Cloud computing CPU allocation and scheduling algorithms using cloudsim simulator

Article · August 2016

DOI: 10.11591/ijece.v6i4.10144

2 authors, including:

Hicham Gibet Tani
Abdelmalek Essaâdi University
9 PUBLICATIONS **22** CITATIONS

SEE PROFILE

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 4, August 2016, pp. 1866~1879
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i4.10144

Cloud Computing CPU Allocation and Scheduling Algorithms Using CloudSim Simulator

Gibet Tani Hicham, El Amrani Chaker
Laboratory of Informatics Systems and Telecommunications (LIST), Departement of Computer Engineering
Faculty of Sciences and Technologies, Abdelmalek Essaadi University
Route Ziaten, B.P. 416 Tangier, Morocco

Article Info

Article history:

Received Feb 11, 2016
Revised Apr 19, 2016
Accepted May 3, 2016

Keyword:

Cloud computing
Cloud computing simulation
First come first served
Round robin
Scheduling algorithms

ABSTRACT

Cloud Computing is an emerging computing model, whereas Cloud providers and users are looking forward to profit and enhance their IT exploitation. In this paper, we describe and discuss the Cloud Computing basic compute resources scheduling and allocation algorithms, in addition to the working mechanism. This paper also presents a number of experiments conducted based on CloudSim simulation toolkit in order to assess and evaluate the performance of these scheduling algorithms on Cloud Computing like infrastructure. Furthermore, we introduced and explained the CloudSim simulator design, architecture and proposed two new scheduling algorithms to enhance the existent ones and highlight the weaknesses and/or effectiveness of these algorithms.

Corresponding Author:

Gibet Tani Hicham,
Laboratory of Informatics Systems and Telecommunications (LIST), Department of Computer Engineering,
Faculty of Sciences and Technologies, Abdelmalek Essaadi University,
Route Ziaten, B.P. 416 Tangier, Morocco.
Email: gibet.tani.hicham@gmail.com

1. INTRODUCTION

Nowadays, Cloud Computing is one of the newest paradigms in the Information Technology (IT) world. This new model of computing consists on offering IT resources (Servers, Storage, Network, Applications…) as services, on demand and over a network [1]. Cloud Computing IT model focuses primarily on the flexibility and the celerity of IT resources allocation, which liberate the end users from concerns about the IT infrastructure and location, and all of this presented in a pay-as-you-go manner.

In order to insure that Cloud Computing offerings and characteristics are at the height of expectations in this new model of computing, performance and allocation policies evaluation is necessary before any real world deployment. While using real infrastructures for testing and assessment is expensive and time-consuming, thus it is not always promising to perform experimental, repeatable and scalable investigations on real world class Cloud environments.

A solution to this problem is the use of simulation tools in purpose of evaluating and testing the cloud-computing model in a controlled and scalable environment, therefore generating specific results based on specific measurements. In the same context, CloudSim is an innovative and comprehensive simulation framework that supports modeling, simulation and experimentation of Cloud computing infrastructures and application services [2].

2. CLOUD COMPUTING SIMULATION

A Cloud Computing offer ranges from proposing a specific IT infrastructure to deploying complicated applications and software solutions. By studying the Cloud Computing service delivery model originates the challenge of managing hundreds of thousands of users and applications requests. Therefore, a Cloud Computing provider should consider intelligent infrastructure deployment in order to establish a Cloud Computing offer, which insures transparency, scalability, security and foremost celerity (QoS) [3].

Cloud Computing assessment and evaluation is mandatory for both Cloud providers that are planning a specific service delivery and Cloud users who are intending to shift their IT infrastructure, platform or software into the Cloud (Internet in case of public cloud or Intranet in case of private cloud). Despite the fact that using real infrastructure for testing and evaluating cloud deployment can give the investigators a real world approach to make critical decision about moving forward with this model of computing, in most cases it can be very expensive:

- Infrastructures, platforms and software high costs
- Necessity to test on scalable environments (more infrastructure)
- Management and maintenance expenses

In addition to this critical factor, we can add the time consumption in order to test a specific scenario:

- Infrastructures Installations and configurations
- Repeatable and variable tests
- Debugging and troubleshooting

A more viable alternative is the use of simulation tools. These tools open up the possibility of evaluating the hypothesis (application benchmarking study) in a controlled environment where one can easily reproduce results [2].

There are variety of simulator tools for modelling and simulation of large-scale Cloud computing environments [4] (figure 1). Generally, we can designate between two types of simulators: graphical user interface (GUI) simulators or programming language based simulators (like Java for example).

