Inf Syst Front (2014) 16:867–885
DOI 10.1007/s10796-012-9387-4

A methodology for the evaluation of high response time on E-commerce users and sales

Nicolas Poggi · David Carrera · Ricard Gavaldà · Eduard Ayguadé · Jordi Torres

Published online: 6 October 2012

Abstract The widespread adoption of high speed Internet access and it's usage for everyday tasks are causing profound changes in users' expectations in terms of Web site performance and reliability. At the same time, server management is living a period of changes with the emergence of the cloud computing paradigm that enables scaling server infrastructures within minutes. To help set performance objectives for maximizing user satisfaction and sales, while minimizing the number of servers and their cost, we present a methodology to determine how user sales are affected as response time increases. We begin with the characterization of more than 6 months of Web performance measurements, followed by the study of how the fraction of buyers in the workload is higher at peak traffic times, to then build a model of sales through a learning process using a 5-year sales dataset. Finally, we present our evaluation of high response time on users for popular applications found in the Web.

Keywords Resource management · Web performance · Business modeling · Conversion rates · E-commerce

N. Poggi (✉) · D. Carrera · R. Gavaldà · E. Ayguadé · J. Torres
Technical University of Catalonia (UPC),
c/ Jordi Girona 29, 08034 Barcelona, Spain
e-mail: npoggi@ac.upc.edu

D. Carrera · E. Ayguadé · J. Torres
Barcelona Supercomputing Center,
c/ Jordi Girona 29, 08034 Barcelona, Spain

1 Introduction

Over the last years, high-speed Internet access has become a commodity for home and at work in many countries, with numbers reaching 91 % in the US (Akamai 2009) and in Europe (OECD 2009) at the workplace. High speed Internet access is changing the way users interact with websites, their expectations in terms of performance (response time), and patience levels (Rheem 2010). Performance is perceived by users as the time it takes web pages to render and be usable. A recent consumer report by Forrester Research (Akamai 2009) shows that users expect Web sites to load faster than in previous years. About 23 % of dissatisfied online shoppers attribute their dissatisfaction to slow Web sites, while 17 % to crashes or errors. Moreover, at conference series as the *O'Reilly Velocity* conference, Web industry leaders such as Google, Bing, AOL, and Amazon have been releasing results on how performance affects their business. Google reports that by adding half a second to their search results, traffic drops by 20 % (Mayer 2009). AOL reports that the average page views can drop from 8 to 3 in the slower response time decile (Artz 2009). Amazon reports that by adding 100 ms, sales drop by 1 % (Mazzucco 2010). Furthermore, Google has announced (Google 2010) that it takes into account response time in their page ranking algorithm, affecting positioning on search results, a major income source for online retailers. Web site performance has become a key feature in determining user satisfaction, and finally a decisive factor in whether a user will purchase on a Web site or even return to it.

Therefore, capacity planning techniques for online Web sites are living a period of change. The cloud computing paradigm and the appearance of advanced

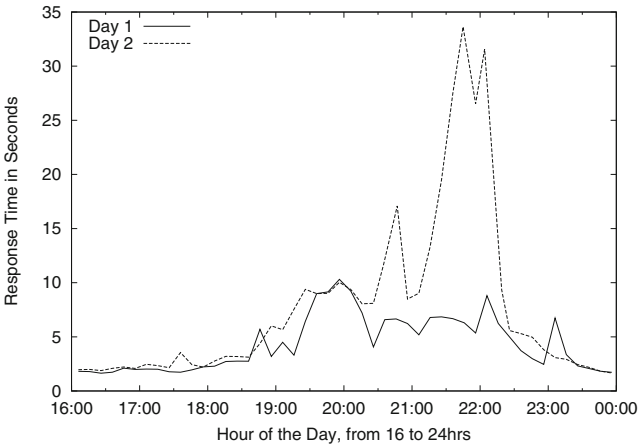
service management products that dynamically adapt provisioned resources, pose new challenges at deployment time. System administrators need to make important decisions about the different parameters that seriously affect the business results of any online Web site. Decisions such as: what is the best performance goal for each online application, usually presented in the form of a response time objective? What is the highest number of servers that the company may afford on peak hours and other workload surges? What would be the effect, in business terms, of limiting the resources for an application and degrading its performance slightly to reduce the bill of the hosting? In this paper we propose a methodology that provides answers to these questions without the need to manipulate the systems in production, as it only requires offline information usually collected by most online Web sites.

Determining the impact of high response time on sales and business volume is something that can be studied injecting delay on pages and using A/B testing methodology to compare user response to different versions of the same page or process, but this approach is not always feasible. Very large companies can intentionally degrade the performance delivered by a fraction of their servers with minimal impact for their business in order to study the effects of performance degradation on their business balance. The same process may have a strong impact for small companies and thus, they hardly decide to use such an approach. Therefore, new methods must be developed to carry on these studies with minimal interference on production systems.

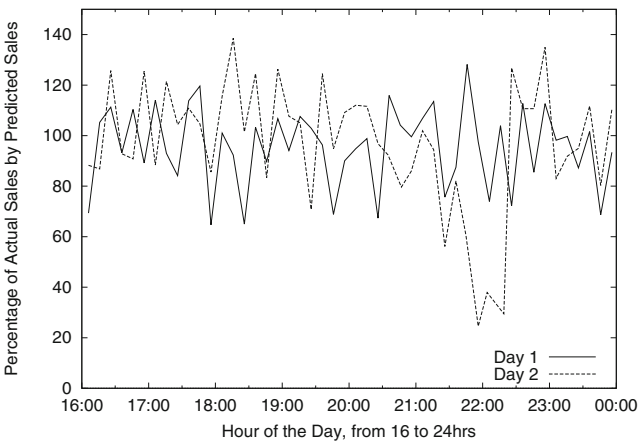
In this paper we present a novel methodology for studying what the total volume of sales lost for an online retailer is, during performance degradation periods. We use Machine Learning techniques to predict the expected sales volume over time and look for deviations over these expected values during the overload periods that introduce performance degradation. Using this technique, we can estimate the total impact of high *response time* in the business activities of an online service in a non-intrusive way. The proposed methodology starts with the study and characterization of the sales dataset, and, using Machine Learning techniques, constructs a model of sales that takes into account the seasonality of sales. Such a model allows for an accurate prediction of expected sales in short time frames. The model is then used to contrast actual sales with expected sales over time, and determine the impact on sales during overload periods that caused degraded Quality-of-Service—measured in the response time—by the online applications.

1.1 High response time example

For the sake of clarity, we include here a simple example that shows the use of the methodology introduced in this article, to a real Web application. Figure 1a shows response time for the same application over an 8hr period for the same day of two different weeks. *Day2* corresponds to a day in which some overload was observed, and as a consequence, performance degradation resulted in a response time surge. On the other hand, *Day1* corresponds to a regular day. A question one may want to answer after observing such performance degradation is: what is the volume of sales that was lost due to response time surge? The result of applying the proposed methodology to the previous example can also be seen in Fig. 1b, where the percentage difference of actual sales obtained to predicted sales are shown. In Fig. 1b, predicted sales by our technique



(a) Observed response time



(b) Relative error between actual sales and predicted sales

Fig. 1 Comparison of response time and predictions in normal and overload operation for two days

corresponds to 100 %, what is plotted is the percentage difference from predictions to actual sales. As it can be seen, the expected sales where higher than what was observed during the surge and the number of lost sales can be quantified. In the following sections we will further elaborate on how to systematically build the proposed sales model to estimate sale loss during overload periods. The model would capture the ratio of web sessions that purchase on the site for every analyzed time and date. Where time frames should be short, to reflect the current QoS given to users by the web infrastructure, that might change by the minute, and sales be prone to seasonality effects as described in later sections.

1.2 Convertion rates

In Internet marketing, the conversion rate (CR) can be generally defined as the ratio between the number of *'business goal' achievements* and the number of *visits* to a site. In the scope of this paper, a *goal* is achieved when a customer purchases a product offered by the company during a browsing session. CR is one of the most widely used indicators of business efficiency on the Web. A high CR indicates that a high proportion of visitors purchase on the site on each visit, while a low CR indicates that most visits use server resources without returning value in terms of direct revenue. Many factors can affect the conversion rate, e.g. type of product, content, brand, SEO ranking, affiliation, availability, and QoS measured in response time. Values for CRs are different for each Web site, and are part of their marketing strategy and business nature; however values should remain within the same range over time for similar products on different sites.

1.3 Contributions

The main contributions of this article are three: first, being the major contribution of the paper, a methodology that uses of a sales model, built using *machine learning* technologies with the goal of quantifying sale losses due to overload periods and response time surges; second, the statement that response time affects in roughly linear way the number of sales and can itself be predicted for similar workloads; and third, the study of CRs for a real company, an online travel agency, showing results for peak load periods not seen in previous literature. To our knowledge, such application of a sales model has not been published before. The proposed methodology is composed of the following four steps, that can be systematically followed to build and use the sales model mentioned above:

- Step 1: Workload and performance characterization of the supplied datasets to understand user behavior and the underlying causes of high response time. (Section 3).
- Step 2: Study the conversion rate variation during the day for the OTA based on sales datasets (Section 4).
- Step 3: The construction of a model to predict sales with *machine learning* techniques and its validation (Section 5).
- Step 4: Characterizing response time thresholds of user satisfaction (satisfied, tolerating, frustrated) for the OTAs applications (Section 6).

The output of Step 4 may be used to build autonomic resource managers that dynamically adjust the resources allocated to each different online application in cloud-like environments while observing conversion rates, performance, and energy constraints. Such use of the proposed methodology is our current subject of research. The results presented in this paper are part of a wider study initiated in Poggi et al. (2009), where machine learning techniques are leveraged to predict anonymous Web visitor's value for the site, and prioritize their sessions on the server to shape its QoS accordingly.

