Priority scheduling of requests to web portals

D.R.W. Holton\textsuperscript{a,\ast}, M. Younas\textsuperscript{b}, I.U. Awan\textsuperscript{a,\ast}

\textsuperscript{a} Department of Computing, The University of Bradford, Bradford BD7 1DP, UK
\textsuperscript{b} Department of Computing, Oxford Brookes University, Oxford OX3 3LB, UK
\textsuperscript{c} Computer Science, KSU, Saudi Arabia

\textbf{A R T I C L E I N F O}

Article history:
Received 16 September 2009
Received in revised form 28 February 2011
Accepted 25 March 2011
Available online xxx

Keywords:
ECommerce systems
Multi-agent simulation
Performance analysis

\textbf{A B S T R A C T}

Web portals work as a point of access to a large volume of information on the web. This paper focuses on the performance of Web portals in an E-commerce environment which involves the processing of a large number of users' requests. It proposes a class-based priority scheme which classifies users' requests into high and low priorities. In E-commerce, some requests (e.g. buy) are generally considered more important than others (e.g. search or browse). We contend that the requests received from a Web portal should generally get higher priority as such requests are more likely to lead to purchases. We believe that assigning such priorities at multiple service levels can improve the performance of Web portals' requests of higher priority. The proposed scheme is formally specified and implemented, and performance results are obtained and compared to a server that does not prioritise requests. The results show significant performance improvements in the processing of high priority requests.

\textcopyright 2011 Elsevier Inc. All rights reserved.

1. Introduction

Web portals collect large volume of information from various distributed sources and present them to the users. Various portals have been developed for various purposes. These include, for example, Yahoo, Opodo.com, and Expedia.com. Portals are useful as they provide a centralised application to access information from various sources distributed across the Web. In this paper, we focus on the performance of portals in the E-commerce environment. Various portals have been developed to provide a one stop facility for E-commerce services such as insurance quotes, flight booking, and hotel reservation. These portals process large number of users' requests which incur excessive load on the underlying servers. Such excessive load generally results in performance degradation of the servers, i.e. response time of requests may increase or servers may drop requests. We aim to alleviate this problem by proposing a class-based priority scheme which classifies requests into high and low priority requests. Generally, some E-commerce requests (e.g. buy) are considered more vital than others (e.g. search or browse). Thus it is crucial to give priority to those vital requests as they are financially important to the E-commerce service providers—their response time must be improved and they should not be dropped. However, request classification gives rise to another issue, i.e. how to determine that a particular request will make purchases? Current approaches employ various methods in order to deal with this issue. For instance, a common method is to use registration information to classify requests into high or low priority (buy or search) requests. That is, registered users are more likely to buy as compared to non-registered users. However, it is unrealistic to assume that registered users will buy items each time they visit an E-commerce web site. In this paper, we address this issue by assigning priorities based on the requests received from Web portals. Our premise is that the requests received from a Web portal should generally get higher priority as such requests are more likely to make purchases. According to a popular Web portal, http://www.travelsupermarket.com/ “... we are able to offer product providers and advertisers access to considerable volumes of informed consumers who, having compared products, are potentially ready to make a purchasing decision”. http://www.travelsupermarket.com is a leading UK travel price comparison website which covers 500 airlines and 34 flight service providers.

The work presented in this paper is a continuation of our previous work (Younas et al., 2008; Awan and Younas, 2004; Younas and Awan, 2003). Here it is extended as follows:

1. The scheme is extended in order to model the performance of Web portals in the E-commerce environment.
2. The scheme has been formally specified to allow us to analyse the correctness of the scheme and to provide a framework for the implementation.
3. An implementation has been constructed to supplement the analytical work performed on the scheme.

---

\textsuperscript{\ast} Corresponding author.
\textsuperscript{\ast \ast} E-mail addresses: dr.w.holton@bradford.ac.uk (D.R.W. Holton), m.younas@brookes.ac.uk (M. Younas), i.u.awan@bradford.ac.uk (I.U. Awan).

