Indian Journal of Science and Technology, *Vol 10(4), DOI: 10.17485/ijst/2017/v10i4/110663, January 2017*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Design of a Fault Tolerant Architecture for Private Cloud Computing Infrastructure

Yugashri Joshi[1*], Deepak Sharma[1], Urvashi Gaur[2], Vaibhav Kumar[2] and Rajesh Kalmady[2]

[1]K. J. Somaiya College of Engineering, Mumbai – 400077, Maharashtra, India; yugashri.j@somaiya.edu, deepaksharma@somaiya.edu

[2]Computer Division, Bhabha Atomic Research Centre (BARC), Mumbai – 400085, Maharashtra, India; urvashi@barc.gov.in, kvaibhav@barc.gov.in, rajesh@barc.gov.in

Abstract

Objectives: Cloud computing has emerged as a cost-effective technology providing services to business-enterprise and IT industry. In this paper, we discuss a novel fault-tolerant scheme from an administration point of view that corresponds to implementation of Redundant Array of Independent Nodes (RAIN) model based architecture for ensuring fault-tolerance in a private cloud infrastructure setup. **Methods:** After exploring various combinations of Redundant Array of Independent Disks (RAID) and Logical Volume Manager (LVM) techniques, we propose a novel idea of having a fault-tolerant architecture for cloud infrastructure. **Findings:** As an alternative to the existing cloud fault-tolerant techniques which use expensive and specialized shared storage system, local server disk pool will be used for storing VM images. This architecture provides a robust infrastructure built over Internet Small Computer System Interface (iSCSI) targets and software RAID. **Application:** This RAIN based model can be deployed over the existing cloud infrastructure. This storage model can be used by cloud toolkits to provide fault tolerant backend for storage.

Keywords: Cloud Computing, Fault-Tolerance, iSCSI Target, RAID, Virtualization

1. Introduction

Due to the soaring cost and efforts required to maintain computing and storage infrastructure, companies are outsourcing the same to cloud service providers. Cloud computing has brought a new trend into computation model which is based on sharing a pool of computing resources. The resources are rapidly provisioned on demand and released with minimal management effort or service provider interaction. The cloud model is basically pay-per-use model. The cloud user has to pay only for what the user uses. It is thus a cost-efficient solution with which resources are remotely provided over the network.

Virtualization is the fundamental technology that influences cloud computing. With virtualization, a consumer can utilize resources such as storage, network, desktops, or other such entities, remotely from his end. Server virtualization is the key technique to allocate isolated virtual spaces i.e. virtual machines to users for running their applications or services.

Cloud computing infrastructure is very complex and vulnerable to various system failures. From an administrator[1] point of view, who is hosting the cloud infrastructure it, is important to provide continuous and uninterrupted services even during failures. Hence, there is a need of implementation of a robust fault tolerant scheme for the cloud infrastructure. A lot of work has been done in the area of fault tolerance with respect to predicting failures and recovering from it. Various Reactive Fault Tolerance[2] techniques like check-pointing, replay-retry and Proactive Fault Tolerance techniques[3] like preemptive migration and software rejuvenation are deployed at

*Author for correspondence

service providers end that help in restoring applications/ software running on the cloud.

In this paper, we propose a fault tolerant architecture that will improve the availability and reliability[4] of the underlying physical servers hosting the cloud[5]. In the event of breakdown, the system will continue to operate and cause a minimal impact on the overall performance, while the maintenance can be done in the background. The rest of the paper is organized as follows. Section 2 discusses the motivation for the research work. Section 3 discusses various schemes that were deployed and tested to investigate usage of techniques such as LVM and RAID. Section 4 covers the proposed design of RAIN model for achieving fault tolerance where the disks from physical servers are imported as iSCSI targets under a software RAID configuration to provide redundancy and recovery from a failure. Section 5 presents our conclusion and future work.

2. Motivation for Design of a New Fault Tolerant Scheme

The private cloud infrastructure that provides Infrastructure as a Service facilitates users to create new virtual machines on demand. These servers are ready to use with installed OS and network connectivity. An infrastructure built over off the shelf commodity hardware servers has no supplementary fault tolerant schemes or specialized storage systems. In such scenario, addressing the issue of fault handling is of major concern.

