International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 1, Issue 1, May-June 2012 ISSN 2278-6856

Fault-tolerant Management for Private Cloud System

Win Win Naing

University Of Computer Studies, Yangon, Myanmar
winwinnaing89@gmail.com

Abstract: ***Cloud computing provides access to large pool of data, applications and computational resources. Many researchers have been proposed several open-source Infrastructure-as-a-Service (IaaS) cloud management frameworks (e.g, Eucalyptus, Nimbus, OpenNebula). However, all these systems have limited scalable and fault-tolerance support. Now, fault-tolerance in new computing environments has become a critical requirement. Checkpointing and rollback recovery is a common approach to achieve fault-tolerance. In this paper, we propose the fault-tolerance management framework for private cloud development based on*** **EUCALYPTUS** ***- an open-source framework for cloud computing that implements what is commonly referred to as Infrastructure-as-a-Service (IaaS).***

Keywords: Cloud computing; Scalability, Fault-tolerance.

1. INTRODUCTION

Cloud computing platform offers on-demand services with scalability, elasticity on a simple pay-per-use manner. Moreover, cloud computing systems provide access to large pools of data, application, and computational resources through a variety of interfaces. Users generally locate resources based on a variety of characteristics, including the hardware architecture, memory and storage capacity, network connectivity and, occasionally, geographic location. Usually this resource location process involves a mix of resource availability, application performance profiling, software service requirements, and administrative connections. A user that requires a large number of computational resources might have to contact several different resource providers in order to satisfy her requirements. When the pool of resources is finally delivered, it is often heterogeneous, making the task of performance profiling and efficient use of the resources difficult. While some users have the expertise required to exploit resource heterogeneity, many prefer an environment where resource hardware, software stacks, and programming environments are uniform. Such uniformity makes the task of large-scale application development and deployment more accessible.

Recently, a number of systems have arisen that attempt to convert what is essentially a manual large-scale resource provisioning programming problem into a more abstract notion commonly referred to as elastic, utility, or cloud computing. As the number and scale of cloud-computing systems continues to grow, significant study is required to determine directions we can pursue toward the goal of making future cloud computing platforms successful. Currently, most existing cloud-computing offerings are either proprietary or depend on software that is not amenable to experimentation or instrumentation (Eucalyptus) [2].

Moreover, several open-source IaaS-cloud management frameworks such as OpenNebula, Nimbus, Eucalyptus [2][3], and OpenStack have been developed in order to facilitate the creation of private clouds. But, the failure rates of those systems increase along with their size and complexity. Therefore, fault-tolerance in this environment has become a critical requirement to guarantee the completion of application execution. A commonly used approach to achieve fault-tolerance is checkpointing and rollback recovery, where the intermediate states of running parallel applications can be saved and used later for restart upon failures. Given the ever growing computing power demands, they also need to scale with increasing number of resources and continue their operation despite system component failures. However, all these frameworks have a high degree of centralization and do not tolerate system component failures. Moreover, as no replication support exists, a failure of the frontend node makes the VM management impossible. In addition, Eucalyptus each non fault-tolerant cluster controller suffers from the same drawbacks. And no VM monitoring is performed thus limiting the support for advanced VM placement policies (e.g., consolidation). Eucalyptus does not also include any self-healing features and strictly distinguishes between cloud and cluster controllers. Therefore, we propose a fault-tolerance management framework over Eucalyptus by using wait-free coordination system – Zookeeper. Yahoo! developed a high-performance highly-available coordination service called Zookeeper [5] that allows large scale applications to perform coordination tasks such as leader election, status propagation, and rendezvous. Embedded into ZooKeeper [5] is a totally ordered broadcast protocol: Zab [1].

The rest of this paper is organized as follows. In the next section, we discuss the related papers with our work. In section 3, we present the system overview of the proposed system. And then we describe the proposed scheme in section 4. Finally, we conclude our paper in section 5.

