# Addressing the Use of Cloud Computing for Web Hosting Providers

J. Oriol Fitó[1] and Jordi Guitart[2]

*Abstract*— **Nobody doubts about cloud computing is and will be a sea change for the Information Technology. Specifically, we address an application of this emerging paradigm into the web hosting providers. We create the *Cloud Hosting Provider* (CHP): a web hosting provider that uses the outsourcing technique in order to take advantage of cloud computing infrastructures (i.e. cloud-based outsourcing) for providing scalability and availability capabilities to the web applications deployed. Hence, the main goal is to maximize the revenue obtained by the provider through both the analysis of Service Level Agreements (SLA) and the application of economic functions. This target is achieved by using an SLA-aware resource (i.e. web servers) management. Through the experimentation we demonstrate that the CHP is able to maximize the hosting provider's revenue while offering to the web applications scalability, availability and very good performance.**

*Keywords*— **Cloud Computing, Web Hosting Providers, Cloud Service Providers and Service Level Agreement.**

## I. Introduction

It is clear that cloud computing is gaining popularity and the research community are focusing a lot of efforts in working for its implantation into several fields. The greatness of this new paradigm is that it has remarkable consequences and advantages in several IT fields. In addition, it also has valuable repercussions into all the parts involved in many IT operations, including vendors, providers and end-users as well. 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, unavailability and failure intolerance.

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 *Infrastructure Providers* in this case.

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. Indeed, the outsourcing operation is also gaining popularity in the recent days.

On the other hand, 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, this contract includes economic penalties and the right to terminate if SLAs are consistently missed.

[1]Barcelona Supercomputing Center (BSC), e-mail: `josep.oriol@bsc.es`

[2]Dpt. d'Arquitectura de Computadors, Univ. Politècnica de Catalunya, e-mail: `jguitart@ac.upc.edu.`; Barcelona Supercomputing Center (BSC), e-mail: `jordi.guitart@bsc.es`

## 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. Mainly, they identify five major sourcing issues: the questions of *why* to outsource, *what* to outsource, *which* decision process to take, *how* to implement the sourcing decision, and what is the *outcome* of the sourcing decision.

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. Firstly, they study advances in cloud computing and discuss the benefits of using cloud services for businesses. Nevertheless, there is a lack of works dealing with this emerging and promising topic. However, in some current works [4], [5] and [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 outsourcing topic are [7], [8], [9], [10], [11] and [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). For this reason, a fair amount of efforts ([13], [14], [15],

[16], [17], [18], [19] and [20]) have focused on assessing the SLA utilization as we could do on the IT outsourcing research line. Generally, what it is of our interest are the works that employ any kind of resource management, in dynamic execution environments, for fulfilling a given Quality of Service (QoS) specified in the SLA contracted between two parties.

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 reallocation 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

This section presents the structure and the operation of the proposed system. It is a web hosting provider (i.e. service provider) with an unlimited amount of resources, available through the use of the outsourcing technique. Hence, it offers to the customers the possibility to host any kind of web applications of whichever magnitude. The most interesting fact is that it is able to use the most emerging cloud computing platforms (i.e. Cloud Service Providers or Infrastructure Providers) to provide both scalability and availability to these web applications. As it is usual in the cloud platforms, the virtualization technique has also a key role in our system.

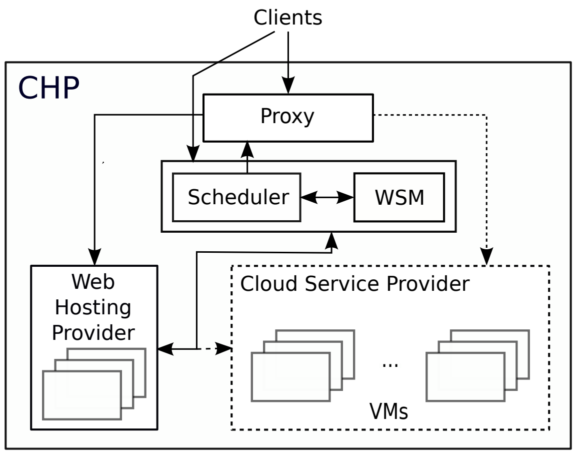

Fig. 1. Cloud Hosting Provider architecture

On one hand, the presented approach let us to configure an isolated 'cloud' environment for each web application that adapts very well to its specific high availability and performance needs. On the other hand, our system tries to maximize the vendor's revenue through the analysis and application of some economic functions (see Section III-A). Thus, the Service Level Agreement (SLA) between the two parties, customer and provider, determines what is the expected performance of the associated web application. Specifically, this SLA has to specify various parameters like how much the customer pays for the service, the penalties if the agreement is not accomplished at all, etc.