In our cloud setup, Openstack has been deployed as the cloud middleware for supporting the cloud environment. It consists of various services like Neutron - Enables Network Connectivity as a Service, Keystone- Provides an authentication and authorization service, Horizon - Provides a web-based self-service portal to interact with underlying OpenStack services, Glance- Stores and retrieves virtual machine disk images and Nova- Manages the lifecycle of compute instances/ virtual machines on demand. In case of failure of a compute node hosting the Nova service, the virtual machines running on that physical host are interrupted. VM migration from a failed node to another requires migration of VM configuration, network configuration and VM disk image. Out of the three, transfer of VM disk image data is the most time consuming and contributes as largest fraction of the downtime.

In a cloudsetup where instead of specialized storage system, local disks of compute nodes are used to keep the VM data, it is required to have a technique that would assist the administrator to replay virtual machines with minimum downtime. Our aim is to provide a fault tolerant scheme that minimizes the VM migration time by leveraging recovery and failure handling on an underlying fault tolerant physical infrastructure which will be built over iSCSI targets, LVM and software RAID volume. For investigating various aspects of the design, combinations of the above mentioned techniques were deployed and tested.

3. Explored Techniques

A key technique for fault tolerance is to use redundant infrastructure where extra equipments or processes are added to make it possible for the system as a whole to tolerate the loss or failure. The present test system scenario consists of a private cloud deployed over a small cluster delivering infrastructure as a service. The users are granted virtual machines on request. In case of a failure, it is important to recover the data of virtual machines that are running at a given point of time. To investigate approaches to be used in RAIN architecture, we deployed and tested various combinations of RAID and LVM.

RAID[6] is a technology that virtualizes multiple independent Hard Disk Drives (HDDs) and arranges the data and parity on multiple disks using striping in order to provide redundancy and fault tolerance. Total RAID array capacity, and number of disk failures allowed depends upon the type of RAID array built and the number and size of disk drives. A RAID array thus improves overall performance, and increases the storage capacity in a system as the multiple disks are seen as a single disk[7]. There are two different types of RAID implementations: Hardware RAID and Software RAID[8]. A hardware RAID device connects to the SCSI controller and presents the RAID arrays as a single SCSI drive. Hardware raid has a dedicated hardware to control the array, where as array control processing in the software RAID is done via software. In software RAID, the Linux kernel contains an MD driver that allows the RAID solution to be completely hardware independent. The performance of a software-based array depends on the server CPU performance and load.

LVM[9] is an abstraction layer between the operating system and physical hard drives. With LVM, disks and partitions can be abstracted to contain multiple disks and partitions into one device. Because volume groups and logical volumes aren't physically tied to a hard drive, it makes

it easy to dynamically resize and create new disks and partitions. LVM thus allows resizing of partitions inside a physical volume. It supports effortless management of logical volumes, or file systems on various hard drives.

Following methods were tested on the physical systems to compare performance:

3.1 RAID over LVM

Figure 1 shows the combination of RAID over LVM scheme. Here, a physical disk is first partitioned into a number of smaller disk partitions. Each of the smaller partition is of the LVM type. Using 'mdadm' utility, a software RAID was created over the logical volumes. Virtual Machines were instantiated on the Software RAID volume.

3.2 LVM over RAID

In this approach, the physical disks were partitioned and in contrast to the above approach, first the software RAID was built over these disk partitions using the mdadm utility. Logical Volumes were built over Software RAID volume. Virtual Machines were instantiated on the Logical Volumes.

To compare performance of both the schemes, files of various sizes in range of gigabytes were written and read. It was observed that the block level access via LVM over RAID had a better performance. Hence, this combination was chosen for further deployment of our RAIN model based of fault tolerant architecture shown n Figure 2.

4. Proposed RAIN Model Based Architecture

In cloud computing, virtual machine migration from a failed/faulty node to a working node is one of the major tasks of operation and maintenance administrator. To have minimum downtime, we propose a novel idea of having a RAIN based architecture for ensuring fault tolerance in a cloud infrastructure. Physical server redundancy will be analogous to the ubiquitous RAID implementation of disks, where a hard disk is hot plugged in and out of the system without any data loss or disruption of service. In this proposed architecture, each physical server hosting the compute node service will mimic the role of a hard disk in RAID, which can be plugged in/out without any service disruption and data loss. Instead of using heavy software, the cloud infrastructure itself can be built in the following way to cater the needs. As the data of the virtual machines reside in the underlying host server disk space, we propose to build this infrastructure fault tolerant such that in case of a hardware or software failure, the VM data is readily available. Figure 3 shows the proposed architecture.

