Load Balancing Strategy for Optimal Peak Hour Performance in Cloud Datacenters

Ashwin Kumar Kulkarni[1], Annappa .B[2]
Department of Computer Science & Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India
ashwin.kulkarni7@gmail.com[1], annappa@ieee.org[2]

***Abstract*—Cloud computing is a growing computing model that is influencing every other entity in the global business industry. Efficient load balancing techniques plays a major role in cloud computing by allocating requests to computing resources efficiently to prevent under/over-allocation of Virtual Machines (VMs) and improve the response time to clients. It is observed that during peak hours when request frequency is high, active VM load balancer (packaged in cloudAnalyst) over-allocates initial VMs and under-allocates later ones creating load imbalance. In this paper we propose a novel VM load balancing algorithm that ensures uniform allocation of requests to virtual machines even during peak hours when frequency of requests received in data center is very high to ensure faster response times to users. The simulations results suggest that our algorithm allocates requests to VM uniformly even during peak traffic situations.**

Keywords- Load balancing; uniform utilization; virtual machines; cloud computing.

I. INTRODUCTION

Cloud computing is classified as a new paradigm for the dynamic provisioning of computing resources delivered by the state-of-art data centers using virtualization technology. Cloud computing allows renting of the IT infrastructure, runtime environment and services on pay-per-use basis to its users. Cloud computing has helped entrepreneurs to setup and start their businesses without the need of heavy initial investment for expensive hardware, software, setup costs and also free them from IT maintenance issues. The cloud computing services can be classified into three categories Infrastructure-as-service(IaaS), platform-as-service(SaaS) and software-as-service(SaaS).

Virtualization is an important and core technology for cloud computing. It allows the abstraction of fundamental elements of computing such as hardware, storage and networking. Virtualization technology has helped the cloud data centers to effectively increase resource utilization, reduce electricity costs and ease management complexities. But there are many challenges in providing services with reliability and performance guarantee in such a complex virtualized environment involving server consolidation. Hence, research community is always working hard to seek new solutions that can solve these challenges and enhance virtual machines (VM) scheduling algorithms, resource allocation, and migration strategies.

Load balancing is one of the critical aspect in cloud computing environment that can significantly improve resource utilization, performance and save energy by properly assigning/re-assigning computing resources to the incoming requests from users. Therefore how to schedule virtual machines (VMs) effectively by considering various parameters that can influence its decision becomes an important research point for cloud computing.

The rest of the paper is organized as follows: The section II discusses the background and related work, in section III we explain active VM algorithm, its problem and propose a novel VM load balancing algorithm, in section IV the detailed experimental setup using cloud analyst is explained to integrate proposed load balancing algorithm with cloud analyst and test using the GUI based framework, section V gives the comparison results of active VM algorithm and proposed algorithm. Finally the section VI analyzes the results and presents the conclusion.

II. BACKGROUND AND RELATED WORK

Load balancing is an important issue in parallel and distributed systems. So far, extensive research work has been done to propose various load balancing approaches. Few of the important load balancing algorithms that are proposed by researchers are explained below.

A load balancing algorithm for VM resources based on genetic algorithm is proposed in literature [1]. Considering the historical data and current state of the system and by using genetic algorithm, a strategy is proposed which can compute the influence it will have on the system after the needed VM resources are deployed and then algorithm chooses the least-affective solution, authors claims to achieve the best load balancing with this algorithm and also it found to reduce or avoid dynamic migration.

An improved Artificial bee colony algorithm [4] proposed solves the problem with existing artificial bee colony algorithm when requests of same type queue up on a single server creating an imbalance. The paper proposes to replace next served requests with a different request type so that it can end the accumulation of requests.

Ren [6] proposed an algorithm for dynamic load balancing in cloud based on an algorithm called weighted least connection(WLC) which allocates the tasks to node having least number of connections however this algorithm does not

978-1-4799-1823-2/15/$31.00 ©2015 IEEE

considers the capability in terms of CPU, memory or network bandwidth to make such decision. The proposed algorithm called ESWLC(Exponential smooth forecast based on weighted least connection) [5] improves the weighted least connection algorithm by considering the capabilities of each node, time series and trials then goes on to decide assignment of a certain task to a node.

