Efficient Parallel Data Processing for Multi-Master Cloud Architecture with Improved Self-Healing Mechanism

Megha Rode[1], Amruta Amune [2], Sachin Runwal [3]

[1 2 3] *Computer Network , G.H.Raisoni College of Engineering & Management chas, Ahmednagar, University of pune, Maharashtra, India*

---***---

Abstract - *Cloud computing has emerged in a broad range of applications domain for several open source services such as Infrastructure-as-a- Service (IaaS) and it has become well an established building block in IT infrastructure During the process of cloud middleware development Major Cloud computing companies have started to integrate frameworks not only for the high availability services for end-user, but also it unfortunately neglected the availability of middleware components which leads failure. Therefore failures in middleware components which usually lead to a partial failure or even total blackout of the cloud infrastructure.*

In this paper, the new proposed system presents the design and implementation scalable and highly available multi-master cloud architecture for the cloud middleware. In contrast the system suffer Not only from lack of load balancing technique but also it suffer from redundancy and data loss, so In the new proposed system which develop a load balancing algorithm and fair scheduling algorithm .The new system introduce a concept of dynamic load balancing algorithm for the cloud middleware architecture implemented by using multi-master cloud pattern .The new proposed system implement a backup server which will handle all task of failed object so that it has guarantee there will be no data loss as the backup server is doing all task.

Key Words: *Cloud computing, Resource allocation, Self Healing, High availability*

1. INTRODUCTION

Cloud computing has appeared as a new paradigm which promise virtually unlimited resource. Customer uses on-demand services and charged for resources based on pay –as-you –go principal .Today growing numbers of companies have to process vast amounts of data in a cost-efficient manner and these companies are operators on Internet search engines like Google, Microsoft or Yahoo. Running application on the cloud Infrastructure-as-a-Service (IaaS) layer makes faults tolerance as an important issue because failure in this layer result poor quality of service .The proposed system focus on Management Software Failures[1] which is based on management software in Iaas cloud. In this paper the cloud structure is differentiated into three levels they are Virtual Machine Level, Cluster Level, and Cloud Level

1. Virtual Machine Level: In this level have the failure that gives data loss if no suitable replication mechanisms are applied.

2. Cluster Level: In this level failure that gives a node or even whole bunches of node are no longer available

3. Cloud Level: A failure in this level is most severe, which results in an unavailability of the entire cloud.

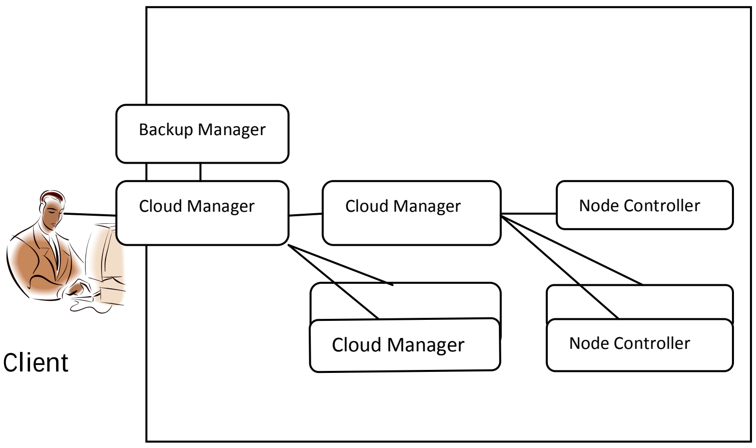


Fig -1: Common Cloud Middleware Stack Master –Worker Architecure

In this paper the proposed system describe the system implement a backup Manager which handles all the task and information of failed object so that there is no data loss as a backup server is doing all task. This paper is organized as follows: section II discusses related work in this field and which will show the differences to other approaches. Section III describes Proposed Work of the system The section IV Dynamic Load Balancing Algorithm section V explain fair scheduling algorithm. Section V I gives the mathematical model section VII gives the performance evaluation and we conclude our results

