Comparing Replication Strategies for Financial Data on Openstack based Private Cloud

Deepak Bajpai, Ruppa K. Thulasiram
Computer Science department
University of Manitoba
Winnipeg, Canada
Email: {bajpaid, tulsi}@cs.umanitoba.ca

***Abstract*—Private Cloud has substituted the traditional infrastructure due to its flexible model and privacy in the form of administration control and supervision. In this study, we have built a private Cloud using openstack Cloud based open-source software solution. We deployed financial application on VMs and generated a disaster recovery solution using openstack component Cinder and Swift. We presented an experimental analysis of two strategies, block storage and object storage, to derive the best solution for an organization using private Cloud. From the set of experiments considered Cinder proved preferable over Swift.**

Keywords- Cloud Computing, Private Cloud, Replication, Disaster Recovery, Block Storage, Object Storage

I. INTRODUCTION

With the exponential growth of Information Technology (IT) Infrastructure and increasing cost of IT from small scale to high end enterprise sectors, a workaround to effectively reduce the cost associated with infrastructure has become essential. Also, as the uncontrolled data growth raised concerns for the enterprise environment, storage moved outside the servers to individual identity that provided a significant solution [1]. Integration of Virtual Machines (VM) in the virtualization layer to the operating system environment added a robust and effective measure to share the CPU load and processing time [2]. With more advancement in technology, more cost effective solutions were required. It was this time when Cloud computing came into emergence.

Cloud computing is a major shift on how the information is stored and shared on the distributed platform to provide services to customer as per customer demand. Cloud has been defined as "pay-as-you-go" architecture where customer pays for the resources when used. Introduction of Cloud service by Amazon, Microsoft and other Cloud service providers induced leading technology hubs like Oracle, HP, IBM, Adobe, etc., to expand their service areas for Cloud technology. When it comes to financial, health and defense services, public Clouds have several vulnerabilities, including security and privacy issues, and this is where private Cloud comes into the picture. Private Cloud has shown reliable solutions for these components of business and hence many organizations have jumped into private Cloud [3]. A private Cloud is more independent as the organisation build their own infrastructure, use their own data storage blocks and servers, and at times private Cloud owners may also outsource their requirement to the third party.

In public Clouds, the service provider has control of the IT infrastructure and, eventually, they control customers sensitive data which reside in their datacenter. Even though there could be regulatory procedures (such as Service Level Agreement) in place that ensures fair management and supervision of the customers privacy, this condition can still be perceived as a threat or as an unacceptable risk that some organizations are not willing to take. In particular, institutions such as government and military agencies will not consider public Clouds as an option for processing or storing their sensitive data [4]. When we put financial application and operations in public Cloud then there is severe threat of vulnerability as all the data is stored with third party Cloud vendor. To address this issue we built private Cloud and deployed financial application to provide security and privacy.

The main problem in the market of Cloud services is implementation and performance evaluation for the critical data. IT infrastructure organization has a central data repository that contains all the important data required for the proper functioning of the organization. This data can be classified as critical on the basis of usage by the users. In a Cloud environment, data remains at different geo-locations; VMs at different geo-locations hold that data which can be migrated from one host to another host as per the requirement of load balancing [5].

Some of the replication techniques which are used by market leaders are object storage or block storage replication. A cost effective dynamic replication management (CDRM) scheme has been introduced for storage clusters in Cloud [6] which has used Hadoop data filesystem. However, in CDRM, filesystem should be in a mounted state for initiating the replication. Various object storage replication techniques have been in market [7] and they are efficient, but for block storage replication in Cloud storage cluster, there is still demand for cutting edge strategy.