Load balancing Min-Min algorithm [7] proposes a three level load balancing framework that uses the opportunistic load balancing [8].OLB algorithm's goal is to keep every node in the cloud busy however the opportunistic load balancing algorithm does not consider the execution time of each task into account for making decisions, this results in slow processing of tasks and introduces performance bottleneck. The load balancing Min-Min algorithm solves this problem by using 3-level architecture, in first level the request manager receives request and allocates to one service manager in the second layer and service manager may break each tasks into smaller sub-tasks for faster processing and allocate each smaller sub-tasks to service nodes in the last layer by considering service node's CPU, Memory and network bandwidth.

In Equally Spread Current Execution (ESCE) load algorithm [9] also known as active VM, load balancer makes an effort to equally spread the execution load on different VMs. Load balancer maintains an index table of VMs along with the number of requests currently allocated to the VM. If there is request comes from the data centre for execution, load balancer search the index table for least loaded VM. If more than one VM is found, first identified VM is selected and allotted for request execution. The load balancer updates the index table by increasing the allocation count of identified VM. When VM finishes the execution of allotted request, load balancer again update the index table by decreasing the allocation counts for identified VM by one.

The proposed algorithm explained in next section ensures even allocation of requests to all VMs to avoid any over-allocation or under-allocation to any particular VM even during the peak hour traffic situations to improve the response time for the user requests arriving at the cloud data enters.

III. PROPOSED ALGORITHM

The section is divided into three sub-sections. The first sub section explains the active VM algorithm as implemented in cloudAnalyst [3], then the problem observed with the current algorithm is explained in the second sub-section and at the end of this section introduces proposed algorithm to overcome the problem.

A. Active VM algorithm

The active VM algorithm allocates the requests to the first least loaded VM at a particular instance of time to maintain uniform allocation for all the VMs. The algorithm for active VM load balancer is explained in this section. The active VM load balancer is utilized by the data center controller for VM allocation to incoming requests.

Algorithm: Active VM load balancer

Initialization:

1. **for** (all VM ids) **do**
   Set allocation table entries to zero
   end
   Also get the VM status table for all VMs from data center controller.
2. Data center controller requests for VM id to the active VM load balancer.

Allocation:

3. **if** (all VMs are not allocated) **then**
   for (for all VM ids) **do**
   Check for index of allocation statistics table with zero allocation. If found return VM id.
   end
   end
4. **else if** (all VMs are busy) **then**
   for (for all VM) **do**
   Find the index of the allocation statistics table with least number and return VM id.
   end
   end
5. Active VM load balancer recommends a VM id to the data center controller unit.
6. Data center controller assigns the VM id to the request.
7. A notification is sent from data center controller to load balancer about recent allocation for the corresponding VM id.
8. Active VM load balancer increments the count for allocation in the allocation statistics table.

De-allocation:

9. When the request completes the processing on VM, data center is notified about completion.
10. Data center then sends the notification to the active VM load balancer to signal VM de-allocation.
11. Active VM load balancer's allocation statistics table is updated by decrementing the count of requests against the allocated VM id.

B. Problem with current Active VM load balancer

The problem with active VM load balancer is observed when request frequency in the data center is very high. During the peak traffic situation, the data center controller queries the active VM load balancer frequently to get the suitable VM id for the allocation to the received request. The active VM load balancer returns the suitable VM id to the data center controller at step 5 but it has to wait till load balancer get the notification from the data center controller about the allocation to update its allocation statistic table at the step 8. If any request is received by the load balancer between step 5 and step 8, the load balancer is ignorant about the VM id allocation and allocation statistics table is not updated to reflect the VM ids returned at step 5 by previous requests.