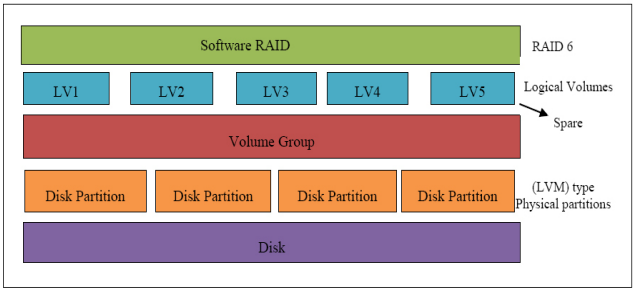

Figure 1. RAID over LVM.

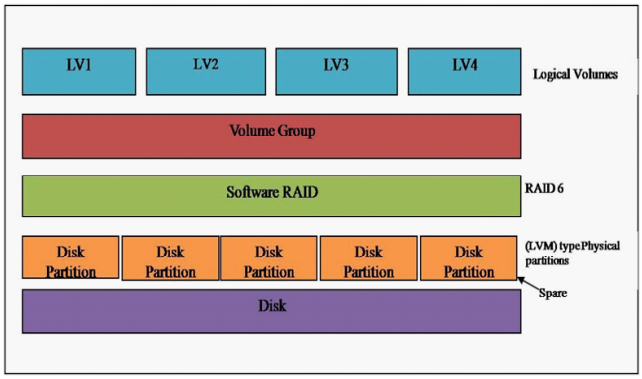

Figure 2. LVM over RAID.

In our design, disks of compute nodes were imported as iSCSI Targets. iSCSI[10] is an Internet Protocol (IP)-based storage networking standard. It works by running SCSI commands over IP networks like Local Area Networks (LANs) or Wide Area Networks (WANs) and can enable location-independent data storage and retrieval. Various iSCSI target stacks are available with Linux viz. LIO, TGT, SCST, STGT. We have used iSCSI (TGT[11]) for creation, discovery of targets as well as for logging into the targets.

To further explain the approach, let us consider three physical servers. Disk partitions were created on each of the servers and iSCSI targets were created on them. The targets on these servers are the physical disks of the servers on which fault tolerance scheme is to be applied. Using one of the servers as iSCSI initiator, we login into the targets (hard disks of other servers) such that they are now visible as disk partitions on the initiator. For example, consider Server 1 as the iSCSI initiator. When we login into the targets on the other two server viz. Server 2 and Server 3, their local targets are now visible on Server 1.

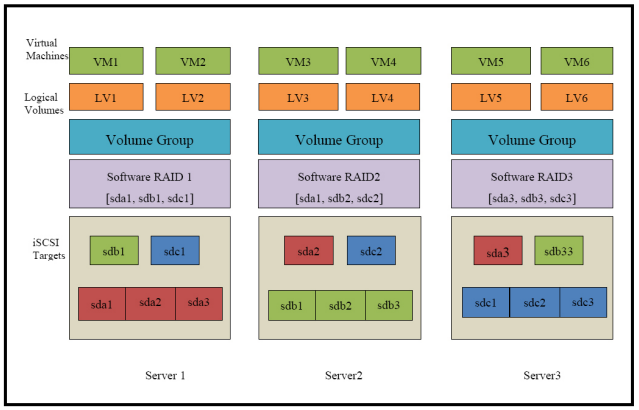

Figure 3. Proposed fault-tolerant scheme: RAIN model.

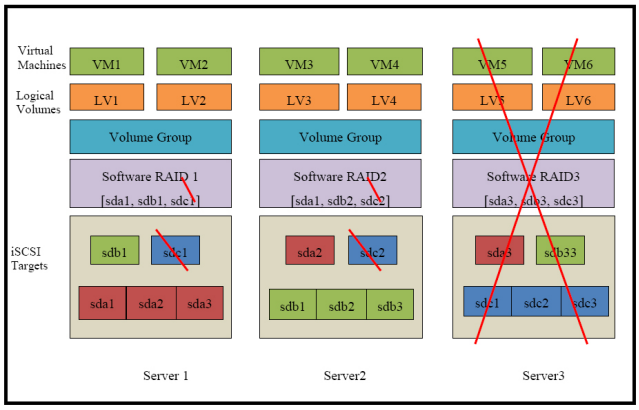

Figure 4. Handling failure in RAIN architecture.

