Efficient Restoration Method of Virtual Resources on OpenStack

Siddagani Vijaya Sirisha, M.Tech
Department of Computer Science and Engineering
SRKR Engineering College
Bhimavaram, India

K. Aruna Kumari, Asst.Prof
Department of Computer Science and Engineering
SRKR Engineering College
Bhimavaram, India

***Abstract*— Efficient Restoration Method of Virtual Resources on OpenStack proposes a method for handling when physical server or virtual machine are down. Many providers have recently started cloud services, and the use of OpenStack, which is open source IaaS software, is increasing. Pacemaker detects a physical server failure only and notifies the failure to a virtual resource arrangement scheduler. Then a virtual resource arrangement scheduler determines multiple physical servers to restore virtual resources and calls OpenStack APIs to rebuild. The virtual resource arrangement scheduler also detects virtual machine failures by using a Libvirt monitoring module. Virtual resource arrangement scheduler take more time for VM clearance. Failures of physical server and virtual machine can be handled by JRobin API. Using Best Partition Searching algorithm we can balance the load and achieve high availability of services.**

Keywords—OpenStack; Pacemaker; Libvirt; JRobin API; Best Partition Searching algorithm

I. INTRODUCTION

Cloud is a technology that liberates the end User from either software dependencies or hardware dependencies or both. According to the definition of NIST [1], cloud service models can be of three types, Software as a Service (SaaS): In this client can access their application running on the cloud but not have any control over cloud infrastructure eg:- web–mail. Platform as a Service (PaaS): In this, the client can deploy onto the cloud infrastructure and can create their own application based on services supported by the provider. Here client will have only control over deployed application and their settings eg:- Google App Engine. Infrastructure as a Service (IaaS): In this client can have control over processing, storage, network and other fundamental cloud resources where the client is able to deploy and run arbitrary software. Here client will not have control over cloud environment but have control over that deployed application and possibly limited control of select network component eg:- Amazon Computing cloud(EC2) [2] and Rackspace Cloud Servers[3] are production IaaS services.

Rackspace uses open source software OpenStack [4] as an IaaS infrastructure. OpenStack, CloudStack [5], and Eucalyptus [6] are major open source IaaS. Eucalyptus provides its service for the private company that wants their own cloud for their own use with help of Amazon Web Services (AWS). CloudStack is suitable for datacenter virtualization. Coming to OpenStack is the most widely deployed open source software for building the cloud. Enterprise uses OpenStack to support rapid deployment of new products.

OpenStack provides control API of the virtual resource to the client in a particular situation; it does not support the restoration of virtual resources when a physical server or virtual machine (VM's) is down. The high-availability (HA) mechanisms of some virtual network resources have been discussed in the OpenStack community [7], but the discussions are scattered for each virtual resource type and there is no uniform method to restore plural types virtual resources.

In fast and reliable restoration method of virtual resources on OpenStack [8] proposes a uniform method for both physical server, virtual machine failures. In that method physical server failures are identified by Pacemaker [9],virtual machine failures are identified by Libvirt [10] and sends the notification to virtual resource arrangement scheduler. Then a virtual resource arrangement scheduler determines multiple physical servers and VM's to restore virtual resources and calls OpenStack APIs to rebuild. Above method takes more time for VM clearance.

This work objective is to restore virtual resource efficiently with a JRobin API's [11] can be used to identify both physical server or virtual machine failover's and send notification's to OpenStack APIs to deploy or restore backup VM.

II. LITERATURE SURVEY

A. The NIST Definition of cloud computing

[1] In this paper, we known about Cloud computing, it is a model for enabling ubiquitous, convenient, on-demand network provided through the internet or local network computing resources are networks, servers, storage, applications, and services that can be provisioned and released speedily with less management effort or service provider interaction. This cloud model consists of some essential characteristics**,** three service models Software as a Service(SaaS), Platform as a Service(PaaS), Infrastructure as a Service(IaaS) and Deployment is of four types private cloud, community cloud, public cloud, and Hybrid cloud.

B. The Review of Virtualization

Virtualization [13] means to create a virtual version of a machine, for an instance a network, server, storage device or even an operating system.

