Modeling of a Dynamic and Intelligent Simulator at the Infrastructure Level of Cloud Services

Submitted: 26th June 2019; accepted: 25th March 2020

Faouzia Zegrari, Abdellah Idrissi

DOI: 10.14313/JAMRIS/3-2020/36

Abstract: *Cloud environments made up of a large number of compute and storage servers provide on-demand services in a usage-based consumption model (pay-as-you-go). Load balancing is one of the major problems in the cloud. Indeed, the dynamics of demand requirements and QoS, as well as the variability of cloud resources and its provisioning models make difficult the operation of performance evaluation of the system. To face this issue and to ensure the viability of cloud computing, IT resources must be managed effectively by a dynamic monitoring of the current workload of virtual machines (VMs). In this study, we propose the design of a cloud services simulation tool at the infrastructure level based on cloud computing simulation platform named CloudSim. It allows real-time monitoring of a load of each VM in terms of CPU utilization, memory utilization and bandwidth utilization ratio. The result of this case study can be useful for carry out dynamic environment simulations for VMs monitoring and fast decision making that can be used in load balancing mechanisms.*

Keywords: *Load balancing, Cloud computing, Resource utilization, Dynamic environment simulation*

1. Introduction

The explosion of numerical data and the need for high availability of service are the critical factors in the emergence of the concept of cloud computing. The cloud model is a new paradigm in IT aiming at modernizing the Internet. It allows access to a pool of computing resources that can be allocated and released on demand with minimal interaction with the service provider [1]. The cloud provides hosted services in Datacenters of high performance, which are categorized according to the technical layer provided. There are three usage models at the disposal [2]: IaaS, PaaS and SaaS. [3] The IaaS layer corresponds to the architecture and IT infrastructure part where the provider hosts virtualized resources like servers, data storage, network and virtualization. The variability of cloud resources, the diversity of requirements for applications in terms of performance and workload are among the most important problems in the field of research. The dynamic of demands can be managed by dynamically provisioning cloud resources capacities.

Several researchers, who integrate state of load control techniques of resources in a cloud datacenter, have proposed various dynamic scheduling algorithms. The simulation of these algorithms in real time for the evaluation of the performances of various metrics is a very difficult job to realize. In our study, we propose a dynamic simulator based on CloudSim Framework [4, 5, 6]. Let us recall that CloudSim allows a modeling and a correct simulation of the infrastructure and application services of the cloud computing. It is an open source framework developed in Java. The communication between the various CloudSim entities, such as Datacenters, hosts, virtual machines VMs and cloudlets, occurs using events with static triggering. Indeed, after launching the simulation, CloudSim does not make it possible to interact with the system or to add tasks dynamically. Tasks are assigned to VMs through the Datacenter broker before the beginning of the execution. That is explained by the fact why the Cloudsim clock is based on events for its operation. Each execution, resumption or stop execution is defined as being an event. At the beginning of its execution, it predicts, based on the information of simulation, the duration of simulation as well as the successions of the events, which will take place. We are not thus able to obtain a load of VMs in real time. There exists in the literature some tools, that measure the use of resources, but these tools are not easy to adapt them in certain contexts.

The rest of this paper is organized as follows: the next section presents the analysis of some frameworks used in previous work for the measurement of workload and the possibility to adapt them in dynamic environment simulations. In section 3, we propose the architecture and modeling of the simulation. The CloudSimulator simulation scenario is described in Section 4. Section 5 is devoted to evaluating and analyzing the results obtained by CloudSimulator. The conclusion of this paper is presented in Section 6 with perspectives for future related work.

2. Related Work

In our study, several tracks have been explored to analysis some frameworks used in previous work in measuring the load of VMs and their possibility

