CS553 HW #2

Benchmarking – BareMetal, Containers, and Virtual Machines

Instructions:

- *Assigned date: Wednesday January 31st, 2024*
- *Due date: 11:59PM on Friday February 9th, 2024*
- *Maximum Points: 100%*
- *This is an individual assignment*
- *Please post your questions to BB*
- *Only a softcopy submission is required; it will automatically be collected through GIT after the deadline*

1 Your Assignment

This project aims to teach you how to understand the overhead of various virtualization technologies through benchmarking as well as familiarize yourself with operating Linux environment on the cloud. You can be creative with this project. Since there are many experiments to run, find ways (e.g. scripts) to automate the performance evaluation. You might find a combination of bash scripting along with tmux/screen helpful. You can use any Linux distribution for this assignment, but you must make sure your program runs and the results are re-producible on **Ubuntu Linux 22.04 on the Chameleon Cloud**.

In this project, by using **sysbench** and **iPerf**, you will perform strong scaling studies for each of the benchmark types: ***CPU***, **Memory**, **Disk** and **Network**. Strong scaling studies mean you will set the amount of work (e.g. the number of instructions or the amount of data to evaluate in your benchmark) and reduce the amount of work per thread as you increase the number of threads. You must incorporate bash scripting for: environment (baremetal/container/VM) orchestration, run benchmark and Python to analyze the results and plotting.

Commands you can use to install tools on Ubuntu 22.04:

- sysbench: sudo apt install sysbench
- iPerf: sudo apt install iperf

Documentations:

- sysbench: https://manpages.ubuntu.com/manpages/jammy/man1/sysbench.1.html
- iPerf: https://manpages.ubuntu.com/manpages/jammy/man1/iperf.1.html

CPU:

- Strong scaling studies: Fixed prime numbers limit at 100,000. Then, measure the performance of each virtualization technologies when varying the number of threads.
- Sample command (you might need to use additional command line arguments):
  $ sysbench cpu --cpu-max-prime=100000 --threads=1 run
- Fill in the below using benchmark results of each scale regarding the processor performance:
  Note that the efficiency denotes a relative performance of a virtualization type vs. baremetal. EX:
    - Baremetal: 10 events per second
    - Container: 9 events per second
    - VM: 8 events per second

  This translates to the efficiency of:
    - Baremetal: 100%
    - Container: 90% (Container is 10% slower than Baremetal)
    - VM: 80% (VM is 20% slower than Baremetal)

Virtualization Type	Threads	Avg. Latency (ms)	Measured Throughput (Events per Second)	Efficiency
Baremetal	1			
Container	1			
Virtual Machine	1			
Baremetal	2			
Container	2			
Virtual Machine	2			
Baremetal	4			
Container	4			
Virtual Machine	4			
Baremetal	8			
Container	8			
Virtual Machine	8			
Baremetal	16			
Container	16			
Virtual Machine	16			
Baremetal	32			
Container	32			
Virtual Machine	32			
Baremetal	64			
Container	64			
Virtual Machine	64			

Memory:

- Strong scaling studies: Fixed total data size in memory at 120GB. Then, measure the performance of each virtualization technologies with the following specifications:
  a. Block size: 1KB i.e., 2^{10} to 2^{20} bytes
  b. Operations: Read
  c. Access pattern: Random
- Sample command:
  $ sysbench memory --memory-block-size=1K --memory-total-size=120G --threads=1 run
- Fill in the below using benchmark results of each scale/type regarding the memory performance: Similar to efficiency example in CPU benchmark, the efficiency denotes a relative performance of a virtualization type vs. baremetal.

Virtualization Type	Threads	Block Size (KB)	Operation	Access Pattern	Total Operations	Throughput (MiB/sec)	Efficiency
Baremetal	1	1	Read	Random			
Container	1	1	Read	Random			
Virtual Machine	1	1	Read	Random			
Baremetal	2	1	Read	Random			
Container	2	1	Read	Random			
Virtual Machine	2	1	Read	Random			
Baremetal	4	1	Read	Random			
Container	4	1	Read	Random			
Virtual Machine	4	1	Read	Random			
Baremetal	8	1	Read	Random			
Container	8	1	Read	Random			
Virtual Machine	8	1	Read	Random			
Baremetal	16	1	Read	Random			
Container	16	1	Read	Random			
Virtual Machine	16	1	Read	Random			
Baremetal	32	1	Read	Random			
Container	32	1	Read	Random			
Virtual Machine	32	1	Read	Random			
Baremetal	64	1	Read	Random			
Container	64	1	Read	Random			
Virtual Machine	64	1	Read	Random			

Disk:

- Strong scaling studies: Fixed total data size on disk at 120GB. Then, measure the performance of each virtualization technologies with the following specifications:
  a. Number of files: 128
  b. File block size: 4,096 bytes
  c. Total file size: 120GB
  d. Test mode: Random Read
  e. IO Mode: Synchronous
  f. Extra IO flag: DirectIO
- Sample commands:
  $ sysbench fileio --file-num=128 --file-block-size=4096 --file-total-size=120G --file-test-mode=rndrd --file-io-mode=sync --file-extra-flags=direct --threads=1 <prepare/run/cleanup>
- Fill in the below using benchmark results of each scale/type regarding the I/O performance:
- Similar to efficiency example in CPU benchmark, the efficiency denotes a relative performance of a virtualization type vs. baremetal.

