International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

Container Live Migration using Docker Checkpoint and Restore

Sankaran. S[1], Dr. C. Vijayakumaran[2], Mr. Karthick Nanmaran[3]

[1,2,3]*Department of Computer Science and Engineering, SRM Institute of Science and Technology Chennai, India*

Abstract – The container-based virtualization makes it easy to host application containers these containers run within the same kernel with isolated resources. This isolation makes container independent of the underlying hardware and operating system which allows the container to move freely between host machines, the movement of the container from one machine to another is known as migration. The increased use of the container has demand scalability and flexibility of the application container and also on demand service with lesser downtime. The application containers are basically an isolated process, these process state can be saved and restored to the same machine or to another machine. In this paper, we presented the experimental setup for containerizing the RYU SDN Controller and Mininet (SDN network emulator) and perform Live Migration using these containers. Docker is used to containerizing the application into application container and CRIU project is used to create the checkpoint for the container state and restore the container to the same host or to the different host machine.

Keywords : *Container platform, Virtualization, Migration, Docker, CRIU, SDN (Software Defined Networking), Mininet.*

1. Introduction

In the early years, we use a less capable system that can perform only limited functions so the hardware cost was really higher but today we have powerful machines that are capable of performing larger tasks in lesser time and the infrastructure cost is reduced way lesser when compared to the early periods. The resources available in servers are abundant they can be used for another process along with the running process. The virtualization concept was proposed in the early period but it was not popularized because of the cost of the infrastructure was really high but today we can use the virtualization to our advantage because of powerful machines even home computer is capable of running a huge process.

Virtual machines are introduced in the later period that is capable of emulating the physical machine and works as a physical machine in an isolated environment. These physical machine resources are partitioned into multiple virtual machines and they are isolated from one another. Each machine runs on its own hardware allocated to it form the physical machine and also has its own operating system. The virtual machines have eliminated the use of multiple physical machines and also the space required for the physical machine. From the server, multiple virtual machines instance can be created and deployed to the end user. The Virtual machines images are depended to the platform and also each VM runs on their own operating system which is an overhead for the virtual machines. The virtual machines have introduced cloud-based architecture in virtualization. The various services are provided over the cloud with the virtual machine by providing the end user with the control over the virtual machine with limited access.

The virtual machines with access control over the layer of the virtual machine provided with the services like

- Infrastructure as a service
- Platform as a service
- Software as a service

Containers are developed in order to overcome the problems faced in virtual machines, the containers have operating system level abstraction whereas virtual machine has hardware level abstraction which eliminates the use of multiple operating systems. Containers are an isolated process that runs on the same kernel the construct of the container is similar to that of virtual machines they are also known as machine containers. These machine containers are similar to virtual machines without the hardware layer. The Linux containers are similar to the machine containers which provides an isolated environment within the Linux operating system. The application container changed the cloud services into machine-based to application based architecture. The application container is independent of the underlying operating system and hardware. The capability of the container to migrate is examined in this experimental setup the basic approach of migrating the container would be stopping the container and resuming it after migrating it to another host but in live migration, the container migration should be seamless to the end-user.

Virtual machines are one class of virtualization with the hardware level abstraction the hypervisor supports migration of virtual machines whereas in the container it gets complicated because the container is a process so we are basically performing process migration.

2. Container Engine

Container engines are used to provide operating system level abstraction the docker uses docker-ce to create container platform whereas LXC doesn't have container engine it is inbuilt functionality of Linux system LXC is a feature for Linux operating system. Docker creates application containers which are different from LXC it is machine container, the docker creates container upon docker image. The container engine is similar to the hypervisor in the virtual machines which creates operating system level abstraction.

2.1 LXC\LXD:

The LXC is a machine container also known as Linux containers which creates the isolated Linux environment feels like a separate machine running within the same kernel. LXC is similar to the virtual machine but it does not have a hardware layer. The Linux operating system provides with the isolated Linux environment with to run applications, LXC allows multiple containers to run within the same kernel. LXC mimic the functionality of the virtual machine with lesser overhead this provides efficient use of Linux kernel.

