High Availability of Network Service on Docker Container

Yikang Liu[1, a]

[1] School of Electronic Information and Electronic Engineering, Shanghai Jiao Tong University, China

[a]LiuYikang@sjtu.edu.cn

Keywords: Container, Virtual Network Function, High Availability.

Abstract. In recent years, a new technology, container, has been developed rapidly. More and more services are deployed on a container. In order to ensure the high performance and high availability of these services, a system which can provide the technology of high availability is essential. Container, as we know, is an approach to virtualization in which the virtualization layer runs as an application within the operating system, so more and more services are deployed on container technology, such as Docker [6], OpenVZ [4,5] and lxc. In this paper, we present an approach of high availability of some special services on container.

Introduction

Virtualization technology is a key technology of cloud computing system architecture [1], which can raise the utilization of physical resources. Operating-system-level virtualization is a server virtualization method in which the kernel of an operating system allows the existence of multiple isolated user-space instances, instead of just one. Such instances are named containers. A container can run like a process on the host, with independent CPU, memory, network resources. Unlike traditional virtualization use hypervisor to isolate and manage resources, container can use namespace and cgroup, both of them are Linux kernel feature, to isolate and control host resources.

High availability is an important feature in a data center, not only traditional data center but also cloud data center. In a cloud data center, virtualization technology is a basic technology to support the architecture, so in order to guarantee the cloud data center's disaster recovery capability, traditional virtualization technology has integrated many high availability features, such as monitor, migration and checkpoint/restore. Migration can move a virtual instance, like a virtual machine or a container, from one host to another without disruption, the most important indicator of migration is the migration time, which can affect the user's experience of the service. Checkpoint/restore is used to backup a virtual instance from one host to another, when backup a virtual instance, the checkpoint/restore process need to checkpoint the runtime state of the virtual instance, then send the state to backup host to record, when the original instance inaccessible, the checkpoint/restore process restore the instance on backup host using the record form original instance. Checkpoint/restore is very useful for the state sensitive service, for stateless service, the record file may be redundancy.

This paper presents a high availability approach for the stateless service in a container, and we choose Docker as our container engine. First, we will present the related work, then we propose our solution, at last, we give our performance evaluation, conclude the solution and propose the future works.

Related Work

Virtualization is an important technology in cloud computing, it refers to the act of creating a virtual version of resources, including virtual computer hardware platforms, operating systems, storage devices, and computer network resources. Traditional virtualization, known as hardware virtualization [3], is using hypervisor to manage the virtual machine. Each virtual machine has a full operating system, and each virtual machine is very complete isolation. But OS-level virtualization, container virtualization, is not a full isolation, all containers on a same host share the host operating system as

their operating system, and they use a union file system to share some basic file, the union file system is the key idea in this paper.

As we know, Hardware Virtualization platform, such as XenServer which is a famous hypervisor platform, offers high availability protection for virtual machine. One high availability technology for XenServer is Remus [2], which has been a part of Xen hypervisor. Remus achieves high availability by using an avtive/backup configuration [5], through the configuration. Remus can continuously checkpoint the state of the active VM to the backup VM. The backup VM using the checkpoint state of the active VM to maintain the consistency, when the active VM failure is detected, the backup VM can execute immediate replacement of the active VM.

OS-level Virtualization's high availability is different from Hardware Virtualization, it achieves the checkpoint/restore high availability needs a third-part software, which is CRIU [9]. CRIU, Checkpoint/Restore In Userspace, is a software tool for the Linux operating system, and it is an independent project of OpenVZ ,which is also a Container platform, using for OpenVZ container checkpoint/restore [4]. CRIU can checkpoint the container's user namespace, process tree and the image file to the disk, and with the checkpoint file, CRIU can restore the container state when the container is checkpointed. With this feature, CRIU can be used to implement container's live migration and active/backup high availability. And CRIU project is also can be used for Docker.

Proposed Approach

This paper also wants to implement an active/backup high availability. But different from traditional high availability, this paper using the feature of Docker container's file system coordinates with keepalived to implement state insensitive services high availability on Docker container.