1.4 Online travel and booking in figures

Online Travel Agencies (OTAs) are a prominent sector in the online services market. According to the 2008 Nielsen report on Global Online Shopping (Nielsen 2008), airline ticket reservation, represented 24 % of the last three month online shopping purchases, hotel reservation, 16 %, and event tickets, 15 %; combined, they represent 55 % of global online sales in number of sales. In our study, we take the case of Atrapalo.com, a top national Online Travel Agency and Booking Site representative of the OTA industry, that features popular E-Commerce applications found in the Web. We have been given access to a 5-year sales log of the OTA, including the times and volume of each operation, as well as several week access and resource usage logs of the company's execution environment. Each dataset features several million HTTP requests, from February to August 2011, some of it comprising overload periods.

1.5 Paper organization

The rest of the paper is structured as follows: Section 2 describes some of the main characteristics of the Web application used by the OTA, their execution environment. Section 3 presents the datasets made available to perform this study and their characterization. Section 4 studies application conversion rates and seasonality effects on sales.

Section 5 describes our prototype implementing Machine Learning predictors to forecast sales and compares classifiers. Section 6 analyses the effect on high response times on users and sales and presents results. Section 7 presents the discussion of obtained results and provide insight and domain knowledge from the OTA business. Finally, Section 8 discusses related work and Section 9 the conclusions of our work.

2 The OTA's E-commerce scenario

Online Travel Agencies such as Atrapalo.com, present a wide range of *products* i.e. flights, hotels, cars, restaurants, activities, vacational packages (or 'trips'), and event booking. For this purpose they rely on a wide range of technologies to support them: dynamic scripting, Javascript, AJAX, XML, SSL, B2B Web services, Caching, Search Algorithms and Affiliation. This results in a very rich and heterogeneous workload. While these technologies enhance users' experience and privacy, they also increase the demand for CPU and resources on the server side. Furthermore, as web applications become more resource-intensive added to the increasing number of potential visitors to a site, system overload incidence grows along (Power 2010). Moreover, visits to travel sites might not depend only on their popularity but to the season e.g. high and low seasons, holidays, search engine ranking (SEO), linking, and the proximity of an event, such as a concert (see Section 4.2). The variability of the traffic creates a bursty workload, making workload characterization and modeling crucial for devising cost effective infrastructures, preventing denial of service, and improving users Quality of Service (QoS) across the application.

2.1 Products

For Atrapalo.com, some of the above mentioned products inventories are maintained internally (e.g. 'restaurant booking'), some are completely external (e.g. 'flights'), and some other products (e.g. 'hotels') are a mix of internal and external providers. Access to the external providers is performed via B2B Web services, which causes QoS to be dependent on external sites for some operations. The company itself is also a B2B provider for some customers and meta-crawlers, acting as their provider via a Web Service API. The application relies on state-of-art caching rules, to reduce request times and load generated by the external transactions. Although the company's main presence and clientele is in Europe, about 20 % percent of the visits are from South America where it has some offices, and a few other visits from the rest of the world. It is important to remark that the site is a multi-application Web site. Each *product* has its own independent application code base and differentiated resource requirements, while sharing a common programming framework.

2.2 Flights product: External requests, meta-crawlers, and caching analysis

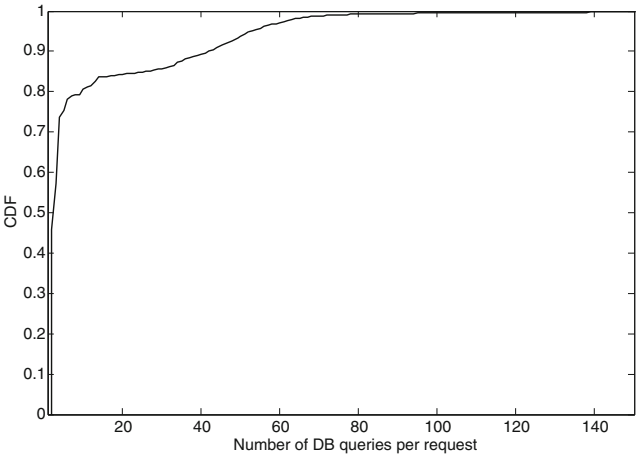
In this section, we will analyze the characteristics of the OTA's most popular product, *flights*, its interaction with external sources and the inner functioning of the product from a technical and business perspective. As an example of external B2B requests, when a web customer makes a search for flight availability, before returning the final HTML, several B2B providers are queried via web services and their results are processed and ordered according to different algorithms of product placement, in a resource intensive operation. In general, each flight availability is costly not only in terms of resources, but by the actual economic cost of each B2B transaction set in the contract between the OTA and the provider, often referred as Global Distribution Systems (GDS). For this reason, meta-crawling sites such as Kayak and Travel Fusion, *scrap* travel websites' results simulating real user navigation, comparing results from different sites and avoiding transactions costs and contracts with GDSs. This situation creates the necessity to automatically identify and sometimes ban such meta-crawlers (Poggi et al. 2007a), as the conversion rates (see Section 1.2) for this requests are more than 10 times lower than direct visits. One commonly used strategy across the OTA industry to reduce costs, is to heavily rely on result caching, this way preventing excessive searches from meta-crawlers and speeding up results for users. However, caching can result in outdated availability, sometimes preventing the *bookability* of flights or even in increased prices, and the user in general finds about these changes in the moment of booking, after completing forms, compromising user satisfaction. The caching strategy to be adopted by an OTA has to be used effectively to lower response times, while providing accurate results

for users. Another issue arising from external transactions is that the QoS for the application becomes dependent on an external source, and when external providers offer low QoS, the site's application is also affected. To alleviate this situation—besides relying on caching—timeouts for connections are set by the company and sometimes the application would switch or disable specific providers in case the there is more than one available to choose from with similar results.

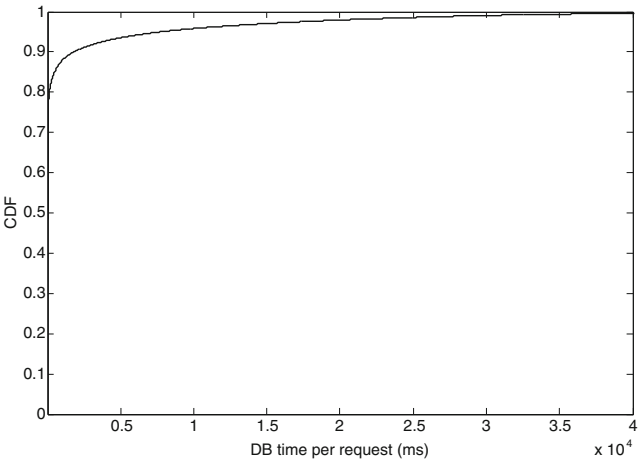
Looking at the interaction between the OTA and external information providers, it has been observed that the probability of accessing an external provider follows a Binomial distribution with parameters $n = 1,125,969$; $p = 0.1306$ for the *flights* product. For those requests that did involve access to an external site, Fig. 2c shows the CDF of the time spent waiting and processing the information provided by the external source. As it can be derived from this information, caching techniques are effectively used for this application, avoiding in many cases (more than 75 %) the cost of collecting information from external providers. For the cases in which accessing an external provider is required, the process is usually completed in less than 20 s. Further discussion on CDFs can be found at the end of Section 3.3.1.

2.3 Computing infrastructure

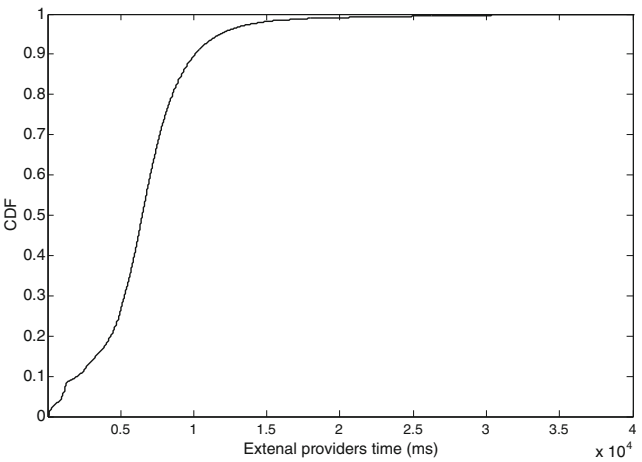
To understand the infrastructure on which performance measurements are taken from this section describes the physical infrastructure on which the applications are deployed. At the time of writing, Atrapalo.com infrastructure is composed of about 40 physical servers running Debian GNU/Linux, connected via Gigabit switches, including: a set of redundant firewall and load-balancer servers acting as entry points and SSL decoders; about 15 dynamic web servers; five static content servers; two redundant file servers, eight high end database servers including masters and replicas running MySQL; plus auxiliary servers for specific functions such as monitoring and administrative purposes. The web application runs on the latest version of PHP on Apache Web servers; load is distributed using a weighted round-robin strategy by the firewalls according to each server capacity. Access to databases is balanced by using DNS round-robin rules for replica slave servers, most of the READ/WRITE strategy and data consistency is performed in the application itself, which also caches some queries in memory and local disc. Static content such as images and CSS is mainly served by Content Distribution Networks (CDNs), to reduce response time on the client end; the rest of the static content is served by servers running *lighttpd*. Notice that static content is not part of this study.



(a) DB queries



(b) DB response time



(c) External provider response time

Fig. 2 CDF of resource consumption for most popular product during a stationary, high load, 500-min period

In previous work (Poggi et al. 2007b) we have determined that for every requested dynamic page, about

13 static resources are accessed in average. There are several caching mechanisms in place, for web content: there is a reverse-proxy caching static content generated by the dynamic application running *Squid*; there is a per-server caching HTML template caching; distributed memory key-value storage (Redis and Memcached), database query cache and scheduled static HTML page generators.