2.2 Docker:

Docker is a container platform used to the containerized application, application container is a package of application, libraries and configuration file they are independent of underlying operating system and hardware. The Docker platform is used to create a containerized application by a building docker image. The docker image is built using docker file which has a set of instruction to build the containers, the docker file has instruction to create an environment for the application to run along with the dependencies. LXC run directly on the top of Linux kernel whereas docker container runs on the top of container engine.

3. Container Orchestrator

The containers are lightweight and multiple instances of containers can be deployed on the system and also on the virtual machine, to manage the multiple instances the container orchestrator is used. The container orchestrator provides the solution to manage the container instance on the dashboard or terminal even with the larger containers network the complexity of managing the container can be solved using the container orchestrator. Each container platform has its own orchestrator, Docker has Docker Swarm and LXC has LXD and kubernetes is an opensource container orchestrator which works with docker container it is similar to docker swarm and also has better functionality than docker swarm.

3.1 Kubernetes

Kubernetes is an opensource project for container orchestration developed by Google it helps in managing the Docker containers. Kubernetes helps to perform automation in container deployment and scaling of containers, it can be deployed on-premises for cloud services. Kubernetes can also work with other container platform and docker with kubernetes provides flexible container management services and automation. Kubernetes provides the platform to work with containers not only docker containers but also with other containers, it gives means to do deployments, scaling and monitoring of containers. Kubernetes container cluster will monitor all the containers if any container goes down it tries to heal the container by itself or redeploy the container. In this experiment, container orchestrator is not required as we use two containers but if we have to handle multiple containers the orchestrator like kubernetes comes in handy.

3.2 Docker Swarm:

Docker swarm is also a container orchestrator developed by Docker, Inc. The Docker swarm is used for container cluster management provides the ability for administrators and developers to manage the containers. It helps with managing the group of containers, the docker swarm has swarm manager node and Docker container node the swarm manager takes care of the docker container nodes. A swarm is basically a group of containers forming a cluster. The docker swarm also helps in monitoring all the containers health and ensuring all containers are up and running scaling the containers and updating the changes in the containers.

4. Container Deployment

The Virtual machines isolated machines that emulate the function of the physical machine and works just like a physical machine inside the physical machine. Each virtual machine has a virtual hardware layer allocated to the virtual machine, running multiple virtual machines can degrade the performance of the physical machine. The virtual machine is a package of hardware, operating system, and application whereas the containers are lighter when compared to a virtual machine with lesser overheads.

Docker Container platform is software used to create the containerized version of the application. Docker allows the user to build the application in a layered approach where the user can write the docker file for building the

container the docker file includes the dependencies and environment for the application to build upon.

4.1 Experimental Setup

In this experimental setup, two virtual machines are created with the configuration that satisfies the requirement table mentioned in table 2 and table 3. Docker software is installed in both the machines to work with the container docker can perform several functions like build container image, create a container, and also destroy the container. To perform migration the Docker version should be version 13.1 and above and it should run in experimental mode, to enable experimental mode the experimental value should be set true in docker daemon.json file. CRIU is also installed in both the machines, Docker supports CRIU only if the experimental mode is set to true.

Table -1: Experimental setup

Host Name	Component to Install
master.docker.io	Docker, CRIU
node.docker.io	Docker, CRIU

Master Docker machine should have the minimum following requirements

Table -2: Master machine minimum requirement

Master-Docker-Machine	• Docker version 13.1 or higher • Minimum 2GB of RAM • Minimum 10GB of free disk space • Linux kernel 3.11 or above

Node machine should have the minimum following requirements:

Table -3: Node machine minimum requirement

Node-Docker-Machine	• Docker version 13.1 or higher • Minimum 2GB of RAM • Minimum 10GB of disk space • Linux Kernel 3.11 or above

Both the machine contains a docker image from which the container will be created we can also create a common repository where the machine can share the docker images, Migrating the docker image won't be an efficient way so we can pull the docker image from the common repository. But for this experimental setup, we can have required docker image on both the machines to create containers.