to adapt them according to our context to perform simulations in real time. The first track was to analyze the CloudSim distributions that were developed in order to find one that would be able to meet our request. Two distributions have attracted our attention: Dynamic CloudSim and Real Time CloudSim [6], but none of these distributions overcomes our problem. As a second solution, the CloudSched [7]. This tool compares existing simulation systems at the application level for the cloud and defines a new lightweight simulation system for dynamic resource scheduling in cloud datacenters. The results are then analyzed and discussed. The goal of CloudSched's analysis is to understand the logic of real-time simulation in order to adapt it to our solution. Since the source code is not available, we were able to access the compiled code and rewrote the source code. After analysis, we noticed that it randomly generates the tasks and executes them without defining a real allocation policy. In addition, the defined classes do not allow us to reuse them in order to implement our solution since several elements are not defined, like policies of supply and allocation at the level of the hosts and VMs. The third possible solution was to install a virtual machine hypervisor, free virtualization software, which will be responsible for the management of VMs. The principle of the hypervisor is to run the operating system in the same kernel and not emulate them, which allows keeping performances close to the native ones. There are several hypervisors and we chose XEN [8] which is (para) virtualization software. This distribution integrates a XenMon monitoring application [9] to monitor a Xen-based environment. It allows running several operating systems on the same hardware resource (PC, Server...), but consumes many resources and today, it is considered hypervisor completely overstepped. Similarly, for the VMWare hypervisor, the physical server on which the VMs are hosted must have a high memory capacity to be able to share it between the VMs and it is a software, which requires the purchase of a license for its use.

Since no explored solution was conclusive, we then opted for the development of a simulator called CloudSimulator, adapted to our need to determine in a more flexible way, performance metrics in terms of use of CPU, ram and bw. Taking into account its success, we decided to develop our simulator around CloudSim to take advantage of the plethora of algorithms and basic models, which it proposes.

The design and implementation of this proposed load measurement tool first required an analysis of the architecture of CloudSim and its various modes of operation and then a stage of understanding its source code in order to be able to redefine some classes and methods of the CloudSim simulator. The use of this tool in load balancing mechanisms represents a definite advance in real-time monitoring of cloud resource status and load balancing decision-making.

3. Architecture and Simulation Modeling

The operation of the simulator proposed is not based on events in order to make the dynamic simulation. Its architecture is described by the whole of the components as illustrated below, in Fig. 1.

Datacenter: it manages and groups hundreds of physical machines connected to each other and characterized by physical resources such as mips, ram, bandwidth and storage, also logical specifications like architecture, hypervisor Vmm, operating system, time zone and pricing at the second of the various resources used making it possible to bill the cost of consumption to its customers. It implements resource allocation policies for hosts and VMs.

Host: This class models a physical node assigned to one or more VMs by a VMs allocation policy named VmScheduller. A host is characterized by CPU speed (mips), storage capacity; one or more physical processing cores Pes, bandwidth and memory capacity.

Datacenter Broker: the broker allows access to Datacenter and plays the role of mediator. It manages the execution of virtual machines and acts on behalf of the cloud provider.

Virtual machine: This class models a VM, which is managed by a host and allows running cloudlets according to the scheduling policy that it uses. The elements characterizing a VM are CPU capacity, memory capacity, the number of CPUs, bandwidth and storage size.

VmAllocationPolicy [4]: This abstract class represents provisioning policies for allocating hosts to virtual machines with the least Pes used.

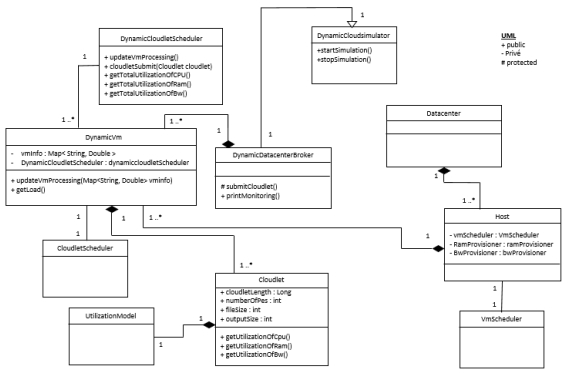

Fig. 1. Class Diagram of the CloudSimulator Simulation System

VmScheduler [10]: This abstract class models how to distribute the available processing capacity of each core of a host between the VMs that host them, according to the chosen allocation policy: time-shared or space shared. For proposed CloudSimulator, we used VmSchedulerTimeShared because it allows Pes sharing.

CloudletScheduler [10]: This class establishes the virtual Pes allocation policy for a given VM to run cloudlets. It implements two strategies: space-shared and time-shared. We have created a third

type to adapt it to the dynamic environment, which we called DynamicCloudletSpaceShared based on the principle of shared space type. Evaluating the effect of these two policies on cloudlet completion time was described in [11] and proved that space shared policy has been better than time-shared policy.