State Insensitive Service

State insensitive service is the service which the runtime state does not affect the service state or the service can automatic recovery the runtime state, the typically state insensitive service is virtual network [10] service include virtual router, virtual firewall and so on, this service can use the mechanism itself to recovery the service, but they can not achieve host level high availability. In our solution, we just backup the read-write image layer of the container when there is a write signal on the read-write image layer.

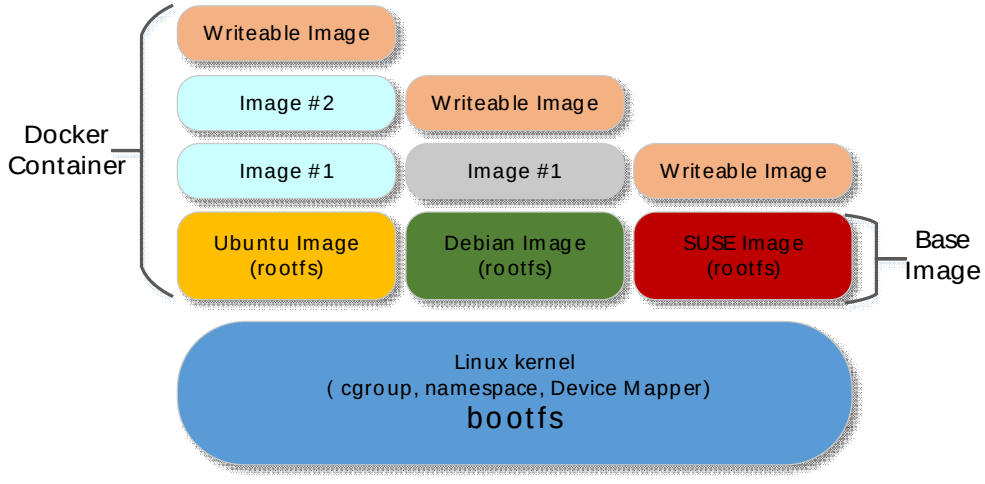

Figure.1 Docker Container Image Layer

Docker File System

As mentioned about, Docker uses the union file system [6] as the basic file system. Union file system allows files and directories of separate file systems, named as branches, to be transparently overlaid, forming a single coherent file system. The different branches may be either read-only or read-write file systems, in Docker only the topmost branch can be read-write file system, the others are all read-only file system.

As Figure.1 shows, different containers have different image layers, but they all have bootfs and rootfs layers. The bootfs contains the boot loader and the kernel. The boot file system can be never made any changes. The rootfs includes the typical directory structure with operating systems: /dev, /proc, /bin, /etc, /lib, /usr, and /tmp plus all the configuration files, binaries and libraries required to run user applications. Above the basic image layer, we can add different service image layer to support the service we need, and this image layer is read-only image layer. On the top of the image layers, is our service image layer also is the read-write image layer. This layer supports our solution of high availability of state insensitive service.

Keepalived

As we can continue backup the read-write image layer of the container, we need a mechanism to change the network collection to the backup host when the primery host is failure. In our solution, we use the keepalived [7] to achieve this goal. Keepalived is a software which can use VRRP protocol to achieve primery/standby host high availability. Keepalived can make two host use a same virtual IP to collect the network and maintains the heartbeat between the prime host and the standby host, if the prime host is failure, the standby host will take the place of prime host to collect to the network.

In our solution, both the prime and standby host are installing the Docker Engine, on the prime host, there are some active containers offer network service, and each active container will have a backup container on the standby host, and these backup containers are all shutdown. When there are write signal on one container's read-write image layer, we will backup the read-write image layer to the backup container's read-write image layer, then if the prime host is failure, the standby host will become the prime and the backup containers will become the active containers.