2. RELATED WORK

In this section, we review some paper that is related to the proposed system. Firstly, D. Nurmi, R. Wolski, C. Grzegorczyk [2] [3] proposed Eucalyptus- an open-source software implementation of cloud computing that utilizes compute resources. In the paper, they outlined the design of Eucalyptus, described their own implementation of the modular system components, and provided results from experiments. Moreover, E. Feller, L. Rilling, C. Morin [4] proposed Snooze: a scalable and automatic virtual machine management framework for Private clouds. They mentioned that Eucalyptus do not provide fault-tolerance. Thus they are limited to simple static VM scheduling policies. Y.M. Teo, B.L. Luong, Y. Song and T. Nam [6] proposed a cost performance of fault tolerance in cloud computing. Their paper presented a simple unified analytical model to advanced understanding of the cost-performance tradeoffs for four commonly-used fault tolerant techniques.

In [5], P. Hunt, M. Konar, F. P. Junqueira, B. Reed proposed Zookeeper: wait-free coordination for Internet-scale systems for a service for coordination processes of distributed applications. In this study, we aim to introduce the concept of EUCALYPTUS and a fault-tolerance framework for private cloud.

3. SYSTEM OVERVIEW

In this section, we present the overview of the EUCALYPTUS architecture and their components.

3.1 EUCALYPTUS Design

The architecture of the EUCALYPTUS system is simple, flexible and modula with a hierarchical design reflecting common resource environments found in many academic settings. In essence, the system allows users to start, control, access, and terminate entire virtual machines using an emulation of Amazon EC2's SOAP and "Query" interfaces. That is, users of EUCALYPTUS interact with the system using the exact same tools and interfaces that they use to interact with Amazon EC2. There are four high-level system components, each with its own Web-service interface, that comprise a EUCALYPTUS installation:

- **Node Controller (NC)** controls the execution, inspection, and terminating of VM instances on the host where it runs.
- **Cluster Controller (CC)** gathers information about and schedules VM execution on specific node controllers, as well as manages virtual instance network.
- **Storage Controller (Walrus)** is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.
- **Cloud Controller (CLC)** is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes high-level scheduling decisions, and implements them by making requests to cluster controllers [2].

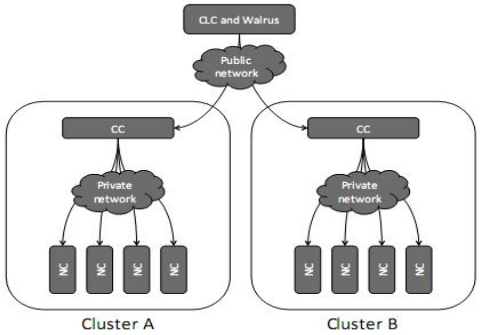

Figure 1 EUCALYPTUS employs a hierarchical design to reflect underlying resource topologies.

The CLC itself is composed of a collection of services that handle user requests and authentication, persistent system and user metadata (e.g., VM images and ssh key pairs), and the management and monitoring of VM instances. The services are configured and managed by an enterprise service bus that publishes services and mediates handling of user requests while decoupling the service implementation from message routing and transport details. Figure 1 shows the relationships and deployment locations of each component within a typical small cluster setting.

4. THE PROPOSED SCHEME

In this section, we present a new framework for fault-tolerance management system for private cloud. This proposed system based on the EUCALYPTUS architecture. In this system, we use the new embedded component Cluster Controller Manager (CCM) to control the system and to gain the fault-tolerance properties.

4.1 Cluster Controller Manager (CCM)

CCM includes the following tasks: stores incoming CC monitoring summary information, dispatches incoming VC submission requests, periodically announces its presence. CCM can use snapshot features to periodically save CC and NC information. CC summary information is periodically sent by each CC to the current CCM in order to support high-level scheduling decisions of the CCM such as delegating VMs to be scheduled on the CCM, and to detect CCM failures. This information is also used in cased of CC failure to rebuild the CCM system view. CCM summary includes the total amounts of used and free capacity available on all the managed NCs. Moreover, to support CC failure detection, each CC is part of a heartbeat multicast group on which if periodically announces its presence. We need to replicate the CLC information at least three. By using replication, we can improve fault-tolerance. We use ZooKeeper – wait-free coordination to elect CCM from among CC.

