Data Storage Management based on KVM and Docker

Leave Author List blank for your IMS2014 Summary (initial) submission.
IMS2014 will be rigorously enforcing the double-blind reviewing requirements.
*Leave Affiliation List blank for your Summary (initial) submission

***Abstract*—With the maturity and popularity of cloud computing technologies, more and more users choose to deploy their applications in virtual servers. Users have to consider several issues when deploying applications in virtual machines. If the virtual machine can not be used normally because of unknown reasons, how to deal with the files in the virtual machine? The capacity of the virtual machine's hard disk can not be changed, but more and more data in the virtual machine may cause the virtual machine to crash. How to dynamically expand the capacity of the hard disk is a problem that needs to be solved? Data sharing is often performed among tenants, but how to realize the data with authority sharing is a problem? This paper presents the corresponding solutions. Platform provides two modes of service, respectively alone-machine mode and multi-machine mode. Alone-machine mode uses single-tenant multi-bucket policy to separate the file data from the virtual service. Using this strategy can solve the data loss caused by the crash of the virtual server. It also addresses server crashes caused by the inability of virtual servers to dynamically expand hard disk space. Multi-machine mode uses multi-tenant multi-bucket strategy, not only can implement alone-machine mode functions, but also can solve the problem of sharing files with permission between tenants. Use KVM virtualization technology to provide tenants with virtual services and use Docker containers to deploy Minio services for data storage independence. Separating data storage from the virtual service environment can improve resource utilization and data security.**

***Index Terms*—Kernel based virtual Machine, Docker, LXC, Minio**

I. Introduction

With the development of cloud computing technology, more and more users choose to use virtual services to deploy applications, which can save costs and improve resource utilization. There are many cloud computing service platforms providing virtual service domestic and overseas, such as Alicloud, Qcloud, AWS and so on. In fact, in deploying applications, development teams often use Git to track code and use continuous integration(*CI*) and continuous delivery(*CD*) to automate project construction, testing, and deployment cycles to quickly interact with high-quality products. However, users deploying applications on virtual servers may not consider isolating the code from the data. So when users use the virtual service may encounter the following questions. The first problem is that the tenant deploys the application on the virtual machine. If the virtual machine fails, the files in the virtual machine can not be removed and can only be discarded. Discarding data in the big data era is a tremendous loss to the tenancy. The second problem is the amount of data. If use a machine learning method to Train a model, then the amount of data in the learning process is continuously accumulated. However, the capacity of a virtual machine is fixed, there is no way to meet the growing demand for storage of data. Therefore, how to implement a storage system that can be dynamically expanded and separated from the virtual machine is a problem that needs to be solved. The third problem is the sharing of data with authority. For example, When using the same dataset for different studies or using another dataset for further research, the copy method is not an efficient sharing strategy if the amount of data is large. And when sharing data, the owner of the data should set the permissions of the data in order to protect the data security. This paper presents two models to solve the above three problems respectively.

II. Technical introduction and choice

A. Virtualization technology

Virtualization technology can help us to strengthen IT resources into single unit that can be shared over a network and fulfill the shrinking resources and improve resource utilization[1]. The virtual machine is an abstract entity between the system hardware and the user, like a physical computer,runs an operating system and applications. Similar to the operating system's process management, each running process considers itself the only process running in the system. The virtual machine is running on the host computer by simulating hardware input and output at the software level. The goal of virtualization technology is simulate another operating systems execution environment on existing operating systems. Common virtualization technologies include Xen, kernel-based virtual machine(*KVM*)and Hyper-V.

Xen is an open source project developed by the Computer Lab at University of Cambridge. Xen inserts a virtualization layer between the system hardware and the virtual machine, translating the system hardware into a logical compute resource pool, and Xen dynamically allocates resources in the resource pool to the operating system or application. One of the basic principles of Xen system design is the separation of schemes and mechanisms. The Virtual Machine Monitor(*VMM*) in a Xen system is in a mixed mode. An authorized privileged VM exists in the Xen system to help the VMM system. The Xen program to make the schemes and the privileged VM to be responsible for the implementation of the mechanisms. A privileged VM licensed by Xen is called Domian0. Design principles that separate program and mechanism improve the flexibility and performance of Xen systems. Xen system core code is rare, this is because Xen runs at a privileged level higher than the operating system level, if there is a bug in the Xen kernel will endanger the safety of the entire system.