In this paper, we have proposed a new replication strategy for the duplication of data in openstack based private Cloud. In this strategy, we created the VM using the block storage component of openstack called Cinder. After the testing of VM on a financial application, we created snapshots and used them for later recovery process. This strategy is new, and we

 ISBN: 978-1-61208-460-2

show that this is 53% better in time than object storage based replication technique. Rest of the paper describes the attributes of openstack, the deployment of private Cloud and the empirical analysis of the replication strategy all in the related work section. Implementation and disaster recovery mechanism are described in Section III. We did experimental comparison with object based replication using Swift component of openstack as described in Section IV. After comparing these action plans, we have analyzed the results on the basis of best strategy available in Section V. We present our conclusions in Section VI. One of the financial applications we have considered is option pricing. Note that due to space limitation we are not describing the financial option pricing application and the data related to this application, brief description on which are available in [2].

II. Related Work

There are multiple software sources available to build a Cloud. CloudStack is an open source software that allows for a Private Cloud and Hybrid Cloud deployment [8]. The main functionality of CloudStack is to operate a large scale deployment of a virtual infrastructure by managing a large number of virtual machines. Eucalyptus is an open source software used to build Private Clouds and Hybrid Clouds that are compatible with Amazon Web Services [9]. Eucalyptus manages resources that allow for some dynamic allocation of resources. OpenNebula is an open source software that allows for the deployment of Public Clouds, Private Clouds, and Hybrid Clouds [10]. The main functionality of OpenNebula is to manage data that is distributed among datacenters and to ensure that these datacenters are able to exchange information with each other regardless of their infrastructure. OpenStack is an open source software that is primarily used to manage a virtual infrastructure using the Dashboard or the OpenStack API. What makes OpenStack advantageous compared to the other options is that OpenStack supports small scale deployment. OpenStack supports deployment onto a single, local machine for rapid application development and testing with minimal setup required. For this reason, OpenStack was selected in this study as the Cloud computing software to develop the disaster recovery application.

A. Openstack

Originally OpenStack was a project which was developed by Rackspace and NASA [11]. In 2010, OpenStack was released under the name Austin. Austin initially had very limited features. At the time Austin had support for object storage only. Companies started to contribute to OpenStack because they began to see its potential in the virtualization market. As contributors added more features to OpenStack, eventually in 2012 support for Cinder was integrated [3]. OpenStack provides a means for small companies to provide their customers with services without the need of a significant investment initially. It became easier for new service providers to start out who did not have a lot of investors. At the same time, OpenStack provided a means for big organizations to implement private clouds within their corporation without the need of big investments into physical hardware. OpenStack is currently an open source project and has hundreds of contributors. What makes OpenStack so advantageous is that if there is a need for a feature there are contributors constantly available to implement them and fix them if any bugs are discovered. Another reason OpenStack is advantageous is that it is an open source project and it does not require any subscription or an annual fee to use.

All of the modules contribute various components to OpenStack. Of the nine modules listed in Figure.1, only three of those modules provide storage mechanisms for OpenStack. Each of these three modules provide a different storage mechanism. The three modules are, Swift, Cinder, and Glance. Swift provides an object storage capability, Cinder provides a block storage capability, and Glance provides a repository to store the virtual machines a user creates in OpenStack or downloads them from the internet.

1) Cinder: Block storage was a fundamental milestone for Cloud computing because it provides the capability to store virtual machines along with the data those virtual machines use. Before block storage was integrated into OpenStack virtual machines used something called ephemeral storage [11]. Virtual machines using ephemeral storage have a significant, fundamental flaw. The flaw is that when the virtual machine powers down, all of the data and contents of the virtual machine are lost. This is a problem because a virtual machine may contain important data required to provide services to customers. Say the service being provided is banking transactions. If the virtual machine powers down, all of the customers banking information is lost. The organization wont know what the last balance on the customers account was if that information wasnt backed up. Simply powering on the virtual machine wouldnt solve the problem because even though we have the environment to provide the service, we need the data of past transactions. Finding a solution to ensure a virtual machine never powers down is not a realistic answer to the problem.

There was a need to provide a mechanism that allows data to persist in the event a virtual machine powers down. Block storage provides one possible solution to this problem. The data persists within the volume even though the virtual machine may power down and the data is available the next time the virtual machine powers up. Cinder was developed and integrated into OpenStack to provide access to and management of a block storage created by a user.

