2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing

SLA-driven Elastic Cloud Hosting Provider

J. Oriol Fitó, Íñigo Goiri and Jordi Guitart
Barcelona Supercomputing Center
Technical University of Catalonia
Barcelona, Spain
{josep.oriol, inigo.goiri, jordi.guitart}@bsc.es

***Abstract*—It is clear that Cloud computing is and will be a sea change for the Information Technology by changing the way in which both software and hardware are designed and purchased. In this work we address the use of this emerging computing paradigm into web hosting providers in order to avoid its resource management limitations. Thanks to the Cloud approach, resources can be provided in a dynamic way according with the needs of providers and end-users.**

In this paper, we present an elastic web hosting provider, namely *Cloud Hosting Provider* (CHP), that makes use of the outsourcing technique in order to take advantage of Cloud computing infrastructures for providing scalability and high availability capabilities to the web applications deployed on it. Furthermore, we pursue the main goal of maximizing the revenue earned by the provider through both the analysis of Service Level Agreements (SLA) and the employment of an economic model.

The evaluation exposed demonstrates that the system proposed is able to properly react to the dynamic load received by the web applications and it also achieve the aforesaid revenue maximization of the provider by performing an SLA-aware resource (i.e. web servers) management.

***Index Terms*—Cloud computing, web hosting providers, resources outsourcing, Service Level Agreement**

I. Introduction

The Cloud computing paradigm is gaining a lot of popularity throughout the research community. Indeed, it is influencing the development of Information Technology and changing the way that vendors, providers and end-users view computation. As a matter of fact, Cloud computing makes software available as a service over the Internet and has changed the way in which hardware is designed and purchased.

Furthermore, Internet is now turning into the new global way of communication. Mainly, we can affirm that its basis are both web and application servers. For this reason, it is suitable that these servers be able to take advantage of the promising Cloud's possibilities in order to overcome the well-known server's limitations: non-scalability and poor high-availability.

In this work we address the use of Cloud computing for web hosting providers by creating what we have named *Cloud Hosting Provider (CHP)*. It is a web hosting provider primarily characterized by an unlimited set of available resources able to run any web application that attends any number of users. Really, this boundless amount of resources is obtained through the use of resources outsourced to external third-parties, i.e. *Cloud Service Providers* or *Cloud Infrastructure Providers*.

By definition, *outsourcing* is the action of subcontract a process, such as product design or manufacturing, to a third-party company. Thus, it involves the transfer of the management and/or day-to-day execution of an entire business function to an external service provider.

Furthermore, the needed performance (or level of service) of a given web application is formally defined by a contract between two parties: the provider and its customer. This contract is known as Service Level Agreement (SLA). In our environment, it includes both service level parameters and economic values that govern the agreement.

A. Motivation

On the one hand, Cloud computing makes feasible the dreamed vision of computing as a utility, such as power grid, water or telephone. Actually, it is said that Cloud computing has the potential to change a large part of the IT industry.

Furthermore, 'Cloud' applications need to scale up and down quickly according to the input load in order to save money. This situation is known as *Cloud Elasticity* (see Figure 1): the ability to acquire and release resources at fine grain and in a short period of time (order of minutes). The merging of this capability with the way in which these external resources are paid for makes Cloud computing a profitable technology for both users and service providers. Going further, users of Cloud computing can access a service deployed in 'the Cloud' at any time and anywhere. For this reason, both mobile interactive and web applications are good candidates to be moved to the Clouds.

Summarizing, a web application offered as a service over the Internet will be always accessible and its operation and maintenance is more economic for both clients and suppliers.

On the other hand, due to the growing demand that web applications are undergoing nowadays, it is crucial to endow them of two capabilities which are becoming key topics:

Scalability: until today, having a scalable web application meant buying a large amount of hardware equipment in order to attend its peak demands. This fact implies the main problem that we have to buy all the hardware (i.e. resources capacity) required to attend those highest demands. As it shown in Figure 1, this over-provisiong signify that our local servers are underutilized in low demand situations, with the corresponding energetic and administration expenditures. On the contrary, if we build an under-provisioned datacenter we will not pay so

1066-6192/10 $26.00 © 2010 IEEE
DOI 10.1109/PDP.2010.16

much for these costs. Nevertheless, we will lost part of the clients as we are not able to attend the aforesaid peak demands.

Although we think that we have all the needed resources capacity, this also has limitations from a certain point. Thus, our system is still not scalable to high peak demands which has not been foreseen when we made the provision of the local server farm.

Availability: it is essential that web applications be available whenever. Hence, any web application must be accessible at any moment and the users of it will experience better sensations.

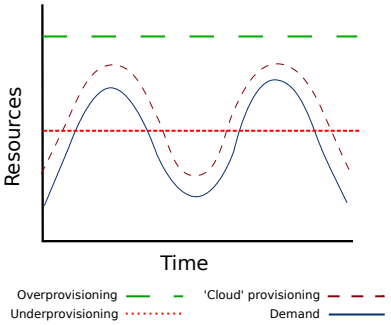

Fig. 1. Differences between the capacity of local and 'Cloud' datacenters

We look upon Cloud computing as a great opportunity to solve these current web applications' limitations. In fact, we need a couple of extra pieces for a successful matching with Cloud computing: the outsourcing technique and Cloud infrastructure providers.