3 Step 1: Datasets and workload characterization

The next sections, present the different datasets provided by Atrapalo.com from their production environment and the characterization of their most relevant features on Web user sessions and their resource usage. We categorize datasets in: *sales datasets*, which present sales over the last 5 years; and *performance datasets*, custom generated performance logs of six months of detailed per request performance measurements. To comprehend the causes of high response time, the first step is to understand the workload, and how user sessions consume server resources and the external dependencies involved in providing a high QoS.

3.1 Sales datasets

Sales dataset are used for modeling sales through Machine Learning techniques, as described later in Section 5. For the purpose of this study, we were given access to sale history datasets for the OTA's different *products*. For each *product* we have the exact date and time for each purchase that was made. However, we were not given access to sales amounts or margins, only the moment of each user sale. Sales datasets range from 01/01/2007 to 09/01/2011, comprising a period of close to 5 years. There is a great variation of sales between products and times of the year due to seasonality effects (see Section 4.2). Vacational products suffer from great variation of sales and visits according to high a low seasons, day of the week, or the proximity of a holiday. Ticket sales vary according to the availability of the event on competing sites and the limited availability, sometimes causing rush periods. Sale volumes for these datasets are not reported due to confidentiality limitations.

3.2 Workload performance datasets

Performance datasets are used to evaluate high response time effects both in user behavior and sales. They were produced through existing probes in the PHP dynamic application and provided by Atrapalo.com. These transactions were collected from February to August 2011, a total of 185 full days that contain 768 million dynamic requests, representing 210 million distinct sessions—of which 110 are one-click sessions—, and 15,488 different pages (non-ambiguous URLs).They contain all kind of system and application metrics from regular HTTP access data, to per-node application resource usage, database, B2B requests and server status. All the information is correlated in time with the sales dataset.

At the end of each executing script, code was added to record regular HTTP data: access time, URL, referring URL, client IP address, HTTP status code, user-agent (type of browser), and replying server. Data from the application itself: total processing time, exact user session id, real page processed by the server (some URLs might be ambiguous or perform different actions), accessed product(s), type of request (regular, AJAX, Administrative, API, batch, etc), CPU percentage, memory percentage and peak memory, CPU time both in *system* and *user mode*, total database time, total external request time. As well as current server load (*load average*) and number of opened Apache processes.

We also had access to the Apache logs and monitoring system access for the days covered in the dataset and other random weeks of 2009, 2010 and 2011. These auxiliary logs have been used to validate and explain obtained results from the workload. Notice that the studied datasets do not include pages that were cached by the reverse proxy or by the user's browser.

These performance datasets present several features not available in most Web workload characterizations, such as the dependency of external providers, database access and mix of differentiated products (multi-application site). Results from this paper will support the future building of a workload generator that is able to simulate the real life characteristics of complex workloads such as the one presented here.

3.3 Workload characteristics

On our recent work (Poggi et al. 2010), we have performed a complete workload and resource characterization of a similar, but shorter, 1 week dataset of 2009. This section provides the most relevant features of this new dataset, for a more in-depth characterization refer to the previous paper. For the new dataset, the mean page view for the whole dataset is 5.1 pages per session, with 6:45 min spent on the site, an average of 1.5 s response time for dynamic page generation, and 10MB of RAM memory consumption. The highest traffic day for the OTA is on Monday and then traffic decreases

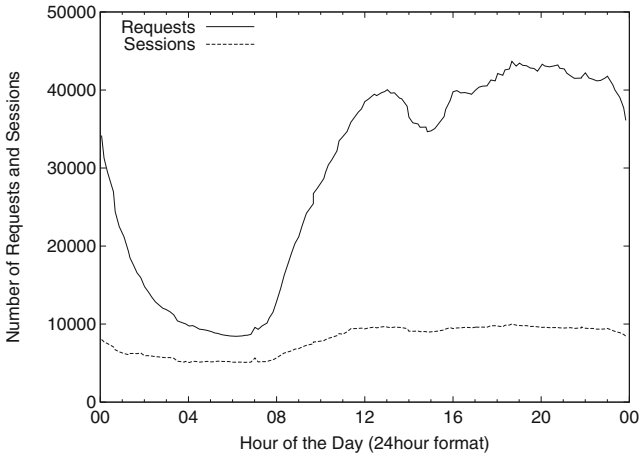


Fig. 3 Number of requests and sessions for 24 h period

to the end of the week. The opposite effect is observed on average *page views* as well as the time spent on the site; they both increase during the week, peaking at the weekend, from: 5.82 and 6:40 on Mondays to 6.27 and 7:31 on Sundays, page views and time spent respectively. An interesting feature of the dataset is that mobile devices represent over 6 % of the total visits at the end of the dataset while this figure was not significant the previous year.

Figure 3 presents the number of *requests* and *sessions* for an averaged 24 h day from the full dataset. As it can be observed, traffic starts growing from 8 to 14 h, lunch time in Spain, where most of the traffic comes from, then it decreases until 16, return to office hours, and continues to grow peaking around 22 h. After midnight, traffic volume decreases until it finally reaches the beginning of the cycle again.

$$f(x) = \sum_{i=1}^{5} a_i * \sin(b_i * x + c_i) \quad (1)$$

The characterization of the normalized shape of the mean request rate for a 24 h period, in 1 min groups can be done following the Sum of Sines expression found in Eq. 1, with the parameters described in Table 1.

Table 1 Variables of the normalized request rate function

i	a	b	c
1	8.297	0.002157	1.134
2	8.072	0.002325	4.232
3	0.1572	0.009136	1.542
4	0.04958	0.01732	2.356
5	0.02486	0.02197	2.045
R-Square: 0.9701			

3.3.1 Workload mix

The workload is composed of several different request types, and for each page view that the user finally sees on his browser, several dynamic requests may have been executed. In the studied dataset we have identified the following request categories: Regular user page 46.8 %, AJAX 18.4 %, dynamically generated Javascript 6.7 %, dynamically generated CSS 4.6 %, HTTP redirect page 11.9 %, Administrative 5.5 %, internal scheduled batch 2.2 %, API Web Service 11.3 %, and Extranet 6.9 %. It is an important feature of this dataset (and probably other real-life logs) that less than 50 % of the total dynamic requests corresponds to user clicks on their browsers.

As a brief analysis on the number of *crawler* and *meta-crawlers* requests by looking at the *agent* field (the reported browser type) that identify themselves as bots. Our findings indicate that the number of *bot* requests is about 15 % of the total traffic. This is consistent with previous work on a similar dataset 5 years before (Poggi et al. 2007a) that identified between 10 and 15 % total bot content. Even more traffic may correspond to crawlers and meta-crawlers assuming that some might simulate being a real user when accessing the site, which would show a growing trend in the proportion of automated bot traffic.

Our study concluded that 52.38 % of the sessions only contained 1 hit, that is, the user only accessed 1 page, and then left the OTA site. This fact is mainly due to many visitors reaching the site through external banners that redirect them to especial *landing pages*, and many of these users do not continue browsing the OTA site after this initial page. Furthermore, most crawlers start a new session with each request as they might not send *COOKIES* with the request.

In Fig. 2 we pick the most popular product of the OTA, *flights* and characterize the interaction of its code with both the database tier and external providers of information. The characterization is done by building the CDF of each metric, what can be approximated using the functions seen in Table 2. All the services related to this product require accessing at least once at the DB tier. Figure 2a and b show the CDF of the number of DB queries per request and the time spent per request waiting for the result of DB queries. Recall that this information corresponds only to the most popular product of the OTA. As it can be observed, 50 % of the requests issue 1 or 2 queries to the DB tier, and around 80 % of the requests require less than 10 queries to complete. However, a significant fraction of the requests produce complex results and require a large number of DB queries to be completed, reaching

Table 2 DB and external time CDF fits for most popular product

Metric	Model	Parameters
DB time	Weibull	$a = 30141.4$; $b = 0.251286$
DB queries	Generalized Pareto	$k = 0.61979$; $\sigma = 4.65092$; $\mu = -0.1$
Ext. provider time	Logistic	$\mu = 6.17049e+07$, $\sigma = -191940$

more than one hundred DB requests in some cases. Looking at the time spent waiting for data from the DB tier, most of the requests exhibit just a couple of seconds of DB query waiting time, but some cases can reach up to nearly 40 s of DB time. In the building of the CDFs, 1-click sessions were excluded as we want to study customer's characteristics; 1-click sessions are included in the rest of the experiments.

3.3.2 Response time during the day

While studying the dataset, we have noticed that there was a great variation of response time during the day for the different applications of the OTA, as shown in Fig. 7. We have identified for the OTA that *response time increase* was directly related to: the number of concurrent user sessions at a given time at the site, database response time, and external providers response time. Database and external providers response time also increased at peak hours of the OTA as can be seen in Fig. 4, causing part of the slowdown. Some resource contention is also present at peak load times, from Fig. 7 there is contention between 18 and 23 h. Causes of high response time are further developed in Section 6, however surges are not so frequent in the dataset as can be seen in Fig. 5.

