Improved Scheduling Algorithm In VCL Cloud Computing Environment On CloudSim

Omar Khedher
Innov'com Laboratory, E.N.I.T
Tunisia
e-mail: khedher.omar@gmail.com

Mohamed Jarraya
College of Computation and Informatics
Saudi Electronic University
Dammam, Saudi Arabia
e-mail : m.jarraya@seu.edu.sa

Abstract— **Cloud Computing has empowered the academic communities by reducing the IT infrastructure administration and cost. As one of several cloud computing solutions, Virtual Cloud Laboratory (VCL) has improved many universities experience by providing hand-on labs for each course per class. VCL has multiple deployment requirements. Although it represents a simplified architecture design, treating a large size VCL deployment might be challenging. In addition, putting the performance of resource allocation policies and different scheduling algorithms under scope could help to quantify the workload running in a real and expanded VCL environment. We aim through this study at identifying the limitations of the default scheduler in VCL and propose an enhanced scheduler. By simulating a VCL cloud computing environment, it might be possible to prove different scheduling algorithms and their impact from performance perspective.**

Keywords— Virtual Cloud Laboratory, On Demand Resources, Scheduling Algorithm, CloudSim, Performance Analysis

I. INTRODUCTION

Cloud solutions have spread the terminology of 'on-demand' service covering both software and infrastructure. Such cost effective solution have brought several benefits for medium to large educational institutions over the world. The resources usage in universities is not constant where the demand increases during academic calendar as well as during assignments [1] [2].

Thus, switching to 'on-demand' solution has drastically optimized the usage of resources regardless the variety of user demands. From any location and within any type of hardware, it might be possible in a cloud environment to shift additional capacity to a given application and transfer even services between data centers within the same cloud provider.

Another factor key which highlights the success of the cloud paradigm is the automation of resources; reaching the limit of resources in use is being managed automatically by identifying the needs and reply by increasing the resources up to the required level. Such key factor brings more availability to a given infrastructure by scaling up resources.

Another value added is the flexibility of cloud computing application. This is something not to ignore when considering the consumption of a given service by several components of an institution geographically distributed.

A virtual lab in a cloud computing model is meant to be a distributed application that reflects how the requests are being performed between the gateway of the datacenter and the end user in one hand and how are being processed and executed in the datacenter itself on the other hand.

However, running an application in the cloud is not always under control or predictable [3]. The exponential increase of the demands in a cloud environment may constitute a meaningful degradation of the service exposed and in some cases results a blocking state or bottleneck of the overall system supposed to be available at any time.

VCL is a cloud solution which has grown to handle more user requests [Nearly 8,500,000 HPC CPU-Hrs were delivered by this environment to North Carolina State faculty and students during 2009] [5][13].

Ultimately, VCL presents several challenging research issues. These include the dynamic resource management at the Infrastructure as a Service level [4][7][9].

As any other typical IaaS model, any request made by users will be forwarded to a specific host or entity in a cloud environment based on task scheduling [6][10][11].

Basically, a scheduler as a 'decision-maker' entity provides the way how resources should be provisioned and managed. The VCL scheduler component manages as well resources availability and aims to achieve a high performance computing.

Thus, mapping users VM requests to the suitable compute instance cannot be limited to one standard scheduling algorithm. In the research work reported in this paper, we consider specifically the following questions: (1) what are the possible improvements that can be performed in the default scheduler of VCL? (2) Does the default scheduler take into consideration a large scale cloud environment with different locations?(3)If not, what are the factors that are the most influential on the scheduler's decision [8]?

These questions aim to identify the limitations of the default VCL schedule, and ultimately, would help to extend it. To do so, it might be required to dive in the VCL architecture in the first place. Therefore, we experiment by the means of a simulation toolkit 'CloudSim' the default VCL scheduling algorithm. The preliminary simulation will guide us to conclude the VCL scheduler behavior and introduce few enhancements [3].

The rest of the paper is divided as the following: In Section II we present an architecture outline of the VCL environment [4]. The subsequent section will put under scope the VCL scheduler, cover a succinct overview of the CloudSim and how VCL environment was implemented and designed for simulation. We detail in section IV a simulation analysis followed by comparison between the default VCL scheduler algorithm and the improved one. Finally, we conclude our topic and some orientation for the future work.