Cloudlet: This class represents the task to run on a VM, defined by a number of instructions and a quantity of data to be transferred, expressed by a size of the input and output files.

As for the allocation of resources, such a scheduling of VMs [10] uses policies at the host level and the VM level: the first policy relates the sharing of host cores between the VMs assigned to it. The second policy allows the VMs to allocate from the available capacity, a quantity of mips to the tasks for their executions. Two modes of execution are suggested: time-shared and space-shared. [4] In the space-shared policy, the cloudlet is executed once the resource is released, the other cloudlets are queued. The completion time of the cloudlet depends on the number of Pes necessary for its execution and the capacity of the assigned processing element. In time-shared, a scheduler is used to allocate resources to a cloudlet during a certain time, once the usage time has elapsed, the cloudlet is queued to execute the next entity from the queue waiting, and however, running the cloudlets happens almost at the same time.

The resource utilization of cpu, ram, and bw by a Cloudlet can be determined according to the usage model chosen. The models proposed by CloudSim are UtilizationModelFull, UtilizationModelNull and UtilizationModelStochastic. For our simulator, we choose the stochastic model, which consists in assigning to each Cloudlet for its execution on a VM, a percentage of random use between 0 and 1. A load is specified as a rate of utilization resource, associated with a duration of use. In our case, since time is variable, we associated it with the length of the task (expressed in Millions of instructions MI) which is a fixed value. When the simulation is launched, we want to collect in real time information on the load rate of each VM.

The utilization rate of the resource is the average value of utilization rates for each cloudlet in the execution list. The total utilization rate on a VM is expressed by the following equation:

$$\overline{Use}_{Vm}\% = \left(\frac{\overline{Use}_{cpu} + \overline{Use}_{ram} + \overline{Use}_{bw}}{3} \right) \times 100 . \quad (1)$$

The execution time of cloudlet in real time is calculated by using the following formula:

$$Execution_{time} = finishTime - startTime \quad (2)$$

where finishTime is the time where this Cloudlet completes and startTime is the start time of executing this Cloudlet.

The communication latency of the node can be easily deduced by:

$$Wait_{time} = startTime - submissionTime \quad (3)$$

where $Wait_{time}$ is the communication latency of the node and submissionTime is Cloudlet's submission time to a Cloud Resource.

The response time is calculated as hereafter:

$$Response_{time} = Execution_{time} + Delay_{time} \quad (4)$$

Knowing that:

$$Delay_{time} = Wait_{time} + Transfer_{time} \quad (5)$$

where DelayTime is the transmission time of the cloudlet and ExchangeData is the size of data exchanged between cloudlets that communicate with this cloudlet.

The workload of each VM is determined from the formula defined in (1) and can be categorized by three states: under loaded, normal and overloaded.

4. Simulation Scenario

- Initialize the simulator clock;
- Create a Datacenter and a Datacenter broker;
- Generate VMs assuming nodes are heterogeneous;
- Generate cloudlets dynamically;
- Submit the VMs and Cloudlets created to the broker;
- Start the simulation;
- Save and display the monitoring of the execution of each VM:
  - The measurement of the load of CPU, ram and bw load as described in (1);
  - The execution time as defined in (2);
  - Simulation start time, Start time of executing cloudlet and time where this Cloudlet completes;
  - The number of cloudlets processed;
- Stop the simulation.

Tab. 1. Host configuration

N° of Host	1
Processing Power (MIPS)	6 000
RAM (MB)	10 000
Bw (Mb/s)	10 000
Storage (MO)	1000000

Tab. 2. Virtual Machines configuration

Virtual machines	VM0	VM1	VM2	VM3
P.Power (MIPS)	250	1 500	1 000	500
RAM (MB)	256	1 024	2 048	1 024

Tab. 3. Cloudlets configuration

Number of Cloudlets	1 100
Length (MI)	40 000 à 140000
File Size (KO)	300
Output Size (KO)	300

5. Experiments and Evaluation of Results

In order to evaluate the behavior and the efficiency of our proposed simulator, we present the experiments and the evaluations of the results of the simulation that we undertook along this study. The simulation was carried out on a PC with an Intel Core i5 CPU 2.40GHz, 32-bit Windows 8.2 Professional Operating System, 4GB Ram, Development Environment: NetBeans IDE 8.2, Cloudsim-3.0.3 Framework and JAVA development language. The experiment consists of measuring the workload on each VM in real time.