2) Swift: In OpenStack, the module named Swift provides the object storage component in the application. Swift is used to implement something called an OpenStack cluster. The OpenStack cluster is used to provide a distributed object store. The object store uses HTTP protocols PUT to update an object or GET to retrieve the most recent version of the object. Swift implements something called an OpenStack ring. This ring consists of a proxy server and a storage node. These two

components are used to store and retrieve objects upon a users request. The main functionality of the proxy server is to track the location of the most recent version of the object and to determine which storage server it should send the most recent version of the object. These OpenStack rings can be divided in various ways. The reason for the division is if in the event a request for an object fails, then there is another entity available who can fulfill this request. Four common divisions of OpenStack rings are by disk drive, server, zone, or region.

Within each of these divisions the data is replicated in the event one of the providers has a failure so another entity can fulfill the request. Through various testing it was determined maintaining three replicas was sufficient for reliability so by default OpenStack maintains three replicas of the data [3]. Swift implements eventual consistency. This means that when an object is updated not all of the storage servers are updated immediately to contain the most recent version of the object. Only one of the storage servers receives the most recent version of the object from the proxy server and then the storage server propagates that object to the rest of the storage servers as a background task. A risk with eventual consistency is that a storage server may receive an updated object but then may fail before it has a chance to propagate that updated object to another storage server. In a future updated versions of OpenStack, code named Grizzly, allowed users to maintain as many replicas as they wanted within the OpenStack ring. Another feature implemented was time based sorting where the proxy server would request the most recently updated object from the fastest responding storage server. The main reason time based sorting was implemented was because storage servers were beginning to be distributed over larger areas, for example, the other side of the country. A major advantage Swift provided was that it allows the capability to store objects on different platforms for example, Cleversafe, Scality, and Amazon S3[5].

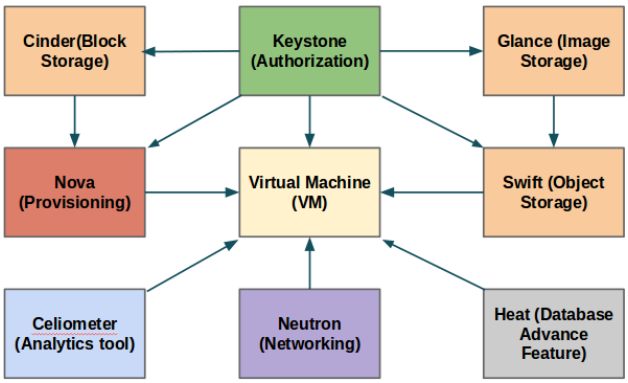

Figure. 1. Internal component connection in openstack.

III. Disaster Recovery and Implementation

”Disaster recovery (DR) is defined as the use of alternative network circuits to re-establish communications channels in the event that the primary channels are disconnected or malfunctioning, and The methods and procedures for returning a data center to full operation after a catastrophic interruption (e.g., including recovery of lost data). Disaster recovery involves in restoring the technology aspect of a service. In 1978, Sun Information Systems was the first American company to provide a hot site to organizations [12]. A hot site is a facility that an organization can relocate to in the event of a system failure due to a natural disaster or a human induced disaster to resume the operation of a service.

As businesses became more reliant on their IT infrastructure during the late 20th century, there was pressure for organizations to have a disaster recovery plan in place in the event a service becomes unavailable. As a result different mechanisms were being provided as solutions. Data centers were being built because on site recovery was no longer necessary due to the development of the internet. It has been statistically proven that spending one dollar on disaster recovery planning will save you four dollars in the long run when trying to recovery from a disaster[12]. An important step in disaster recovery planning is to identify which services are considered vital for an organization, how long can they afford to keep the service unavailable and identify which systems are associated with providing the service [12].