KVM is a full virtualization solution for the Linux kernel. The fundamentals KVM developers followed were the same as the Linux kernel: "Don't reinvent the wheel". KVM development did not create a hypervisor by changing the Linux kernel code. In contrast, KVM virtualization technology uses the Linux kernel as its hypervisor. Support for KVM virtualization has become the default part of the mainstream Linux kernel since 2.6.20. But KVM is just a module in the Linux kernel, and more tools are needed to manage and create a complete KVM virtual machine. QEMU is a powerful virtualization tools that virtualizes different CPU frameworks. KVM system uses the X86 part of QEMU, after development and transformation to form a user space tool that can control KVM kernel module QEMU. Although QEMU tools can create and manage KVM virtual machines, QEMU is inefficient and not easy for users to use. Therefore, RedHat has developed more tools for users such as libvirt, virsh and virt-manage. Tools libvirt provides a convenient, reliable, multi-language supported programming interface for a variety of virtualization tools. Users can connect to a host on a KVM or Xen system using functions provided by libvirt to control different virtual machines. In addition, virsh is a set of text-based virtual machine management command tools provided by libvirt. Users can use virsh to implement all the features of libvirt. Auxiliary tools virt-manage is a set of virtual machine management graphical interface software developed in the python language, that is implemented using the libvirt API for intuitive operation of virtual machines.

Hyper-V is a virtualization product based hypervisor. The purpose of Hyper-V is to provide a broader range of users with more familiar, cost-effective virtualization infrastructure software. Hyper-V supports full virtualized guests and paravirtualized guests. Paravirtualized guests requires that the virtual machine and the physical host have the same operating system, and full virtualized guests requires that the physical host's CPU support virtualization. Full virtualized can create virtual machines for different operating systems. But Hyper-V requires that the host system must be Windows.

Because this platform uses a Linux system, it does not use Hyper-V. Both KVM and Xen are open source virtualization technologies that have their own strengths and weaknesses. Xen uses a self-developed hypervisor to manage virtual machines and related resources but KVM directly uses the Linux kernel as a hypervisor. Since KVM is a special module in the Linux kernel, after the Linux kernel loads the module, it can turn the Linux kernel into a hypervisor so that KVM can well schedule and manage hardware resources. For the above reasons, the underlying virtual machine technology in this article uses KVM.

B. Container virtualization technology

Container technology is an operating system-level virtualization solution. Container can divide individual operating system managed resources into isolated groups to balance resource requirements. Compared with virtualization technology, container technology do not require instruction-level simulation and real-time compilation.Common container virtualization technologies include Linux Containers(*LXC*) and Docker. LXC is a Linux system lightweight container technology that includes resource management and security management. LXC uses the cgroup mechanism and namespace mechanism to implement resource management functions and security management in the Linux kernel. LXC can supports all the features and capabilities available in the Linux environment. Docker is an improved version of Linux Container. Docker encapsulates applications and dependencies into a container, implementing a project migration that requires the container to be started in new environment. Docker containers isolate software from its development environment, such as differences between the development environment and staging environment, which can reduce conflicts between running different software on the same infrastructure.

The main difference between container technology and virtualization technology is in their architectural approach. Virtualization technology is implemented at the hardware level and requires a hypervisor and virtual machine operating system that must load a complete operating system and virtualize system resources at startup. In contrast to virtualization, container technology is implemented at the operating system level and shares the operating system kernel with the host computer. It does not need to start a complete operating system at startup, so the container boots much faster than the virtual machine. Virtualization technology provides an operating system environment, that is the same as the actual production environment, mainly used to build applications or environments. Traditional virtualization technology usually uses block storage to implement virtual machines data storage, but can not separate the data from the operating system. With the growing amount of data, data storage should be independent of operating system and can be dynamically mounted to other virtual machines. However, the traditional virtualization technology can not realize the dynamic storage of data. Docker container technology and Minio unstructured object storage technology provide a new way for us to solve the problem of mounting remote data to a virtual machine.

