ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

A Dynamic and Effective Load Balancing Method using Horizontal Virtual Machine Scaling

[1] Athokpam Bikramjit*, [2] Dr Rio D'souza

[1][2] St Joseph Engineering College, Dakshina Kannada, Karnataka, India
Email: bikramjits@sjec.ac.in

Abstract— ***The infrastructure, software, or platform made available over a network is known as cloud computing. Utilizing virtualization techniques to effectively manage and create virtual machines is the expected norm for cloud computing. Recently, user demand for various services has been expanding dramatically in the field of cloud computing in direct proportion to the number of users. As a result, load balancing has become one of the most sought-after study fields for effectively managing the demand for resources. In this area of work, many algorithms have already been suggested. In this research, we suggest and put into practice two ways for balancing the load of virtual machines.***

Index Terms— ***Cloud Computing, Load Balance, Virtualization, Virtual Machine***

I. INTRODUCTION

With the use of cloud computing technologies, IT infrastructure, platform, and applications are made available as network/internet services that are dynamically expandable and metered. According to Singh et al. (2017a), the cloud primarily offers three service types: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). According to the architectural design, cloud computing is divided into four categories: private, public, hybrid, and community [Singh et al.,(2017b)].

Virtualization is an important aspect in cloud system. A single system can be virtualized into many numbers of virtual systems through virtualization. Virtual Machine (VM) is software implementation of any physical resource [Singh et al,(2017c)]. Hypervisor (low level program or a firmware) is solely responsible for sharing physical instance on cloud among many tenants [Celestia(2016)]. Virtualization can be classified broadly into two types or categories i.e. Container based (at OS level) and Hypervisor based (at hardware level).

Key benefits of virtualization techniques are effective resource management, server merging, energy preservation and fewer space requirements. Instantaneous usage of data in virtualization is a major shortcoming towards data security.

II. LOAD BALANCING ON CLOUD

Architecture of cloud systems is both, distributed and parallel. Services and resources are evenly distributed in the topographical area but due to reasons such as randomness, there are possibilities of uneven distribution in cloud environment. This may lead into disparity i.e. under load and overload on the processor. That is the reason why load balancing comes into the picture and it helps in distributing the resources or services evenly across the cloud.

In VM, main objective is to balance the load and transfer the overloaded machines to free or unload VM's [Singh et al,(2017c)]. In addition to this, there are several other objectives such as optimization, throughput, and response time. Load balancing mainly classifies into two, namely: static and dynamic [Singh et al,(2017a)]. In static, prior knowledge of resources or tasks is known so it is simple to design. These type of approaches can be applied when the processing capacity of the host in the cluster are same. Dynamic algorithms are reliable, flexible and huge number of request can be easily handled. It is self-adaptive and agreements among the cloudlets request which usually create different workloads, which are often not that easy to predict.

III. FUNDAMENTAL LOAD BALANCING ALGORITHMS

The ultimate goal of cloud service provider is designing of efficient load balancing policies and to increase the resource utilization. Using scheduling algorithms in VMs, in virtual environment, we can effectively allocate the resources to the VM's whenever it is required. Typical VM load scheduling operation is to allocate the request to VM whenever the user requests for it. Various VM's have been projected and are discussed here. Following are some of the load balancing algorithms [Nayak and patel (2015)][Kumar and Prashar(2015)].

3.1 Round Robin VM Load Balancing

It is very modest and this load balancing algorithm distributes the newly coming cloudlets to the existing VM in a circular fashion. Drawback is prior information of end user task and resource availability in system as well as instating nous state information is not considered.

3.2 Throttled VM Load Balancing

It is a dynamic method and user requests are sent to DCC

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

(Data Centre Controller). DCC requests VM load balancer in decision making process of allocating suitable virtual machines. It works by keeping VM list along with status. Whenever appropriate VM is discovered, the algorithm responds to cloudlet request and gives allocation request to virtual machine. Else, cloudlet request will be waiting until it gets a suitable VM. One of the best approaches for load balancing is when it preserves the present state of VM's. Main disadvantage of this approach is that DCC would need to agree on same hardware agreement [Nayak and patel (2015)].