As a classic web hosting provider, our system is composed of multiple *web servers* and/or *application servers* that are who actually process the clients' requests. However, note that in our CHP these servers could be encapsulated within both the local (*Web Hosting Provider*) servers and the third-party (*Cloud Service Provider*) Virtual Machines *(VMs)*. The idea behind this separation is because initially the CHP deploys the web applications in local servers, as hosting providers have always done. Afterwards, we will outsource to an external third-parties part of the operation of any web application if these local servers become overloaded. Note that with this outsourcing mechanism, we avoid our servers to be overloaded and, thus, the deployed web application are scalable and available for any time.

In order to protect and control the load among all these deployed servers, a *proxy* (i.e. front-end server) is required. Because of the system's features, we have the need of constantly monitor the web applications' performance. Through this performance monitoring, we will be able to determine the best action to be done with the aim to maximize the total earning on the vendor's part. With this purpose in mind, we have created the *Web Server Monitoring* module. Due to its representativeness, the response time is the high-level performance metric that we decided to obtain and analyze. In addition, we create the *Scheduler entity* in order to well-adapt our system to the web applications' requirements and control the management of the servers where these web applications are deployed (Figure 1).

### A. *Whether or not to Outsource for Maximizing Provider's Revenue*

As we already said, the main purpose is to address the resources outsourcing mechanism on the vendor's part and maximize its revenue. With this aim in mind, we have designed an economical model and various heuristics that let us to achieve this aforementioned goal.

For our purpose of takings 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. Normally, is composed by a penalty scheme or various sanctioning policies.

∘ **Cost of Execution** $CoE_i$ is the cost for the web hosting provider for executing a web application with an specified SLA $S_i$. In our environment, this cost has 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 the web application to a third-party. Commonly, the companies which rent their own resources, such as computing capacity, disk space and network bandwidth, among others, have well-defined charges for use them. Specifically, we consider this cost as the price of an outsourced virtual machine containing the web server.

After that, we also determine the below functions:

∘ **SLA Satisfaction Function** (SSF) 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** $MVRT_i$ determines the minimum response time from which an SLA $S_i$ is violated. A more detailed explanation about SLA violation is given at Section III-B.

∘ The **Outsourced Virtual Machines** $OVM_i$ is the number of virtual machines that we have externalized for running the servers which contains 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:

$$\Delta Rvn(t, M) = \sum_{i=1}^{n} Prc_i - \sum_{i=1}^{n} \big(CoE_i + Pen_i\big) =$$
$$\sum_{i=1}^{n} Prc_i - \sum_{i=1}^{n} \big(CoH_i + (CoO_i * OVM_i) + Pen_i\big)$$

## B. SLA violation

In our working environment, it is suitable to use changeable penalties for SLA violations. This means that the penalty the provider must pay to the customer depends on the 'degree' of violation. Actually, the penalty is determined by the two following variables:

∘ **Time of Violation** $ToV_i$ 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 for the purpose of 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. For our purposes, we have adapted this function in order to obtain result values from 0 to 50:

$$Gom(S_i) = 50 \cdot e^{-e^{-\frac{ToV_i}{100} + \frac{e^1}{2}}}$$

∘ **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 limited with an upper asymptote of 25:

$$Sat(S_i) = \begin{cases} MoV_i \cdot 5 & \text{if } MVRT(S_i) <= \\ & MoV_i \cdot 5 <= \\ & MVRT(S_i) \cdot 6 \\ 25 & \text{otherwise} \end{cases}$$

Finally, to calculate the penalty, 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 the provider to lose more than 75% (50% of the $ToV_i$ and 25% for the $MoV_i$) of the price that the customer pays. We apply this policy in order to guarantee a minimum revenue for the provider, as it is usual in real environments.

