International Journal of Emerging Technology and Advanced Engineering

MOOCaaS: MOOCs as a Service

Yassine Tabaa¹, Hassan Ahansal², Sara Ibn Elahrache³, Azza Lajjam⁴, Abdellatif Medouri⁵
¹,⁴,⁵Modeling and Information Theory Team, College of Sciences, Abdelmalek Essaâdi University, Morocco
²National School of Commerce and Management, Abdelmalek Essaâdi University, Morocco
³National School of Applied Sciences, Abdelmalek Essaâdi University, Morocco

Abstract—Cloud Computing has laid the ground for a new generation of educational systems, by providing scalable anytime/anywhere services simply accessed through the Web from multiple devices without worrying about how/where those services are installed, maintained or located. According to numerous studies and reports, the Web has ushered in a new era of possibilities and expectations for transforming education. In this paper we present Jamiaati.org, a Cloud based service delivered as a SaaS (Software as a Service) to the community of academic institutions wishing to set up their own MOOC platform, in order to offer massive open online courses (MOOCs) and ensure a large audience over the globe. To implement this service, we have designed a Cloud-based environment prototype composed of several open source components and a SaaS Handler whose main aim is the dynamic resizing of the environment hosting the MOOC service. First, we present some of the main objectives of this service, and then we describe the design, architecture and deployment environment of Jamiaati.org. Finally, we implement a small scale scenario in order to test the proposed system.

Keywords — MOOCs, Cloud Computing, Software as a Service.

I. INTRODUCTION

Nowadays, the Web has revolutionized our vision of how to deliver courses in a radically transformed and enhanced way. Boosted by Cloud Computing, the use of the Web in education has revealed new challenges but allowed to look forward to new aspirations such as MOOCs (Massive Open Online Courses) [1] as a technology-led revolution ushering in a new generation of learning environments. Expected to deliver effective education strategies, pedagogies and practices, which lead to student success, massive open online courses, considered as the “linux of education”, are increasingly developed by elite US institutions such MIT, Harvard and Stanford by supplying open/distance learning for large online communities without paying any fees. Definitely, MOOCs have the potential to enable free university-level education on an enormous scale.

One key feature of cloud computing is elasticity. This feature allows cloud users to create as many instances of virtual machines as needed to support their computing and storage needs. As it was presented previously in many works, Software as a Service (SaaS) is a cloud hosted service whose distribution model consists of developing and maintaining this service by a Cloud service provider. It is often accessible through Internet, while the billing is based on a subscription model [2].

One of the challenges of the SaaS service model is the design in terms of equipment sizing that will host these services. In fact, the design of the suggested environment should be optimal in terms of resource consumption. That is, the environment should be able to serve all queries submitted by users without having to oversize/undersize the Cloud-hosting. Among the problems often encountered in SaaS models is saturation. This problem is mainly caused by an influx of queries as shown in Figure 1. In this figure, Koller [3] presents the number of students on a MOOC Coursera [4] type through time. We distinguish consultation peaks which correspond to time set for homework or to complete a review. Hence, a bad experience with this service due to its poor performance, caused essentially by service saturation, made most users to stop using the service permanently.

Moreover, in spite of many MOOC initiatives that have been launched by different universities around the world, there are nearly no “open” platforms that offer this service freely to individuals or organizations in order to publish their educational content packaged as a MOOC courses.

Thus, in this paper we focus our work on the following axes:

- Making available an on-demand service named Jamiaati.org dedicated to universities wishing to publish MOOC courses.
- Implementing a “smart” component named SaaS Handler whose main goal is to avoid service saturation by dynamically allocating additional resources and resizing the hosting environment.
Subsequently, we present the design of the platform that will host the jamiaati.org. The platform is an elastic MOOC service based on a private cloud which integrates a component named the SaaS Handler. This component will handle the dynamic extension of MOOC’s instances after detecting overutilization (saturation) risk.

It shall be noted that the scope of this paper is limited to the examination of the overloading problem of SaaS services. The issue of resources’ underutilization will be the subject of a detailed study in future works.

II. MOOC AS A SERVICE: ARCHITECTURE

In this section we describe the overall architecture of the proposed service as well as the implementation of the architecture adopted to set up Jamiaati.org service.

A. The overall architecture

To design our service, we opted for a cloud-based architecture which ensures elasticity and dynamicity of the proposed platform. We deployed the service on a small-scale environment, in this case a private cloud. Moreover, the choice of virtualization facilitates the automatic deployment of the service and the ability to increase and decrease the number of virtual servers operated to deliver the service.

