Journal of Web Engineering, Vol. 14, No. 1&2 (2015) 003–021
© Rinton Press

SURFING THE WEB USING BROWSER INTERFACE FALILITIES: A PERFORMANCE EVALUATION APPROACH

RAÚL PEÑA-ORTIZ[1,2], JOSÉ A. GIL[1], JULIO SAHUQUILLO[1], ANA PONT[1]

e-mail: raul.research@icloud.com, {jagil,jsahuqui,apont}@disca.upv.es

(1) Departament d'Informàtica de Sistemes i Computadors
Universitat Politècnica de València
Camí de Vera s/n, 46022, Valencia, Spain

(2) Facultad de Ingeniería en Sistemas, Electrónica e Industrial
Universidad Técnica de Ambato, Campus Huachi, Edificio Zeta
Ambato, Tungurahua, Ecuador

The user interaction with the current web contents is a major concern when defining web workloads in order to precisely estimate system performance. However, the intrinsic difficulty to represent this dynamic behavior with a workload model has caused that many research studies are still using non representative workloads of the current web navigations. In contrast, in previous works we demonstrated that the use of an accurate workload model which considers user's dynamism when navigating the web clearly affects system performance metrics.

In this paper we analyze, for the first time, the effect of considering the User-Browser Interaction as a part of user's dynamic behavior on web workload characterization in performance studies. To this end, we evaluate a typical e-commerce scenario and compare the obtained results for different behaviors that take the user interaction into account, such as the use of the back button and parallel browsing originated by using browser tabs or opening new windows when surfing a website.

Experimental results show that these interaction patterns allow users to achieve their navigation objectives sooner, so increasing their productivity up to 200% when surfing the Web. In addition, results prove that when this type of navigations is taken into account, performance indexes can widely differ and relax the stress borderline of the server. For instance, the server utilization drops as much as 45% due to parallel browsing behavior.

Keywords: Web performance evaluation, Web dynamism, User's dynamic behavior, Dynamic web workload, Human-Computer Interaction

1 Introduction

Web applications and services have been introduced as a part of our daily life, evolving users from passive consumers to active contributors to the available dynamic content [1]. Furthermore, this trend is rising in the current Web where desktop web interfaces are making way to mobile devices like smart phones or tablets, which increase users' contributions [2].

New interaction methods have influenced user's habits, such as tabbed browsing, mouse gestures or screen interaction. Consequently, web user's navigations are different than they were a few years ago [3]. Moreover, the most important behavioral changes concerning user's dynamism are still expected to come in the incoming Web.

Nevertheless, this kind of user's behavior is not properly modeled in performance evaluation studies, because typical workload models do not consider the implicit dynamism of users when characterizing their navigations. However, these studies are necessary to provide sound proposals when designing new web-related systems [4], such as web services, web servers, proxies or content distribution policies. Consequently, the lack of accurate and representative workload models can negatively affect the validity of the obtained results.

To deal with this shortcoming, we focused on modeling user's dynamism in a previous work [5]. The resulting model and web workload generator are able to consider user's dynamic behavior when characterizing and reproducing dynamic web workloads. By using them, we proved [6] that the dynamic behavior of web users is a crucial point that must be addressed in web performance studies in order to accurately estimate system performance. Otherwise, performance studies can lead to results quite distant from the obtained with real systems.

This paper analyzes and measures the effect of considering the User-Browser Interaction (UBI) when modeling user's behavior on dynamic web workload characterization. To the best of our knowledge, this behavior has not been taken into account to generate workloads in existing web performance studies yet.

Results show that parallel browsing behavior, which is originated by using browser tabs or opening new windows when surfing the Web, permits users to increase the productivity of their navigations. That is, they achieve their objectives sooner than when browsing in a sequential way. This behavior also affects the usage and throughput of the main resources of the system. Moreover, the stress borderline of the server is relaxed permitting the system either to devote more resources to other applications or to serve more users.

The remainder of this paper is organized as follows. Section 2 discusses relevant state-of-the-art work. Section 3 briefly describes the experimental testbed devised to carry out fair comparison studies. In Section 4 we present workloads considering UBI when modeling user's behavior. Section 5 shows the effect of the proposed workloads on the web system performance. Finally, we draw some concluding remarks and future work in Section 6.

2 Motivation and Related work

We are not aware of any previous work dealing with modeling UBI on dynamic web workload characterization. The closest pieces of related work are some previous attempts at modeling user's behavior on workload characterization.

There are three main points that must be addressed when modeling the user's behavior on realistic dynamic workloads. First, the user's dynamic behavior must be modeled [4]. Then, the different user's roles in the Web must also be characterized [3]. Finally, continuous changes in these roles must be represented and considered [7]. Nevertheless, the real challenge is to consider them all together in a general web scenario.

In contrast, the most relevant works in the open literature that model user's behavior to obtain more representative dynamic workloads have been proposed for specific web applications. For instance, the Customer Behavior Model Graph [8] describes patterns of user's behavior in the workloads of e-commerce sites. This model was applied for workload definition of blogspace [9], and extended to capture an application inter-request and data dependencies [10]. The Clickstream Model was introduced by [11] to characterize user's behavior in online social networks. However, as mentioned above, the main drawback of these approaches is not only that they focus on specific paradigms and applications, but also they do not propose a user's model for a general context that can represent the dynamic behavior in an appropriate and accurate way (first challenge), and consider the different user's dynamic roles (second and third challenges).

These shortcomings led us to propose the Dynamic WEB workload model (DWEB) and the Universal Generator of Dynamic Workload under WWW Platforms (GUERNICA) [5]. DWEB permits us to model dynamic web workload for general contexts, taking into account the mentioned challenges by introducing user's dynamic behavior in the workload characterization. GUERNICA was also developed in order to exploit DWEB on generating dynamic workload by mimicking the behavior of the real web users community.

On the other hand, there is an evidence for an important change of user interaction with the Web. For instance, a recent study showed that 57.4% of web sessions involve parallel browsing behavior [12]. This behavior was originally found in the experienced web users, who surf the Web by using multiple browsers tabs or windows to support backtracking or multitasking with the aim of enhancing their navigation productivity [13, 14]. Moreover, the history-back button, included in any current web browser, is still one of the most heavily used user interface components in the web context, and accounts for up to 31% of all revisits [15].

This important change has been considered in several studies and tools to improve the website usability [16], to test web applications [17] or learning user preferences [18]. However, to the best of our knowledge, UBI has not been taken into account when modeling user's dynamism on workload characterization in web performance studies yet.

3 Performance evaluation

The testbed used in this work [19, 20] results from the integration between GUERNICA and a commonly used benchmark for e-commerce. It was developed with the aim of providing support to evaluate the system performance considering dynamic workloads generated with the DWEB model.

This testbed models an on-line bookstore environment, which is a representative e-commerce system. Figure 1 depicts a simplified website map for the on-line bookstore, where pages with similar functionality belong to the same group: *ordering, shopping, browsing, admin* and *search*. Navigation hyperlinks among them are also indicated.

A book searcher is provided by the *search* group, which embraces the Search and the Search Results pages to request the query and to show the list of results, respectively. The bookstore catalog is arranged according to the sales and the publication date by the pages of the *browsing* group (Best-sellers page and New Products page). The *shopping* group provides a sale functionality by managing the shopping cart (ShoppingCart page) and a buying process (Buy Request page and Buy Confirm page) that finishes with a payment through a secure navigation (Customer Register page). The *ordering* group (Order Inquiry page and Order Display page) offers the required functionality for checking the order status. The *admin* group manages the catalog of books. Finally, Home and Product Detail pages are also included since they are the most referenced ones.