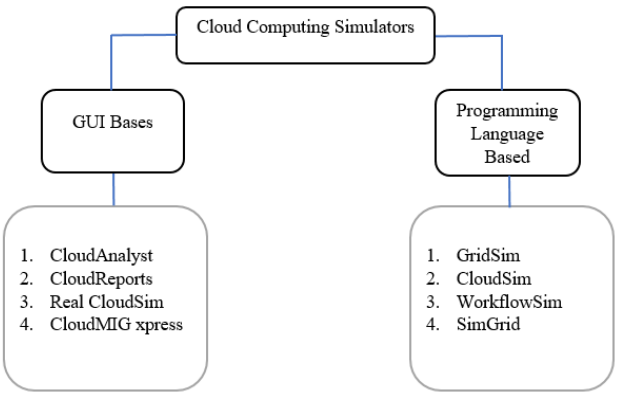

Figure 1. Cloud Computing Simulation frameworks

3. CLOUD COMPUTING SIMULATORS EVALUATION

As described on the previous section, there are a diversity of Cloud Computing simulators, each with specific characteristics and oriented for a specific objective. Choosing the finest Cloud Computing simulator is a challenging mission. To the best of our knowledge, it was not very difficult to spot CloudSim as the core platform for the most used Cloud simulators up to this moment. CloudSim was established as an extension of the GridSim simulator in order to introduce the Cloud Computing virtualization layer that was not present on the original simulator.

CloudSim is a programming language based simulator and even though it does not support a graphical user interface for simulation, it proposes the CloudAnalyst (which is an extension of CloudSim) for investigators who prefers using a user-friendly interface to carry out their researches. CloudSim presents itself to the cloud-computing researchers as a Java based framework that supports the main characteristics of Cloud Computing (IaaS) with virtualization support and task scheduling (PaaS and SaaS) and open up the door for emerging, integrating and testing new algorithms for task scheduling or new characteristics development, which helped on delivering new simulators.

Our choice for CloudSim stems from it openness and clear logic which is deficient on the other simulators specifically with GUI based simulators where we were not able to tackle the Cloud infrastructure layer to better test algorithms related to CPU allocation and introduce new algorithms. CloudSim was the accurate selection for our research, which is focused on evaluating and assessing the CPU scheduling algorithms for resources allocation in order to help Cloud providers and users make precise decision about Cloud Computing model adoption.

4. CLOUDSIM ARCHITECTURE AND DESIGN

CloudSim is a Java application that was founded on GridSim, which is a simulator and a toolkit for modeling and simulation of entities in parallel and distributed computing [5],[6]. CloudSim [5] was designed in a layered architecture as showed on figure 2. At the lowest layer, we find the "SimJava" (Discrete Event Simulation) which implements the core functionalities needed by the higher level of simulation (Data Center, Host, Virtual machine…). Just above the SimJava we find the "GridSim" toolkit for modeling multiple Grid infrastructures, including networks and associated traffic. At the next layer, we find the CloudSim simulation layer, which provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. This layer handles the fundamental issues, such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state [2]. The top layer in the CloudSim simulation toolkit is the "User Code" which is the main interface for simulation specifications and characteristics configuration (number of machines, applications, tasks, users, scheduling policies and their basic structure).

