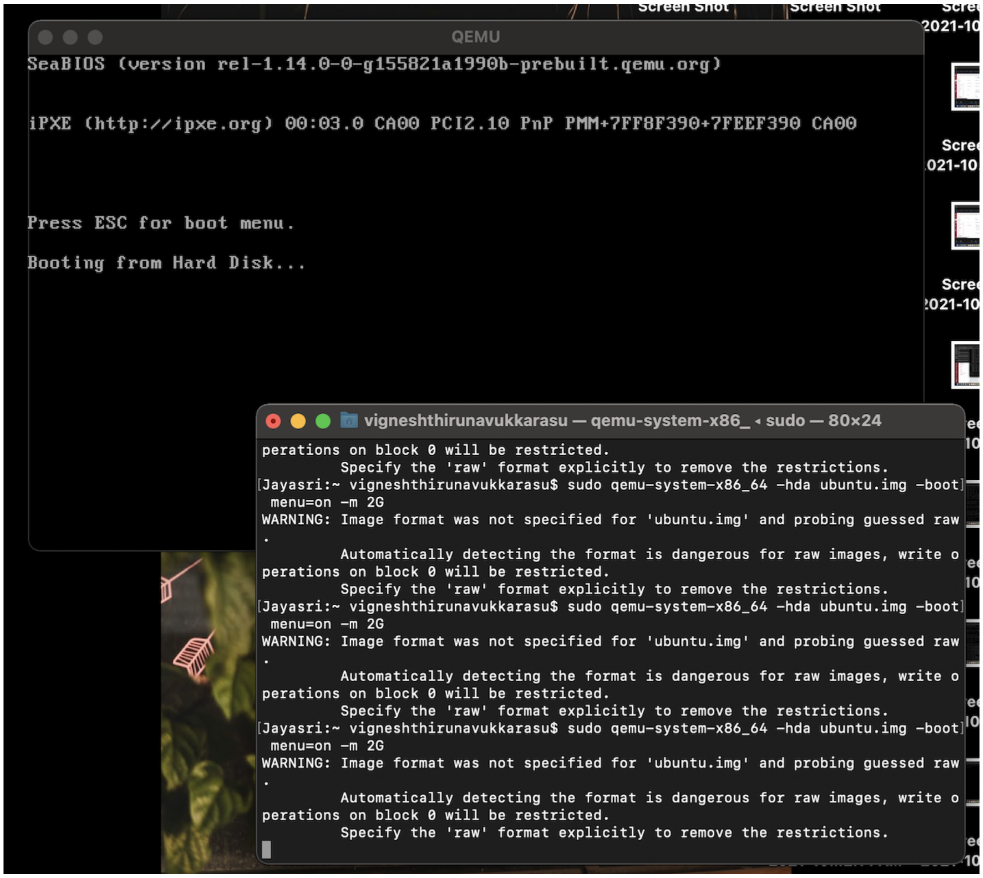
Sppedup -accel:

-accel can be used to speedup qemu. We have used hvf for speedup as shown in the main command.

```
[Jayasri:~ vigneshthirunavukkarasu$ qemu-system-x86_64 -accel help
 Accelerators supported in QEMU binary:
[tcg
 hax
 hvf
```

Menu:

When given menu=on, it gives the option of from where to boot the image from - cdrom or hard disk or dvd drive. You can see from the below screenshot that “Press ESC for boot MENU” appears

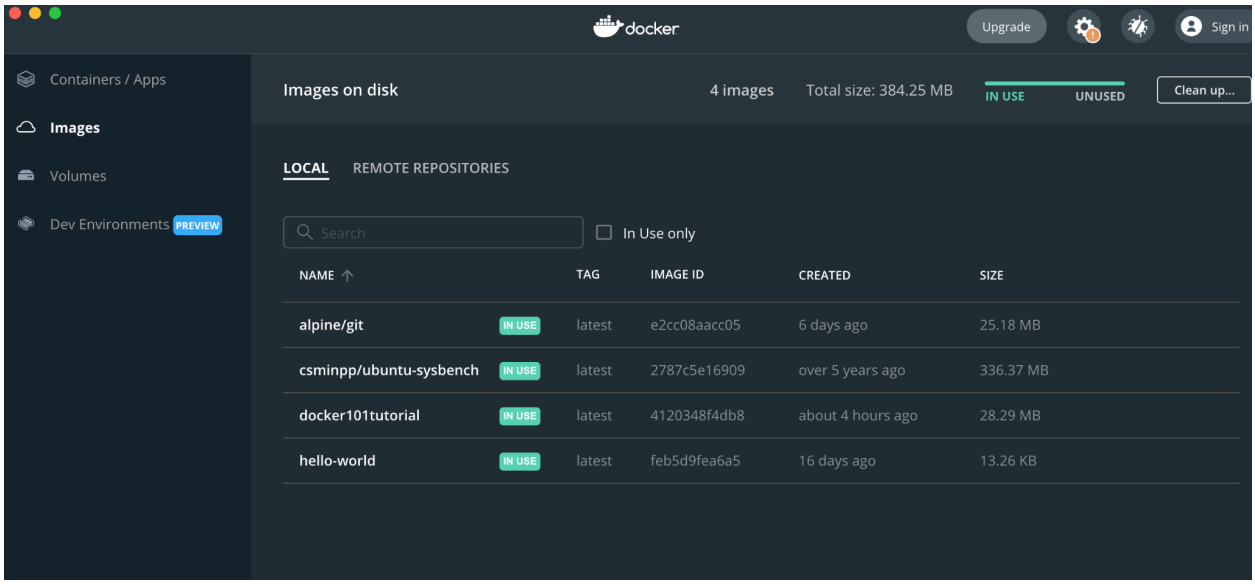


DOCKER

Docker is a set of platforms that enable OS-level virtualization to deliver software in packages called containers. Docker is the most popular container management platform.