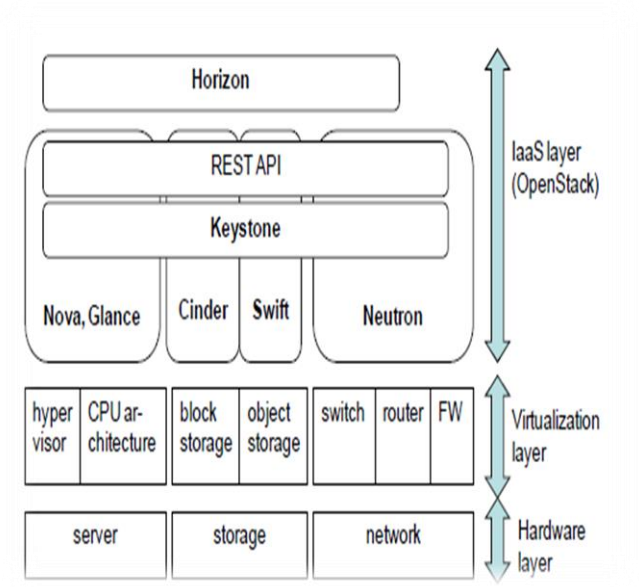

Figure 1: Architecture of OpenStack [4]

It is a technique that divides a physical computer into several partly or completely isolated machines commonly known as guest machines or virtual machines (VM). An assortment of virtual machines can run on a host computer, apiece possessing its own applications and OS. This presents an illusion to the processes on these virtual machines as if they are running on a corporal computer, they are sharing the physical hardware of the host machine in reality. The software that allows multiple operating systems to use the hardware of the physical machine is called a control program or a hypervisor. Hypervisors sit between the operating system of the host machine and the virtual environment.

C. AMAZON EC2 - Virtual Server Hosting

In this paper we know the about Amazon Elastic Compute Cloud (Amazon EC2) [2] is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

D. A Comparative Study Of Open Source Iaas Cloud Computing Platforms

In this paper, the targeted on open source cloud platforms [12] provide flexibility, on demand services and allow great amount of customization. This paper compares the six most popular and commonly used open source IaaS platform-Cloudstack, Eucalyptus, Nimbus, OpenStack, OpenQRM and OpenNebula. It is found that Eucalyptus, OpenNebula and OpenQRM are suitable for private company that want their own cloud. OpenStack is suitable for rapid deployment of new products and Nimbus is well suitable for scientific community. The analysis and summarization done in this paper would help the users to understand the characteristics and would allow users to choose better services according to their requirements and also make more unified decision on the open source cloud platform according to their compatibility, interfaces and deployment requirement. By understanding some of the main differences between them, one can decide where and when each solution may be appropriate for its use.

E. Development of resourses server on OpenStack

In this paper[16], we show the development of resource management server to enable production Cloud services easily based on OpenStack. In recent days, Cloud computing technologies have progressed and many providers have started Cloud services. Some providers use proprietary systems but others use open source IaaS software such as OpenStack and CloudStack. Because the community of OpenStack development is very active, we expect OpenStack will become a de facto standard open source IaaS software. Because OpenStack target is providing primitive APIs for IaaS control, there are some problems to use OpenStack as it is for production services. For example, there are some problems that logical/virtual resources CRUD transactions are insufficient, nova-scheduler which determines hypervisors for virtual machines deployment does not consider operators business requirements and logical checks of unsuitable API calls are insufficient. Therefore, we propose a resource management server which manages physical resources and logical/virtual resources to enable production IaaS services easily based on OpenStack. The resource management server mediates users and OpenStack, provides added actions such as logical checks of API calls, multiple API combination uses, scheduling logic of hypervisors for virtual machines. We implemented the proposed resource management server and showed that operators can operate reliable IaaS services without conscious of OpenStack problems. Furthermore, we measured the performance of multiple API combination uses and showed our method could reduce users waiting time of image deployment or image extraction from volume.

F. JRobin RRD tool

JRobin is a pure java implementation of RRDTool's functionality. It follows the same logic and used the same data sources, archive types and definitions as RRD Tool does. JRobin supports all standard operations on Round Robin Database(RRD) files: create, update, fetch, last, dump, export and graph. JRobin API is made for those who are familiar with RRD Tool's concepts and logic, but prefer to work with pure java. If you provide the same data to RRDTool and JRobin, you will get exactly the same result and graphs. JRobin is made from the scratch and it uses very limited portions of RRD Tools original source code.

