A Framework for Performance Analysis of Computing Clouds

Mintu M. Ladani[1] Vinit kumar Gupta[2]
[1, 2] M. E. (Computer)
[1, 2] Hasmukh Goswami College of Engineering,Vahelal
[1, 2] Gujarat Technological University

Abstract-- Cloud Computing provides data storage capacity and use of Cloud Computing have increased scalability, availability, security and simplicity. As more use of cloud computing environments increases, it is more difficult to deal with the performance of this environments. We have presented Some virtualization and network related communication issues and finally we have designed and implemented modified load balancing algorithm for performance increase. In market use of cloud many issues occurred like as security, privacy, reliability, legal issues, open standard, compliance. so, we have stated to solve these issues such algorithm to assess increase performance of computing clouds. Secondly, i.e. 'Modified Weighted Active Monitoring Load Balancing Algorithm' on cloud, for the balancer on Cloud Controller to effectively balance load requests between the available Node Controller, in order to achieve better performance parameters such as load on server and current performance on the server. By Existing Algorithm like in RRA (Round Robin Algorithm) load balance sequentially, we have designed this proposed algorithm on cloud and how to balance load randomly and display by existing algorithm and proposed algorithm comparison.

I. INTRODUCTION

Framework, users can assess the overhead of acquiring and Releasing the virtual computing resources, they can Different Configurations and they can evaluate different scheduling algorithms.

II. LITERATURE REVIEW

Modeling techniques is for workloads and storage devices in order to do load balancing of workloads over a set of devices. The workloads were modeled independent of underlying device using the parameters inherent to a workload such as seek distances, read-write ratio, average IO size and outstanding IOs. For device modeling we used statistics such as IO latency with respect to outstanding IOs which is dependent on the storage device. It is Also described a load balancing engine that can do migrations in order to balance overall load on devices in proportion to their capabilities [18].

A. Research problem:

1) How to characterize dynamic workloads for load balancing? Are percentile values for workload parameters good enough in real systems?
2) Can we use static values of constants such as K1 to K4 for workloads running on different devices or do we need online estimation?
3) How to measure and predict interference among various workloads accessing a device?
4) How to suggest storage configuration changes to an administrator based on online workload monitoring. Ultimately the task of workload monitoring, device modeling and load balancing needs to happen in a feedback loop over time to handle churn in today's storage environments.
5) Cloud computing has emerged as a new technology that lets users deploy their applications in an environment with a promise of good scalability, availability, and fault tolerance. As the use of cloud computing environments increases, it becomes crucial to understand the performance of these environments in order to facilitate the decision to adopt this new technology, and to understand and resolve any performance problems that may appear. In this paper, we present Framework, which is a framework for generating and submitting test overloads to computing clouds. By using A systematic literature review is a means of Identifying, Evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomena of interest (Kitchen ham, 2004).[14]

B. Reasons for performing SLR

1) It can identify gaps in current research for further investigation.
2) It can be used as framework for new research activities.
3) To generate a new hypothesis**.**

C. Importance of systematic literature review

1) Literature review has less scientific value.
2) Systematic literature review approach is fair.
3) It summaries all existing information about some phenomenon in a fair and
4) Unbiased manner.

D. Advantages

1) Methodology is well defined and due to this results of literature will have less repeatability.
2) New research activities can be developed.

E. Disadvantages

1) More effort is required

F. Features of SLR

1) Systematic review starts by defining review protocol.
2) Based on search strategy systematic review is defined.
3) Search strategy is documented.
4) Quality criteria for paper selection of primary studies

are considered.

5) Systematic review can be used as a prerequisite for quantitative Meta-analysis.

G. Conclusion of Literature Review:

By Conclude this all above paper how to characterize workload maintains in virtualization After Concluding all above paper how to characterize how to load-balance in Cloud Computing. Some virtualization security risks and problems create in dynamic workload in load balancing. so, I will do maintain load balancing in cloud computing.

III. PROBLEM STATEMENT

In earlier load balancing algorithms or in existing methods only how to balance load is stated but in those schemes only some factors Load on the server, Current performance of server are considered like, i.e. 'Modified Weighted Active Monitoring Load Balancing Algorithm' on cloud, for balancer to effectively load balance requests between the available Node controller assigning a load, in order to achieve better performance.

