An Efficient Cloud Computing Scaling on Internet using Ant based Techniques

Bhavana Singh
M.Tech scholar
Department of Computer Science and Engineering, T.I.T (Excellence) Bhopal

Sandeep Rai
Assistant Professor
Department of Computer Science and Engineering, T.I.T (Excellence) Bhopal

Rajesh Boghey
Professor & HOD
Department of Computer Science and Engineering, T.I.T (Excellence) Bhopal

ABSTRACT

In this paper a new and efficient Hybrid Technique for the Automatic Scaling of Internet Things in Cloud Computing is proposed using Ant based techniques. The Proposed methodology applied here is used for the load balancing over cloud computing and hence scales over cloud for internet on Things. The methodology performs better in terms of Scalability and Decision Time and number of placements. The Various Experimental Results Performed on Cloud Environment proofs to be more efficient in terms of Decision Time and Response Time in Comparison. . The Proposed Methodology implemented here is based on Ant based Clustering Techniques, where Scaling of Internets is done by grouping the ants moving from one source Node to Another.

Keywords

Cloud Computing, Internet on Things, Data Centers, Virtual Machines, Ant based techniques, service level agreement.

1. INTRODUCTION

Cloud computing is a model for enabling on-demand access to a shared pool of computing resources. With virtually limitless on-demand resources, cloud environments enable the hosted Internet applications to quickly cope with the spikes in workload. However, the overhead caused by the dynamic resource provisioning exposes the Internet applications to periods of under-provisioning and performance degradation. Moreover, the performance interference, due to the consolidation in cloud environments, complicates the performance management of Internet applications. Internet applications' usage is crucial part of everybody's cyber life. Whether provided as profit (e.g., online retailer) or non-profit services (e.g., Wikipedia), Internet applications are likely to be delivered with a high quality of service (QoS). The workload of an Internet application varies according to the time of the day and rises sharply on occasions. Internet-connected real-time applications which request the processing of real-time tasks in remote application servers running in the public cloud with metric based auto-scaling solutions. Our target applications include remote patient monitoring systems [9, 10], real-time traffic control systems, drone navigate cloud platforms, and Internet of Things (IoT) devices requiring transmission of deadline-sensitive data and periodic task executions. We assume that the public cloud infrastructure provides proper security and data backup solutions with a Service Level Agreement (SLA) and mechanisms to fairly share its virtual resources among all its running VMs. The last few years have witnessed the emergence of cloud computing as a rapid, limitlessly scalable, and cost-efficient alternative in contrast to the in-house (i.e., on-premise) data centers. The IaaS model delegates more control to the customers over the provisioned resources. Hosting Internet applications in the IaaS environment is an efficient way to start a new and a sustainable business that expands the IT infrastructure gradually with the business growth.

A simple architecture of cloud computing consist the data centers servers for the web application as well as a switch whose function is balancing the loud and distribute load to set of application server also having set of backend storage server. Fig. 1 shows the typical architecture of data center servers for Internet applications. It consists of a load balancing switch, a set of application servers, and a set of backend storage servers. The front end switch is typically a Layer 7 switch [5] which parses application level information in Web requests and forwards them to the servers with the corresponding applications running.

As each server machine can host multiple application so it is important that application should be stateless because every application store their state information in backend storage servers, so that is why they can be replicated safely but it may cause storage servers becomes overloaded but the focus of this work is on application tire presenting a architecture is representative of a large set of internet services hosted in the cloud computing environment even through providing infinite capacity on demand.

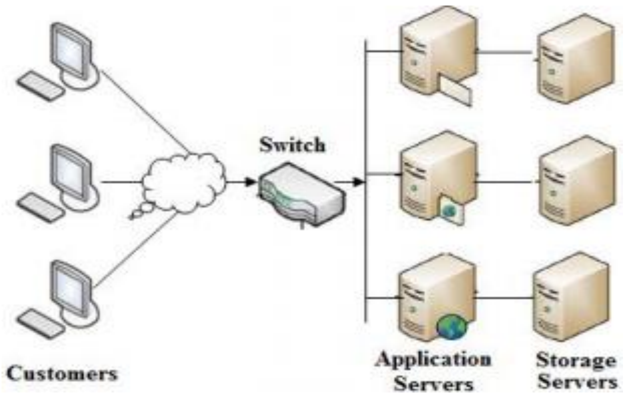

Fig 1: Architecture of Internet application in cloud computing [3]

Moreover, on-demand provisioning provides a cost-efficient way for already running businesses to cope with unpredictable spikes in the workload. Nevertheless, an efficient scalability for an Internet application in the cloud requires a broad knowledge of several areas, such as scalable architectures, scalability components in the cloud, and the parameters of scalability components. With the increase in numbers and size of on-line communities there has been an increasing effort to exploit cross-functionalities across these communities. However, application service providers have encountered problems due to the unpredictable demand for their application(s), especially when external events can lead to unprecedented traffic levels to and from their application [13]. This dynamic nature of demand and traffic drives the need for a massively scalable solution to enable the availability (and reliability) of Internet-based applications.