Next, the main features of both the experimental setup and the performance metrics are summarized.

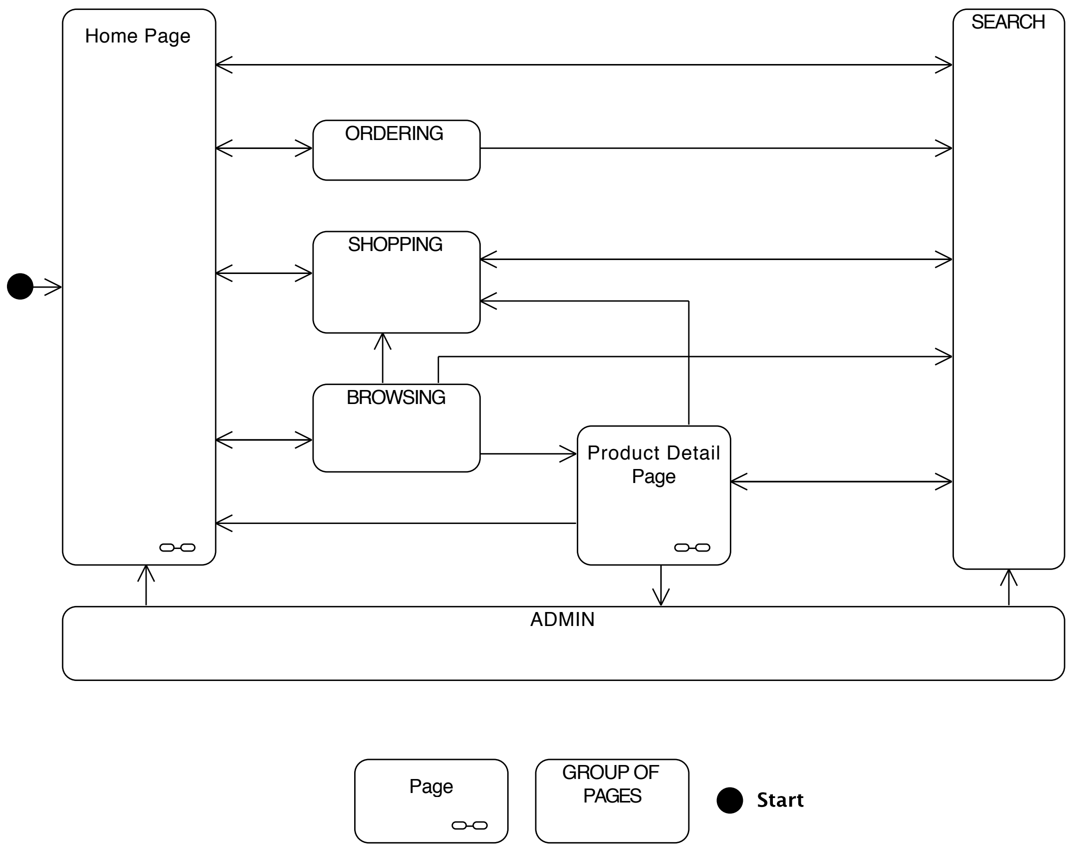


Fig. 1. Testbed website map

3.1 Experimental setup

The experimental setup used in this study is a typical two-tier configuration consisting of an Ubuntu Linux Server back-end tier and an Ubuntu Linux client front-end tier. The back-end runs the on-line bookstore, whose core is a Java web application (`web app`) deployed on the Tomcat web application server. Requests to static content of this web application, such as images, are served by the Apache web server, which redirects requests for dynamic content to Tomcat. The web application generates the dynamic content by using synchronous calls which fetch data from the MySQL database. On the other hand, the front-end tier is able to generate the workload using the DWEB model. Both web applications and workload generators are run on the SUN Java Runtime Environment 5.0 (`JRE 5.0`). Figure 2 illustrates the hardware/software platform of the experimental setup used in this work.

Given the multi-tier configuration of this environment, system parameters (both in the server and in the workload generators) have been properly tuned to avoid that middleware and infrastructure bottlenecks interfere the results. The on-line bookstore has been configured with 300 users and a large number of items (100,000 books) that forced us to balance accesses in the database (e.g. the pool connection size), static content service by Apache (e.g. the number of processes to attend HTTP requests), or dynamic content service by Tomcat (e.g.

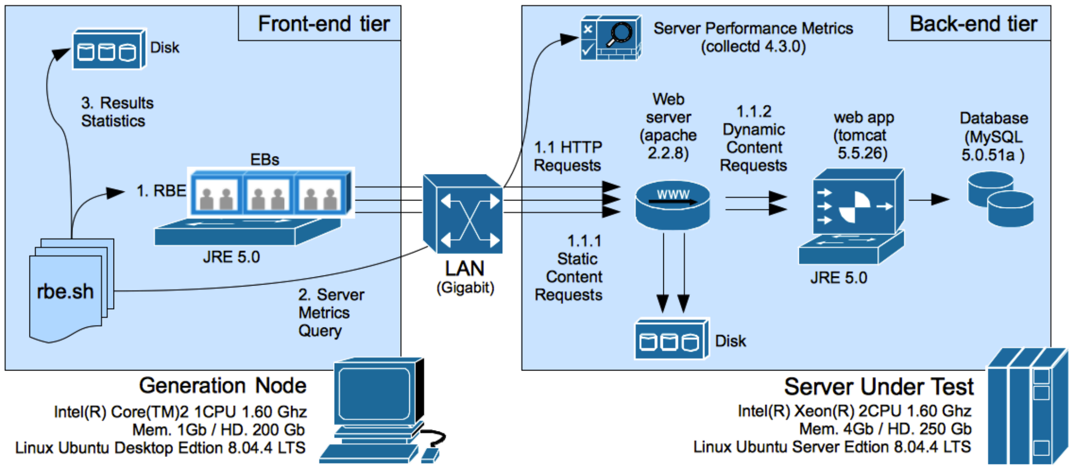

Front-end tier
Disk
3. Results Statistics
EBs
1. RBE
JRE 5.0
rbe.sh
2. Server Metrics Query
LAN (Gigabit)
Generation Node
Intel(R) Core(TM)2 1CPU 1.60 Ghz
Mem. 1Gb / HD. 200 Gb
Linux Ubuntu Desktop Edtion 8.04.4 LTS
Back-end tier
Server Performance Metrics (collectd 4.3.0)
1.1 HTTP Requests
Web server (apache 2.2.8)
www
1.1.2 Dynamic Content Requests
web app (tomcat 5.5.26)
Database (MySQL 5.0.51a)
1.1.1 Static Content Requests
JRE 5.0
Disk
Server Under Test
Intel(R) Xeon(R) 2CPU 1.60 Ghz
Mem. 4Gb / HD. 250 Gb
Linux Ubuntu Server Edtion 8.04.4 LTS

Fig. 2. Experimental setup

the number of threads providing dynamic contents). Each experiment was 35-minute length, consisting of a 15-minute warm-up period followed by a 20-minute collecting-data phase where measurements were performed. During an experiment we simulated a group of users surfing the bookstore according to different behaviors modeled with DWEB. All the experiments were done repeating 50 runs to obtain a margin of error with 99% confidence level.

3.2 Performance metrics

The experimental setup provides more than one hundred metrics, which can be classified in two main groups according to the side they are collected: metrics from the client side and metrics from the server side.

The *response time* and the *total number of requests per page* are the main metrics used in performance studies on the client side, but we also considered other metrics related to the user's navigation sessions, such as the *number of finished sessions*, the *session length* or the *number of visited pages during a session*.