5.1. Experimentation

CloudSimulator allows us to perform multiple tasks on multiple Vms. The simulation is carried out with a single host created in a single Datacenter. The goal is to perform an experiment for various quantities of tasks, which we add dynamically, and then measure a load of CPU, memory and bandwidth on each VM.

The scenario begins with a simulation configuration step: We proceed by setting the parameters of the various components of the simulator: a Datacenter made up of 4 virtual machines whose capacities are respectively 250, 1500, 1000 and 500 MIPS as shown on Tab. 2. These VMs are instantiated on a host whose configuration is set on Tab. 1. The tasks to be performed on the VMs are set to an interval [40 000, 140 000] Million instructions, as shown on Tab. 3. In our experiment, we suppose that the tasks communicate with each other by the exchange of a quantity of data, to send or to receive. After this initialization phase, we proceeded to the application of the simulation scenario as described above. The program that we have developed uses a multithreaded environment. Among the classes that have threads: EntitiesGenerator.java, DynamicVm.java DynamicDatacenterBroker.java. EntitiesGenerator.java is a class that allows creating tasks dynamically, to assign them to the appropriate VMs, and to send them to the Datacenter broker. When launching the simulation, the thread of the DynamicDatacenterBroker class performs four operations: the first operation starts the thread of each VM, which triggers the updateVmProcessing(vmInfo) method of the DynamicCloudletScheduler class, for updating the task state (in execution, finished) and updating the processing time. The second operation sends the cloudlets received on the broker to the waiting list of the corresponding VM. The third operation calls on with the method, which will collect on each VM, the information of the load of the CPU, ram and Bw and also the processing time of the accomplished tasks. Lastly, the fourth operation executes the method that ends the simulation. The results of simulation are shown in the curves below.

5.2. Results Evaluation

The real-time monitoring performed throughout the simulation on each VM showed the resource utilization that varies according to the size of running cloudlets expressed in million instructions. The measured values vary in the interval [0, 1] from which we can determine the state of underload and overload.

Fig. 2 illustrates that the CPU load varies depending on the amount of data processed, the sizes of the cloudlets vary from 40 000 to 140 000 Millions of Instructions as shown on Tab. 3, which leads to a variability of the workload of the processor that can range from 0.81% to 98.38%.

Fig. 3 shows that the rate of memory usage varies rapidly depending on the number of tasks. The values of the measurements collected depend on the amount of data in execution and range from 0.89% to 95.23%.

Fig. 4 shows that the different load measurements of the bandwidth collected during a period on each VM as a function of the number of tasks take values ranging from 1.75% to 93.69% representing the occupancy rate of the data transferred on the bandwidth.

Fig. 5 shows the evolution of the execution time as a function of the number of tasks that uses a shared-space policy in a dynamic environment. The required time increases gradually as the tasks are added dynamically. VMs with large capacity are faster and take less time to complete a task, as it is clearly visible on the curves.

