

International Journal of Advanced Research in Computer Science and Software Engineering

ISSN: 2277 128X

Research Paper

Available online at: www.ijarcsse.com

Trends towards Failover Techniques for Cloud Computing

Environment Radhika Kalyan*, Anil Kumar

Computer Science GNDU Punjab, India

Abstract-- Cloud computing is a technology that provides the processing huge amount of data which depend on sharing computing resources like networks, servers, storage, applications, and services. Cloud provider provides the various services to the users. The failures that occur in Cloud Computing can be classified into two classes. One of them is Data Failures which involves failures due to corruption of data, missing source data and other flaws in the data. Other is Computation Failures which involves all types of hardware or infrastructure failures like faulty or slow VMs, storage access exception, etc. To overcome various failures in cloud computing, failover techniques are used. There are different types of failover techniques. One of them is Fault tolerance in which the ability of the Cloud to withstand the rapid changes which occur due to hardware faults, software faults, network congestions etc. Other is Fault management depends on two important parameters namely Recovery Point Objective (RPO) and Recovery Time Objective (RTO). The metric RPO defines the amount of data to be lost during a fault or disaster whereas RTO determines the minimum downtime for recovering from faults. This paper has focused on high availability techniques of cloud computing to evaluate the various gaps in existing research on cloud computing.

Keywords-- Cloud Computing, Failures, Fault Tolerance

I. INTRODUCTION

Cloud Computing is a model for supporting useful, on-demand network access to a shared group of configurable computing resources which can be quickly provisioned and released with least management effort or service provider interaction. The different services provided by Cloud Computing. On the basis of the services provided, Cloud is differentiated into Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In SaaS, Cloud providers develop, host and operate domain specific applications which may be accessed by the end users [1]. In PaaS, the platform is provided as service where application developers can build application without any stress of managing or underlying developing tools. And in IaaS, infrastructure resources like operating system, storage, processors and networking components are offered as service where companies can deploy and run arbitrary software.

A. Types of cloud

This section introduces the various kinds of cloud models. Generally, there are four types of cloud models: public cloud, private cloud, hybrid cloud and community cloud [2].

1) Public cloud

Public cloud is a form of cloud where computing resources are available over internet for multiple users and are organized and managed by some third party. The third party mentioned here are the cloud service providers, who provide on-demand scalable and flexible services to the client. This model is managed by some business, academic, or government organization or their combination. It exists on the premises of the cloud provider. This is the most used model among all the models as it helps the consumers deploy a service in the cloud with a very good cost. Most computer users have been using a few of the software and services provided by this model such as Google Drive, Dropbox, and Gmail.

2) Private cloud

Private cloud is a computing model where the cloud infrastructure is deployed and provisioned for a single organization. It is also defined as the internal data centers of a business that aren't available to general public. Using private cloud will be a better choice when consumers have to manage large data sets where security and data privacy is a primary concern.

3) Hybrid cloud

As the name itself suggests, hybrid cloud is a mix of several distinct models. Hybrid cloud is a model where a cloud infrastructure is formed by the combination of public and private cloud models which can be in-house or provided externally. This cloud model is wonderful for the organization that is more prone to move public cloud to private and vice-versa across plenty of different resources. By the utilization of portability they meant that any organizations whose data and applications are hosted in public cloud can shift in private cloud according to their need. Generally, private and hybrid cloud infrastructures are only used for specific business purposes, where public cloud is not a suitable choice.

4) Community cloud

Community cloud is a cloud model where the computing resources are shared among several organizations and is managed by a number of participating organizations. This cloud model is basically designed for working in a joint project. For example, a joint software development environment for an open source project community.

B. Service Models in Cloud Computing

Cloud Computing provides various types of service models. Some of them are as follows:-

1) Platform as a Service (PaaS)

The potential provided to the consumer is obviously to deploy onto the cloud infrastructure consumer-produced or acquired applications created by using encoding languages, class libraries, system services, and tools prolonged by the service provider. The client doesn't administer or command the underlying cloud infrastructure including network, servers, systems, or storage, but has domination on the deployed applications and perhaps configuration settings for the application form hosting background.

2) Infrastructure as a Service (IaaS)

The facility provided to the final user is definitely require to dispensation, storage, networks, and other elemental computing resources where the final user has the ability to deploy and run random software that could comprise systems and applications. The final user doesn't supervise or cope with the fundamental cloud infrastructure although have control upon systems, mass-storage, and deployed applications; and probably inadequate control of selected networking components for example host firewalls.