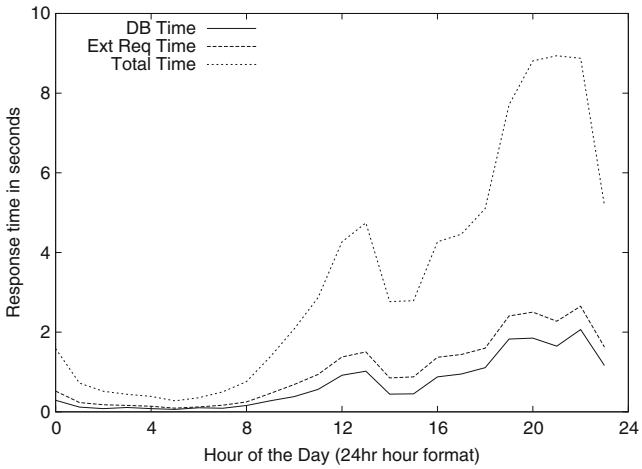


Fig. 4 Variation of DB and external requests time during an abnormal busy day

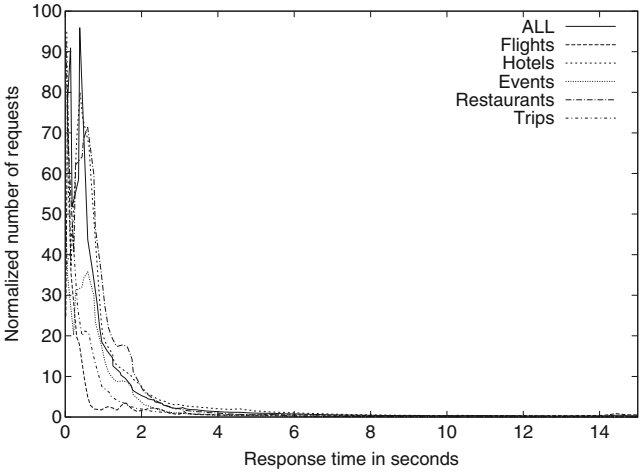


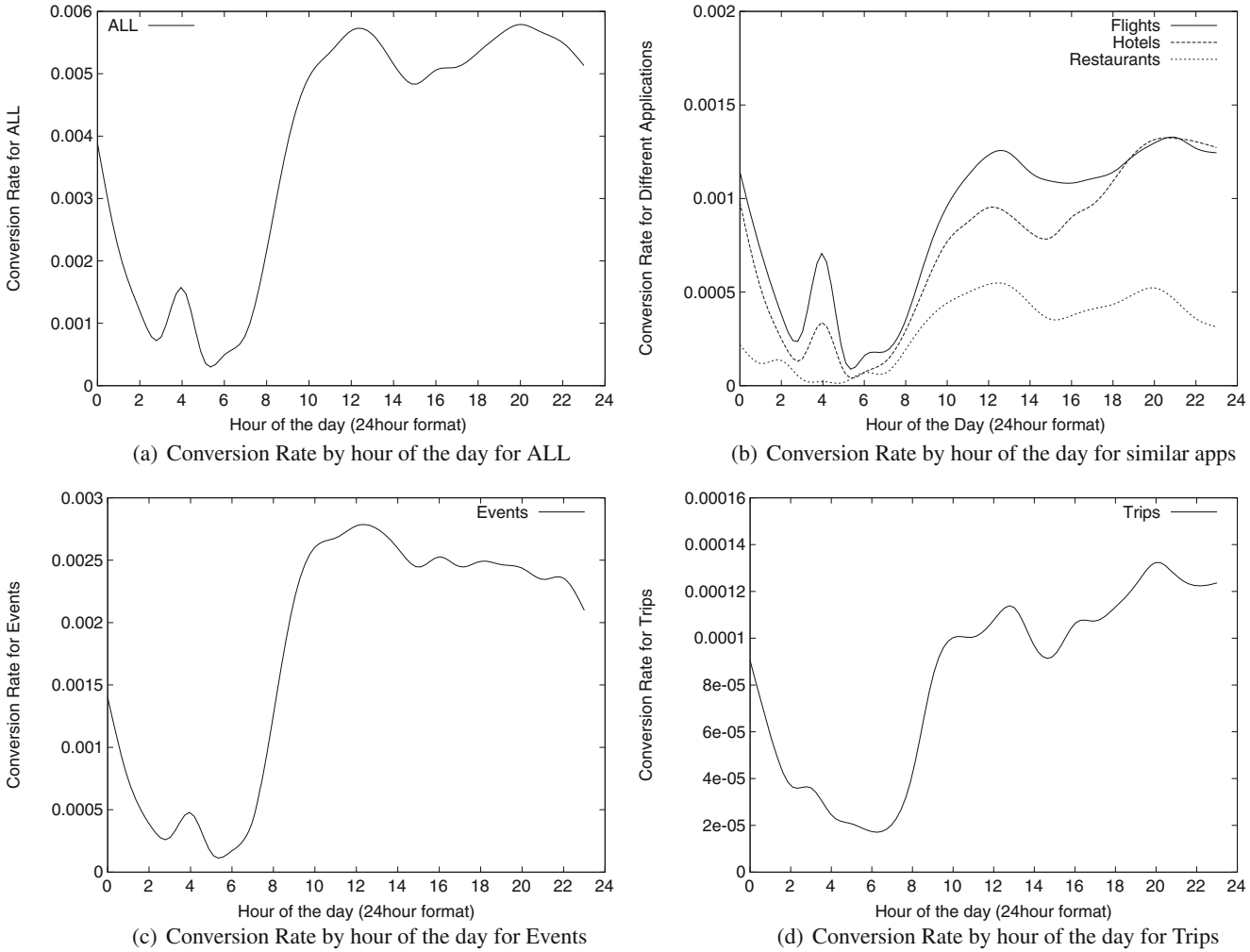
Fig. 5 Normalized percentage of number of requests by response time for different products

4 Step 2: Conversion rate study

The main concepts of Conversion Rates, CRs, were briefly introduced in Section 1.2. The following section provides and analysis of the CR of each product of the OTA during the 6 month period of the *performance datasets*. CRs are calculated from the number of sessions in the *performance datasets* and the number of sales for each product in the *sales datasets* for the corresponding period of time. The objective of such analysis is to understand how selling hotspots are distributed over time for the studied business. Such study is very relevant to measure and quantify the effects of QoS on the sales volume observed for the OTA. Notice that the CR does not necessarily change with oscillations in volume of traffic that can be observed for any Web site (e.g. Fig. 3), as CR has to do with the fraction of visitors purchasing products, and not with the overall volume of visitors.

4.1 Conversion rates by product

Figure 6 presents the CR of the different products of the site; averaged and grouped in a 24 h period. Figure 6a shows the CR of *ALL* the products combined; Fig. 6b groups products with similar CRs; while Fig. 6c and d, *events* and *trips* (vacational packages) respectively.



(a) Conversion Rate by hour of the day for ALL

(b) Conversion Rate by hour of the day for similar apps

(c) Conversion Rate by hour of the day for Events

(d) Conversion Rate by hour of the day for Trips

Fig. 6 Conversion rate by hour for different applications

As an average for all products of the site, the conversion rate is higher at the site's peak traffic time in number of requests. The CR of the averaged application follows very closely the daily patterns of the number of sessions in the system (compare to Fig. 3) during a 24 h period. From 1 am to 8 am there is a low CR, sleep time and also higher percentage of bots and crawler activity. From 8 am until 14 h, lunch time in Spain where most of the traffic comes from, the CR increases progressively during the morning as the number of sessions increase. From 14 until 16 h, the CR lowers during lunch time, and then continues to grow in the afternoon along with sessions, until 18 h where the CR decreases due to end of office hours. It increases again from 20 h peaking around 22 h. A remarkable conclusion must be pointed from the observation of these figures: the CR for all the products available from the OTA follows a pattern that is strongly correlated to the traffic pattern observed in Fig. 3. The interpretation of this fact is that the hours at which the traffic volume is the highest, the fraction of customers that are actually purchasing products is also higher, resulting in the best CR for the day. Recall that such a result indicates that the relation between volume and sales is not constant over time, and leads to a very important increase of sales over peak periods, were not only traffic volume grows but also the average CR. Most users buy at similar time-frames.

Notice that it is a usual case that many infrastructures are not dimensioned for sustaining QoS at peak hours, as they are considered surges in the traffic and static provisioning of resources to manage punctual very high traffic volumes is unaffordable. Of course, such decision results in worse response time and QoS in general during peak periods. Looking at the charts for Atrapalo.com, it can be observed that these are not only surges in traffic, but also the best selling periods of the day. Although industry and consumer studies (see Section 1) report that a high response time has a direct effect on sales, as conversion rates are higher at

peak times, the total loss in sales might not be apparent in most cases. Figure 6b, shows the CR for products with similar magnitudes: flights, hotels, and restaurants. The *flights* application has similar CR during day, while *hotels* has a higher peak than the other applications between 21 and 23 h. *Restaurants* on the other hand, show subtle differences, a morning peak at 13 h, just before lunch time at 14 h with the same happening at 18 h before dinner time. The CR for the *Trips* application on Fig. 6d also follows a pattern correlated with the number of sessions on the system as previous applications; however its magnitude is 10× lower than the products on Fig. 6b. *Trips*, has two pronounced peaks: just before lunch time and at 20 h.

Figure 6c, shows the CR for *events* which has a distinct pattern. It grows fast in the morning and peaks at 12 h then decreases throughout the day. For *events* the most important time frame to provide a high QoS is between 10 and 13 h, while for *hotels* is between 19 and 22 h. Another interesting feature of the CR of *events*, is the magnitude of the CR, which is twice as high compared to applications in Fig. 6b. This might be due to the fact that certain events are sold exclusively by the site. This fact makes this product more inflexible to loss in sales due to poor performance.

4.2 Seasonality

Vacational products such as flight and hotel reservations vary greatly during the year according to season. During spring there is a high season for reservations before summer. The opposite effect is appreciated during fall, where the number of sessions drops significantly. Holidays and school vacation periods also affect touristic products, e.g. Christmas and New Year's Eve, that create peak times during the first days of December. A similar but more condensed effect is seen in ticket booking and as tickets for certain events tend to be limited, a rush of users might flock the site to get hold of the tickets, also creating peak loads. Moreover, as users are a click away from any site on the web, certain events such as mail promotions, high-impact banners, social linking can create peak loads in the workload within minutes.