C. Storage technology

Traditional storage methods are block storage and file storage, but both storage methods have their own advantages and disadvantages, the emergence of object storage, a good combination of block storage and file storage advantages. Object storage introduces two concepts object and object-based storage device(*OSD*), which object contains the file data and related properties and OSD is a collection of objects. Object contains four content, which is ID, data, metadata and property. ID is used to mark the object, the data is the subject of the object, the metadata used to describe the object storage and storage location, the property is defined according to the needs of other data describing the object. Objects can maintain their own property, simplifying the storage system management tasks and improving system flexibitigonglity. The size of the object can be different, the object can even contain

the entire data structure, such as files, databases and so on. The bottom data of the traditional storage methods are organized in a block form, and the data underlying the data pair is found through a tree structure. However, traditional storage takes a long time to find a large number of small files. Object storage uses Hash indexing method, the direct position of data storage location, without layer-by-layer search. Object storage provides Key-Value RESTful data read and write interfaces, that often provides data access in the form of web services. Through the use of HTTP requests PUT, GET, DELETE to complete the file upload, download and delete operations. Compared with the file system, the object storage represented by AWS S3 and Swift has two notable features: a RESTful interface and a flat data organization structure. The directory tree can introduce huge spending to the storage system. The flattened data organization of the object store eliminates the need to maintain a large directory tree by dropping nested folders. The flattened data organization of the object store avoids the need to maintain a large directory tree, so lookups are fast.

Minio is an open source project by Anand Babu Periasamy. Minio is a distributed object storage server, written in Goand open sourced under Apache License Version 2.0. AWS S3 has become the standard for object storage. Minio implements AWS S3 v2/v4 Application Programming Interface. So users can access the Minio server using the Minio SDK, the Minio Client, the AWS SDK, and the AWS CLI. Minio using erasure and bitrot detection to prevent hardware failure. User can set Minio server to continuously mirror data between Minio and any Amazon S3 compatible server. Minio provides confidentiality, integrity and authenticity assurances for encrypted data with negligible performance overhead. The Minio server provides perfect technical support for Docker container technology. Use Minio Docker's image to quickly launch the AWS S3 compatible object storage server. Minio server provide object storage API access to Docker volumes. Attach Docker volumes to Minio containers and access data with REST API. Minio can be used as a cloud storage solution to save massive images, videos and documents.

Due to Minio's advantages, the platform's storage implements the way Docker deploys Minio. When multiple clients exist, each user needs to turn on a Minio service, which is inconvenient for platform management. Therefore, the docker technology is used to deploy the Minio service and docker is used to manage the Minio container to implement the multi-tenant multi-Bucket mode.

III. Platform Management Model And Architecture

A. Platform management architecture

In the platform management architecture model, there is a dedicated network cable between the VM server and the Minio server for data operations, and each server has external network access. Use KVM virtualization technology on VM servers to provide tenants with virtual services. Deploy the Minio service with a Docker container on a Minio server to provide data storage services for the purpose of separating data from virtual machines. Create a docker container for each user in the Minio server to use the minio service. There is also a special container containerX in the Minio server for storing tenant shared data. All containers in the Minio server are managed by the daemon. There are two kinds of Application Programming Interface(API) for tenant operation data in the Minio server. The Minio API is responsible for providing services for tenants to operate their own data. The Share API provides services for sharing data with permission between tenants.