During peak hours, the data center controller experiences high requests frequency and makes frequent calls to the load balancer to request suitable VM for allocation, this results in

some requests being received and handled by the load balancer between step 5 and step 8 with allocation statistics table not properly updated. This result in over allocation of initial VM's as more requests gets assigned to the VM's listed at the top of the table.

C. Proposed VM load balancer algorithm

We propose a variant of active VM algorithm to solve the issue during peak hours by using a Reservation table.

The proposed VM load balancer maintains an internal reservation table to maintain the information of VM reservations suggested by the load balancer to data center controller but not updated in allocation table until the notification arrives of allocation. The proposed load balancer takes into consideration both reservations table entry and allocation statistics table entry for particular VM id by the load balancer for VM selection for next request. The following proposed algorithm makes sure the algorithm does uniform allocations even during peak hours.

Algorithm: Proposed VM load balancer

Initialization:

1. **for** (all VM ids) **do**
   Set allocation table entries to zero **and reservation table entries to zero**
   end
   Also get the VM status table for all VMs from data center controller.
2. Data center controller requests for VM id to the active VM load balancer.

Allocation:

3. **if** (all VMs are not allocated) **then**
   for (for all VM ids) **do**
   Check for index of allocation statistics table with zero allocation. If found return VM id.
   end
   end
4. **else if** (all VMs are busy) **then**
   for (for all VM) **do**
   Find the index of the allocation statistics table and reservation table with
   min_count(allocation count + reservation count).
   end
   end
5. Active VM load balancer recommends a VM id to the data center controller unit **and updates the reservation table to reflect the allocation against the VM id by incrementing the reservation count**.
6. Data center controller assigns the VM id to the request.
7. A notification is sent from data center controller to load balancer about recent allocation for the corresponding VM id.
8. Active VM load balancer increments the count for allocation in the allocation statistics table **and decrements the reservation table count for corresponding VM id.**

De-allocation:

9. When the request completes the processing on VM, data center is notified about completion.
10. Data center then sends the notification to the active VM load balancer to signal VM de-allocation.
11. Active VM load balancer's allocation statistics table is updated by decrementing the count of requests against the allocated VM id.

The proposed algorithm takes into consideration both allocations and reservations for a particular VM id to make a choice for assigning an incoming request as shown in pseudo code below.

```
Integer findLeastLoadedVM(){
    for (all VM ids in datacenter) {
        next_VM_ID_with_minCount = MIN(
        next_VM_ID_with_minCount ,(ALLOC_COUNT(VM_id) +
        RESERVE_COUNT(VM_id) ));
    }
    return next_VM_ID_with_minCount;
}
```

The call sequence flow of the proposed VM load balancer is shown in Figure 1.

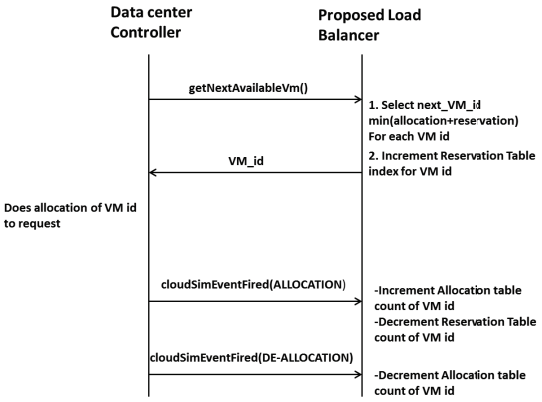

FIGURE 1:CALL FLOW OF PROPOSED ALGORITHM

IV. EXPERIMENTAL SETUP

CloudAnalyst [3] has been used to carry out evaluations of the proposed VM load balancer and compare the result with current active VM algorithm. Cloud analyst simulation tool is based on cloudsim library written in java and provides a GUI interface to configure various parameters to perform the experimental work. Figure 2 shows the architecture of CloudAnalyst tool.

