**CSEN 241: Cloud Computing HW 1**

**Title: System VS OS Virtualization**



**By:**

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Email ID |
| Kartiki Rajendra Dindorkar | 1651519 | [kdindorkar@scu.edu](mailto:kdindorkar@scu.edu) |

Table of Contents

[System Configurations 2](#_Toc18406315)

[GitHub Repository 2](#_Toc363816391)

[QEMU Installation and Ubuntu 20.04 2](#_Toc599329988)

[Ubuntu 20.04 Configuration 2](#_Toc2147442289)

[Docker Installation 3](#_Toc1580737364)

[Ubuntu 20.04 with Sysbench on Docker 3](#_Toc1001665997)

[Experiments 3](#_Toc2084448540)

[QEMU (System Virtualization) 3](#_Toc826707973)

[Docker (OS Virtualization) 3](#_Toc2041271932)

[Analysis 3](#_Toc1623930116)

[Vagrant 3](#_Toc1336601727)

[Automation 3](#_Toc499295888)

[Conclusion 3](#_Toc23091508)

# System Configurations

|  |  |
| --- | --- |
| Operating System | MAC OS Sonoma 14.2.1 |
| Chip | Apple M2 |
| System Architecture | arm64 |
| Memory | 8 GB |
| Number of Cores | 8 (4 performance and 4 efficiency) |

# GitHub Repository

[GitHub - CloudComputing](https://github.com/Kartiki19/CloudComputing)

# QEMU Installation and Ubuntu 20.04

Refer to [QEMU and Ubuntu 20.04 Setup](https://github.com/Kartiki19/CloudComputing/blob/main/HW1/QEMUUbuntuSetup.md) for Installing QEMU on Apple MAC M2 chip and setting Ubuntu 20.04 OS.

# Ubuntu 20.04 Configuration

After [creating raw image](https://github.com/Kartiki19/CloudComputing/blob/main/HW1/QEMUUbuntuSetup.md#create-qemu-raw-format-image) to [install it using installation command,](https://github.com/Kartiki19/CloudComputing/blob/main/HW1/QEMUUbuntuSetup.md#demo) let us look what all the used flags means:



|  |  |
| --- | --- |
| qemu-system-aarch64 | QEMU [package](https://github.com/Kartiki19/CloudComputing/blob/main/HW1/QEMUUbuntuSetup.md#check-available-qemu-packages) to simulate 64-bit ARM machine . |
| -nodefaults | Don’t create default devices like serial port, parallel port, VGA adapter, etc. |
| -accel | Enable an accelerator depending on target device. (kvm, xen, hvf, nvmm, whpx or tcg). |
| -m | Guest startup RAM size, default is 128 M, here we are allocating 2 Gb RAM. |
| -smp | Number of CPU allocating to the VM, here we are allocating 2 CPUs. |
| -drive file=file | Defines a new drive and which disk image to use with this drive. |
| -device | Adds device driver, here we have added, keyboard, mouse, ehci, network device and block device. |
| -netdev | Configures user mode host network backend. |
| -vga | Selects type of VGA card. |
| -cdrom file | Use file as CD-ROM image to boot the ISO file. |
| -usb | Enables USB emulation on machines with on-board USB host controller. |

# Install SysBench on QEMU Ubuntu VM

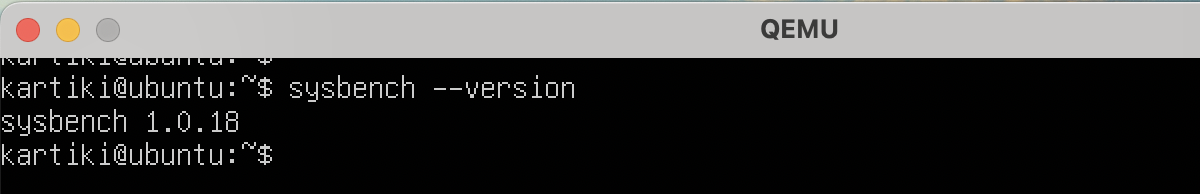
1. Once you sign in with your username and password on Ubuntu VM, install “SysBench” using following commands:

$ sudo apt update

$ sudo apt install sysbench

1. Check the SysBench version using:

$ sysbench –version



# SysBench

1. The general format of SysBench is:

$ sysbench [common-options] --test=name [test-options] command

1. Use “cleanup” command to remove temporary data after the test run:

$ sysbench --test=fileio --file-total-size=3G cleanup

1. Multiple Common options are:

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Default Value** |
| --num-threads | Total no of threads to create | 1 |
| --max-requests | Total number of requests (0=Infinite) | 10000 |
| --max-time | Limit total execution time (sec) | 0 |
| ---forced-shutdown | Amount of time to wait after –max-time to force shutdown. (sec / %) | off |
| --test | Name of the test mode to run | REQUIRED |
| --debug | Display more debug info | off |
| --validate | Validate the test results | off |
| --batch | Periodic dump of current status of run | off |
| --batch-delay | Delay between batch dump (sec) | 300 |

1. Batch option can be utilized as:

$ sysbench –batch –batch-delay=5 –test=cpu –cpu-max-prime=500 run

1. Test Modes:
   1. CPU: using *–cpu-max-primes* option, we will request to calculate prime numbers up to specified value. Each thread executed concurrently until calculation is completed or *–max-time* is completed

*$ sysbench –test=cpu –cpu-max-prime=50000 run*

* 1. Memory: It is used to benchmark sequential memory reads or writes. Using –memory-block-size we can provide the size of memory block to use, its default value is 1 KiB.

$ sysbench –test=memory –memory-block-size=1000K run

* 1. Fileio: This test mode can help to produce multiple types of file I/O workloads
     1. Prepare stage – SysBench creates specified number files with specified total size.
     2. run stage – each thread performs a specified operation.
     3. Supported I/O Operations:

|  |  |
| --- | --- |
| seqwr | Sequential Write |
| seqrewr | Sequential Rewrite |
| seqrd | Sequential Read |
| rndrd | Random Read |
| rndwr | Random Write |
| rndrw | Random Read + Random Write |

* + 1. Test specific options:

|  |  |  |
| --- | --- | --- |
| Option | Description | Default Value |
| --file-num | No. Of files to create | 128 |
| --file-block-size | Block size to use in all I/O operations | 16 K |
| --file-total-size | Total Size of files | 2 G |
| --file-test-mode | Type of workload (supported I/O operations) | Required |

$ sysbench --test=fileio --file-total-size=1G --file-test-mode=rndrw prepare

$ sysbench --test=fileio --file-total-size=1G --file-test-mode=rndrw run

$ sysbench --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup

# Docker Installation

# Ubuntu 20.04 with SysBench on Docker

# Experiments

## QEMU (System Virtualization)

The experiment involves multiple test cases where there are variations in CPU and RAM allocated to the VM. Also, 3 test modes of SysBench will involve 2 variations each. Thus, the variables and their variations are listed below:

1. VM – No. Of CPUs: 2, 4
2. VM – RAM allocated: 2, 3
3. SysBench Test Mode – CPU:
4. SysBench Test Mode – Memory:
5. SysBench Test Mode – Fileio:

These experiments will be done on 2 different types of QEMU image formats – raw, qcow2.

### Image Format: raw

### Image Format: qcow2

## Docker (OS Virtualization)

# Analysis

# Vagrant

# Automation

# Conclusion