3.3 ESCE VM Load Balancing

Equally Spread Current Execution (ESCE) is an active algorithm built on spread spectrum mechanism [Kumar and Prashar(2015)]. It works by equally distributing workload in VM of data center. It maintains a job queue in VM list and allocates if it can find some free VM and if notices many VM's over loaded, it transfers some load to free, idle or less loaded VM's. The major disadvantage here is, it's very high computational overhead.

IV. RELATED WORK

[Somani R, et al,(2015)] performance of hybrid method for balancing workload between VMs, shows better results in comparison to the methods discussed above. [Mohapatra S, et al(2013)] executed some algorithms namely Round Robin, Throttled, ESCE, FCFS and shows that Round Robin is best in terms of performance. S Kumar et al, executed load distributed algorithms between VM's by considering least regularly used VM's and this algorithm beats the algorithms listed above. [Jinhua hu et al(2010)], gives insight of effective utilization of VM's.

V. METHODOLOGY

The major goal of our work is to provide an effective and dynamic load balancing algorithm for the cloud environment that can balance the load by dynamically constructing VMs while taking the arrival of cloudlets into account. At predetermined times, cloudlets are generated and sent to the broker. The Horizontal Virtual Machine Scaling (HVMS) technique is the load balancing algorithm that is suggested. In this method, a scaling mechanism is installed on each newly generated virtual machine and monitored at regular intervals to determine if it is overloaded or not. A scaling (up/down) choice is then made based on that information. In this case, the data broker is in charge of scaling, and the datacenter broker establishes the amount of time the broker must wait before destroying or creating VMs. If a time is not given, the broker simply destroys VMs once all cloudlets that are currently operating have completed their operations or if no cloudlet is waiting to be created.

5.1 Pseudo Code

```
Step 1: Horizontal scaling mechanism dynamically creates VM's according to the arrival of cloudlets
Step 2: If number of cloudlets is more in existing VM's, create new VM's to balance the load
Step 3: Scaling is performed by creating or destroying VM's, whenever necessary
Step 4: When VM's become under loaded, they are destroyed after the complete execution of cloudlets present in it
Step 5: Create VM's, when the load balancer detects the current brokers VM's are overloaded.
Step 6: Repeat Step 3, if the load in the system is not balanced
```

5.2 Algorithm

```
Step 1: Create Class for HorizontalVmScaling
SET interval in which the Datacenter will schedule events
SET interval to request the creation of new Cloudlets
SET HOSTS Value
SET HOST_PES
SET VMS
SET CLOUDLETS
Step 2: Create DatacenterBroker with HostList, vmList and Cloudlets
Create CLOUDLETS_LENGHT of variable length in the form of list Assign random value to CLOUDLETS_LENGHT
Step 3: Build simulation scenario and start simulation
Remove the seed parameter to get a dynamic one, based on current computer time
Define the Vm Destruction Delay Function
Create new Cloudlets at every second, up to some a certain time interval
Call methods every time the simulation clock advances
Record the information about the OnClockTick event
Step 4: Create a Datacenter and its Hosts with scheduling intervals
Step 5: Create a list of initial VM's in which each VM is able to scale horizontally when it is overloaded
Notify number of VM's to create
Return the list of scalable VM's
Step 6: Create parameter of VM for which the Horizontal Scaling will be created
Check if VM is overloaded or not, based on upper CPU utilization threshold with reference value. Take required actions
```

5.3 Implementation

```
Public class HorizontalVmScalingSimple extends
VmScalingAbstract implements HorizontalVmScaling {
    private Supplier<Vm> vmSupplier;
    private long cloudletCreationRequests;
    private Predicate<Vm> overloadPredicate;
    public HorizontalVmScalingSimple(){
```