The proposed service Jamiaati.org is essentially based upon three components (as shown in figure 2):

- A MOOC platform, the main application framework, will allow users to manage and upload their educational content. Until the time this was written, four projects lead the change on MOOC’s World: OpenMOOC [5], an initiative of the National University of Distance Education in Spain; Google Course Builder, a MOOC platform created by Google; Class2Go [6], developed by Stanford University and merged with open-edX [7] platform published on June 2013 through an initiative led by tow prestigious universities namely MIT and Harvard university.

- A “smart” component, the SaaS Handler, which enables us to dynamically instantiate MOOCs on our cloud and detect possible saturation of the service by making periodic metering of resources consumption and service availability.

Fig. 1. Progress of a MOOC Coursera type

Fig. 2. Overall architecture of Jamiaati.org service
- A private cloud that will host Jamiaati.org service and which is based on several open source components.

**B. Deployment architecture**

![Deployment architecture of Jamiaati.org service](image)

Figure 3 depicts the deployment architecture of Jamiaati.org service. In this implementation, we adopted a private cloud due to infrastructure limitation, and also this architecture will give us the possibility to get real-time information on resources' utilisation and consumption (CPU, Memory, network ... etc) which is not usually the case in public clouds.

Despite of the benefits offered by cloud computing in terms of flexibility in delivering on-demand resources on a large scale, there is still a need for some automation when deploying platforms over Cloud environments.

This is the case of SaaS type, where the saturation of service can be caused by peaks of uses, a fact that may cause service unavailability due to the lack of resources and to the insufficient number of virtual machines instances deployed.

Traditionally, we detected an overload in a hosting system through the maximum number of concurrent users in order to subsequently allocate additional resources, hence avoiding service saturation. We implemented a component designed to collect status reports of all resources operated by the environment of our platform and which are directly responsible of the quality of responses to various requests submitted by users. This component, called the SaaS Handler, mainly helps our system to decide when it should allocate additional resources in order to satisfy massive users' requests.

To define overutilization, we use a measurable data called “point of tiredness”. This data represents the point of exhaustion and is typically defined in [8] as the point when a limiting resource (such as CPU, memory or storage) reaches 100% of its utilization. In Cloud environments, the “point of tiredness” can be defined as the maximum overload supported by a single virtual machine while maintaining an acceptable quality of service.

Saturation is generally caused by an overutilization of resources when they operate beyond the “point of tiredness”. This means that at least one virtual machine for each physical level of service must be controlled. In some cases, such as when the workload begins to degenerate, it is necessary to minimize operating system tasks such as context switching, CPU consumption, the management of virtual memory and so on.

Mainly, service saturation happens when resource utilization climbs quickly and consequently the quality of service drops to the point of not allowing services to operate properly, this is called low productivity.

**C. Deployment environment**

In order to evaluate the proposed architecture presented in the previous section, we have implemented a system prototype using the following components:

- Class2Go: We have chosen to implement this MOOC platform, an open source in Python and a set of free components developed by Stanford University available on Github web service.
- OpenStack Essex: A Cloud operating system used as the Cloud management system of our service.
International Journal of Emerging Technology and Advanced Engineering

- XCP 1.6: This is the stable version of XCP which proposes several interesting features; it represents our Dom0, DomU which are based on the Ubuntu 12.04.1 distribution.
- MySQL 5.4 as the DBMS of the platform.

III. EXPERIMENTATION AND EVALUATION

We define the performance of the throughput of our platform delivering MOOCaaS as the useful work performed by the service. In order to measure the throughput of our SaaS platform, we conduct an experiment by first simulating users’ requests using JMeter tool [9] which allows submitting massive HTTP requests, and then collecting the workload of each VM using the SaaS Handler component tool developed to exploit OpenStack APIs as it allows us to have real-time status of each virtual machine.

The throughput value is obtained from the number of applications submitted by unit time, where time represents the duration of tests from the beginning of the first request until the end of the last request, these requests include intervals because they represent the response time of the server. The formula for throughput can be defined as follow:

Throughput (SaaS) = (Requests number) / (time).

In previous works [10-12], the authors used this throughput to define an inflection point as the percentage of utilization that is achieved when the throughput starts to decline. Fundamentally, a system which is able to avoid resources’ overutilization is a system which able to efficiently identify this point of inflection.

By overlaying throughput trend on the pattern of resources’ utilisation, it is possible to highlight the common point where throughput and resource utilization become inversely proportional. On the other hand, by correlating the percentage of resources usage to maximum throughput, it is possible to detect when resource utilization is saturated. As well, inflection points are measured for each virtual machine hosting Jamiaati.org service.