II. VCL ARCHITECTURE

A. VCL Overview

VCL is uniquely a set of general-purpose cloud management stack. From one service portal, students are able to access any possible cloud services including:

- *IaaS*: Infrastructure as a Service including access to specific user defined hardware and services running on a different hardware products on-demand [14].
- *HaaS*: Hardware as a Service including access to specific compute, storage and network products on-demand [15][18].
- *PaaS*: Platform as a Service including access to a set of virtualization platforms and middleware enabling user specific applications running on either HaaS or IaaS on demand [16][17].
- *SaaS*: Software as a Service including access to a user defined application that encloses services from PaaS on demand.
- *CaaS*: Cloud as a Service including access to user defined sub-clouds that may encompass anything from IaaS and HaaS on demand [15][18].
- *The VCL graphical User Interface*: through which end user submit reservation requests for services and resources.
- *VCL Database*: stores all variety of system requests, resources state data and usage statistics.
- *VCL Management node*: represents the scheduler allocation resources and responsible for executing requests.
- *Image repository* which incorporates any base-line operating system and any application running on a given operating system.

The VCL daemon eventually predefines the computers existing in the VCL environment. The management node executing the VCL Daemon (vcld) will not be able to expose some set of data center resources without defining and creating entries to both physical and virtual machines.

Under hood of each reservation request, the vcld will deploy images to computers. Note that a management node can define a mix of virtual and physical computers. Reservation depends as well on schedule time of the availability of computer Labs. Basically, schedules define what times during the week a host or a computer is available. Resource availability depends on its current schedule, reservation activity state in the pool and image availability [19]. Figure 1 depicts the mains components of the VCL:

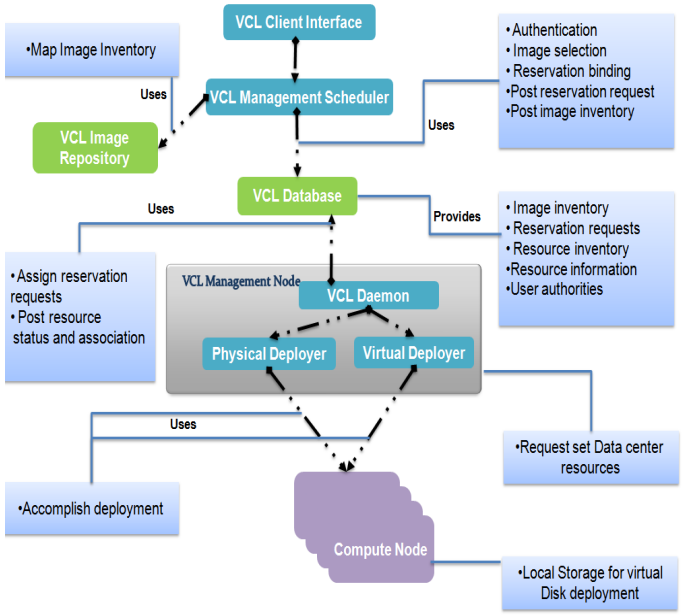

Figure 1. VCL Architecture overview Workflow description

B. VCL Scheduler

The scheduler running by vcld will determine which host or computer will provision the new image depending on its availability.

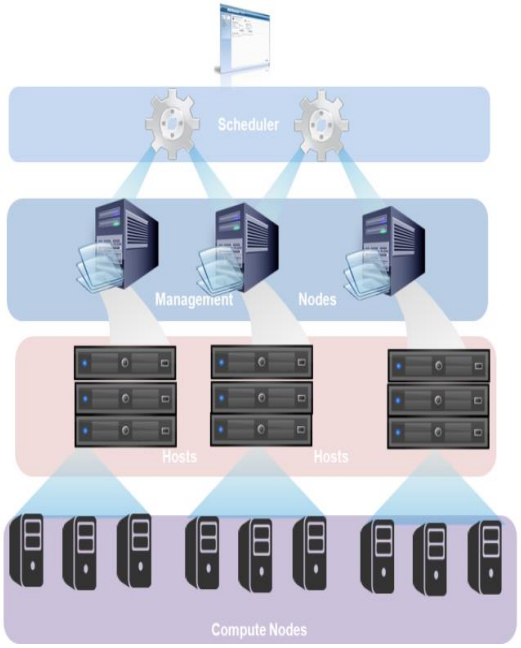

Figure 2. Scheduler orchestration image reservation

In a typical VCL environment, the scheduler checks whether any blade server is preloaded with user requested image. Practically, it will instruct the management node that controls VCL resources to load image request issued by a user onto a computer server that can be either a blade server or virtual machine.

