Characterization of Workload and Resource Consumption for an Online Travel and Booking Site

Nicolas Poggi*‡, David Carrera*†, Ricard Gavaldà*, Jordi Torres*† and Eduard Ayguadé*†
*Technical University of Catalonia (UPC), Barcelona, Spain
†Barcelona Supercomputing Center (BSC), Barcelona, Spain
‡Contact author. E-mail: *npoggi@ac.upc.edu*

***Abstract*—Online travel and ticket booking is one of the top E-Commerce industries. As they present a mix of products: flights, hotels, tickets, restaurants, activities and vacational packages, they rely on a wide range of technologies to support them: Javascript, AJAX, XML, B2B Web services, Caching, Search Algorithms and Affiliation; resulting in a very rich and heterogeneous workload. Moreover, visits to travel sites present a great variability depending on time of the day, season, promotions, events, and linking; creating bursty traffic, making capacity planning a challenge. It is therefore of great importance to understand how users and crawlers interact on travel sites and their effect on server resources, for devising cost effective infrastructures and improving the Quality of Service for users.**

In this paper we present a detailed workload and resource consumption characterization of the web site of a top national Online Travel Agency. Characterization is performed on server logs, including both HTTP data and resource consumption of the requests, as well as the server load status during the execution. From the dataset we characterize user sessions, their patterns and how response time is affected as load on Web servers increases. We provide a fine grain analysis by performing experiments differentiating: types of request, time of the day, products, and resource requirements for each. Results show that the workload is bursty, as expected, that exhibit different properties between day and night traffic in terms of request type mix, that user session length cover a wide range of durations, which response time grows proportionally to server load, and that response time of external data providers also increase on peak hours, amongst other results. Such results can be useful for optimizing infrastructure costs, improving QoS for users, and development of realistic workload generators for similar applications.

I. Introduction

Online Travel and ticket booking have become one of the mayor E-Commerce industries. According to the 2008 Nielsen report on Global Online Shopping [1], Airline ticket reservation represented 24% of last 3 month online shopping purchases, Hotel reservation 16%, and Event tickets 15%; combined representing 55% percent of global online sales in number of sales. Popular travel sites such as Expedia, Orbitz and Travelocity offer a mix of products including: flights, hotels, cars, cruises and vacational packages; some sites even include restaurants, activities and event tickets. Travel sites generally work as intermediaries, e.g. for airline companies by connecting themselves to Global Distribution Services (GDS) providers such as Amadeus, Galileo or Sabre, via B2B XML Web services. Depending on the product, the Travel site might also act as the provider, managing the product inventory themselves, to be used for regular users, affiliated sites, and meta-crawlers.

To offer search results, e.g. flight availability, several providers are queried and results are offered according to different algorithms of product placement in a resource intensive operation. As some of this searches are costly —not only in terms of resources— but by contract of the GDS services, meta-crawling sites such as Kayak and Travel Fusion, *scrap* travel websites simulating real user navigation in order to compare results from different sites. This situation creates the necessity to automatically identify and sometimes ban such crawlers. One commonly used strategy across the industry is to heavily rely on caching, to prevent excessive searches from meta-crawlers and speed up results for users. Moreover, visit to traffic sites might not depend only on their popularity but to current year season, holydays, search engine ranking (SEO), linking and the proximity of an event, such as a concert. 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.

Evaluation of Web application resource consumption requires realistic workload simulations to obtain accurate results and conclusions. In this paper we take real production logs from a top Online Travel Agency (OTA) and make a complete characterization of both its client workload and the resource consumption, observed in the 35+ physical node cluster in which the application is deployed. The logs include several million requests over a high load week of 2010, to a 3-tier AJAX-enabled application implemented over popular LAMP[1] open-source technologies. Obtained server logs not only include HTTP data but also a detailed accounting for the resources consumed to process each request: CPU time in user mode, CPU time in system mode, number of requests and total access time to the database, time spent accessing external B2B request, as well as the current web server load status.

The characterization of the workload is approached from two different perspectives: firstly, the client workload pattern is studied, considering the request arrival rate, session arrival rate and workload pattern in a representative and generic 7 day access log. Secondly, the same 7 day log is studied from

[1]LAMP: Linux, Apache, MySQL, and PHP software

the point of view of resource consumption and the effect of server load on response time. The outcome of this study is the complete characterization of both user access pattern and non-simulated resource consumption of a Web application. Moreover, the studied dataset presents several features not present in most Web workload characterizations, such as the dependency of external providers, database access and mix of differentiated products (multi-application site). Resutls 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.

The rest of the paper is structured as follows: Secion II describes some of the main characterstics of the Web application (II-A) used by the OTA, their execution environment (II-B) and the datasets (II-C) made available to perform this study. Section III describes the characterization of the web workload, based in its transaction mix, intensity and burstiness. Section IV analyses the source of response time for the studied application. Section V studied how hardware resource are consumed by the application. Finally, section VI shows the related work and section VII the conclusions of our work.

II. Scenario

In this Section we provide details about the applications hosted by the OTA, its deployment scenario, and the information logged and collected for our workload characterization.

A. Applications characteristics

The application follows typical travel site structure such as the one described in Section I, offering the following products: flights, hotels, car, restaurants, activities, vacational packages, and event booking. Some of the products inventories are maintained internally e.g. *restaurant booking*, some are completely external e.g. *flights*, and some products like *hotels* are mix of internal and external providers. Access to the external providers is performed via B2B Web services. 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 advanced caching rules, to reduce request times and load generated by the external transactions. The company's main presence and clientele is in Europe, while a small percentage of the visits are from South America, and few 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 code base and differentiated resource requirements, while sharing a common programming framework.

B. Computing Infrastructure

The studied OTAs infrastructure is composed of about 35 physical servers running 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; 5 static content servers; 2 redundant file servers, 8 high end database servers including masters and replicas running MySQL; plus auxiliary servers for specific functions such as monitoring and administrative purposes. Web servers characterized in this study have double dual core *Intel Xeon* processors, 8G RAM and SATA hardisks. 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 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* and are not part of this study. In our previous work [20] we have identified that for every requested page 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 web template caching; distributed memory key-value storage, database query cache and scheduled HTML page generators. The log file use in this study is produced by the PHP dynamic application. At the end of each executing script, a log line is generated with execution information and resource usage, detailed in the next section.

C. Dataset Properties

The main dataset used in the next experiments, consists of transactions collected over a period of one week, from Monday 03/01/2010 2AM to Monday 03/08/2010 2AM, containing 19,221,382 total dynamic requests, representing 3,269,428 distinct sessions, and 15,488 different pages (non-ambiguous URLs). The dataset is generated by the PHP dynamic application, 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), replying server. Data from the application itself: total processing time, nom-ambiguous user session id, real page requested (some URLs might be ambiguous or perform different actions), accessed product, type of request (AJAX, Administrative, etc), CPU percentage, Memory percentage and total memory, CPU time both in system and user mode, total database time, total external request time. As well as current the current server load and number of Apache processes. We have also had access to the Apache logs and monitoring system for the same week and other random weeks, these auxiliary logs have been used to validate, explain obtained results and seasonality effect. Notice that the studied dataset does not include pages that were cached by the reverse proxy or by the users browser. There is a slight cut in the dataset, where the log generator was stopped during Friday's night, but it should not affect global results.

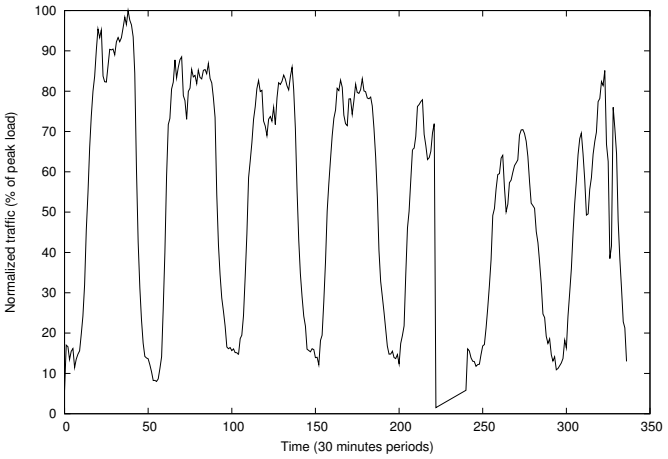


Fig. 1. Traffic volume intensity (relative to peak load). - 1 week

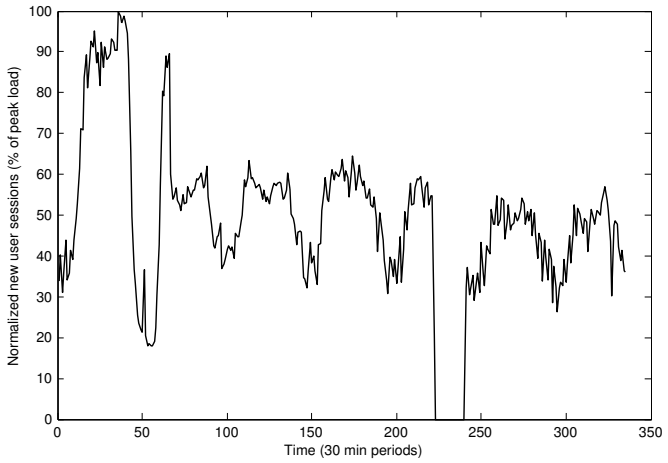


Fig. 2. New user sessions intensity (relative to peak load). - 1 week

III. Workload Characteristics and Descompositon

Figures 1 and 2 show the traffic pattern for the studied dataset, including number of hits (Figure 1) and number of user sessions started (Figure 2) over one week, grouped in 30-minute periods. Notice that data has been anonymized through being normalized to the peak load observed for each metric. As it can be observed that a problem with the logging infrastructure caused a short period of no information that can be clearly observed in Figure 2.

As it can be observed in Figure 1, the traffic decreases over the night, until it starts growing again soon after 7am in the morning. It keeps growing until noon, when it slightly decreases. Finally the workload intensity starts increasing again over the afternoon until it reaches its maximum around 9pm. Over the night, the traffic volume decreases until it finally reaches the beginning of the cycle again. Notice that client requests are conditioned by the response time delivered by the web application (next request in a user session is not issued until the response corresponding to the previous request is not received). For this reason, we made sure that the while logs were collected no severe overload conditions took place in the web infrastructure, but still capturing the a representative volume of traffic for a normal day in the online travel agency. We followed the same approach to characterize not client requests, but new web sessions in the system, that is, the number of new clients connecting to the system. The relevance of this measure, when taken in non-overloaded conditions, is that reveals the rate at which new customers enter the system. We also grouped data into 1 minute periods, that can be seen in Figure 2. As expected, per-session data follows the same trends observed for the per-request study, but with a smoother shape.

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			

TABLE I
Variables of the Normalized Request Rate Function

The mean page view for the whole week is 6.0 pages per session, with 6:48 minutes spent on the site, an average of 3.0s response time for dynamic page generation, and 8MB of RAM memory consumption. Recall that the highest traffic is on Mondays and decreases to the weekend. 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.

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

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

A. Workload Mix and Intensity

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 19.8%, dynamically generated Javascript 16.6%, HTTP redirect page 9.1%, Administrative 4.5%, internal scheduled batch 3.1%, API Web Service 0.04%, and Extranet 0.02%. It is an important feature of this dataset (and probably other real-life logs) that less than 50% of the total dynamic requests correspond to user clicks on their browsers.

Figure 3 shows the fraction of dynamic traffic volume that corresponds to different types of request categories, focusing on most relevant ones: Regular, AJAX, Redirects and JavaScript contents. As it can be observed, AJAX content

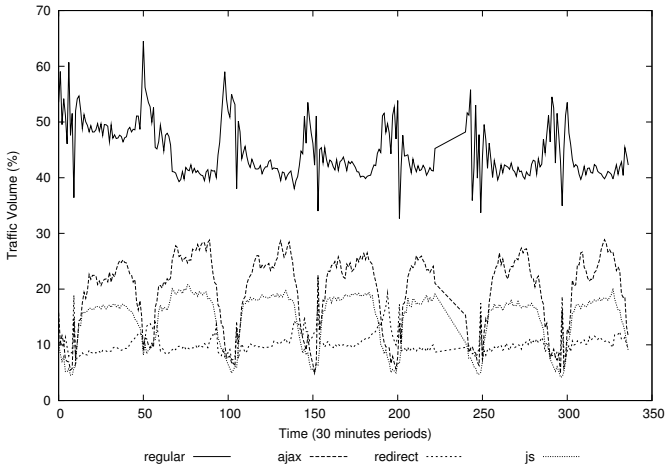


Fig. 3. Traffic mix over time (transaction type) - 1 week

Application	Percentage
App 1	27%
App 2	17%
App 3	15%
App 5	6%
App 6	5%
Other	23%

TABLE II
PERCENTAGE OF REQUESTS PER APPLICATION

fraction is mainly correlated to site's load, as such traffic is usually generated by human-generated actions (e.g. auto-completion of forms when indicating flight origin and destination airports during a search). During low traffic periods, basically overnight, most of the traffic is identified as regular pages. Night traffic also involved most of the internal batch and crawler activities.

A brief analysis on the number of crawler requests and meta-crawlers by analyzing the *agent* field (the reported browser type) identifying themselves as such; 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 3 years before [19], 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, that would show a growing trend in the proportion of automated bot traffic.

Table II shows traffic distribution across anonymized products (applications) offered by the OTA described in II-A. As it can be observed, almost 60% of the overall traffic come from only three applications, representing the most popular products of the company. Although each application is implemented independently, they share a common code base (e.g. user logging and shopping cart). Such *common* activity is not included in the specific per-application traffic volume, and is considered as a separate application by itself, corresponding to App 3, 15% of the total requests; this distribution is site specific.

Next step in the workload characterization is to study the per-session characteristics of the OTA visitors. Each session is started when a new visitor comes into the system, and is identified through a single identifier in the workload trace. We will look at four different session-specific characteristics: number of different products visited during the session, number of different pages visited per session, number of hits per session (notice that a hit can be initiated by a user click or by Javascript events such as auto-complete controls), and the session length. For each one of these characteristics, we construct a CDF chart as shown in Figure 4. Each CDF is built from the information collected during the lifetime of all the sessions started within a 30 minutes period. Recall that the completion time of a session can be much later than the end of the 30 minutes period. We have explored 4 different time ranges for each property, selecting time ranges corresponding to 4 points of time with different traffic characteristics, including night, morning, afternoon and evening traffic. The selected time ranges are 5:00am to 5:30am, 11am to 11:30am, 4:00pm to 4:30pm, and 10:00pm to 10:30pm. It can be seen from the Figures that all properties remain unchanged for all time ranges except for the night one. Session characteristics are approximately the same for morning, afternoon and evening time ranges, but a clear difference can be seen for the night (5am) traffic. Notice that the OTA is international, most of the traffic come from European countries located within the time zones with a maximum of 2h of difference. Obviously, the different characteristics of the nightly traffic come from the fact that the many sessions are initiated by non-human visitors (bots), including crawlers and meta-crawlers. This result supports the results presented before in Figure 3. Daytime (10pm) CDFs can be approximated using the probability distributions and parameters listed in Table III.

Our study concluded that 75.03% of the sessions only contained 1 hit, that is, the user only accessed 1 page, and then just quit the OTA site. This is mainly due to many visitors reaching the site through external banners that redirect them to espcial *landing pages*, and many of these users do not continue browsing the OTA site after this initial page. 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.

Figure 4(a) shows number of different pages visited per session (notice that a page is a unique URL here). Most users visit few pages during a session, and they may remain in one single page running searches or browsing the OTA catalog. Some users visit up to 14 pages in one single session, but that is the least of them. Figure 4(b) shows number of hits per session, with half of the visitors producing 10 or less requests to the OTA site. Notice that a request can be initiated by a user click, or by an AJAX action, such as field auto-completion in search forms. A significant percentage of visitors produce many more requests, reaching a few tenths in many cases. Figure 4(c) shows number of products visited per session. As the OTA site contains several different products, each one associated to a particular web application, we were interested

in studying how many different products were visited by each individual session. It can be seen that around 50% of the customers are interested in only two different products, but in some cases 8 or even more products may be visited in one single session. Finally, Figure 4(d) shows session length CDF, showing that while most visitors sessions last only a few minutes, some of them may be active for several hours. That may be explained by users coming back to the OTA site quite often over a long period of time, or by the presence of crawlers that periodically poll the OTA site contents.

Finally, we look at the burstiness properties of the workload, paying special attention to the session arrival rate and its changes over time. For such purpose, we have characterized the Index of Dispersion for Counts (IDC) of the entire workload as well as for a shorter time period which presents stationary properties. The IDC was used for arrival process characterization in [11], and has been leveraged to reproduce burstiness properties in workload generators in [5]. IDC was calculated by counting sessions started in 1 minute periods In a first step, we characterized the IDC for session arrival rate for the full dataset, covering one week period. The result for this step is shown in Figure 5(a). In a second step we picked the stationary period shown in Figure 5(c), corresponding to a 500 minutes high-load period, and characterized its burstiness through its IDC, as shown in Figure 5(b). Both figures indicate, given the high value of IDC observed, that the workload shows a high degree of burstiness as it is expected for any web workloads. And it remains true at both scales, including one week of processed data and a short and clearly stationary period of logged data.

IV. Response Time Analysis

In the following section we perform an analysis on response time: how it varies during the day, how it is affected by server load, how it affects the different applications, and finally it effects on *user behavior*. Total response time for a request is the time it takes Web servers to start sending the reply over the network to the user's browser. 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*, if supported by the user's browser. There is an additional time for the browser to render the final webpage, but it is not present in our dataset and is not part of this study as it deals with the actual HTML and media content.

A. Response Time Decomposition

From the available dataset, response time can be decomposed into: CPU time in system mode, CPU in user mode (including I/O times), database wait time, and external request wait time. Figure 6 presents the total response time for the complete dataset grouped by hour of the day. If we contrast it with Figure 1, by each daily period it can be seen clearly that response time increases with the total number of requests. Figure 6 also divides total time by the different resources, where the database response time also increases at peak hours. External request response time is not affected in the same proportion. CPU in system mode is not plotted on the graph as it was too low in comparison to the rest of the features; however it also presented noticeable higher response times at peak load. At peak time, from 18 to 22hrs, as Web server process more requests, they also present some resource contention due to high *load average* detailed in the next section.

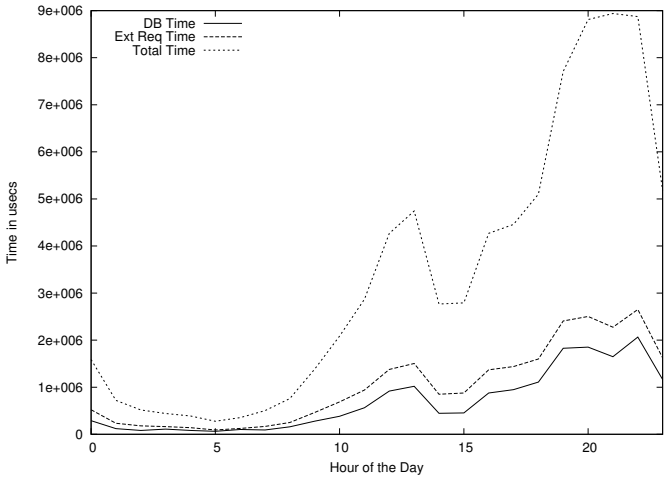


Fig. 6. Variation of response time during day from 0 to 24hrs

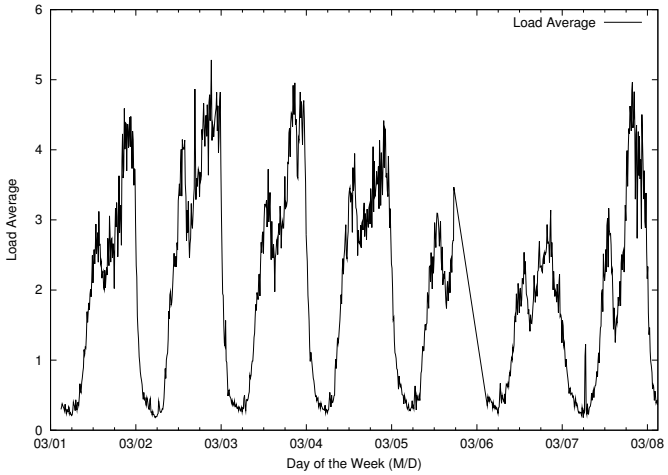
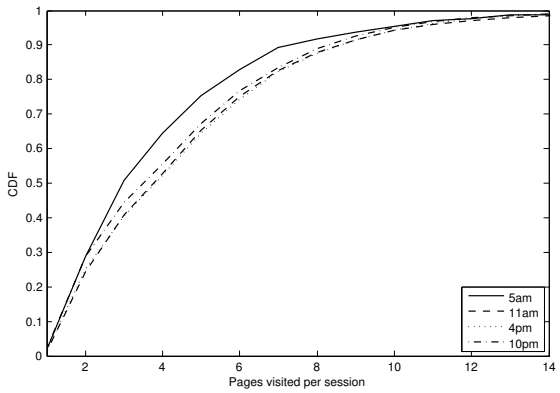


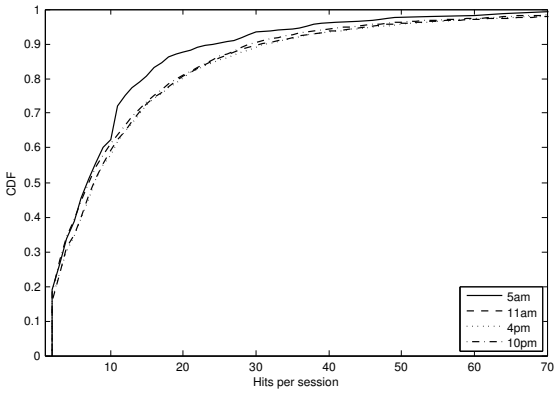
Fig. 7. Load Average values for the week

B. Response Time and Server Load

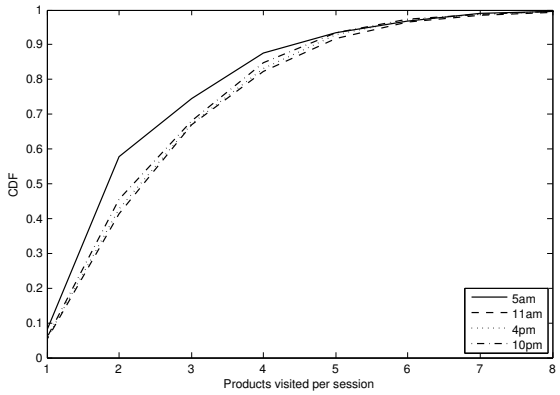
The next section analyzes how response time is affected as the *load* on the Web servers increases. To measure server load, we take the *load average* system value present in most UNIX systems [10], recall that the value of the *load average* is related to the number of scheduled jobs pending to be processed by the CPU. *Load average* is a very extended, simple, but accurate value for measuring current load on a server; in this study we use load averaged to 1 minute —opposed to 5 or 15 minutes— to have higher detail. To understand how loaded is a server by the load average, it is important to notice that each Web server has 2 CPUs with 2 cores each (described in II-B), giving a



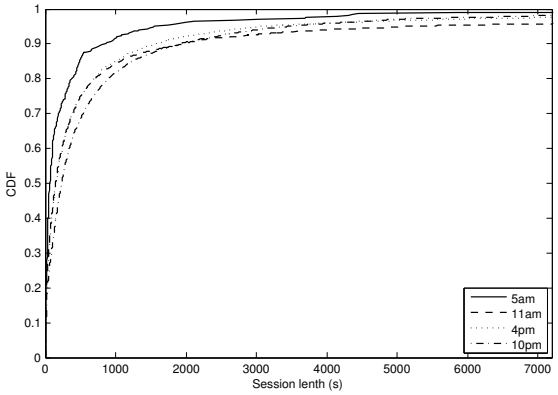
(a) Pages per session CDF



(b) Hits per session CDF



(c) Products per session CDF

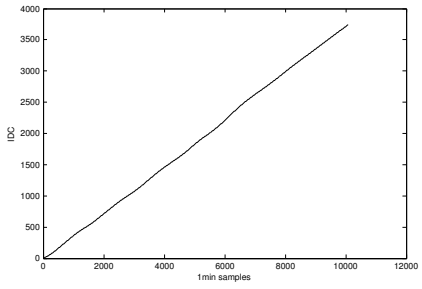


(d) Session length CDF

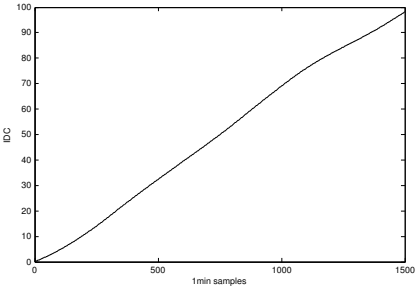
Fig. 4. Session properties - grouped according to session start time in 30-minute bins

Pages per Session	
Log-normal	$\mu = 1.37536; \sigma = 0.60544$
Hits per Session	
Log-normal	$\mu = 2.05272; \sigma = 1.00659$
Products per Session	
Log-normal	$\mu = 1.01541; \sigma = 0.457558$

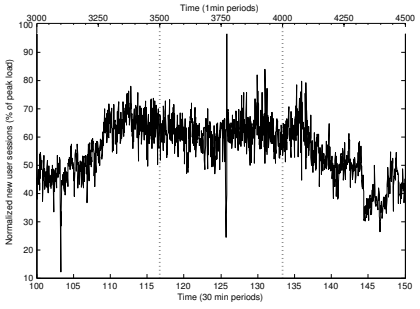
TABLE III
PER SESSION CDF FITS



(a) 1 week period - 1 min bins



(b) Stationary period - 1500 minutes - 1 min bins



(c) New user sessions intensity (relative to peak load). - 1 week

Fig. 5. Workload Burstiness: Index of Dispersion for Counts (IDC) of Initiated User Sessions

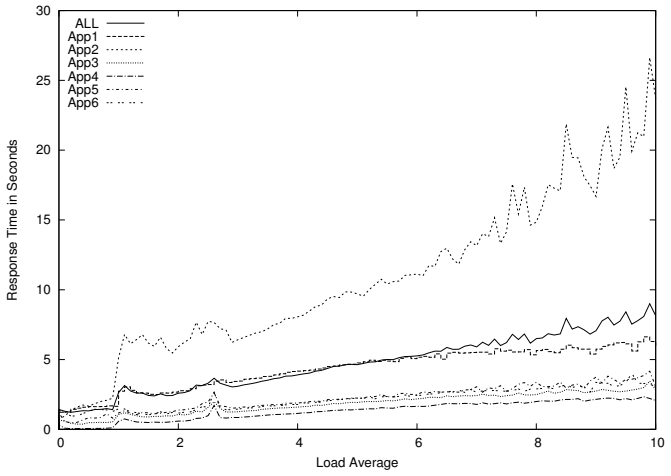


Fig. 8. Response Time by Load for different Apps

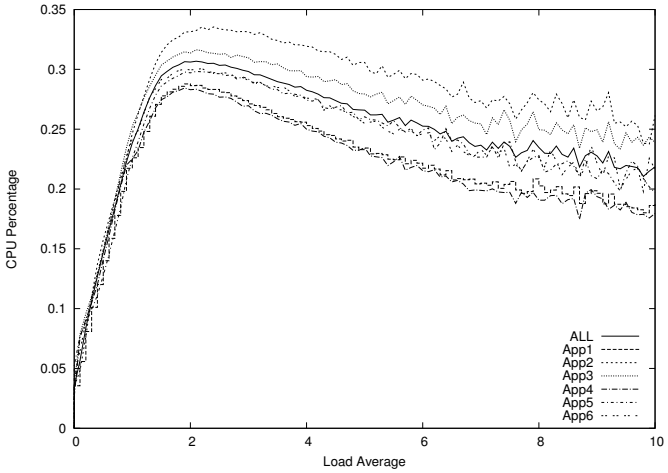


Fig. 9. Percentage of CPU Assignation by Load for Different Apps

total of 4 cores per server; as a rule of thumb, after 4 units of *load average* servers are considered overloaded (1 for each core).

Figure 7 presents the *load average* of the servers during the one week dataset. If we compare Figures 1 and 7 we can correlate how *load average* is affected directly by the number of concurrent requests on a given time, and that it follows the daily request pattern.

In Figure 8 we plot response time (Y axis) and *load average* (X axis) for the most popular applications, Apps 1 through 6 and the average *ALL* for all applications. Load average starts from 0, being the least busy value, to 10, the maximum value recorded on our dataset. From Figure 8 it can be appreciated that response time increases almost linearly as server load increases. From load 0 to 10, it increases almost to 10x in *ALL* for all applications, and up to 25x for App 2.

Response time increases with server load for three main reasons: server resource starvation (less dedicated system resources for a request), external B2B requests increased response time (low QoS), and contention between resources (jobs waiting for blocked resources).

For server resource starvation, Figure 9 shows how the percentage of CPU assigned by the OS to a specific Apache thread (request) reaches a maximum at load 2 (saturation point), and then starts decreasing leading to higher response time values. The same effect happens with the percentage of assigned memory, Figure 10, plots how memory percentage to Apache threads decreases very steeply from load 2.

As for external resource QoS, Figures 11 and 12 shows the response time for database queries and external B2B requests respectively. In Figure 11 we can see how the database response time also increases 3x in average for all applications, and up to 8x for App 2, which has a higher database usage. Figure 12 shows the average response time to external requests, we can see that App 2 is affected highly by the QoS of the external provider(s), while for App 1 it stays constant. The effect on App 2 is caused by the external providers getting overloaded at similar day times, than the

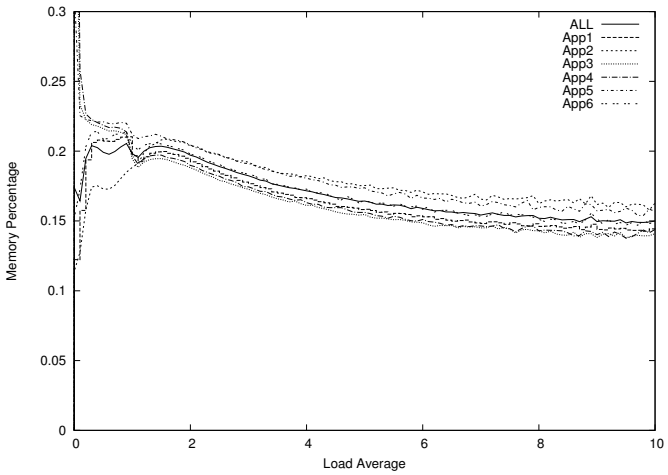


Fig. 10. Percentage of Memory Assignation by Load for Different Apps

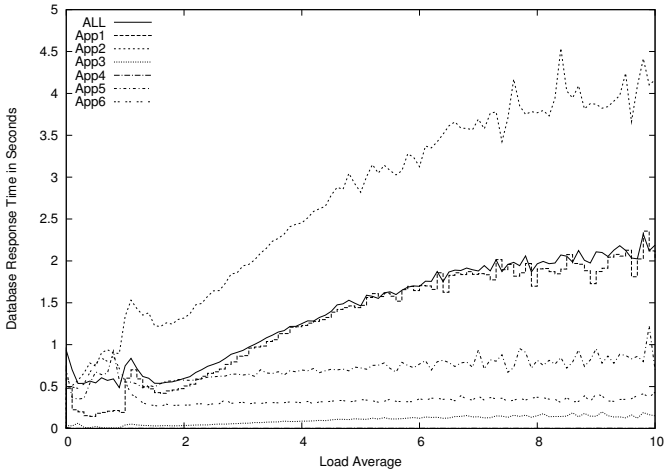


Fig. 11. Database Response Time by Load for Different Applications

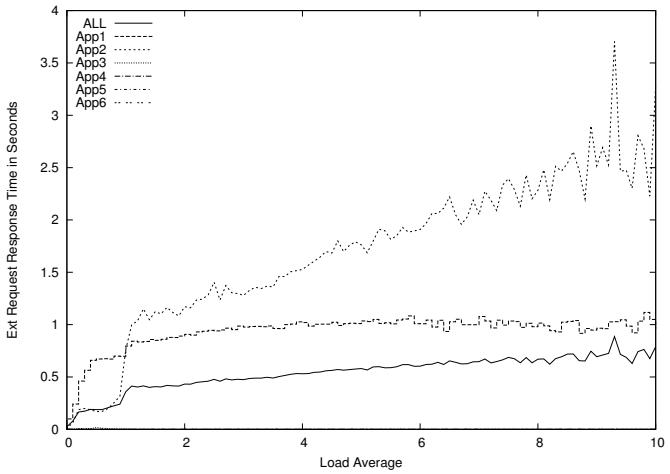


Fig. 12. External Request Time by Load for Different Applications

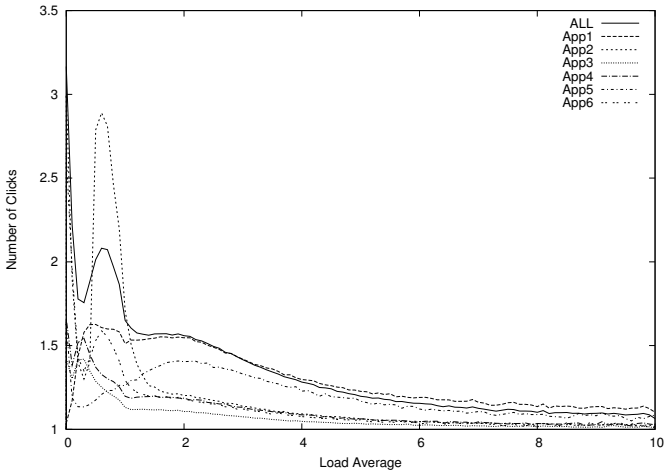


Fig. 13. Number of Clicks by Load

analyzed site; while the QoS for App 1 is stable at the sites peak hours.

It is important to notice that the less affected application by server load, is App 3, it is clear from Figures 11 and 12 that it does not rely on database or external requests, having the lowest response time increase, 2.5x. The other extreme, App 2, is heavily affected by both the database and external request times. An important feature from the last figures, if we zoom into the graph, is that the best response time is not at load 0, but is between load 0 and 1, as at load 0 (mainly at night time) *cache* hit rate is lower which leads to slightly higher times, although not comparable to high *load average*.

C. Sever load effect on Users

In the previous subsection, we have established how *response time* increases as load on servers increases following a linear trend. *Response time* has a direct impact on user behavior on the site, Figure 13 shows how the average number of clicks decreases as load and consequently, response time increases. This is consistent with previous works on user behavior [7], [21].

V. Resource Consumption

Figure 14 shows resource consumption distribution across anonymized applications.

When modeling sessions and requests they also have different characteristics and resource requirements. Figure 15 shows the different resource percentage used by each type of requests.

In Figure 16 we pick the most popular product of the OTA company 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 IV.

All the services related to this product require accessing at least once at the DB tier. Figures 16(a) and 16(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.

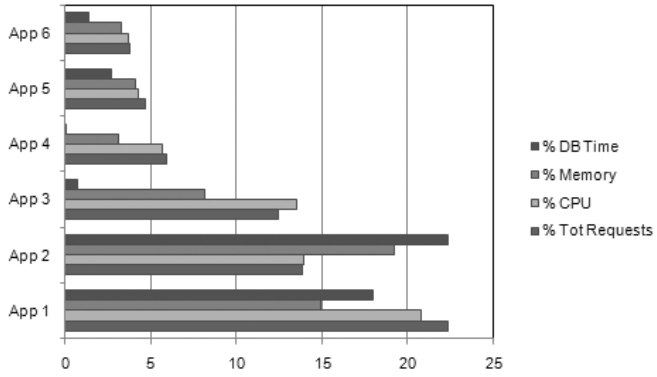


Fig. 14. Percentage of Resources by Application

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. But a significant fraction of the requests produce complex results and require a large number of DB queries to be completed, reaching more than one hundred DB requests in some cases. Looking at the time spent waiting for data from the DB tier,

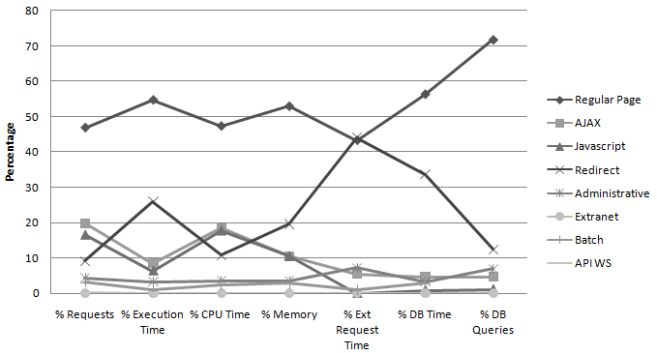
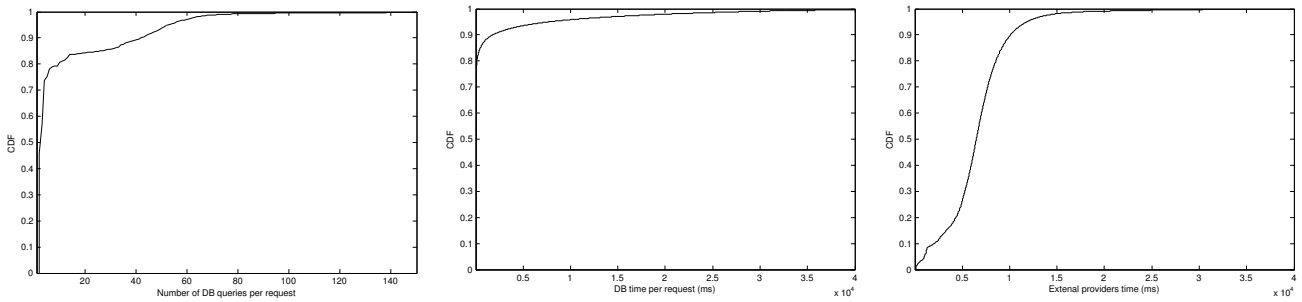


Fig. 15. Percentage of resource usage by request type



(a) DB queries (b) DB time (c) External Providers Time

Fig. 16. CDF of resource consumption for most popular product during an stationary, high load, 500 minutes period

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	$\mu = 6.17049e + 07, \sigma = -191940$

TABLE IV
DB AND EXTERNAL TIME CDF FITS FOR MOST POPULAR PRODUCT

most of the requests exhibit just a couple of seconds of DB query waiting time, but some cases can reach up to nearly 40s of DB time. Notice that some OTA operations may require complex optimization operations, as well as may provide a long list of results based on user search parameters.

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 App1. For those requests that did involve access to an external site, Figure 16(c) show 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 gathering information from external providers. For the cases in which accessing an external provider is required, the process is usually completed in less than 20s.

VI. RELATED WORK

In the context of Web workload analysis, there are few published studies based on real e-commerce data, mainly because companies consider HTTP logs as sensitive data. Moreover, most works are based on static content sites, where the studied factors were mainly: file size distributions, which tend to follow a Pareto distribution [2]; and file popularity following Zipfs Law [2], [12], [24]. Also, works such as [13] have studied logs from real and simulated auction sites and bookstores; there are no studies that we know about which are concerned with travel sites, like the one studied here, where most of the information comes from B2B providers and have a different behavior. Other works come to the same conclusions, but from the point of view of generating representative workload generators, such as [3] for static workloads and [6] and [16] for dynamic applications. None of these studies looked in detail at a complex multi-product Web 2.0 application in production of the scale of what is studied in this paper.

Similar work was conducted in [8] but following a black box approach for the enterprise applications, not exclusively web workloads, meaning that they shown no information about the nature and composition of the studied applications. Their work was data-center oriented, while our work is application-centric.

Recent studies have performed similar workload characterizations as the one presented here. In [4] Benevenuto et al. characterizes user behavior in *Online Social Networks* and Duarte et al. in [9] characterizes traffic in Web *blogs*. Previous work on the characterization of collaborative web applications was conducted in [23]. Although both the *blogosphere* and the OTA application used in our work are similar in the sense that they are user-oriented, user behavior is different in these scenarios. Moreover a more detailed analysis is presented in this paper, as the site is multi-application, and applications are further subdivided to perform a separate analysis by day, type of request, applications, as well as the resource consumption by each.

Few studies present both a characterization of workload and resource consumption. In [17] Patwardhan et al. perform a CPU Usage breakdown of popular Web benchmarks with emphasis on networking overhead, identifying that network overhead for dynamic applications is negligible, while not for static content. In [25] Ye and Cheng present a similar characterization of resource utilizations as the one presented here, but for *Online Multiplayer Games*. In this paper we also cover how response time affects user behavior in session length and number of clicks, validating results from previous studies [7], [21].

While [11], [5], [14] and [15] discuss about the need of stationarity of arrival processes to study and characterize workload burstiness, in [22] the authors work on non-stationary properties of the workloads to improve performance prediction. In our work, we have leveraged the techniques presented in some of these studies to characterize workload burstiness in stationary periods, but have not extended this work.

VII. Conclusions

In this paper we have presented a workload and resource consumption characterization for an Online Travel Agency. We were given access to a large dataset including millions of records for over one week of web activity, including workload information as well as delivered response time, and resource consumption levels observed on the underlying infrastructure. The online site is developed following modern technologies commonly used in the Web 2.0. In our work we decomposed the workload to create a clear picture of products (applications), unique pages, request types, system resources, interactions with databases, and external web services such as Global Distribution Services (GDS).

Results have been grouped into three categories: workload characterization, including transaction mix, intensity and burstiness; response time decomposition, showing sources of delay and effects of server load on response time and user clicks; and resource consumption, distinguishing between applications, putting emphasis to databases and external providers. Results show that the workload is bursty, as expected, that exhibit different properties between day and night traffic in terms of request type mix, that user session length cover a wide range of durations, that response time grows proportionally to server load, that response time of external data providers also increase on peak hours, and that automated crawler traffic is increasing and can represent more than 15% of total traffic, amongst other results.

As future work we plan to build a synthetic workload generator that mimics the studied application to model resource usage and bottlenecks. We also plan to analyze further, how response time affects user satisfaction and what would be the optimal server configuration to improve user QoS, while minimizing server costs by using techniques such as dynamic server provisioning [18].

Acknowledgements

This work is partially supported by the Ministry of Science and Technology of Spain and the European Union under contracts TIN2007-60625, TIN-2008-06582-C03-01, and by the Generalitat de Catalunya (2009-SGR-980).

References

[1] Trends in online shopping, a Nielsen Consumer report. Technical report, Nielsen, Feb. 2008.

[2] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing reference locality in the www. In *DIS '96: Proceedings of the fourth international conference on on Parallel and distributed information systems*, pages 92–107, Washington, DC, USA, 1996. IEEE Computer Society.

[3] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. *SIGMETRICS Perform. Eval. Rev.*, 26(1):151–160, 1998.

[4] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 49–62, New York, NY, USA, 2009. ACM.

[5] G. Casale, N. Mi, L. Cherkasova, and E. Smirni. How to parameterize models with bursty workloads. *SIGMETRICS Perform. Eval. Rev.*, 36(2):38–44, 2008.

[6] E. Cecchet, J. Marguerite, and W. Zwaenepoel. Performance and scalability of ejb applications. *SIGPLAN Not.*, 37(11):246–261, 2002.

[7] D. F. Galletta, R. Henry, S. Mccoy, and P. Polak. Web site delays: How tolerant are users. *Journal of the Association for Information Systems*, 5:1–28, 2004.

[8] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload analysis and demand prediction of enterprise data center applications. pages 171–180, Sept. 2007.

[9] M. A. Goncalves, J. M. Almeida, L. G. dos Santos, A. H. Laender, and V. Almeida. On popularity in the blogosphere. *IEEE Internet Computing*, 14:42–49, 2010.

[10] N. J. Gunther. Performance and scalability models for a hypergrowth e-commerce web site. In *Performance Engineering, State of the Art and Current Trends*, pages 267–282, London, UK, 2001. Springer-Verlag.

[11] R. Gusella. Characterizing the variability of arrival processes with indexes of dispersion. *Selected Areas in Communications, IEEE Journal on*, 9(2):203 –211, feb 1991.

[12] M. Levene, J. Borges, and G. Loizou. Zipf's law for web surfers. *Knowl. Inf. Syst.*, 3(1):120–129, 2001.

[13] D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira, Jr. In search of invariants for e-business workloads. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 56–65, New York, NY, USA, 2000. ACM.

[14] N. Mi, G. Casale, L. Cherkasova, and E. Smirni. Burstiness in multi-tier applications: symptoms, causes, and new models. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 265–286, New York, NY, USA, 2008. Springer-Verlag New York, Inc.

[15] N. Mi, G. Casale, L. Cherkasova, and E. Smirni. Injecting realistic burstiness to a traditional client-server benchmark. In *ICAC '09: Proceedings of the 6th international conference on Autonomic computing*, pages 149–158, New York, NY, USA, 2009. ACM.

[16] P. Nagpurkar, W. Horn, U. Gopalakrishnan, N. Dubey, J. Jann, and P. Pattnaik. Workload characterization of selected jee-based web 2.0 applications. pages 109–118, Sept. 2008.

[17] J. P. Patwardhan, A. R. Lebeck, and D. J. Sorin. Communication breakdown: analyzing cpu usage in commercial web workloads. In *ISPASS '04: Proceedings of the 2004 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 12–19, Washington, DC, USA, 2004. IEEE Computer Society.

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

[19] N. Poggi, T. Moreno, J. L. Berral, R. Gavald, and J. Torres. Automatic detection and banning of content stealing bots for e-commerce. *NIPS 2007 Workshop on Machine Learning in Adversarial Environments for Computer Security*, December 8, 2007.

[20] N. Poggi, T. Moreno, J. L. Berral, R. Gavald, and J. Torres. Web customer modeling for automated session prioritization on high traffic sites. *Proceedings of the 11th International Conference on User Modeling*, pages 450–454, June 25-29, 2007.

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

[22] C. Stewart, T. Kelly, and A. Zhang. Exploiting nonstationarity for performance prediction. *SIGOPS Oper. Syst. Rev.*, 41(3):31–44, 2007.

[23] C. Stewart, M. Leventi, and K. Shen. Empirical examination of a collaborative web application. pages 90–96, Sept. 2008.

[24] A. Williams, M. Arlitt, C. Williamson, and K. Barker. Web workload characterization: Ten years later. *Springer*, 2005.

[25] M. Ye and L. Cheng. System-performance modeling for massively multiplayer online role-playing games. *IBM Syst. J.*, 45(1):45–58, 2006.