II. Related Work

There is a wide-range of works around the IT outsourcing topic. In fact, the outsourcing is an old enough technology and has been evolving up to coming to that we know as outsourcing 2.0. Most of the related works are of a theoretical character. Dibbern et al. [1] explore and synthesize the academic literature in Information Systems (IS) outsourcing and outsourcing in general. They offer a roadmap of the IS outsourcing literature, accentuating what has been done so far in the last fifteen years, how the whole work fits together under a common umbrella, and what the future directions might be.

Casale [2] defines the 'new' outsourcing (i.e. outsourcing 2.0), explains the three main drivers behind it and highlights how these change affect those who use outsourcing. Motahari-Nezhad et al. [3] present the opportunities and challenges of using cloud computing services with purpose of outsourcing business.

Nevertheless, there is a lack of contributions dealing with this promising topic. However, in some current works [4], [5], [6] both terms cloud computing and outsourcing are related or, simply, the expression 'cloud-based outsourcing' is named. For further information, another related works around the outsourcing topic are [7], [8], [9], [10], [11], [12].

On the other hand, the increasing confidence of companies on IT outsourcing has turned the management attention to a way of managing the relationships between customers and its service providers. The common key for managing this outsourcing alliances is the Service Level Agreement (SLA).

A comparison of SLA use in six of the european comissions FP6 projects is presented in [13]. The structure of SLAs in IT outsourcing are described in [14] and [15] explains the common metrics used in them. Another important works [16], [17] address the SLA management of utility computing. In additon, a fair amount of efforts have focused on addressing the use of SLAs in dynamic and virtualized environments [18], [19], [20].

Finally, Macias et al. [21] propose a use of an Economically Enhanced Resource Manager (EERM) for resource provisioning based on economic models in a Grid market environment. Moreover, they expose different techniques to support revenue maximization across multiple Service Level Agreements. An interesting characteristic of their scenario is that there are not enough resources for executing all the tasks contracted. Thus, an intelligent resource re-allocation mechanism is required for maximizing revenue and minimizing SLA violation penalties. This work is very interesting for us because our main objective is similar to theirs. Nevertheless, our working scenario is totally opposite: we have plenty of resources for executing all the servers needed.

III. Architecture

In this section we present the architecture of the system proposed, that is the *Cloud Hosting Provider* (CHP). Essentially, it is a web hosting provider (i.e. service provider) with an unlimited amount of resources, available through the use of the outsourcing technique to Cloud computing infrastructures.

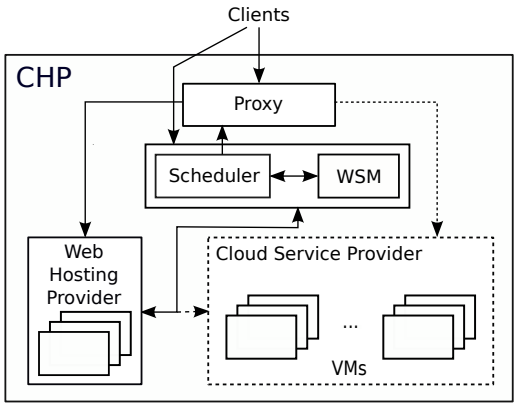

Fig. 2. Cloud Hosting Provider architecture

Its architecture, illustrated in Figure 2, is composed by the following components:

- *Proxy*, is a front-end server that redirects the clients requests to back-end servers. There is a wide range of possibilities and we decided to use Squid [22].
- *Web Server Monitoring (WSM)*, is the logical component that monitors high-level performance metrics, such as the response time, of all the servers deployed. Additionally, it

stores statistical information about these back-end servers useful for a proper servers management. You can find more information in Section V-A.
- *Scheduler*, is the responsible of adapting the whole architecture to the web applications' needs. This means some key actions like reconfigure the proxy according with the number of back-end servers and communicate with the *Cloud Service Provider* to create or destroy those servers.
- *Web Hosting Provider*, is composed by a single (or various) in-house server machines that host the client's web application.
- *Cloud Service Provider or Cloud infrastructure provider*, is the component that allows us to rent an unlimited amount of external resources in order to run any number of virtual machines (i.e. web servers) on top of it. Initially, this component is not present on the CHP, but it is used if the needs require it, i.e. the in-house servers becomes overloaded. The infrastructure provider offers the infrastructure itself as a service, and this allows the CHP to move the computing resources to them, so we can get flexibility and reduce costs. Among all the possibilities, we decided to integrate the system presented with EMOTIVE [23].

IV. SLA-AWARE WEB SERVERS MANAGEMENT

We have designed an SLA-aware web servers management system in order to address the resources outsourcing mechanism on the provider's part with the aim of maximizing its income.