Multi-tier architecture has been used for decades to provide scalability for Internet applications. The emergence of the IaaS model enriches the multi-tier architecture with components (i.e., services) that help scaling resources dynamically (based on the actual demand) to maintain the Internet application performance. To automate the performance management, IaaS's customers are provided with tools that enable provisioning and terminating resources remotely. Many approaches are developed to improve Internet applications performance in the cloud [3-2]. Nevertheless, less attention is given for evaluating the scalability implementation in the current running production environments. Thus, we evaluate the current implementation of the scalability in the large-scale production environments, namely, Amazon EC2. Modeling an application behavior is crucial for maintaining the Internet application performance and avoiding resource bottlenecks. Nevertheless, modeling an Internet application is complex due to the nature of their architecture. For instance, each tier in the Internet application runs different software which itself has a different behavior. Moreover, the dependency between the Internet application tiers propagates the impact of resource bottlenecks from one tier to the others [7] [12]. Moreover, work [14] has found that the most efficient technique to conserve energy is to revolve the whole server off. The application placement difficulty is fundamental to accomplishing a high demand approval ratio without wasting energy.

2. LITERATURE SURVEY

Here author has to present [1] a system that provides automatic scaling for Internet applications in the cloud environment. Here they encapsulate each application instance within a virtual machine (VM) and make use of virtualization technology to provide fault separation. We model it as the Class Constrained Bin Packing (CCBP) problem where each server is a bin and each class characterizes an application. The class constraint replicates the convenient limit on the number of applications a server can run concurrently. Here they expand a proficient semi-online color set algorithm that accomplishes good demand agreement ratio and saves energy by reducing the number of servers used when the load is low. Experiment analysis shows that our system can develop the throughput by 180% over an open source accomplishment of Amazon EC2 and restore the normal QoS five times as fast during flash crowds. Experiments also show that their system can restore the normal QoS five times as fast when a flash crowd happens and this demonstrate that their algorithm is extremely efficient and scalable which can accomplish high demand agreement ratio, low placement change frequency, short request response time and good energy saving.

In this paper author has to present [5] new concept based on the Cloud Operating System (COS), a middleware framework to support autonomous workload flexibility and scalability based on application-level migration as a reconfiguration plan. While other scalable structures (e.g., MapReduce or Google App Engine) force application developers to write programs following specific APIs, COS make available scalability in a general-purpose programming structure based on an actor-oriented programming language. When they are all executing VMs are highly operated, COS extents a workload up by migrating mobile actors over to recently dynamically formed VMs. When VM utilization drops, COS scales the workload down by combining actors and finishing unoccupied VMs. Application-level migration is beneficial evaluated to VM migration particularly in hybrid clouds in which migration costs over the Internet are critical to scale out the workloads. Here they show the general purpose programming approach using a tightly-coupled computation.

As they compare the performance of autonomous (i.e., COS-driven) vs. ideal reconfiguration in addition to the impact of granularity of reconfiguration, i.e., VM migration vs. application-level migration. Their results [5] demonstrate assure for potential fully automated cloud computing resource management systems that proficiently allow truly expandable and scalable general-purpose workloads.

In this paper author [8] has addressing the emergence of virtualization technologies and cloud computing, the admission control, the service differentiation, the service degradation, and sometimes combinations of them have been practical techniques for maintaining Internet applications performance during overloading periods of time. Admission control technique maintains the availability and performance of a server by controlling the number of admitted requests to the server. At specific threshold of utilization, any additional requests are dropped to keep the server utilization within the threshold. Typically, computing systems have different performance thresholds that are importantly measured by the system administrator for maintaining the system performance. In contrary to admission control, service differentiation technique differentiates customers into classes and provides different QoS for each class. For example, at overloading time, an online retailer can give more priority (i.e., dedicate more resources or decline other classes) for buying request over browsing requests. Actually, admission control is a special case of the service differentiation technique [8].