SETUP - INSTALLATION

Downloaded docker desktop executable file from https://docs.docker.com/desktop/mac/install/ and followed the wizard for completion. Screenshot of docker desktop with list of images below



ENABLING A DOCKER CONTAINER

We are going to use the busybox container as an example to see how to enable a docker container. The main steps are:

a. Fetch the image busybox from docker hub repository and store it in the system

```
Jayasri:~ vigneshthirunavukkarasu$ docker pull busybox
```

b. Check images to reconfirm

```
Jayasri:~ vigneshthirunavukkarasu$ docker images
```

c. Run a command in the container using docker run. Docker finds the image, loads the container and runs the command in the container.

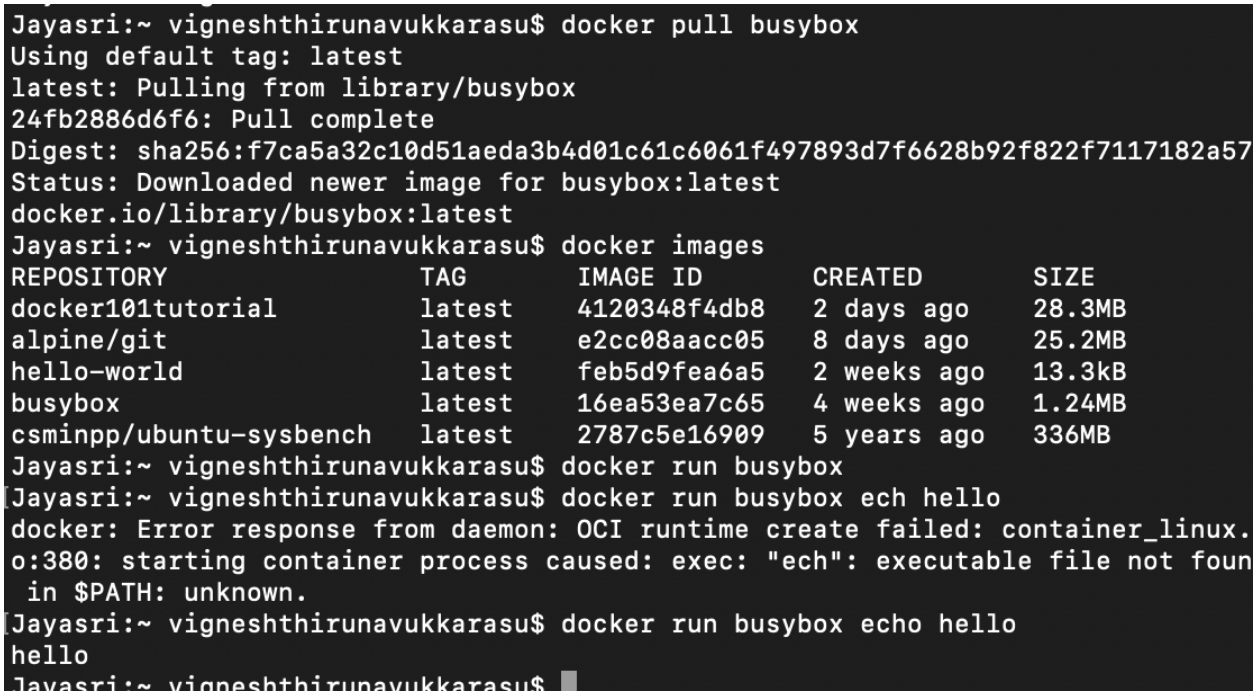
```
Jayasri:~ vigneshthirunavukkarasu$ docker run busybox echo hello
```

d. Docker run command with -it flag lets us get an interactive terminal in the container. This will be used later with sysbench

e. docker ps -a will show all the containers with container id and status

f. Docker rm <id> will remove the container.

Screenshot of all the steps:



SYSBENCH - DOCKER

CPU Tests:

Have run maxprime cpu test for 3 different test scenarios. Each has run 5 times. Automated scripts and corresponding outputs are available in github repo

Script: cpuTest_Docker.sh

Output: cpuTest_docker_output.txt

The consolidated metrics of the different test runs are below

Test 1:

Docker	20000					
	Events/Sec	min	max	avg	std	95 percentile
1	468	2.01ms	18446744073678.15ms	2.21ms	0	2.54ms
2	469	1.96ms	18446744073677.96ms	2.11ms	0	2.47ms
3	466	2.00ms	18446744073678.05ms	2.12ms	0	2.41ms
4	462	1.99ms	18446744073677.48ms	2.16ms	0	2.33ms
5	467	2.01ms	4.82ms	2.14ms	0	2.26ms

Test 2:

Docker	50000					
	Events/Sec	Min ms	max	Avg ms	std	95 percentile ms
1	132	7.08	18446744073683.76ms	7.55	0	8.29
2	135	7.04	18446744073682.76ms	7.42	0	7.79
3	134	7.08	18446744073687.00ms	7.45	0	7.89
4	128	7.05	18446744073686.71ms	7.82	0	8.62
5	14	7.07	18446744073685.14ms	69.31	0	10.54

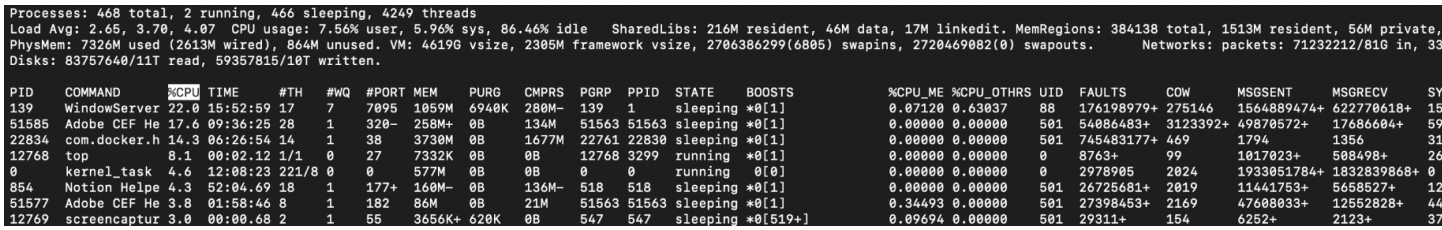
Test 3:

Dcoker	90000					
	Events/Sec	Min ms	Max ms	Avg ms	std	95 percentile ms
1	59	16.27	18446744073699.46ms	17.03	0	18.05

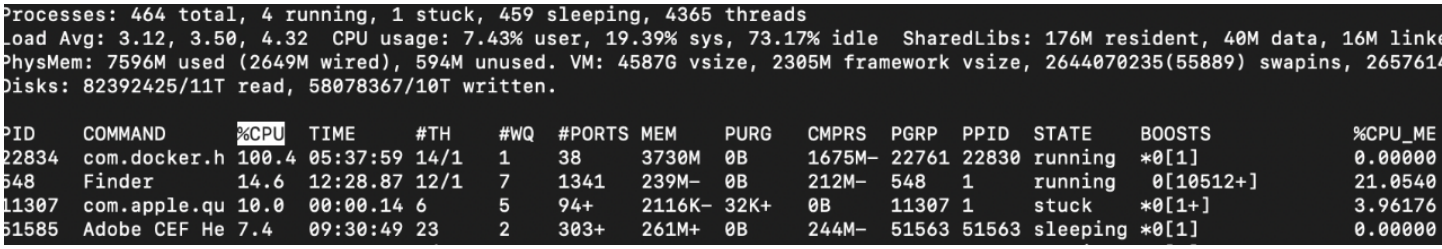
2	59	16.26	18446744073694.41ms	16.96	0	17.9
3	58	16.28	18446744073697.96ms	17.16	0	18.74
4	59	16.24	18446744073693.59ms	17.02	0	18.08
5	58	16.31	18446744073696.18ms	17.11	0	18.5

User vs Kernel Space Comparison:

Before running sysbench tests, the user vs kernel space usage were:



While running sysbench tests, the user vs kernel space usage were:



As you can see, the cpu utilization by docker sysbench test run was at 100% and 7.43% of user space utilized vs 19.39% of kernel space utilized. The system space utilization has gone up.

FileIO Tests

The fileio tests were performed for three different modes with 16 threads and 1GB filesize

Test 1:

Docker	rndwr					
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	72.75	4656	0.02	0.15	0.62	91.93
2	57.67	3691	0.02	0.16	22.69	72.73
3	44.683	2859.74	0.02	0.13	7.29	79.52
4	42.87	2745	0.02	0.18	26.02	104.92
5	67.12	4296.22	0.02	0.18	8.94	91.78

Test 2:

Docker	seqrd					
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	1246	76437	0.01	0.15	145	451
2	1134	75643	0.01	0.16	125	645
3	996	76437	0.01	0.15	312	342
4	1276	74861	0.01	0.18	145	621
5	1134	73281	0.01	0.16	145	451

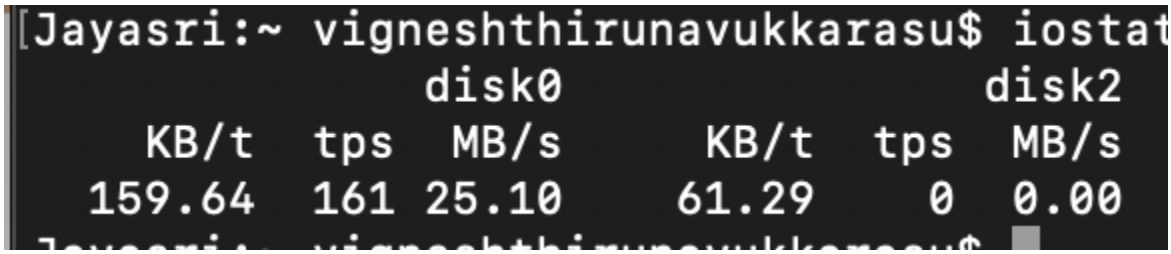
Test 3:

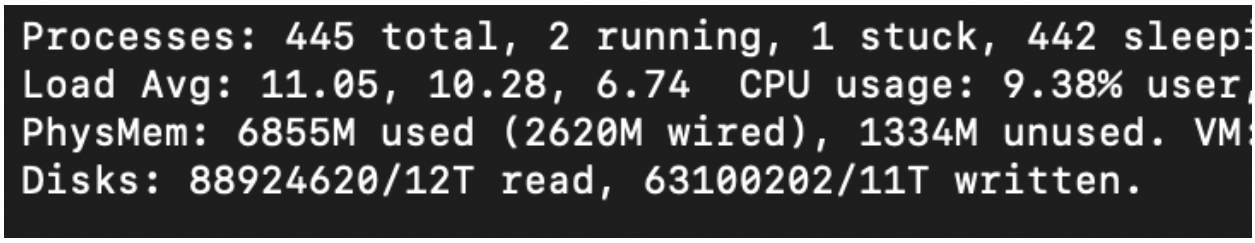
Docker	seqrewr					
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	200.4	12825.63	0.03	0.78	228.79	466.62
2	201	12906	0.03	0.71	659	475
3	151	9668	0.04	1.35	323	368

4	215	13767	0.03	0.71	414	613
5	186	11913	0.03	0.56	226	491

Disk Utilization

The disk utilization before running docker fileio tests was:





While running sysbench tests,

SYSBENCH - QEMU - UBUNTU

CPU Tests:

Have run maxprime cpu test for 3 different test scenarios. Each has run 5 times. Automated scripts and corresponding outputs are available in github repo

Script: benchmarkcpu.sh

Output: cpuTest_QEMU_Output.txt

The consolidated metrics of the different test runs are below

Test 1:

QEMU	20000					
	Events/Sec	Min ms	Max ms	Avg ms	std	95 percentile
1	140	6.63	16.79	7.1	0	8.13
2	141	6.61	13.53	7.03	0	7.56
3	134	6.62	6.11	7.43	0	9.22
4	138	6.62	28.81	71.9	0	8.13
5	138	6.67	23.88	7.19	0	7.98

Test 2:

QEMU	50000					
	Events/Sec	min	max	avg	std	95 percentile
1	41.34	23.13	43.24	24.11	0	25.28
2	41.32	22.9	30.65	24.1	0	25.28
3	41.24	23.04	33.52	24.16	0	25.28
4	41.2	23.15	43.63	24.17	0	25.28
5	41.34	23.3	28.61	24.1	0	24.83

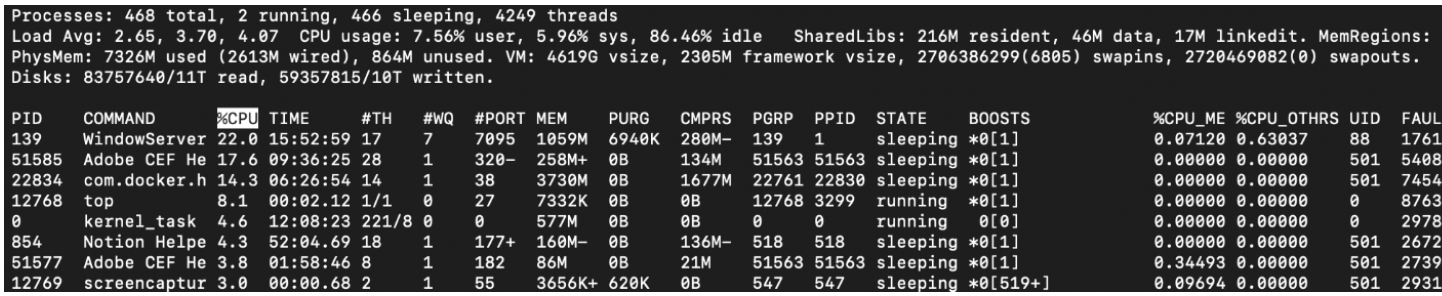
Test 3:

QEMU	90000					

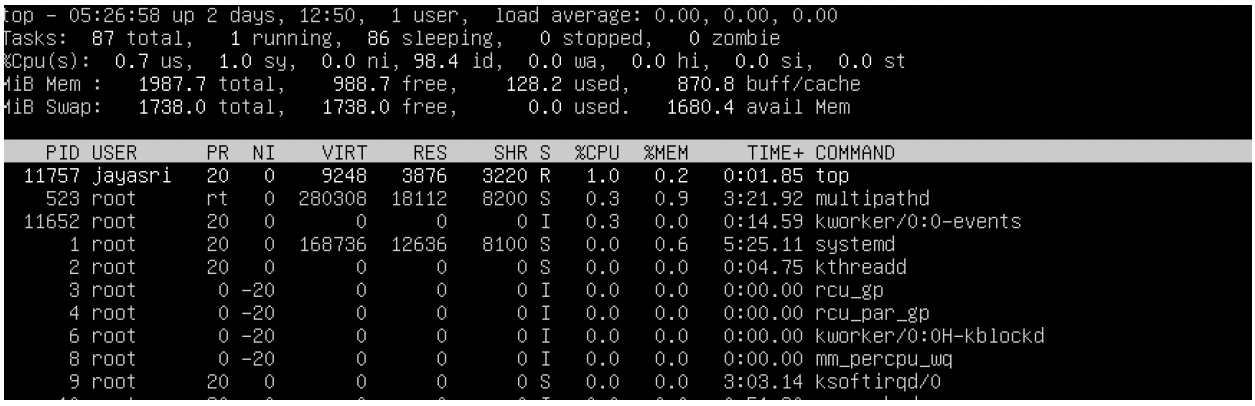
	Events/Sec	Min ms	Max ms	avgms	std	95 percentile
1	18.59	52.15	62.32	53.63	0	55.82
2	18.75	51.61	58.11	53.2	0	54.83
3	18.24	51.99	167.36	54.67	0	55.82
4	18.63	52.17	61.77	53.5	0	54.83
5	18.54	52.16	86.25	53.79	0	54.83

