ISSN No. 0976-5697

Volume 6, No. 7, September-October 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Research on Analysis of various Load Balancing Algorithm in Minimizing Load Balancing

Deepika Bhanot
M. Tech Computer Science
Punjab Technical University, India

Abstract: Load balancing was identified as a major concern to allow Cloud computing to scale up to increasing distributed solution is required, as it is not practical or cost efficient in many cases to maintain idle service/hardware provision merely to keep up with all identified demands. Equally, when dealing with such complexity, it is impossible to fully detail all system future states. Therefore, it is necessary to allow local reasoning through distributed algorithms on the current system state. Our early work suggested that efficient load balancing cannot be achieved by individually assigning jobs to appropriate servers. In this paper we have proposed algorithm HARA algorithm for load balancing in cloud computing.

Keywords: green Cloud,HRA ,ARA, Cloud Computing

I. INTRODUCTION

With the growth[1] of high speed networks over the last decades, there is an alarming rise in its usage comprised of thousands of concurrent e-commerce transactions and millions of Web queries a day. This ever-increasing demand is handled through large-scale datacenters, which consolidate hundreds and thousands of servers with other infrastructure such as cooling, storage and network systems. Many internet companies such as Google, Amazon, eBay, and Yahoo are operating such huge datacenters around the world.

The commercialization of these developments is defined currently as Cloud computing [1], where computing is delivered as utility on a pay-as-you-go basis. Traditionally, business organizations used to invest huge amount of capital and time in acquisition and maintenance of computational resources. The emergence of Cloud computing is rapidly changing this ownership-based approach to subscription-oriented approach by providing access to scalable infrastructure and services on-demand. Users can store, access, and share any amount of information in Cloud. That is, small or medium enterprises/organizations do not have to worry about purchasing, configuring, administering, and maintaining their own computing infrastructure.[1]

Cloud is Common, Location-independent, Online Utility provisioned on-Demand. Cloud computing has recently revealed technology that is used for hosting and delivering services over the Internet. Figure1 shows the diagrammatical representation of Cloud Computing. Some studies show that Cloud computing can actually make traditional datacenters more energy efficient by using technologies such as resource virtualization and workload consolidation. The traditional data centres running Web applications are often provisioned to handle sporadic peak[2]

Cloud computing is a collection of a variety of computing concepts in which thousands of computers communicate in real-time to provide a seamless experience to the user, as if he/she is using a single huge resource. This system provides multiple facilities like - web data stores, huge computing resources, data processing servers, etc. The concept of cloud computing is around the early 1950s, although the term was not coined back then[7].

Cloud is the metaphor for the Internet, based on how it is depicted in computer network and is an abstraction for complex infrastructures. Cloud computing is an evolving paradigm which is enabling outsourcing of all IT needs such as storage, computation and software through large internet. Computing is a pool of resources. It provides mandatory application environment. It can deploy, allocate or reallocate computing resources dynamically and monitor the usage of resources at all times. The user needs not to care about how to buy servers, softwares, solutions

and so on. They can buy the computing resources through internet according to their own needs.[8]

Computing is a pool of resources. It provides mandatory application environment. It can deploy, allocate or reallocate computing resources dynamically and monitor the usage of resources at all times. The user needs not to care about how to buy servers, softwares , solutions and so on. They can buy the computing resources through internet according to their own needs.

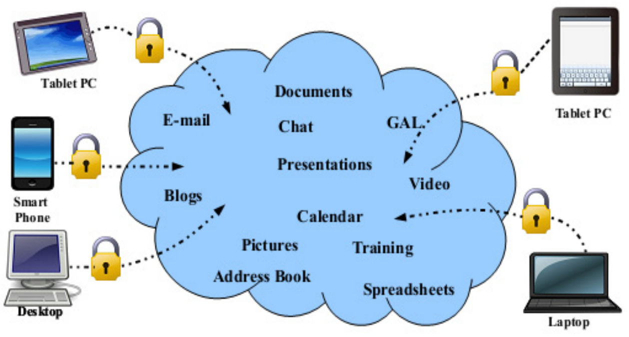

Figure 1: Cloud Computing

loads, which can result in low resource utilization and wastage of energy. Cloud datacenter, on the other hand, can reduce the energy consumed through server consolidation, whereby different workloads can share the same physical host using virtualization and unused servers can be switched off.