Desell et al. [11] investigates component-level malleability using the SALSA programming language. They both support dynamic granularity change by splitting and merging of processes and application components respectively; however, they require users to implement how to split and merge. The middleware presented in this paper does not need cooperation from application programmers when the number of actors is larger than the number of VMs, but when the number of actors is smaller than the number of VMs, an actor has to split or make a clone to properly balance the load. We plan to keep working to improve the performance of COS-driven workload reconfigurations. Here they also plan to experiment with more heterogeneous and practical workloads. They also try to develop a model to understand high-level policies such as time-constrained, budget-constrained, and energy-constrained with QoS parameters (such as throughput or dead-lines) [4] to constrain reconfigurations in hybrid clouds.

Despite the emergence of the cloud computing model, which assures a virtually limitless scalability, the traditional techniques are still needed for the following reasons: First, resources in the cloud are limited per an account (e.g., 20 EC2 instances limit associated with each Amazon AWS account). On the other hand, the user can ask for raising this limit. Second, if the system scalability is not limited by the resources, it will be limited by a definite budget [6]. Finally, raising the resources' limit does not defend the Internet application from the denial of service attack (DoS). It also representations the company budget to an unexpected enhance. To save from harm the Internet application from attacks that target the service availability, performance, or budget, Go Grid load-balancer for example limits the number of accepted connections per a client.

3. PROPOSED METHODOLOGY

The model is fully distributed, i.e. every node behaves separately as well as each ant or agent, and this denotes that every node or ant is autonomous. Figure represents the table attached to each node or ant. In the model, each node contains a table that includes information about other nodes in the system. At the initial state, the table entries are Null. In each ant tour, the ant will carry the updated information about all nodes that the ant has been passed throughout. Upon arrival of the ant at every node, the following events will be done:

Assume a network is setup and a number of packets send from source to destination and the value of pheromone deposited at each nodes and shortest path is selected using Max-Min, Rank based and Fuzzy System.

1. Suppose 'N' of packets to be send from Source 'S' to destination 'D'.
2. Initialize all the pheromone table of the node to zero.
3. When first packet is send from one node to another pheromone value is updated accordingly at that node and update all the tables of the network.
4. Proposed methodology uses the limitation s of the existing ant based techniques; hence at each node of the network the possibility of various paths from that node to next node is computed and updated.
5. After first iteration the value of the value of pheromone is calculated at each node of the network.
6. More value of pheromone attracts more ants; hence the next packet is send to that particular node where pheromone value is maximum.

```
If N → pkts send from one node to other nodes
Compute next node based on Max-min ();
Compute next node based on Rank();
Compute next node based on Fuzzy();
Repeat till 'N' packets send from source to destination
For each N → pkt to traverse from nod1-> nod2
If Vpher → nod2 == Vpher → nod3v && If Rnod2 > Rnod3
&& Rnod4<Rnod3
Stores the path from nod2 → nod3
End
End
Repeat for each N → pkt from 'S' to 'D'
Call Max-Min();
Call Rank();
Call Fuzzy();
Traverse the nod1 → nod2 based on stored path.
End
End
End
```

Here in the proposed methodology the shortest route from source to destination will depends on the stored routes from Max-Min, Rule based and Fuzzy based System.

At each step of the node in the network instead of checking of only two nodes the next possible path from 3 ant based techniques is checked and if the chances of traversing fails to apply then the next traversing path is stored, which is then used in the proposed methodology.

The proposed methodology uses the wrong traversed routes from one to another where the decision is based on only values from one node to another.

The proposed algorithm implemented here for the shortest path between source to destination with various notations are given as:

N	No. of packets to be send from source to destination
P	Total pheromone to be deposited during the transmission of packets
E	Total Evaporation rate Rate
T	Total time for the evaporation
Tmax-min	Total value of pheromone to be deposited or evaporated during Max-Min algorithm in the pheromone Table.
Trank	Total value of pheromone to be deposited or evaporated during Rank based algorithm in the pheromone Table.
Tfuzzy	Total value of pheromone to be deposited or evaporated during Fuzzy algorithm in the pheromone Table.
P	Packet to be send from source to destination
Sht	Shortest Path between Source to Destination
Tp	Total Pheromone deposited during the shortest path
Tsp	Total time to send the packet
N1 → N2	Node 1 to Node 2
Initial Node	Starting or Source Node