On the server side, we used a middleware named `collectd`[21] that gathers system performance statistics periodically. This middleware collects the required server performance statistics that can be classified into two main groups according to the kind of resource they evaluate: metrics related to the usage of main hardware resources (*CPU and memory utilization*, *system I/O activity* and *network statistics*), and performance statistics for the software components of the back-end (*web server statistics*, *web application server activity* and *database I/O metrics*).

4 Workload design

This work uses DWEB to introduce different levels of user's dynamism on workload characterization, specifically dynamism related to the interaction with the web browser. Section 4.1 focuses on user's goals when surfing a website and defines user's navigations according to this end. After that, a more realistic dynamic workload is defined considering rapid return to recently visited pages by using the history-back button on web browsers. Section 4.2 introduces parallel tab browsing behavior on workload characterization.

4.1 The back button: rapid return to recently visited pages

An important factor of the back button success is the chance of rapid return to recently visited pages [22], which can avoid new HTTP requests and consequently changes the causes of the stressing conditions of a given website. However, only certain visited pages can be cached by a web browser with the aim of going back using the back button. That is, some web contents such as audio or video streaming, dynamic contents or web forms are not cached according to their HTTP headers. For these types of contents the back button does not have effect because pages are completely reloaded.

In a first step, a dynamic workload (LOY) conducted by user's goals is defined. For this purpose, we assume a common scenario of an e-commerce website that pursues to avoid the defection of customers as objective. A large percentage (more than 60% in some sectors) of new customers defect before their third anniversary with an e-commerce website [23]. Consequently, these websites care deeply about customer retention and consider loyalty of huge importance to the success of their on-line operations.

The scenario consists of an on-line bookstore that defines a loyalty promotion considering a general discount only for those customers who buy at least once a month. The promotion introduces a new user behavior with five cases of dynamism as summarized in Table 1.

Table 1. Cases of dynamism in the loyalty promotion behaviors

Case	**Description**
1	If customers do not remember their last order status, they will check them by navigating into the *ordering* group of pages.
2	Because the customer has an obligation of buying at least once a month to keep the discount, a buying session must finish with a payment when he has not bought anything during that month.
3	An experienced customer only buys a book when its cost is 25% cheaper than in other markets.
4	The higher the number of provided search results, the longer the time that a user takes to read and think about them.
5	A customer leaves the website when the buying session finishes because his/her goals have been satisfied.
	Extra case in the extended behavior
6	A customer can return to recently visited listings of books (browsing listings or search results) without repeating a request to the web server by using the back button.

Secondly, a new DWEB workload (LOYB) is defined by extending the LOY workload with an extra case of dynamism that characterizes the use of the back button on web browser (see

Table 1).

Figure 3 and Figure 4 show the workloads generated using DWEB for the loyalty promotion behavior conducted by goals (LOY) and its extended version (LOYB) considering the back button, respectively.