A. Proposed Algorithm

Modified 'Weighted Active Monitoring Load Algorithm' is designed Balancer assigning a load to each Node Controller. Existing in Round Robin Algorithm load is balanced sequentially Hence optimizing the given performance parameters. After with Load on server and with current Performance of server is necessary to find.

Algorithm: For better performance of our algorithms two parameters are taken in to consideration.

1. Load on the server.
2. Current performance of server.

1) Load on the server:

We are more interested in free resources available on node. The node having more free resources will able to handle more requests easily without degrading its performance.

2) Load on the server:

We are more interested in free resources available on node. The node having more free resources will able to handle more requests easily without degrading its performance.

Current performance of Sever: A request is send to the node at regular interval and in response, the performance parameter is measured. It may be the case that the response time of node may change every time depending on the client usage of its resources. So, the above two parameters are used to built a new queue for future allocation. This information of the entire node is calculated by a mathematical function to count z-parameter value for each node.

The pseudo code of the algorithm is as under

```
Pseudo Code
Step 1: [Calculate Load Factor x]
x ← (Total _Resources
– Used _Resources)
 where x is free memory in terms of percentage.
Step 2: [Calculate Performance Factor y]:
y1 ← average (current _response _time)
y ← y1 - (previously calculated y1)
y ← y/(previous y1)*100 //counting y in terms of previously counted y1.
Step 3: [finding z]
z ← x - y; If (z < 0)
z = 0;
Step 4: [Find minimum of all z except the nodes with z value 0]
Min _z= min (all z's)
Step 5: [Find min _factor and divide all z by that factor] Min _factor ← min _z
Z ← z / min _factor
Step 6: [Generate Dynamic Queue on base of z]
```

In the above algorithm x is considered as a free load on server, y for the performance on the server and y1 is the current response time.

Explanation

Step-1:

The value of x is calculated by considering the total available resources and allocated resources on the server. The available resources would be calculated using the equation

x = (Total _Resources – Used _Resources).

Once x value is calculated for all nodes servers we get the available free load on the servers.

Step 2:

Here the performance factor calculates the increase or decrease in performance on the server and the calculated value is stored y. Now for calculating y a request is send to all the nodes at regular interval of time and the response time(total of both request time + response time) is calculated. So every hour sever would have various values of y and averaging all the value of y1 would be calculated to generate a queue. Now, the previously calculated y1 will be deducted from current values y1 which was currently used to calculate the performance (i.e. Response time increases or decreases). The same way the increased or decreased performance is calculated and the value of y is calculated as the percentage of previously counted y1. That is y/(previous y1)*100.

Step 3:

Counting z = x - y; Here Y value is subtracted from x value to count the z value Here we are interested in the node with the lowest response time hence we subtract the y value from x. i.e. nodes having more response time will contain less z value and they will get less number of requests to handle.

Now suppose in the worst case node have very less memory available and very large response time than z = x – y may get negative value so we need to remove that node from queue (think that it is temporary unavailable) and it will not consider in any step of the algorithm and in this iteration of algorithm it will not get any request to handle.

If any node is temporarily unavailable the response time will be infinite (or very large) of that specific node. So the y value also becomes too much large for that node which will leads to minus z value.

And in step 4 of algorithm that node will discarded for future process in this irritation of algorithm. So with this step we can also detect the unavailable node (with value of y as infinite or to large).

Step 4:

Once the z is calculated then the minimum of all z which we calculated are stored in min _z.
Here we will not consider node with 0 z value so, this node will be remove from any further process and for this iteration of the algorithm.
In the next iteration of the algorithm the z value of node will count again so it will get the next chance to join cloud environment if it is ready for it i.e. available.
Step 5:
Here minimum factor is calculated and divided by all the factors of z.
z values for node 1 to 5 are given below:

Node	z-value
N1	22.30
N2	12.23
N3	43.00
N4	14.36
N5	28.29

Now N2 has a list z value and so every nodes z value are divided by N2's z value and stores a float or cell value of it to make it easy and less complex. So now the z values are 2, 1, 4, 1, and 2.
Node with z value 0 will get 0 as z / min_fact so they can't get any request to handle.
Step 6:
From the above value N1 has the capacity to handle two requests, node N2 can handle only one while N3 will handle 4 to keep load balanced on each node. So the temporary queue will look be prepared as follows:

N3	N3	N3	N3	N1	N1	N5	N5	N4	N2

Fig. (1): Dynamic Queue

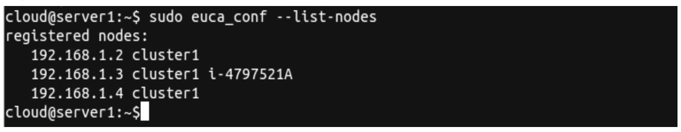
So, once the temporary queue and permanent queue will be changed and accordingly, first 4 requests will go to node 3 than 2 will go to n1 and so on until the end of queue. Once the queue is over it will assign next 4 to N3 and so on.
In this way the whole algorithm will work and would help in increasing the performance and improve load balancing.

IV. IMPLEMENTATION AND RESULTS

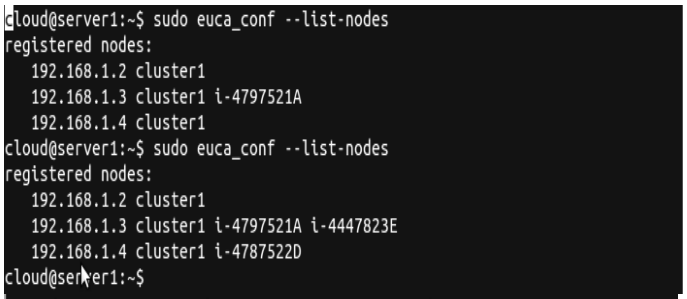
A. *Proposed Architecture*

1) Cloud Controller: Works for authenticate for request and response handling. Also provides a web interface to users for managing certain aspects of the cloud infrastructure.
2) Node Controller: it stores data after load balancing or before. Also it interact with the OS and hypervisor running on the node on one side and the Cloud controller on the other side.
3) Balancer: it distributed (balance) load on node controller.
4) VM: it generates as service as output.

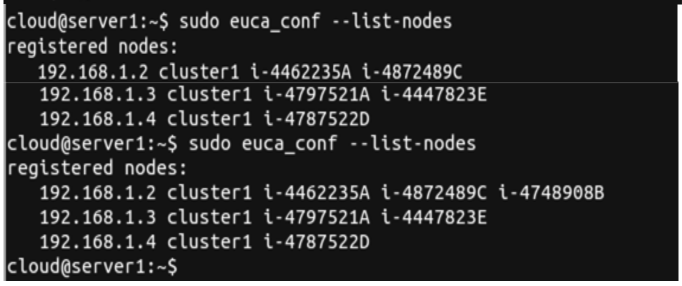
B. *Results*



```
cloud@server1:~$ sudo euca_conf --list-nodes
registered nodes:
   192.168.1.2 cluster1
   192.168.1.3 cluster1 i-4797521A
   192.168.1.4 cluster1
cloud@server1:~$
```



```
cloud@server1:~$ sudo euca_conf --list-nodes
registered nodes:
   192.168.1.2 cluster1
   192.168.1.3 cluster1 i-4797521A
   192.168.1.4 cluster1
cloud@server1:~$ sudo euca_conf --list-nodes
registered nodes:
   192.168.1.2 cluster1
   192.168.1.3 cluster1 i-4797521A i-4447823E
   192.168.1.4 cluster1 i-4787522D
cloud@server1:~$
```



```
cloud@server1:~$ sudo euca_conf --list-nodes
registered nodes:
   192.168.1.2 cluster1 i-4462235A i-4872489C
   192.168.1.3 cluster1 i-4797521A i-4447823E
   192.168.1.4 cluster1 i-4787522D
cloud@server1:~$ sudo euca_conf --list-nodes
registered nodes:
   192.168.1.2 cluster1 i-4462235A i-4872489C i-4748908B
   192.168.1.3 cluster1 i-4797521A i-4447823E
   192.168.1.4 cluster1 i-4787522D
cloud@server1:~$
```

Parameter	Round Robin	Equally Spread Current Execution	Throttled	Proposed algorithm
Number Of Request Per 3 Hour	35	35	35	35
Response Time(s)	142.25s	142.16s	124.62s	122.5s
Data Center Processing Time	35.78s	35.69s	18.26s	18.20s