Figure 2 illustrates how the VCL scheduler orchestrates image reservation requests in a large-scale environment having multiple management nodes, blades or hosts (can include image storage) on different locations [12].

The user is able to place a reservation for a certain period of time. By default, the VCL scheduler will assign the requested image to blade server using its default algorithm or a random resource selection algorithm.

The host will be marked as reserved till the end of the session, then it can be reloaded and transited to an available state to fulfil other user requests.

C. *Analysis of VCL scheduler*

The scheduler makes use of host state information from the schedule database sent to it periodically as the basis of any scheduling decisions it makes.

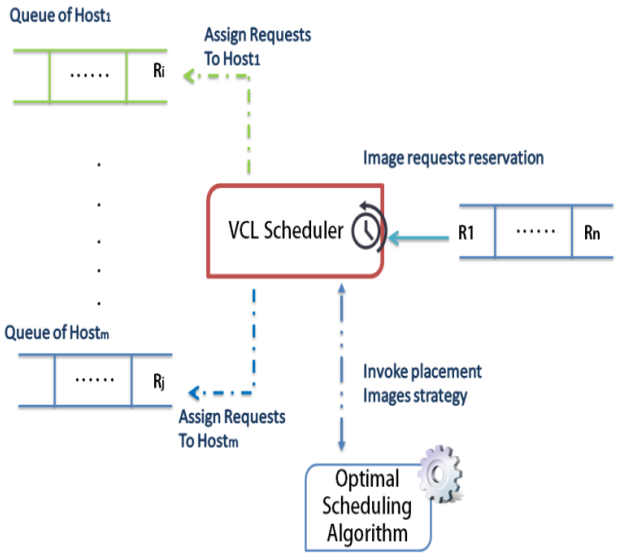

Figure 3. User Image requests assignment

Basically, image requests from several end users are relatively independent. Assuming there are N independent image requests as R1,R2,...,RN. Considering there are M Blades of Hosts [Can be VMs] Host1, Host2,..., HostM, the scheduler sends a request Ri to the queue of a Host based on its scheduling policy periodically as depicted in Figure 3. The existing ranking for computers is as the following:

1) Generate a set of all computers meeting the minimum requirements of the image.

2) Check if the users has access to the filtered set.

3) Check if the new filtered set is mapped to the computers.

4) Order the list by node specifications from low to high

5) Remove from that set any computers that are already assigned reservations.

6) If the image is virtual then it removes any VMs for which the host doesn't have enough memory to load the VM (without overbooking)

7) Assign the new reservation the first computer from that set for which an active management node can be found.

Optionally, the current scheduler algorithm can select randomly host resources. The motivation behind this option is for sites that have homogeneous virtual machines and hosts.

Considering a large-scale VCL deployment environment with multiple regions or pools, it will be essential to highlight a scheduling improvement where resources can be fairly used with the minimum of performance degradation.

With multiple management nodes, the weight of the distributed workload might be randomly distributed between regions with heterogeneous hosts which increase the possibilities of a bottleneck or an infinite waiting time. It will be more convenient to load the balance between different regions or endpoints across the whole infrastructure.

Thus, we take into consideration different metrics for job scheduling algorithms in VCL environment. This can include total execution time in the first place.

For example, in a typical VCL environment, waiting time to submit a reservation for a given Lab can vary between 1 up to 30 minutes to bring a blade/virtual machine ready to use by loading the requested image to the specific host.

In VCL, each "image" or "environment" consists of a set of disk image files and the associated information about the image that is in the database.

Initial "base images" must be created ahead manually. Then, further images are derived from those base images.