2. RELATED WORK

The described approaches based on Common Cloud Middleware Stack which is similar to the known open source IaaS platforms like Eucalyptus, Open-Stack, open-nebula[8], snooze, Nephele, CloudDisco .The importance for failover mechanism technique was recognized by Eucalyptus. Eucalyptus[9] consist of following parts, Cloud Manager, Cluster Controller(CC), Node Controller(NC), storage controller(SC) and Walrus on a single machine called as a Controller which perform continuously scheduling and monitoring and it consist of two types of controller the primary Controller and the secondary Controller. It is used for simulation and evaluation function in that 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 makes an error case not only the interface to the cloud would be affected, but also additional components of the installation will be affected.

A.stanik has noted the CloudDisco[4] architecture it consist of Cloud Manager (CM), Cluster Controller (CC), and Node Controller (NC). On the top layer consist of the Cloud Manager represents the interface for the user to request list of available hardware. The cloud Manager receiving request and sent it to cluster controller. Cluster controller propagates all requests to node controller which execute request and provide all resource information

Odej kao describe a Nephele's architecture [5] consist Job Manager (JM), Task manager(TM), Cloud controller and Persistence storage. Job manager which receive client or user request and it is responsible for scheduling and co-ordinates their execution. 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 to the job manager .with the help of cloud controller the job manager allocate and reallocate resources .persistence storage is used to store input data and to receive output data so here also it uses fast switchover mechanism so this approach having some disadvantages with the high availability support faster switchover mechanism are irrelevant and it affect on all components.

Similarly in Snooze [6] architecture the Group Leader(GL) which is interface with the user and it send request which gives a list of available Resources and to setup the connection .The Group Leader receiving a request for all available Resources sent it to all connected Group Manager(GM) which is same as cluster controller(CC) in Eucalyptus . The Group Manager collects the list of components and returns it to the Group Leader which is same as cloud Manager which passes it back to the user. If the Group Leader gets unavailable then the whole service will becomes unavailable, even though the Group Manager and thus the resources are still available The Local Controller(LC) which is same as Node controller(NC) which performing scheduling and monitoring function and it assign all information to the group Manager. In the further course of this text, proposed system restrict ourselves to the naming of Eucalyptus components, as Cloud Manager is equivalent to Group leader in Snooze Besides, the Cluster Controller and Group Manager which is same as in Eucalyptus [3]. In OpenStack the components are called API Server (nova-api), Scheduler (nova-scheduler) and Compute Worker (nova-compute) [1]. Note that the mentioned components have similar or even equal functions within the cloud system and the architectures are alike the pattern

3. PROPOSED WORK

The pattern described in this paper is based Common Cloud Middleware Stack Master –Worker Architecure which is same as that of master-worker architecture [1] . It follows the master -worker pattern as depicted in figure 2 In contrast to master-slave architecture. In this system the master nodes called as Cloud Manager (CM) and they are interconnected with each other by using a full-mesh network topology and they are communicating with each other by using full mesh network topology. the system consist of number of worker node here each worker node called as the Cluster Controller (CC) and they are connected to exactly one of the master nodes which announces all the updates as well as it gives a list of all information known CMs in the system since all CMs are equal a user and a Cloud controller can connect to any Cloud Manager. Middleware architecture in Multi-master pattern can be subdivided in the following four components.

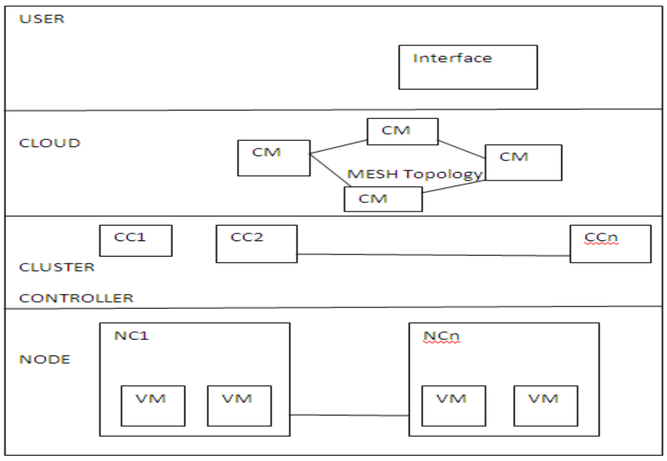


Fig -2: Middleware architecture in Multi-master pattern

1. User/ Client:

A user must start the virtual machine (VM) which runs so it is called as cloud Manager (CM). A user must register itself in cloud to interact with the cloud interface after that user has to interact to cloud manager first.