3) Software as a Service (SaaS)

The facility provided to the consumer is utilizing the service provider's applications running on a cloud infrastructure. Most widespread forms of such web-based applications are Facebook, Salesforce.com, Google Apps, SAP, Taleo, and WebEx etc. Each one of these web applications are full-service applications and are normally obtainable from everywhere on the Internet. End user doesn't cope with or organize the fundamental cloud infrastructure which include system network, web servers systems, mass-storage, in addition to person application capabilities, with the promising exemption of restricted user-defined application design settings.

II. FAILURES IN CLOUD COMPUTING

The failures that occur in Cloud Computing can be categorized into two classes to be precise [1]:

A. Data Failures

This type involves failures due to exploitation of data, omitted source data and other faults in the data.

B. Computation Failures

It involves all types of hardware or infrastructure disasters like defective VMs, storage access exception, etc.

The majority of the claims hosted by the Cloud are real-time High Performance Computing (HPC) systems which involve higher level of fault tolerance. The majority of the faults in Cloud occur as a result of following:

1) Hardware Faults

Hardware faults mainly occurs in processors, hard disk drive, integrated circuits sockets and memory. There are large number of processors provisioned in Cloud. These processors create virtual instances, communication links, integrate circuit sockets etc. These processors are prone to failure. It has been predicted that a system with 1,00,000 processors will experience a processor failure every few minutes.

2) Software Faults

Software faults cause application failure.

3) Network Faults

Network faults occurs due to server overload, network congestion etc. which inhibits the communication between the Cloud and the end users.

III. THREE-NODE ARCHITECTURE

Figure Fig. 2demonstrate the 3-tier architecture of cloud computing environment. In Fig. 1 there is a request manager (central cloud), clients send their requests to it [3]. Then request manager first divide the given job into threads and then allocate one of the sub cloud (service manager) to the threads and global checkpoint will be updated.

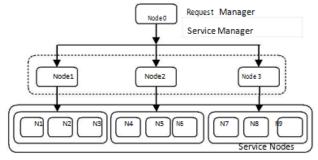


Fig 1: 3-node architecture

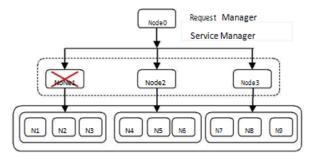
Each sub cloud first selects threads in First Come First Serve (FCFS) fashion and allocate lightly loaded service node (service node) to it. The service node then start execution of that thread or it may add this thread in its waiting queue if it is already doing execution of any other thread. N1 to N9 are service nodes which will provide services to the clients.

Sub cloud failure

Fig. 2 illustrates the sub cloud failure. In Figure Fig.2 sub cloud 2 has been failed. Sub cloud 2 failed means any node that belong to failed sub cloud is no longer available is to provide any service to the clients. In existing techniques there exist not such algorithm which migrate the exact load of all node to other nodes except redundant node technique. It means it must have protection (secondary) node to take load of primary node in the event of failure.

Service node failure

Fig. 3 illustrates the service node failure. In Fig. 3 service node (N5) has failed which belong to sub cloud 2. N5 failed means it is no longer available to provide any service to the demand of clients. In existing techniques there exist not such algorithms which will migrate the load of N5 to other nodes. However check pointing without load balancing can achieve this task but it is founded on random decision means load of failed node may be shifted to another heavily loaded node than lightly loaded nodes.



Service Nodes Fig 2: Sub cloud failure

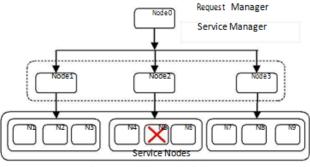


Fig 3: Service node failure

Job restartation due to node failure

Fig. 4 is illustrating the parallel job execution in distributed environment. Initially clients submits their jobs for execut ion, if no sub cloud is free then jobs are queued after this phase node filtering will be done that will detect the currently active nodes by checking the status of all nodes. After node filtering load balancing algorithm will come into action to balance the load of the given jobs among active nodes. It also shows that the main source of parallelism is provided by the Fork and Join concept, whose role is to split each job into multiple subsets which can be executed on multiple nodes in parallel. After successful execution of threads, each node send its final results back to the sub cloud, then sub cloud conquer the results of threads using Join concept and the output will be passed to the client.

