North South University

Department of Electrical & Computer Engineering

Assignment

Faculty Name: Rashed Mazumder

Course Code: CSE 323

Course Title: Operating System Design

Name: Townim Faisal Chowdhury

ID: 1721327042

Section: 08

Cloud Operating System: Architecture Evolution

Introduction:

The cloud operating system (cloud OS) is used for managing the cloud resources such that they can be used effectively and efficiently. And also it is the duty of cloud OS to provide convenient interface for users and applications. Many traditional applications have been migrated to the cloud systems. For developers and service providers, services and applications are hosted without the concern about infrastructure construction, application deployment, and hardware updating or data center maintenance. Now, the platform components running inside the cloud providers are considered as cloud operating systems (OS).

Cloud computing systems are often built from the existing single machine OS, usually Linux. Now, the other operating systems like Windows have been used in the data center infrastructure. Most of the cloud computing system components are constructed as user applications from the traditional OS point of view. However, the platform for running cloud applications can still be called an OS as the platform serves the same two critical goals as a traditional OS. On the one hand, the cloud OS is used for managing the large scale distributed computing resources, similar to the traditional OS managing hardware in a single machine. On the other hand, the cloud OS provides abstraction for running user applications.

Requirements of Cloud OS:

In a single computer, it is quite clear that there needs an OS abstracting the bare metal hardware for facilitating the applications as well as users to use the computing resources conveniently. However, most of the components in the cloud OS are implemented as user-level programs. The fundamental necessity of building a cloud OS needs further discussion as cloud applications can always be built directly on the current operating systems without concerning the management of hardware in the kernel mode. The true necessity relies on the programming APIs and runtime support in the cloud environment. Local OS's mainly focus on a single machine. They usually do not consider any cloud application logically as a whole unified application spanning over a large

number of machines. Thus traditional OS's just provide the basic facilities for communications. In addition, cloud applications have different forms such as micro services for building web applications, batch processing for big data analysis, query-based data analytics, and real time processing of streaming data. All the applications have their own internal logical organizations of multiple components running on multiple, or even thousands of machines. Exposing very simple communication interfaces is just not enough. Developers need higher levels of abstraction for building their applications without struggling with details related to the complex communication patterns and machine architecture. Exposing APIs that can run on top of multiple machines will be very helpful. This is a very common case for the current cloud OS practitioners such as Google GAE○5, Amazon AWS○6, Microsoft Azure○7 and Alibaba Cloud○8. Cloud application developers can get the programming interfaces without considering the physical resources. For example, developers can store objects in the cloud without knowing the final disks storing the data. Programming is always about abstractions and we need the cloud OS to provide the cloud programming abstraction for building cloud applications. The other reason why a cloud OS is necessary is that running cloud application is different from running applications in a single machine. For each single machine OS, a task scheduler will be used for managing the processes created in the system. However, the runtime characteristics are quite different for running different types of cloud applications such as batch processing, steam processing or graph processing. Cloud operating systems should be built for managing the computing resources related to thousands, or even tens of thousands of machines. And they should provide different management schemas for different applications. The coordination and interference of different applications should be taken into consideration while building such a job scheduler in the cloud environment. Thus, the administrators do not need to manage each individual machine. Based on the above observation, cloud OS is quite necessary and important.

Architecture Evolution of Cloud OS:

Cloud computing can be considered as a successfully commercialized distributed computing paradigm. Traditional distributed operating systems such as Amoeba try to make the distributed environment transparent to the OS layer and the application layer. In the OS layer, distributed shared memory and process migration can achieve the goal. In the application layer, remote

procedure call and inter process message communication can be used to achieve the same goal. The transparency of distributed environment is valid only in the situation of enough low communication latency. Now the computing resources are more flexible, ubiquitous, and heterogeneous. Together with the newly developed cloud applications like MapReduce, all these factors improve the development of cloud system software. MapReduce, proposed by Google first, provides a loosely coupled framework dealing with a large number of heterogeneous and unreliable nodes and is used to build data processing applications.

The current architecture of cloud OS is based on the platform software. The software components are just simply stacked together to provide the functions needed. The architecture is simply the OS combined with network middleware. The network components are typically the distributed database, and distributed storage and middleware for message communication.

They are used to hide the distribution of the underlying computing resources. Sometimes, the infrastructure can do the optimization based on different types of applications. The virtualized underlying platform makes the cloud OS transparent by using VM live migration. However, VMs can introduce great overhead, leading to lower efficiency. The supporting of network data transmission is not enough either.

In recent years, the emerging and the proliferation of containers have improved the development of cloud OS. Containers use the isolation capability from the existing OS. This can reduce the number of abstractions, thus improving the efficiency. By using containers as the basic building blocks for cloud OS, the unified management framework can support extremely large scale computing systems and storage systems. The jobs supported can be a mixture of various types like servicing and batch processing. Complex and comprehensive resource description language can be used for task assignment, scheduling and fault tolerance. In the cloud OS software stack, the application containers are suited for hosting micro services. Now, the management tool like Docker becomes very popular in practice. With its standard build file and the flexible RESTful APIs, the management tool can greatly help improve the automation of software packaging, testing and deployment. However, the current implementation of containers lacks performance isolation and security isolation. Thus, the containers are usually run inside VMs for performance

isolation. This obvious redundancy brings some new opportunities to bring the containers one level lower in the OS by using the microkernel or customized kernel to remove the problem of performance isolation.

The architecture of traditional loosely coupled cloud OS makes it complex to use and introduces a lot of redundant work. It lacks application workload perception and the policy cannot notify the underlying component effectively. Thus, the architecture should consider the vertical integration of OS to reduce the levels of abstraction. More perception points can be put in the OS which can consider the scheduling policies in the distributed and cloud environment. This can help to build a unified and flat management framework for cloud computing platform. Such architecture is beneficiary for function convergence of each component, workload perception based optimization, and orderly evolution. Traditional monolithic single node OS is not easy to decouple the policy from mechanism. Single node OS needs the enhancement of internal structure to improve the capability of application workload perception. The mechanism can be implemented in the small kernel and provide users with more customizable policy interfaces. And more advanced research includes the data distribution, function distribution and space reuse to replace time reuse in OS. This can make the single node OS more scalable and flexible.

Conclusion:

As a conclusion, the cloud OS should vertically integrate single node OS and network middleware for application workload perception. At the bottom layer, the virtualization technology and containers can be used to support multi-core and heterogeneous processors. Also, the decoupling of application development and hardware should be supported as high efficient sandbox. Micro-kernel technologies can be used to improve the isolation of containers and the flexibility of single node OS. At the higher layer, the unified resource management framework can achieve the goals of mixed job deployment, global scheduling optimization and application workload perception. Service mash up can be implemented by using the application package management. For each stage of cloud OS evolution, new APIs are introduced. With the new APIs and their standardization, new applications can push the next stage of OS architecture evolution, expanding cloud technologies adoption.