2. Cloud Manager(CM):

The Master node which is cloud manager or cloud controller. It is the entry point for both the user and administrator. It provides three type services resource service, data service and interface service .This interface between the cloud environment and a user is a significant component which should provide high availability support and scalability mechanisms to achieve high quality of service requirements which is described approach it can be adapted to these systems. The Cloud Manager is the interface for the user request which gives a list of available resource and it will setup the connection to one of components. The Cloud Manager receiving a request for all available resource components sent it all these are connected Cluster Controller. The Cluster Controller collects all the list of components and returns it to the Cloud Manager, which passes it back to the user. If the Cloud Manager gets unavailable then the whole service becomes unavailable .Even though the Cluster Controller and thus the resource components are still available

3. Cluster Controller(CC)& Node Controller(NC):
4.

The Cloud Manager receives requests from the user that is list of available resources and it transfers them to its Cluster Controllers. The Cluster Controllers in turn transfer the entire request to the Node Controllers which execute the request and provide all 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 and whole bunch of cloud will fail so it will be no longer together.

5. Backup Manager(BM):

When some Cloud Managers fail and the Cluster Controllers lose the connection due to network problem the cloud management interface a self-healing of the cloud middleware component is initiated automatically and the self-healing accomplishes a reconnection mechanism 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. Here proposed system will propose the backup server for cloud manager. There will be separate backup server for each cloud manager which takes the backup of that CM's CC and NC. Means it will take the backup of that particular cloud area. Whenever CC or NC fails then at that time backup server will be activated and handles the task of that particular object. If CM fails then also backup server will take charge of that CM .proposed system develop a system which takes the backup after particular time span. It ensures the no loss of data guarantee

4. DYNAMIC LOAD BALANCING ALGORITHM

Dynamic Load balancing is a technique [7] to enhance resources by exploiting dynamic resource allocation for parallel data processing in cloud computing and to increase throughput improvisation . It is used to reduce response time and to increases an appropriate distribution of the application. Load balancing algorithms has different type of policies Information policy, Resource type policy, Triggering policy, Location policy, Selection policy. Load balancing algorithms is divided into two types such as static algorithm and dynamic algorithm. Static load balancing in this algorithm which allocate the resources and task to the workstations which distribute the task parallel. Multicomputer with dynamic load balancing algorithm which allocates or reallocate resources at runtime based on task information present in the system which may determine when and whose tasks can be migrated or transfer they use current or recent load balancing information when making distribution decisions.