4.2 SDN Testbed:

The SDN Testbed is used to perform research on the software-defined network, the testbed consists of an SDN controller and Mininet to created emulated the SDN switches and host. The tradition network has decision logic and forward plane coupled together whereas in SDN Architecture the decision logic is centralized, it provides with a programmable network which is flexible and easy to manage the larger networks.

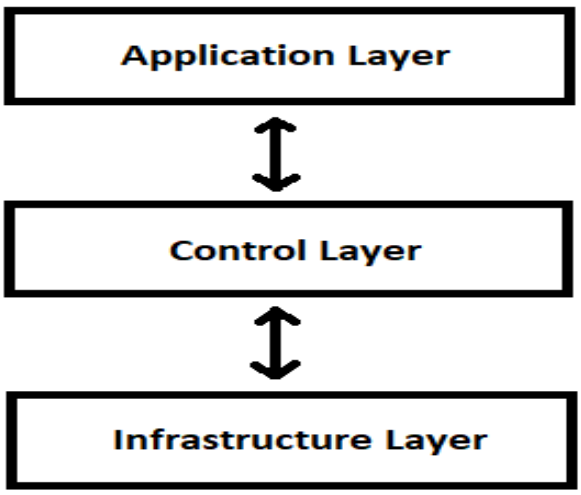

Fig -1: SDN Architecture

The physical SDN switches functionality is emulated in the Mininet we can create a emulate a simple network and connect the network with an SDN controller. The control layer consists of the SDN controller which is the brain of the network and the Infrastructure layer consists of all the network devices to forward data in response to controller decision.

In this experimental setup, the SDN controller is containerized using Docker and the network emulator software is also containerized to perform the migration. Basically, the virtual machines are used for the testbed but the virtual machine has a lot of overheads so in this setup the docker image is created for RYU controller and Mininet and migration of the container is performed.

5. Virtual Machine Migration:

A virtual machine is a software that emulates the function of the physical machine the virtual machine consists of emulated hardware and operating system and on top of it the applications of the virtual machine. A virtual machine allows the user to run multiple operating systems simultaneously in a single physical machine with multiple virtual machines inside it. A hypervisor is a software that creates one or more virtual machines and also monitors the virtual machines that are available, the server in which the hypervisor is installed is known as host machine from which the guest machines or virtual machines are created this is known as server virtualization. The hypervisor allocates the resources like CPU, RAM and hard disk space for the virtual machine from the available resource pool and handles the resource allocation conflicts between virtual machine. The virtual machine has isolation from the host machine and all from another virtual machine which can different operating system and also with the different application running on it. The hypervisor is controlling and monitoring the completely emulated machine, the system suspends and resume is supported entity for virtual machines. The hypervisor supports migration of virtual machine which is a solved problem the virtual machine entity can be suspended and migrated to the other host or same host and can be resumed on the same host.

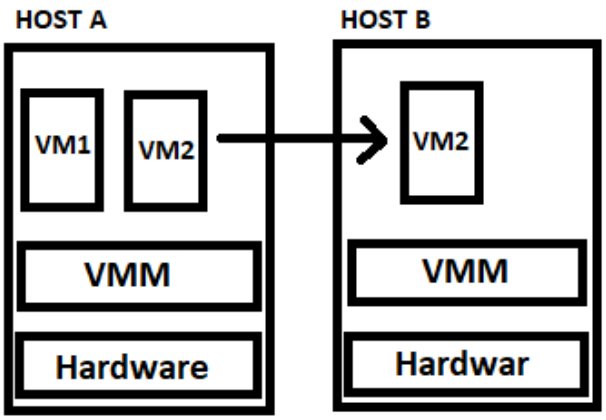

Fig -2: Virtual machine Live Migration

The migration involves several metrics like application downtime, total time is taken for migration and the data that has to be transferred. The total time for a migration is proportional to the size of data to be copied to the destination.

6. Container Live Migration:

Virtual machines are works, operates just like physical machines which actually emulates the hardware that imposes overheads which is very inefficient. Container machine is a construct which is exactly like a virtual machine without the hardware emulation layer, the container machine is isolated from the other process which makes it appear like a separate machine running on the same kernel on the same hardware. Container machine gives the full machine experience just like Virtual machine with lesser Overheads. This container machine is also similar to the Linux container LXC/LXD whereas Docker shrinks the envelope of the Linux container into a single application process. Docker container is basically an application process running in an isolated environment, it is a software package.