5.3. Graphical Representation

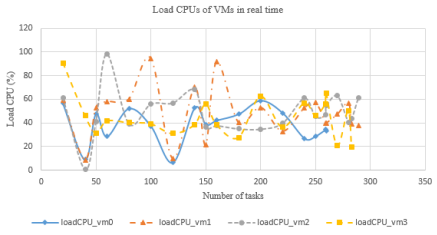

Fig. 2. Real time CPU utilization based on the number of tasks

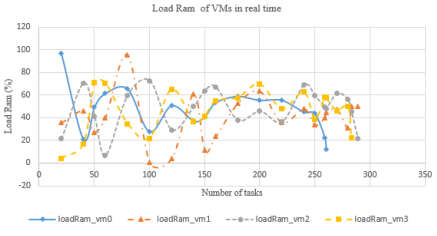

Fig. 3. Real-time memory utilization according to the number of tasks

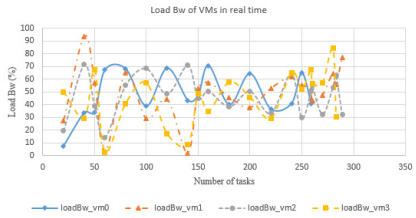

Fig. 4. Real time bandwidth utilization based on the number of tasks

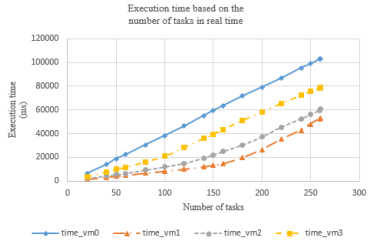

Fig. 5. Evolution of the execution time according to the number of tasks

6. Conclusion

Processing large amounts of data in a high heterogeneity system such as the cloud results in a variability of the workload is not a simple task. In this paper, we developed a simulator that we named CloudSimulator based on CloudSim. It enables dynamic simulation in the cloud environment and provides monitoring of the current workload of VM resources such as CPU, memory and bandwidth. We proceeded by defining a class diagram that is adapted to the dynamic environment. Some classes of CloudSim have been modified by redefining methods and others have been newly created, and a new dynamic resource allocation policy has been implemented using the principle of space-shared policy. The information collected on the current load of resources determines the state of load of each VM that can be used to solve load balancing problems. When an overload is detected on a node, the overloads are transferred to the less loaded nodes. The goal of this study is effectively managed Cloud Computing resources to improve system performance.

In future work, our proposed simulator may well be integrated into load balancing mechanisms and resource allocation algorithms. The information collected on the measurement of the execution time can be used as allocation metric in algorithms for assigning tasks to VMs based on a minimal processing time.

AUTHORS

Faouzia Zegrari* – Intelligent Processing Systems Team (IPSS), Computer Science Laboratory (LRI), Computer Science Department, Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco, email: z.faouzia@gmail.com.

Abdellah Idrissi – Intelligent Processing Systems Team (IPSS), Computer Science Laboratory (LRI), Computer Science Department, Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco, email: idrissi@fsr.ac.ma.

*Corresponding author

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", Technical Report, DOI: 10.6028/NIST.SP.800-145. https://csrc.nist.gov/publications/detail/sp/800-145/final. Accessed on: 2020.12.16.

[2] V. Sangeetha, V. Jaganraja and T. Gnanaprakasam, "A General Study of Homomorphic Encryption Algorithm with Cloud Computing", *Global Journal of Advanced Engineering Technologies and Sciences*, vol. 3, no. 3, 2016.

[3] S. Rajan and A. Jairath, "Cloud Computing: The Fifth Generation of Computing". In: *2011 International Conference on Communication Systems and Network Technologies*, 2011, 665–667, 10.1109/CSNT.2011.143.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, vol. 41, no. 1, 2011, 23–50, 10.1002/spe.995.

[5] T. Goyal, A. Singh and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modeling", *Procedia Engineering*, vol. 38, 2012, 3566–3572, 10.1016/j.proeng.2012.06.412.

[6] "The CLOUDS Lab: Flagship Projects – Gridbus and Cloudbus". www.cloudbus.org/cloudsim/. Accessed on: 2020.12.16.

[7] W. Tian, Y. Zhao, M. Xu, Y. Zhong and X. Sun, "A Toolkit for Modeling and Simulation of Real-Time Virtual Machine Allocation in a Cloud Data Center", *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, 2015, 153–161, 10.1109/TASE.2013.2266338.

[8] "xen [Wiki ubuntu-fr]". https://doc.ubuntu-fr.org/xen. Accessed on: 2020.12.16.

[9] D. Gupta, R. Gardner and L. Cherkasova, "XenMon: QoS Monitoring and Performance Profiling Tool", Technical Report – HPL-2005-187, www.hpl.hp.com/techreports/2005/HPL-2005-187.pdf. Accessed on: 2020.12.16.

[10] R. Kumar and G. Sahoo, "Cloud Computing Simulation Using CloudSim", *International Journal of Engineering Trends and Technology*, vol. 8, no. 2, 2014, 82–86, 10.14445/22315381/IJETT-V8P216.

[11] S. Mehmi, H. K. Verma and A. L. Sangal, "Simulation modeling of cloud computing for smart grid using CloudSim", *Journal of Electrical Systems and Information Technology*, vol. 4, no. 1, 2017, 159–172, 10.1016/j.jesit.2016.10.004.