## C. SLA criteria

As commented before, we have the necessity of monitoring all running web servers in order to know its performance. It determines the main operation of the whole CHP because, through this monitoring, we are going to be able to resize correctly the number of back-end servers and, thus, adapt the web application's environment to its needs. Really, what it is of our interest is to determine if the performance of any web application deployed is not the agreed in the corresponding SLA. This means to answer the next question: Are we violating the SLA of any of the web applications deployed?

We have created the function $MVRT(S_i)$. 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 $MVRT(S_i)$.

Thus, we want to avoid (or minimize) that the response time of any web application is within the *FAIL* range. This fact has the direct implication that we are violating the SLA and we should avoid it as far as possible.

In a general view, the procedure of the scheduler implemented according to the previously defined ranges is:

∘ If the response time is in *GOOD* range, we follow monitoring the servers' performance.

∘ If it is in *TOLERABLE* range, we start an SLA-preventive mechanism that create a VM due to it is very likely that we need it in the near future. In this way, we will have an outsourced server ready when it will be needed.

∘ Otherwise (it is within the *FAIL* range), we start comparing, at all time, the penalty that the provider have to pay with the cost of outsourcing a new VM. When the penalty overcomes this cost, then the CHP reconfigures the given application with a new back-end server. If not, we decided that the provider pays the penalty in order to maximize its revenue.

### *D. EMOTIVE Cloud*

We have used the EMOTIVE (Elastic Management of Tasks in Virtualized Environments) [22] as the Cloud Service Provider for experimenting with our system. It is a middleware which allows executing tasks and providing virtualized environments to the users without any extra effort in an efficient way. Furthermore, it is a virtualized environment manager which aims to provide virtual machines that fulfills with the user requirements in terms of software and system capabilities.

It is mainly composed by three different layers: the data infrastructure, the node management (VRMM), and the global Scheduler. Specifically, the CHP has been integrated with the VRMM component, the one that allows us to create customized virtual machines and destroy them whenever our system needs.

## IV. Experimental Environment

We use Apache Tomcat v5.5 [23] as the back-end servers. It is an open-source implementation of the Java Servlet and JavaServer Pages technologies, hence it is a pure Java web server. We use Squid [24] as the proxy server. It supports HTTP and HTTPS protocols, among others, and is capable to carry out an efficient load balancing. Moreover, the experimental environment includes the deployment of the SPECweb2005 banking application on Tomcat. SPECweb2005 [25] is a benchmark for evaluating the performance of World Wide Web Servers. Its goal is to evaluate a system's ability to act as a web server. The web application used is based on Internet personal banking. All the requests performed are based on SSL. In addition, the workload for the experiments has been produced using Httperf [26]. The workload's requests generated were extracted from a characterization of the SPECweb2005 client emulator. For all the tests, we configured an average user think time and a connection and client timeout of 10 seconds. All the back-end servers are encapsulated in a Sun JVM v1.6 (with a maximum heap size of 512MB) and run on local or virtual machines with one processor unit and 512MB of memory. All the machines are connected through 1 Gbps Ethernet interface and run Xen Linux with kernel 2.6.18.

Finally, we use EMOTIVE as the Cloud Service Provider (see Section III-D).

## V. Experimentation

In this section we analyze how works the system presented and how it is able to maximize the revenue of the provider. Firstly, we present how the CHP acts in front of a real and variable workload of one day. It has been extracted from [27]. On the other hand, we have performed a set of tests with several fixed value of user sessions to determine the gain obtained in each case. In addition, we examine the performance offered by the web application deployed on different hosting environments.

### *A. Real workload of One Day*

In this section we want to show what is the functionality of the CHP in front of an incoming and variable load. The workload used in this experiment represents the typical one that the web sites of nowadays receive.

As shown in Figure 2, the CHP readjusts the number of back-end servers to the web application's needs. When the servers' response time overcomes the minimal time of violation and the cost of outsourcing is less than the penalty, the CHP commences setting up new outsourced servers. As you can see in fourth subfigure, the CHP obtains better revenue at all time. Note that in third and fourth subfigure, the CHP is represented by a continuous line, the one static server by a dotted line and the four static servers by a dashed line.