The **Container migration** is basically the movement of the container from one host to another host, it is mainly performed during hardware maintenance or upgrading the system to reduce the downtime. Live migration is the process of retaining the state of the container even after the migration, it should not interrupt the user work and change host should seamless to the end user. Most of the hypervisors support live migration this capability is used as an advantage by the virtual machine to migrate them in any situation but Virtual machines are one class of virtualization now that containers are widely adopted and growing eventually. OpenStack ecosystem also uses containers like Magnum, Kolla, Kubernetes as their project so their uses are growing there is an expectation that like virtual machines, the container also should have the capability to perform migrate. Hypervisor is controlling a completely emulated machine which has a capability to suspend and resume which solves the live migration problem in the virtual machine whereas containers are the process on Linux system that is isolated so basically the container migration is process migration, the process is migrated from the one Linux system to another which is not easy to perform until the development of CRIU project.

6.1 CRIU:

Checkpoint Restore In Userspace(CRIU) is used to freeze and restore the process state in the Linux system. For creating a checkpoint and restore the process state kernel should support this work is accomplished in Linux kernel 3.11, CRIU uses this capability to freeze all the process state which can be migrated to the other host and restore the process to continue from the frozen state. CRIU support is implemented on the docker engine 1.13 release and above, CRIU allows the user to create a checkpoint for the docker container process and the checkpoint metadata can be migrated to restore the container process on the different machine.

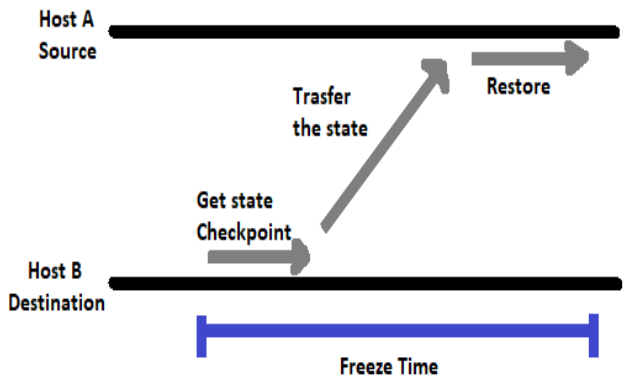

Fig -3: Container Live Migration

6.2 Case Study - Busybox Container Migration:

In this case study, a simple container is migrated from one machine to another the setup has docker of the same version on both the machine with experimental mode enabled and both machines should be within OpenShift cluster. Busybox Image is pulled on the host system the container image is available in the Docker Hub. OpenShift pulls the image from the Docker Hub image repository, Busybox is basically an executable file with the common Unix utilities. In this scenario the Busybox is programmed as a counter which simple counts from zero and goes on, the pod is created and the route is exposed for the container. The main idea is to migrate the Busybox container running in the pod in the master machine has to be migrated to the node machine. The container migration is performed using the open source project known as CRIU which creates process checkpoint by freezing the container state the process info, thread, memory and log file metadata is transmitted to the node machine. In the node machine, the metadata is used to initiate and restore the process state of the busybox container.