Experiment and Performance Evaluation

In our experiment, there are two independent hosts, and each host is installing Ubuntu14.04.4 system with Intel SandyBridge CPU (E5-2620) clocked at 2GHz, 8GB DRAM. Openssh is installed on each host to let them can send data to another, keepalived is used to maintain the heartbeat between two hosts (primary and standby). Docker engine 10.0 is installed on two hosts, and docker container is the object in our experiment, the base image we choose the ubuntu14.04.4, then we install virtual network function services in the container as our experiment container. Click modular [11] and quagga [8] is the software we used to design our virtual network services.

This paper compares the CRIU checkpoint/restore and our solution on container with the state insensitive services. Remus of XenServer is also a high availability solution, but virtual machine's image is too big and remus need the backup virtual machine running all the time, different from virtual machine, container can boot very quickly, so the backup container can be shutdown when the active container is running, when the active container is failure, the backup container start and maintain the service.

Table 1. Synchronized time and Recovery time of single container with different service

Service Image Size	Synchronized time		Recovery time	
	Checkpoint/Restore solution	Image layer solution	Checkpoint/Restore solution	Image layer solution
46MB	409ms	56ms	410ms	230ms
107MB	985ms	127ms	903ms	891ms
206MB	1967ms	320ms	1890ms	2034ms

Table 1 presents the synchronized time and recovery time of difference between Checkpoint/Restore solution and our Image layer solution. As we can see, the Image layer solution can reduce 80%-90% synchronizing time, that is because the image layer solution just synchronizing the image layer, the service state is not synchronized as this services are state insensitive services. But as the image size become bigger, the Image layer solution's recovery time become longer than the Checkpoint/Restore solution, that because the service states of the Image layer solution are recovery by the mechanism of

the service itself, as the image become bigger and bigger, the state recovery time become longer and longer, but the total time of the Image layer solution is shorter than Checkpoint/Restore solution.

Table 2 presents the average synchronized time and recovery time of different number of service on a same host. As we use the write signal to determine whether synchronise or not, we will not synchronizing all the container at the same time, so the average synchronized time is smaller than the number multiplies the single container synchronized time.

Table 2. Average Synchronized time and Recovery time of different number of service

Service Image number(46MB)	Synchronized time(Average)		Recovery time	
	Checkpoint/Restore solution	Image layer solution	Checkpoint/Restore solution	Image layer solution
10	1140ms	130ms	4035ms	2004ms
20	3150ms	530ms	9043ms	5500ms
40	8980ms	1120ms	15350ms	12300ms

Summary

A high availability solution of state insensitive service, typically virtual network service, on container was presented in this paper, using the feature of the union file system of Docker container, we can simply synchronizing and recovery the container, and the experiments show the performance of the solution. For future work, how to use the idea of this solution to recover the state sensitive service is our target.

References

[1] Kratzke N. A lightweight virtualization cluster reference architecture derived from open source paas platforms[J]. Open J. Mob. Comput. Cloud Comput, 2014, 1: 17-30.

[2] Cully B, Lefebvre G, Meyer D, et al. Remus: High availability via asynchronous virtual machine replication[C]//Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. 2008: 161-174.

[3] Raho M, Spyridakis A, Paolino M, et al. KVM, Xen and Docker: A performance analysis for ARM based NFV and cloud computing[C]//Information, Electronic and Electrical Engineering (AIEEE), 2015 IEEE 3rd Workshop on Advances in. IEEE, 2015: 1-8.

[4] Li W, Kanso A, Gherbi A. Leveraging linux containers to achieve high availability for cloud services[C]//Cloud Engineering (IC2E), 2015 IEEE International Conference on. IEEE, 2015: 76-83.

[5] Li W, Kanso A. Comparing Containers versus Virtual Machines for Achieving High Availability[C]//Cloud Engineering (IC2E), 2015 IEEE International Conference on. IEEE, 2015: 353-358.

[6] Docker on https://www.docker.com

[7] Keepalived on http://www.keepalived.org

[8] Quagga on http:// http://www.nongnu.org/quagga

[9] CRIU on https://criu.org

[10] Kohler E, Morris R, Chen B, et al. The Click modular router[J]. ACM Transactions on Computer Systems (TOCS), 2000, 18(3): 263-297.