Existing System

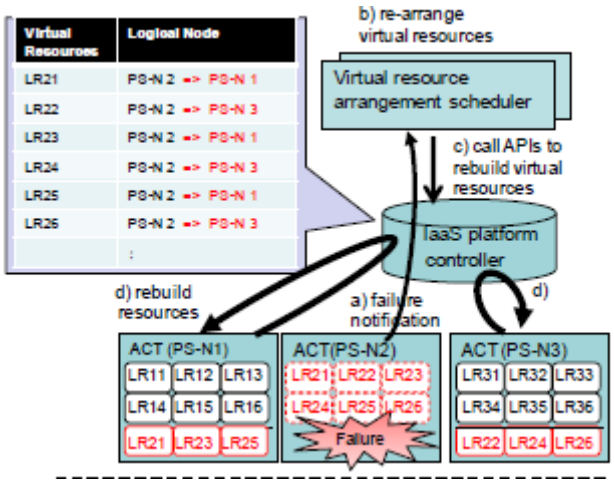

Figure 2 Overview of current methods

III. EXISTING SYSTEM

OpenStack [4] architecture consist a dashboard horizon, it acts as an important component in providing an interface to administrators and users for provisioning of resources. Nova is
designed for provisioning of virtual machines and its components. Cinder manages block storage and can attach a logical volume to VM like a local disk. Glance manages an image. Swift provides objects storage to be used for storing necessary images to run virtual machines. Neutron takes care of network and it provides necessary services which are used for communication within the virtual machine. Keystone provides single sign-on authentication. OpenStack uses Representational state Transfer (REST) API's has Figure 1 show Architecture of OpenStack.

Mainly we use OpenStack API to have, service of our application which is deployed in that environment to make available at any time without fail. If any failure occurs at any constraint provider should make it available to the client as fast as possible without any loss of data.

A. Problems in a uniform method

A uniform method works on plural types in which if a physical server fails Pacemaker [9] will notify to Virtual resource arrangement scheduler, in same way if virtual machine fails Libvirt [10] also notifies to the scheduler.

These both notifications are handled by Virtual resource arrangement scheduler to re-arrange virtual resources. To do that it calls OpenStack API to rebuild virtual resource. This will be done by Nova. The resources that are available will be provided to client. Figure 2 shows how a uniform method works. The notifications that are raised to Virtual resource arrangement scheduler are placed in a queue. Table 1 represents how these notifications are handled. If any failure notification occurs then it start to rebuild a new virtual resource up to that services get down.

B. Problem in loadbalancing

The usage of application that is deployed on OpenStack API is increased more than the limit provided in the existing system they convert these load to newly generated virtual resources.

Algorithm 1 Best Partition Searching

```
begin
  while job do
    searchBestPartition (job);
    if partitionState == idle || partitionState == normal then
      Send Job to Partition;
    else
      search for another Partition;
    end if
  end while
end
```

Figure 3 Best Partition Searching algorithm

C. Using two API for detecting failuers

The physical server failures are detected using Pacemaker API , for VM's failure are detected using Libvirt. Instead using two API's we can achieve using single open RRD Tool.

IV. PROPOSED SYSTEM

Physical server or virtual machine is failed it is detected by JRobin. To deploy the application on OpenStack API environment or restore backup VM to support service continuation is done using JRobin. The API provides a unified programming interface to, abstract from the configuration differences among provider physical server or Virtual Machine implementations. OpenStack and JRobin find's the optimal combination of cloud services with the best network available. Backup VM's for a certain application provided to available with regards of Quality of Services(QoS). It can deploy or redeploy an IAAS application once, then run it anywhere, including support for runtime migrations in a case of a failure.VM services restoration performance time is reduced significantly compared to prior approaches. In this combination, we can overcome limitations of pacemaker and libvirt during VM restorations with an efficient JRobin driven virtual resource arrangement/management scheduler especially with respect to VM deployment duration. In a prior approach for VM restoration, four cases can be considered where 1, 2, 4, and 15 servers were used for restoration. But for demonstration feasibility we use 2- 4 servers.

Load balancing raises to two ways, one is static way, others is a dynamic way we should handle any type of load. In static ways, we set system information irrespective to network IP assigned to the system. It is less complex to handle load obtained by a static scheme. Coming to dynamic schemes network will change frequently for the system, then we need to configure network in the application. A dynamic load balancing should be done for its flexibility. JRobin has a main controller and load balancer to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. This system status then provides a basis for choosing the right load balancing strategy.

The load balancing model proposed by us is aimed at the cloud platform (OpenStack) which has a node

cluster(numerous nodes) with distributed computing resources in various geographic locations. Thus, this model divides the cloud into several virtual resources. The OpenStack has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each virtual resource is a choice of the best load balancing strategy.

