Research Article

iMedPub Journals
http://www.imedpub.com

American Journal of Computer Science and Engineering Survey

2020
Vol. 8 No.3:11

A Simulative Study on the Performance of Load Balancing Techniques over Varying Cloud Infrastructure Using Cloudsim

Rafat Khan*

Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh

***Corresponding author:** Rafat Khan, Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh, E-mail: bit0421@iit.du.ac.bd

Citation: Khan R (2020) A Simulative Study on the Performance of Load Balancing Techniques over Varying Cloud Infrastructure Using CloudSim. Am J Comput Sci Eng Surv Vol. 8 No. 3: 11.

Abstract

Cloud computing is a recent evolution in computer technology. A cloud server has to deal with a lot of workload. So, the process of distributing load over the virtual machines or among the servers in a distributed cloud system is very important to get better performance. Before launching a cloud based application or a new load balancing policy in the cloud server, it is very difficult to know whether the performance will be good or bad. Because testing an application or load balancing policy's performance in Cloud for different workload in a repeatable manner under varying system and user configurations and requirements are very complex and expensive to achieve. So simulating the performance of an application or a load balancing policy is a good alternate. This paper is about simulating some existing load balancing approaches as well as a proposed improved load balancing approach and the comparison of the performance using CloudSim simulator.

Keywords: Cloud computing; Workload; Load balancing; Simulation; CloudSim

Received: November 06, 2020; **Accepted:** November 20, 2020; **Published:** November 27, 2020

Introduction

Cloud computing [1] is defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud computing is comparable to grid computing, a type of computing where unused processing cycles of all computers in a network are capable to solve problems too intensive for any stand-alone machine. The application services hosted under Cloud computing model have complex provisioning, structure, configuration, and deployment requirements. Before launching a Cloud based application in the Cloud server, it is very difficult to know whether the performance of the application will be good or not. Most of the Cloud-based application services content delivery or real-time instrumented data processing. Each of these application types has different structure, configuration, and deployment requirements. Measuring the performance of provisioning which includes scheduling and allocation policies in a real Cloud computing environment for different application models under different conditions is extremely challenging because clouds demonstrate varying demands, supply patterns, system sizes, and resources; moreover users have heterogeneous, dynamic, and competing requirements and applications have varying performance, workload, and dynamic scaling requirements. Cloud simulation up the possibility of evaluating the hypothesis related to Cloud in a controlled environment where one can easily reproduce results. Simulation-based approaches offer significant benefits to IT companies or anyone who wants to offer his application services through clouds by allowing them to test their services in repeatable and controllable environment and tune the system bottlenecks before deploying on real clouds.

In this research, I compared the performance of some current load balancing approaches using CloudSim simulator. For this I created a Cloud server in a single computing node and setup Cloud environment by creating data centers, hosts, cloudlets and virtual machines. This survey will be the keystone for further research in cloud computing as simulation is the must do staff specially to us, who do not have a cloud server nearby to implement any theoretical assumptions. With the knowledge gained by these, I proposed a new approach for load balancing which I named Vigilant Optimization.

Background

Cloud computing

Cloud computing is a recent advancement where IT infrastructure and applications are provided as 'services' to end-users under a usage-based payment model. The application services hosted under Cloud computing model have complex provisioning, structure, configuration, and deployment requirements. Cloud computing refers to sharing resources, software, and information via a network, in this case the Internet. User can use cloud storage as well as applications, which means users can execute tasks over cloud infrastructure rather than executing tasks in their own machine. For example, Google drive is a pure cloud computing

service, with all the storage found online so it can work with the cloud apps: Google Docs, Google Sheets, and Google Slides. Drive is also available on more than just desktop computers; you can use it on tablets or on smartphones, and there are separate apps for Docs and Sheets, as well. In fact, most of Google's services could be considered cloud computing: Gmail, Google Calendar, Google Maps, and so on.