User vs Kernel Space Comparison:

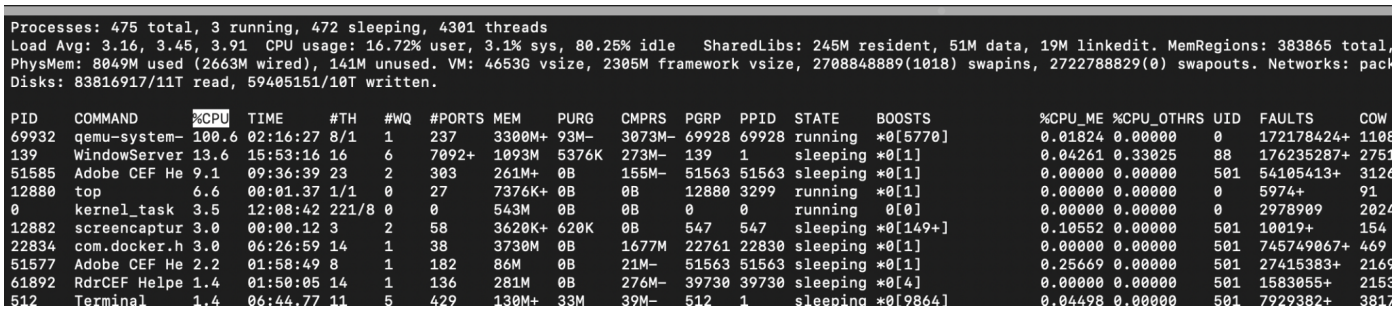
Before running sysbench test in QEMU, the host OS's user vs kernel space CPU usage % was:



And in guest OS ubuntu:

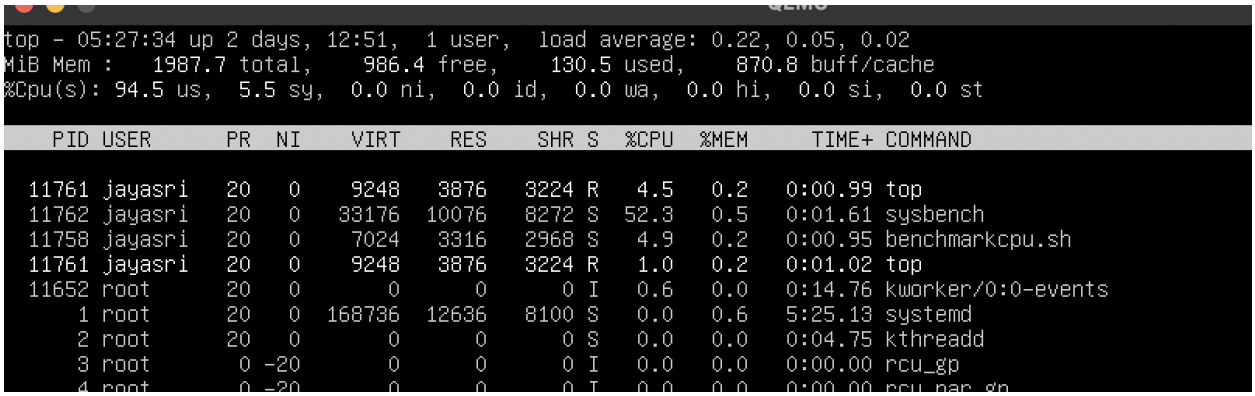


While running sysbench, utilization in host OS:



The cpu utilization has gone upto 100% and the user space utilization has increased to 16.72%

In Guest OS,



The user space has shot up by 94.5% from 0.7%

FileIO Tests

File IO tests were for three modes with 16 threads and 1GB total file size

Test 1:

QEMU	rndwr					
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	16.46	2394	0.03	6.26	121.86	118.78

2	16.24	2369	0.03	6.38	336.74	169
3	16	2330	0.03	6.5	95.53	133.93
4	16.83	2451	0.03	6.13	128.22	110.31
5	16.17	2353	0.03	6.42	99.28	147.69

Test 2:

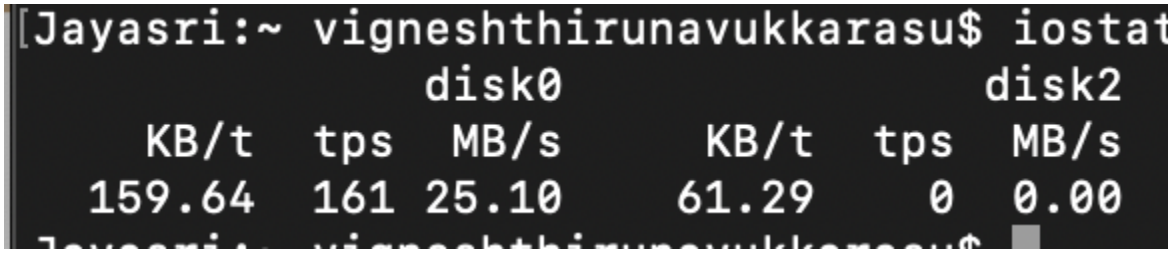
QEMU	seqrd					
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	321.85	20669	0.01	0.58	1485	536
2	620	40393	0.01	0.24	189	309
3	197.25	12911.3	0.01	0.75	9875	1732.92
4	634.81	49670	0.01	0.24	134.3	293.44
5	686	44011	0.01	0.22	127.47	251.56

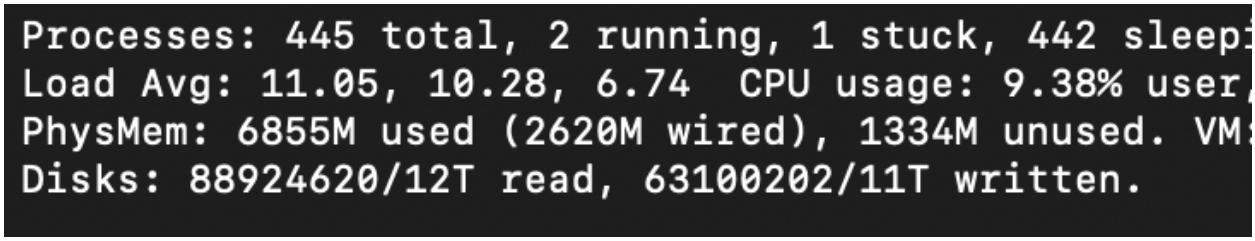
Test 3:

QEMU	seqrewr					
	Transfer Rate Mb/sec	Requests/sec	min ms	avg ms	max ms	std
1	6.63	961.24	0.04	12.82	1893.64	154.92
2	10.59	1537	0.04	9.11	372.52	174.95
3	16.39	2619	0.04	5.87	224.71	242.95
4	24	4722	0.03	3.18	135.99	185.6
5	37.34	5831	0.03	2.68	206.73	242

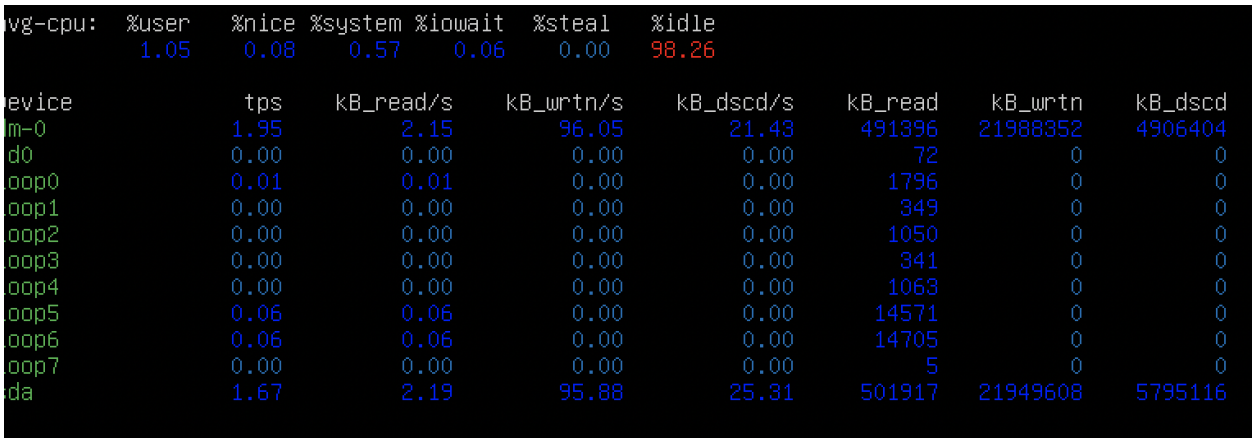
Disk Utilization

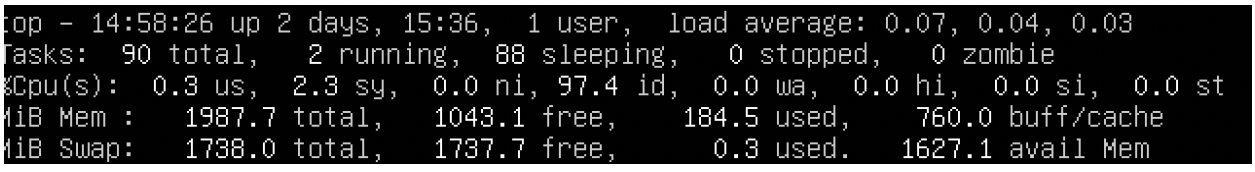
The disk utilization before running of filieo tests in host os was





And in Guest OS:





While running sysbench tests,

```
Processes: 426 total, 3 running, 423 sleeping, 4169 threads                 08:14:4
Load Avg: 4.28, 5.10, 5.61  CPU usage: 18.74% user, 6.16% sys, 75.8% idle
SharedLibs: 159M resident, 30M data, 12M linkedit.
MemRegions: 368087 total, 2390M resident, 57M private, 599M shared.
PhysMem: 7810M used (2780M wired), 380M unused.
VM: 4379G vsize, 2305M framework vsize, 2906241365(40287) swapins, 2920417583(0) s
Networks: packets: 71589185/82G in, 33734833/12G out.
Disks: 89329433/12T read, 63498593/11T written.

PID    COMMAND      %CPU  TIME     #TH    #WQ  #PORTS MEM    PURG   CMPRS  PGRP
69932  qemu-system- 98.5  02:43:04 8/1    1    237    3356M  2308K+ 2065M- 69928
```

In guest OS,

```
top - 15:42:53 up 2 days, 16:10,  1 user,  load average: 2.64, 0.70, 0.82
MiB Mem :   1987.7 total,   1121.5 free,    185.9 used,    680.3 buff/cache
%Cpu(s):  0.7 us, 93.2 sy,  0.0 ni,  0.0 id,  5.3 wa,  0.0 hi,  0.7 si,  0.0 st
MiB Mem :   1987.7 total,    736.8 free,    186.7 used,   1064.2 buff/cache
MiB Swap:   1738.0 total,   1737.7 free,      0.3 used.   1621.1 avail Mem
Maximum tasks = 0, change to (0 is unlimited) WARNING: the --test option is deprecated. Y
e command line without any options.   0      0 S   1.2   0.0   1:27.18 jbd2/dm-0-8
_ 13395 jayasri   20   0   33384  10324   8704 R  81.4   0.5   0:05.18 sysbench
```

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          1.07    0.08    0.75    0.07    0.00   98.04