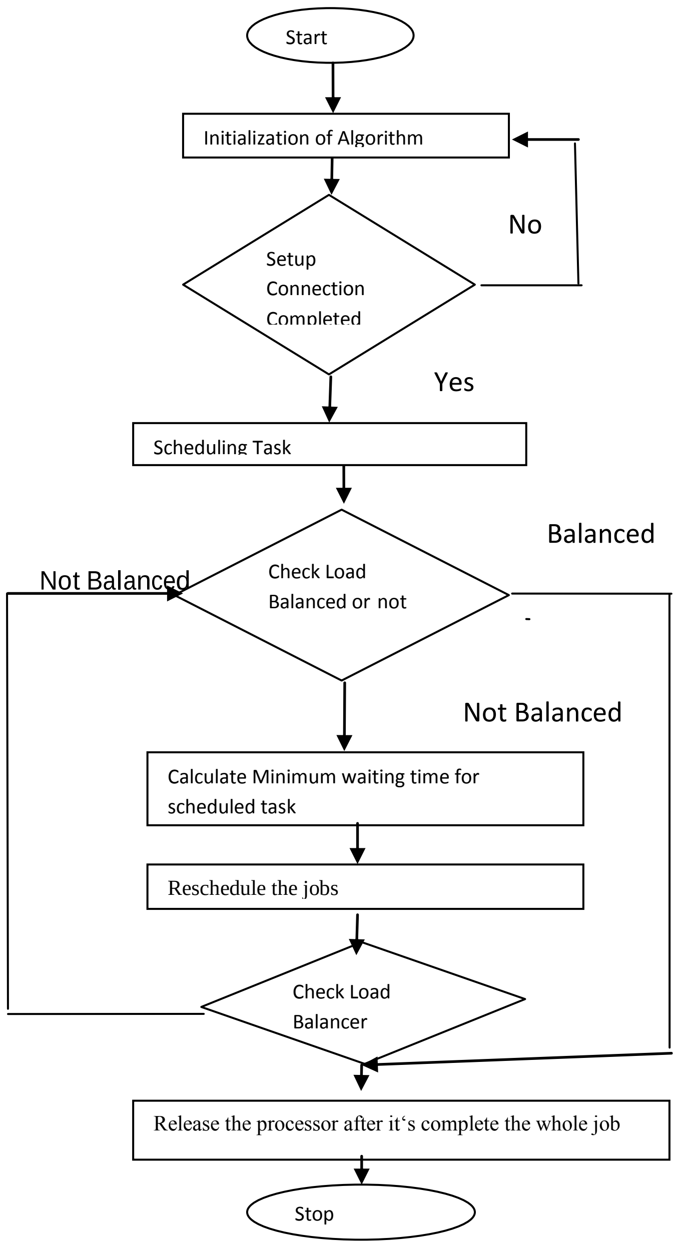

Start
Initialization of Algorithm
Setup Connection Completed
No
Yes
Scheduling Task
Check Load Balanced or not
Balanced
Not Balanced
Not Balanced
Calculate Minimum waiting time for scheduled task
Reschedule the jobs
Check Load Balancer
Release the processor after it's complete the whole job
Stop

Fig.3. Flowchart of Dynamic Load Balancing algorithm

Multicomputer uses dynamic load balancing algorithm with dynamic load balancing it can perform allocation and reallocation resources at runtime based on a priori task information, which may determine when and whose tasks can be migrated. As a result, dynamic load balancing algorithms can provide a significant way for the improvement in Performance over other algorithms. Load balancing should take place when the scheduler schedules the task to all processors. In Dynamic load balancing algorithm the Arrival of any new job is putted in ready queue and passes it to any particular node. Scheduler will schedule the job to particular processor. It will reschedule the jobs if load is not balanced and it Allocate the job to processor when it's free and lastly it will Release the processor after it competition of the whole process.

5. FAIR SCHEDULING ALGORITHM

The Fair scheduling algorithms [7] is dynamic It continuously checks the processes. Fair scheduler does not allocate job directly to the refer processor it first check the balance of all process if there other processor whose load is low then refer processor then allocate job to other processor rather than refer processor .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 each task to obtain load balance. In Fair Scheduling algorithm task is divided into multiple processor .In that the task with unsatisfied demand gets equal shares. Tasks are present in ready queue and it scheduled with fair completion time. The fair completion time of the task is approximate by its fair task rates using a max-min fair sharing algorithm. The tasks are assigned to processor by increasing order of fair completion time. In this algorithm, tasks with a higher priority order get 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 and load balance is not guaranteed. Let N be the number of tasks scheduled as time given Ti, here i=1, 2... N it gives the duration of the task when it is executed on a processor in million instructions per second. Let P be the number of processors and its total computation capacity it is indicated by C and it is defined as Let p is the multiprocessor system and its computation capacity for processor j is defined as Cj The earliest time of the task i started from the processor j is the maximum communication delay and the completion time between the task ith and jth processor. The completion time of the task is zero when there is no task allocated to processor j, otherwise it estimated the remaining time that are already allocated to processor j. The demanded computation rate is given by Xi of a task T_i it will play an important role to estimate. It is estimated by the computation capacity

Input: A set of N be the number of task and P be the number of processor having the computational capacity c_j.