While users want to execute some tasks over cloud, virtual machines are provided by the service. A virtual machine (VM) is an operating system or application environment that is installed on software which imitates dedicated hardware. The end user has the same experience on a virtual machine as they would have on dedicated hardware. Virtualization saves costs by reducing the need for physical hardware systems. Virtual machines more efficiently use hardware, which lowers the quantities of hardware and associated maintenance costs, and reduces power and cooling demand. They also ease management because virtual hardware does not fail. Virtualization does however require more bandwidth, storage and processing capacity than a traditional server or desktop if the physical hardware is going to host multiple running virtual machines. VMs can easily move, be copied and reassigned between host servers to optimize hardware resource utilization. While user or client submit a task for execution, the system allocate that task to a suitable VM for execution via load balancer. After execution, the result is delivered to the client.

Cloud computing is built for the world of tomorrow, where we each use many different kinds of computing devices: desktop, laptop, cell phone, or tablet. The intention is to make the functionality and data we need always accessible no matter where we are in the world, and no matter what we're using to access the Internet.

Figure 1 shows Cloud infrastructure.

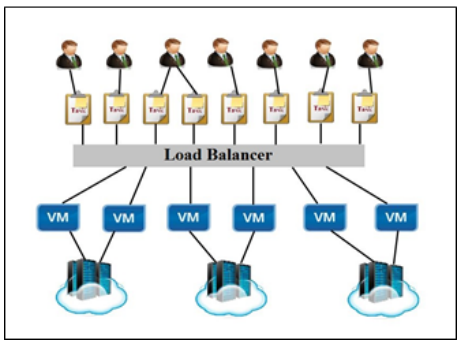

Figure 1: Cloud infrastructure.

Cloud computing

Chronic kidney disease (CKD) has become a global health issue and is an area of concern. It is a condition where kidneys become damaged and cannot filter toxic wastes in the body. The proposed system predominantly focuses on predicting this life-threatening disease Chronic Kidney Disease (CKD) using Classification algorithms (KNN and Naive Bayes). Proposed system is automation for chronic kidney disease prediction using classification techniques and supervised learning algorithms. The data for dataset is obtained from UCI machine learning repository which contains 25 parameters (features) including the class (CKD or NOT CKD) is given in **Table 1**.

DC ID	Mips	RAM	Bandwidth	Storage
1	1000	2048	1000	10000

Table 1: Data center configuration for scenario 1.

Load balancing

Load balancing [2] is dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time and, in general, all users get served faster. Cloud load balancing is the process of distributing workloads across multiple computing resources. Cloud computing brings advantages in cost, flexibility and availability of service users. Those advantages drive the demand for Cloud services. The demand raises technical issues in Service Oriented Architectures, such as high availability and scalability. As a major concern in these issues, load balancing allows cloud computing to scale up to increasing demands by efficiently allocating dynamic local workload evenly across all nodes.

Cloud simulator

Simulation is the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time. Simulation tool is based on the process of modeling a real phenomenon with a set of mathematical formulas. It is, essentially, a program that allows the user to observe an operation through simulation without actually performing that operation. Simulation tools are used widely to design equipment so that the final product will be as close to design specs as possible without being expensive in process modification.

A cloud simulation tool refers to the design and implementation of cloud computing environment simulator, which aims to help developers to model and test policies. As it is very difficult to know whether the performance of the application will be good or not before launching a Cloud based application in the Cloud server, because testing an application's performance in Cloud for different provisioning policies and workload in a repeatable manner under different scenario and user configurations and requirements are very complex and expensive to achieve, here comes the concept of simulation tools. A cloud simulator [3] is a tool or framework where different cloud scenarios can be demonstrated and different load balancing and resource provisioning algorithms can be implemented. Using a cloud simulator, we can test performance of load balancing and resource provisioning algorithms in various scenarios.

Related Work

Buyya et al. [4] identified modeling and simulation techniques of scalable cloud computing environments in CloudSim toolkit. In this paper they have explained how the CloudSim simulator works and provided some simulation results in graph for different

scenarios.