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
dm-0             3.38         2.36       186.34        21.24     544624   43052900    4906404
fd0              0.00         0.00         0.00         0.00         72          0          0
loop0            0.01         0.01         0.00         0.00       1796          0          0
loop1            0.00         0.00         0.00         0.00        349          0          0
loop2            0.00         0.00         0.00         0.00       1050          0          0
loop3            0.00         0.00         0.00         0.00        341          0          0
loop4            0.00         0.00         0.00         0.00       1063          0          0
loop5            0.06         0.06         0.00         0.00      14571          0          0
loop6            0.06         0.06         0.00         0.00      14705          0          0
loop7            0.00         0.00         0.00         0.00          5          0          0
sda              3.06         2.40       186.10        25.08     555145   42997180    5795116
```

The usage of physical memory has gone up from 6855M to 7810M in host OS

INFERENCE:

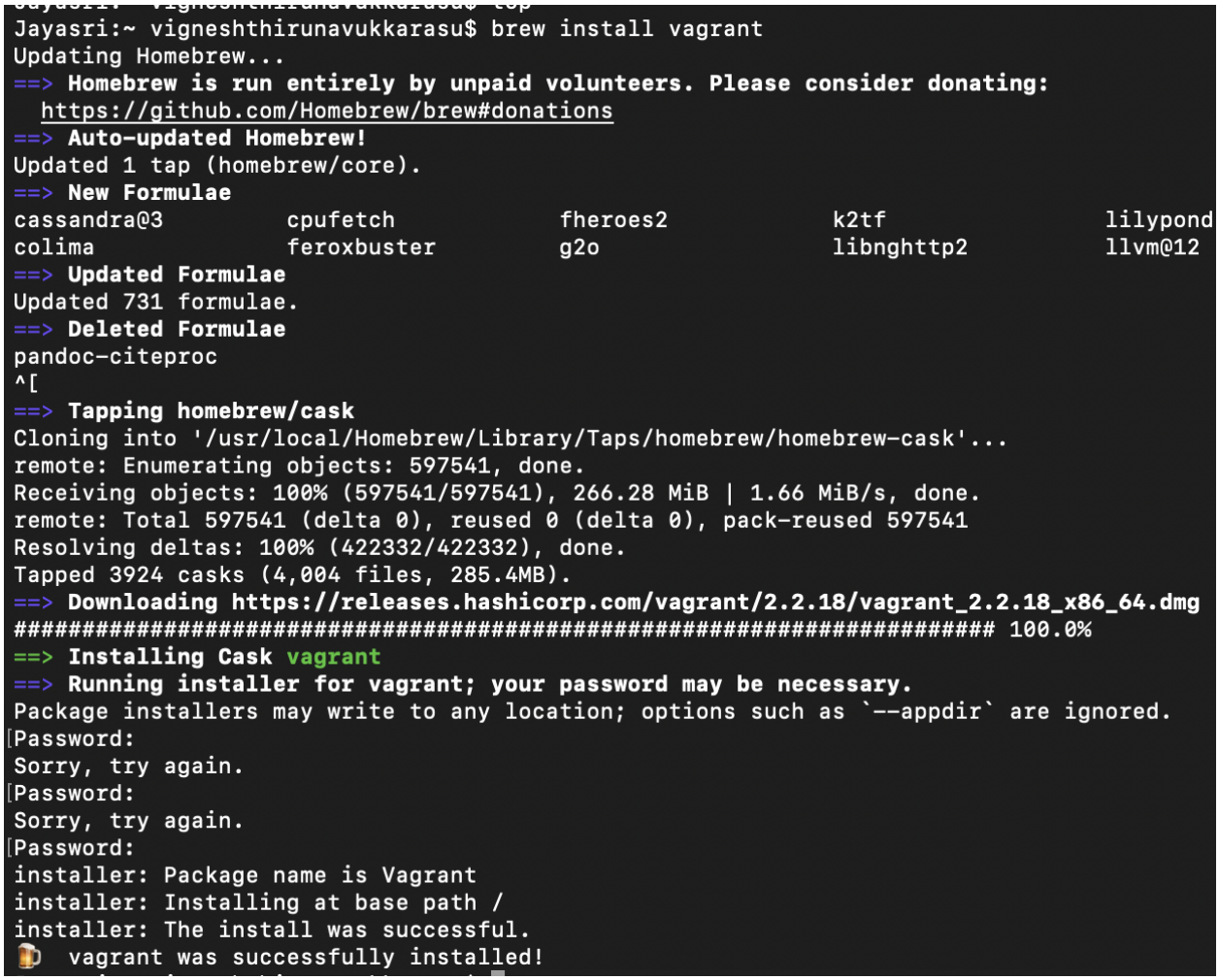
1. QEMU boots up faster when memory is increased, cores are increased and acceleration is set for speedup
2. The CPU events per second is higher in OS virtualization compared to System Virtualization. Its around 460 events/sec in Docker vs 123 events/sec in QEMU

3. The CPU events per second decreases when the maxprime limit is set higher for both System and OS virtualization
4. Even with the decrease, OS virtualization performs better even for higher maxprime values
5. In terms of user space vs kernel space utilization, qemu sysbench tests seem to burden the user space. From the above stats, its visible that sysbench tests took about 94% of userspace from 0.7%
6. Docker sysbench tests don't have that drastic an impact on user vs kernel space utilization. System space % went up marginally upto 19.39%
7. The transfer rate and the events per second is higher in OS virtualization compared to System virtualization
8. The throughput increases when we increase the num of threads for fileio tests.
9. The disk and physical memory usage of host OS increase dramatically when QEMU fileio tests run
10. OS virtualization performs better compared to System virtualization as seen from the cpu tests and fileio tests

VAGRANT

Vagrant is an open source tool for creating and configuring VMs. In order to automate the bring up of QEMU and ubuntu as guest os in it, we require virtualization management library libvirt.

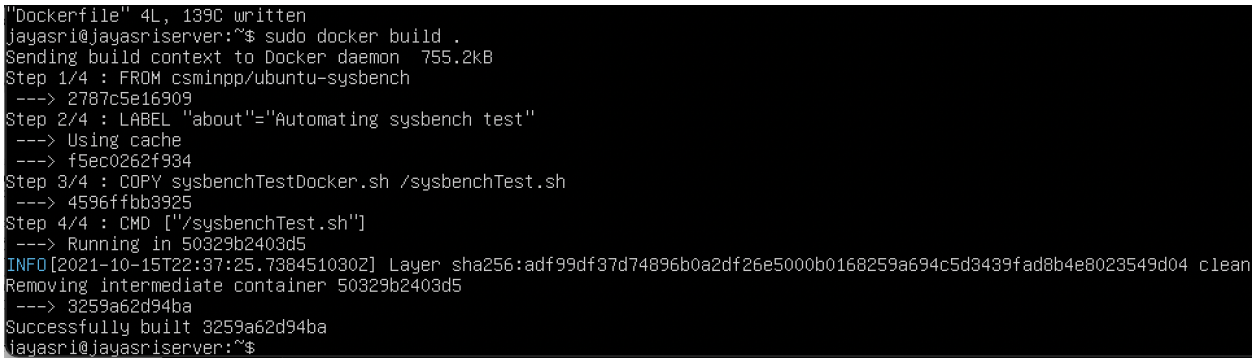
1. Install QEMU - Refer QEMU chapter above
2. Install Vagrant
    a. Brew install vagrant



3. Install libvirt
4. Find the box compatible with libvrt from https://app.vagrantup.com/boxes/search?page=1&provider=libvirt
5. Vagrant file shared in github: VagrantFile

DOCKERFILE

1. Dockerfile placed in github repo
2. Docker build container


"Dockerfile" 4L, 139C written
jayasri@jayasriserver:~$ sudo docker build .
Sending build context to Docker daemon 755.2kB
Step 1/4 : FROM csminpp/ubuntu-sysbench
 ---> 2787c5e16909
Step 2/4 : LABEL "about"="Automating sysbench test"
 ---> Using cache
 ---> f5ec0262f934
Step 3/4 : COPY sysbenchTestDocker.sh /sysbenchTest.sh
 ---> 4596ffbb3925
Step 4/4 : CMD ["/sysbenchTest.sh"]
 ---> Running in 50329b2403d5
INFO[2021-10-15T22:37:25.738451030Z] Layer sha256:adf99df37d74896b0a2df26e5000b0168259a694c5d3439fad8b4e8023549d04 clean
Removing intermediate container 50329b2403d5
 ---> 3259a62d94ba
Successfully built 3259a62d94ba
jayasri@jayasriserver:~$