This similar process can be done on all the three servers. On Server 1, we now have three disks(one local 'sda1' and two iSCSI targets 'sdb1,sdc1') over which we create a software RAID level 5 using mdadm, considering the targets as the devices for the RAID array (since three disks are available in our example, RAID 5, which allows failure up to one disk, is possible.) A volume group is then created on the RAID array which has the size of the RAID array created. On this volume group, we create LVs. Using any cloud middleware, Virtual Machines (VMs) can be created on the Logical Volumes (LV) such that one VM is instantiated on one LV. The virtual machines so created can then be allocated to the cloud users. The RAID volume is visible on all the three servers but can be active only one server at a given point of time.

4.1 Failure Handling

In case of failure of a compute node, the local disks of that node which are mounted as iSCSI targets will not be available with other servers, but as we have a RAID level 5 built over the disk pool, failure of one server will cause no impact on the running system. Figure 4 shows the scenario in which server3 has failed, we can see that RAID will work in degraded mode on server1 and server 2 causing no harm to the running VMs. For migrating and restoring the VMs running on server 3, we just need to reassemble the RAID built on server 3 to other healthy node under degraded mode. As the software RAID is stripped over all the three servers even in case of fault in one server, the data of the virtual machine is not lost. It is only required to reassemble the RAID in degraded mode and activate the volume group built over it from another healthy server and we get the fully functional VMs running on LVs. This model can be extended by deploying minimum 4 physical servers with RAID level 6 and up to two physical server failures can be sustained. This scheme is scalable and can be implemented over the whole cloud infrastructure to provide fault tolerance.

5. Conclusion

In the case of system failures, Cloud Service provider has a major task of migrating and restoring virtual machines. We propose a novel Redundant RAIN based model for a cloud setup built over off the shelf commodity hardware which uses local disk pool for storing VM images, as an alternative to expensive, fault tolerant specialized shared storage system. Instead of leveraging recovery and fault tolerance on application layer, a robust infrastructure built over iSCSI targets and software RAID will provide the necessary fault tolerance. This architecture can be used as a substitute volume manager by Cloud toolkits to provide a fault tolerant backend for volume storage.

6. References

1. Jhawar R, Piuri V, Santambrogio M. Fault Tolerance Management in Cloud Computing: A System-Level Perspective, Ieee Systems Journal. 2013 Jun; 7(2):288–97.
2. Rajesh S, Devi RK. Improving Fault Tolerance in Virtual Machine Based Cloud Infrastructure, International Journal of Innovative Research in Science, Engineering and Technology. 2014 Mar; 3(3):2163–68.
3. Tchana A, Broto L, Hagimont D. Fault Tolerant Approaches in Cloud Computing Infrastructures, ICAS, 2012, Proceedings of the Eight International conference on Autonomic and Autonomous Systems; 42–48.

4. Padmakumari P, Umamakeswari A. Methodical Review on Various Fault Tolerant and Monitoring Mechanisms to Improve Reliability on Cloud Environment, Indian Journal of Science and Technology. 2015 Dec; 8(35):1–6.
5. Giriesh S, Sindhuja V, Padmakumari P, Umamakeswari A. Dynamic Data Fault Tolerance Mechanism to Enhance Reliability and Availability in Cloud, Indian Journal of Science and Technology. 2015 May; 8(S9):300–05.
6. RAID (Redundant Array of Independent Disks). Date accessed: 2016. http://searchstorage.techtarget.com/definition/RAID.
7. RAID-redundant Array of Independent Disks. Date accessed: 2016. http://www.webopedia.com/TERM/R/RAID.html.
8. Red Hat Enterprise Linux 3: System Administration Guide, Chapter 3. Redundant Array of Independent Disks (RAID). Hardware RAID Versus Software RAID. Date accessed: 2003. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/System_Administration_Guide/s1-raid-approaches.html.
9. The Linux Logical Volume Manager. Date accessed: 30/07/2016.https://en.wikipedia.org/wiki/Logical_Volume_Manager (Linux).
10. iSCSI. Date accessed: 07/08/2015. http://linux-iscsi.org/wiki/ISCSI.
11. TGT iSCSI Target. Date accessed: 08/10/2015. https://wiki.archlinux.org/index.php/TGT_iSCSI_Target.