4.3 Conversion rates as a function of response time

It has been shown above that, in general terms and for most products, high conversion rate times coincide with the rest of the product's peak hours and worst response times. This can be seen by comparing Figs. 3, 6b, 7 and 8. Notice that for each application, its CR will inherently exhibit a different behavior in result

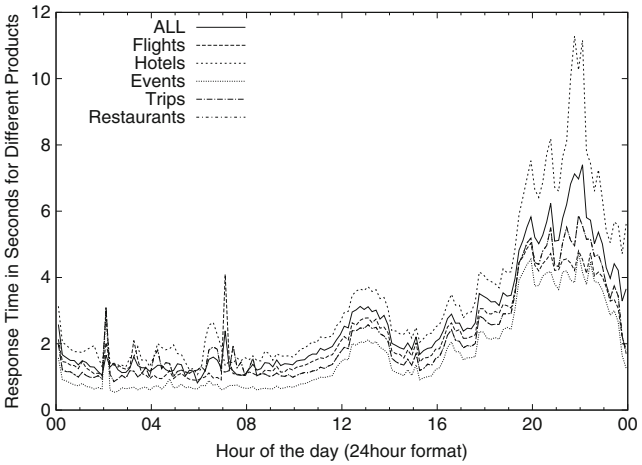


Fig. 7 Average response time by product in 24 h period

of changing QoS, which is caused by the nature of the application.

Figure 8 explores the *conversion rate* as a function of average response time by combining data fom the sales dataset and data from the performance datasets (see Section 3) grouped in 10 min intervals from the performance dataset. By analyzing the figure there is no clear indication that a high response time yields fewer sales. On the contrary, most applications are able to maintain CR and sales even in the periods of higher response times. For instance, *flights* usually involve complex search it would indicate that a high response time maintains or improves sales. To study this effect further, the next section presents our methodology for forecasting expected sales in short time frames for each application. In order to measure how a low QoS during peak times affects more sales than previously reported.

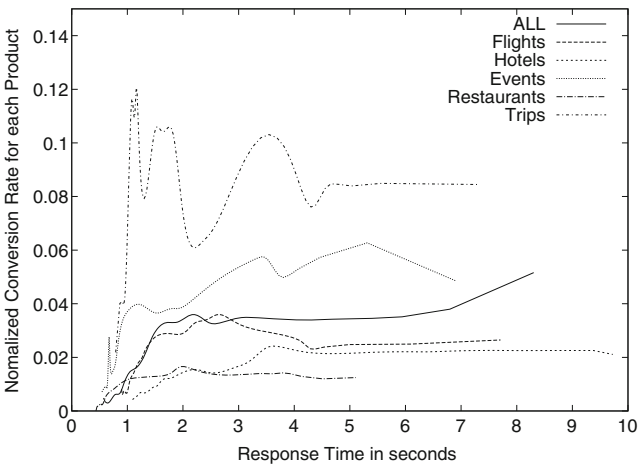


Fig. 8 Conversion rate as a function of response time

5 Step 3: Predicting sales with machine learning

Most traditional sales forecasting techniques involve some form of linear or multiple *regression analysis*. In our preliminary work we have tried several methods i.e. *linear*, *quadratic*, and *cubic* regressions to predict expected sales for different weeks in short periods of time bins, e.g. 10 min, using the *sales* dataset. We found that predictions were too general and not precise enough for the short time frames we wanted to measure response time in. To overcome this situation and improve predictions, we have built a sales forecasting prototype implementing Machine Learning numerical algorithms. The prototype implementation learns from the different CRs during the day and across seasons, with training from the five years of the sales dataset to overcome the effects described in the previous section about peak loads and seasonality effects.

To forecast sales, we have built a prototype based on the WEKA (Hall et al. 2009) open-source Machine Learning package, which contains several ready to use classifier algorithms. As we want to predict expected sales for short time bins, representative of the *response time* (QoS) in the system at that moment, while also having precise prediction results, which depend on the volume of sales per time bin in the *sales* dataset. We have tested predictions for 30, 15, 10 and 1 min bin intervals. One minute bins turned out to be too low to have accurate prediction results, and as Web sessions for buying visits are longer (Poggi et al. 2007b), it only partially reflected the response time for the session. Thirty minutes proved to be too high, as server status might have changed drastically in this time frame. After some experimentation, we have decided to use 10 min as the time frame for the rest of the experiments: it is low enough to represent the current QoS on the system and high enough to cover most of the Web session with accurate sales predictions in our sale's dataset.

In WEKA, a *predictor* model is trained using a specially formatted *training dataset* that should contain the most relevant available variables to predict sales. After training the *predictor* with the training dataset, a *test dataset* with the same format is used to perform predictions. The predictor reads the test dataset, ignoring the *class*—the unknown variable, in our case the number of sales—if present, and according to the training and the algorithm used, outputs a prediction of sales—the class—for each of the time bins.

Figure 9 presents the general process of our prototype implementation. It begins by preprocessing the historic *sales* dataset provided by the OTA, by aggregating entries into the specified time-bin length—10 min in our case—then exporting the aggregated

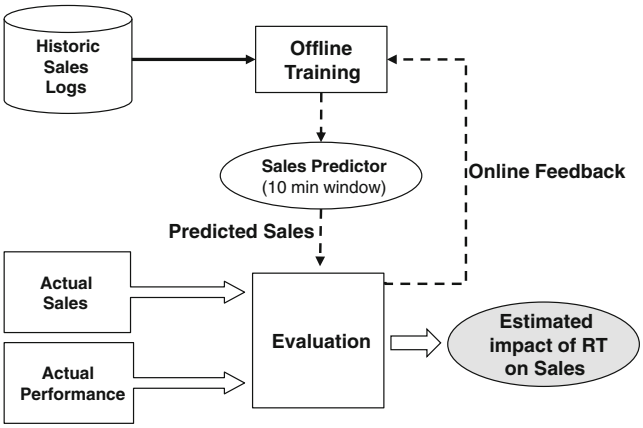


Fig. 9 Sales prediction process

dataset in a format compatible with the sales learner. Following, the sales learner implements the specified machine learning classifier and trains itself to produce the predictor model as an offline process. After the predictor model is trained, for every 10 min bin from the period corresponding to the *performance* dataset, the prototype performs a sales prediction. From this moment two things happen: first, the data on the period corresponding to the time bin, is fed back to the learner module; second, actual and predicted sales are compared and the difference in number of sales is stored along with the current system response time. The following sections detail the prediction process and prototype implementation.

5.1 Prediction methodology

For the training dataset, we have preprocessed the 5 year long sales history dataset (presented in Section 3.1) into 10 min bins, each day containing 144 bins; creating a *training* and *test* datasets for each application of the OTA, and one for *ALL* the applications combined. Resulting predictions should be as close as possible to the actual number, but of course predictions cannot be expected to be totally exact. For this purpose we have implemented in the prototype several *numerical classifiers* available in WEKA, which hopefully predict numerical values with a low percentage of average error.

5.1.1 Attribute selection

After some experimentation selecting attributes the training dataset contains the following attributes:

– Number of sales for the 10 min bin. The *class* being predicted.

– Year of the sale.
– Month of the sale.
– Day of the year.
– Day of the week.
– Hour of the day.
– The 10 min bin for the day, from 1 to 144.

The goal is that each attribute adds valuable information when building the *predictor* model. For example, the month of the year should add information on whether we are in low or high season. Each day of the week has differentiated sales volume: Mondays have more sales and decreases through the weekend for most applications, except for *events* which is higher on Fridays. The same goes for the time of the day, which has different conversion rates for each application (Fig. 6). It is important to remark that the *training* dataset only contains data previous to the *test* dataset.

More attributes could be added to improve predictions, especially to cover seasonality effects e.g. of holidays, where the numbers of days to a vacation period could have been added. However the purpose of the prototype is to provide representative predictions and a proof-of-concept of the method. As a note, most numerical algorithms can improve predictions by using *nominal* values instead of numerical ones, e.g. hour of the day as a category not as a number, although this increases resource requirements and default parameters for the algorithms needed to be tuned to improve predictions. We found only negligible improvements using nominal attributes and the combination of numerical and nominal attributes at the same time.

5.1.2 Prototype implementation

For our prototype implementation we have used the 5 year long sale history dataset to create the training dataset, we cut the dataset the day before we want to apply the predictions, in this case at the beginning of the performance dataset, so no information of the future known when building the model. Our prototype implementation has as a parameter on how long (in days) the test dataset should be and creates several training and test datasets incrementally. As an example, setting as input seven days to the prototype, the first training dataset would contain data from the beginning of the sales dataset, to where the performance dataset starts—the first week of February 2011—. The second training dataset would contain same data as the previous training dataset plus the first week of February, the second test dataset would be for the second week of February, and so on.

We have tested predictions with different lengths, as the shorter the time frame, the better predictions should be, as behavior from previous days is learned by the model and should improve predictions. However, we have noticed that since the sales dataset covers a great time span, for the tested classifiers previous days do not influence or improve results significantly. Moreover, if on the previous day there was a surge, we do not want the model to predict lower values for the next day, but what would be the expected sales taking into account the conversion rate for this product. In a real-time online implementation, the model will be up to date to the previous time bin. Seven days (1 week) was used as test dataset time span for the presented results in the next sub-section.

5.2 Prediction results

The first training dataset used by the prototype to build the *predictor* contained 214,849 instances, while the first test dataset contained 1008 instances, for a total of 24,192 tested instances for the full length of the performance dataset. We present results for the following numerical classifiers found in WEKA: LinearRegression, REPTree, Bagging(M5P), and Bagging (REPTree). More complex classifiers could have been used e.g. *neural networks*, but the processing time required for training seems too high to consider them in a real-time application and were discarded for the time being. Table 3 presents accuracy for the different selected classifiers: *LinearRegresion* is the least performing classifier with a Relative Absolute Error of 64.33 %, while *M5P* and *REPTree* have 24.63 and 24.41 % Relative Absolute Errors respectively. The *Bagging* meta-classifier was also tested implementing both *M5P* and *REPTree*, improving precision for both algorithms to 23.6 % for both *Bagging(M5P)* and *Bagging(REPTree)*. Meta-classifiers split the training dataset in instances, testing different parameters of the selected classifier and selecting the most precise ones, they perform several iterations of the regular classifiers with different attributes and selecting the best for each case, but take longer to train. As a note, all experiments were performed using default values in WEKA. Linear regression has essentially no parameters. RepTree and M5P, like all decision tree methods, have a parameter controlling the size of the tree. We verified that, in our case, optimizing over that parameter gave only negligible advantage over the default value. Figure 10 presents the averaged class results for a 24 h day of actual sales vs. predictions. As it can be seen, *LinearRegression* does not follow closely sales patterns, while the rest of the classifiers correctly detect shifts by time of the day

Table 3 Classifier evaluation

	Linear regression	M5P	REPTree	Bagging(M5P)	Bagging(REPTree)
Correlation coefficient	0.7215	0.9515	0.9465	0.9571	0.9556
Mean absolute error	16.8619	5.892	5.8393	5.6465	5.6517
Root mean squared error	19.9969	8.1515	8.067	7.8087	7.8263
Relative absolute error	64.3319 %	24.6382 %	24.418 %	23.6116 %	23.6333 %
Root relative squared error	65.8654 %	29.9866 %	29.6757 %	28.7257 %	28.7904 %

especially day and night patterns with great accuracy for the different evaluated products.

Although *LinearRegression* was the least performing algorithm, the model is simple enough to illustrate how classifiers work:

$$\begin{aligned} 10\,\text{minute\_sales} = {} & 1.48 \times \text{year} + 0.17 \times \text{month} \\ & - 0.03 \times \text{day} + 0.43 \times \text{day\_of\_week} \\ & + 1.08 \times \text{h} + 0.08 \times 10\,\text{min} + k \end{aligned}$$

where k is a constant not disclosed for confidentiality limitations. Parameters for each variable are dependent on the training dataset values and in this case specific for the OTA and presented as an example.

In this section we have presented our methodology for predicting future sales for short, 10 min, time bins by implementing and testing different Machine Learning classifiers on the sales dataset. Tree classifiers—M5P and REPTree—as well *Bagging* meta-classifier implementing both algorithms, offer high accuracy predicting sales for normal Web site operation. As response time increases, classifiers become less precise and over predict sales from actual results. In the next section we evaluate how high response time affects predictions and user sales.

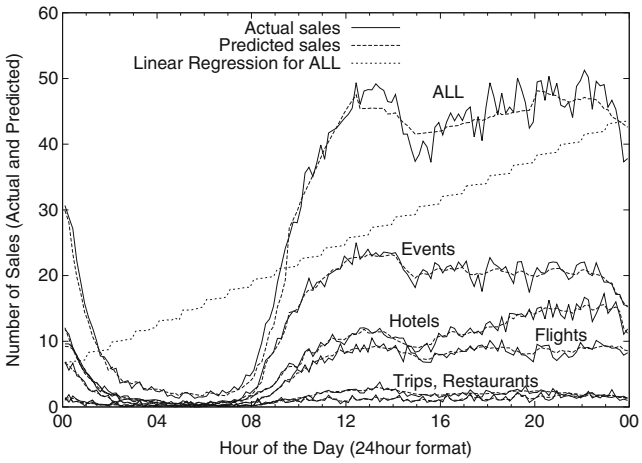


Fig. 10 Classifier precision by App for an averaged 24 h day

6 Step 4: Study of response time effect on users

In Section 4.3 we have shown that the workload's peak times coincide with high conversion rate for most applications of the OTA. Therefore, a loss in sales due to high response time is not apparent (Fig. 8) as the fraction of buyers in the workload is also higher at these times. To counter-measure this effect, on the previous section, we have presented our prototype implementation to forecast expected sales for short time bins of the workload with great accuracy results. However, as response time increases, classifiers over-predict sales; this section analyses this effect further.

Total response time for a request in our dataset, is the time it takes for Web servers to start sending the reply back to the user. It is important to notice that in the OTA's application, *output buffering* is turned *on* for all requests, so no data is sent over the network until the full request is processed and *gzipped*. There is an additional time for the browser to render the final Web page, but it is not present in our dataset and is not part of this study as it depends on the actual HTML, media content, and the user's hardware.

In Section 3, we have seen how Load on the Web servers increases with the concurrent number of requests and sessions. High load on servers usually translates into an increase in response time, which increases for three main reasons: server resource starvation, less dedicated system resources for a request; external B2B requests and database increased response time, low QoS on dependencies; and contention between resources, jobs waiting for blocked resources. Details of these reasons are further developed in Poggi et al. (2010). Furthermore, not all application's response time increases in the same proportion with server load, as each application has differentiated resource requirements, especially for external resources. Going back to Fig. 7, we can appreciate how *Hotels* is the most affected application by server load, as it has a higher number of database queries compared to other applications, and its peak time coincides with external providers worst QoS. While *events* is the least affected application as it has no external dependencies besides the database.

Notice that, as shown in Section 4.3, peak load times coincide with high conversion rate periods for most applications of the OTA, where we have shown that all of the analyzed applications have a corresponding high CR when there are more users on the system and QoS is worst. Therefore, a loss in sales due to high response time may not be apparent as the fraction of buyers in the workload is also higher at these times. Figure 8 exemplifies this situation where a high response time seems to maintain or even improve sales for most applications and the loss in sales might be undetected by system administrators, and most importantly, by management on high level reports. Comparing the actual sales with predicted sales (based on mostly non-overloaded system state) will highlight the net loss of sales due to surges in response time.

6.1 Determining response time thresholds and its effect on user satisfaction

A drop in sales might not be evident due to low QoS, especially at peak times, as it might coincide with a high conversion rate (CR) moment or the product might have slow search pages, increasing averages to buying customers. To overcome this situation without modifying the OTAs infrastructure (Section 5), and intentionally adding delay to users like in Mayer (2009) and Mazzucco (2010), we have proposed the use of Machine Learning classifiers to predict expected sales for short time bins, in our case 10 min bins.

Figure 11 plots the relative percentage difference between actual sales and predicted sales as response time increases for each product of the OTA. It can be read as follows: the *Y-axis* is the percentage difference from 0 to −100 % representing the loss in sales; and the *X-axis* plots the response time from 0 to 30 s. Notice that some applications do not start with values at 0 s, as their response might not be that low for the averaged 10 min bin.

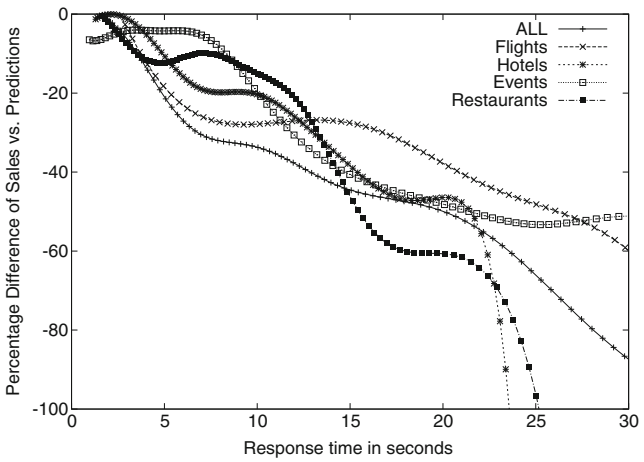


Fig. 11 Percentage difference of sales and predictions by response time

To produce the predicted data for this experiment, we have selected the *M5P* classifier, as it has lower computational requirements than *bagging* meta-classifiers and the output tree model for predictions is more complete than REPTree. The output model is used for validation and to understand what features in the training dataset are the most relevant for predicting sales (see Section 5.1.1). Recall that each application has differentiated response time averages and also different conversion rates for each time bin.

For *ALL* products, actual sales start to deviate from expected values between 3 to 7 s, and from 11 s have a huge drop compared to the expected sales. Next is the *flights* product, sales start to drop significantly between 4 and 8 s, and a huge drop after 14 s. For *hotels*, the first drop is between 3 and 7 s and the second drop after 10 s. The *events* product registers only one drop, after 7 s; recall that this product is more inflexible due to exclusivity and also has its peak CR during the morning contrary to the rest of the products. Restaurants on the other hand, has a very low first drop, between 2 and 4 s, and then also a low second drop after 7 s.

Table 4 summarizes inflection times for each application, where we have separated inflection points in two thresholds in accordance to the APDEX standard (Sevcik 2002): *tolerating* and *frustration*. At the *tolerating*, 1st threshold, some sales are lost, but most users remain on the site until the *frustration*, 2nd threshold, where a high percentage of sales is lost and more users abandon the site. Also, an average of sales loss is presented for each range. The rest of the users with response time lower than the thresholds are considered to be *satisfied*.

For the analyzed OTA, there is a *tolerating* response time threshold between 3 to 11 s, where some sales are lost. In average the *frustration* threshold for the analyzed OTA is at 11 s, where each increase in 1 s increases total sale loss by 3 %. Even within the same Web site, each application has differentiated thresholds, and as products exhibit different flexibility, they also show different *tolerating* and *frustration* times and each should be treated separately from one another.

6.2 Experiments validation

To validate the model and the methodology we have compared results for different days in the dataset where overload was not present as shown in Fig. 1; where periods of high response time presented greater sales loss difference than regular days. Machine learning

Table 4 Percentage of sale loss by increasing response time: two incremental response time thresholds, and corresponding drop of sales in percentage

Appl	1st thresh	1st drop	2nd thresh	2nd drop
ALL	3–7 s	5 %	11–22 s	55 %
Flights	4–8 s	28 %	14–30 s	55 %
Events	–	–	7–25 s	50 %
Hotels	3–7	20 %	10–18 s	45 %
Restaurants	2–4 s	10 %	7–18 s	60 %

predictions are needed as sales are prone to seasonality effects and volumes for different times of the day cannot be fully predicted by linear tendencies. Results indicate that the prediction technique is valid to countermeasure conversion rate effects on peak times as models capture conversion rate variability by hour of the day, day of the week, and season.

6.3 Response time effect on user clicks

User clicks and session length are also affected by response time. Figure 12 plots the average number of clicks per session as response time increases for each product. As it can be seen for most products, there is a huge drop in the average number of clicks per session after 2 s. For sites that do not sell products, the average session number of clicks or session length can be used to determine abandon times for users. However, as we have explained in Section 3.3, average session length is different by day of the week, so a learning and prediction approach would be recommended to capture these features more precisely, but it is out of the scope of this study. These results are consistent with previous works on user behavior (Galletta et al. 2004; Sevcik 2002).

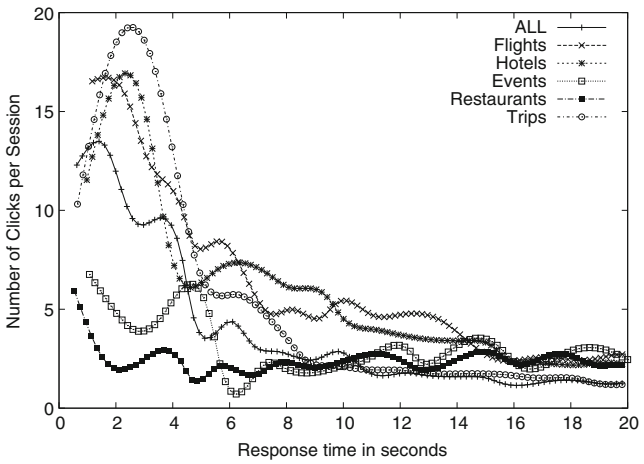


Fig. 12 Average number of clicks decrease as response time increases

7 Discussion of results

In this Section we introduce some discussion about the results presented in this paper. In Section 7.1, we compare our results with existing industry reports in terms of user patience levels and analyze the lowering conversion rate trend for the studied OTA scenario. Then, we discuss the applicability of the proposed methodology in production deployments in Section 7.2. Followed by the analysis of other possible performance metrics that can be used for determining system state described in Section 7.3. Finally, in Section 7.3 we discuss how to improve prediction results and how our methodology could be used to optimize revenue by leveraging dynamic server provisioning.

7.1 User patience levels to response time

Results from the previous section show that the user's tolerating *response time* thresholds are higher for most applications of the OTA from previous literature, especially industry reports. Where our response time numbers are in line with Miller's 1968 work on threshold levels of human interaction and attention (see Section 8). We believe that most of industry reports take static pages as a base measurement, which might not be representing the reality of many e-commerce sites. Transactional e-commerce pages such as a *flight availability search*, usually take a long time to be generated, and have a special waiting page from which users assume a higher complexity of the process, and thus are more patient for the results. The same can be observed for pages that involve external transactions, such as booking a restaurant, where many checks need to be performed i.e. credit card fraud, availability, re-check rates, web service calls, before the booking is made and the customer is already engaged in the reservation process.

Moreover, *tolerating* and *frustration* times are different for each application. For example the *events* application has exclusive content that cannot be purchased in online competitors, making it more inflexible than other applications such as *flights*, that has multiple competitors. Having different conversion rates and thresholds per application poses new challenges for differentiated and dynamic per-application QoS management during the day. Considering the current trend in Web ecosystems to observe lower conversion rates due to different factors i.e rising competition, affiliation, meta-crawling, and changes in user habits such as *multi-tabs* (Poggi et al. 2007a); online retailers will progressively support more traffic for less direct revenue by visit, increasing the importance of

optimizing the number of servers to reduce server costs without sacrificing sales.

7.2 Method applicability

The presented methodology does not rely on having any previous performance information to predict values for expected sales—normal system state—for a particular time *bin*. In our experiments the predictor model is trained using attributes from the 5 year sales dataset after the attribute selection described in Section 5.1.1), relying mainly on the exact time and date for the model to learn about conversion rates and sales patterns. This is an important feature of the method as performance surges might not be previously recorded or be infrequent, but still be able to detect system state changes when it is underperforming on the prediction class, *sales* in our case. Most of the requests in our dataset have a response time below two seconds as shown in Fig. 5, so even in peak times periods, the system is generally in a normal state and most users are within the *satisfied* threshold. By testing if the system is in a *normal state* which each prediction iteration, if sales are lower than expected and response time is above the thresholds, an action should be triggered to enable more servers in the case it should improve conversion rates and revenue.

Performance objectives to optimize revenue for the site can be built both offline and online following the presented methodology. In our experiments, as an example in compliance with the APDEX standard, we have set two thresholds of user satisfaction as response time increases: tolerating and frustrated. Table 4 shows how performance objectives and sale-loss can be built offline at scheduled intervals from all available data and performance values, that can be updated with each run, or have the performance objectives calculated in real-time for the last time *bin*, having not only two thresholds—tolerating and frustrated—, but progression of values as response time degrades to further optimize the model. In our experiments we have selected a low 10 min time bin, which we believe is applicable to dynamic server provisioning in cloud environments the main focus of our study. Moreover, obtained results can have other applications, as issuing a warning to system administrators when sales are low, similarly on how it is being used currently by the OTA, and e.g. having an alarm at three consecutive warnings—in this case half hour—of lower than expected sales. Furthermore, an implementation of the method can have more than one prediction per time bin: different prediction classes, e.g. sales and number of clicks; fine-tuned for each application; or even different length time bins.

7.3 Other performance metrics and prediction classes

We have tested several performance indicators to determine system state i.e. current database response time, external request times, CPU percentage for the request, and server *load average*; each has had good accuracy, but we have chosen response time as the performance indicator to determine sales inflection points as it is widely available and can be measured by routers, load balancers, web servers, the application and from the web client itself (e.g. in web analytics software). Response time is also key performance indicator in management reports in popular *web analytics* tools. As for the prediction *class*—in our case sales—, we have also tested with the average number of clicks for a web session as prediction *class*, presented in Section 6.3 and used for validation of the method, with similar results. We have chosen *sales* as it is in line with our previous work on Ecommerce website modeling, and no-or-low sales for the volume size of the OTA is a clear indication that there is a problem and the system is not in a *normal state* and has be contrasted with experts from the OTA.

7.4 Improving prediction results

In the presented experiments and prototype implementation, we have obtained less than 25 % of absolute error between predictions and real sales for the whole 6 month period. We believe that this accuracy is a good indicator for our proof of concept implementation, as we rely on simple and low-requirement numeric classifiers for predictions, which are apt to be used in a real-time environment without much fine tuning. From experience in our previous work (Poggi et al. 2007a), where we predicted user intentions in purchasing a product on every click of the session, predictions could be further improved by adding more attributes to the training datasets, implementing higher requirements classifiers such as *neural networks*, and converting some numeric attributes to *nominal* (See Section 5.1.1); with the expense of higher computational requirements. Furthermore, for this particular scenario, increasing the length for the time bin—we have used 10 min—and training and predicting for shorter periods—we have used 1 week predictions—also improves predictions. From results of this study, we have also demonstrated that each product has differentiated behavior and conversion rates, even for the same site i.e. some products such as flights rely on external requests for availability (Section 2.2), and they have differentiated CR during the day, week and season. Predictions could be further

improved by fine-tuning specifically for each product particularities.

Our model could have benefited from more overload periods in the dataset to improve precision. However, even with the low number of samples of high response time for the less popular products, the main inflection points and loss of sale tendencies can be obtained from it.

Results from Section 6 show that response time affects in roughly linear way the number of sales; that is a short but important take-home message for companies that performance is a key determining user satisfaction, and finally sales. In fact, the gap between predictions and real sales can itself be modeled and predicted. So one can calibrate the model to take into account response time as an attribute to help set performance objectives. The presented methodology enables online retailers to determine inflection points where sales start to be affected by the current application's *response time*. Where resulting values can be applied on autonomic resource managers to optimize the number of servers and reduce infrastructure costs in cloud computing environments. Most importantly, optimizations should not be made to accommodate all load, but to provide the best QoS when conversion rates are higher, generally at peak loads. Furthermore, optimizations should not be made to the lowest response time, but to provide a response time just before the *tolerating* threshold minimizing server costs. As an additional contribution, results from this study have led the presented OTA to implement important changes in their infrastructure to avoid high response times, especially at peak times, with positive results.

8 Related work

Response time effect on user behavior has been studied as long as 1968, where Miller (1968) describes the "Threshold Levels of Human Interaction and Attention with Computers". In 1989, Nielsen (1989) revalidated Miller's guidelines and stated that the thresholds are not likely to change with future technology. These thresholds being: 0.1–0.2 s, instantaneous; 1–5 s the user notices the delay but system is working; and 10 s as the threshold for user attention. Results from our experiments re-validate these results in the context of E-Commerce Web site for the average, ALL, application. However, threshold limits are different for different applications. Other authors (Galletta et al. 2004; Sevcik 2002) adhere to what they call the 8 s rule, where no page should take longer than 8 s to reply. The are several industry reports stating that users expect faster response times than previous years, specially the younger generation (Rheem 2010; Akamai 2009) that might reduce these numbers in the future.

To prevent loss in sales due to overloads several techniques have been presented such as *session-based admission* control systems (Cherkasova and Phaal 2002; Guitart et al. 2005) used to keep a high throughput in terms of properly finished sessions and QoS for limited number of sessions. However, by denying access to excess users, the Web site loses potential revenue from customers. Later works include service differentiation to prioritize classes of customers, in Garcia et al. (2009) and Ewing and Menascé (2009) authors propose the use of utility functions to optimize SLAs for gold, silver and bronze clients. In Poggi et al. (2009) authors propose the use of machine learning to identify most valuable customers and prioritize their sessions. Nowadays, Cloud computing enables sites to be configured at a minimum number of server resources and provision according to the incoming load, with a de facto unlimited number of servers, enabling autonomic resource managers to auto-configure server numbers (Al-Ghamdi et al. 2010; Mazzucco 2010). In Mazzucco (2010), Mazzucco proposes the use of utility functions to optimize auto-provisioning of web servers. None of these works however, explore the effects of higher conversion rates at peak loads, and by ignoring this fact, QoS of service is no optimized and potential sales are lost.

9 Conclusions

We have argued that the full effect of response time degradation can be hidden by the fact that peak load times can coincide with high conversion rates, i.e. when higher fraction of visitors have intention to purchase. To overcome this situation, we have introduced a novel methodology for studying what is the volume of sales lost in an online retailer due to performance degradation without modifying its application. We use machine learning techniques to predict the expected sales volume over time and look for deviations over the expected values during overload periods that may introduce performance degradation. Using such technique, we can quantify the impact of response time in the business activities of an online service without modifying production system.

We have tested the approach on logs from a top Online Travel Agency, using a 5 year long sales dataset, HTTP access log, and resource consumption datasets for six months of 2011. From the obtained results we are able to identify inflection points where sales start to drop for different applications when response time is

high. For the OTA, there is a *tolerating* response time threshold from 3 to 11 s, where some sales are lost, and a *frustration* threshold at 11 s, where each increase in 1 s interval increases total sale loss by 3 %.

We are currently generalizing the model to be integrated in an autonomic resource manager for cloud-like environments, which takes into account conversion rates, while observing performance and energy constraints. The goal for such resource manager is to feature dynamic server provisioning, optimizing the number of servers according to the incoming workload, without surpassing the thresholds of user satisfaction and maximizing revenue for each of the hosted applications of a site.

Acknowledgements We would like to thank Atrapalo.com, who provided the experiment datasets and domain knowledge for this study. This work is partially supported by the Ministry of Science and Technology of Spain under contracts TIN2007-60625, TIN2011-27479-C04-03, and by the Generalitat de Catalunya (2009-SGR-980, 2009-SGR-1428).

References

Akamai (2009). *E-Commerce web site performance today*. An updated look at consumer reaction to a poor online shopping experience.

Al-Ghamdi, M., Chester, A.P., Jarvis, S.A. (2010). Predictive and dynamic resource allocation for enterprise applications. In *International conference on computer and information technology* (pp. 2776–2783). doi:10.1109/CIT.2010.463.

David, A. (2009). *The secret weapons of the aol optimization team*.

Cherkasova, L., & Phaal, P. (2002). Session-based admission control: a mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers, 51*, 669–685. ISSN 0018-9340. doi:10.1109/TC.2002.1009151.

Ewing, J.M., & Menascé, D.A. (2009). Business-oriented autonomic load balancing for multitiered web sites. In *MASCOTS* (pp. 1–10).

Galletta, D.F., Henry, R., Mccoy, S., Polak, P. (2004). Web site delays: how tolerant are users. *Journal of the Association for Information Systems, 5*, 1–28.

Garcia, D.F., Garcia, J., Entrialgo, J., Garcia, M., Valledor, P., Garcia, R., Campos, A.M. (2009). A qos control mechanism to provide service differentiation and overload protection to internet scalable servers. *IEEE Transactions on Services Computing, 2*, 3–16. ISSN 1939-1374. doi:10.1109/TSC.2009.3.

Google (2010). Using site speed in web search ranking. Webpage: http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html. Accessed 9 May 2010

Guitart, J., Carrera, D., Beltran, V., Torres, J., Ayguad, E. (2005). Session-based adaptive overload control for secure dynamic web applications. In *International conference on parallel processing* (pp. 341–349). ISSN 0190-3918. doi:10.1109/ICPP.2005.72.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H. (2009). The weka data mining software: an update. *SIGKDD Explorations, 11*(1), 10–18.

Mayer, M. (2009). In search of a better, faster, stronger web.

Mazzucco, M. (2010). Towards autonomic service provisioning systems. In *IEEE International Symposium on Cluster Computing and the Grid* (pp. 273–282). doi:10.1109/CCGRID.2010.125.

Miller, R.B. (1968). Response time in man-computer conversational transactions. In *AFIPS '68 (Fall, part I): Proceedings of the 9–11 December 1968, fall joint computer conference, Part I* (pp. 267–277). New York: ACM. doi:10.1145/1476589.1476628.

Nielsen (2008). *Trends in online shopping, a Nielsen consumer report*. Technical report, Nielsen.

Nielsen, J. (1989). Usability engineering at a discount. In *Proceedings of the 3rd international conference on human-computer interaction on designing and using human-computer interfaces and knowledge based systems* (2nd ed., pp. 394–401). New York: Elsevier. ISBN 0-444-88078-X.

OECD (2009). Oecd report on broadband penetration. Available at: http://www.oecd.org/sti/ict/broadband. Accessed 4 Dec 2009.

Poggi, N., Carrera, D., Gavald, R., Torres, J., Ayguadé, E. (2010). Characterization of workload and resource consumption for an online travel and booking site. In *IISWC—2010 IEEE international symposium on workload characterization*.

Poggi, N., Moreno, T., Berral, J.L., Gavald, R., Torres, J. (2007a). Automatic detection and banning of content stealing bots for e-commerce. In *NIPS 2007 workshop on machine learning in adversarial environments for computer security*.

Poggi, N., Moreno, T., Berral, J.L., Gavald, R., Torres, J. (2007b). Web customer modeling for automated session prioritization on high traffic sites. In *Proceedings of the 11th international conference on user modeling* (pp. 450–454).

Poggi, N., Moreno, T., Berral, J.L., Gavald, R., Torres, J. (2009). Self-adaptive utility-based web session management. *Computer Networks Journal, 53*(10), 1712–1721. ISSN 1389-1286.

Power, S. (2010). Metrics 101: what to measure on your website.

Rheem, C. (2010). Consumer response to travel site performance.

Sevcik, P.J. (2002). Understanding how users view application performance. *Business Communications Review, 32*(7), 8–9.

Nicolas Poggi is a PhD candidate at the Computer Architecture Department (DAC) of the Technical University of Catalonia (UPC BarcelonaTech), where he has obtained a MS degree in Computer Science in 2007. He received his IT Engineering degree (best student award) with a minor in Business Administration at the American University (UA) in 2005. He is part of the High Performance Computing group at DAC and a collaborator at the Barcelona Supercomputing Center (BSC), also a Visiting Research Scholar at IBM Watson in 2012. Since 1999, Nicolas has been working as a consultant for ISPs and ECommerce projects in the areas Web Performance and Scalability, currently with focus on Cloud Computing, Machine Learning and Big Data.

Nicolas Poggi received his IT Engineering degree with a minor in Business Administration, at the American University (UA), 2005. He is now a PhD candidate at the Technical University of Catalonia in the Computer Architecture Department, DAC, where he has obtained the Diploma of Advanced Studies (DEA). He is part of the Barcelona eDragon Research Group, integrated with the High Performance Computing Group at DAC. Since 1999, he has been working as a consultant for ISPs and eCommerce projects, also, holds a technology transfer research scholarship.

David Carrera received the MS degree at the Technical University of Catalonia (UPC) in 2002 and his PhD from the same university in 2008. He is an associate professor at the Computer Architecture Department of the UPC. He is also an associate researcher in the Barcelona Supercomputing Center (BSC) within the "Autonomic Systems and eBusiness Platforms" research line. His research interests are focused on the performance management of data center workloads. He was a summer intern at IBM Watson (Hawthorne, NY) in 2006, and a Visiting Research Scholar at IBM Watson (Yorktown, NY) in 2012. He has been involved in several EU and industrial research projects. He received an IBM Faculty Award in 2010. He is an IEEE member.

Ricard Gavaldà received his Degree (1987) and Ph.D. (1992) in Computer Science from Universitat Politècnica de Catalunya, UPC. He has had a permanent position at the Departament de Llenguatges i Sistemes Informàtics of UPC since 1993, where he became full professor in 2008. His main research field was initially Computational Complexity Theory, and has gradually evolved towards Computational Learning Theory and algorithmic aspects of Machine Learning and Data Mining. He has recently started working on applications of Machine Learning to Autonomic Computing.

Eduard Ayguadé received the Engineering degree in Telecommunications in 1986 and the Ph.D. degree in Computer Science in 1989, both from the Universitat Politècnica de Catalunya (UPC), Spain. Since 1987 he has been lecturing on computer organization and parallel computing. Currently, and since 1997, he is full professor of the Computer Architecture Department at UPC. He is currently associate director for research on Computer Sciences at the Barcelona Supercomputing Center (BSC). His research interests cover the areas of multicore architectures, and programming models and compilers for high-performance architectures. He has published around 250 publications in these topics and participated in several research projects with other universities and industries, in framework of the European Union programmes or in direct collaboration with technology leading companies.

Jordi Torres has a Masters degree in Computer Science from the Technical University of Catalonia (UPC Barcelona Tech, 1988) and also holds a Ph. D. from the same institution (Best UPC Computer Science Thesis Award, 1993). Currently he is a full professor and has more than twenty years of experience in research and development of advanced distributed and parallel systems in the High Performance Computing Group at UPC. He has more than a hundred publications in these topics. He is currently a manager of the autonomic systems and ebusiness applications research line at the Barcelona Supercomputing Center (BSC) and participates in several research projects with other universities and industries, in framework of the EU programmes or in direct collaboration with technology leading companies on Cloud Computing, Green Computing, Big Data and Smart Computing.