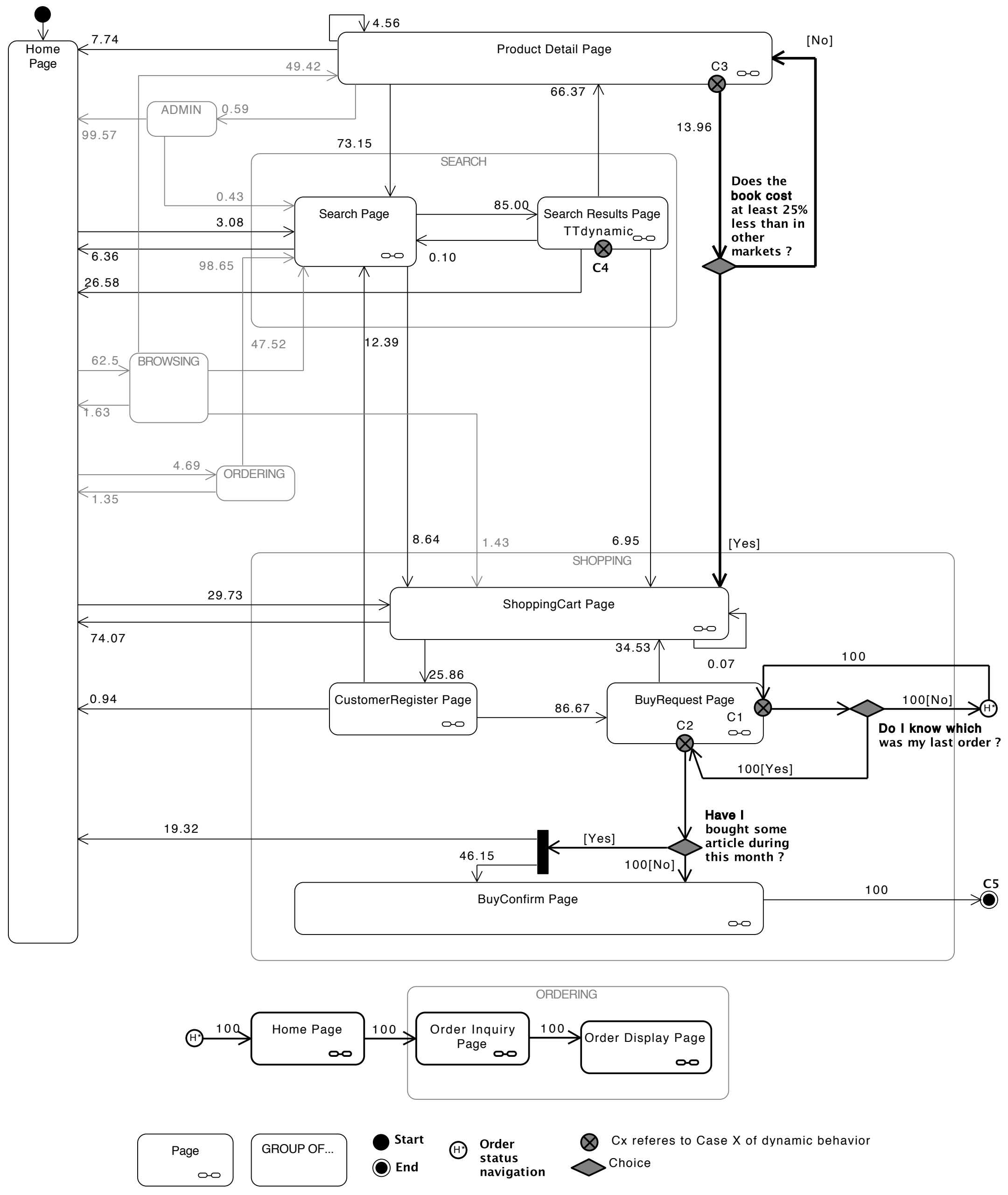


Fig. 3. LOY workload: loyalty promotion behaviors conducted by goals

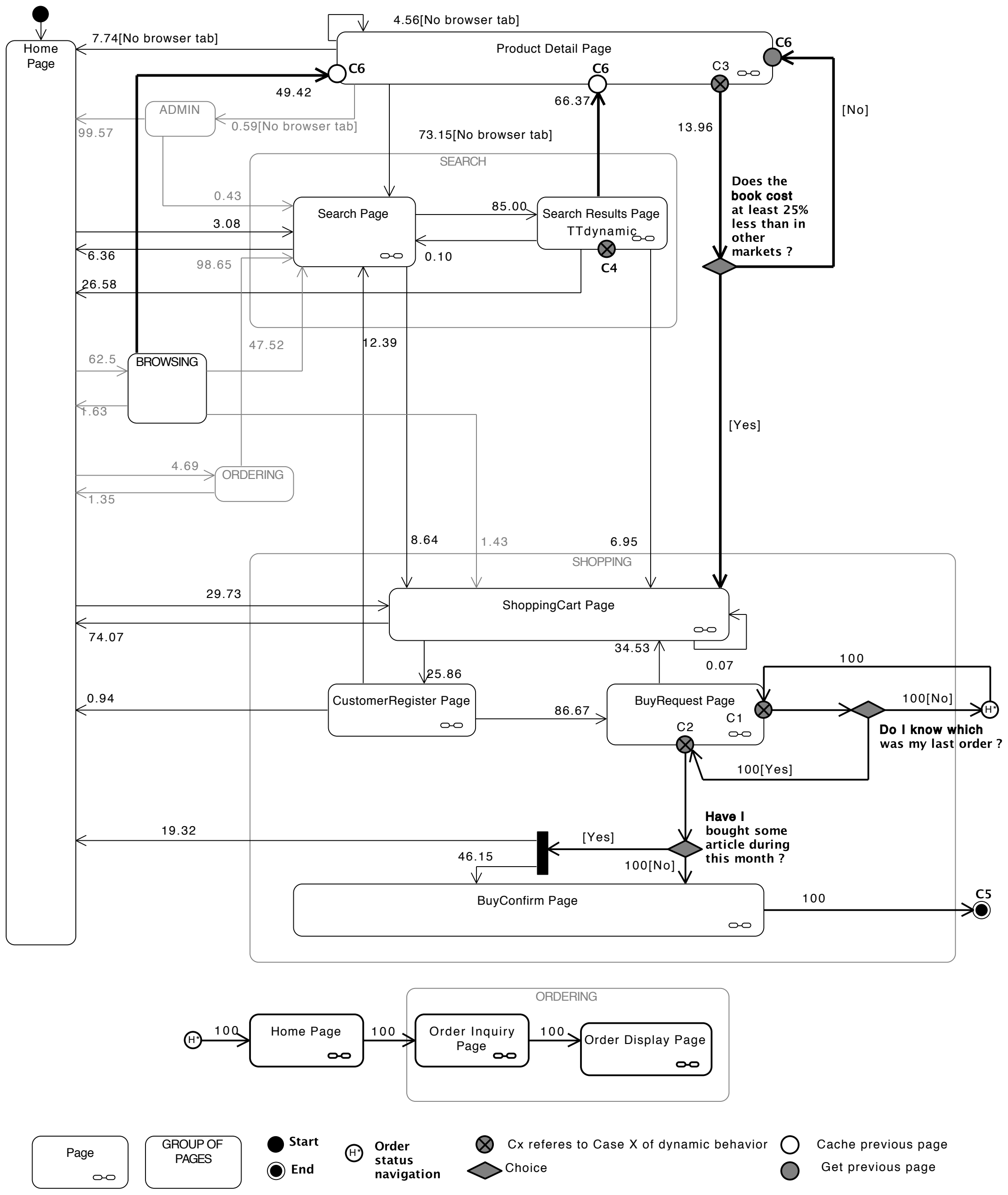


Fig. 4. LOYB workload: LOY workload considering the back button

For the less dynamic pages in the website (e.g. the Home page or the Product Detail page), both workloads assume a *think time (TT)* as defined in [24], that is, $TT = T2 - T1$, where $T1$ is the time measured at the browser when the last byte of the last web interaction is received from the server, and $T2$ is the time measured when the first byte of the first HTTP request of the next web interaction is sent from the browser to the server. This definition considers that each think time must be taken independently from a negative exponential distribution, with the restriction that the average value must be greater than 7 seconds and lower than 8 seconds as shown in equation (1). An important drawback of this approach is that it does not consider the results of the current search (web contents) on the actual user think time as occurs in real navigations.

$$TT = -ln(r) * p \ , \ where \ (0 < r < 1) \ \wedge \ (7 \leq p \leq 8) \tag{1}$$

In contrast, DWEB allows us to define a *dynamic think time* ($TT_{dynamic}$) according to the number of items returned by the search as shown in equation 2, which is closer to real web activities like the one defined in the *case 4* listed in Table 1. Both workloads use $TT_{dynamic}$ in the Search Result page.

$$\begin{gathered} TT_{dynamic} = -ln(r) * p \ , \ where \ (0 < r < 1) \ \wedge \\ (p = 7 + \frac{Number\ of\ Search\ Results}{Max.\ Search\ Results}) \end{gathered} \tag{2}$$

The remaining cases of dynamism have been characterized using conditional transitions with DWEB. The transition from the Buy Request page to the Buy Confirm page depends on the last customer's purchase. If the customer does not buy any book during a given month, he has to commit the buying process, which means the end of the navigation session (*case 5*). Otherwise, he may finish the purchase or navigate to the Home page according to estimated probabilities of arcs as defined in *case 2* and shown in Figure 3 and Figure 4. Notice that when a customer does not remember the date of his last purchase, he must visit the *ordering* group in order to find out it, as defined in *case 1*. *Case 3* has been implemented with a conditional transition between the Product Detail and the Shopping Cart pages. This transition represents users adding a book to the shopping cart because its cost is 25% cheaper than in other markets. Finally, the back button has been characterized in LOYB workload as a cache that only stores the immediately previous listing page (*case 6*) as shown in Figure 4.

4.2 Optimizing user productivity: the parallel tab browsing behavior

Parallel browsing describes a behavior where users visit web pages in multiple concurrent threads by using web browsers tabs or windows. To help the understanding of how parallel browsing works, Figure 5 illustrates a parallel tab browsing session applied to the on-line bookstore. It shows how a web user uses parallel browsing based on three web browser tabs to improve his navigation time avoiding searches of books. The user begins a navigation session in a window with the aim of buying a book that fulfills several requirements as soon as possible. After he visits some pages, a search result is provided. At this point of time, he starts a parallel tab browsing behavior by opening three new tabs with the detail of three different books selected from the results. After that, he takes some time to evaluate the content of each tab until he finds a book satisfying his requirements, such as *Think Time 4*

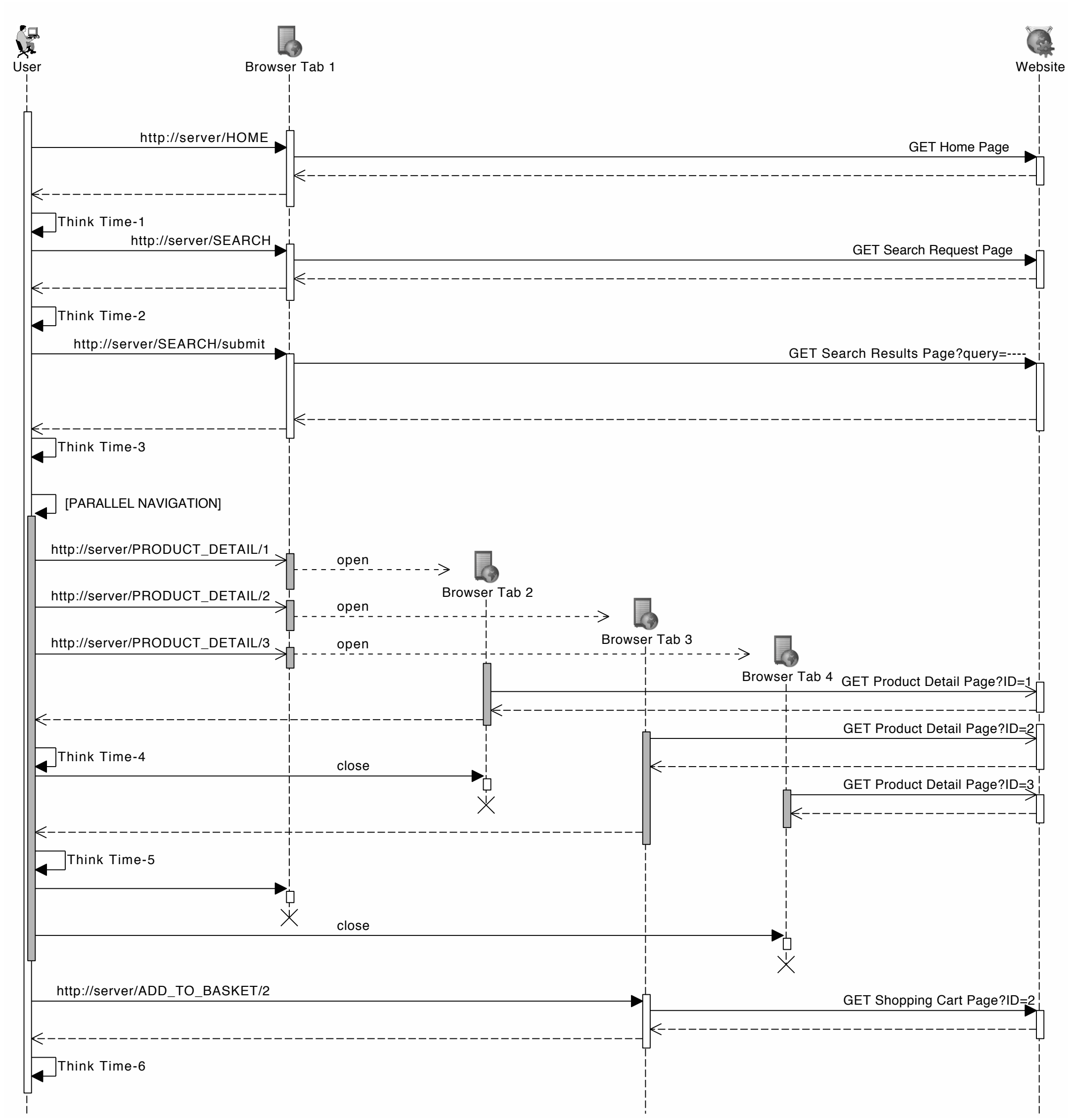

User
Browser Tab 1
Website
http://server/HOME
GET Home Page
Think Time-1
http://server/SEARCH
GET Search Request Page
Think Time-2
http://server/SEARCH/submit
GET Search Results Page?query=----
Think Time-3
[PARALLEL NAVIGATION]
http://server/PRODUCT_DETAIL/1
open
Browser Tab 2
http://server/PRODUCT_DETAIL/2
open
Browser Tab 3
http://server/PRODUCT_DETAIL/3
open
Browser Tab 4
GET Product Detail Page?ID=1
GET Product Detail Page?ID=2
Think Time-4
close
GET Product Detail Page?ID=3
Think Time-5
close
http://server/ADD_TO_BASKET/2
GET Shopping Cart Page?ID=2
Think Time-6

Fig. 5. Example of parallel tab browsing session

or Think Time 5 (see Figure 5) that are calculated according to equation (1). When a book does not fulfill the requirements, the user closes its associated tab and switches to the next one. Otherwise he discards the rest of tabs and adds the book to the shopping cart, finishing the parallel browsing session. Notice that the user does not take any *think time* on a book detail page when he discards its tab.

In the third scenario, we define a new DWEB workload (LOYT) that reproduces parallel tab browsing behavior in the loyalty promotion. With this aim, we extended the loyalty promotion behavior presented above by characterizing the navigation on web browser tabs as summarized in Table 2.

Figure 6 depicts the LOYT workload. This characterization defines the same behavior than LOY workload (Figure 3) except the navigations related to parallel tab browsing (*cases 6 and 7*). *Case 6* has been implemented with a pool of three parallel navigation threads, one for each tab. A navigation thread is killed when its book does not fulfill the user's requirements, or when the user finds the required book in other thread, as defined in *case 7*. The navigation becomes sequential again when only a thread is kept alive. Notice that even though LOYT workload does not consider the possibility of opening new tabs from another existing tab in this study, DWEB and GUERNICA provide mechanisms to model and execute multi-level in tab branching, respectively, if required.

Table 2. Extra cases of dynamism in the loyalty promotion behavior to represent parallel tab browsing

Case	**Description**
6	When a user has to review a listing of books, such as the result of a search or a browsing request, he begins a parallel tab browsing session with three tabs.
7	A user closes a tab when its book does not fulfill his buying requirements or when he has found the the wished book in other tab.

5 Impact of UBI on web performance

Experimental tests have been carried out to compare the performance achieved with the loyalty promotion workload (LOY) against those of the extended versions (LOYB and LOYT) which consider UBI when modeling the user's behavior. With the aim of finding out the stress borderline of the server for each workload, we varied the number of users ranging from 50 to 250 in 50-user steps.

Additional metrics measured at the client side have been defined in order to quantify the user's productivity, such as *number of finished sessions*, *session length* and *number of visited pages per session*. Notice that a navigation session finishes when a user leaves the website after his goals are achieved; that is, after he buys some books in the case study. Thus, the higher the number of finished sessions, the higher the user's productivity.

Although we measured all the performance metrics summarized in Section 3, only those showing significant differences in the studied workloads are discussed below.

Figure 7 shows how the user's productivity (measured in number of finished sessions) increases when considering UBI on workload characterization, specially when a parallel browsing behavior is introduced, because both the *session length (seconds)* and the *number of visited pages per session* are lower in these workloads (see Table 3 considering 100 simultaneous users as example). Therefore, user's productivity is improved when the user changes his navigation patterns as a result of parallel browsing behavior (up to 200%) or using the back button (up to 100%).

In general, the extended workloads also increase the server throughput, despite of the LOY workload degrades the service conditions more than considering the interactive behaviors.

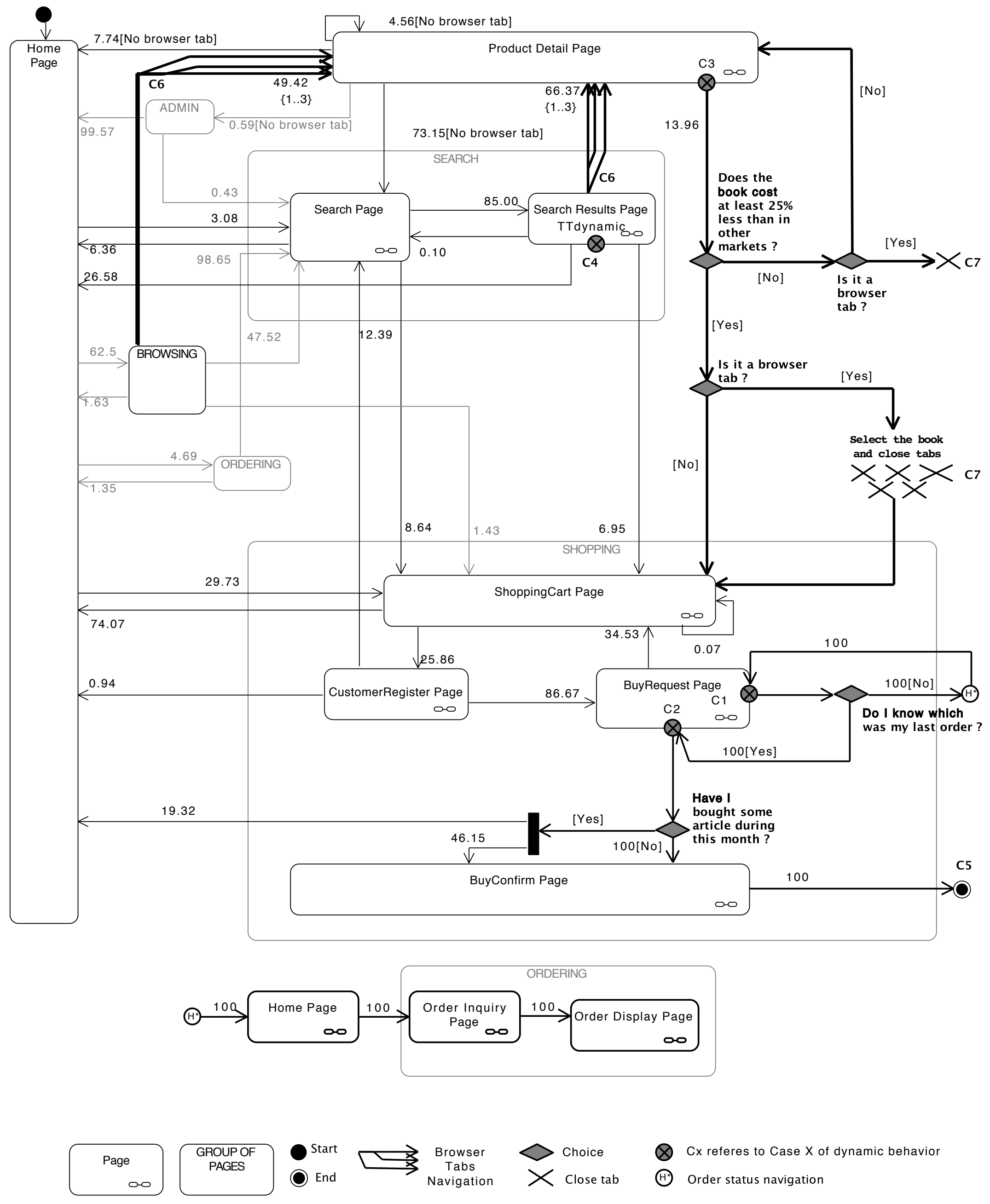


Fig. 6. LOYT workload: parallel tab browsing behavior in loyalty promotion

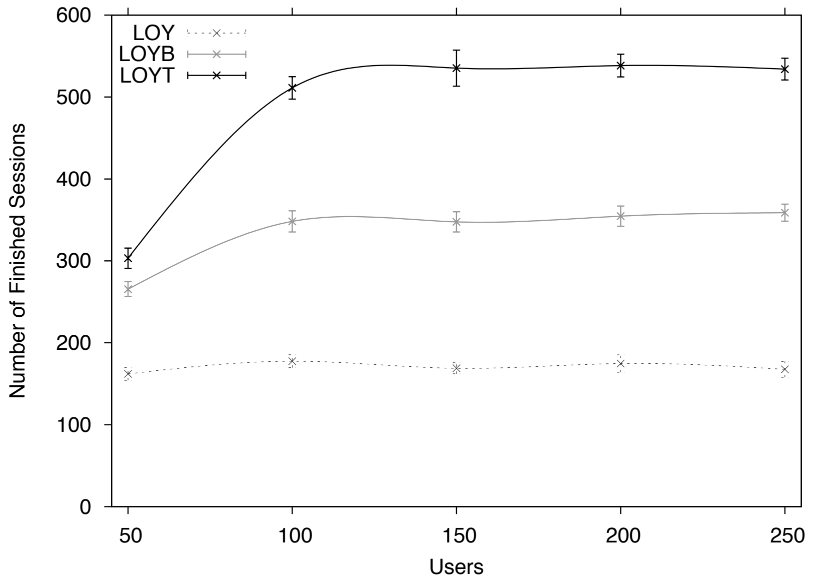


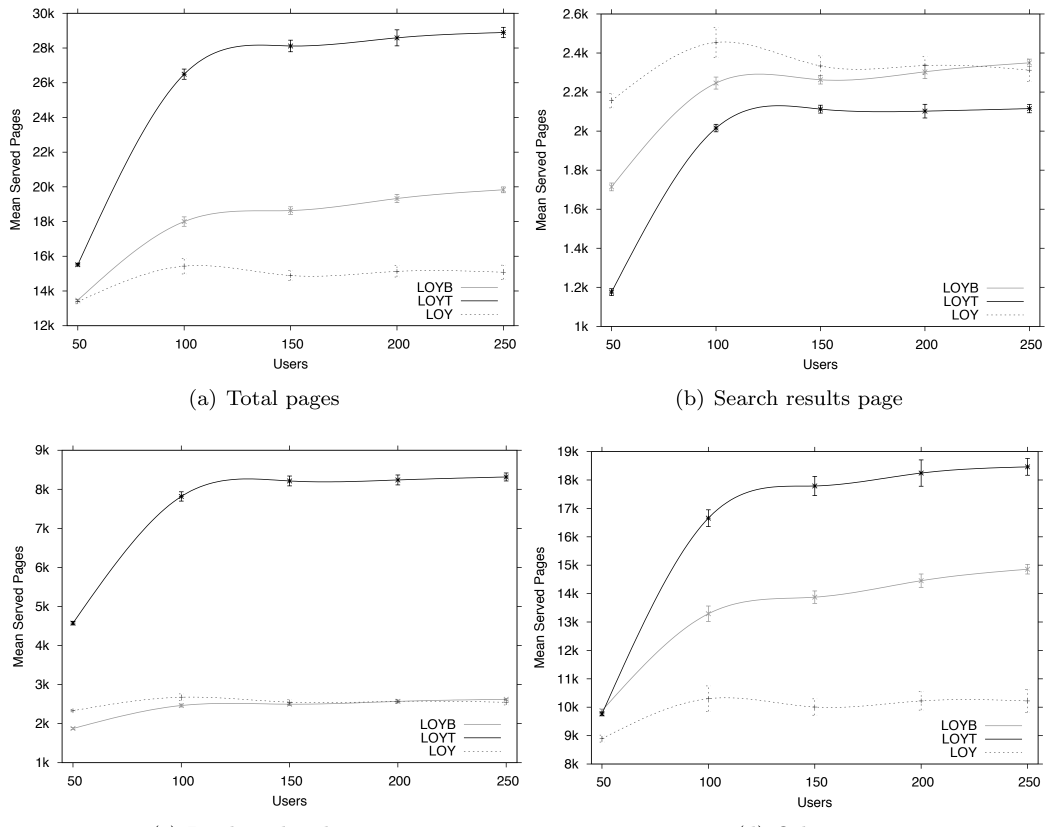
Fig. 7. User's productivity evolution

Table 3. Mean user productivity considering 100 simultaneous users in the system

Metric	**LOY**	**LOYB**	**LOYT**
Number of finished sessions	177.530	348.180	526.380
Session length (sec)	598.024	230.032	132.839
Number of visited pages per session	78.909	26.146	18.814

Figure 8 shows the *mean served pages* per experiment. Considering a significative number of simultaneous users (e.g. from 100 users on), the *total served pages* is on mean by 25% and 90% higher for the extended workloads (Figure 8(a)). To identify the causes of this high increase, served pages are classified in three main types: *search results page*, *product detail page* and *others*. As observed, the interactive behaviors generate less requests to the search engine (Figure 8(b)) and increase the number of requests to the product detail page (Figure 8(c)) and to the others (Figure 8(d)), specially for the LOYT workload. This new pattern of HTTP requests reduces the complexity of database queries when decreasing searches, so the throughput of the web server increases as shown in Figure 9(a). Specifically, the *Apache HTTP requests per second* generated by the LOY workload is by 25% and 75% lower than those generated by the LOYB and the LOYT workloads, respectively.

Figure 9(b) shows the *CPU utilization*, which is the only element that presents significant differences among the main hardware resources. Stress conditions have been classified in three levels according to the CPU utilization values: low stress ($U_{CPU} < 50\%$), significant stress ($50\% \leq U_{CPU} \leq 80\%$), and high stress - here by overload at server - ($U_{CPU} > 80\%$). As observed, the processor utilization for the extended workloads is always lower than for the LOY workload. For instance, considering 100 users, the utilization decreases by 15% and 45% for the LOYB and the LOYT workloads, respectively. Notice that the high CPU utilization value denotes that the processor acts as the main performance bottleneck but the



(a) Total pages

(b) Search results page

(c) Product detail page

(d) Others

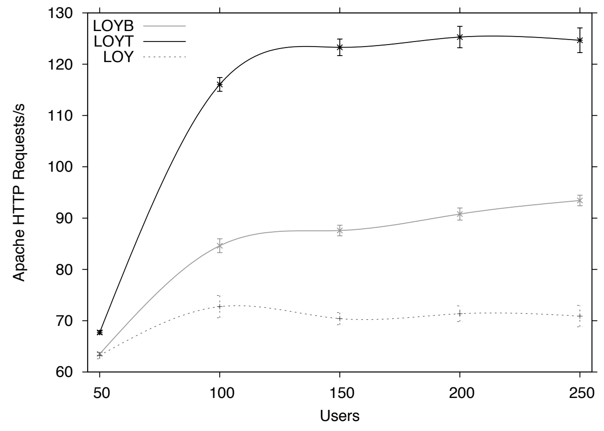
Fig. 8. Mean Served pages

stress borderline moves from 100 users for the LOY workload to 150 and 200 users for the LOYB and the LOYT workloads, respectively. This means that the web server allows the system to serve about 50% and 100% more users for the LOYB and the LOYT workloads.

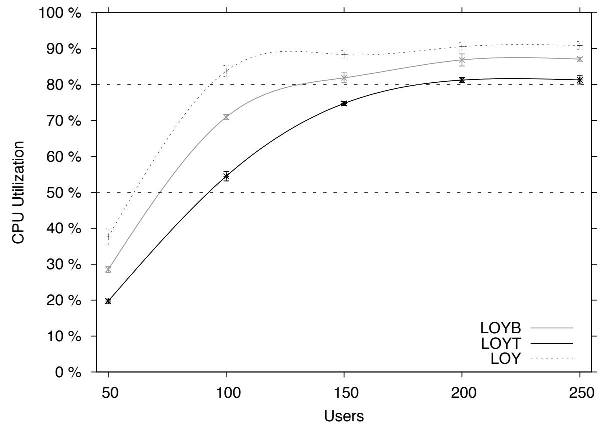
To better understand why the CPU utilization decreases, we studied how the main software components (Apache, Tomcat and MySQL) make use of the processor. As observed in Table 4, MySQL almost monopolizes the processor for the different workloads since its execution time is more than two orders of magnitude higher than the time devoted to Tomcat and Apache. Consequently, MySQL database is the major candidate to be a software bottleneck. However, despite of the executed queries rate for the extended workloads can be as large as 30% and 93% higher than for the LOY workload (Figure 9(c)), the CPU time consumed by MySQL for the extended workloads is by 30% and 55% lower than for the LOY workload.

Moreover, the CPU consumption of Tomcat and Apache increases for the extended workloads. That is, there is a change in the patterns of HTTP requests and consequently in the type of executed queries at MySQL.

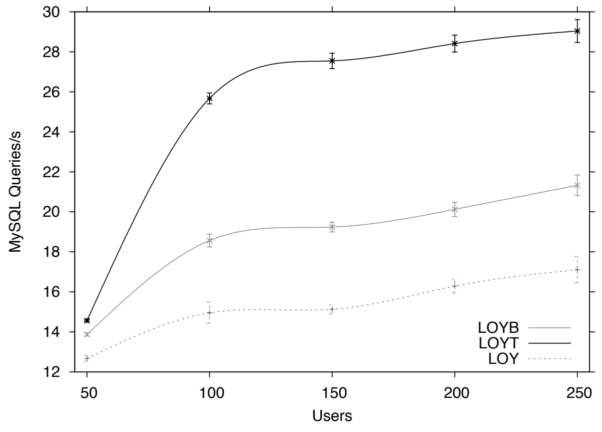
For a deeper study, we analyze how the database is used by these workloads. MySQL database includes `qcache`[25] as a cache of executed queries, where a query results in *hit* or *miss*. Figure 10 shows the mean execution time for hits, misses and total queries considering



(a) Apache throughput



(b) CPU utilization



(c) MySQL Throughput

Fig. 9. Server performance

Table 4. Main software components CPU time (jiffies)

(a) LOY

Soft. \ Users	**50**	**100**	**150**	**200**	**250**
MySQL	1315	2883	2807	2768	2897
Tomcat	10	22	22	23	23
Apache	4	9	10	10	10

(b) LOYB

Soft. \ Users	**50**	**100**	**150**	**200**	**250**
MySQL	931	2439	2606	2540	2624
Tomcat	9	24	26	27	28
Apache	3	9	11	12	13

(c) LOYT

Soft. \ Users	**50**	**100**	**150**	**200**	**250**
MySQL	567	1808	2381	2434	2424
Tomcat	8	27	33	34	34
Apache	3	10	15	17	18

50, 100 and 250 simultaneous users in the system as examples of the different stress conditions: low stress (Figure 10(a)), significant stress (Figure 10(b)) and overload at server (Figure 10(c)). As observed, the mean execution time for a query when considering the extended workloads is always lower than for LOY workload because decreasing the number of searches reduces the complexity, on average, of misses (the execution time decreases for the extended workloads) and increases the number of hits.

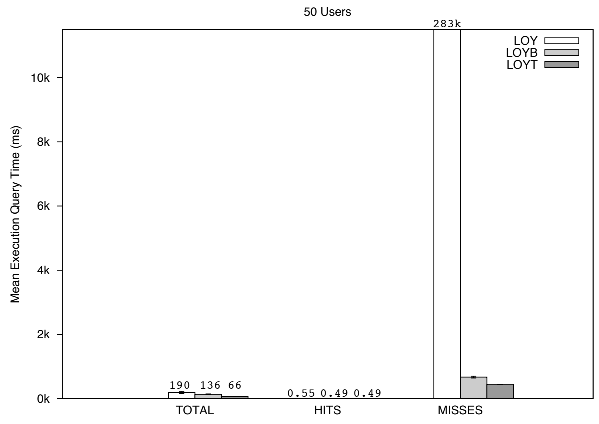
In summary, results show that considering user interaction with web browser facilities positively affects the system performance. This is because of considering UBI when modeling user's behavior introduces new patterns of HTTP requests. These patterns, in general, increase the number of requests to objects but reduce requests to the search engine allowing users to achieve a higher productivity. This fact causes noticeable differences in the performance metrics, specially in the processor usage and the server throughput. As a result, the stress borderline of the server is affected.

6 Conclusions and future work

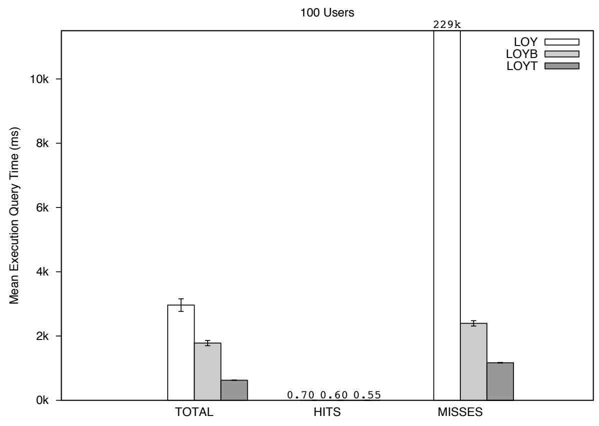
The web evolution introduces a new paradigm where users become contributors to the dynamic contents and services offered. As a result, the behavior of users and their interaction patterns have evolved and are more active and dynamic. Consequently, web performance studies have to take this new user behavior into account in the workload models to guarantee the validity of the results.

This paper has explored the effects of considering User-Browser Interaction in web performance evaluation studies. To this end, a scenario based on a typical e-commerce website has been recreated and a dynamic workload conducted by user's goals has been reproduced. Additionally, with the aim of improving the user's productivity, we included in the navigation patterns the use of the back button and the parallel tab browsing, as examples of user interactions with web browsers facilities. DWEB model has been used to take these behaviors into account in an easy and flexible way when modeling web browsing patterns.

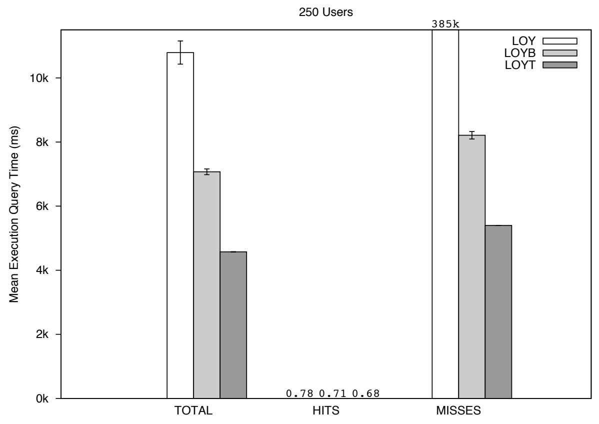
The study has considered a wide set of traditional performance metrics measured both at



(a) Low stress conditions



(b) Significative stress conditions



(c) Overload at server

Fig. 10. Execution time per query type

client side and server side. In addition, new metrics have been defined in order to quantify the productivity in the web navigations from the user point of view, such as number of finished session, number of visited pages per session or session length.

This paper proved that navigations using the back button or opening new tabs, which result from considering a user dynamic interaction with the provided contents, clearly allow users to achieve their goals in less time, so increasing productivity. These browsing patterns also affect the utilization and the throughput of the main system resources, and consequently the stress borderline on the server. Experimental results have shown that parallel tab browsing behavior noticeably increases the user's productivity (measured in number of finished sessions) up to 200% with respect to browsing the website in a sequential way. The new navigation patterns also increase the number of served pages (up to 90%). Nevertheless, they generate less requests to the search engine so reducing the complexity of the executed database queries and decreasing their execution time. This means an important drop in the processor utilization (up to 45%) and a noticeable rise in the web server throughput (up to 75%). As a consequence, the stress borderline is relaxed permitting the system either to support more applications or serve more users.

As future work, we plan to focus on a deeper evaluation of the effect of dynamic user workloads on the web performance, exploiting the characteristics and capabilities of DWEB model in a wider way.

Acknowledgements

This work has been partially supported by the Spanish Ministry Economy and Competitiveness under grant TIN-2013-43913-R.

References

1. Graham Cormode and Balachander Krishnamurthy. Key differences between Web 1.0 and Web 2.0. *First Monday Journal*, 13(6), 2008.
2. Pablo Rodriguez. Web Infrastructure for the 21st Century. In *Conference on World Wide Web*. Telefónica I+D, January 2009.
3. Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. Off the beaten tracks: exploring three aspects of web navigation. In *Conference on World Wide Web*, pages 133–142, May 2006.
4. Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *SIGMETRICS '98/PERFORMANCE '98, Joint International Conference on Measurement and Modeling of Computer Systems*, pages 151–160, 1998.
5. Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil Salinas. Dweb model: representing Web 2.0 dynamism. *Computer Communications Journal*, 32(6):1118–1128, April 2009.
6. Raúl Peña-Ortiz, José Antonio Gil Salinas, Julio Sahuquillo, and Ana Pont. The impact of user's dynamic behavior on web performance. In *IEEE International Symposium on Network Computing and Applications*, August 2012.
7. Sharad Goel, Andrei Broder, Evgeniy Gabrilovich, and Bo Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In *ACM International Conference on Web Search and Data Mining*, pages 201–210. Yahoo Research, 2010.
8. Daniel A Menascé and Virgilio A F Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, 2000.
9. Fernando Duarte, Bernardo Mattos, Jussara Almeida, Virgilio Augusto Fernandes Almeida, Mariela Curiel, and Azer Bestavros. Hierarchical characterization and generation of blogosphere workloads. Technical report, October 2008.

10. Mahnaz Shams, Diwakar Krishnamurthy, and Behrouz Far. A model-based approach for testing the performance of web applications. In *International Workshop on Software Quality Assurance*, pages 54–61, November 2006.
11. Fabricio Benevenuto, Tiago Rodrigues de Magalhães, Meeyoung Cha, and et al. Characterizing user navigation and interactions in online social networks. *Information Sciences*, 195:1–24, July 2012.
12. Jeff Huang and Ryen W. White. Parallel browsing behavior on the web. In *Conference on Hypertext and hypermedia*, pages 13–18. ACM, June 2010.
13. Anne Aula, Natalie Jhaveri, and Mika Käki. Information Search and Re-access Strategies of Experienced Web Users. In *Conference on World Wide Web*, pages 583–592, May 2005.
14. Andrew Thatcher. Web search strategies: The influence of Web experience and task type. *Information Processing & Management*, 44(3), May 2008.
15. H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *Conference on Human factors in computing systems*, pages 597–606, April-May 2007.
16. E. Arroyo, T. Selker, and W. Wei. Usability tool for analysis of web designs using mouse tracks. In *Conference on Human factors in computing systems*, pages 484–489. ACM, April 2006.
17. G.A. Di Lucca and M. Di Penta. Considering browser interaction in web application testing. In *International Workshop on Web Site Evolution*, pages 74–81. IEEE, September 2003.
18. Y.W. Seo and B.T. Zhang. Learning user's preferences by analyzing Web-browsing behaviors. In *International conference on autonomous agents*, pages 381–387. ACM, June 2000.
19. Raúl Peña-Ortiz, José Antonio Gil Salinas, Julio Sahuquillo, and Ana Pont. Providing TPC-W with web user dynamic behavior. *CLEI electronic journal*, 15(2):1–12, August 2012.
20. The Web Architecture Research Group. GUERNICA and TPC-W integration. [online] `http://www.gii.upv.es/web_architecture/tools/GUERNICA_TPCW.zip`.
21. Collectd – The system statistics collection daemon. [online] `http://collectd.org/`.
22. Andy Cockburn and Saul Greenberg. Issues of Page Representation and Organisation in Web Browser's Revisitation Tools. *Australasian Journal of Information Systems*, 7(2), May 2000.
23. Frederick F. Reichheld and P. Schefter. E-Loyalty: Your Secret Weapon on the Web. *Harvard Business Review Magazine*, 78:105, December 2000.
24. TPC BENCHMARK(TM) W Specification. Technical report, Transaction Processing Performance Council, February 2002.
25. MySQL documentation: How the Query Cache Operates. [online] `http://dev.mysql.com/doc/refman/5.1/en/query-cache-operation.html`.