Ray et al. [2] have identified qualitative components for simulation in cloud environment and then based on these components; he has explained execution analysis of load balancing algorithms. This paper presents a review of a few load balancing algorithms or technique in cloud computing. The objective of this paper is to identify qualitative components for simulation in cloud environment and then based on these components, execution analysis of load balancing algorithms are also presented.

Calheiros et al. [5] provide an overview of CloudSim simulator as well as some simulation results. The usefulness of CloudSim is demonstrated by a case study involving dynamic provisioning of application services in the hybrid federated clouds environment. The result of this case study proves that the federated Cloud computing model significantly improves the application QoS requirements under fluctuating resource and service demand patterns.

James et al. [6] have proposed a better allocation policy called weighted active monitoring load balancing by assigning weights to each VM for establishment of an effective load balancing algorithm and how to use Cloud computing resources efficiently for effective and efficient cloud computing.

Sethi et al. [7] have demonstrated 'Firefly', an efficient load balancing in cloud computing using Fuzzy logic. Firefly Load balancing algorithm for the cloud in a partitioned cloud environment to balance the load across the variety of partitions. The central idea of the proposed work is to prepare the nodes capable to conceive the arriving load and attract the workload after analyzing various parameters that characterize the load and the node. A balance factor is calculated, which probabilistically defines the extent of balancing that ensues. Fuzzy logic is applied in accretion, to resolve the time uncertainties following the initial stage of load balancing. By applying this logic it is observed that, a better performance is achieved in terms of computational time, load arrived, task migration and the cost incurred.

Randles et al. [8] have investigated a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. He has also have demonstrated Honeybee Foraging Algorithm which is derived from the behavior of honey bees for finding and reaping food.

Methodology

CloudSim architecture

At the lowest layer is the SimJava discrete event simulation engine [9] that implements the core functionalities required for higher-level simulation frameworks such as queuing and processing of events, creation of system components like services, host, data center, broker, virtual machines, communication between components, and management of the simulation clock. Next follows the libraries implementing the GridSim toolkit [10] that support high level software components for modeling multiple Grid infrastructures and fundamental Grid components such as the resources, data sets, workload traces, and information services. The CloudSim is implemented at the next level by programmatically extending the core functionalities exposed by the GridSim layer. CloudSim provides novel support for modeling and simulation of virtualized Cloud based data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the instantiation and execution of core entities such as VMs, hosts, data centers etc. during the simulation period. This layer is capable of concurrently instantiating and transparently managing a large scale Cloud infrastructure consisting of thousands of system components. The fundamental issues such as provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring are handled by this layer. In CloudSim, a Cloud Information Service (CIS) is responsible for distributing data center characteristics information to the cloud service broker [11].

A Cloud provider, who wants to study the efficacy of different policies in allocating its hosts, would need to implement his strategies at this layer by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer on how a host is allocated to different competing VMs in the Cloud. A Cloud host can be concurrently shared among a number of VMs that execute applications based on user-defined QoS specifications. The top-most layer in the simulation stack is the User Code that exposes configuration related functionalities for hosts as number of machines, their specification, number of tasks and their requirements, VMs, number of users and their application types, and broker scheduling policies [12].

Communication between data center and broker in CloudSim is shown in **Figure 2**.

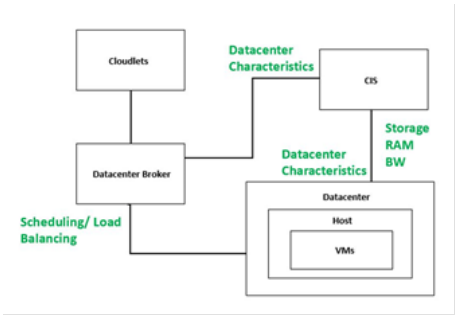

Figure 2: Communication between data center and broker in CloudSim.