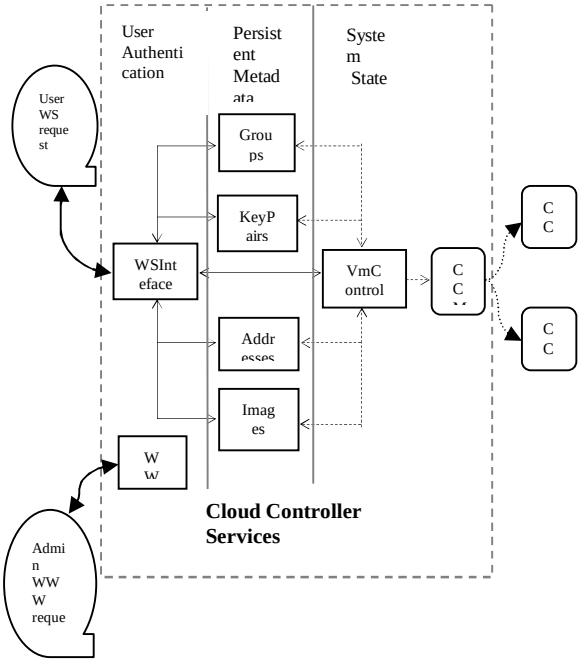

Figure 2 Overview of the cloud controller service that comprise the CCM

In this system, we use Apache ZooKeeper [4][5] leader election algorithm to elect CCM from among CCs. ZooKeeper is highly available and reliable coordination system. ZooKeeper service is installed on the entry points of the cloud controller service. When the CC connects to the ZooKeeper service, creates an ephemeral node in its hierarchical namespace and attaches the CC description to it. This node is assigned a unique sequential identifier by the service in the namespace and is used by the CCs in order to discover the current CCM and elect a new CCM in cases of failure. After the node creation each CC first tries to find another node in the namespace with a lower identifier. If such a node already exists, the CC starts watching it and initiates the CCM heartbeat multicast listener. Finally, upon reception of a CCM heartbeat message, the CC sends a join request along with its description to the CCM. Otherwise if no node with a lower identifier could be detected, the current CC becomes the new CCM and starts announcing its presence by sending multicast messages on the CCM heartbeat multicast port. Figure 2 shows the overview of the cloud controller service that comprises the CCM.

4.2 System Model

In this section, we present the analytical model of a failure recovery system that gains fault-tolerance properties. In this system, the system occur the failure with the Poisson arrival rate λ. We assume number of failure is f ∈ {1..n}. We labeled figure 2 as (f,rc), where f indicates number of failure and rc indicates number of recovery. To construct this model, we use markov chain.

In this model, state (0,0) indicates there is no failure in the system. After that, state (0,0) transits to state (1,0) with rate λ. State (1,0) indicates one failure has occurred and there is no recovery process. Similarly, state (1,0) transits to state (1,c) that indicates the system is checkpointing to recover the system. After checkpointing, the state (1,c) transit to state (1,r). State (1,r) indicates the recovery process. State (1,1) shows the condition that it is finished by recovering the failure.

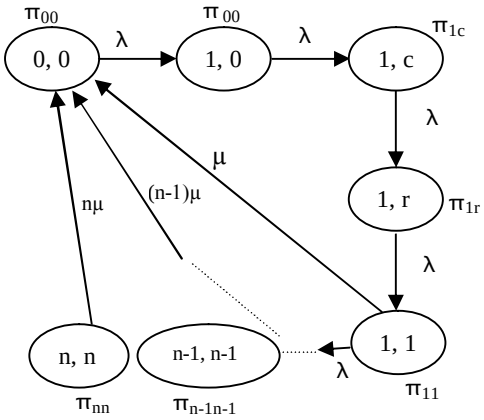

Figure 3: State Diagram

(f, rc) = f is number of failure and rc is number of recovery.