Virtualization Type	Threads	Block Size (KB)	Operation	Access Pattern	I/O Mode	I/O Flag	Total Operations	Measured Throughput (MiB/s)	Efficiency
Baremetal	1	4	Read	Random	SYNC	DirectIO			
Container	1	4	Read	Random	SYNC	DirectIO			
Virtual Machine	1	4	Read	Random	SYNC	DirectIO			
Baremetal	2	4	Read	Random	SYNC	DirectIO			
Container	2	4	Read	Random	SYNC	DirectIO			
Virtual Machine	2	4	Read	Random	SYNC	DirectIO			
Baremetal	4	4	Read	Random	SYNC	DirectIO			
Container	4	4	Read	Random	SYNC	DirectIO			
Virtual Machine	4	4	Read	Random	SYNC	DirectIO			
Baremetal	8	4	Read	Random	SYNC	DirectIO			
Container	8	4	Read	Random	SYNC	DirectIO			
Virtual Machine	8	4	Read	Random	SYNC	DirectIO			
Baremetal	16	4	Read	Random	SYNC	DirectIO			
Container	16	4	Read	Random	SYNC	DirectIO			
Virtual Machine	16	4	Read	Random	SYNC	DirectIO			
Baremetal	32	4	Read	Random	SYNC	DirectIO			
Container	32	4	Read	Random	SYNC	DirectIO			
Virtual Machine	32	4	Read	Random	SYNC	DirectIO			
Baremetal	64	4	Read	Random	SYNC	DirectIO			
Container	64	4	Read	Random	SYNC	DirectIO			
Virtual Machine	64	4	Read	Random	SYNC	DirectIO			

Network:

- Strong scaling studies using one server vs. *N* number of clients. Measure the performance of each virtualization technologies with the following specifications:
  a. Server TCP window size: 1MB
  b. Client TCP write buffer size: 8,192KB
  c. Client TCP window size: 2.5MB
  d. Naggle algorithm: Off
- The configuration of client/server should communicate using TCP over local loopback.
- Sample commands:
  $ iperf -s -w 1M
  $ iperf -c 127.0.0.1 -e -i 1 --nodelay -l 8192K --trip-times --parallel 1
- Fill in the below using benchmark results of each scale/type regarding the I/O performance:
- Similar to efficiency example in CPU benchmark, the efficiency denotes a relative performance of a virtualization type vs. baremetal.

Virtualization Type	Server	Client Threads	Latency (ms)	Measured Throughput (Gbits/s)	Efficiency
Baremetal	1	1			
Container	1	1			
Virtual Machine	1	1			
Baremetal	1	2			
Container	1	2			
Virtual Machine	1	2			
Baremetal	1	4			
Container	1	4			
Virtual Machine	1	4			
Baremetal	1	8			
Container	1	8			
Virtual Machine	1	8			
Baremetal	1	16			
Container	1	16			
Virtual Machine	1	16			
Baremetal	1	32			
Container	1	32			
Virtual Machine	1	32			
Baremetal	1	64			
Container	1	64			
Virtual Machine	1	64			

Other requirements:

- You must clearly state which type of Chameleon instance & site name was used for your experiment as well as other hardware specification details regarding your instance such as CPU, Memory, Disk and Network card. For example:
    - Chameleon Instance: compute_haswell at CHI@IIT
    - CPU: 2x Intel® Xeon® E5-2670 v3 @2.30GHz
    - Memory: 8x 16GB (128GB) of DDR4-2,133 ECC Registered RAM
    - Disk: 1x Seagate ST9250610NS SATA 7,200 RPM HDD
    - Network: Broadcom NetXtreme II BCM57800 1/10 Gigabit Ethernet
- Screenshots of your container and virtual machines setup are also required.
- Include raw output logs containing the output of each of a benchmark that you conducted
- For each benchmark, raw output data must be summarized into tables and graphs

2 Where you will submit

You will have to submit your solution to a private git repository created for you at https://classroom.github.com/a/4oeFqbG6. You will have to firstly clone the repository. Then you will have to add or update your source code, documentation and report. Your solution will be collected automatically after the deadline. If you want to submit your homework later, you will have to push your final version to your GIT repository and you will have let the TA know of it through email. There is no need to submit anything on BB for this assignment. If you cannot access your repository contact the TAs. You can find a git cheat sheet here: https://www.git-tower.com/blog/git-cheat-sheet/

3 What you will submit

When you have finished implementing the complete assignment as described above, you should submit your solution to your private git repository. Each program must work correctly and be detailed in-line documented. You should hand in:

1. **Source code:** All of the source code, including proper documentation and formatting.
2. **Readme:** A detailed manual describing the structure of your files and directory organization. The manual should be able to instruct users how to run the program step by step. The manual should contain example commands. This should be included as readme.txt in the source code folder.
3. **Report:** A written document (typed, named hw2-report.pdf) describing the overall assignment completion, along with screen shots and text answering the key questions. Make sure to label answers with the appropriate question number from the homework writeup.
4. **Logs:** Raw logs containing outputs of each benchmark that were conducted

Submit code/report through GIT.