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

Table I
Cost of execution, penalty and revenue comparison

| | Avg.CoE | Avg.Pen | Avg.Rev |
|---|---|---|---|
| CHP | 11 | 0.16 | 88.84 |
| 1 server | 6 | 46.1 | 47.81 |
| 4 servers | 21 | 0 | 79 |

The average revenue of the CHP is around 89 currencies, whereas if we have the number of servers required to meet the highest peak (4 servers), the gain

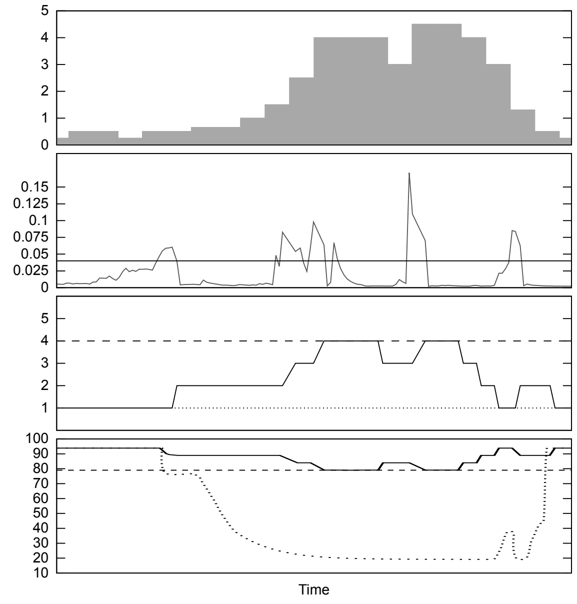

Fig. 2. Input load (sessions/second); servers' response time (milliseconds); number of back-end servers and revenue earned by the CHP, one static server and four static servers (monetary units), from top to bottom.

is around 79 because we must pay for maintaining the servers. 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 2 is the same that a given web application receive every day. Considering an entire year, we would earn 17450 monetary units if we only have one local server; 28835 currencies if we have enough local resources to be able to attend the peak load; and 32430 with the CHP. This fact results in an increase of 85.8% and 12.5% of the revenues of having one and four static servers, respectively.

## *B. Revenue Maximization*

The aim of this second study 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 3 shows the revenue, expressed in monetary units, earned by the three different scenarios with an online-bank application deployed and as a function of the number of user sessions.

Before entering in detail in the results, note that the price that the client pays to the provider 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 a local server in all the configurations tested. 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 itself. On the other hand, the average revenue obtained by the static configurations is penalized by the

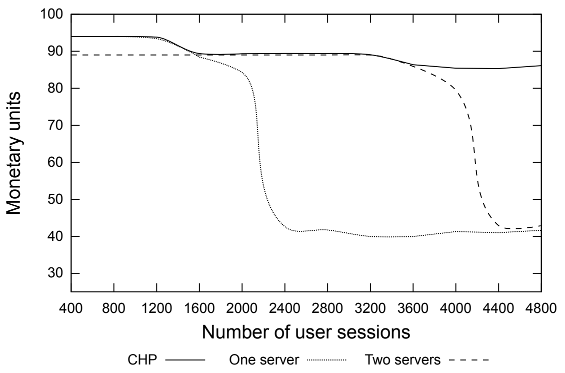

Fig. 3. Average revenue comparison between CHP and static configurations as a function of the number of user sessions

fact of having to pay at all time the cost of maintaining the number fixed of back-end servers, whether local or outsourced.

We want to highlight that the main difference of having or not CHP is 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.

Finally, note that the revenue does not decrease more than 40 monetary units in any case. The reason is because the penalty that the provider has to pay is proportional to the time that the SLA is violated, and this time is limited in the tests performed. Despite this, with an unlimited time of experimentation, the average revenue in static configurations would tend to the minimum when the servers become overloaded, which in this case is 25.

## *C. Performance: Throughput and Response Time*

To conclude, in this section we evaluate the performance offered by a given web application deployed on different hosting scenarios. In fact, these scenarios are the same that those of the previous section.