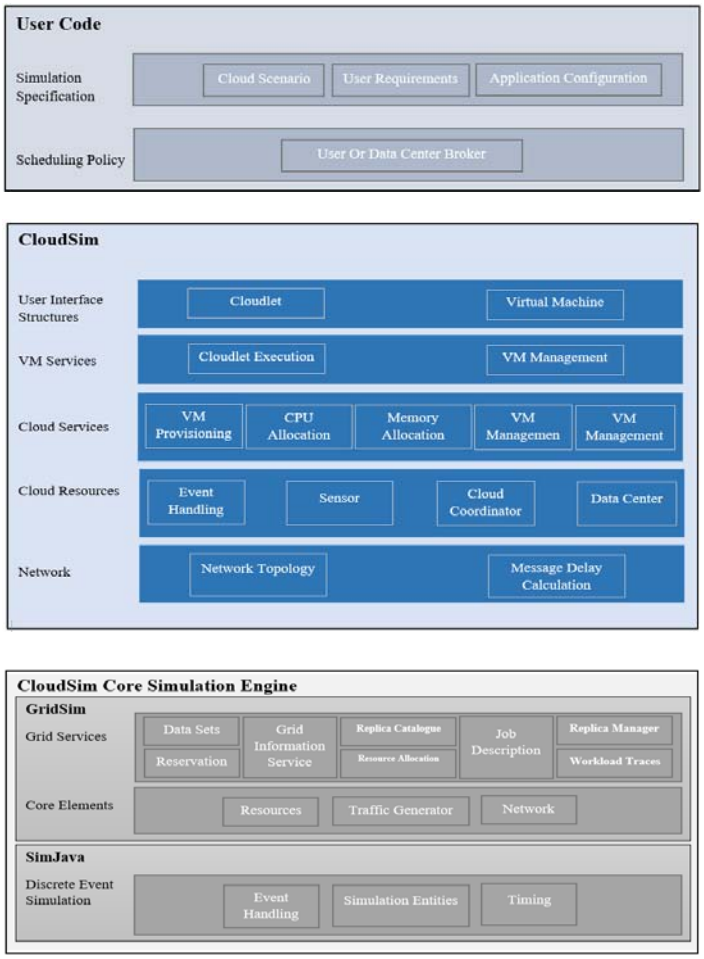

Figure 2. CloudSim Architecture

5. CPU SCHEDULING ALGORITHMS

A cloud provider main intrest is to increase profits by achieving high levels of users' satisfaction, which comes with providing the end user with the best experience. Hence, comes the importance of choosing the finest scheduling algorithms for resources allocation and task scheduling. The key purpose of scheduling algorithms is the appropriate allocation of task or a job to the appropriate resource. Therefore, it goes back always to the period necessary to carry out the execution of a specific task in order to evaluate the quality and performance of the scheduling algorithms [3],[7].

5.1. First Come, First Served (FCFS) Scheduling

The simplest [8]-[10], algorithm for resources scheduling is the "FCFS" algorithm (It is also called FIFO=First In First Out). This algorithm is based on the arrival time of the resource request. To clarify the performance of the "FCFS" algorithm, let take the following Example:

Table 1. First Example Tasks list

Task Order	Task Name	Burst Time
1	Task-1	10
2	Task-2	4
3	Task-3	5
4	Task-4	20

The execution schedule will be as follow:

Table 2. First Example Gantt Chart

Task-1	Task-2	Task-3	Task-4
0	10	14	19 39

The waiting time for each task will be the following:

Table 3. First Example Tasks Waiting Time

Task Order	Task Name	Waiting Time
1	Task-1	0
2	Task-2	10
3	Task-3	14
4	Task-4	19

The average waiting time will be calculated as follow:

$$\mathbf{AwT = (\sum Tn) / NT}$$

Whereas:

- AwT: Average Waiting Time
- Tn: Tasks Waiting time for execution
- NT: Number of tasks

Consequently, the average waiting time will be: **10,75.** Now, let change the task arrival order to match the following:

Table 4. Second Example Tasks list

Task Order	Task Name	Burst Time
1	Task-2	4
2	Task-3	5
3	Task-1	10
4	Task-4	20

The execution schedule will be as follow:

Table 5. Second Example Gantt Chart

Task-2	Task-3	Task-1	Task-4
0	4	9	19 39

The waiting time for each task will be the following:

Table 6. Second Example Tasks Waiting Time

Task Order	Task Name	Waiting Time
1	Task-2	0
2	Task-3	4
3	Task-1	9
4	Task-4	19

The average waiting time will be: **8.** From the two examples above, we can conclude that tasks order changed massively the average waiting time for task execution. Consequently, the "FCFS" presents a weakness on tasks and resources allocation scheduling, because if a heavy task takes on the lead of the queue list, all the other small tasks will have to wait until the execution end for the leading tasks.

This basic scheduling algorithm gave birth to a new algorithm called "**Shorted Job First**" or "**SJF**" which we recommend on this paper in order to enhance the CloudSim algorithm based on the "FCFS". The concept [8],[11], of this new algorithm is the same as for "FCFS", however, the "**SJF**" algorithm introduce a test on the beginning to choose the task with the shortest execution time in order to take the lead of the queue, whereas this ordering of tasks reduced extremely the average waiting time (Second Example above).

5.2. Round Robin (RR) Scheduling

The "Round Robin" algorithm [11]-[13], was designed based on the distribution of the CPU time among the scheduled tasks. On the same context, all the tasks get on a queue list whereas each task get a small unit of CPU time (Quantum, usually 10-100 milliseconds). In order to deepen [10] the understanding of the "RR" algorithm, let take the same example as before:

Table 7. Third Example Tasks List

Task Order	Task Name	Time for Execution
1	Task-1	10
2	Task-2	4
3	Task-3	5
4	Task-4	20

If we consider the "RR" algorithm with a static CPU time quantum of "4", the execution schedule will be as follow:

Table 8. Third Example Gantt Chart

Task-1	Task-2	Task-3	Task-4	Task-1	Task-3	Task-4	Task-1	Task-4	Task-4	Task-4
0	4	8	12	16	20	21	25	27	31	35 39

The waiting time for each task will be the following:

Table 9. Third Example Tasks Waiting Time

Task Order	Task Name	Waiting Time
1	Task-1	0+(16-4)+(25-20) = 17
2	Task-2	4
3	Task-3	8+(20-12) = 16
4	Task-4	12+(21-16)+(27-25) = 19

The average waiting time will be ((17 + 4 + 16 + 19) / 4) = **14.** On CloudSim, The Round Robin based algorithm used for tasks scheduling acts as the following:

- ✓ Sorting the tasks submitted to the VM in ascending way based on the time needed for execution of each task (Burst Time)
- ✓ Extracting a static time Quantum which is calculated based on the number of instructions the processor can execute per second:

TQ = (NP * MIPS) / 1000

Whereas:
TQ => Time Quantum
NP => Number of Processors
MIPS => Million Instruction per Second

- ✓ All tasks are queued on the ready list for execution
- ✓ For each task on the queue list:
  - o The CPU allocates the time quantum for the task execution
  - o If the task is executed, it is sent to the finish queue list
  - o Else the task is sent to the waiting list

Consequently, once the number of a task and amount of instructions get higher, the time quantum becomes smaller and the waiting list gets longer, hence, a high average waiting time.

