ASSIGNMENT 4 Report

Background,

Docker Background:
Docker is a popular platform for developing, shipping, and running applications inside containers. Containers provide a lightweight and efficient way to package applications and their dependencies into a portable unit that can run consistently across different computing environments, from development laptops to production servers. Docker uses containerization technology to create isolated environments for applications to run, using Linux kernel features like namespaces and control groups (cgroups).

Containers created with Docker encapsulate the application and all its dependencies into a single, executable package. This package includes everything needed to run the application, such as code, runtime, libraries, and configuration files. Docker images are used to build these containers and can be easily shared and deployed across different Docker hosts.

CRIU Background:
CRIU (Checkpoint/Restore In Userspace) is a software toolset that allows you to freeze a running application (or process) in its current state and checkpoint it to disk. Later, this checkpoint can be used to restore the application back to the exact same state, allowing for process migration or restarts. CRIU operates entirely in user space and does not require special kernel patches.

CRIU can be used for various purposes, such as live migration of processes or containers between hosts, process debugging, system call tracing, and more. It provides a powerful mechanism for capturing and restoring the state of a running process or application, including its memory, open file descriptors, network connections, and other resources.

Problem Description,

The aim is to migrate a docker container. Using docker primitive

Details–-design details,

We have a shared location using NFS and using it to share the checkpoints.

The host checkpoints and stores the checkpoint in the NFS, and the potential host then picks up the checkpoint from the NFS and resumes the execution of the container.

We checked the checkpointing for a container running

- One process
- Multiple process
- A server

And were able to resume the execution after checkpointing.

Experiments

Since we are using a kind of stop-and-restore method, a good metric will be down time measurement, which we define as after the checkpoint time to the time it got resumed in a new container.

To checkpoint, we have used the following code

```
docker run -d --name looper --security-opt seccomp:unconfined busybox
\
         /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1);
sleep 1; done'


sleep 10
# wait a few seconds to give the container an opportunity to print a
few lines, then
docker checkpoint create --checkpoint-dir=/home/roy_linux/cs_a4
looper chk2
#cp -r ~/cs_a4/chk2/ ~/live_migration_checkpoint/
#rm -rf chk2
# check your container & print log file
docker ps
docker logs looper

docker logs looper > logs.txt
```

```
docker stop looper
docker rm looper
```

Plot

Discussion.

When we were trying to checkpoint a running server we were able to checkpoint it successfully. A probable;e reason for that is the docker was not able to handle an active connection using the port. But once the connection stops, we are able to checkpoint and resume from that point only