A. Model description

For our purpose of earnings maximization, we define the following economic variables:

- **Price** (Prc_i) is the amount of money that a customer will pay if a provider brings to completion the SLA S_i. It is specified in the same SLA and usually has a predetermined value.
- **Penalty** (Pen_i) is the amount of monetary units that the provider must pay to the client if the accorded SLA S_i is violated. It is also specified in the SLA and is a function of diverse parameters.
- **Cost of Execution** (CoE_i) is a cost for the web hosting provider for executing a web application with an specified SLA S_i. In our environment, this cost have to be splitted in two prices:
  - **Cost of Hosting** (CoH_i) is the price of maintaining a local server that hosts a web application with an associated SLA S_i. It includes its administration, its cooling, the area where it is placed, etc.
  - **Cost of Outsourcing** (CoO_i) is the total amount of money that we have to pay for outsourcing the operation of a given web application to a third-party.

After that, we also determine the below functions:

- The **SLA Satisfaction Function** ($SSF(S_i)$) specifies if an SLA S_i is fulfilled or it is violated:

$$SSF(S_i) = \begin{cases} 1 & \text{if SLA } S_i \text{ is fulfilled} \\ 0 & \text{if SLA } S_i \text{ is violated} \end{cases}$$

- The **Minimum Violating Response Time** ($MVRT(S_i)$) determines the minimum response time from which an SLA S_i is violated.
- The **Outsourced Virtual Machines** ($OVM(S_i)$) is the number of virtual machines that we have externalized for running the servers whose contain the web application with the associated SLA S_i.
- **Revenue** ($Rvn(S_i)$) is the economic benefit that a provider obtains with the execution of a web application whose SLA is S_i. It is defined as:

$$Rvn(S_i) = Prc_i - (CoE_i + Pen_i)$$

- **Punctual Revenue** ($\Delta Rvn(t, M)$) is the total gain (or loss) obtained by hosting a set of web applications with the corresponding set M of n SLAs.

B. SLA criteria

We have the necessity of monitor all running web servers in order to know their performance and act according to the web applications' needs. Really, what it is of our interest is to determine if the performance of any web application deployed is not the contracted in the corresponding SLA.

Pursuing this objective, we have created the $MVRT(S_i)$ function. It specifies which is the minimum response time from which an SLA S_i is violated. In order to previously detect this undesirable situation as early as possible, the response time obtained by the *WSM* component is recorded for each page requested and assessed following these criteria: *Good* if the mean response time is below 50% of the $MVRT(S_i)$; *Tolerable* if the mean response time is between the 50% of the $MVRT(S_i)$ and the value of $MVRT(S_i)$ itself; *Fail* if the mean response time obtained is greater than the value of the function $MVRT(S_i)$ (Figure 3). Thus, we are able to categorize the servers' response time and the *Scheduler* can perform the appropriate operations.

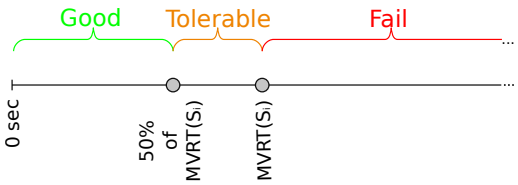

Fig. 3. Response Time criteria

C. Provider penalty

In our working environment, it is convenient to use changeable penalties due to SLA violations. Actually, the penalty is determined by two variables:

- **Time of Violation** (ToV_i) is the total amount of time (in seconds) during which we are violating the SLA S_i.
- **Magnitude of the Violation** (MoV_i) It is equal to a relation between the response time *R* offered by the server and the minimum violating response time ($MVRT(S_i)$):

$$MoV_i = \frac{R - MVRT(S_i)}{MVRT(S_i)}$$

Furthermore, we apply two mathematical functions to calculate the provider's penalty:

- **Gompertz function** ($Gom(S_i)$) determines the penalty associated with the time of violation ToV_i of a given SLA S_i. It is a kind of *sigmoid function*: a type of mathematical model for a time series, where growth is slowest at the start and end of a time period. Its basic form is:

$$y(t) = ae^{be^{ct}}$$

where:

a is the upper asymptote;
c is the growth rate;
b, c are negative numbers.

For our purposes, we have adapted this function in order to obtain result values from 0 to 75 with the goal of limiting the maximum provider's penalty:

$$Gom(S_i) = 75 \cdot e^{-e^{-\frac{ToV_i}{100}+\frac{e^1}{2}}}$$

- **SLA Satisfaction** ($Sat(S_i)$) specifies the sanction related with the magnitude of violation MoV_i of a particular SLA S_i. It is the typical function of SLA satisfaction, but once again limited with an upper asymptote of 50 with the aim of restrict the maximum penalty caused by an SLA violation:

$$Sat(S_i) = \begin{cases} MoV_i & \text{if } MVRT(S_i) <= \\ & MoV_i <= MVRT(S_i) \cdot 50 \\ 50 & \text{otherwise} \end{cases}$$

Finally, we treat these two previous values like a percentage of the price that a customer pays for a particular SLA S_i:

$$Pen_i = \frac{Gom(S_i) + Sat(S_i)}{100} \cdot Prc_i$$

To summarize, note that the penalty considered avoids that the provider loses more than 125% (75% of the $Gom(S_i)$ and 50% for the $Sat(S_i)$) of the price that the customer pays. We apply this policy in order to both limit the maximum penalty that the provider could pay and guarantee a compensation to the customer if the service offered is far below of what was agreed in the SLA.