Deploying a new VCL image based on the request information can be a time consuming task and even spanning a heavy workload that can degrade the system performance [12]. Since VCL use either a default Scheduling reservation algorithm or a random selection one, we aim in the subsequent section to compare the existing default scheduler within an improved one.

Our involvement regarding the image lifecycle management in VCL will take part in Deploy, Manage and Capture/re-master phase's processes on which the VCL scheduler will decide on which host a given image will be deployed, managed and then how should be reassigned to a specific available image pool or repository.

III. FRAMEWORK AND METHODOLOGY

A. *Design of simulation VCL in CloudSim : CloudSim Framework*

We experimented with CloudSim toolkit to simulate the scheduling algorithms in a scale VCL environment [15].

CloudSim is a cloud computing simulator developed by the University of Melbourne, Australia [16]. The platform aims to generate a cloud model based on simulating virtualized Cloud

resources such as virtual machines, hosts and Datacenters. CloudSim aims to quantify the performance for different services scheduling and allocation models. The cloud simulator is written in Java and defines several classes that construct a fully simulated cloud environment.

It might be essential to highlight in nutshell the main classes existing in CloudSim:

1) Datacenter: is composed of a set of hosts. It is responsible for receiving requests for VMs creation from datacenter class and deploys them into hosts.

2) DatacenterBroker: A Class acting on behalf of a user which performs submitting VM requests to the datacenter as well as submission of tasks to the VMs.

3) Host: Managing VMs provisioned during their lifecycle from creation till destruction. It involves the definition of a set of policies concerning the provision of resources available in each host such as memory, processing element to the virtual machines.

4) VM: A class that models an instance of VM as a software implementation of a machine. Based on a predefined sharing policy, a host can instantiate multiple VMs at the same time in two different resources shared mode: Space-shared and Time-Shared.

5) Cloudlet: In CloudSim simulation, the class Cloudlet is used to mimic a workload of task. It exposes the scheduling policy which is implemented and might be extended in DatacenterBroker.

6) VmAllocationPolicy: is an abstract class that represents the provisioning policy of hosts to virtual machines in a Datacentre. It supports two-stage commit of reservation of hosts: first, we reserve the host and, once committed by the user, it is effectively allocated regarding to the submitted request.

The aforementioned abstract class represents the provisioning policy that a VM Monitor utilizes for allocating VMs to Hosts. The chief functionality of the VMScheduler is to select available host in a datacenter, which meets the memory, storage, and availability requirement for a VM deployment. The default SimpleVMProvisioner implementation provided by the CloudSim package allocates VMs to the first available Host that meets the aforementioned requirements.

B. Implementation of VCL Model on CloudSim

As mentioned before, using CloudSim is to provide easier and cost effective way to interpret and analyse the VCL model. In the current setup, we map some native object classes with the implemented model.

In order to mimic a standard cloud computing environment in CloudSim, we proceed by mapping the right class that will present the right VCL component.

Extensions and modifications are made on native CloudSim code, including Datacenter, DatacenterBroker, VmAllocationPolicy, Host and VM classes.

The main purpose of our implementation is to expand and improve scheduling strategy in DataCenterBroker. We transplant Default VCL scheduler policy and Random scheduling policy algorithm to DatacenterBroker VmAllocationPolicy as extended code. A second iteration will introduce the improved VCL scheduler algorithm by adopting the terminology of VCL management pool and VCL regions that each has a weight or ranking assigned. Each computer would be assigned to a host aggregate. Then, when the initial set of computers is generated, they could be ranked by the host aggregate weight/ranks. It would then be up to an administrator or external application to manage the weight/rank of each aggregate. In order to depict the implementation of the improved VCL scheduler algorithm, we outline its pseudo code as shown in the pseudo code algorithm 1. During the allocation of resources, loading images to VCL compute in our case, scheduling algorithm should be applied in two levels: At the region of DC level which ranked as having shortest path in terms of connectivity performance with more hosts rank available and second level in the DC itself where the blade or the host fulfils the resources requirements.