Figure 4 depicts an example of inflection point identification. As shown, taking three measures (M1, M2 and M3), three consecutive values (R1, R2 and R3) will be recovered; these values represent the use of a resource such as Memory or CPU consumption. Then three values of throughput will be calculated (P1, P2 and P3). Indeed, an inflection point is triggered when the following occurs: R1 < R2 < R3 and P1 > P2 > P3. As shown in Figure 4, we can say that the resource is saturated at time M2 because from this point throughput decreases sharply causing poor performances. Note that the conducted experiments take into account only memory and CPU consumed by virtual machines to determine the inflection point, knowing that the virtual machine only runs the application environment of MOOC.

A. Experimental environment

In order to simulate the workload, we used a request generator called Apache JMeter, which is open source software developed entirely in Java, and designed to test functional behaviour of services by measuring its performance. Originally designed for testing Web applications, it has since been expanded to other test functions.

Apache JMeter can be used to test performance of both static and dynamic resources (files, Servlets, Perl scripts, Java Objects, databases and applications, FTP servers ... etc.). It can be used to simulate workload on a server, network or simply an object with the objective of testing its strength and analyze overall performance in different situations.

The experiment was conducted using a private cloud infrastructure, which was previously deployed [13], it consists of a blade with a 1000e chassis with 9 M605/M610 blade servers and a SAN of 14 To over three storage arrays associated with one Gigabit link frame. For reasons of limited resources, we reserved three blades for this experiment, each blade will host two virtual machines (VM) that will host our MOOC platform (6 VMs in total) and each VM has a CPU Quad Core (2.4 GHz) and 2 GB of RAM memory, two Ethernet (1 Gbit/s gigabit) network interfaces and a SATA hard drive with a capacity of 140 gigabytes.
The experiment was conducted by generating queries using a rate based on peaks usage model as is the case in Figure 1 for 5 hours (one hour is equivalent to a week of classes MOOC knowing that average duration of a MOOC course is 5 weeks). For each period, a workload starts from zero and increments user to achieve the maximum peak of users in the middle of the period. Then, the load begins to decrease to zero at the end of the period.

B. Evaluation

In this section, we present the results of the experiment conducted to evaluate the effectiveness of the proposed architecture, namely a MOOC as a service proposed essentially for universities wishing to publish a MOOC in order to insure a global audience. As shown in the figure below, we present the results obtained when the service Jamiaati.org is stressed using Apache JMeter tool, this is simulated Jamiaati.org access to the platform by a large number of users in decreasing each test the number of users, it also reflects the reality in the MOOC environments experiencing dropouts during the course of the teaching activity and that will be the next chapter.

The service starts with 3 virtual machines, where a machine is used as a front end server that will guide web queries to the servers hosting application tools for managing the MOOC course.

Table 1: Experiment results

<table>
<thead>
<tr>
<th>Time</th>
<th>Throughput / users on MOOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users on MOOC</td>
<td>Throughput</td>
</tr>
<tr>
<td>Week 0</td>
<td>---------</td>
</tr>
<tr>
<td>Week 1</td>
<td>50000</td>
</tr>
<tr>
<td>Week 2</td>
<td>40000</td>
</tr>
<tr>
<td>Week 3</td>
<td>20000</td>
</tr>
<tr>
<td>Week 4</td>
<td>10000</td>
</tr>
<tr>
<td>Week 5</td>
<td>1000</td>
</tr>
</tbody>
</table>

Figure 5 depicts the obtained results (c.f. table 1) after executing the experiment described in previous section, we see clearly that in the first wave of test, the throughput of our service remained unchanged and does not evolve even if the number of requests during the first week increases. However, this throughput improves significantly until establishing an acceptable value during two minutes, which is the time required to launch new virtual machines, hence to ensure continuity of service and improve quality of service. Indeed, it took less than two minutes to launch the 3 virtual machines using a stored preconfigured image to be integrated into our environment.

It can therefore be observed that productivity has improved while progressing in test; this behavior is due to resources’ status increasingly running on normal conditions. However, this result reveals a new topic to consider in our future works: the underutilization of resources which can cause an over-sizing of MOOC’s environment. This leads us to further improve our SaaS Handler in such a way that to be capable of detecting over-sizing (system underutilization) and take actions to resize the environment and rationalize its use.
IV. CONCLUSION

In this paper, we implemented a small scale prototype of an innovative service called Jamiaati.org, whose main aim is to deliver a SaaS service for universities and academic communities. Indeed, we designed a Cloud based service which enables the automation of creation and launching of MOOCs environments. This service operates several components and open source tools, including the SaaS Handler which allows us to facilitate creation of MOOC’s environment, using another open source component namely Class2go, and able to detect service overutilization (saturation) and therefore call for additional resources to avoid low performances of the proposed service.

As a future work, we will focus on improving the proposed service Jamiaati.org in order to deal with underutilization in addition of overutilization of available resources; we will propose also a SPOCs (Small Private Online Courses) service which is a completely integrated to Jamiaati.org and which is also based on MOOCs concept.

REFERENCES