V. Cloud Hosting Provider Operation

A. Monitoring Servers' Performance

It is carried out by the *WSM* component and it is a crucial procedure in the system proposed. Due to its representativeness, the response time is the high-level performance metric that we decided to capture and analyze. It is defined as the amount of time (expressed in seconds) that the server takes to return to the user the result of a given request. We know that the value of this metric is highly-dependent on the server status: while it is not overload, the response time is always more or less the same. However, when the server is overloaded, the response time grows first linearly and then exponentially. Seeing this pattern we could have the challenge of detecting the form in which the response time is growing but, considering our requirements, it is not enough. Thus, we need some mechanism for that purpose: the SLA criteria explained in Section IV-B.

Tailoring the response time: the *WSM* measures the response time of the servers by performing four repeated tests every 10 seconds. For each of these tests, it calculates the median and the standard deviation. Then, the two medians with less deviation are selected to obtain a global median that is stored as the new response time calculated. We have designed this method in order to avoid the outlayers.

Because of CHP highly depends on the servers' response time, we also decided to use another statistic mechanism in order to tailor the aforesaid median of this metric. Actually, the response time of a server can be very variable in short periods of time and we want to filter this variability to obtain a more regular pattern. In fact, these changing measures would affect the operation of the *Scheduler* and we must avoid it. We decided to use the *Exponentially Weighted Moving Average* (EWMA) statistic method. It is a type of finite impulse response filter used to analyze a set of data points (response times in our case). Specifically, EWMA applies weighting factors which increase exponentially. In fact, the degree of weighting increase is expressed as a constant *smoothing factor* α, a number between 0 and 1. Anyway, it determines the weight of the most recent observation (Y_{t-1}) with respect to the previous ones (S_{t-1}). The general formula used to calculate the EWMA at time periods *t > 2* is:

$$S_t = \alpha \cdot Y_{t-1} + (1 - \alpha) \cdot S_{t-1}$$

Our application of EWMA method to servers' response time is carried out by the following expression:

$$R_t = \left(e^{-(1/T)}\right) \cdot R + \left(1 - e^{-(1/T)}\right) \cdot R_{t-1}$$

where:
R_t represents the new time tailored using EWMA;
R_{t-1} is the last time tailored;
T is the total time since we are monitoring the servers;
R is the last response time calculated by the WSM.

We decide to use a *smoothing factor* equal to $(e^{-(1/T)})$. In this way the obtained response time continues the pattern of time calculated by the *WSM* but avoiding occasional peaks.

B. Anticipated Outsourcing

We also need to create another mechanism concerning with both SLA and outsourcing operation. Mainly, its goal is to detect probable SLA violations in the not too distant future. We require this additional procedure because the creation and initialization of a server into an outsourced virtual machine require an unpredictable time of a few seconds (or minutes) which we cannot despise.

In particular, we need to predict if a future penalty for SLA violation will overcome the cost of outsourcing. Observe that

this is the case in which is more economically-valuable to externalize a new web server than pay the penalty for SLA violation. Then, following the purpose of revenue maximization of the provider, we need to have a new outsourced server for this moment in which will be beneficial for the income of the supplier.

In our implementation, we have created an independent thread that is responsible of performing this mechanism. In any way, note that the nedeed time for create and initialize a VM with, for example, EMOTIVE and Amazon EC2 is, approximately, 10 and 60 seconds, respectively. Thus, we have to adapt our system to each Cloud service provider that we want to use. Now we are presenting the CHP with its coupling with EMOTIVE and for this reason we work with a time of 10 seconds.

Finally, we want to remark that the implementation of this mechanism have been performed by following the revenue maximization of the web hosting provider. Without this method, we might not have a new outsourced web server for when the economic heuristics determines that it is needed.

C. Scheduler Operation

In this section we explain the logic of the CHP. In particular, we describe how it manages the number of back-end servers to meet the web applications requirements. We have implemented an SLA-aware resource (i.e. web server) management.

When a new web application with an associated SLA arrives into the system, the *Scheduler* is the responsible of deploying it. The general procedure is to initially create one local server and the content of the web application deployed on it. Then, if the performance of this initial server decreases, we have to detect it and act to avoid a loss in the total gain. The general idea is to create and destroy a replica of this initial server created according to the needs of the web application in question. Entering in more detail and lowering the level of granularity, we also define the four modes related with the web applications deployed within CHP (see Figure 4 for a graphical explanation). The main reasons for changing those states are both SLA violations and whether the penalty is greater or lower than the cost of outsourcing.

