A Novel Superlative Flawless Solution For Load Balancing In Cloud Computing

Durga.S[1], Manikandan.M[2]

[1]PG Student, Department of Computer Science, Vidyaa Vikas College of Engineering and Technology Tiruchengode, India

[2]Associate Professor, Department of Computer Science, Vidyaa Vikas College of Engineering and Technology Tiruchengode, India

ABSTRACT

"Innovation is necessary to ride the inevitable tide of change".

The cloud is a futuristic platform that provides dynamic resource pools, virtualization, and high availability and enables the sharing, selection and aggregation of geographically distributed heterogeneous resources for solving large-scale problems in science and engineering. Load Balancing is an important aspect of cloud computing environment in which the workload is evenly distributed among set of servers. In this paper we define an power-optimal operation regime and attempting to maximize the number of servers operating in this regime. Idle and lightly-loaded servers are switched to one of the sleep states to save energy. We propose power saving load balancing application scaling algorithm to distribute the load evenly and save energy.

Keywords:-load balancing, cloud computing, application scaling, idle servers.

1. INTRODUCTION

Cloud computing describes a data processing infrastructure in which the application software and often the data itself is stored permanently not on your PC but rather a remote server that's connected to the Internet. As the name suggests, the function of the cloud is to provide individuals and small and mid-sized businesses access to an array of powerful applications and services through the internet and not concerned about the basic underlying complexities involved in delivering services. 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. Cloud elasticity is the ability to use as much resources as needed at any given time and low cost, a user pays only for the resources it consumes, represent solid incentives for many organizations to migrate some of their computational activities to a public cloud.

Load balancing is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded. A load balancing algorithm which is dynamic in nature does not consider the previous state or behavior of the system, that is, it depends on the present behavior of the system. The goals of load balancing are to improve the performance substantially, to have a backup plan in case the system fails even partially, to maintain the system stability, to accommodate future modification in the system.

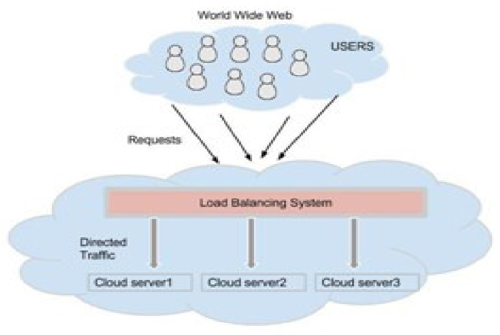

Fig 1: Load Balancing system in cloud computing

Scaling is the process of allocating additional resources to a cloud application in response to a request consistent with the Service Level Agreement and a cloud user. We distinguish two scaling modes, horizontal and vertical scaling. Horizontal scaling is the most common mode of scaling on a cloud; it is done by increasing the number of Virtual Machines (VMs) when the load of application. Vertical scaling keeps the number of VMs of an application constant, but increases the amount of resources allocated to each one of them. This can be done either by migrating the VMs to more powerful servers, or by keeping the VMs on the same servers, but increasing their share of the CPU time. This research work is based on simulation technique and it uses the cloud simulator, CloudSim.

2. EXISTING LOAD BALANCING ALGORITHMS

Virtual machine enables the abstraction of an operating system and Application running on it from the hardware. The interior hardware infrastructure services interrelated to the Clouds are modelled in the simulator by a Datacenter element for handling service requests. These requests are application elements sandboxed within Virtual Machines, which need to be allocated a share of processing power on Datacenter's host components. Data Center object manages the data center management activities such as Virtual Machine creation and destruction and does the routing of user requests received from User Bases via the Internet to the Virtual Machines. The Data Center Controller, uses a Virtual Machine Load Balancer to determine which Virtual Machine should be assigned the next request for processing. Most common Virtual machine load balancer are throttled and active monitoring load balancing algorithms.

I. Token Routing: This algorithm allows fast, efficient routing decisions, without requiring accurate knowledge of the complete global state.The main objective of the algorithm is to minimize the system cost by moving the tokens around the system. But in a scalable cloud system agents have no knowledge of their neighbours working load distribution; therefore they have no idea where to forward the tokens. To assist their decisions, agents build their knowledge base. In this algorithm, agents' knowledge base is build solely from the tokens previously received. Therefore, no additional communication and observation are required.

II. Round Robin: In this algorithm, the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. This algorithm is mostly used in web servers where Http requests are of similar nature and distributed equally.

III. Randomized : In Randomized algorithm The process allocation order is maintained for each processor independent of allocation from remote processor. .It is of type static in nature. In this algorithm a process can be handled by a particular node n with a probability p. This algorithm works well in case of processes are of equal load. However, problem arises when loads are of different computational complexities. Randomized algorithm does not maintain deterministic approach. It works well when Round Robin algorithm generates overhead for process queue.

IV. Central queuing: This algorithm works on the principal of dynamic distribution. Each new activity arriving at the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. If no ready activity is present in the queue the request is buffered, until a new activity is available. But in case new activity comes to the queue while there are unanswered requests in the queue the first such request is removed from the queue and new activity is assigned to it. When a processor load falls under the threshold then the local load manager sends a request for the new activity to the central load manager. The central manager then answers the request if ready activity is found otherwise queues the request until new activity arrives.

V. Connection mechanism: This algorithm is based on the least number of connection mechanisms which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server to estimate the load. The load balancer will record the number of connections for each server. The number of connection increases by one when a new connection is dispatched to it while it decreases the number by one when connection finishes or timeout happens.

VI. Throttled Load Balancer: Throttled algorithm is completely based on virtual machine. It will find the node for assigning the new task. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is give by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

3. PROPOSED LOAD BALANCING ALGORITHMS

Power saving load balancing algorithms is proposed to ensure that the largest possible number of active servers operate within the boundaries of their respective optimal operating regime. The actions implementing this policy are: (a) migrate Virtual Machines from a server operating in the undesirable-low regime and then switch the server to a sleep state; (b) switch an idle server to a sleep state and reactivate servers in a sleep state when the cluster load increases; (c) migrate the VMs from an overloaded server, a server operating in the undesirable-high regime with applications predicted to increase their demands for computing in the next reallocation cycles.

The clustered organization is maintained which allows us to accommodate some of the desirable features of the strategies for server consolidation. The cluster leader has relatively accurate information about the cluster load and its trends. The leader could use predictive algorithms to initiate a gradual wake-up process for servers in a deeper sleep state, C4 - C6, when the workload is above a “high water mark" and the workload is continually increasing. We set up the high water mark at 80% of the capacity of active servers; a threshold of 85% is used for deciding that a server is overloaded. The leader could also choose to keep a number of servers in C1 or C2 states because it takes less energy and time to return to the C0 state from these states. The energy management component of the hypervisor can use only local information to determine the regime of a server.

Scaling decisions followed in our proposed system are the Server Application Manager SAMk is a component of the Virtual Machine Monitor (VMM) of a server Sk. One of its functions is to classify the applications based on their processing power needs over a window of w reallocation intervals in several categories: rapidly increasing resource demands (RI), moderately increasing (MI), stationary (S), moderately decreasing (MD), and rapidly decreasing (RD). This information is passed to the cloud leader whenever there is the need to migrate the VM running the application.

SAMk interacts with the cluster leader and with the application managers of servers accepting the migration of an application currently running on server Sk. A report sent to the cluster leader includes the list of applications currently running on Sk, their additional demands of over the last reallocation cycle and over a window of w reallocation intervals, and their classification as RI/MI/S/MD/RD over the same window. The scaling decisions are listed in the order of their energy consumption, overhead, and complexity:
(1) Local decision - whenever possible, carry out vertical application scaling using local resources.
(2) In-cluster, horizontal or vertical scaling - migrate some of the VMs to the other servers identified by the leader; wake-up some of the servers in a sleep state or switch them to one of the sleep states depending on the cluster workload.
(3) Inter-cluster scaling - when the leader determines that the cluster operates at 80% of its capacity with all servers running, the admission control mechanism stops accepting new applications. When the existing applications scale up above 90% of the capacity with all servers running then the cluster leader interacts with the leaders of other clusters to satisfy the requests.

The leader of a cluster maintains several control structures. OptimalList - includes servers operating in an optimal regime or those in suboptimal regime projected to migrate back to the optimal regime; the list is ordered in the increasing order of computing power. Within a group of servers with similar k, the servers are ordered in the increasing order of available capacity. WatchList - includes servers running in two suboptimal regime and the undesirable-high regimes whose applications are candidates for VM migration. MigrationList - includes servers operating in undesirable regimes. The leader selects candidates for migration and identifies possible targets for migration. SleepList - includes servers in one of the sleep states; the list is ordered on the type of sleep state and then in the increasing order of computing power reflected by the constant k.

4. SIMULATORS

The main aim of simulator is to test the implementation work in the absence of the required environment. Here we use CloudSim simulator. It is the open source and a new generalized and extensible simulation framework that enables seamless modeling, simulation, experimentation of emerging Cloud computing infrastructures and management services.It implements generic application provisioning techniques that can be extended with ease and limited effort.

The simulation framework has the following novel features: (i) support for modeling and instantiation of large scale Cloud computing infrastructure, including data centers on a single physical computing node and java virtual machine; (ii) a self-contained platform for modeling data centers, service brokers, scheduling, and allocations policies; (iii) availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node; and (iv) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

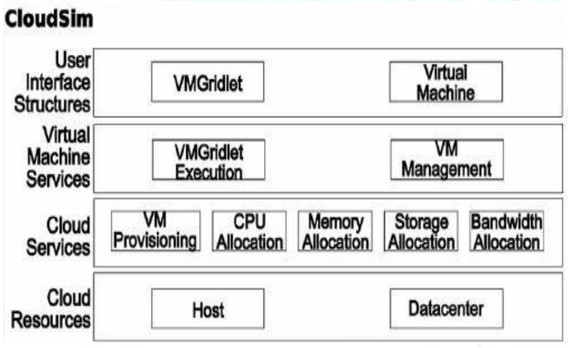

Figure 2: CloudSim Architecture

5. CONCLUSION

Load Balancing is an essential task in Cloud Computing environment to achieve maximum utilization of resources. This paper aims towards the establishment of performance qualitative analysis on existing VM load balancing algorithm and then implemented in CloudSim and java language.Here we emphasis on the scheduling of jobs for proper load balancing in virtual machines and then putting the idle or unused machines to sleep mode for better power management. Thus we provide cloud elasticity and save energy at low cost. Our future work will evaluate the overhead and the limitations of the algorithm proposed in this paper; it will also include the implementation of a Server Application Manager and the evaluation of the overhead for the algorithm proposed in this paper.

6. REFERENCES

[1].Basic concept and terminology of cloud computing- http://whatiscloud.com

[2].L. Wang, J. Tao, M. Kunze,"Scientific Cloud Computing: Early Definition and Experience", the 10th IEEE International Conference Computing and Communications 2008.

[3].Load Balancing in Cloud computing, http://community.citrix.com/display/cdn/Load+Balancing

[4].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.

[5].Ram Prasad Padhy , P Goutam Prasad Rao, "LOAD BALANCING IN CLOUD COMPUTING SYSTEMS"

[6].Yi Zhao, Wenlong Huang, 2009 "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud" Fifth International Joint Conference on INC, IMS and IDC.

[7].P.Jamuna,R.Anand, "Optimized Cloud Partitioning Technique to Simplify Load Balancing"

[8].T. Kokilavani J.J. College of Engineering & Technology and Research Scholar, Bharathiar University, Tamil Nadu, India" Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing" International Journal of Computer Applications.

[9].P.Warstein, H.Situ and Z.Huang (2010), "Load balancing in a cluster computer" In proceeding of the seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE

[10].Jaspreet kaur / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 2, Issue 3, May-Jun 2012, "Comparison of load balancing algorithms in a Cloud".