Output: A schedule of N task

1. Create set of Ready Queues.
2. qsize < N/P

```
3. for each queue $q_i$ in Q
4. While there are the tasks present in the Ready Queue do,
5. Assign demanded computation rate of the task Xi
6. k= C/N
7. If $X_i < k$
8. Assign demanded computation rate $X_i$ to ith task as fair rate
9. Else
10. Assign k to ith task as fair rate.
11. Calculate the fair completion time $t_i(x)$
12. End while
13. End Loop
14. Arrange the task in growing order or in increasing based on their $t_i(x)$ and submitted to processor.
15. While the Load of any processor is greater than average load processors do
16. Calculate the mean waiting time [MWT] for each scheduled task
17. If $Z_x^y > 0$
18 Migrated or transfer the tasks are determined by using criteria of processor capacity.
19. Each processor which has least capacity is selected for migration.
20. End If
21. End While
```

6. MATHEMATICAL MODULE

Let System S is set of User request with a cloud manager, cloud controller and node controller.

S={I_{ur}, CM, CC, NC, BM}

Where S= System

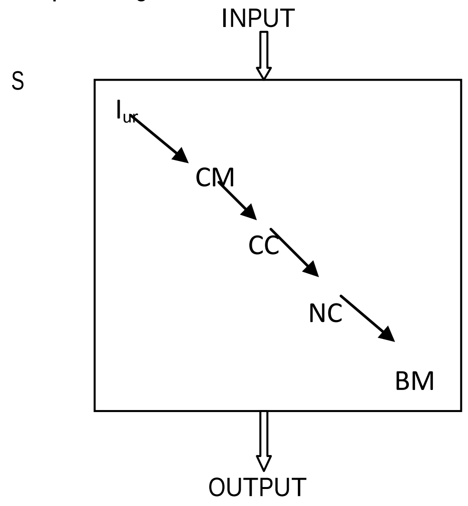
I_{ur}= User Request

CM= Cloud Manager

CC= Cluster controller

NC= Node controller

BM= Backup Manager



1. User Login Phase
Set I
I_0= User login Id
I_1= User Password
I_2= Credential validation
User Interface
Set H
H_0= object creation for specific logged user H_1= Loading plans
User Request
Set R
R_0= create request ticket
R_1= send user request to server
R_2= Validation check at CM
4. Cloud Manager
Set M
M_0= Take user request
M_1= Checking user request
M_2= checking all server status
M_3= allocation server
5. Cloud Controller
Set C
C_0= Create instance for user request
C_1= Allocate/ Deallocate task
6. Node Controller
Set N
N_0= create instance for task
N_1=execute task at VM level
N_2= Send result to CC
7. Backup Manager
Set B
B_0= create instance for cc
B_1= manage task
B_2=backup all work
B_3= send result to CM
Union theory
I= {I_0,I1,I2}
H={H0,H1,I0}
SET I U H : {I0, I1, I2, H0, H1}
SET R= {I0, H0, H1, R0, R1, R2}
SET I U H U R: {I0, I1, I2, H0, H1, R0, R1, R2}
SET M= {I0, H0, H1, R0, R1, M0, M1, M2, M3}
SET I U H U R UM :{ I0,I1,I2,H0,H1,R0,R1,R2,M0,M1,M2,M3}
SET C= {I0, H0, H1, R0, R1, M0, M1, M2, C0, C1}
SET I U H U R U M UC :{I0,I1,I2,H0,H1,R0,R1,R2,M0,M1,M2,M3,C0,C1}
SET N ={I0,H0,H1,R0,R1,M0,M1,C0,C1,N0,N1,N2}
SET I U H U R U M U C U N: {I0,I1,I2,H0,H1,R0,R1,R2,M0,M1,M2,M3,C0,C1,N0,N1,N2}
SET B ={I0,H0,H1,R0,R1,M0,C0,N0,B0,B1,B2,B3}
SET I U H U R U M U C U N U B: {I0,I1,I2,H0,H1,R0,R1,R2,M0,M1,M2,M3,C0,C1,N0,N1,N2, B0,B1.B2.B3}