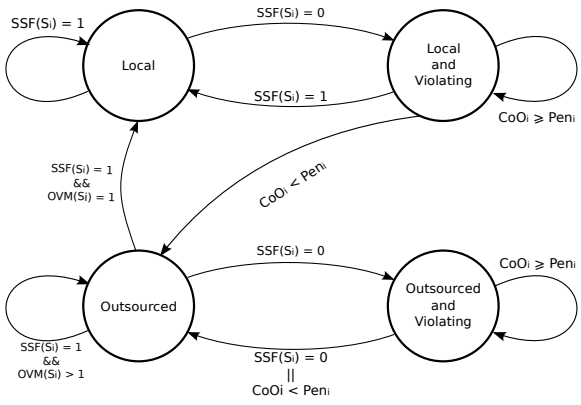

Fig. 4. Web applications states within the CHP system

VI. Experimentation

In Section III we have exposed the architecture of a system that uses Cloud-based outsourcing for overcome some limitations regarding with web hosting providers. Now, in this section, we present an evaluation concerning with its functioning.

A. Experimental Environment

We use Apache Tomcat v5.5 [24] with the hybrid architecture [25] (multithreaded and event-driven) as the back-end server. It is an open-source servlet container developed at the Apache Software Foundation. Moreover, we use Squid [22] as the proxy server. It supports HTTP and HTTPS protocols, among others, and is capable to carry out an efficient load balancing. The experimental environment also includes the deployment of the SPECweb2005 [26] banking application on Tomcat. Going inside the design of this application, it is important to note that all the requests are based on SSL and the session timeout is implemented. Specifically, about 20% of the incoming users do not logout, thus allowing the session timeout. Besides, and after each page request, there is a likelihood that the user goes through a think time, which averages about 9.98 sec (data extracted from [27]). In addition, the workload for the experiments has been produced using Httperf [28] instead of using the clients of the SPECweb2005 benchmark. We decided that due to httperf allow us to make more configurable and variable test cases. In fact, the workload's requests generated by httperf were extracted from a characterization of the SPECweb2005 client emulator. For all the tests, we configured an average connection and client timeout of 10 seconds.

All the back-end servers are encapsulated in a Sun Java Virtual Machine v1.6 (with a maximum heap size of 512MB) and run with one processor unit and 512MB of memory available for the server. All the machines are connected through 1 Gbps Ethernet and run Xen 3.3.1 over linux kernel 2.6.18. We use EMOTIVE as the Cloud Service Provider. Note that outsourced web servers can access, in a secure way, to central user databases through a VPN.

Finally, the software of the CHP has been written in Java and runs under JRE 1.6. Moreover, the scripts for managing the proxy configuration are written in Bash script.

B. Anticipated Outsourcing

In this section we illustrate the procedure of the anticipated outsourcing mechanism explained in Section V-B. Figure 5 shows both how a virtual machine containing an outsourced web server is created when it is needed and how the *Scheduler* acts when the *WSM* detects an SLA violation. We differentiate the following five stages:

1) The response time of the initial server (i.e. local server) is within the *Good* range of the SLA criteria defined in Section IV-B. Therefore, the *Scheduler* does not have to perform any operation. Note that the server of *Web Hosting Provider* is not overloaded during this phase (see the third subfigure).

2) The *Web Server Monitoring* component classifies the response time as *Tolerable*. During this stage is when the local server starts to be overloaded and, consequently, its reponse time is getting bigger.

3) After checking that the response time is within the *Fail* range, the *Scheduler* waits until the *WSM* estimates that the penalty will exceed the cost of contracting an external server. Then, the anticipated outsourcing mechanism is initialized. As you can see in the fourth subfigure, the load of the *Cloud Service Provider* physical machine is increased (see continuous line), fact that confirms that its operating system is creating the new virtual machine.

4) The outsourced virtual machine is being created and initialized during this period. At the same time, the response time of the server continues in the *Fail* range. Consequently, the penalty for violating the SLA achieve the cost of the outsourcing operation. Thus, we have the new web server ready for when we need it and we can affirm that we successfully create a mechanism which allows us to maximize the revenue of the provider. Additionally, note that there is a peak in the load of the new virtual machine (see dotted line of fourth subfigure) due to the web server initialization.

5) Finally, the *Scheduler* reconfigures the proxy in order to make available the new outsourced web server for the web application in question. Hence, the load between the local and the outsourced server is balanced by the proxy (see the third and fourth subfigures). Consequently, there is a decreasing of the response time and now it is classified as a *Good* time according to the SLA criteria.