Algorithm 1 ALGO1: Default VCL Scheduler

```
1: Initialize:
2: Number of Datacenters ← M
3: Number of Hosts in Datacenter[i] ← N
4: Host Limit ← Sum of All VMs available in the host
5: Number of Virtual Machines VMList ← V
6: HostList=ArrangeComputerByCapacity()
7: HostList=ArrangeComputerByMappedImages()

8: loop1: for each Host in HostList
9: Remove reserved computer within requested image
10: Update HostList from Low to High
11: endloop1;

12: loop2: for each Host[j] in HostList in DataCenter[i]
13: if Host[j] is having sufficient RAM and PE then then
14:     if HostLoad <= HostLimit then
15: loop3: for each VM in VMList
16:     if Image is virtual and Host[j] is not having sufficient RAM then
17:         Remove VM within Host[j]
18:     else
19:         if VMImage requested is preloaded then
20:             Filter Host[j][k] And assign lowest Rank
21:             if Management node found within Ranked Host then
22:                 Assign new reservation to VM within Host[j]
23:             else
24:             goto loop2.
25:         EndIf;
26:     EndIf;
27: EndIf;
28: endloop3;
29: EndIf;
30: endloop2;
```

The scheduler must ensure the availability of a preloaded image [whether it is bare metal or virtual machine] to the nearest 'Region' from where the request has been submitted. Therefore, it assigns based on last the host free resources a rank which be populated. According to its scheduler, unavailability for virtual server or real one, fulfilling the requested image already loaded will proceed by selecting any other available host meeting the image specifications or even within the most appropriate VM loader.

Considering a full busy environment where all hosts are being under heavy load, reservation requests will be submitted in a queue for a specific Lab schedule. Here, a region based workload process scheduling system is designed. Algorithm 2 presents the pseudo-code of the improved algorithm.

There are some caveats to the default VCL scheduler algorithm that must be considered though where hosts can be overbooked in terms of processor elements. Thus, the request will be queued and the response time of the image provisioning can be longer. As many requests are being processed, the load on the hosts can bring it to failed status.

Algorithm 1 ALGO2: Improved VCL Scheduler

```
Initialize:
Number of Regions ← N
Number of Datacenters ← M
Number of Hosts in Datacenter[i] ← P
Host Limit ← Sum of All VMs available in the host
Number of Virtual Machines VMList ← V
HostList=ArrangeComputerByCapacity()
HostList=ArrangeComputerByMappedImages()

loop1: for each Host in HostList
Assign a rank host by region
endloop1;

loop2: for each Host in HostList
Remove reserved computer within requested image
Update HostList from Low to High
endloop2;

loop3: for each Host[j][k] in HostList in DataCenter[i][k]
if HostLoad <= HostLimit then
loop4: for each VM in VMList
    if Image is virtual and Host[j][k] is not having sufficient RAM or PEs
then
        Remove VM within Host[j][k]
    else
        if VMImage requested is preloaded then
            if Network Performance of Region[k] is the MAX in all Region[k]
then
                Filter Host[j][k] And assign lowest Rank
                if Management Host found within Ranked Host then
                    Assign New reservation to VM within Host[j][k]
            else
                goto loop4.
            EndIf;
        EndIf;
    EndIf;
    endloop4;
EndIf;
endloop3;
```

The new improved algorithm contribution comes in the first place to take into consideration the selection of the VM host based on available RAM as well as the load on the hypervisors.

Practically, since the vcld code is running in the VCL frontend, there is not currently a way to know what the load on the hypervisors is to be able to use that information for selection of VM host.

Thus, to add this, the backend would need to collect hypervisor load information periodically and push that to the database so that vcld could analyze it in selecting VM hosts.

Furthermore, as was mentioned previously, it might be more efficient to expose a large scale VCL environment as a set of regions where each management node takes care of several hosts. Including a generic overview of all the hosts by regions and image status in each one would reduce the contention load on hosts by decreasing queuing requests.

Even though in case of cloning images between hosts or regions when no images preloaded are available, choosing the best region and best host to load from might perform better in terms of response time to bring a reservation task completed.

The improved algorithm simplified in workflow model exposes the major changes occurred compared to the default VCL scheduler algorithm.

Figure 4 shows the different transitions: Selection of Region, Selection by Host Rank, testing on VM Load by Processor Elements or cores and investigation on region connectivity performance before preceding the reservation.