There are multiple strategies in designing an efficient disaster recovery plan, here are a few strategies. Backups of systems can be made and transported to an offsite location. A second alternative is to only forward data to an offsite location instead of copying the entire system. A third alternative is to use a Private Cloud to replicate virtual machines, disks, templates and store them in the Private Cloud. A fourth alternative is to use a Hybrid Cloud to back up the data stored onsite and offsite and in the event of a disaster launch the system from within the Cloud. A fifth alternative is to backup both the system and the data at an offsite location. For the purposes of this current study, a backup will be made of both the state of the system and the data it contains.

Implementation of private Cloud is a complex and cumbersome process. Its architecture is same as distributed system but the association of various components to make it service oriented architecture brings the complexity in design. After the implementation it is important to test the performance on the basis of various performance parameters such as response time, replication status, data integrity, accessibility on the data critical and time constraint application. The most suitable application are financial application where data is critical and time is also a big constraint. We built Openstack private cloud on three node architecture as shown in Figure 2. Below are the description of each nodes.

1) Controller node- Controller node is responsible for running the basic openstack services required for private Cloud environment to function. These nodes: (a) Access to API which is accessible by user for various functionality. It is also

 ISBN: 978-1-61208-460-2

entry point for the accessibility; (b) Run numerous services in high availability, utilizing components such as Pacemaker and HAProxy to provide a load-balancing and virtual server allocation functions so the controller node is being used; (c) Provide highly available ”infrastructure” services, such as MySQL and RabbitMQ that combine all the services; (d) Provide what is known as ”persistent storage” through services runs on the host as well. This persistent storage is backed onto the storage nodes for reliability.

2) Compute Node- Compute nodes host the virtual machine instances in OpenStack environment. They: (a) Run the minimum of services required to run these instances; (b) Use local defined storage on the nodes for the VM so that disaster recovery in case of node failure is possible.

3) Network Node- Network nodes provide communication channel and the virtual networking needed for users to create public or private networks. It also provide uplink to their virtual machines into external networks. (a) Form the only ingress and egress point for access and security feature of the Openstack running instance; (b) Run the environment's networking services other than networking API service. Virtual

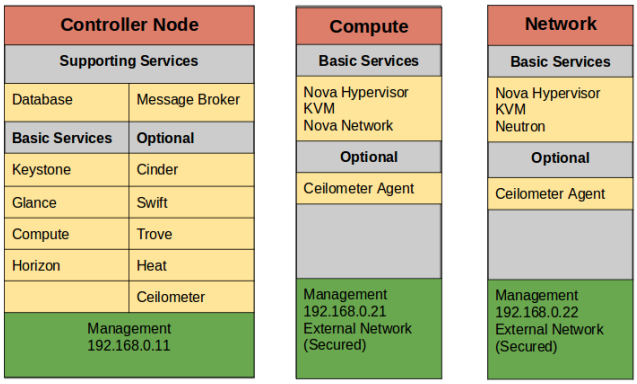

Figure. 2. Architecture of Openstack based Cloud.

machine provisioning is an important step for the private Cloud. Following are the basic components required for the VM provisioning:1) Block Storage (Cinder)- Provides persistence block storage for running VMs and create VMs as well using images, 2) Object Storage (Swift)-Stores and retrieve arbitrary unstructured data objects via a RESTful, Http based API, 3) Network (Neutron)-Provide communication channel as a service for Openstack services, such as Openstack compute, 4) Authorization(Keystone)-Provides an authorization and authentication service for all Openstack services, 5) VM Image (Glance)-Stores and retrieve VM disk images, 6) Provisioning (Nova-Compute, KVM hypervisor integrated), 7) Controller provisioning (Nova-Controller). Openstack provide all these as separate components as shown in Figure 1. These components help to establish services within the framework. The VMs on the compute nodes are working on arbitrary computation algorithms which is be memory intensive. After that, we used snapshot based technology to capture the state of VM image and these snapshots are scheduled to be triggered in a timely manner [13]. Along with the snapshots, block replication is initiated on the VMs, which captures the raw data on the data blocks and store them on other passive systems. These data blocks can be retrieved in case of disaster and which will provide same data as an active node. With snapshot and synchronous replication, data will be more secure.