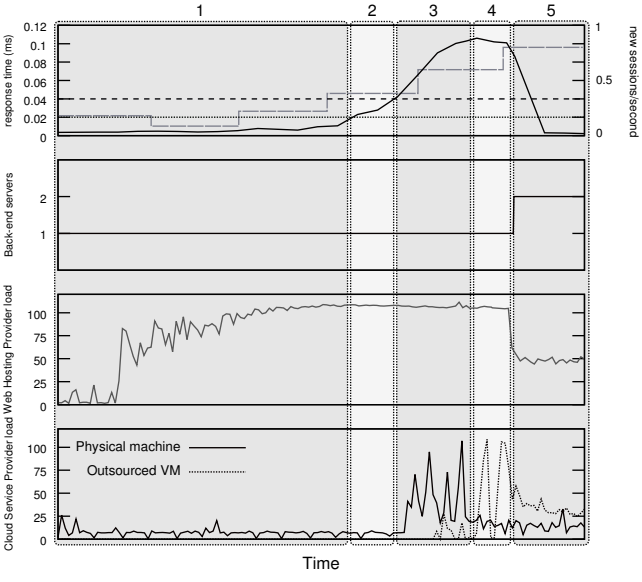

Fig. 5. Scheduler's evaluation: anticipated outsourcing operation and replication of back-end servers. From top to bottom: input load and response time of the servers, number of back-end servers, load of the Web Hosting Provider (i.e. local machine) and load of the Cloud Service Provider machine where outsourced servers run.

C. Revenue Maximization

In this section we analyze how the CHP is capable of maximize the revenue earned by a web hosting provider in two different scenarios: with an incoming fixed load of new user sessions per second and with a variable input load that represents a typical load of a whole day of web sites of nowadays. Note that the price that the client pays to the provider for hosting the web application is 100 monetary units and the cost of hosting a local and an outsourced server are 6 and 5 currencies, respectively. Thus, the maximum revenue is 94 monetary units because we initially have one local server in all the configurations.

1) Fixed Incoming User Sessions: the aim of this first part is to assess how the CHP is able to maximize the provider's revenue with different input loads. Actually, we compare the revenue obtained by the CHP with the revenue acquired by two static configurations: one local server and two servers (one local and one outsourced) at all times. Figure 6 shows the revenue, expressed in monetary units, earned in three different scenarios with an online-bank application from SPECweb2005 deployed and as a function of the number of user sessions.

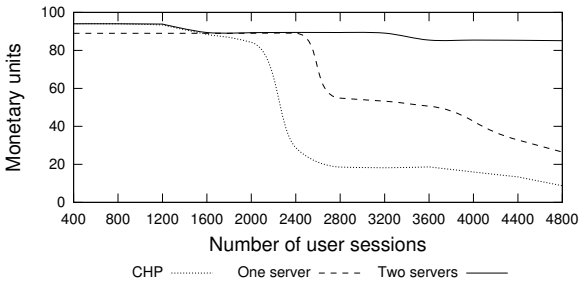

Fig. 6. Average revenue comparison between CHP and static configurations as a function of the number of user sessions

As you can see, the CHP obtains better revenue in all the cases. In fact, this is the situation expected because it adapts the number of back-end servers according to the needs detected by it. On the other hand, the average revenue obtained by static configurations are penalized by the facts of paying at all time both the cost of maintaining a fixed number of back-end servers, whether local or outsourced, and a substantial amount of penalty due to they are not scalable configurations and the servers meets overloaded.

2) Typical Workload of One Day: in this second part we want to reproduce how the system presented would act in front of a typical load of web sites of nowadays [29]. In fact, Figure 7 shows how the CHP readjusts the number of back-end servers to the needs of the web application according to the servers' response time classification.

Seeing in detail the figure, there is a pattern that repeats itself through the execution illustrated:

- When the input load causes the server(s) to be overloaded, its response time gets bigger. This implies that the *WSM* detects this growing and waits for the confirmation of the SLA violation. Then, it informs the *Scheduler* that a new outsourcing operation is required.

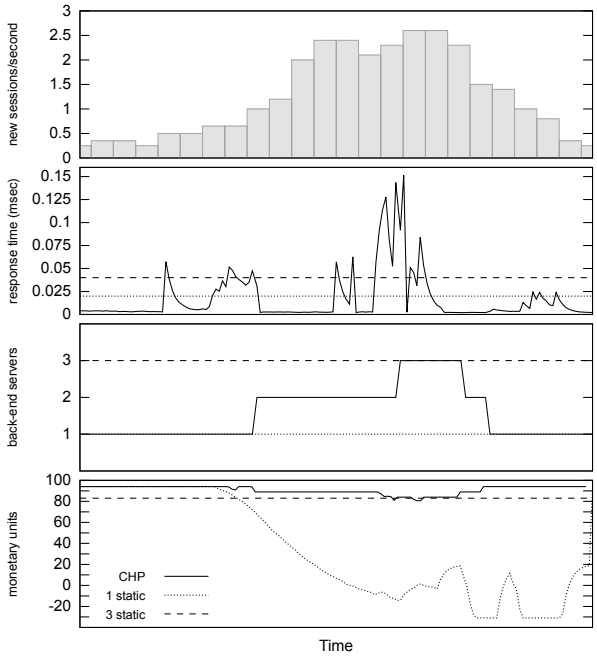

Fig. 7. Revenue Maximization in a typical workload of one day. From top to bottom: Variable input load (new user sessions/second); servers' response time (milliseconds); number of back-end servers and revenue earned by the CHP, one static server and three static servers (expressed in monetary units).