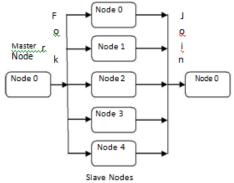


Fig.4: Parallel job execution in distribution environment

Job restartation

In Fig. 5, node 2 has been failed. As existing methods do not use check pointing therefore it is not feasible to migrate the load of failed node (node 2) to other active nodes

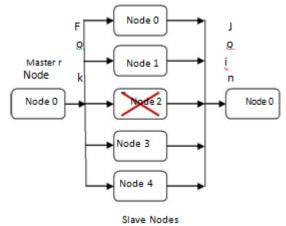


Fig.5: Job restartation

The major problem is that if other nodes successfully accomplish the execution of the threads belong to same job, will be wasted as no log files are used. Therefore job restartation will be there. In Fig. 5, failure of node 2 will waste the successful execution of the other nodes. Therefore a single node failure results in too much loss of computation. So this can be a major drawback of existing methods.

IV. FAULT TOLERANCE

Fault Tolerance (FT) deals with quick repairing and replacement of defective devices to maintain the system. Whereas in Cloud Computing, fault tolerance is the facility of the Cloud to withstand the rapid changes which occur due to hardware faults, software faults, network congestions etc[1].

In a Cloud network, fault management depends on two important parameters:

- (a) **Recovery Point Objective:** Recovery Point Objective(RPO) defines the quantity of data to be lost during a fault or disaster.
- **(b) Recovery Time Objective:** Recovery Time Objective (RTO) determines the minimum downtime for recovering from faults.

Fault Tolerance Policies

There are mainly two Fault Tolerant (FT) policies accessible for real-time applications presented in Cloud. Based on these fault tolerant policies, various techniques are accustomed to provide fault tolerance[1].

A. Proactive Fault Tolerance

Proactive fault tolerance policy is to avoid failures by proactively taking preventative measures. These measures are reserved by studying the pre-fault indicators and predicting the underlying faults. The second step is to apply proactive remedial measures at the development time by changing the code or replacing the components which are prone to failure. Preemptive migration and Software Rejuvenation are two fault tolerant techniques which are used based on proactive fault tolerance policy. Preemptive Migration employs feedback-loop control system where applications are constantly monitored and analyzed. Software Rejuvenation is a technique in which periodic reboots are scheduled for the system. After each reboot, the system resumes with a clean state.

B. Reactive Fault Tolerance

Reactive fault tolerance policy deals with measures which are applied to reduce the effect of the faults already occurred in Cloud. A few of the fault tolerant techniques which are used on the basis of the reactive fault tolerance policy are Check pointing/Restart, Replication & Task Resubmission. In Check pointing/Restart when a failure occurs, the applications can be restarted from the checkpoint before the point of failure rather than rebooting from starting point. Replication is a popular fault tolerance technique used based on reactive fault tolerance policy. Replication in Cloud Computing is the process of keeping multiple copies of data or object. In a replication technique, client requests for a copy from a couple of replicas. In Task Resubmission when a fault is detected, the task is submitted either to the same or to another resource at a runtime without interrupting the workflow of the system.

V. RELATED WORK

A.Ganesh et al. [1] has highlight the basic concepts of fault tolerance by accepting the different FT policies like Reactive FT policy and Proactive FT policy and the related FT techniques used on several types of faults. A study on numerous fault tolerant methods, algorithms, frameworks etc., has been carried out which are developed and implemented by research experts in this field. This was an area where lot of research occurred and these studies guided