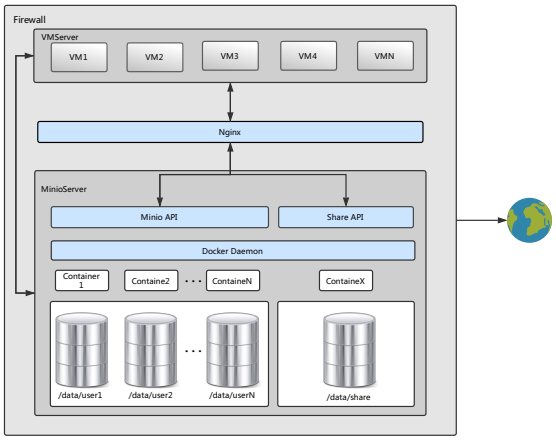

Fig. 1. Platform management architecture

B. Alone-machine mode

In alone-machine mode, the tenant deploys the application in a virtual machine and the following two problems may occur while using the virtual machine.

- If a virtual machine fails, the files in the virtual machine will not be available and will continue to be used.
- Because the virtual machine's hard disk space does not change, if the data in the virtual machine becomes more and more, it may cause the virtual machine can not be used normally.

In order to solve the above two problems, this paper presents a solution. In alone-machine mode, only one tenant is using virtual machine services, so use docker container technology to open a Minio service for the tenant. This will be separated from the data and code in the application, improved data security and portability. Even if the virtual server crashes for other reasons, it will not lose data. This solution also solves the problem of virtual machine crashes caused by too much data volume, so that tenants need not worry about the shortage of virtual hard disk space. Provides the tenant with a storage policy that can be dynamically expanded and separated from the virtual server. When tenants want to share data with other servers, they only need to use the interface provided by the Minio service and do not need to perform operations

such as copy and paste, thus saving computing resources and improving work efficiency.

C. Multi-machine mode

Similar to alone-machine mode, tenants deploy applications to virtual servers in multi-machine mode. Unlike alone-machine mode, multiple users share the same service in multi-machine mode. As multiple tenants use the service together, there are two problems below.

- Multiple tenants using virtual services will open multiple Minio service processes, how to manage Minio service processes on the server. If the Minio service process is not properly managed, it will cause the tenant's virtual service to fail or even crash.
- When a user frequently reads large amounts of data, the file I / O is too large and the bandwidth resource is occupied, and the server or intermediate network may be crashed and the tenant's virtual machine will not work normally.

This article presents a solution to the above two problems. In multi-tenancy mode, the platform needs to provide one Minio service for each tenant, which means that multiple Minio service processes need to be run on the server. The management of a large number of Minio server processes is a burden on the server, reducing server resource utilization. So we used Docker container technology to deploy the Minio service. Because Docker comes with its own container management technology, the Minio service can be easily managed using the Docker API. This will not be due to improper management of the Minio service process caused that the tenant's virtual server is not working properly. In order to solve the problem of the hard disk reading too fast leading to the server crashes, we added a network connection between the servers for requesting the Minio service to alleviate the bandwidth resource consumption.

Whether alone-machine mode or multi-machine mode, there are tenants have permission to share data problems. How tenants have permissions to share their own data to other tenants, and to ensure data security. Minio service can only set the file read and write permissions, but sometimes tenants just want to share part of the data to other tenants instead of all the data. In fact, each user has a Minio service. Users can create multiple buckets to store data in Minio service. After setting permissions for the pre-shared buckets, data sharing can be achieved simply and efficiently through the Minio API service.

IV. CONCLUSIONS

This management design meets our actual production needs and will be used in a variety of complex virtual machine environments such as Tensorflow and Spark in today's increasingly large data sets.

Virtual machine hard disk space will never be enhanced, only need to be able to accommodate the minimum computing environment or application resources.When deploying an application, data storage should be kept separate from the development environment so that the service delivers maximum effectiveness and security.

REFERENCES

[1] Manik V K, Arora D, "Performance comparison of commercial VMM: ESXI, XEN, HYPER-V and KVM," *International Conference on Computing for Sustainable Global Development. IEEE.*, 2016.

Note: For the Summary paper submission only, references to the authors own work should be cited as if done by others to enable a double-blind review. **Citations must be complete and not redacted, allowing the reviewers to confirm that prior art has been properly identified and acknowledged.**