Cloud Computing (CS-G527)

Assignment #2

4.2 Docker Checkpointing
4.2.1 Analysis of Image size and Checkpointing time

System details

Hardware Configuration

The system used for this task has the following configuration:

- Intel i7-3770 clocked at 3.4GHz
- 8 GB RAM

Operating System

Ubuntu 18.04

Softwares

- Docker 19.03.3
- CRIU 3.6

Dockerfile

For both tasks, a custom docker image has been built. To build the image, a Dockerfile has been used where the base image, the program to run and commands or parameters to run the program are specified. The latest version of Python, that is, Python 3.8.0 has been used as the base image. Two Python programs have been created for the respective tasks and have been specified in the Dockerfile with the run commands and parameters.

Task 4.2.1.1

Python Program

A Python program to print the next Fibonacci number every 4 seconds has been developed and used in the Dockerfile.

Shell Script

A shell script is used to automate the following tasks in the same order as specified below:

1. Create the image
2. Create and run the container
3. Print logs
4. Create checkpoint
5. Stop the container
6. Restart the container from the previously created checkpoint
7. Remove the checkpoint
8. Stop and remove the container
9. Remove the image

Results

The program printed the Fibonacci numbers till it was stopped abruptly. After stopping the program, it was resumed from the time the checkpoint was created by restoring the checkpoint explicitly.

Observations and Inferences

The program freezes when CRIU creates the checkpoint and resumes when the checkpointing is finished. It is to ensure that a process does not change its state or create new processes during the checkpoint creation.

Task 4.2.1.2

Python program

A Python program has been developed which takes ‘Number of child processes’ as an input and creates the specified number of child processes. These child processes just print their process id in an infinite loop.

Shell Script

A shell script is used to automate the following tasks in the same order as specified below:

1. Build an image.
2. Create and run a container with specified number of child processes(1, 5, 10, 15 ... , 50).
3. Create Checkpoint of the container and determine the time required to checkpoint.
4. Stop the container.

5. Restore the container and determine the time required to restore the checkpoint.
6. Repeat steps 2 to 5, five times for each image size and calculate the average of the metrics.
7. Delete the checkpoints, containers and images and store the metrics in a .csv file.

Results

1. The Application image size remains constant (918 MB).
2. The metrics ‘Checkpoint time’ and ‘Restore Time’ are plotted on the below line graph with ‘Number of child processes’ on the x-axis and time on the y-axis.

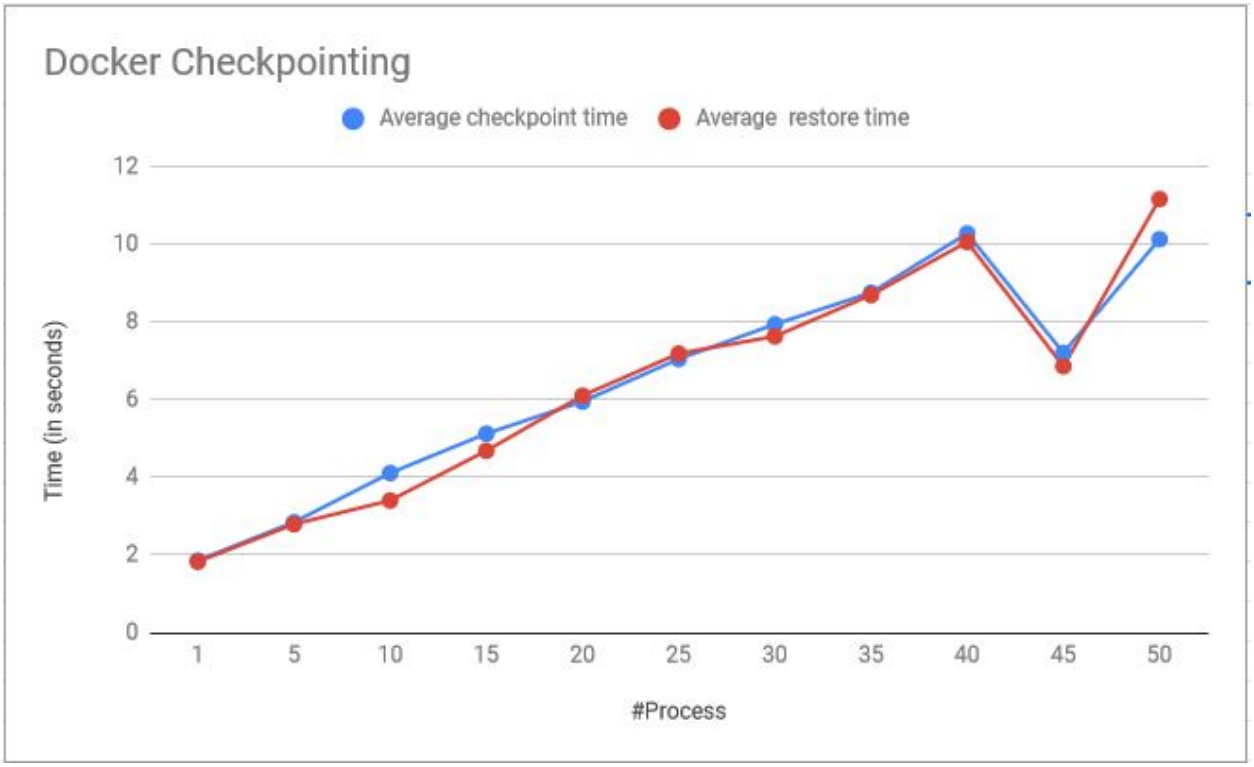

Fig - 1

Observations and inferences

1. The image size does not change because the parameter ‘number of child processes’ is associated with the image’s containers and not the image itself. The configuration of the image does not change by changing the number of child processes. It has the same Python base image and the same Python program and thus the same size.
2. It can be determined from the above figure that the average checkpointing time and the average restore time increases linearly (except an anomaly at #process = 45) with the increase in the number of processes.

The reason behind this is that the checkpointing procedure relies on **/proc** file system. The /proc file system provides a convenient and standardized method for dynamically accessing process data held in the kernel. CRIU uses /proc file system to obtain process data like the files descriptors information, pipes parameters and memory maps. CRIU then collects the process tree and freezes it. It then parses through the directory, collects the task's resources and writes them to the dump files.
The checkpointing process involves a lot of changes to the memory data structures like allocating memory for its own process, changing the memory maps, changing the contents of the Instruction Pointer register etc.
As the number of processes are increased, the changes in the memory, the size of the process tree and contents of the /proc directory also increase. Therefore, it can be inferred that there is a strong correlation between the memory changes and checkpointing/restoring time.

3. It can also be inferred from the above graph that the average checkpointing time and average restore time are directly related to each other, that is, restoring time increases when the checkpoint time increase. This happens because CRIU needs to perform reverse of the checkpointing steps like restoring basic task resources, forking the process tree etc.

Conclusions

1. The checkpointing/restoring time increases with increase in number of processes.
2. The restoring time is directly proportional to the checkpointing time.