The Round Robin based algorithm we are proposing on this paper to solve the problem related to smaller time quantum works with a dynamic time quantum [14] and this is as following:

- ✓ Sorting the tasks submitted to the VM in ascending way based on the time needed for execution of each task (Burst Time)
- ✓ Extracting a time Quantum which is calculated based as the Average Execution time for tasks:

TQ = ∑ Tn / NT

Whereas:
TQ => Time Quantum
Tn => Time needed for executing a specific task (Burst Time)
NT => Number of tasks

- ✓ All tasks are queued on the ready list for execution
- ✓ For each task on the queue list
- ✓ If the task burst time is smaller than the time quantum
  - o The Time Quantum is set to the Burst Time of the task
  - o The CPU allocates the time quantum for the task execution
  - o The task is executed and sent to the finish list
- ✓ Else
  - o The CPU allocates the time quantum for the task execution
  - o The task is sent to the waiting list
- ✓ If the waiting list is not empty
  - o Send tasks from waiting list to the ready list
  - o Restart from the beginning

Here is an illustration of the algorithm steps flow:

First, all the processes are sorted in ascending way based on the burst time of tasks (the time needed for execution of each task) and sent to the ready queue list
nt → number of tasks
i → counter
While (RQ != NULL)
TQ = ∑ Tn / NT
// NT = Total number of tasks
// Tn = Time needed for the task execution
//RQ = Ready Queue
//TQ = Time Quantum
Second, Assign TQ to (1 to n) task on the ready queue list
for i = 1 to nt

```
{
        if (Ti < TQ)
                TQ → Ti
                 Ti → TQ
                 Send task to finish list
                  i → i +1
        else
                 Ti → TQ
                 Send task to waiting queue list
        end if
}
end for
// Assign TQ to all the available tasks.
Third, Test if the waiting queue list is empty
if (waiting list != empty)
        Send tasks from waiting list to ready list
        Go to step 1
else
        Finish
end if
```

Let consider the following tasks burst time:

Table 10. Fourth Example Tasks List

Tasks	Arrival Time	Burst Time
T1	0	40
T2	0	90
T3	0	20
T4	0	83

The algorithm we are proposing arranges tasks in ascending way based on their burst time:

Table 11. Tasks List Arrangement

Tasks	Arrival Time	Burst Time
T3	0	20
T1	0	40
T4	0	83
T2	0	90

The time quantum is calculated based on tasks amount and their burst time: **TQ = 58,25.** The execution schedule will be as follow:

Table 12. Fourth Example Gantt Chart

Time Quantum of 58,25			
T3	**T1**	**T4**	**T2**

0 20 60 118,25 176,5

The tasks T3 and T1 are executed (Burst Time is smaller than the Time Quantum). The algorithm sends the remaining tasks to the waiting list:

Table 13. Fourth Example Waiting List

Tasks	Burst Time
T4	24,75
T2	31,75

Dynamically, the time quantum will be recalculated as described before: **TQ = 28,25.** The execution schedule will be as follow:

Table 14. Fourth Example Gantt chart After Time Quatum Recalculation

Time Quantum of 28,25	
T4	T2

176,5 201,25 229,5

The Task T4 is executed (Burst Time is smaller than the Time Quantum). The algorithm sends the remaining task to the waiting list:

Table 15. Fourth Example Waiting List

Tasks	Burst Time
T2	3,5

Dynamically, the time quantum will be set to the burst time of task T2**: TQ = 3,5**. The last execution schedule will be as follow:

Table 16. Fourth Example Last Gantt chart

Time Quantum of 3,5
T2

229,5 233

Let calculate the average waiting time of each task:

Table 17. Fourth Example Tasks Waiting Time

Tasks	Burst Time	Waiting Time
T3	20	0
T1	40	20
T4	83	(20+40)+(176,5– 118,25)= 118,25
T2	90	(20+40+58,25)+(201,25–176,5)= 143

Therefore, The average waiting time for all tasks is: **70,31.** Using the same settings with CloudSim gave us the following results:

Table 18. CloudSim Output

Tasks	Arrival Time	Burst Time	Finish Time
T3	0	20	80
T1	0	40	140
T4	0	83	226
T2	0	90	233

From this output, we extracted the average waiting time as follow:

AwT = (∑ AwTn) / NC
AwTn = FTn – EnT

Whereas:

- AwT: Average Waiting Time
- AwTn: Cloudlet/task (n) Average Waiting Time
- NC: Number of Cloudlets
- EnT: Cloudlet/task (n) Execution Time (Burst Time)

- FTn: Finish Time of Cloudlet/task execution
- NCI: Number of Cloudlet/task Instructions

Consequently, The average waiting time for all tasks is: **111,5.** The "RR" scheduling algorithm introduced a new perception of scheduling that did not exist for the "FCFS" algorithm and which is the concurrent execution of tasks. By comparing the examples stated before, we can see that the average waiting time is higher with the "Round Robin" algorithm used on CloudSim and thus the algorithm we are proposing on this paper is more interesting.