The remainder of the paper is structured as follows. Section 2 presents an overview of the architecture of web portals, and Section 3 surveys related work. Sections 4 and 5 describe the scheme and analyse the experimental results. In Section 6 a more realistic model of client behaviour is investigated. Finally Section 7 presents our conclusions.

2. Architecture of web portals

This section presents a generalised architecture of Web portals and the processing of requests in an E-commerce environment. The aim is to give readers an understanding of the working mechanism of Web portals.

The generalised architecture is shown in Fig. 1. In Web portals, users interact with (E-commerce) web servers through a series of requests in order to acquire information or buy products. For example, to find a mobile phone from a web site, a user first either enters a keyword or clicks on a category link of a required mobile phone. The user’s request is sent to the web portal. The web portal collects the required information from the respective web servers.

Web servers generally use application servers (e.g. BEA WebLogic, IBM WebSphere) which are commonly used to generate dynamic contents from databases. Note that the user’s request can be directly sent to the web sever as represented by the dotted lines in Fig. 1.

3. Related work

This section reviews related work on request classification mechanisms that have been developed in order to improve the performance of web-based systems. The rationale behind request classification is that the number of low priority requests (such as search or browse) is significantly higher than higher priority requests. For example, Menasce et al. (1999) shows that the percentage of customers who buy items is significantly lower than those who usually browse or search the web for information such as finding air fares or book prices. Such a large number of search and browse requests has performance consequences for E-commerce web servers as they affect the response time of high priority requests.

A priority mechanism for transactions in classical database systems is proposed in McWherter et al. (2004). This work presents a detailed analysis of the resource utilisation by transactions in a database system. It also improves the performance of high priority transactions in classical database systems. However, this approach is employed at the database level and it does not take into account web portals and E-commerce applications.

Our previous work (Younas and Awan, 2003; Awan and Younas, 2004) employs active network priority scheduling mechanisms in order to improve the performance of transaction commit protocols in Web-database applications. These approaches give preferential treatment to the processing of decision messages (such as transaction commit, abort, compensate) over data related messages.

The aim of Elnikety et al. (2004) is to improve the performance of E-commerce applications. This work proposes a method for admission control and request scheduling for multi-tiered e-commerce applications. This method differentiates between the different types of requests and devises a preferential scheduling policy in order to assign different priorities to different requests. A proxy server, called Gatekeeper, is developed that enables admission control and implements the preferential scheduling policy. The preferential scheduling is based on the shortest job first (SJF) policy. However, such a policy may fail to improve the response time if e-commerce requests are homogeneous, that is, requiring the same service time.

A Customer Behaviour Model Graph (CBMG) is proposed in Menasce et al. (1999) that describes the behaviour of customers who follow similar navigational patterns in submitting requests to E-commerce web sites. In these sites on-line shoppers issue requests such as browse, search, and pay. CBMG is used to describe the sequence of such requests. CBMGs are constructed by analysing logs of an E-commerce site that contains information related to a user’s profile based on their previous navigation patterns. This approach incurs processing overhead in constructing the CBMG using the past information stored in the log files that describes the customers’ profiles—a CBMG is constructed even if a customer visiting a web site does not buy items. Another alternative is to use registration information to classify customers into occasional buyers or heavy buyers. That is, registered users are more likely to buy as compared to non-registered users who are less likely to buy. However, it is unrealistic to assume that registered users will buy items each time they visit an E-commerce web site.

A two-dimensional (2D) service for classifying on line transactions into inter-session and intra-session transactions is proposed in Zhou et al. (2004). The former provide differentiated QoS to the sessions according to customer classes, while the latter provide differentiated QoS according to states of a particular session.

A LIFO-Pri priority scheduling scheme is proposed in Singhmar et al. (2004) in order to give service priority to buy requests over browse requests. This scheme is based on a large number of queues which are extremely difficult to manage. The proposed scheme works by moving buy requests from one queue to another queue based on its current state during its processing. This requires that requests are tracked throughout their entire execution. It may be manageable for small numbers of requests but will show performance degradation for larger numbers.