TABLE I. RESTORATION POLICIES OF FAILURE CONFLICTIONS

1st notification	2nd notification	timing	restoration policy
physical server	physical server	During VM clear for failed physical server	Keep 1st restoration process and ignore 2nd notification because of same notification.
		During VM create on other physical servers (Double failures of	Stop 1st restoration process and start 2nd notification restoration for new failure.
	VM failure	During VM clear for failed physical server	Because VM clear is done for stopped physical server, it must not happen VM failure notification. Keep 1st restoration process in case of much delayed VM failure
		During VM create on other physical servers	Stop 1st restoration process of the VM and start 2nd notification restoration for new failure.
VM failure	physical server	During VM instance delete on the physical	Stop 1st restoration process because it may fail and start 2nd notification restoration.
		During VM instance create on a physical server	Stop 1st restoration process of the VM and start 2nd notification restoration for new failure.
	VM failure	During VM instance delete on the physical	Keep 1st restoration process and ignore 2nd notification because of same notification.
		During VM instance create on a physical server	Stop 1st restoration process of the VM and start 2nd notification restoration for new failure.

This model uses the following Best Partition Searching algorithm Figure 3, along with previously known round robin algorithms for load and idle estimations of load balancer themselves to deliver the best performance by handling cloud server loads better.

In this method, if any physical server or virtual machine is failed it is detected by JRobin API and call's OpenStack to rebuild virtual resource, at that point of time to provide a service for high availability. We proposed to create more than one virtual resources which should be readily available by that if any failure occurs alternate resources provide the service. By adopting this we can achieve high availability of services. Using best partition searching algorithm we manage the load for all available virtual resources in this it select any of the available resources by that we can eliminate failures for maximum conditions.

V. CONCLUSION

In this paper we proposed Efficient Restoration Method of Virtual Resources on OpenStack which is a method for handling when physical server or virtual machines are down. Failures of physical server and virtual machine can be handled by JRobin API. Using Best Partition Searching algorithm we can balance the load and achieve high availability of services. It is achieved by using more than one virtual resource which should be readily available by that if any failure occurs alternate resources provide the service. By adopting this we can achieve high availability of services. Using best partition searching algorithm we manage the load for all available virtual resources in this it selects any of the available resources by that we can eliminate failures for maximum conditions.

REFERENCES

[1] P. Mell, and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, SP 800-145, Sep. 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[2] Amazon Elastic Compute Cloud web site, http://aws.amazon.com/ec2/
[3] Rackspace public cloud powered by OpenStack web site, http://www.rackspace.com/cloud/
[4] OpenStack web site, http://www.openstack.org/
[5] CloudStack web site, http://CloudStack.apache.org/
[6] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," In Proceedings of Cloud Computing and Its Applications, Oct. 2008.
[7] OpenStack virtual network HA bluprints web site, https://blueprints.launchpad.net/neutron/+spec/l3-high-availability
[8] Yoji Yamato, Yukihisa Nishizawa, Shinji Nagao and Kenichi Sato *Member of IEEE* "Fast and reliable restoration method of virtual resources on OpenStack" DOI 10.1109/TCC.2015.2481392,IEEE Transactions on Cloud Computing 2015 Volume PP Issue 99 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7277013
[9] Pacemaker web site, http://www.linux-ha.org/wiki/Pacemaker/
[10] Libvirt web site, http://libvirt.org/
[11] JRobin web site https://oldwww.jrobin.org/
[12] Comparative Study of Eucalyptus, Open Stack and Nimbus (IJSCE) ISSN: 2231-2307, Volume-4 Issue-6, January 2015 http://www.ijsce.org/attachments/File/v4i6/F2460014615.pdf
[13] Y. Yamato, Y. Nishizawa, M. Muroi and K. Tanaka, "Development of Resource Management Server for Carrier IaaS Services Based on OpenStack," Journal of Information Processing, Vol.23, No.1, pp.58-66, Jan. 2015.
[14] Sunanda Assistant professor, Department of Computer Science & Engineering, Ludhiana College of Engineering & Technology, Ludhiana, Punjab, India "The Review of Virtualization in an Isolated Computer Environment" http://www.ijarcce.com/upload/2015/may-15/IJARCCE%2010.pdf
[15] Y. Yamato, Y. Nishizawa, M. Muroi and K. Tanaka, "Development of Resource Management Server for Carrier IaaS Services Based on OpenStack," Journal of Information Processing, Vol.23, No.1, pp.58-66, Jan. 2015.