6. CLOUDSIM SCHEDULING POLICIES

The virtualization technology is one of the fundamental concepts of Cloud Computing infrastructures. CloudSim deploy enormously the virtualization technology in order to simulate IaaS and PaaS provisioning and to use it as a base for users' applications execution. On the same perspective comes the challenge of deploying the finest resources allocation and scheduling algorithm. For example, one (1) Data center that consists of one (1) host with two (2) processing units and where the cloud user is trying to instantiate two (2) virtual machines with two (2) processing units each. Logically, there is a separation between the two virtual machines, but in reality, each virtual machine is limited to the processing power offered by the physical host, therefore we cannot instantiate both virtual machine on the same host at the same time without an appropriate scheduling algorithm [2].

In reference to this critical factor, CloudSim proposes two levels of resources allocation policies based on two basic scheduling policies, which are the time-shared and space-shared allocation policies. These allocation policies are implemented during the virtual machines construction and throughout the application execution. The Space-Shared policy and Time-Shared policies are depictions of the "FCFS = First Come First Served" and "RR = Round Robin" algorithms respectively.

In order to illustrate clearly the concept of each allocation policy [8],[15], we propose the following example:

- One (1) data Center with One (1) host
- The host has two (2) processing units
- The user instantiate two (2) virtual machines that require one (1) processing unit each
- The user then try to execute two (2) tasks (Cloudlets) in each virtual machine (each task requires one (1) processing unit for execution)

Figure 3 represents a space-shared policy for both virtual machines and tasks. While each virtual machine requires one (1) processing unit, each virtual machine will reserve one (1) of the two (2) processing units of the host, nevertheless only one (1) task can get executed at a specific time and the second one will wait for the first task to end in order to get executed.

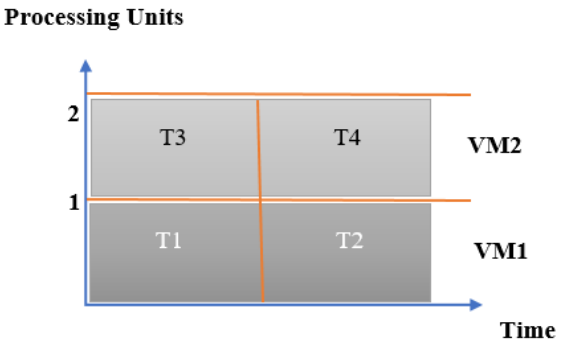

Figure 3. Space Shared Policy for VMs and Tasks

Figure 4 presents a space-shared policy for virtual machines and time-shared policy for tasks. Each virtual machine will reserve one (1) of the two (2) processing units of the host, and while each tasks needs one (1) processing unit to get executed, the policy algorithm will give each task a slice of the processing unit time until both tasks are executed.

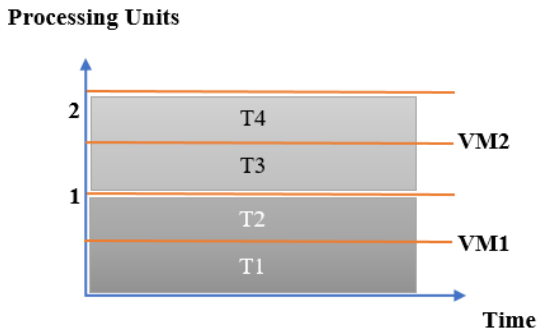

Figure 4. Space Shared Policy for VMs and Time Shared for Tasks

Figure 5 presents a time-shared policy for virtual machines and space-shared policy for tasks. Each virtual machine gets a slice of the processing unit time. The first virtual machine get to hold the first processing unit of the host in order to execute the first task and the same thing goes for the second virtual machine. As a result, both first tasks of both virtual machines get to run simultaneously. The second task of both virtual machines will hold until the first task is executed for both VMs.

