ISSN: 2321-7782 (Online)
Volume 2, Issue 11, November 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

High Availability of Cloud through Different Self-Healing and Consistency Mechanism

Megha Balnath Rode[1]
Department of Computer Network
G.H. Raisoni College of Engineering & Management
Chas, Ahmednagar
University of Pune, Maharashtra, India

Amruta Amune[2]
Department of Computer Network
G.H. Raisoni College of Engineering & Management
Chas, Ahmednagar
University of Pune, Maharashtra, India

Abstract: Cloud computing IAAS has became one of the main building blocks in many IT companies. Neglecting the middleware availability can be a major issue if not properly handled while developing the cloud services which may lead to partial or total failure of cloud.

In this paper, we present a novel scalable and highly available multi-master pattern for cloud middleware. Load balancing and data loss are the important factors for making the middleware more fault-tolerant. Backup server can be implemented to avoid the data loss.

Keywords: cloud computing, Resource allocation, self Healing, High availability

I. Introduction

Today a growing number of companies have to process large amounts of data in a cost-efficient manner these companies are operators on Internet search engines like Google, Microsoft or Yahoo. Running applications in the cloud on a large number of shared-nothing commodity computers makes fault tolerance an important issue. Therefore, failure prevention in Cloud Computing has become a well discussed topic in the last years and is differentiated in several types: power failures, server failures, network failures, and management software failures[1].

Cloud computing systems fundamentally provide access to large amount data and computational resources through a variety of interfaces similar in spirit to HPC resource management and programming systems. These types of services offer a new programming output for scalable application developers and have gained popularity over the past few years. However, most of the cloud computing systems in operation today depend upon infrastructure that is invisible to the research community. Especially on the Infrastructure-as-a-Service (IaaS) layer fault tolerance is an important issue in recent years, because failures in this layer result in loss of data, error-proneness and poor quality of services. Depending on the application running on the infrastructure a poor quality of service may be for example data loss or a long off time of a virtual machine. For IaaS providers it is highly important to prevent failures or at least recover from failures transparent to the user and as fast as possible. At the same time the solution should be feasible without special hardware or complex changes in the cloud infrastructure.

The described approaches based on our cloud middleware which is similar to the known IaaS platforms like Eucalyptus, Open-Stack, CloudDisco, snooze, Nephele, or Hadoop . In this paper we focus on Management Software Failures which can be subdivided in the following three levels:

1. Virtual Machine Level: Error In this level is a Crashed VM which gives results in data loss.
2. Cluster Level: Errors in this level is a node or even a whole bunch of nodes are no longer together.
3. Cloud Level: An error in this level is results in unavailability of the entire cloud.

Common Iaas middleware they are typically built in master-worker architecture. It consist of single master instance is the cloud interface called as Cloud Controller. This interface between the cloud environment users are comminuting with cloud controller which should not only provide high availability support but also scalability and consistency mechanisms to achieve quality of service requirements.

In Eucalyptus [2] architecture the Cloud Controller receives requests from the user which gives available list of resources and Transfer to them into its Cluster Controllers. The Cluster Controllers is then translating this request to the Node Controllers, which execute the request and provide the resource allocation information. In such architecture a failure of the Cloud Controller is an error in the Cloud Level which leads to an outage of the entire cloud and a failure in a Cluster Controller Which leads to an error in the Cluster Level and it will affect only an individual part of the cloud.

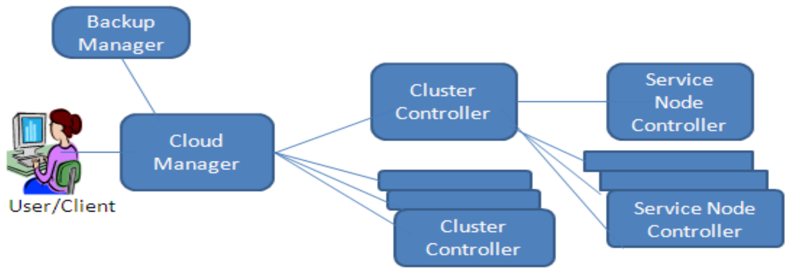

Fig .1 Middleware Master-Worker architecture.

Similarly in Cloud Disco[3] middleware stack is hierarchical design architecture is to establish a cloud for sharing virtual hardware components .The cloud Manager is the interface for the user request it does not managing any resources directly but also gives establishment of the cloud resources.The cluster controller which is responsible for scheduling and managing access. The Node controller which is responsible for the connected resources and invoking the cloud based connection of the hardware.