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 11, November 2022

```
    super();
    this.overloadPredicate = FALSE_PREDICATE;
    this.vmSupplier = () -> Vm.NULL;
    }
    public Supplier<Vm> getVmSupplier() {
    return vmSupplier;
    }
    public final HorizontalVmScaling setVmSupplier(final
Supplier<Vm> supplier) {
    Objects.requireNonNull(supplier);
    this.vmSupplier = supplier;
    return this;
    }
    public Predicate<Vm> getOverloadPredicate() {
    return overloadPredicate;
    }
    public VmScaling setOverloadPredicate(final
Predicate<Vm> predicate) {
    Objects.requireNonNull(predicate);
    this.overloadPredicate = predicate;
    return this;
    }
    protected boolean requestUpScaling(final double time) {
    if(!haveNewCloudletsArrived()){
    return false;
    }
    final double vmCpuUsagePercent =
getVm().getCpuPercentUsage() * 100;
    final Vm newVm = getVmSupplier().get();
    Log.printFormattedLine(
    "\t%.2f: %s%d: Requesting creation of Vm %d to receive
new Cloudlets in order to balance load of Vm %d. Vm %d
CPU usage is %.2f%%",
    time, getClass().getSimpleName(), getVm().getId(),
newVm.getId(), getVm().getId(), getVm().getId(),
vmCpuUsagePercent);
    getVm().getBroker().submitVm(newVm);
    loudletCreationRequests =
getVm().getBroker().getCloudletCreatedList().size();
    return true;
    }
    private boolean haveNewCloudletsArrived(){
    return
getVm().getBroker().getCloudletCreatedList().size() >
cloudletCreationRequests;
    }
    public final boolean
requestUpScalingIfPredicateMatches(final
VmHostEventInfo evt) {
    if(!isTimeToCheckPredicate(evt.getTime())) {
    return false;
    }
    setLastProcessingTime(evt.getTime());
    return overloadPredicate.test(getVm()) &&
requestUpScaling(evt.getTime());
    }
    }
```

VI. RESULTS AND DISCUSSION

Results

```
Cloudlet|    Status |DC|Host| Host PEs |VM|    VM PEs|CloudletLen|CloudletPEs|StartTime|FinishTime|ExecTime
      ID|           |ID |  ID |CPU cores| ID |CPU cores|         MI|  CPU cores|  Seconds|    Seconds|Seconds
-----------------------------------------------------------------------------------------------------------
       0 |SUCCESS |  1|   0|        32|   0|        2|      2000|          2|        0|         5|      6
       4 |SUCCESS |  1|   0|        32|   0|        2|     10000|          2|        0|        14|     15
       8 |SUCCESS |  1|   0|        32|   0|        2|      2000|          2|        2|         7|      6
       1 |SUCCESS |  1|   1|        32|   1|        2|     10000|          2|        0|        22|     23
       5 |SUCCESS |  1|   1|        32|   1|        2|     20000|          2|        0|        32|     33
       9 |SUCCESS |  1|   1|        32|   1|        2|      2000|          2|        2|         8|      7
       2 |SUCCESS |  1|   2|        32|   2|        2|     20000|          2|        0|        36|     37
       6 |SUCCESS |  1|   2|        32|   2|        2|     16000|          2|        2|        34|     33
       3 |SUCCESS |  1|   3|        32|   3|        2|      4000|          2|        0|         6|      7
       7 |SUCCESS |  1|   3|        32|   3|        2|      2000|          2|        2|         6|      5
-----------------------------------------------------------------------------------------------------------
```

Fig 1: Simulation test results

The above result shows the details of the simulation and this result is compared with the existing algorithms in terms of execution time, response time and waiting time.