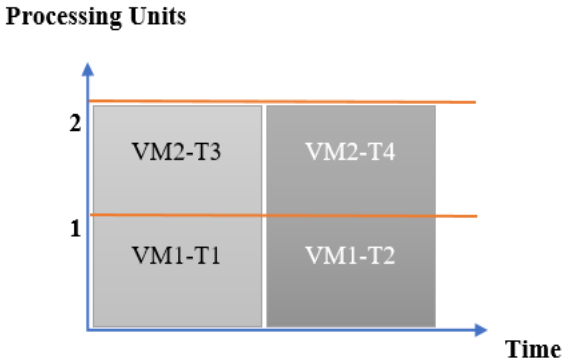

Figure 5. Time Shared Policy for VMs and Space Shared for Tasks

Figure 6 presents a time-shared policy for both virtual machines and tasks. The time of both processing units of the host will be shared simultaneously by the four tasks deployed on the two virtual machines.

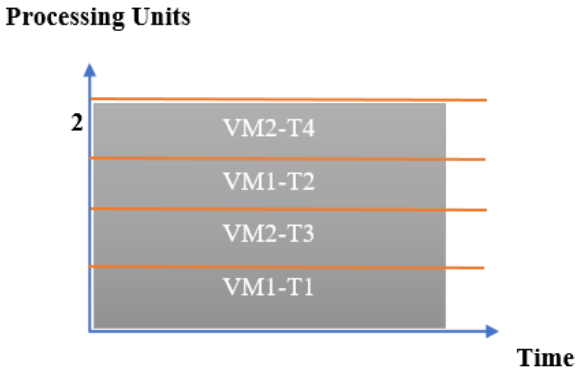

Figure 6. Time Shared Policy for VMs and Tasks

7. EXPERIMENTS AND EVALUATIONS

In this section, we present the experiments conducted to simulate and assess the performance of the allocation policies and algorithms presented in the previous sections. The CloudSim simulation configuration used is the following:

- One (1) data Center with One (1) host
- The host has two (2) processing units:
  - 4000 MIPS (Million Instructions Per Second)
  - 4 GB of RAM
- The user instantiate one (1) virtual machine that requires one (1) processing unit with the following configuration:
  - 1000 MIPS
  - 1 GB of RAM

The evaluation scenario we selected, aims to evaluate the performance of the two algorithms "FCFS" and "RR" through the variation of cloudlets amount (tasks), needed resources for each cloudlet (task) and then calculating the average waiting time for execution and compare them to the proposed algorithms. The evaluation fields were set to the following:

Table 19. Experiments Evaluation Fields

Cloudlet	CloudSim representation of a job or a task
Cloudlet Order	Defines the order of cloudlets submission to execution
Cloudlet Length	The cloudlet instructions amount (In **MI** = **Million Instruction**)
Start Time	Starting time of the cloudlet execution **(Second)**
Finish Time	Finishing time of the cloudlet execution **(Second)**
Execution Time	Time needed for tasks execution completion **(Second)**
Average Waiting Time	The average waiting time between the execution of one cloudlet and the next one on the queue list **(Second)**

7.1. FCFS algorithm (Space Shared policy for Cloudlets)

The first examinations consists of elevating the amount of cloudlets submitted to the Cloud infrastructure and each time we multiply the cloudlets resources needed by two "2". These tests were conducted based on the FCFS with SJF algorithm proposed, which adds a condition to the FCFS algorithm in order to organize the tasks from shorter to greater (the initial order doesn't matter). After that, we used the same inputs for the "FCFS" algorithm used with CloudSim and we reassessed the results again.

Figure 7 represents how the average waiting time for cloudlets execution exploded when we changed the cloudlets arrival order:

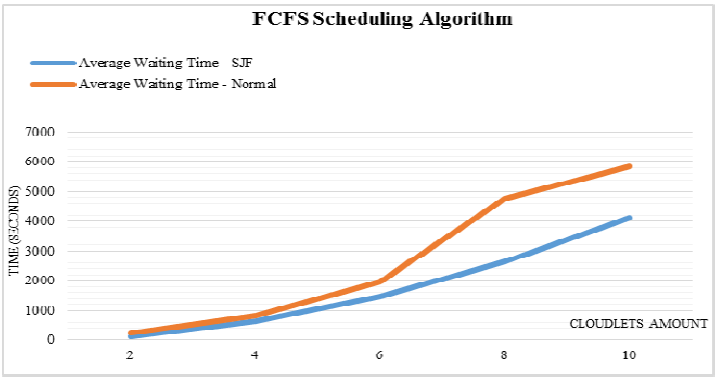

Figure 7. Cloudlets Order effect on Average Waiting Time

7.2. RR algorithm (Time Shared Policy for Cloudlets)

We kept the tests organization as for the "FCFS" tests. The order of cloudlets is not essential when using the CloudSim "RR" algorithm or the one we are proposing because both algorithms sorts the tasks depending on their burst time before submitting them for execution.