In Snooze [4] the Group Leader (GL) is the interface for the user to request a list of available Resources as well as to setup the connection .The Group Leader receiving a request for all available Resources sent it to all connected Group Manager (GM). The Group Manager collects the list of components and returns it to the Group Leader which passes it back to the user. If the Group Leader gets unavailable the whole service becomes unavailable, even though the Group Manager and thus the resources are still available.The Local Controller (LC) performing monitoring and reporting all information to assigned group Manager.

In Nephele's[5] architecture follows a classic master-worker pattern. The Job Manager (JM) receives the User's jobs which is responsible for scheduling .It is communicating with the interface to the cloud operator is nothing but the Cloud Controller. The Cloud Controller the Job Manager can allocate and deallocate VMs. Each instance runs a so-called Task Manager (TM). A Task Manager receives tasks from the Job Manager at a time, executes it and after that completion it will give or possible errors to the job manger.

The Hadoop Distributed File System (HDFS) [5] is a distributed file system designed to run on commodity hardware. HDFS has master/slave architecture. It consists of a single Name node or that manages the file system. There are a number of Data nodes which manage storage attached to the nodes that they run on. The Data Nodes are responsible for serving read and write requests from the file system's clients. Note that as the mentioned components have similar components or even equal functions within the cloud system and the architectures are same alike. Therefore the pattern described in this paper is also being adapted to any other IaaS platform. As the main benefit of the cloud computing nodes is the ability to parallel data processing applications and a failed machine usually leads to a loss of data and information. In order to achieve the fault tolerance it is necessary to make this all information available to the system again. Depending on the application, there are several approaches for this issue some of like replication in DDBMS [6] or saving of intermediate data in parallel data processing systems like Map Reduce [7].In this paper we describe a new technique to deal with machine failures in a multi-master cloud The disadvantage of the common architectures there is no mechanism to avoid redundancy for cluster controller to improve also its availability. If the connection from node controller to cluster controller is lost then there is possibility of error and data loss. There is very Lack of load balancing technique to distribute the user requests according to the load of the master nodes.

To prevent a partial or even an entire blackout of the cloud, we introduce and a new Load balancing algorithm for fair scheduling. It addresses the fairness issues by using mean waiting time. It scheduled the task by using fair completion time and rescheduled by using mean waiting time of each task to obtain load balance.

In this paper we describe a Dynamic Load Balancing Algorithm which is implemented to a multi-master pattern for cloud middleware's on resource type policy. Also We will implement a backup Manager which handle all the task of failed object so that there is guaranties no data loss as a backup server is doing all task.

The rest of this paper is organized as follows: section II discusses related work in this area and will show the differences to other approaches. Section III describes Proposed Work The section IV explains fair scheduling algorithm section V Dynamic Load Balancing Algorithm .section V I we conclude our results

II. Related Work

The importance for failover mechanism technique was recognized by Eucalyptus. It consist of Node Controller , Cluster Controller , Cloud Controller, and Walrus on a single machine called as a Controller which perform continuously monitoring and synchronization function and it is carried out between the primary Controller and the secondary Controller and it simulates in his evaluation of a fast switch over mechanism occurred which is in an error case in Eucalyptus. This approach has the some disadvantages with the high availability support a faster switchover mechanism are irrelevant, because it is an error case not only the interface to the cloud would be affected, but also additional components of the installation.

The CloudDisco [2] stack is a layered architecture consisting of Cloud Manager (CM), Cluster Controller (CC), and Node Controller (NC). On the top layer consist of the Cloud Manager represents the linking element of all other components, providing scheduling and monitoring negotiation function and identity management access control. The Cluster Controller which is used for the selection of computers offering their resources to the cloud which is responsible for scheduling and managing access. The Node controller which is responsible for the connected resources and invoking the cloud based connection of the hardware. This approach has the some disadvantages with the high availability support that is switchover mechanism affect on hardware components.

Nephele's architecture [5] consist cloud controller (CC), Job Manager (JM), Task manager(TM) and Persistence storage. Job manager perform continuously monitoring and scheduling function .Each instances runs so called as task manager a task manager receive one and more task form job manager execute it and after completion it gives possible error .so here also fast switchover mechanism used . This approach has the some disadvantages with the high availability support a faster switchover mechanism are irrelevant and it affect on all components.