Table 1 Various Notations Used in the algorithm

The methodology starts with the initialization of the existing methodologies such as Max-Min algorithm and Rank based algorithm and Fuzzy algorithm. Each of the technique implemented provides shortest path between sources to destination and hence on the basis of the existing methodologies proposed algorithm is implemented.

```
Input: No. of Packets to be Send (N)
       Value of Pheromone Deposition (P)
       Value of Pheromone Evaporated (E)
       Time of Evaporation (T)
       Pheromone Table of Max-Min, Rank based and Fuzzy
Logic (Tmax-min, Trank,Tfuzzy)
Output: Shortest Path between Source to Destination (Sht)
        Total Pheromone Deposited in Shortest Path (Tp)
        Time to send Packets (Tsp)
```

```
Algorithm
1.  Initialize all the parameters.
2.  If 'N' number of packets send from source to destination.
3.  For every Packet 'P' send from Node 'N1' → 'N2'.
4.  Count initial Tsp
5.  Deposit the value of pheromone value at Node 'N1' & 'N2'.
6.  Tp =Tp + P;
7.  If there are 'Npath' from one node 'N1' → 'N2'
8.  Check the pheromone table of Tmax-min and Trank and Tfuzzy.
9.  If for every algorithm shortest path from one node to next node is same
10. Choose that node as the next node.
11. Sht = Initail Node + Sht;
12. Tp= Tp +p;
13. Else
14. Check the two best algorithms having shortest path from one node to next as same.
15. Choose that node as the next node.
16. Sht =Previous node + Sht;
17. Tp=Tp+p;
18. End
19. Count Time after sending 'N' packets Tsp1;
20. TotalTime= Tsp1-Tsp;
```

4. RESULT ANALYSIS

	Response Time (ms)	
Time	**Existing Work**	**Proposed Work**
0	0	0
10	1000	670
20	500	340
30	450	210
40	500	337
50	5	2
60	6	1

Table 2: Analysis of Response Time

	No. of APM	
Demand Ratio	**Existing Work**	**Proposed Work**
0	0	0
0.2	280	340
0.4	580	670
0.6	800	950
0.8	900	1030
1	1000	1200

Table 3: Analysis of No. of APM as Demand Increase

	Decision Time	
Demand Ratio	**Existing Work**	**Proposed Work**
0	0	0
0.2	0.18	0.1
0.4	0.2	0.12
0.6	0.22	0.16
0.8	0.24	0.18
1	0.26	0.2