Creating custom scenario using cloudsim

The cloud system is practically a collection of Data Centers [13]. This Data Centers can be all together or globally distributed. Each data center has its own Virtual Machines to execute tasks provided by clients.

To implement a scenario for performance analysis we have to create some entities. They are:

- Data Center
- Host

- Virtual Machine
- Cloudlet
- Scheduling and Load Balancing Policy

Creating a data center: To create a Data Center CloudSim class Datacenter class was used which contain the methods for creating a Data Center with certain characteristics described in the Datacenter Characteristics class along with a FCFS VM allocation policy.

datacenter=new Datacenter(name, characteristics, new VmAllocation Policy Simple(hostList), storageList);

Here, characteristics are

characteristics=new Datacenter Characteristics(arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

Where arch is the system architecture, os is the operating system, vmm is the name of VM. Other characteristics are used for simulation of distributed cloud system and cost evaluation.

Creating a host: Host executes actions related to management of virtual machines (creation and destruction). A host has a defined policy for provisioning memory and bw, as well as an allocation policy for processing element's to virtual machines. A host is associated to a datacenter. It can host virtual machines. One Data Center can have multiple hosts. To create a Host, CloudSim class host was used which contain two basic VM allocation policy (Time shared and Space shared), and a basic provisioning policy for ram and bandwidth.

hostList.add(new Host) hostId, new RamProvisionerSimple(ram), new BwProvisionerSimple(bw), storage, peList, new VmSchedulerTimeShared(peList)

Here, peList is the list of processing elements. That is a single core processor has one PE while duel core has two. Am certain amount of RAM and Bandwidth need to allocate for the host.

Creating Virtual Machine (VM): VM runs inside a Host, sharing hostList with other VMs. It processes cloudlets. This processing happens according to a policy, defined by the CloudletScheduler. Each VM has an owner, which can submit cloudlets to the VM to be executed. Each VM has its own storage capacity, mips, ram etc. To create a VM, core CloudSim class is Vm which uses the scheduler policy from class CloudletScheduler.

Vm vm=new Vm(vmid, brokerId, mips, pesNumber, ram, bw, vmm, new CloudletSchedulerSpaceShared());

Here broker is a method from the class DatacenterBroker. It represents a broker acting on behalf of a user. It hides VM management, as vm creation, submission of cloudlets to this VMs and destruction of VMs.

Creating cloudlet: Cloudlets are the tasks assigned to VMs. It stores, despite all the information encapsulated in the Cloudlet, the ID of the VM running it. Workloads are assigned to VMs by creating cloudlets and assigning them to VMs.

Cloudlet cloudlet=new Cloudlet (id, length, pesNumber, fileSize, outputSize, utilizationModel);

Here length is the size of workload in MB. Utilization model refers to how it will utilize the resources it has for execution.

Binding cloudlets to VMs: We can manually bind cloudlets to VMs using their id.

broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());

Otherwise we can bind cloudlets to VMs using some sort of scheduling policy.

public void scheduleTaskstoVms() {int reqTasks=cloudletList.size(); int reqVms=vmList.size(); for(int i=0;i<reqTasks;i++) {bindCloudletToVm(i, (i%reqVms));}}

After creating a scenario, different load balancing policies can be implemented and the execution time can be found by running the simulation.

Proposal for a new load balancing approach

Vigilant optimization: Vigilant optimization is a load balancing approach which uses the advantage of MinMin and Round Robin approach. The basic problem of Round Robin approach is that it does not calculate the processing power of VMs. So if there are two VMs with different processing power, it will allocate equal task to each VM. For this reason, overall execution time becomes high. To solve this problem, Round Robin has been modified and Weighted Round Robin was proposed. In this approach, the processing power or the weight of the VMs has been noticed. In this approach, task has been allocated to the most powerful VM first. But the problem still remain is that, in a cloud scenario different task arrives all the time. These tasks have different size, complexity and execution time. In Weighted Round Robin, task size, complexity and execution time is not noticed. That's why in real time execution, this technique does not provide better performance.