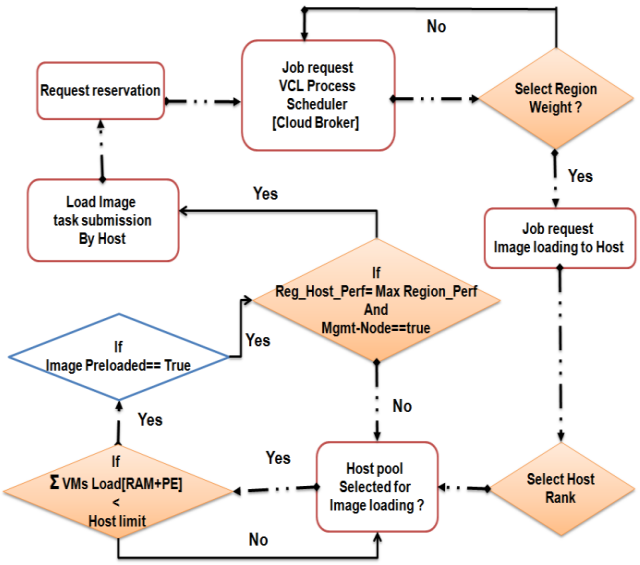

Figure 4. Flowchart of the Improved VCL scheduler algorithm

It is important to note that in CloudSim, two policies namely TimeShared and SpaceShared used for task scheduling in the simulator native code. In a time-shared environment VM task time shared scheduler will distribute the fraction of PEs [Processor Elements] among running elements. In the other hand, in Space shared environment the scheduler assigns specific PEs to specific cloudlet or task. In case of overbooked of cores, it turns to proceed last arrived elements to queue until needed resources become free.

IV. Performance Analysis

In this section, we highlight the present model by CloudSim based on SpaceShared scheduler since it mimics a realistic VCL setup by assigning VMs one cloudlet or job at a time.

A. Simulation Parameters:

The environment is composed of one or many datacenters and several physical hosts.

We define consistent parameters that are going thought the entire simulation as the following:

TABLE I. HOST PARAMETERS

Parameters	Datacenter	Host	VM
PE	10	2	1
RAM	16384	4096	512
Storage	100000	10000	-
MIPS	1000	1000	1000
Bandwidth	10000	10000	1000
SSD	-	50%	-
VMM	KVM	-	-
Arch	x86_64	-	-
Image Size	-	-	10000

We note that the class Host has been extended to support SSD driver for disk I/O improvements when it is enabled in the class. Moreover, considering the importance of the VM limitation when the sum of each one created should not overbook the Host size.

B. Methodology of Validation

To validate the VCL model implemented, we are going to run several simulations; each one will include different number of regions, datacenters, and hosts.

We will consider for all scenarios one shared image which we aim to evaluate its loading response time in the VCL environment.

For all scenarios, we will compare the response time of each scheduling algorithm: VmAllocationPolicySimple, [VAPS] VCLS [VCL Scheduler], and VCLSI [VCL Scheduler Improved]. To give generality for the next scenarios, all the reservations will be executed in homogenous environments where each datacenter will have same number of hosts with same specifications.

The response time is calculated using total capacity of the Datacenter and Hosts including the CPU power and RAM, SSD driver enabled, Image preloading and network performance in a given host aggregate multiplied by respective weight coefficients as the following:

$$\text{Response Time (sec)} = \text{Default\_Load\_Image\_T} \times \text{rand}(\alpha, \beta)$$

Default_Load_Image_T: Default time required to load image residing in a data store attached to a host.

α : Randomized CloudSim weight for the host load image size

β : Randomized CloudSim weight for the host SSD driver enabled

We start with the simplest first scenario one which consists of modeling the case where a single, centralized VCL Cloud data center is used to host 10 machines. In this model, all requests from all students or users around one and same region are processed by a single datacenter.

The next scenario will consider 2 datacenters each running 5 hosts. The hosts are aggregated in different datacenters but in different regions. Each datacenter will have five hosts which randomly expose SSD driver enabled as well as a set of preloaded images. Each scheduling algorithm will define the best host specs to load the image. It is important to note that the VCLS does not take into consideration a sharing load between datacenters: The default implementation takes only into consideration any host mapped to a VCL management node in a given host pool of a datacenter.