Snooze's[4] architecture consist of Local Controller(LC), Group Manager(GM), Group Leader(GL) The principle consist of following instance and this has two states first it either runs in the Group Manager node or it run in the Group Leader node. If an error occurs at one of these components an event is failed .This event is followed by an election algorithm in which a new Group Leader will be elected by a given identifier .During this reconnection the Group Manager which will be the new Group Leader and it terminates all its tasks , all connections to the underlying Local Controller. After that, all Local Controller need to reconnection with the network need to establish new connections with another Group Manager. This approach improves the availability and scalability problem, because at any time only a one Group Leader is elected to provide all its functionalities for upper layer requests and all its Group Managers. In contrast this approach to multi-master pattern for cloud middleware's on resource type policy. Furthermore in our approach we implement backup server, which will handle all task of failed object so that it will guaranties no loss of data as backup server is doing all task

III. PROPOSED WORK

The pattern described in this paper is based on a multi-master in that communication in the middleware layer is discussed. It follows master worker pattern as depicted in figure 2 In contrast to master-slave architecture. These master nodes are called Cloud Manager (CM) are interconnected in a full-mesh topology and can communicate with each other. Each worker node called Cluster Controller (CC) it is connected to exactly one of the master nodes which announces and updates a list of all known CMs in the system since all CMs are equal a user as well as a CC can connect to any CM of the cloud. Middleware architecture in Multi-master pattern can be subdivided in the following four components.

1. *User/ Client:*

Its cloud user who registered with cloud system. To interact with cloud interface user has to interact to cloud manager first.

2. *Cloud Manager(CM):*

The single master instance is the cloud interface it is called Cloud Manager or Cloud Controller. This interface between the cloud environment and a user (client) is a significant component which should not only provide high availability support but also scalability mechanisms to achieve quality of service requirements the described approach can be adapted to these systems. The Cloud Manager is the interface for the user to request a list of available hardware as well as to setup the connection to one of the hardware components. The Cloud Manager receiving a request for all available hardware components sent it to all connected Cluster Controller. The Cluster Controller collects the list of components and returns it to the Cloud Manager, which passes it back to the user. If the Cloud Manager gets unavailable the whole service becomes unavailable, even though the Cluster Controller and thus the hardware components are still available

3. *Cluster Controller(CC)& Node Controller(NC):*

The Cloud Manager receives requests from the user that is list of available resources and propagates them to its Cluster Controllers. The Cluster Controllers in turn propagate the request to the Node Controllers which execute the request and provide the resource information. In such architecture a failure of the Cloud Controller is an error in the Cloud Level and will lead to an outage of the entire cloud. On the other hand, a failure in a Cluster Controller is an error in the Cluster Level and will affect only an individual part of the cloud.

4. *Backup Manager(BM):*

When some Cloud Managers fail and the Cluster Controllers lose the connection to the cloud management interface a self-healing of the cloud middleware is initiated automatically. The self-healing accomplishes a reconnection of the Cluster Controllers. Thus, the cloud resources are still available after a Cloud Manager crashed, since the Cluster Controllers are re migrated by using another failover instance. There by a failure in the Cluster Level can be eliminated. Moreover, the reconnection attempts of multiple Cluster Controllers are load-balanced between all residual Cloud Managers in order to avoid inherited error and to raise the performance .Backup Manager is in picture when, Cloud Manager get failed while communicating to user, as per previous architecture they not consider this failure case, so when cloud manager gets failed all calls from user passed to backup manager to handle all process.