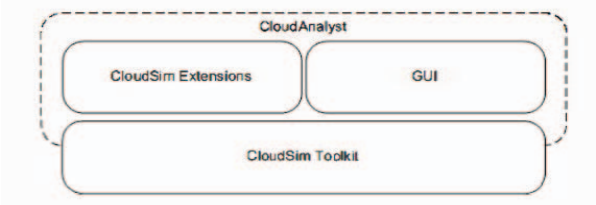

FIGURE 2: CLOUDANALYST ARCHITECTURE

A. *Proposed Algorithm Implementation for cloudAnalyst*

The proposed algorithm is implemented using Java language and integrated into cloudAnalyst source base. The important segments of source code is given in following section for reader's reference.

```
public class ProposedVmLoadBalancer
extends VmLoadBalancer
implements CloudSimEventListener {
    /* Data structure definition section
     .....
    */
    public ProposedVmLoadBalancer(DatacenterController dcb){
          /*
          Initialization section for data structures
           .....
          Memory allocations
          ----
          Register call back for notifications
          with data center controller */
    }
    @Override
    public int getNextAvailableVm(){
        /* Compute the suitable VM id for task allocation here */
        if ( not all VMs are allocated ) {
          /*
            find the first free unallocated VM id and
            mark it suitable for task allocation
          */
        } else {
          /* If all VMs are allocated, then find the least
          loaded VM with proposed algorithm */
          for (all entries in allocation statistics table) {
              //search for VM id with min count
               MIN(SUM(allocation_count + reservation_count));
          }
        }
        /* Increment reservation table*/
        return vmId;
    }
    public void cloudSimEventFired(CloudSimEvent e) {
      if (e.getId() ==
        CloudSimEvents.EVENT_CLOUDLET_ALLOCATED_TO_VM) {
          /* decrement reservation table entry
              increment the allocation statistics table
          */
     } else if (e.getId() ==
         CloudSimEvents.EVENT_VM_FINISHED_CLOUDLET){
        /* Decrement the allocation count for VM */
        if (count reaches zero)
          /* reset reservation count to zero */

    }
  }
}
```

The above source file is placed in the "clousim.ext.datacenter" package.

B. *cloudAnalyst Simulation Configuration*

For experimentation, internet users at four different Continents are considered i.e. four user bases and peak and non-peak users are given in the Table 1.

TABLE 1: USER BASES

RegionName	Region-wise statistics of Users		
	Region	*Peak time Users*	*Off-Peak Users*
North America	0	35000	3500
South America	1	25000	2500
Europe	2	15000	1500
Asia	3	5000	500

Data center hosts physical machines possessing configuration of 100 GB of storage space, 4 GB of RAM with each machine having 4 CPU and a power of 10k MIPS. Evaluations are carried out with the Data center configuration of 5 VMs (running on 2 Physical Machines) and 25 VMs (running on 10 Physical Machines).

V. EXPERIMENTAL RESULTS

Results are analyzed with major focus on the uniform utilization of the virtual machines by avoiding the under or over loading of certain set of VMs in the data centers and then comparing the proposed VM load balancer algorithm with the existing active VM load balancer.

The request allocation numbers of each VM for both current active VM algorithm and proposed VM Load balancer are tabulated in *Table 2* and plotted in the form of a column chart in *Chart no. 1* for the case of Data center configuration of 5 VMs.

TABLE 2: COMPARISON RESULTS FOR 5 VMS IN DC

VM Id #	*# of allocations* By active VM load balancer	*# of allocations* By proposed VM load balancer
0	39554	18502
1	19112	18507
2	14902	18503
3	10097	18504
4	8855	18504

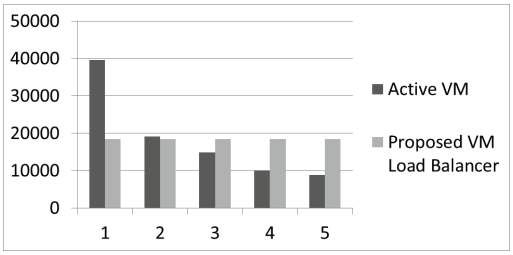

CHART 1: COMPARISON RESULTS FOR 5 VMS IN DC

Table 3 tabulates the results obtained for 25 VMs and *Chart no. 2* plots the data obtained in the form of column chart for both current active VM algorithm and proposed VM Load balancer algorithm.