The third scenario will consider 4 datacenters each running 10 hosts but in 4 different regions. Datacenters will share the peak load sharing and queuing.

C. Simulation Evaluation

SIMULATION 1:

The resulting of response time of each algorithm is represented in Table II.

TABLE II. SIMULATION 1 RESULTS

Algorithm	Response Time (sec)	Datacenter	Host	Region
VAPS	20.209	1	1	1
VCLS	10.5092	1	2	1
VCLSI	10.5092	1	2	1

As shown in Table II, VCLS and VCLSI show better response time of loading the image since the selection of the host was based mainly on the preloaded image in the host under question. In the other hand, VAPS was limited only on the first available host with the minimum workload.

SIMULATION 2:

The resulting of response time of each algorithm is as the following:

TABLE III. SIMULATION 2 RESULTS

Algorithm	Response Time (sec)	Datacenter	Host	Region
VAPS	20.208	1	1	1
VCLS	10.50816	2	2	2
VCLSI	4.9388352	2	3	2

It is apparent from the Table III that using the VAPS algorithm kept nearly the same response time since the host selected was the first one with less busy PEs. However, VCLS ranked a new host in different DC based on the preloaded image factor but within same region. In the other hand, VCLSI had decreased the response time where a new host was ranked in different region that responds to an image preloaded plus SSD driver availability.

SIMULATION 3:

Considering geographically distributed VCL environment, another factor will be added to the input list which simulates

the network performance between datacenters in different regions. Network degradation or latency can affect the value of response time. Thus, we divide mainly the speed of connectivity between different regions as the following:

- Line 1 : 1GB/s
- Line 2 : 10 GB/s
- Line 3 : 40 GB/s

In order to simulate the affected response time in CloudSim, we assign an extra weight coefficient for network performance μ respectively: 10, 1 and 0.25 and will be inclusively added to region in a random set. The scheduler is intended to select the best path when filtering a similar set of hosts. Therefore the decision should be in favour of the fastest line.

The network performance might play a critical role in deciding the best placement to load the image. The motivation behind such factor is to avoid as much as possible VM migration when a certain host in a given DC does not fulfil the requested image.

The resulting of response time of each algorithm is as the following:

TABLE IV. SIMULATION 3 RESULTS

Algorithm	Response Time (sec)	Datacenter	Host	Region
VAPS	20.209	1	1	1
VCLS	5.2546	2	2	1
VCLSI	1.2347088	3	2	3

VCLSI has reduced drastically the response time based on multi region scenario where network performance factor comes into play. From this perspective, we can assume that considering an overbooked VCL environment in different regions, VCLS can result an error prone to load the image to the requested host in case when there is no PEs available. However, in this case, the VCLSI keeps dynamically updating status in all regions by keeping the request queued and forward the image reservation request to nearest host. Furthermore, the more regions, VCLSI algorithm provides more efficiency.

V. CONCLUSION

In this paper, we have implemented a VCL model in CloudSim and verified it with simulation study based on scheduling for image assignment. Since CloudSim does not give much perspective in the image loading in computers when a user submit a reservation request, we had to proceed by implementing the VCL model by modifying and extending some core functions and classes existing in CloudSim. Although the present model is not able to represent a real VCL environment, the results shown by comparing different scheduling algorithms provides better understanding and perspective at VCL scheduler model and what misses as improvement when considering a large scale environment. The terminology of multi regions and host aggregates in VCL is a new aspect that put under scope tradeoffs of several performance metrics which stresses the scheduler to find the optimized parameters in order to fulfil user reservation requests especially in peak periods. The default VCL scheduler can in certain conditions increase the probability of image cloning and VM migration between regions which may affect the response time and increase the workload. The improved VCL scheduler has decreased that probability by increasing the resource utilization across different regions.

As future work, we aim to elaborate the implemented VCL simulation in real world scenarios. We also plan to analyse another aspect missing in VCL scheduler that manifests on the selection of VM based on load of set of hypervisors as well as bare metal provisioning using xCAT.

REFERENCES

[1] L. Q. Zhang, "Cloud Computing (CLOUD)," IEEE Sixth International Conference, 2013, pp. 204-211.