Table 4: Analysis of Decision Time

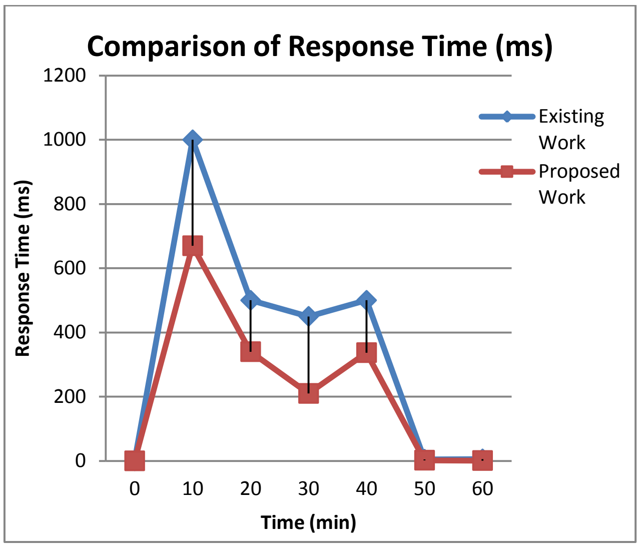

Figure 2. Comparison of Response Time

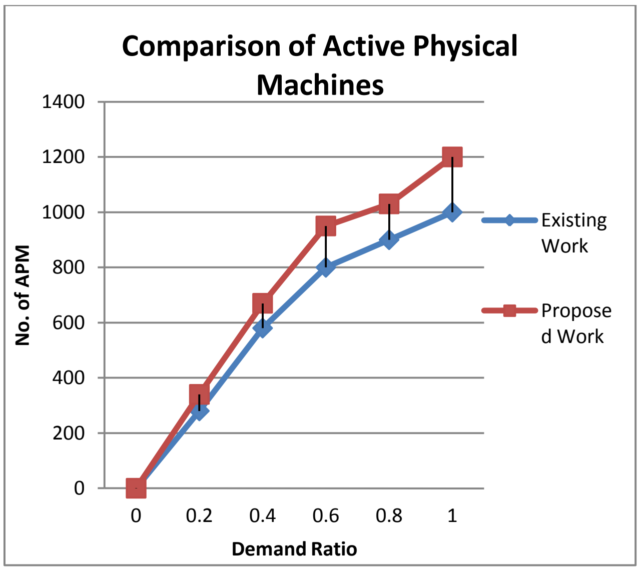

Figure 3. Comparison of No. of APM

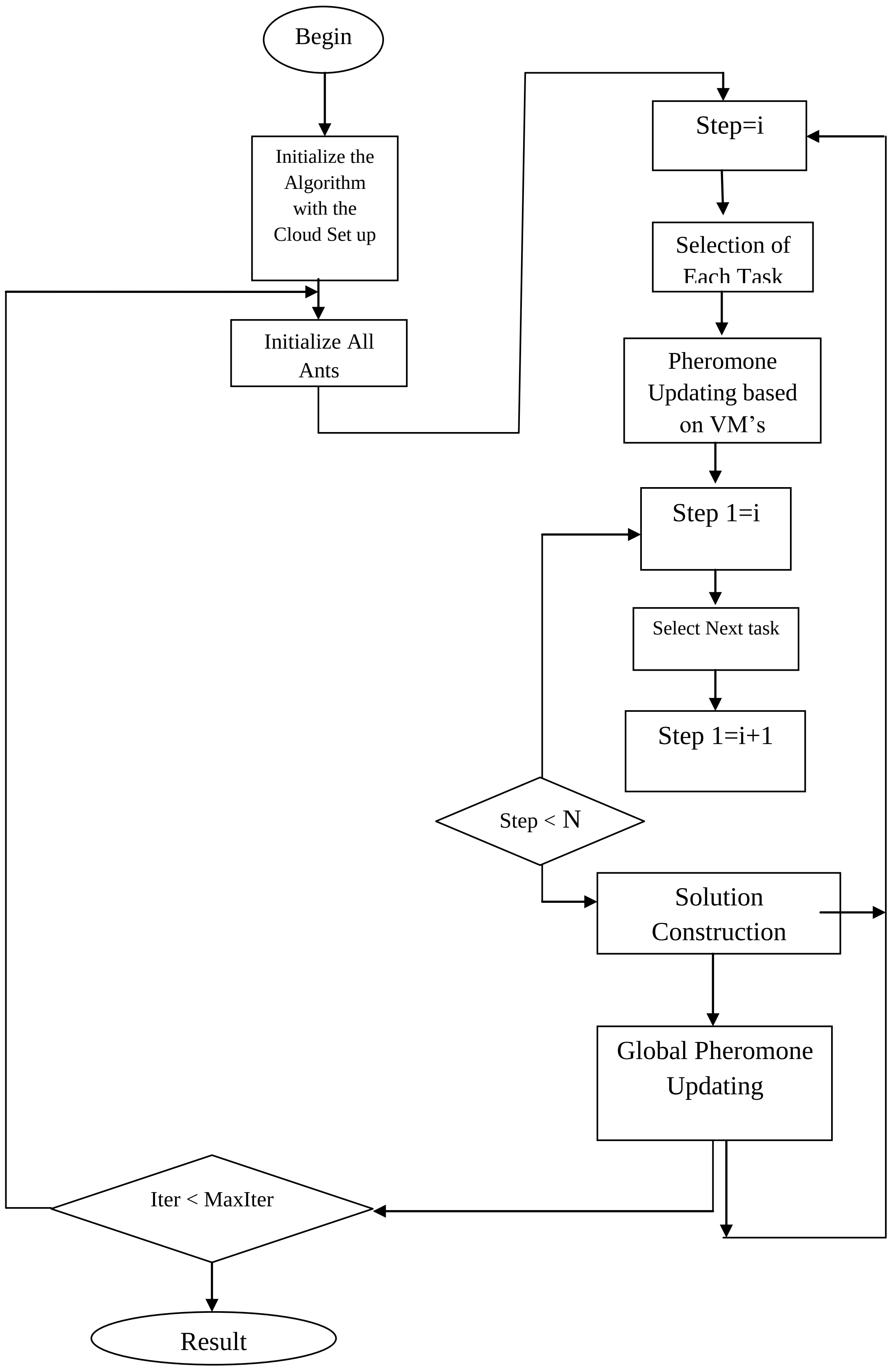

Figure: Flow Diagram of Proposed Algorithm

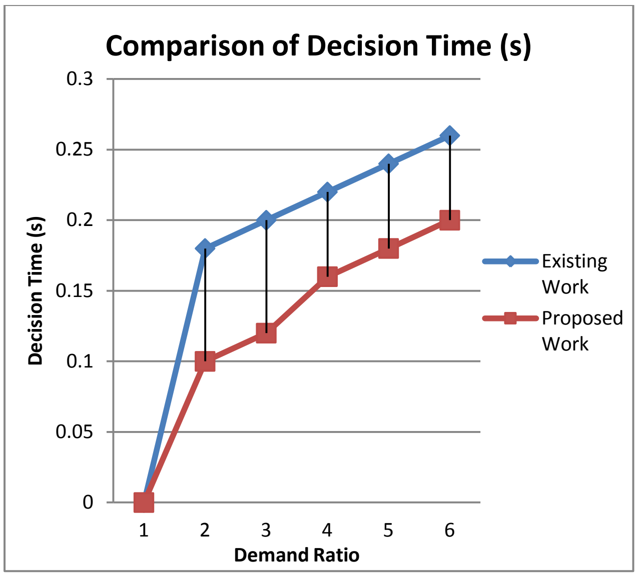

Figure 4. Comparison of Decision Time

5. CONCLUSION

Cloud Computing has been widely adopted by the industry, though there are many existing issues like Load Balancing, Server Consolidation, Virtual Machine Migration, Energy Management, etc. Central to these issues is to find shortest route from source to destination by using different methodology, which is required to distributing the excess dynamic local workload evenly to all the nodes in the whole Cloud to achieve a high user satisfaction and resource utilization ratio. The Proposed Methodology implemented here provides better Request and Response time as compared to the existing methodology. It also provides efficient Decision Time and Stability Probability.

6. REFERENCES

[1] Zhen Xiao, , Qi Chen, and Haipeng Luo, "Automatic Scaling of Internet Applications for Cloud Computing Services" IEEE Transactions On Computers, Vol. 63, No. 5, May 2014.

[2] Jack Li, Qingyang Wang, Deepal Jayasinghe, Simon Malkowski, Pengcheng Xiong, Calton Pu, Yasuhiko Kanemasa, and Motoyuki Kawaba. Profit-Based Experimental Analysis of IaaS Cloud Performance: Impact of Software Resource Allocation. In 2012 IEEE Ninth International Conference on Services Computing, pages 344-351. IEEE, jun 2012.

[3] Rui Han, Li Guo, Moustafa Ghanem, and Yike Guo. Lightweight Resource Scaling for Cloud Applications. In CCGRID, pages 644-651.IEEE, 2012.

[4] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds," Future Generation Comp. Syst., pp. 861-870, 2012.