II. RELATED WORK

The various algorithms are there for load balancing following are:

In this second load balancing approach, the load on a server is represented by its connectivity in a virtual graph. A full analysis of this mechanism is found in, with this section providing a brief overview. The initial network is constructed with virtual nodes to represent each server node, with each in-degree mapped to the server's free resources or some measure of desirability. As such, a number (consistent with its available resources) of inward edges are created, connected from randomly-selected nodes. This approach creates a network system that provides a measure of initial availability status, which as it evolves, gives job allocation and usage dynamics. Edge dynamics are then used to direct the load allocation procedures required for the balancing scheme. When a node executes a new job, it removes an incoming edge; decreasing its in-degree and indicating available resources are reduced. Conversely, when the node completes a job, it follows a process to create a new inward edge; indicating available resources are increased again. In a steady state, the rate at which jobs arrive equals the rate at which jobs are finished; the network would have a static average number of edges. In ideally-balanced conditions, the degree distribution is maintained close to an Erdős-Rényi (ER) random graph degree distribution [3]. The increment and decrement process is performed via Random Sampling. The sampling walk starts at a specific node; at each step moving to a neighbor node chosen randomly. The last node in the sampling walk is selected for the load allocation. The effectiveness of load distribution is considered to increase with walk length, referred to herein as w. However, experimentation demonstrated an effective w threshold is around log (n) steps, where n is the network size . Therefore, w is used to control the behavior of a node upon receiving a job. When a node receives a job it will execute it if the job's current walk length is greater than or equal to the walk length threshold. This node is then referred to as the job's executing node. Alternatively, if the walk length is less than the threshold, the job's w value is incremented and it is sent to a random neighbor, thus continuing the Random Sampling approach. When the job reaches the executing node, this allocation is reflected in the graph with the deletion of one of the executing node's in-edges. Once the job has completed, the result of the allocation is reflected by the creation of a new edge from the initiating node to the executing node. To summaries, the balancing graph is altered in the following manner by job execution and completion the executing node's in-degree (available resources) decreases during job execution, followed by the allocating node's out-degree (i.e. allocated jobs) and the executing node's in-degree (available resources) increases after job execution, thus directing future load allocation. The result is a directed graph, where the direction of the edges leads the propagation for random sampling. The load-balancing scheme is both decentralized and easily implemented using standard networking protocols. As the scheme is decentralized, this makes it suitable for many large network systems such as those required for Cloud computing platforms. As noted in [4], the performance of this load-balancing technique can be further improved by biasing the random sampling toward specific nodes. Hence, selection will be based on a predefined criterion (e.g. computing power or communication latency) rather than selecting the last node in the walk. For instance, the random sampling walk may be directed towards unvisited nodes or nodes with certain properties. Accordingly, the load balancing technique is improved by assigning the new job to the least loaded (highest in-degree) node in the walk, instead of the last node in the walk. Therefore, the scalability is identical to standard random sampling, yet the balancing performance is much improved. workload evenly to the entire node in the whole cloud to achieve a high user satisfaction and resource utilization ratio. It also ensures that every computing resource is distributed efficiently and fairly. There are various researchers who have used the load balancing techniques to propose new strategies. Their work done in the domain of load balancing is analyzed and compared. But the issues of laod balancing are still open for research work so that higher user satisfaction and resources utilization can be achieved. We have discussed on basic concepts of cloud computing and load balancing and studied some existing load balancing algorithms, which can be applied to clouds. The performance of these strategies with respect to timing and to effect of link clods. The performance of these strategies with respect to the timing and the effect of link and measurement speed were studied. A comparison is also made between different strategies. Future Scope

Cloud computing is a vast concept and load balancing plays a very important role in case of clouds. There is huge scope of improvement in this area. We have discussed only two divisible load scheduling algorithms that can be applied to clouds, but there are still other approaches that can be applied to balance the load in clouds. The performance of the given algorithms can also be increased by varying different parameters.

It was discovered that implementation of the honeybee-foraging distributed load balancing solution at the application layer caused a particular topology to emerge at the resource layer. This manifested itself as a small number of services attracting a disproportionate amount of connectivity from cooperating services whilst most services had only a small number of links. As such, well used services may be grouped to deal with load balancing through the topology of the Cloud's resources. *Active Clustering* is considered in [5] as a self-aggregation algorithm to rewire the network. This procedure is intended to group like (i.e. similar service type) instances together. Many load balancing algorithms only work well where the nodes are aware of "like" nodes and can delegate workload to them [6]. Active Clustering consists of iterative executions by each node in the network:

III. PROPOSED WORK

Prposed Algorithm

Algorithm: Version of ARA

```
 1. initialize a. number of candidates: K = k;
```

```
b. information query delay: D = d;
 /* load information updating*/
2. for each window of D time
a. send queries to all computing sites for load information;
b. update load information received from all computing sites;
end
/* site selection process */
3. upon each job arriving
a. sort all sites Si, 1 ≤ i ≤ N, by current load information;
b. set S = {S1, S2, ..., SK};
/* get K sites with least load */
c. set s = uniform(1, K);
/* randomly select one site from the candidate set S */
d. submit the job to site Ss;
end
```

Given an incoming job and N available computing sites, ARA finds K sites, where K≤ N, as the best candidates for serving that job, using queue length as the ranking criterion. Then, that particular job will be randomly submitted or enqueued to one site among the selected K candidates. The value of K in ARA is critical for system performance, which in turn should be set appropriately based on the intensity of burstiness in workloads. For example,

• under the case of no burstiness in arrivals, K is set to small values (i.e., close to 1). It turns out that ARA performs exactly the same as the “greedy” load balancer, always selecting the best site with shortest queue length;

• under the case of extremely strong burstiness in arrivals, the number of best candidates is set equal (or close) to the total number of available sites, i.e., K = N. Consequently, ARA has behavior similar to the “random” method, which allows the bursty workload to be shared among all sites, therefore alleviating the imbalance of load;

• otherwise, K is set to the value between 1 and N.

As a result, ARA dispatches the load among sites

In proposed priority based scheduling algorithm we have modified the scheduling for executing highest priority task with advance reservation by preempting best-effort task as done shows the pseudo codes of priority based scheduling algorithm (PBSA)

```
Algorithm  Priority Based Scheduling Algorithm (PBSA)
1. Input: UserServiceRequest
2. //call Algorithm 1 to form the list of task based on
priorities
3. getglobalAvailableVMList and
gloableUsedVMList and also
availableResourceList from each cloud schedular
4. // find the appropriate VM Listfromeach cloud
scheduler
5. if AP(R,AR) != ɸ then
6. // call the algorithm 1 load balancer
7. deployableVm=load-balancer(AP(R,AR))
8. Deploy service on deployableVM
9. deploy=true
10. Else if R has advance reservation and best-effort
task is running on any cloud then
11. // Call algorithm 3 CMMS for executing R
with advance reservation
12. Deployed=true
13. Else if globalResourceAbleToHostExtraVM then
14. Start newVMInstance
15. Add VMToAvailbaleVMList
16. Deploy service on newVM
17. Deployed=true
18. Else
19. queueserviceReuest until
20. queueTime>waitingTime
21. Deployed=false
22. End if
23. If deployed then
24. return successful
25. terminate
26. Else
27. return failure
28. terminate
```

As shown above in Algorithm PSBA, the customers’ service deployment requests (R), which is composed of the SLA terms (S) and the application data (A) to be provisioned, is provided as input to scheduler (line 1 in Algorithm 1). When service request (i.e. job) arrive at cloud scheduler, scheduler divide it in tasks as per there dependencies then the Algorithm PSBA is called to form the list of tasks based to their priority (line 2). In the first step, it extracts the SLA terms, which forms the basis for finding the VM with the appropriate resources for deploying the application. In next step, it collects the information about the number of running VMs in each cloud and the total available resources (AR) (line 3). According to SLA terms appropriate VMs (AP) list is form, which are capable of provisioning the requested service (R) (lines 4-5). Once the list of appropriate VMs is form, the Algorithm 1- load-balancer decides which particular VM is allocated to service request in order to balance the load in the data center of each cloud (lines 6-9). When there is no VM with the appropriate resources running in the data center of any cloud, the scheduler checks if service request (R) has advance reservation then it search for best effort task running on any cloud or not, if it found best-effort task then it calls

IV. CONCLUSION

In this thesis the load balancing issue in cloud computing and analyzed various Algorithm techniques used in load balancing. In cloud computing load balancing is the main issue. Load balancing is required to distribute the excess dynamic local. We proposed algorithm for efficient load balancing. Various factor and parameter can be used for further future scope

V. REFERENCES

[1] Saurabh Kumar Garg, Rajkumar Buyya, “Green Cloud Computing and Environmental Sustainability”,IEEE (2012)

[2] Gaganpreet Kaur Sehdev, Anil Kumar, “ Power Efficient VM Consolidation Using Live Migration- A step towards Green Computing”,IJSR,Vol.3,Issue 3,(March 2014).

[3]E. Di Nitto, D.J. Dubois, R. Mirandola, F. Saffre and R. Tateson, Applying Self-Aggregation to Load Balancing: Experimental Results. In Proceedings of the 3rd international Conference on Bioinspired Models of Network, information and Computing Systems (Bionetics 2008), Article 14, 25 – 28 November, 2008.

[4]O. Abu- Rahmeh, P. Johnson and A. Taleb-Bendiab, A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks, INFOCOMP - Journal of Computer Science, ISSN 1807-4545, 2008,

[5]G. Cybenko, Dynamic Load Balancing for Distributed Memory Multiprocessors. Journal of Parallel and Distributed Computing Vol. 7(2), pp: 279-301, 1989.

[6]W.M.P. van der Aalst, Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72-76, 2003.

[7]Ankita Atrey, Nikita Jain and Iyengar N.Ch.S.N.. " A Study On Green Cloud Computing", International Journal Of Grid and Distributed Computing Vol.6, No.6 (2013).

[8]Paulami Dalapati , G.Sahoo, "Green Solution for Cloud Computing with Load Balancing and Power Consumption Management", IJETAE, Vol. 3,Issue 3,(March 2013)