[2] D.M Chandra, "Role of cloud computing in education," Computing, Electronics and Electrical Technologies (ICCEET), International Conference, 2012, pp. 3-8.

[3] A.. Batra, "High Performance Computing into Cloud Computing Services," Computing Sciences (ICCS) , 2012, pp. 4-8.

[4] S. A. M. Mladen Vouk, "Using Virtual Computing Laboratory (VCL) Technology to Power Cloud Computing," ICVCI, 2008, pp. 1-5.

[5] D. Chandra and M. Borah, "Cost benefit analysis of cloud computing in education," Computing, Communication and Applications (ICCCA), International Conference, 2012, pp. 1-6.

[6] R. Garcia, H. Pluraldom Syst. LLC and J.-M. Chung, "XaaS for XaaS: An evolving abstraction of web services for the entrepreneur, developer, and consumer," Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium, 2012, pp. 2-4.

[7] M. S. Jayasinghe, J. Li, Q. Wang, Z. Wang and C. Pu, "Variations in Performance and Scalability: An Experimental Study in IaaS Clouds Using Multi-Tier Workloads," Services Computing, IEEE Transactions, 2013, vol. 7, no. 2, pp. 2-8.

[8] A. Rindos, "Innovation within IBM (and WebSphere) development through university collaborations," Information Technology Interfaces (ITI), Proceedings of the ITI 34th International Conference, 2012, pp. 10-12 .

[9] J. G. L. O'Loughlin, "Towards Performance Prediction for Public Infrastructure Clouds: An EC2 Case Study," Cloud Computing Technology and Science (CloudCom) , 2013, pp. 2-5.

[10] R. N. Jeyarani, R. Ram and Nagaveni, "Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment," Cluster, Cloud and Grid Computing (CCGrid), 10th IEEE/ACM International Conference, 2010, pp. 4-10.

[11] O. X. Udomkasemsub and T. Achalakul, "A multiple-objective workflow scheduling framework for cloud data analytics," Computer Science and Software Engineering (JCSSE),International Joint Conference, 2012, pp. 3-10.

[12] W. Long, L. Yuqing and X. Qingxin, "Using CloudSim to Model and Simulate Cloud Computing Environment," International Conference on Computational Intelligence and Security (CIS), 2013, pp. 2-8.

[13] A. Rindos, S. Averitt, J. Bass, M. Bugaev, A. Kurth, A. Peeler, H. Schaffer, E. Sills, S. Stein, J. Thompson and M. Valenzisi, "Using VCL technology to implement distributed reconfigurable data centers and computational services for educational institutions," IBM Journal of Research and Development, 2008, vol. 53, no. 4, pp. 3-7.

[14] Y. Duan, "Value Modeling and Calculation for Everything as a Service (XaaS) Based on Reuse," Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 13th ACIS International Conference, 2012, pp. 1-5.

[15] C.-T. Yang, Y.-T. Liu, J.-C. Liu, C.-L. Chuang and F.-C. Jiang, "Implementation of a Cloud IaaS with Dynamic Resource Allocation Method Using OpenStack," Parallel and Distributed Computing, Applications and Technologies (PDCAT) , 2013, pp. 71-78.

[16] A. H. M. Stanik and O. Kao, "Hardware as a Service (HaaS): The completion of the cloud stack," Computing Technology and Information Management (ICCM), 8th International Conference, 2012, vol. 2, pp. 830-835.

[17] Y. Zhang, G. Huang, X. Liu and H. Mei, "Tuning Adaptive Computations for Performance Improvement of Autonomic Middleware in PaaS Cloud," Cloud Computing (CLOUD), IEEE International Conference, 2011, pp. 732-733.

[18] S. Padmavathi, P. Rajeshwari, P. Pradheeba and R. Mythili, "Achieving cost efficiency using CaaS model in the cloud," 2012 Fourth International Conference in Advanced Computing (ICoAC), 2012, pp. 1-5.

[19] P. V Reddy, "Evaluation of different Operating Systems performance in the Private Cloud with ESXi hypervisor using SIGAR framework," Confluence The Next Generation Information Technology Summit (Confluence), 5th International Conference, 2014, pp. 18-32.