us to build a robust FT technique in Cloud. D.Singh et al. [3] has improved the checkpoint efficiency for integrated multilevel checkpointing algorithms and prevent checkpointing from becoming the bottleneck of cloud data centers. To be able to find an efficient checkpoint interval, checkpointing overheads has also considered in this paper. Traditional checkpointing methods stores persistently snapshots of the present job state and use them for resuming the execution at a later time. The attention of this research is strategies for deciding when and whether a checkpoint should be taken and evaluating them regarding minimizing the induced monetary costs. By varying rerun time of checkpoints performance comparisons are which is used to evaluate optimal checkpoint interval. The purposed fail-over strategy will work on application layer and provide highly availability for Platform as a Service (PaaS) feature of cloud computing. L.Jian et al. [4] has a scalable failover method for large scale DCNs. Because all the current DCNs were achieved in a logically centralized manner with a specialized topology and growth model, Fat-Tree as the reference DCN topology was adopted and design failover method using an OpenFlow-based approach. Further, to provide scalability, they design failover algorithm in a local optimal manner, with which only three switches must be modified for handling a single fault, whatever the size of the target network. They estimated their failover method with regards to failover time by varying the network size and load balancing capability during failover. The experiment results show that their technique scales well, even for a large-scale DCN with more than ten thousands hosts. W.Liying et al. [5] has discussed about data fault-tolerance of cloud storage, they had proposed a dynamic data fault-tolerance for cloud storage(DDFMCS) that dynamically describes the data fault tolerance mechanisms by the file access frequency ratio kept in the file access frequency table, the number of file fault tolerance conversions and the time of files stored in the system. They implement DDFMCS based on Hadoop and conduct some experiments. Experimental result shows that DDFMCS can advance the exploitation of cloud storage space and improve data access performance. I.Egwutuoha et al. [6] has described fault tolerance framework for high performance computing in Cloud. This framework proposes using process level redundancy (PLR) techniques to reduce the wall clock time of the execution of computational intensive applications. A fault tolerance framework for HPC in cloud. Their earliest experimental results show that overhead associated with Checkpointing can be significantly reduced by 40% with the PLR approach proposed in that paper. They had planned to form a prototype that fully implements this framework as their future work. L.Min et al. [7] has used the Queuing Petri nets to model the hybrid cloud which can definitely analyze the performance. Based on the real time performance analysis from the model, they had proposed a fault tolerant strategy for the hybrid cloud computing which consider both performance and request completion. The results simulated by QPME show the improvement of their strategy. P.Bellavista et at. [8] has presented novel framework for Easy Monitoring and Management of IaaS (EMMI) solutions, with the main objective of making easier the adoption of the private cloud paradigm. The EMMI framework is based on open-source software components, i.e., OpenStack for IaaS management, Collected for distributed system monitoring, and Apache jk for load balancing, allowing to adopt the IaaS model easily and in a cost effective manner. The reported performance results establish that the EMMI framework efficiently supports two primary features of the IaaS model, i.e., failover and scale-out, promptly identifying crucial events and autonomously taking proper countermeasures. M. Fares et al. [9]they has shown how to leverage mainly commodity Ethernet switches to support the full aggregate bandwidth of clusters comprising of tens of thousands of elements. Comparable to how clusters of commodity computers have mainly replaced more specialized SMPs and MPPs, they argue that properly architected and interconnected commodity switches may distribute more performance at less cost than available from today's higher-end solutions. Their approach involves no modifications to the end host network interface, operating system, or applications; critically, it is fully backward compatible with Ethernet, IP, and TCP. J.Jaroodi et al. [10] has discussed delay-tolerant fault tolerance algorithm that effectively reduces execution time and adapts for failures while minimizing the fault discovery and recovery overhead in the Cloud. Distributed tasks that can use this algorithm contain downloading data from replicated servers and executing parallel applications on multiple independent distributed servers in the Cloud. The experimental results show the efficiency of the algorithm and its fault tolerance feature. R. Ramos et al. [11] has described a data center networking approach based on encoded paths accepted in the packet header. They presented an OpenFlow-based test bed implementation of a data center architecture governed by a logically centralized Network Manager, which is accountable to transparently provide the networking and support functions to operate the data center network. They evaluate the proposal in terms of fail recovery time, state requirements, and load balancing capabilities. The results of the experiments show that their proposal improves the fail recovery time, while preserving scalability requirements. D. Singh et al. [12]has presented an approach for providing high availability to the requirements of cloud's clients. To achieve this objective, failover strategies for cloud computing using integrated checkpointing algorithms were purposed. Purposed strategy integrate checkpointing feature with load balancing algorithms and also make multilevel checkpoint to decrease checkpointing overheads. For implementation of purposed failover strategies, a cloud simulation environment is developed, which has the ability to provide high availability to clients in case of failure/recovery of service nodes. Also in this paper comparison of developed simulator is made with existing methods. The purposed failover strategy work on application layer and provide highly availability for Platform as a Service (PaaS) feature of cloud computing. R. Niranjan et al.[13] considers the requirements for a scalable, easily manageable, fault-tolerant, and efficient data center network fabric. Trends in multi-core processors, end-host virtualization, and commodities of scale are pointing to future single-site data center with millions of virtual end points. Existing layer 2 and layer 3 network protocols face some combination of limitations in such a setting: lack of scalability, difficult management, inflexible communication, or limited support for virtual machine migration. To some extent, these limitations may be inherent for Ethernet/IP style protocols when trying to support arbitrary topologies. We observe that data center networks are often managed as a single logical network fabric