TABLE 3: COMPARISON RESULTS FOR 25 VMS IN DC

VM Id #	*# of allocations By Active VM Load Balancer*	*# of allocations By Proposed VM Load Balancer*
0	42374	3680
1	17324	3705
2	10477	3703
3	6783	3704
4	4683	3705
5	3249	3703
6	2229	3705
7	1613	3702
8	1191	3702
9	835	3702
10	609	3699
11	424	3703
12	299	3703
13	195	3700
14	120	3703
15	59	3702
16	36	3702
17	17	3703
18	8	3703
19	3	3700
20	3	3703
21	5	3702
22	1	3702
23	1	3702
24	2	3702

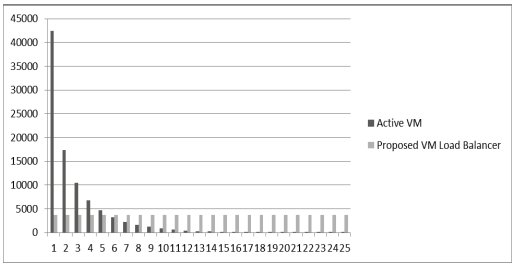

CHART 2: COMPARISON RESULTS FOR 25 VMS IN DC

With the above results it is evident that current active VM load balancer over-allocates the initial VMs and under-allocates the later ones. The results also suggests that proposed VM load balancer allocated the requests to VMs evenly by overcoming the limitation of active VM load balancer.

VI. CONCLUSION

In this paper, we proposed an efficient VM load balancing algorithm that distributes the load evenly across all VMs in the data center even when the incoming request frequency is high during peak hours. It is observed from the experimental results that current active VM load balancer heavily loads the initial VMs where-as the proposed VM Load balancer evenly distributes the incoming requests to all VMs. Our proposed algorithm solves the problem with current active VM algorithm of non-uniform allocation of requests to VMs by using a reservation table between the phase of selection and allocation of VMs. Further, we are planning to investigate the feasibility of applying the technique across all data centers to allocate the load uniformly to all VMs across data centers situated at different geographical locations.

REFERENCES

[1] Ibrahim Takouna, Wesam Dawoud, and Christoph Meinel, "Analysis and Simulation of HPC Applications in Virtualized Data Centers", IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference, 2012.

[2] Jinhua Hu, Jinhua Hu, Guofei Sun, Tianhai Zhao,"A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment", 3rd International Symposium on Parallel Architectures, Algorithms and Programming.2010.

[3] B.Wickremasinghe, R.N.Calheiros,R.Buyya, "Cloudanalyst: A cloudsim based visual modeller for analysing cloud computing" in: Proceedings of the 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia,, 2010.

[4] Jing Yao, Ju-hou He,"Load Balancing Strategy of Cloud Computing based on Artificial Bee Algorithm" in 8th International Conference on Computing Technology and Information Management (ICCM), 2012.

[5] Ren, X., R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE, pp: 220-224, September 2011.

[6] Lee, R. and B. Jeng,"Load-balancing tactics in cloud," in proc.International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, pp:447-454, October 2011.

[7] Wang, S-C., K-Q. Yan, W-P. Liao and S-S. Wang,"Towards a load balancing in a three-level cloud computing network," in proc. 3rd International Conference on. Computer Science and Information Technology (ICCSIT), IEEE, Vol. 1, pp:108-113, July 2010.

[8] Sang, A., X. Wang, M. Madihian and RD. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in Wireless Networks, Vol. 14, No. 1, pp: 103-120, January 2008.

[9] Ajit M,Vidya G," VM level load balancing in cloud environment", Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013.