IV. Experiments

To build a private Cloud, we used Dell R420 servers with multiple Ethernet ports. All servers have 4GB RAM and 8 Intel Xeon processors on each of them. Ubuntu 14.04 server was used as the operating system running on each of them. After the setup of virtual machines, IP association is done and testing for the connection is done using Address resolution protocol (ARP). Java runtime environment and MYSQL are deployed on all the nodes for multiple operations. After the setup of all three nodes using Openstack with VMs, we run the arbitrary computation tasks on the multiple VMs hosted on the compute node. The snapshots are captured as per pre-defined schedule using scripts. After that extensive testing is performed on financial application, where one of the VMs are triggered with configuration errors that is followed by powering off that VM. At that point, snapshot are retrieved for that VM and passive data block are activated. This produce a replicated system of powered off VM. By using this method, we are able to test the reliability of time constraint data and data critical application.

In our disaster recovery experiments, we checked if the data can be retrieved by using snapshot and volume replication technique then it will generate the model for disaster recovery within the Cloud. On the basis of data retrieval following the VM failures, we also analysed the best replication strategy that can be used in Cloud environment. A typical comparison of snapshot and volume replication will be part of the evaluation process that will yield best disaster recovery solution. The baseline used to check the best replication strategy would be synchronous data recovery.

There were four VMs deployed in the Cloud. Three of them were virtual machines instances and the fourth system was an instance being launched using a volume block created containing the Ubuntu image. Of the three virtual machine instances deployed, two of them were clients and the third was a server. The fourth instance was used as a server as well, but deployed using a volume block. Cinder would be used to take a snapshot of the instance deployed from the volume block and Swift would be used to take a snapshot of the server virtual machine instance. After the systems were deployed, a simple RMI application was developed and deployed on the instances to ensure they could communicate and transfer data between each other.

We created multiple shell scripts for automation of snapshot and recovery process. These scripts were helpful to generate timely triggers to create and delete snapshots. We also created script for meaningful data generation on Linux system. Scripts that take snapshots of each instance at timed intervals and

 ISBN: 978-1-61208-460-2

maintain a specified number of backups were needed to perform the experiment. For object storage, there were three scripts created for taking a snapshot of the instance. The first script would maintain the iterations, decide which instance to target for the snapshot, and finally launch the other two scripts over every iteration which performed the actual snapshot. The second script would first delete any old snapshots exceeding their lifespan and then create a new snapshot. The third script was used to force the data to be written into memory and prevent the instance from accepting input while the snapshot is taken to prevent any data inconsistencies as this was documented as best practice in the OpenStack documentation [14]. For taking a snapshot using block storage, there were two scripts associated with the process.

Again, one script was used to maintain the iterations, which volume to target, and to launch the other script at timed intervals. The second script would delete old snapshots exceeding their lifespan and then create a new snapshot of the volume. Generating data for the experiment manually would have been a time consuming process. For this reason a script was developed in order to create text files with readable information within in order to have sample data during each iteration. Due to the time it took to create the script for generating the data, a snapshot was taken every two hours instead of every three hours. Here is a tabular form of the data collected during the experiment.

TABLE I
VM ACCESS AND PREPARATION

Iteration	Test-Mem	Mem-G	ST	VT	SD	VD
I-1	100 MB	105.4 MB	108	1	143	13
I-2	200 MB	207.6 MB	128	2	147	13
I-3	300 MB	309.8 MB	151	2	150	13
I-4	400 MB	411.0 MB	170	3	153	13
I-5	500 MB	513.2 MB	193	3	155	13

Table-1 is used to determine which methodology, object storage or block storage, is faster in terms of the time it takes to create a snapshot, the time it takes to deploy a snapshot and the amount of data associated with each iteration. Here Test-Mem denotes Test memory used as financial data block size, Mem-G denotes memory generated after taking snapshot, ST refers to time taken to create VM snaphsot(in seconds), VT is time taken to create Volume snapshot(in seconds), SD and VT are the time taken to deploy Vm and volume snapshot respectively.