Figure 8 represents the average waiting time for cloudlets execution using the CloudSim "RR" algorithm and the proposed RR algorithm (RRABT: Round Robin based on the Average Burst Time of tasks):

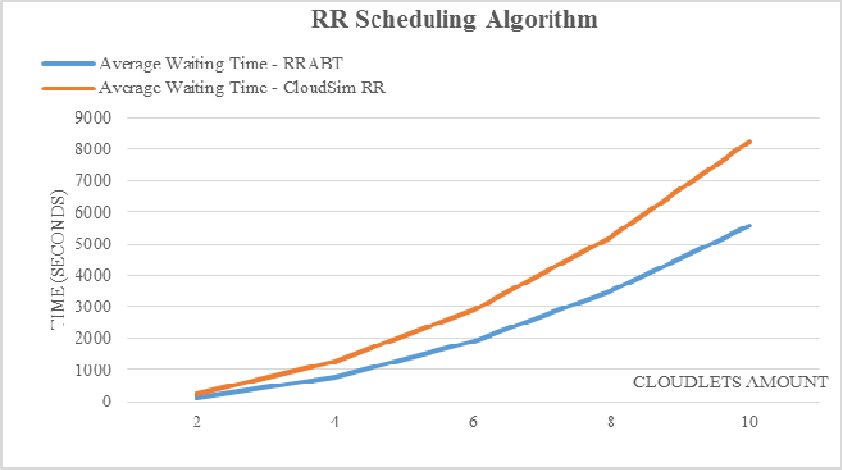

Figure 8. RR algorithm effect on Average Waiting Time

7.3. Results evaluation

Figure 9 is a comparison between the "FCFS" and "RR" algorithms performance in regards of the average waiting time for cloudlets (tasks). The "First Come First Served" with "Shorter Job First" algorithm that we proposed, gave the best performance because of its resources dedication strategy, in this algorithm, the shortest job get to hold the processing unit until the end of execution, which results on executing all short jobs quickly and queuing the heavy tasks at the end. However, the "First Come First Served" algorithm used on CloudSim represents a major flaw related to the cloudlets organization, which affects the average waiting time massively.

On regards to "Round Robin" algorithm, the CloudSim version of the algorithm calculate the time quantum from the processor capacity to execute the tasks instructions. Consequently, if a user submits a thousand task with an inconstant number of instructions on each task, the algorithm instructs the processor to divide its time among all tasks evenly. Using the calculation method for time quantum stated before, the time dedicated for execution of each task gets smaller and smaller in comparison with the tasks number of instructions and therefore queuing the execution of the same task each time this period ends and thus a long time to finish the execution. The Round Robin based algorithm we proposed, calculates the time quantum from the average burst time of all tasks, hence, the algorithm produce an intermediate time quantum that will be recalculated dynamically from tasks that are ready for execution and this is a major enhancement that adapts the CPU resource allocation to the required jobs or tasks. From "Figure 9", we can see how the proposed "Round Robin" algorithm gave a remarkable performance in regards of the average waiting time of tasks and gets on the second position after the "First Come First Served" with "Shorter Job First" Algorithm".

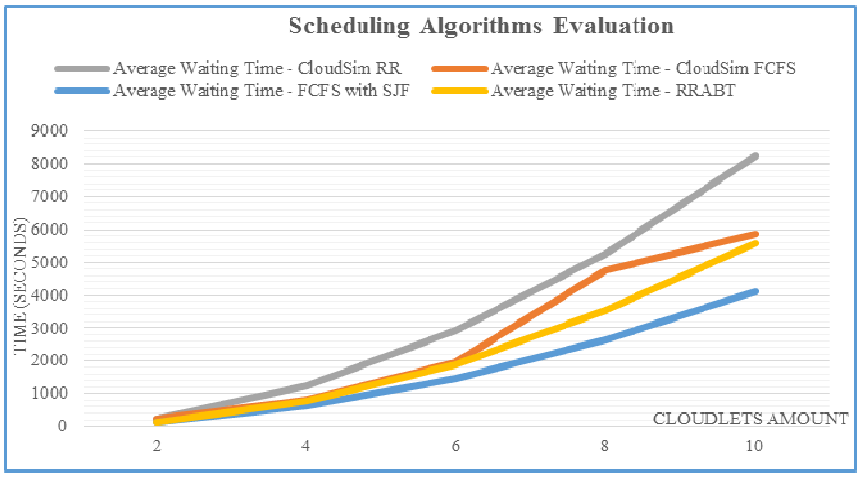

Figure 9. FCFS algorithm vs RR algorithm

8. CONCLUSION AND FUTURE WORK

With the recent great shift into the Cloud new model of computing, Cloud providers are competing to implement and extend their Cloud offer in order to advance their business and rise their profits. On the other hand, Cloud Computing users and potential users are evaluating the most comprehensive Cloud solutions that would help them shrink their costs, increase their IT quality and therefore growing their benefits. To test or develop this new model of computing, both providers and users requires tools that would help them make critical decision about implementing or moving towards an IT Cloud based infrastructure. The majority of researchers and investigators opt for Cloud simulation tools to improve their knowledge of this new paradigm architecture, test the Cloud performance, and design the infrastructure resources scheduling (CPU, storage, network…) algorithms.