$$\pi_{00} = \frac{\mu\pi_{11} + \ldots + (n-1)\pi_{n-1n-1} + n\mu\pi_{nn}}{\lambda} \quad (1)$$

$$\pi_{00} = \pi_{10} \quad (2)$$

$$\pi_{10} = \pi_{1c} \quad (3)$$

$$\pi_{1c} = \pi_{1r} \quad (4)$$

$$\pi_{1c} = \pi_{11} \quad (5)$$

$$\pi_{11} = \frac{\lambda\pi_{1r}}{(\mu+\lambda)} \quad (6)$$

$$\pi_{nn} = \frac{\lambda\pi_{n-1n-1}}{n\mu} \quad (7)$$

We define the fraction of recovery time is $\rho = \frac{\lambda}{\mu}$, so the probability of n number of recovery is

$$\pi_{nn} = \rho \frac{\pi_{n-1n-1}}{n} \quad (8)$$

Let p (f) be the probability of failure recovery. So,

$$p(f) = \sum_{i=1}^{n} \rho \frac{\pi_{n-in-i}}{n}, \qquad \text{where } i \le n.$$

Table 1 Definition of notations

Symbol	Description
λ	Failure arrival rate of the system
N	Total number of failure
C	Number of checkpointing
R	Number of recovery
μ	Average Recovery Rate
ρ	The fraction of recovery time
π	The probability of failure recovery

In this system, we assume that arrival rate for the recovery rate λ is 5, 10, 15, 20, 25, service rate μ is 32 second per recovery and the probability of starting state (0, 0) is 0.15625. The recovery probability for the number failure is shown in figure 4 and definition of notations is shown in table 1.

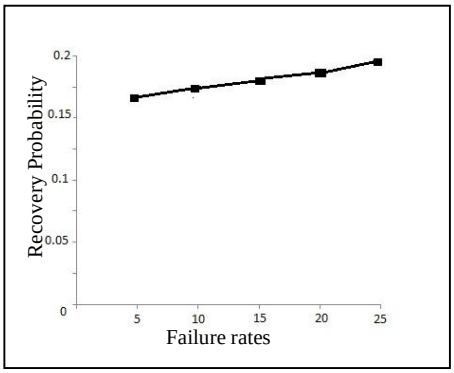

Figure 4: Recovery Probability for failure rates

5. CONCLUSIONS

In this paper, we proposed a fault-tolerance management framework for private clouds development. Previous researchers developed Eucalyptus in order to facilitate the creation of private clouds. But Eucalyptus is non fault-tolerant system and no VM monitoring is performed thus limiting the support for advanced VM placement policies (e.g., consolidation). Eucalyptus does not also include any self-healing features and strictly distinguishes between cloud and cluster controllers. Therefore, we propose the fault-tolerance management framework over Eucalyptus by adding new component Cluster Controller Manager (CCM). We choose CCM as a manager to control the failure occur. We use ZooKeeper leadership algorithm to select CCM as a leader from many CCs. So, we can develop fault-tolerance management framework for private cloud environment.

References

[1] B. Reed, F. Junqueira, "A Simple Totally Ordered Broadcast Protocol".

[2] D. Nurmi, R. Wolski, C. Grzegorczysk, "The Eucalyptus Open-source Cloud-computing System", University of California, Santa Barbara, 2009.

[3] D. Nurmi, R. Wolski, C. Grzegorczysk, "Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems", *UCSB Computer Science Technical Report Number 2008-10*, University of California, Santa Barbara, 2008.

[4] E. Feller, L. Rilling, C. Morin, "Snooze: A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds", December 2011.

[5] P. Hunt, M. Konar, F. P. Junqueira, B. Reed, "ZooKeeper: Wait-free Coordination for Internet-scale Systmes".

[6] Y. M. Teo, B. L. Luong, Y. Song, T. Nam, "Cost-Performance of Fault Tolerance in Cloud Computing", October 28, 2011.

Win Win Naing received the B.C.Sc and M.C.Sc in Computer Science from the University of Computer Studies, Yangon, Myanmar in 2005 and 2008, respectively. She is currently working to receive the Ph.D degree at University of Computer Studies, Yangon, Myanmar (UCSY). Her research interests in the cloud computing.