TABLE II
VM RECOVERY DATA

Iteration	Tot-Bytes	V-Snap	Lost-Vm	V-vm	Lost-vol
I-1	10538598	10538598	0	10538598	0
I-2	20762214	20762214	0	20762214	0
I-3	25480806	25480806	0	25480806	0
I-4	15257190	15257190	0	15257190	0
I-5	26929875	26929875	0	26929875	0

Table-2 is used to determine which methodology, object storage or block storage, is better at preserving consistent data by measuring how many bytes of data are lost at each iteration.In this table, V-snap refers to bytes in Vm snapshot, Lost-Vm denotes bytes lost in Vm snapshot after deploying as new Vm, V-Vm refers to bytes in volume snapshot and Lost-V as bytes lost after deployment of new Vm using volume snapshot.

V. RESULTS

Analyzing the results from Figure 3. we can see that in terms of creating a backup, and deploying the backup, block storage (Cinder) is much faster than object storage (Swift). For an organization, this is a significant factor to consider when creating a disaster recovery plan because we want to make backups quickly to minimize overhead on the system so there is minimal interference with service quality. The system can be recovered a lot quicker using Cinder instead of Swift which means the system will be down for shorter periods of time. After some statistical calculations it was determined that using Cinder to deploy a backup is approximately 53 times faster than deploying a backup using Swift. As more data is stored on these systems we can expect the time to create a backup and backup deployment to increase. Let us assume the ratio calculated holds for various sizes of data. If it takes one minute to deploy a backup using block storage, then we can expect a deployment using object storage to take approximately 53 minutes. This is a significant difference and having a service down for approximately an hour may not be acceptable to an organization who relies heavily on their IT infrastructure. In terms of recovering a system from a disaster, it appears using block storage is the better option.

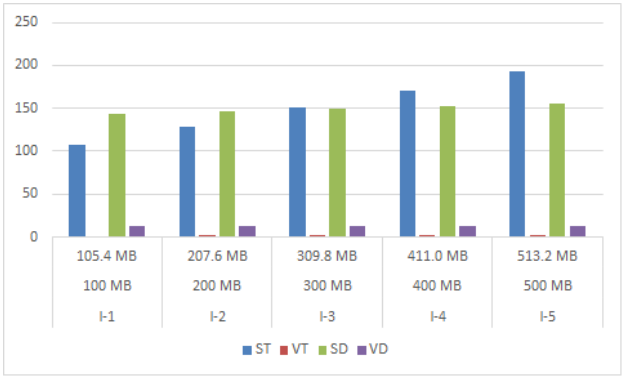

Figure. 3. VM preparation analysis

Analyzing the results in Figure 4. we can observe that both components, Cinder and Swift, experienced no data loss. As the amount of data increases on each system, it would be expected Swift and Cinder would begin to experience some data loss. Therefore we can conclude, based on the results, that both Cinder and Swift are effective in preserving small sets of data. In future experiments, bigger sizes of data would be used to test how much data we can have before we experience loss from using Cinder and Swift. We would expect to lose data when more data is stored on the system when using Swift or

 ISBN: 978-1-61208-460-2

Cinder but the goal would be to determine which component provides more consistent data when it gets large. In terms of providing consistent data, for this experiment, both Swift and Cinder performed equally well.