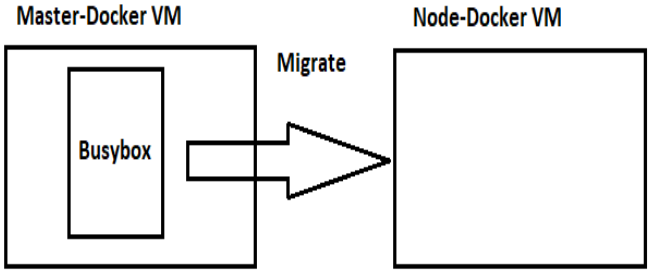

Fig -4: Busybox Container Migration Scenario

6.3 Case Study - SDN Testbed:

In this case study SDN test bed is created in a container environment which has the Ryu Controller container and Mininet Container. These two containers are custom built containers they are built using docker, the docker compose file is used to create a docker image from which the containers are created on top of the image. The docker image creates an isolated application from the same image multiple containers can be created and deployed. The Ryu docker image and Mininet docker image is present on both the machines the Ryu container created and it can be used to connect to virtual or real SDN network. The Mininet container is created, inside the Mininet container virtual SDN network is created with virtual switches and hosts. The Ryu controller is connected to the virtual network through the Ryu container IP address now the Mininet container is connected with the Ryu controller container. Both the containers are running in the Master machine now the Ryu container is migrated to the node machine without any connection interruption with the controller. The Ryu container process checkpoint is created and the container checkpoint metadata is migrated to the node machine and restore the container process to the end user it will be container changed between the nodes is unnoticeable with lesser downtime.

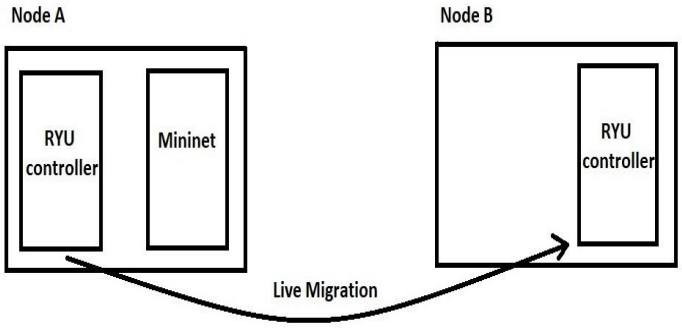

Fig -5: SDN Testbed Container Migration Scenario

7. Conclusion

The CRIU is used to create a checkpoint and restore the container process on the same machine or to another machine. The virtual machines are widely used all over but the overhead in virtual machine degrades the resource utilization and the migration required a lot of preprocessing and the files should be migrated without any network interrupts to have proper and successful migration the migration metadata is larger for the virtual machine when compared to the containers. The container image size is smaller when compared to the size of the virtual machine and multiple container shares the same kernel save the storage space. Even the migration metadata is comparatively less so the downtime for migration is also less for containers.

When creating the checkpoint, the downtime appears when it involves the file system for migration there can also be conflicts if the migrated container is assigned with the same IP address and port as the container

already present in the destination machine. The Application containers can replace many applications with the micro services and modular approach when these containers provided as a platform as a service using OpenShift

References

1. Bazel: {fast, correct}—choose two; http://bazel.io.
2. Burrows, M. 2006. The Chubby lock service for loosely coupled distributed systems. Symposium on Operating System Design and Implementation (OSDI), Seattle, WA.
3. cAdvisor; https://github.com/google/cadvisor.
4. Kubernetes; http://kubernetes.io/.
5. Metz, C. 2015. Google is 2 billion lines of code—and it's all in one place. Wired (September); http://www.wired.com/2015/09/google-2-billion-lines-codeand-oneplace/.
6. Schwarzkopf, M., Konwinski, A., Abd-el-Malek, M., Wilkes, J. 2013. Omega: flexible, scalable schedulers for large compute clusters. European Conference on Computer Systems (EuroSys), Prague, Czech Republic.
7. Verma, A., Pedrosa, L., Korupolu, M. R., Oppenheimer, D., Tune, E., Wilkes, J. 2015. Large-scale cluster management at Google with Borg. European Conference on Computer Systems (EuroSys), Bordeaux, France.
8. P. Purohit, R. Kadikar, M. Susila and B. Amutha, "Study of Service Chain Optimization in Cloud Environment," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai.
9. Sharma, A. Saxena and K. Nanmaran, "A Survey on Live Virtual Machine Migration," 2017 UKSim-AMSS 19th International Conference on Computer Modelling & Simulation (UKSim), Cambridge
10. Valluvan. S, T. Manoranjtham, Dr. V. Nagarajan, "A study on SDN controllers, International Journal of Pharmacy and Technology", Open Access Volume 8, Issue 4, December 2016, Pages 5234-5242 Scopus SNIP (.237)
11. OpenShift; https://docs.openshift.com/container-platform/3.9/upgrading/index.html
12. CRIU; https://www.criu.org/Main_Page
13. OpenVZ. http://openvz.org