Table (1): Comparison of Existing and Proposed Algorithm

V. CONCLUSION AND FUTURE WORK

A. *Conclusion*

We have presented some virtualization and network related communication issues more use of cloud computing environment. In market use of cloud many issues occurred like as security, privacy, relibility, legal issues, open standard, compliance. so, we have stated to solve these issues such algorithm to assess increase performance of computing clouds. And finally we have designed and implemented modified load balancing algorithm for performance increase. i.e. 'Modified Weighted Active Monitoring Load Balancing Algorithm' on cloud, for the balancer on Cloud Controller to effectively balance load requests between the available Node Controller, in order to achieve better performance parameters such as load on server and current performance on the server. By Existing Algorithm like in RRA (Round Robin Algorithm) load balance sequentially ,we have designed this Proposed algorithm on cloud and how to balance load randomly and display by existing algorithm and proposed algorithm comparison.

B. *Future Work*

we have presented modifying load balancing algorithm to achieve better performance parameters such as load on server and current performance on the server.By Existing Algorithm like in RRA(Round Robbion Algorithm) load balance sequentially ,we have designed this Proposed algorithm on cloud and how to balance load randomly and display by existing algorithm and proposed algorithm comparison. I have implemented algorithm on private cloud but in future algorithm will be implemented in real world

REFERENCES

[1] The Economist, "A Special Report on Corporate IT," October2008,"http://www.economist.com/specialReports/showsurvey.cfm? Issue=20081025".
[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," CoRR, vol. abs/0808.3558, 2008.
[3] T. Killalea, "Meet the virts," Queue, vol. 6, no. 1, pp. 14–18, 2008.
[4] A. Iosup and D. Epema, "GrenchMark: A Framework for Analyzing, Testing, and Comparing Grids," in Proc. of the 6th IEEE/ACM Intl.Symposium on Cluster Computing and the Grid (CCGrid06). IEEE Computer Society, 2006, pp. 313–320, an extended version can be found as Technical Report TU Delft/PDS/2005-002, ISBN 1387-2109.
[5] Amazon, "http://aws.amazon.com/ec2/," 2008.
[6] Open Grid Forum, "Job Submission Description Language (JSDL) Specification, Version 1.0," 2005.
[7] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong, "Theory and practice in parallel job scheduling," in 3rd Workshop on Job Scheduling Strategies for Parallel Processing, vol. LNCS 1291, 1997, pp. 1–34.
[8] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling Using System- Generated Predictions Rather Than User Runtime Estimates," IEEE Trans. Parallel & Distributed Syst., vol. 18, no. 6, pp. 789–803, Jun 2007.
[9] S. L. Garfinkel, "An Evaluation of Amazons Grid Computing Services: EC2, S3 and SQS," Center for Research on Computation and Society School for Engineering and Applied Sciences, Harvard University, Tech. Rep. TR-08-07, 2007.
[10] S. Garfinkel, "Commodity Grid Computing with Amazon's S3 and EC2," ;login, vol. 32, no. 1, pp. 7–13, 2007.
[11] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientific Computing," ;login, vol. 33, no. 5, pp. 18–23, 2008.
[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "Eucalyptus : A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems," Department of Computer Science, University of California, Santa Barbara, Tech. Rep. 2008-10, 2008.
[13] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?" in DADC '08: Proceedings of the 2008 international workshop on Data-aware distributed computing. New York, NY, USA: ACM, 2008,.
[14] Karthik Paladugu Sumanth Mukk "Systematic Literature Review and Survey on High Performance Computing in Cloud" Systematic literature review (SLR) September 2012
[15]By Cloud Computing Ajith Singh."An Approach on Semi-Distributed Load Balancing Algorithm for Cloud Computing System" N Dept. of Computer Science Karpagam University M. Hemalatha Dept. of Computer Science Karpagam University,oct-2012
[16]By Mariana Carroll1,, Paula Kotzé1,3,"Secure virtualization Benefits, Risks and Controls " Alta van der Merwe Secure Virtualization
[17]By Rajkumar Buyya1,2, Chee Shin Yeo1, and Srikumar Venugopal" Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities "
[18]By Ajay Gulati, Chethan Kumar, Irfan Ahmad "Modeling Workloads and Devices for IO Load Balancing in Virtualized Environments"