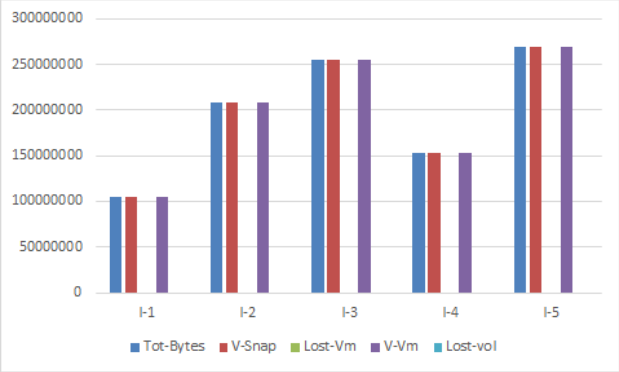

Figure. 4. VM data recovery analysis

Based on the results from Table 1 and 2, we can conclude that if our main objective of a disaster recovery plan is to restore service as quickly as possible then Cinder would be the correct path to go down. We can also conclude that both components function with equal reliability if our system is small. At the beginning all systems begin relatively small therefore if we have 2 options with both providing equal reliability and Cinder being 53 times faster than Swift, then Cinder would be the optimal choice at that particular point in time. As the system grows, more experiments could be performed to help determine which provides more consistent data as the size of the data grows and determine the trade-offs if any.

VI. Conclusion

This paper shows how private cloud built on Openstack platform was used to identify new unique strategy of volume replication. It provides efficient storage mechanism in comparison to object based replication. In consideration with strategies used in recent times, this study can help derive an effective replication solution for openstack private cloud. Given the following mechanism, we can implement similar strategies in different private Cloud platforms such as Cloudstack, OpenNebula and Eucalyptus. We can see some limitations in public clouds as datacenters can be situated in different geo-locations which can add to hop count wait time and produce delay in replication.

In conclusion, the experiment was a success for the reason being that we wanted to find which method, block storage or object storage, provided the best disaster recovery plan. In this particular environment with a relatively small dataset Cinder was proven to be the preferable choice as it held no additional advantage over Swift in terms of data consistency but it had a significant advantage in terms of the time it takes to create a backup and deploy it. Overall, we have achieved both goals for this study. We were able to build an environment for the purposes of deploying a Private Cloud, deploying a financial application utilizing RMI, and creating scripts for the purpose of managing backups of systems deployed in the Private Cloud. This work can be explored in future for the other private Clouds and Inter-Clouds. This strategy can also be used on Bigdata, which can produce interesting result.

Acknowledgment

The first author acknowledges graduate enhancement of tri-council stipends (GETS) from the Faculty of Graduate Studies, University of Manitoba. The second author acknowledges Natural Sciences and Engineering Research Council (NSERC) Canada and University of Manitoba for partial financial support for this research through Discovery Grant and University Research Grant Programs.

References

[1] M. Mesnier, G. R. Ganger, and E. Riedel, "Object-based storage," *Communications Magazine, IEEE*, vol. 41, no. 8, pp. 84–90, 2003.

[2] A. N. Toosi, R. K. Thulasiram, and R. Buyya, "Financial option market model for federated cloud environments," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2012, pp. 3–12.

[3] G. Suciu, E. G. Ularu, and R. Craciunescu, "Public versus private cloud adoptiona case study based on open source cloud platforms," in *Telecommunications Forum (TELFOR), 2012 20th*. IEEE, 2012, pp. 494–497.

[4] S. Srirama, O. Batrashev, and E. Vainikko, "Scicloud: scientific computing on the cloud," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 579–580.

[5] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*. IEEE, 2010, pp. 89–96.

[6] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster," in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*. IEEE, 2010, pp. 188–196.

[7] M. J. Brim, D. A. Dillow, S. Oral, B. W. Settlemyer, and F. Wang, "Asynchronous object storage with qos for scientific and commercial big data," in *Proceedings of the 8th Parallel Data Storage Workshop*. ACM, 2013, pp. 7–13.

[8] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.

[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 124–131.

[10] D. Milojičić, I. M. Llorente, and R. S. Montero, "Opennebula: A cloud management tool," *IEEE Internet Computing*, no. 2, pp. 11–14, 2011.

[11] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[12] V. Chang, "Towards a big data system disaster recovery in a private cloud," *Ad Hoc Networks*, vol. 35, pp. 65–82, 2015.

[13] V. Padhye and A. Tripathi, "Scalable transaction management with snapshot isolation on cloud data management systems," in *Cloud computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 542–549.

[14] K. Pepple, *Deploying openstack*. " O'Reilly Media, Inc.", 2011.

 ISBN: 978-1-61208-460-2