[5] Shigeru Imai, Thomas Chestna, Carlos A. Varela, "Elastic Scalable Cloud Computing Using Application-Level Migration", IEEE/ACM Fifth International Conference on Utility and Cloud Computing 2012.

[6] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. SLA- based admission control for a Software-as-a-Service provider in Cloud computing environments. Journal of Computer and System Sciences, 78[5]:1280-1299, sep 2012.

[7] Waheed Iqbal, Matthew N. Dailey, and David Carrera. SLA-Driven Dynamic Resource Management for Multi-tier Web Applications in a Cloud. In 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 832-837. IEEE, May 2010.

[8] Jordi Guitart, Jordi Torres, and Eduard Ayguad e. "A survey on performance management for internet applications. Concurrency and Computation: Practice and Experience, 22[1]:68-106, 2010.

[9] D. Niyato, E. Hossain, and S. Camorlinga, "Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization," IEEE J.Sel. A. Communication, vol. 27, no. 4, pp. 412–423, May 2009.

[10] A. Whitchurch, J. Abraham and V. Varadan, "Design and development of a wireless remote point-of-care patient monitoring system," IEEE Region 5 Technical Conference, Fayetteville, AR, pp. 163-166, 2007.

[11] T. Desell, K. E. Maghraoui, and C. A. Varela, "Malleable applications for scalable high performance computing," Cluster Computing, pp. 323–337, June 2007.

[12] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. ACM SIGMETRICS Performance Evaluation Review, 33[1]:291-302, jun 2005.

[13] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites," in Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, 2002, pp. 293–304.

[14] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in Proc. ACM Symp. Oper. Syst. Princ. (SOSP'01), Oct. 2001, pp. 103–116.

[15] A. Cohen, S. Rangarajan, and H. Slye, "On the performance of tcpsplicing for url-aware redirection," in Proc. 2nd Conf.USENIX Symp.Internet Technol. Syst., 199.