with a known baseline topology and growth model. We leverage this observation in the design and implementation of PortLand, a scalable, fault tolerant layer 2 routing and forwarding protocol for data center environments. Through our implementation and evaluation, we show that PortLand holds promise for supporting a "plug-and-play" large-scale, data center network. S.Sharma et al.[14]has centers around fault tolerance of OpenFlow to deploy it in carrier-grade networks. The carrier-grade network has a strict requirement that the network should recover from the failure in just a 50 ms interval. They apply two well-known recovery mechanisms to OpenFlow networks: restoration and protection, and run extensive emulation experiments. In OpenFlow, the controlling software is moved to one or more hardware modules (controllers) which could control many switches. For fast failure recovery, the controller must notify all the affected switches in regards to the recovery action within ms interval. This contributes to an important load on the controller. They show that OpenFlow may not manage to achieve failure recovery in just a 50 ms interval in this situation. They add the recovery action in the switches themselves so the switches can perform recovery without contacting the controller. This method can achieve recovery within 50 ms in a large-scale network serving many flows. T.Sara et al.[15]explains the feasibility of the deployment of multicloud infrastructure spanning an area data center (in-house infrastructure) and two different cloud sites, Amazon and Rackspace (external resources). The clustering infrastructure can overcome any failure caused in nodes. Therefore, configuration of the clustered nodes of local data center and both different cloud sites can improve cost efficiency of the deployment and helps in increasing throughput while solving applications involving a variety of tasks. Whilst the interfaces to cloud providers are very different, a standardized API is highly required, which stands as a limitation. The job assures the credibility of the adopted multicloud system, and it's scalable and flexible with regards to data availability and resource usability. P. Mookdarsanit et al.[16]has introduce the technique for designing an area failover system for Cloud. Since a failover system has less resource and limited scalability; it cannot handle all workloads previously on the Cloud. They overcome this drawback by adapting the total operation performed in the Cloud right into a light-weight operation optimized for a failover system. A light-weight operation consumes less resource but even offers reduced quality of service (QoS) compared fully operation. They also define a write kind of data according to alter and future usage of that data. They adapt the total operation right into a lightweight one by reducing the amount of write types or changing one type to another. Put simply, reduce the caliber of service of the device to be able to serve more requests. Experimental result shows that the system with full operation can sustain typically 472 maximum amounts of requests, an9d a method with light-weight operation can sustain 844 requests, an improvement of 179%. J. Xiong et al. [17] has present a replication-based data availability mechanism made for a large-scale cluster file system prototype named LionFS. Unlike other replicated storage systems that serialize replica updates, LionFS introduces a comfortable consistency model allow concurrent updating all replicas for a mutation operation, greatly reducing the latency of operations. LionFS ensures replica consistency if applications use file locks to synchronize the concurrent conflict mutations. Another novelty with this mechanism is its light-weight log, which only records failed mutations and imposes no overhead on failure-free execution and low overhead when some storage products are unavailable. Furthermore, recovery of replica consistency needs not stop the file system services and running applications. Performance evaluation shows which our solution achieves 50-70% higher write performance than serial replica updates. The logging overhead is proven to be low, and the recovery time is proportional to the total amount of data written throughout the failure.

VI. LIMITATIONS OF EARLIER TECHNIQUES

After reviewing the literature, the following limitations have been encountered. We will try to overcome these limitations in the future:

- ☐ The use of Secondary nodes for failover has been neglected.
- The detection of failure techniques has also been neglected which may cause the issue of too much delay.
- ☐ The load migration has not been utilized in majority of research which may leads for poor response time.

VII. CONCLUSION AND FUTURE WORK

This paper has presented a review on various cloud computing techniques. After reviewing the literature the various limitations have been encountered. The use of Secondary nodes for failover has been neglected in the majority of research. The detection of failure techniques has also been neglected which may cause the issue of too much delay. The load migration has not been utilized in majority of research which may leads for poor response time. This work has not considered any improvement over the existing failover techniques. So in order to improve the failover techniques further a secondary node based environment will be proposed. The secondary node will come in action whenever any failure exists in data centers. In mean time secondary nodes stay in sleep mode to save energy.