- The *Scheduler* waits for the response of the *Cloud Service Provider* informing that the new virtual machine is ready.
- Afterwards, this new web server is made available for the web application.
- Finally, the load of the back-end servers decreases and we follow the monitoring of those servers in order to attend the future web application's demands.

Actually, the economic differences between CHP and the two static configurations are expressed in Table I.

	Avg. Cost of Execution	Avg. Penalty	Avg. Revenue
CHP	8.87	0.24	90.89
1 server	6	60.14	33.46
3 servers	17	0	83

TABLE I
COST OF EXECUTION, PENALTY AND REVENUE COMPARISON IN A TYPICAL WORKLOAD OF ONE DAY.

The average revenue of the CHP is around 91 currencies, whereas if we have an over-provisioned configuration to meet the highest peak (3 servers), the gain is around 83 because we must pay for maintaining the servers also in non-overloaded situations. Furthermore, if we only have one local server, we must pay a great amount of penalties and there is a very low profit.

Imagine a situation where the load shown in Figure 7 is the same that a given web application receive every day. Considering an entire year, we would earn 12213 monetary units if we only have one local server; 30295 currencies if we have enough local resources to be able to attend the peak load; and 33215 with the CHP. This fact results in an increase of 271% and 9% of the revenues of having one and three static servers, respectively.

Finally, we want to conclude this experimental chapter with the highlighting of the main difference between having or not CHP: the reaction time to web applications' workload changes. With it, we are able to attend peak demands, of any size in any moment, very quickly; while without it, we have to provision our server farm of enough local servers to attend those peak demands. Moreover, this over-provisioning of servers is not feasible in almost all the real hosting environments and incurs a huge amount of costs.

VII. CONCLUSIONS

This work has shown a promising use of Cloud computing among the wide range that has arisen with the emergence of this new paradigm. Actually, the proposed system allows web applications to overcome their typical limitations, non-scalability and poor high-availability, by taking advantage of Cloud computing infrastructures. Thus, we have proposed the *Cloud Hosting Provider*: a web hosting provider that makes use of Cloud service providers for scaling the web applications deployed on it. This scalability is performed by outsourcing to these external third-parties the management and operation of the required web servers that we cannot run in local machines.

Furthermore, we assess the revenue maximization of this new type of web hosting providers. We are seeking this earning maximization although the performance of the web applications meets a bit diminished. Following this goal, we have designed an SLA-aware web servers management: the administration of the back-end servers needed for running a given web application is determined by an analysis of the SLA upon it. We have defined a set of economic variables and functions that make feasible, at any moment, the study of the revenue acquired by the provider and how it should act to maximize this income.

With the aim to demonstrate our achievements, we have experimented with the system proposed. As a general interpretation of the results we can affirm that the CHP is able to maximize the revenue obtained by the provider while meeting as much as it can the performance signed in the SLA. Note that in both static over-provisioning and dynamic scenarios, the service could sell any surplus resources but the consideration of this option is out of the scope of this paper. On the other hand, we have presented an individual evaluation of the main procedure of the *Scheduler*, that is, to perform an anticipated outsourcing with the goal of replicate the back-end servers.

A. Future Work

As a future work we are examining the possibility of integrating the system presented with some others Cloud infrastructure providers like Amazon EC2 [30]. Through this multiple integration, we will have diverse Cloud service providers with different costs of having an outsourced web server on it. Afterwards, we could incorporate some additional

methods for choosing the most convenient provider in which to outsource in each case.

Scaling up and down the number of back-end servers according to the input load is a crucial operation of the CHP. Related with this action we are thinking in incorporating a kind of prediction into the anticipated outsourcing mechanism.

Another very important problem of execution environments of nowadays to deal with is the fault tolerance. We are planning to incorporate this capability by adding some resource monitoring of the machines in which web servers run.

In addition, it is common that Cloud service providers offer customized virtual machines. We could take profit of this fact for resizing the virtual machine to outsource in each case.

Finally, we are considering to replace the actual front-end server by an IPVS (IP Virtual Server) [31].

ACKNOWLEDGMENT

This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2007-60625 and grant AP2008-02641 and by the Generalitat de Catalunya (2009-SGR-980).

REFERENCES

[1] J. Dibbern, T. Goles, R. Hirschheim, and B. Jayatilaka, "Information systems outsourcing: a survey and analysis of the literature," *ACM SIGMIS Database*, vol. 35, no. 4, pp. 6–102, 2004.

[2] F. Casale, "Outsourcing 2.0," *Available at www. hroassociation.org/uploaded/documents/outsourcing2.0 whitepaper.pdf*, 2007.

[3] H. Motahari-Nezhad, B. Stephenson, and S. Singhal, "Outsourcing Business to Cloud Computing Services: Opportunities and Challenges," *Available at www. hpl.hp.com/techreports/2009/HPL-2009-23.html*, 2009.

[4] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[5] L. Wang, "Cloud computing: A perspective study," *Available at http://hdl.handle.net/1850/7821*, 2008.