In light of this topic, this paper is a brief overview of CloudSim CPU allocation algorithms that aims to assist Cloud providers and users to make intelligent decision regarding Cloud Computing model implementation. By assessing and evaluating the existing Cloud simulators, we selected and based our work on CloudSim toolkit for Cloud Computing simulation. We weighed existing CPU scheduling algorithms in this tool and we implemented the FCFS with Shorter Job First algorithm as well as we optimized the Round Robin scheduling algorithm that showed there effectiveness in comparison with the ones integrated with CloudSim.

Our future focus will be on merging the proposed scheduling algorithms of this paper with CloudSim Toolkit and also on optimizing and developing new algorithms for CPU allocation and evaluating other cloud computing resources such as storage, memory and network bandwidth from another side. We will also concentrate our work on green computing solutions for Cloud Computing and also on Cloud Computing services cost to illustrate the business perspective of moving on with this new paradigm of computing.

REFERENCES

[1] S. Pal and P. Kumar, "A Simulation-based Approach to Optimize the Execution Time and Minimization of Average Waiting Time Using Queuing Model in Cloud Computing Environment," *International Journal of Electrical and Computer Engineering*, vol/issue: 6(2), pp. 743~750, 2016.
[2] R. N. Calheiros, *et al.*, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Wiley Online Library*, 24 August 2010.
[3] V. Vinothina, *et al.*, "A Survey on Resource Allocation Strategies in Cloud Computing," *International Journal of Advanced Computer Science and Applications*, vol/issue: 3(6), 2012.
[4] S. R. Pakize, *et al.*, "Comparison Of CloudSim, CloudAnalyst And CloudReports Simulator in Cloud Computing," *International journal of Computer Science & Network Solutions*, vol/issue: 2(5), 2014.
[5] T. Goyal, *et al.*, "CloudSim Simulator for cloud computing infrastructure and modeling," *Procedia Engineering,* vol. 38, pp. 3566–3572, 2012.
[6] *Arizona state University Wiki*: http://cloud-simulation-frameworks.wikispaces.asu.edu/
[7] V. Goyal, "Review: Layers Architecture of Cloud Computing," *International Journal of Computing & Business Research.*
[8] M. Gahlawat and P. Sharma, "Analysis and Performance Assessment of CPU Scheduling Algorithms in Cloud using CloudSim," *International Journal of Applied Information Systems*, vol/issue: 5(9), 2013.

[9] *Technopedia*: https://www.techopedia.com/definition/23455/first-come-first-served-fcfs
[10] *GITAM University e-Resources*: http://www.gitam.edu/eresource/comp/gvr(os)/5.3.htm
[11] N. Zanoon and D. Rawshdeh, "STASR: A New Task Scheduling Algorithm For Cloud Environment," *Network Protocols and Algorithms*, vol/issue: 7(2), 2015.
[12] H. S. Behera, *et al*., "Comparative performance analysis of multi-dynamic time quantum Roud Robin (MDTQDRR) algorithm with arrival time," *Indian Journal of Computer Science and Engineering*, vol/issue: 2(2), 2011.
[13] A. Agarwal and S. Jain, "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment," *International Journal of Computer Trends and Technology*, vol/issue: 9(7), 2014
[14] G. S. N. Rao, *et al.*, "Dynamic Time Slice Calculation for Round Robin Process Scheduling Using NOC," *International Journal of Electrical and Computer Engineering*, vol/issue: 5(6), pp. 1480~1485, 2015.
[15] Md. A. Mondal, *et al.*, "Performance Analysis of VM Scheduling Algorithm of CloudSim in Cloud Computing," *International Journal of Electronics & Communication Technology*, vol/issue: 6(1), 2015.

BIOGRAPHIES OF AUTHORS

Gibet Tani Hicham, Phd. Student, Laboratory of Informatics Systems and Telecommunications (LIST), Department of Computer Engineering, Faculty of Sciences and Technologies, Tangier, Abdelmalek Essaadi University, Morocco. Head of IT Systems and Networks at Regional Directorate of Ministry of Justice and Liberties, Morocco

Dr. El Amrani Chaker, Associate Professor, Department of Computer Engineering, Faculty of Sciences and Technologies, Tangier, Abdelmalek Essaadi University, Morocco. Partner country Project Director (PPD), North Atlantic Treaty Organization (NATO). Science for Peace Grant "Real-Time Remote Sensing for Early Warning and Mitigation of Disasters, including Epidemics: The Mediterranean Dialogue Earth Observatory (MDEO)" in collaboration with Tuskegee Univerity (U.S), The George Washington University (U.S), *Boğaziçi* University Kandilli Observatory (Turkey), Abdelmalek Essaadi University (Morocco) and Al Akhawayn University (Morocco) – (August 2011-present).