REFERENCES

- [1] Ganesh, Amal, M. Sandhya, and Sharmila Shankar. "A study on fault tolerance methods in Cloud Computing." In *Advance Computing Conference (IACC)*, 2014 IEEE International, pp. 844-849. IEEE, 2014.
- [2] K. Shah, "Survey on cloud based testing tools." (2014).
- [3] Singh, Dlbag, Jaswinder Singh, and Amit Chhabra. "UML Based Integrated Multilevel Checkpointing Algorithms for Cloud Computing Environment." *International Journal of Computer Network & Information Security* 4, no. 7 (2012).

Kalyan et al., International Journal of Advanced Research in Computer Science and Software Engineering 5(1), January - 2015, pp. 983-989

- [4] Li, Jian JongHwan Hyun, Jae-HyoungYoo, SeongbokBaik, and James Won-Ki Hong. "Scalable Failover Method for Data Center Networks Using OpenFlow." In *Network Operations and Management Symposium* (NOMS), 2014 IEEE, pp. 1-6. IEEE, 2014.
- [5] Wu, Liying, Bo Liu, and Weiwei Lin. "A Dynamic Data Fault-Tolerance Mechanism for Cloud Storage." In *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*, pp. 95-99. IEEE, 2013.
- [6] Egwutuoha, Ifeanyi P., Shiping Chen, David Levy, and Bran Selic. "A fault tolerance framework for high performance computing in cloud." In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pp. 709-710. IEEE, 2012.
- [7] Lu, Min, and Huiqun Yu. "A Fault Tolerant Strategy in Hybrid Cloud Based on QPN Performance Model." In *Information Science and Applications (ICISA), 2013 International Conference on*, pp. 1-7. IEEE, 2013.
- [8] Bellavista, Paolo, Carlo Giannelli, and MassimilianoMattetti. "A practical approach to easily monitoring and managing IaaS environments." In *Computers and Communications (ISCC)*, 2013 IEEE Symposium on, pp. 000016- 000021. IEEE, 2013.
- [9] Al-Fares, Mohammad, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture." In *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63-74. ACM, 2008.
- [10] Al-Jaroodi, Jameela, Nader Mohamed, and Klaithem Al Nuaimi. "An Efficient Fault-Tolerant Algorithm for Distributed Cloud Services." In *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, pp. 1-8. IEEE, 2012.
- [11] Ramos, Ramon Marques, MagnosMartinello, and Christian Esteve Rothenberg. "Data Center Fault-Tolerant Routing and Forwarding: An Approach Based on Encoded Paths." In *Dependable Computing (LADC), 2013 Sixth Latin- American Symposium on*, pp. 104-113. IEEE, 2013.
- [12] Singh, Dilbag, Jaswinder Singh, and AmitChhabra. "High Availability of Clouds: Failover Strategies for Cloud Computing Using Integrated Checkpointing Algorithms." In *Communication Systems and Network Technologies (CSNT)*, 2012 International Conference on, pp. 698-703. IEEE, 2012.
- [13] Niranjan Mysore, Radhika, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. "Portland: a scalable fault-tolerant layer 2 data center network fabric." In *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 39-50. ACM, 2009.
- [14] Sharma, Sachin, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "OpenFlow: Meeting carrier- grade recovery requirements." *Computer Communications* 36, no. 6 (2013): 656-665.
- [15] George, Talit Sara, and V. P. Sreekantha Kumar. "Multicloud computing for on-demand resource provisioning using clustering." In *Sustainable Energy and Intelligent Systems (SEISCON 2012), IET Chennai 3rd International on*, pp. 1-6. IET, 2012.
- [16] Mookdarsanit, Pakpoom, and Sethavidh Gertphol. "Light-weight operation of a failover system for Cloud computing." In *Knowledge and Smart Technology (KST), 2013 5th International Conference on*, pp. 42-46. IEEE, 2013.
- [17] Xiong, Jin, Jianyu Li, Rongfeng Tang, and Yiming Hu. "Improving data availability for a cluster file system through replication." In *Parallel and Distributed Processing*, 2008. IPDPS 2008. IEEE International Symposium on, pp. 1-8. IEEE, 2008.