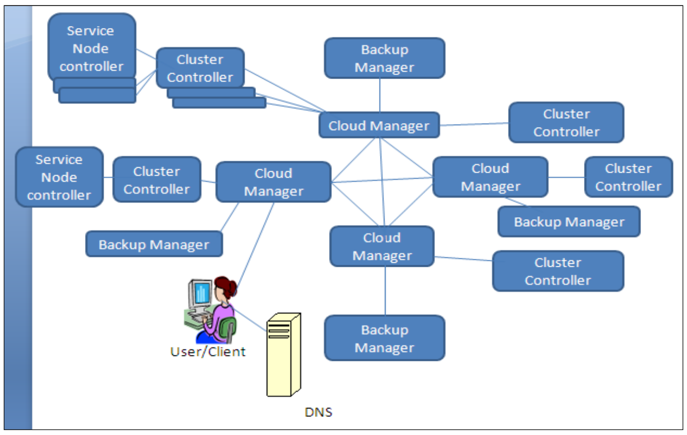

Fig.2. Middleware architecture in Multi-master pattern

IV. Fair Scheduling Algorithm

The Fair scheduling algorithms in that Fairness is most important for scheduling of task. In Fair Scheduling algorithm the tasks are allocated to multiple processors at the same time so that the task with unsatisfied demand get equal shares of time is as follows Tasks are queued for scheduling according to their fair completion times. The fair completion time of a task is calculated by its fair task rates and using a max-min fair sharing algorithm in that the tasks are assigned to processor by the increasing order of fair completion time.

In this algorithm the tasks with a higher order are completed first which means that tasks are taken a higher priority than the others which leads to starvation that increases the completion time of tasks load to the resources so that all task are fairly allocated to processor based on balanced fair rates. The main objective of this algorithm is to reduce the overall make span.

V. Dynamic Load Balancing Algorithm

Dynamic load balancing algorithms used to redistribute available resources among running tasks dynamically such that these tasks could use up the maximum capacity of each resource in a node. Multi computers with dynamic load balancing allocate and reallocate resources allocation at runtime based on a priori task distribution which determine when and whose tasks can be transferred. As a result, dynamic load balancing algorithms can provide a significant improvement in Performance over other algorithms. Load balancing should take place when the scheduler schedules the task to all processors. There are some particular activities which change the load configuration the activities can be categorized as following: Arrival of any new job and queuing of that job to any particular node. The Scheduler schedules the job to particular processor. Then Reschedule the jobs if load is not balanced Allocate the job to processor when its free and then Release the processor after it complete the whole process.

VI. Conclusion

In this paper we describe a failover pattern for cloud middleware in master worker architecture. We show Middleware architecture in multi-master pattern, which prevents an overall cloud failure in case of a failed master node. We describe a communication model for Middleware architecture in Multi-master pattern. Additionally, we introduced a self healing mechanism for this Middleware architecture in multi-master architecture, which leads to an automatic reconnection of worker nodes to another master of the cloud. Furthermore the reconnection has the guarantees the network consistency as well as it is balanced by an individual failover list sorting. We described a better load balancing technique to distribute the user requests according to the load of the master nodes.

We will focus more on high available middleware components in the future to improve the stability of cloud environments. Future work will focus on redundancy In addition; we will show performance measurements and comparisons to other approaches in further publication.

References

1. A. Undheim, A. Chilwan, and P. Heegaard, “Differentiated availability in cloud computing slas,” in Proceedings of the 2011 IEEE/ACM12th International Conference on Grid Computing, ser. GRID ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 129–136. [Online]. Available: http://dx.doi.org/10.1109/Grid.2011.25
2. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soma ,and D. Zagorodnov , “The eucalyptus open-source clou -computingsystem, in Cluster Computing and the Grid, 2009. CCGRID’09. 9th IEEE/ACM International Symposium on. IEEE, 2009, pp. 124–1
3. A. Stanik, M. Hovestadt, and O. Kao, “Hardware as a service (haas):Physical and virtual hardware on demand,” in Proceedings of the 4th IEEE Intl. Conference on Cloud Computing Technology and Science, ser. CloudCom 2012. IEEE publishers, 2012
4. E. Feller, L. Rilling, and C. Morin, “Snooze: A scalable and autonomic virtual machine management framework for private clouds,” in Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), ser. CCGRID ’12.Washington, DC, USA: IEEE Computer Society, 2012, pp. 482–489. [Online]. Available: http://dx.doi.org/10.1109/CCGrid.2012.71
5. Warneke, D., Kao, O.: Exploiting dynamic resource allocation for effcient parallel data processing in the cloud. IEEE Transactions on Parallel and Distributed Systems22 (6), 985{997 (2011)
6. E. Cecchet, G. Candea, and A. Ailamaki, “Middleware-based databasereplication: the gaps between theory and practice,” in Proceedings ofthe 2008 ACM SIGMOD international conference on Management ofdata. ACM, 2008, pp. 739–752.
7. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy,and R. Sears, “Map reduce online,” in Proceedings of the 7th USENIX conference on Networked systems design and implementation, ser.NSDI’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21.
8. U.Karthick Kumar “ A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling”