Load Balancing and Parallelism in Cloud Computing

Pragati Priyadarshinee, Pragya Jain

Abstract- Large-scale heterogeneous distributed computing environments (such as Computational Grids and Clouds) offer the promise of access to a vast amount of computing resources at a relatively low cost. In order to ease the application development and deployment on such complex environments, high-level parallel programming languages exist that need to be supported by sophisticated runtime systems. The anticipated uptake of Cloud computing, built on well-established research in Web Services, networks, utility computing, distributed computing and virtualization, will bring many advantages in cost, flexibility and availability for service users. These benefits are expected to further drive the demand for Cloud services, increasing both the Cloud's customer base and the scale of Cloud installations. This has implications for many technical issues in Service Oriented Architectures and Internet of Services (IoS)-type applications; including fault tolerance, high availability and scalability. Central to these issues is the establishment of effective load balancing techniques. It is clear the scale and complexity of these systems makes centralized assignment of jobs to specific servers infeasible; requiring an effective distributed solution.

Index Terms: Load Balancing, Parallelism, SOA, Virtualization.

I. INTRODUCTION

Cloud computing is an on demand service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. It's a term which is generally used in case of Internet. The whole Internet can be viewed as a cloud. Capital and operational costs can be cut using cloud computing.Load balancing in cloud computing systems is really a challenge now. Always a distributed solution is required. Because it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfill the required demands. Jobs cant be assigned to appropriate servers and clients individually for efficient load balancing as cloud is a very complex structure and components are present throughout a wide spread area. Here some uncertainty is attached while jobs are assigned. This paper considers some of the methods of load balancing in large scale Cloud systems.Our aim is to provide an evaluation and comparative study of these approaches.

II. BACKGROUND STUDY

A cloud consists of three major components. Those are clients, data centers and distributed servers.

Manuscript Received on June 22, 2012.

Pragati Priyadarshinee, Assistant Professor,Chaitanya Bharathi Institute of Technology,Hyderabad, India

Dr. Pragya Jain, incharge of the "grid computing" and "cloud computing" at IIT Delhi, India.

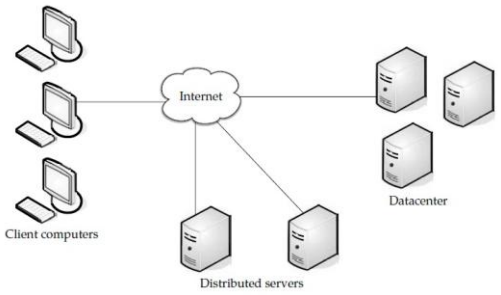

Figure 1:Three components make up a cloud computing solution([1]).

Clients:

End users interact with the clients to manage information related to the cloud. Clients generally fall into three categories as given in [1]:

_ Mobile: Windows Mobile Smartphone, smartphones, like a Blackberry, or an iPhone.

_ Thin: They don't do any computation work. They only dispaly the information.

Servers do all the works for them. Thin clients don't have any internal memory.

_ Thick: These use different browsers like IE or mozilla Firefox or Google Chrome to connect to the Internet cloud.

Now-a-days thin clients are more popular as compared to other clients because of their low price, security, low consuption of power, less noise, easily replaceble and repairable etc.

Datacenter:

Datacenter is nothing but a collection of servers hosting different applications. A end

user connects to the datacenter to subscribe different applications. A datacenter may exist

at a large distance from the clients.

Now-a-days a concept called virtualisation is used to install a software that allow multiple instances of virtual server applications.

Distributed Servers:

Distributed servers are the parts of a cloud which are present throughout the Internet hosting different applications. But while using the application from the cloud, the user will feel that he is using this application from its own machine.

Type of Clouds

Based on the domain or environment in which clouds are used, clouds can be divided into 3 catagories :

1)Public Clouds

2)Private Clouds

3)Hybrid Clouds (combination of bothe private and public clouds)

Virtualization

It is a very usefull concept in context of cloud systems. Virtualization means ”something which isn’t real”, but gives all the facilities of a real. It is the software implementation of a computer which will execute different programs like a real machine.

Virtualisation is related to cloud, because using virtualization an end user can use different services of a cloud. The remote datacenter will provide different services in a fully or partial virtualized manner.

2 types of virtualization are found in case of clouds as given in [1] :

(a) Full virtualization
(b) Paravirtualization

Services Provided by Cloud Computing:

Service means different types of applications provided by different servers across the
cloud. It is generally given as ”as a service”. Services in a cloud are of 3 types as given in
[1] :
(a) Software as a Service (SaaS)
(b)Platform as a Service (PaaS)
(c)Hardware as a Service (HaaS) or Infrastructure as a Service (IaaS)

Goals Of Load Balancing:

As given in [4], the goals of load balancing are :

(1) To improve the performance substantially

(2)To have a backup plan in case the system fails even partially

(3)To maintain the system stability

(4)To accommodate future modification in the system

III. TYPES OF LOAD BALANCING ALGORITHMS

A. Dynamic Load Balancing Algorithm:

In a distributed system, dynamic load balancing can be done in two different ways: distributed and non-distributed. In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. The interaction among nodes to achieve load balancing can take two forms: cooperative and non-cooperative [4].Dynamic load balancing algorithms of distributed nature, usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to interact with every other node. A benefit, of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt, it instead would effect the system performance to some extent. Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. In non-distributed type, either one node or a group of nodes do the task of load balancing. Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed. In the first form, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node. In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load balancing within that cluster.

Hence, the load balancing of the whole system is done via the central nodes of each cluster[4].

Strategies in Dynamic Load Balancing:

1)Transfer Policy: The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy.

2) Selection Policy: It specifies the processors involved in the load exchange (processor matching)

3) Location Policy: The part of the load balancing algorithm which selects a destination node for a transferred task is reffered to as location policy or Location strategy.

4)Information Policy: The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is reffered to as Information policy or Information strategy.

B. Distributed Load Balancing For the Clouds:

(a)Honeybee Foraging Algorithm:

In load-balancing operation,[2] each server takes a particular bee role with probabilities *px* or *pr*. These values are used to mimic the honeybee colony whereby a certain number of bees are retained as foragers – to explore (*px*);rather than as harvesters – to exploit existing sources. A server successfully fulfilling a request will post on the advert board with probability *pr*. A server may randomly choose a virtual server’s queue with probability *px*(exploring), otherwise checking for an advert (watching a*waggle dance*). In summary, idle servers (waiting bees) follow one of two behaviour patterns: a server that reads the advert board will follow the chosen advert, then serve the request; thus mimicking harvest behaviour. A server not reading the advert board reverts to forage behaviour;servicing a random virtual server’s queue request.

An executing server will complete the request and calculate the profitability of the just-serviced virtual server.

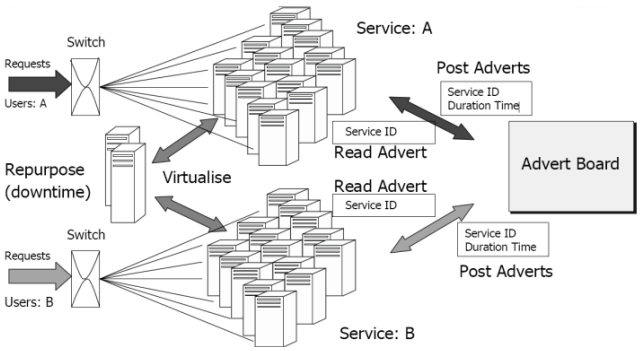

Figure 2: Virtual Servers and Advert Boards[2]

(b)Biased Random Sampling:

Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each indegree directed to the free resources of the server.

Regarding job execution and completion, whenever a node does or executes a job, it deletes an incoming edge, which indicates reduction in the availability of free resource. After completion of a job, the node creates an incoming edge, which indicates an increase in the availability of free resource. The addition and deletion of processes is done by the process of random sampling. The walk starts at any one node and at every step a neighbor is chosen randomly. The last node is selected for allocation of load. Alternatively, another method can be used for selection of a node for load allocation, that being selecting a node based on certain criteria like computing efficiency, etc. Yet another method can be selecting that node for load allocation which is under loaded i.e. having highest in degree. A node upon receiving a job, will execute it only if its current walk length is equal to or greater than the threshold value. Else, the walk length of the job under consideration is incremented and another neighbor node is selected randomly. When, a job is executed by a node then in the graph, an incoming edge of that node is deleted. After completion of the job, an edge is created from the node initiating the load allocation process to the node which was executing the job. Finally what we get is a directed graph. The load balancing scheme used here is fully decentralized, thus making it apt for large network systems like that in a cloud.

(c)Active Clustering:

Active Clustering works on the principle of grouping similar nodes together and working on these groups. The process involved is:

1.A node initiates the process and selects another node called the matchmaker node from its neighbors satisfying the criteria that it should be of a different type than the former one.

2.The so called matchmaker node then forms a connection between a neighbor of it which is of the same type as the initial node.

3.The matchmaker node then detaches the connection between itself and the initial node.

IV. IMPLEMENTATION

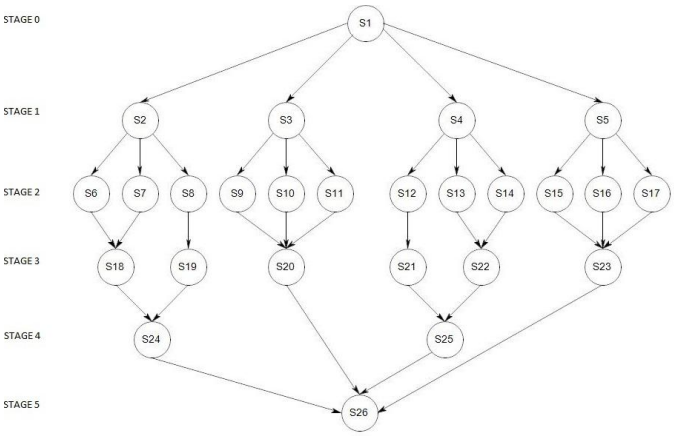

Figure 3. State Diagram

The time required for completing a task within one process is very high. So the task is divided into no. of sub-tasks and each sub-task is given one one job. Let the task S is divided into no. of sub-tasks S1,S2,S3...Sn. Out of these some are executed sequentially and some are executed parallely. So the total time period for completing the task decreases and hence the performance increases. These sub-tasks can be represented in a graph structure known as state diagram. An example is given below.S1 is executed first. S2,S3,S4 and S5 can be executed parallely during the same timeslice. S18 requires the execution of S6 and S7 both, but S19 requires the execution of S8 and so on for all the sub tasks as shown in the state diagram. Our aim is to execute these tasks in different nodes of a distributed network so that the performance can be enhanced.

V. RESULTS AND PERFORMANCE EVALUATION

Here some calculation has been done using Matlab.As per that we plotted the timing diagram based on Master-Slave concepts;where as one master computer and N slaves are there.

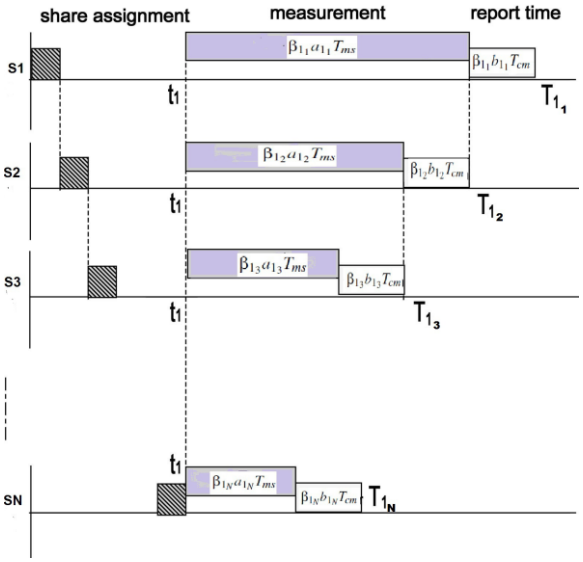

Figure 4. Timing Diagram for Single Level Network(9)

Correspondingly, we have plotted some graphs based on the measurement time or reporting time and number of processors or slaves.

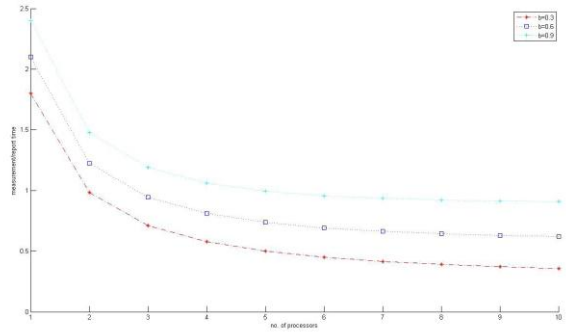

Figure 5.Measurement Time/Report Time Vs No of Processors/Slaves

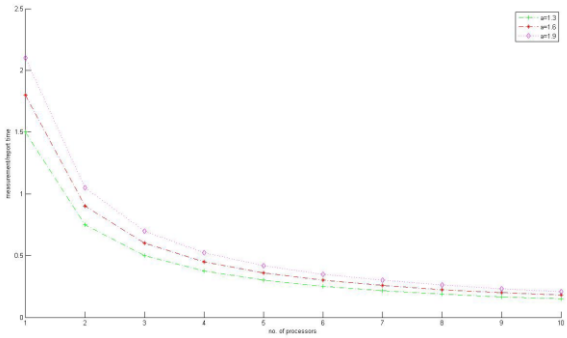

Figure 6.Measurement Time Vs No of Slaves.

Fig. 6 shows for the case when the inverse measuring speed a is varied from 1 to 2 at an interval of 0.3 and the inverse link speed b is fixed to be 0.2.

VI. CONCLUSION AND FUTURE WORK

Cloud Computing is a vast concept and load balancing plays a very important role in case of Clouds. There is a 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.

REFERENCES

1. Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach TATA McGRAW-HILL Edition 2010.
2. Martin Randles, David Lamb, A. Taleb-Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.
3. Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceedings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, 2008, June 23-26.
4. Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.
5. http://www-03.ibm.com/press/us/en/pressrelease/22613.wss
6. http://www.amazon.com/gp/browse.html?node=201590011
7. Martin Randles, Enas Odat, David Lamb, Osama Abu- Rahmeh and A. Taleb-Bendiab, ”A Comparative Experiment in Distributed Load Balancing”, 2009 Second International Conference on Developments in eSystems Engineering.
8. Peter S. Pacheco,”Parallel Programming with MPI”, Morgan Kaufmann Publishers Edition 2008
9. Mequanint Moges, Thomas G.Robertazzi, ”Wireless Sensor Networks: Scheduling for Measurement and Data Reporting”, August 31, 2005

AUTHORS PROFILE

Pragati Priyadarshinee received her BE from Utkal University in 2004 and M.Tech(SE) degree from Indian Institute of Information Technology,Allahabad(IIITA)in 2008.She is Assistant Professor,Chaitanya Bharathi Institute of Technology,Hyderabad.Her research interest include Cloud computing, Software Engineering and Software Architecture.She is a member of IACSIT(INTERNATIONAL ASSOCIATION OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY).She has three paper publications; one in international conference and two in international journals.

Dr. Pragya Jain after her post graduation in Mathematics from Kurukshetra University did her Ph.D. from IIT Delhi in Numerical Analysis.Now,Dr. Jain is in charge of the “grid computing” and “cloud computing” at IIT Delhi.She has been teaching under-graduate and post-graduate courses since last 22 years at IITD.She has also been conducting courses on parallel computing and programming languages for participants from industry,teachers and students of other institutes and colleges.She has guided many M.Tech and B.Tech projects of students from IIT and other institutes.