On the other hand, MinMin load balancing approach only notices the task property. It finds the task with minimum execution time and put it first on the execution list. The problem with this approach is this; the VM which is being allocated with a small task may have big execution power. This can lead to bad resource utilization. That's why this approach might provide very good performance in small scenario, but will not be suitable for real life scenarios.

Now, vigilant optimization uses the positive factors of both Round Robin and MinMin approaches. In this approach, VMs are sorted according to its processing power. The tasks which need to be executed also sorted according to its complexity, size and estimated execution time in a task queue. While executing, small tasks are assigned to the low execution power holder VMs. There is a count that indicates how much task a VM is executing. Using this count the vigilant approach makes sure that no VM will be overloaded and the balance of load for each VM is maintained. While executing tasks in a VM using Round Robin manner using time slice, this approach also provides a way to slicing time frame

to provide better performance.

Algorithm:

Step 1: The scheduler maintains a queue of ready Tasks ordered in descending order by task size and a list of Virtual Machines ordered in descending order by processing power.

Step 2: Tasks will be assign to the VM's in a Round Robin manner. That is, first task will be assigned to the first VM. Second task to the second VM and so on.

for(int i=0;i<reqTasks;i++){bindTaskToVm(i, (i%reqVms));}

Step 3: There will be a Task Count that shows the number of tasks that are executing in a VM.

Step 4: There will be a State for each VM, which shows if the VM is busy or available.

if(vm.State == available){// bind task to vm count++}

Step 5: There will be a threshold value that define maximum number of tasks that can be executed at a time in a VM. If number of tasks is equal throughput, that VM's State will be busy and new task will be assigned to it unless the status becomes available.

if(vm.Count == threshold){// set vm state busy}

Else // set vm state available

Step 6: If any task's execution complete in a VM, Task Count will decrease and the state will change.

If(task.State == Complete){count--}

Step 7: If a new task arrive, the scheduler will assign that task to the VM, which State is available and has the minimum Task Count.

For(vm queue: vm){// find the vm with minimum task count// assign task to the vm}

Step 8: If multiple VM has the same Task Count, task will be allocated to the first VM.

Step 9: If multiple tasks arrive, the first task of the task queue will be assign to the VM first.

Step 10: If waiting time for a task becomes greater than a waiting time threshold value, that task will move on the first of the task queue.

Results and Discussion

Comparison between different load balancing approach

Scenario 1: VM Configuration for Scenario 1 is shown in **Table 2**.

VM ID	Mips	RAM	Bandwidth
1	100	512	1000
2	200	1024	1000

Table 2: Virtual machine configuration for scenario 1.

Figure 3 shows the Performance graph for scenario 1.

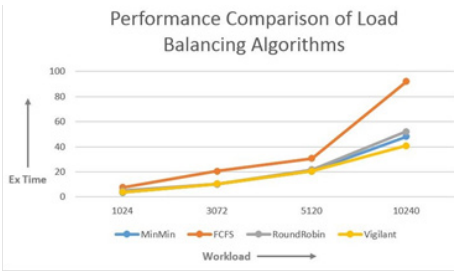

Figure 3: Performance graph for scenario 1.

DC ID	Mips	RAM	Bandwidth	Storage
1	10000	8192	10000	1000000

Table 3: Data center configuration for scenario 2.

Number of VM: 05

VM Configuration for scenario 2 is shown in **Table 4**.

VM ID	Mips	RAM	Bandwidth
1	100	512	1000
2	200	1024	1000
3	100	512	1000
4	400	2048	1000
5	300	1024	1000

Table 4: Virtual machine configuration for scenario 2.

Figure 4 shows the Performance graph for scenario 2.

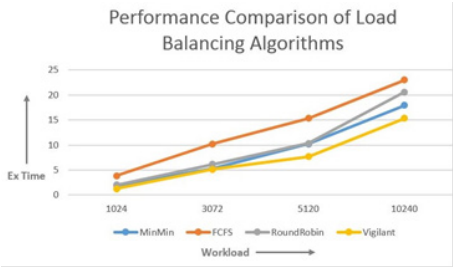

Figure 4: Performance graph for scenario 2.