7. PERFORMANCE EVALUATION

This section describe and analyzes the result obtained from the evaluation of performance analysis, in that comparisons has been made based upon certain metrics such as Mean Waiting time(MWT),Mean Execution time(MET),Throughput(THP). It is expected that the failure pattern will improve as compared to other Existing system .The Result expected from proposed system shown below in Table-I by using comparison table .The Proposed system provide better high availability

Feature	Existing meth-ods	Checkpoint	Integrated	Proposed Meth-ods (predicted)
Checkpoints	No	Yes	Yes	yes
Failover	No	Yes	Yes	yes
Load Balancing	Yes	No	Yes	yes
Multilevel Checkpoint	No	No	Yes	yes
Job Restartation	Yes	No	No	yes
Architecture	2 Tier	2 Tier	3 Tier	n Tier
Resources Utilization	Low	Medium	Maximum	Maximum
Checkpoint over-heads	No	More	Medium	decrease
Checkpoint latency	0	> 0	> 0	> 0
Checkpoint ratio	1	> 0	> 0	> 0

Table -1: Result Evaluation by The Performance Analysis

8. CONCLUSIONS

The proposed system focuses more on high available cloud Middleware components in the future to improve the scalability of cloud environments. Future work will focus on redundancy In addition with performance measurements and comparisons to other approaches in further publication In this paper the proposed system describes the failover pattern for cloud middleware in multi-.master architecture in master-worker pattern, which prevents the failure of cloud in case of the failed master node. The proposed system describe the communication model for Middleware architecture in Multi-master pattern in cloud computing. Additionally systems have introduced a self healing mechanism for this Middleware architecture in multi-master pattern, which leads to an automatic reconnection mechanism of worker nodes to another master of the cloud. Furthermore the reconnection mechanism has the guarantees the network consistency as well as it is balanced by an individual failover list. Here the system described a better load balancing technique to distribute the task from the user requests according to the load of the master nodes. Master of the cloud load balancing technique to distribute the user requests according to the load of the master nodes.

ACKNOWLEDGEMENT

I like to thanks people for helping me for giving importance guidance about project work. I am very thankful to those who help me during my paper publishing as well as project dissertation stage. I am thankful to journal for giving me best opportunity to publish my paper. I am sincerely thankful to my head of department, my project guide and other staff members from department for supporting me.

REFERENCES

[1] Alexander Stanik, Mareike Hoger, and Odej Kao, "Failover Pattern with a Self-Healing mechanism for high availability Cloud Solutions" in 2013 International Conference on Cloud Computing and Big Data.

[2] 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.

[3] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soma ,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. 124–1

[4] 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

[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. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomi 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

[7] U.Karthick Kumar " A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling"

[8] Fault Tolerance 3.8, OpenNebula Project (OpenNebula.org), 2012,

[9] Availableonline http://opennebula.org/documentation:rel3.8:ftguide

[10] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "An availability Model for eucalyptus platform: An analysis of warm-standy replication Mechanism," in Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on, oct. 2012, pp. 1664 – 1669.

[11] M. Treaster, "A survey of fault-tolerance and fault- recovery techniques in parallel systems," *CoRR*, vol. abs/cs/0501002, 2005.

[12] Y. Zhou, P. M. Chen, and K. Li, "Fast cluster failover using Virtual memory-mapped communication," in *Proceedings of the 13th* International conference on Supercomputing, ser. ICS '99. New York, NY, USA: ACM, 1999, pp. 373–382. [Online]. Available: http://doi.acm.org/10.1145/305138.305215

[13] D. Singh, J. Singh, and A. Chhabra, "High availability of clouds:Failover strategies for cloud computing using integrated check pointing Algorithms," in Communication Systems and Network Technologies (CSNT), 2012 International Conference on, may 2012