Table 1: Comparison among load balancing algorithms

	RR	Throttled	ESCE	HVMS
Avg. Execution Time	55.25	52.23	49.21	17.2
Avg. Response Time	0.68	0.45	0.44	0.4
Avg. Waiting Time	0.72	0.63	0.49	0.4

The above comparison shows that HVMS gives the best time with respect to all the comparisons that has been done in this paper. Hence, through this algorithm, we can efficiently execute the request from the client and can even balance the load in cloud computing environment.

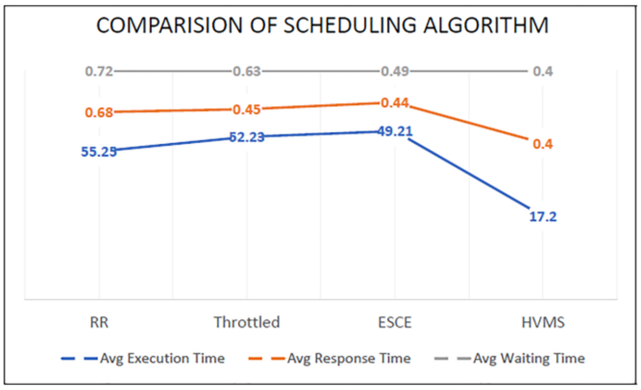

Fig 2: Graph showing the comparisons of various load scheduling algorithms in terms of response, waiting and execution time

VII. CONCLUSION AND FUTURE WORK

The current cloud system mainly accepts IT-based service. However, there are various concerns which are not completely addressed so far such as load balancing, real time scheduling, VM migrations and many more. Stability of the system through load balancing is one of the important factors in cloud environments, which deals with scalability of workload without compromising the efficiency. Our proposed algorithm overcomes all these issues and even takes care of legitimately distributing the resources across the cloud environment. Our algorithm has been implemented with Cloudsim and the outcome of our algorithm outperforms the well-known algorithms like Round Robin, Throttled and Equally Spread Current Execution with respect to response, waiting and execution time. Only average waiting time is shows similarity in comparison with Equally Spread Current Execution. The application of our proposed algorithm can be achieved in real environment, in future.

REFERENCES

[1] Singh, Athokpam Bikramjit, et al. "A comparative study of various scheduling algorithms in cloud computing." *American Journal of Intelligent Systems* 7.3 (2017): 68-72.

[2] Singh, Athokpam Bikramjit, et al. "Survey on various load balancing techniques in cloud computing." *Adv. Computing"* 7.2 (2017): 28-34.

[3] Singh A B, et al (2017c) "A comprehensive investigation of network virtualization", *IJLTET, Special Issue, SACAIM*-2017, 622-628

[4] J. Celestia (2016) "Cloud Computing Infrastructure," 2016. [Online]. Available: http://www.tutorialspoint.com/cloud_computing/cloud_computing_infrastructure.htm. [Accessed 2016].

[5] Nayak, Slesha, and P. Patel. "Analytical Study for Throttled and proposed Throttled algorithm for load balancing in Cloud Computing using Cloud Analyst." *International Journal of Science Technology & Engineering* 1.12 (2015): 90-100.

[6] Ray, Soumya, and Ajanta De Sarkar. "Execution analysis of load balancing algorithms in cloud computing environment." *International Journal on Cloud Computing: Services and Architecture (IJCCSA)* 2.5 (2012): 1-13.

[7] Somani, Rajkumar, and Jyotsana Ojha. "A hybrid approach for VM load balancing in cloud using cloudsim." *International Journal of Science, Engineering and Technology Research (IJSETR)* 3.6 (2014): 1734-1739.

[8] Mohapatra, Subasish, et al. "A comparison of four popular heuristics for load balancing of virtual machines in cloud computing." *International Journal of Computer Applications* 68.6 (2013).

[9] Hu, Jinhua, et al. "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment." 2010 3rd International symposium on parallel architectures, algorithms and programming. IEEE, 2010.