[6] H. Labs, "The benefits of combining business-process outsourcing and service-oriented architecture," *Available at h20195.www2.hp.com/PDF/4AA0-4316ENW.pdf*, 2006.

[7] P. Laplante, T. Costello, P. Singh, S. Bindiganavile, and M. Landon, "The who, what, why, where, and when of IT outsourcing," *IT Professional*, vol. 6, no. 1, pp. 19–23, 2004.

[8] J. Barthelemy, "The hidden costs of IT outsourcing," *MIT Sloan Management Review*, vol. 42, no. 3, p. 60, 2001.

[9] N. Venkatraman, "Beyond outsourcing: managing IT resources as a value center," *Sloan Management Review*, vol. 38, pp. 51–64, 1997.

[10] R. Gonzalez, J. Gasco, and J. Llopis, "Information systems outsourcing reasons in the largest Spanish firms," *International Journal of Information Management*, vol. 25, no. 2, pp. 117–136, 2005.

[11] L. Baldwin, Z. Irani, and P. Love, "Outsourcing information systems: drawing lessons from a banking case study," *European Journal of Information Systems*, vol. 10, no. 1, pp. 15–24, 2001.

[12] J. Gray, "Distributed computing economics," *Queue*, vol. 6, no. 3, pp. 63–68, 2008.

[13] M. Parkin, R. Badia, and J. Martrat, "A comparison of SLA use in six of the european commissions FP6 projects," *Institute on Resource Management and Scheduling, CoreGRID-Network of Excellence, Tech. Rep. TR-0129, April*, 2008.

[14] J. Goo, D. Kim, and B. Cho, "Structure of Service Level Agreements (SLA) in IT Outsourcing: The Construct and Its Measurement," *AMCIS 2006 Proceedings, Acapulco, Mexico*, p. 391, 2006.

[15] I. Hayes, "Metrics for IT Outsourcing Service Level Agreements," *Available at www. clarity-consulting. com/metrics_article. htm*, 2004.

[16] M. Buco, R. Chang, L. Luan, C. Ward, J. Wolf, and P. Yu, "Utility computing SLA management based upon business objectives," *IBM Systems Journal*, vol. 43, no. 1, pp. 159–178, 2004.

[17] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. Pazel, J. Pershing, B. Rochwerger, I. Center *et al.*, "Oceano-SLA based management of a computing utility," in *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings, Seattle, Washington, USA*, 2001, pp. 855–868.

[18] K. Hartig and D. Reedy, "Associating Service Level Agreements to Applications in a Dynamic Environment," *Available at www.comp.lancs.ac.uk/computing/ research/mpg/reflection/papers/rio-dynamic-sca.pdf*.

[19] G. Lodi, F. Panzieri, D. Rossi, and E. Turrini, "SLA-Driven Clustering of QoS-Aware Application Servers," *IEEE Transactions on Software Engineering*, vol. 33, no. 3, pp. 186–197, 2007.

[20] J. Ejarque, M. de Palol, Í. Goiri, F. Julià, J. Guitart, R. Badia, and J. Torres, "SLA-Driven Semantically-Enhanced Dynamic Resource Allocator for Virtualized Service Providers," in *Proceedings of the 2008 Fourth IEEE International Conference on eScience*. IEEE Computer Society Washington, DC, USA, 2008, pp. 8–15.

[21] M. Macías, O. Rana, G. Smith, J. Guitart, and J. Torres, "Maximizing revenue in Grid markets using an economically enhanced resource manager," *Concurrency and Computation: Practice and Experience*, 2008.

[22] Squid Proxy, http://www.squid-cache.org/.

[23] Emotive Cloud - Barcelona, http://www.emotivecloud.net.

[24] Apache Tomcat, http://tomcat.apache.org.

[25] D. Carrera, V. Beltran, J. Torres, and E. Ayguade, "A Hybrid Web Server Architecture for e-Commerce Applications," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)-Volume 01*. IEEE Computer Society Washington, DC, USA, 2005, pp. 182–188.

[26] SPECweb2005 benchmark, http://www.spec.org/web2005/.

[27] R. Hariharan and N. Sun, "Workload characterization of SPECweb2005," *SPEC Benchmark Workshop. SPEC, Available at http://open.spec.org/workshops/2006/papers/02_Workload_char_SPECweb2005_Final.pdf*, 2006.

[28] Httperf tool, http://www.hpl.hp.com/research/linux/httperf/.

[29] J. Torres, D. Carrera, V. Beltran, N. Poggi, K. Hogan, J. Berral, R. Gavaldà, E. Ayguadé, T. Moreno, and J. Guitart, "Tailoring resources: the energy efficient consolidation strategy goes beyond virtualization," in *Proceedings of the 2008 International Conference on Autonomic Computing, Chicago, USA, June 2-6, 2008: Proceedings*. IEEE Computer Society, 2008, pp. 197–198.

[30] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/.

[31] IP Virtual Server, http://www.linuxvirtualserver.org/software/ipvs.html.