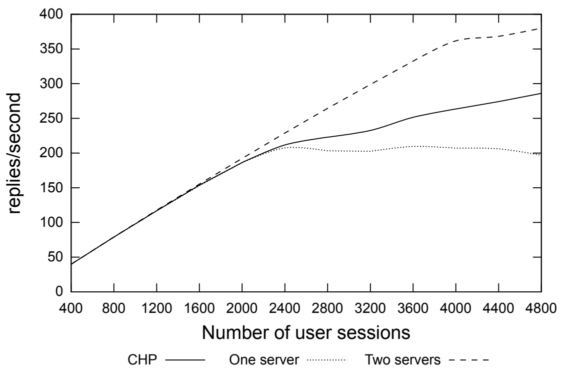

Fig. 4. Throughput comparison between CHP and static configurations

Figures 4 and 5 show, respectively, the throughput, expressed in replies per second, and the response time (in seconds) for different incoming user sessions when the online-bank web application is running in the three contexts.

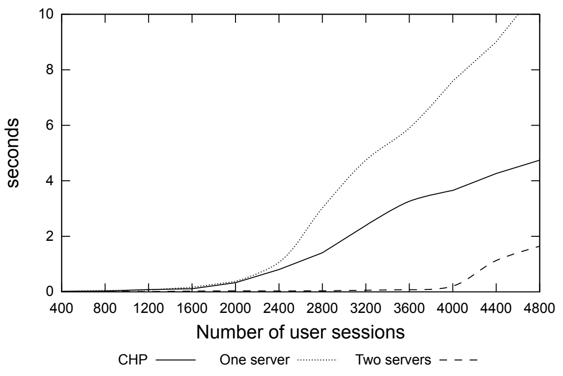

Fig. 5. Response time comparison between CHP and static configurations

It is obvious that with static configurations the web application deployed can achieve better performance in some cases. It is a normal situation because our goal is not to maximize the performance of a given application. For this reason, in some cases the performance is diminished due to we are seeking the revenue maximization. Nevertheless, we show these results in order to demonstrate that the performance offered by the CHP is good enough to meet the quality of service specified in the SLA signed with the client.

## VI. Conclusions & Future Work

In this paper we have presented the use of cloud computing for web hosting providers: the *Cloud Hosting Provider* (CHP). Firstly, we have exposed its architecture and how we take advantage of Cloud Service Providers like EMOTIVE for having a boundless amount of available resources. Thus, we are able to provide scalability and high availability to the web applications deployed. Secondly, we have experimented with the system expounded and have demonstrated that we achieve our main goal: the revenue maximization for the provider, while guaranteeing, as much as possible, the Service Level Agreement signed with the client. In fact, the profit obtained by the CHP is always better than any other configuration without using it.

The future work goes through the integration of the CHP with other cloud service providers like Amazon EC2 [28]. On the other hand, we are considering the option of experimenting with 'cloud' applications like *cloudstone* [29].

## Bibliography

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

[3] H.R. 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.*

[4] A. Greenberg, J. Hamilton, D.A. 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.*

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

[7] PA 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, pp. 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] LP Baldwin, Z. Irani, and PE 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] Jim Gray, "Distributed computing economics," *Queue*, vol. 6, no. 3, pp. 63–68, 2008.

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

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

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

[17] 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.*

[18] 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.

[19] J. Ejarque, M. Palol, I. Goiri, F. Julià, J. Guitart, R.M. Badia, and J. Torres, "Sla-driven semantically-enhanced dynamic resource allocator for virtualized service providers," *eScience, IEEE International Conference on*, vol. 0, pp. 8–15, 2008.

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

[21] M. Macias, 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] Emotive Cloud - Barcelona, ," www.emotivecloud.net.

[23] Apache Tomcat, ," tomcat.apache.org.

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

[25] SPECweb2005 benchmark, ," www.spec.org/web2005/.

[26] Httperf tool, ," www.hpl.hp.com/research/linux/httperf/.

[27] L. Bent, M. Rabinovich, G.M. Voelker, and Z. Xiao, "Characterization of a large web site population with implications for content delivery," *World Wide Web*, vol. 9, no. 4, pp. 505–536, 2006.

[28] Amazon Elastic Compute Cloud (Amazon EC2), ," aws.amazon.com/ec2/.

[29] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," in *1st Workshop on Cloud Computing (CCA)(October 2008).*