DC ID	Mips	RAM	Bandwidth	Storage
1	100000	16384	10000	1000000

Table 5: Data center configuration for scenario 3.

Number of VM: 10

VM Configuration for scenario 3 is shown in **Table 6**.

VM ID	Mips	RAM	Bandwidth
1	100	512	1000
2	200	1024	1000
3	100	512	1000
4	400	2048	1000
5	300	1024	1000

6	200	1024	1000
7	500	2048	1000
8	200	512	1000
9	400	1024	1000
10	300	512	1000

Table 6: Virtual machine configuration for scenario 3.

Figure 5 shows the Performance graph for scenario 3.

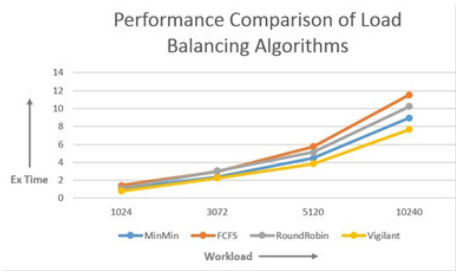

Figure 5: Performance graph for scenario 3.

Conclusion

This research paper of mine is focused with these topics to learn simulation technique and compare the performance of different load balancing techniques to find out the best one with the intension of developing a new load balancing approach. The simulative performance comparison between some of the existing approaches and the proposed approach is shown. From the simulative results, we can see that the proposed approach's performance is better especially in those scenarios where the numbers of VMs are more and workload is heavy. In future I want to develop this approach for dynamic scenarios. I hope the journey of the approach I proposed in this report will reach far in the future.

References

1. Armbrust M, Fox A, Griffith R, Joseph A, Katz RH, et al. (2009) Above the clouds: A Berkeley view of cloud computing. UC Berkeley reliable adaptive distributed systems laboratory: 7-13.
2. Ray S, Sarkar AD (2012) Execution analysis of load balancing algorithms in cloud computing environment. Int J Cloud Comput: Serv Archit 2: 1-13.
3. Kumar P, Rai AK (2014) An overview and survey of various cloud simulation tools. J Glob Res Comput Sci 5: 24-26.
4. Buyya R, Ranjan R, Calheiros RN (2009) Modeling and simulation of scalable cloud computing environments and the CloudSim Toolkit: Challenges and opportunities. 2009 Int Conf High Perform Comput Simul. 1-11.
5. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) CloudSim: A Toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exper 41: 23-50.
6. James J, Verma B (2012) Efficient VM load balancing algorithm for a cloud computing environment. Int J Comput Sci Eng 4: 1658-1663.
7. Sethi S, Sahu A, Jena SK (2012) Efficient load balancing in cloud computing using fuzzy logic. IOSR J Eng 2: 65-71.
8. Randles M, Lamb D, Taleb-Bendiab A (2010) A comparative study into distributed load balancing algorithms for cloud computing. Proceedings of 24th IEEE Int Conf Adv Inf Netw Appl Workshops, Perth, Australia, 2010.
9. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2010) CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exper, 41: 41: 23-50.
10. CloudSim: A framework for modeling and simulation of cloud computing infrastructures and services. The cloud computing and distributed systems (CLOUDS) laboratory, University of Melbourne, 2009.
11. Malhotra R, Jain P (2013) Study and comparison of various cloud simulators available in the cloud computing. Int J Adv Res Comput Sci Softw Eng, 2013.
12. Jena SR, Padhy S, Garg BK (2014) Performance evaluation of load balancing algorithms on cloud data centers. Int J Sci Eng Res 5: 1137-1145.
13. Calheiros RN, Ranjan R, De Rose CAF, Rajkumar Buyya R (2009) CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. arXiv.org: 1-9.