4. The proposed scheme

This paper investigates a class based priority scheme based on multilevel queueing (Silberschatz and Peterson, 2004) for multiple classes of web requests with the intention of providing a simple, easily implemented method for reducing the response time for high priority requests without causing starvation for the low priority requests. The scheme works as follows (Fig. 2):

1. requests are classified into two types (search or browse requests, and buy requests—with all requests coming from portals being treated as buy requests) and directed to a queue devoted to holding that type of request (thus there are two distinct queues);
2. if the queue containing high priority requests is not empty and we have not processed c consecutive high priority requests, then the first request in that queue is processed;
3. if the queue containing high priority requests is empty or c consecutive high priority requests have been processed, then process the first b requests from the low priority queue.

Putting a limit on the number of consecutive high priority requests processed prevents starvation for low priority requests, and varying that limit, along with adjusting queue lengths, allows the system to be tuned for various traffic conditions.

The \( \pi \)-calculus (Milner, 1999) was used to specify the scheme. The motivation for constructing a formal specification was to allow us to rapidly investigate a number of alternative schemes, and to act as a framework for the implementation. The \( \pi \)-calculus was chosen because of its support for mobility (to be addressed in future work).
4.1. The client

In order to collect experimental results of the performance of the scheme, it was necessary to specify a client component (Specification 1). The behaviour of the client is independent of the protocol used to process its requests, and is simple: it sends a number \( n \) of browse and buy requests to a server, waiting for a response to each request before sending the next. After all \( n \) requests have been processed the client terminates. The client sends its response action as a parameter so it can receive a reply from the server once the request has been processed. The specification shows the client making a non-deterministic choice between browse and buy requests; this is appropriate since the purpose of this specification is to describe how the components may interact, not why a particular action occurs. The implementation replaces this non-deterministic choice with one based on the relative probabilities of the two actions.

4.2. Server with unprioritised queueing

Before it can be shown that the proposed scheme improves on existing approaches, it is necessary to provide a point of comparison. This is done by examining the performance of a reference
The client server which serves requests in a strictly FIFO order, regardless of their priority.

The server is made up of a buffer component and a processor component. These components are composed to define the server, using a new action to connect them (Specification 2).

**Specification 2** The server

\[
\text{Server(request)} \triangleq \text{new connection Processor(connection)}
\]

The processor receives requests from the buffer, processes them, and responds to the appropriate client (Specification 3). As with the client, details of the processing of the request are abstracted from.

The buffer maintains a sequence of the requests in the order they arrive (Specification 5). If the buffer is full (i.e., has \( m \) elements in it) when a new request arrives, then the new request is dropped. A sequence is an ordered collection of items enclosed in square brackets. The concatenation operator (\( \triangleleft \)) which joins two sequences is used to enforce a FIFO ordering of requests.

**Specification 5** The gatekeeper

\[
\text{Gatekeeper(request, q\text{browse}, q\text{buy})} \triangleq \text{request(type, who)}.
\]

The components are composed as follows to create the new server (Specification 7). Note that the interface (i.e., the action request) to this version of the server is identical to the version with un prioritised queuing: the client would not be able to determine which version it was interacting with, yet it is intended that the improved protocol gives better utility to the operators of the server.

**Specification 7** The fair server

\[
\text{Server(request) = new buff\text{browse}, buff\text{buy}, buy, browse}
\]

### 4.4. Constructing the complete system

A population of clients and a single server are composed together (Specification 8), connected by a single action service used to request a service from the server, and a vector of actions reply to allow a response to be made to individual clients.

**Specification 8** The complete system

\[
\text{new service, reply}
\]

As described above, either definition of the Server component could be used in this composition, allowing for a true comparison of the performance of the two schemes.
5. Experimental results

The π-calculus models above were implemented in Ada following the procedure from Gilmore et al. (1996). The following performance results were obtained with the following parameters:

- the time taken to process high and low priority requests was the same;
- a total of 16 requests can be buffered, split equally between the two queues in the multilevel queueing scheme;
- the values of \( b \) and \( c \) were varied to allow performance results to be obtained for a range of experiments.

Since one goal was to design a simple scheduling scheme, it is necessary to show that the processing overhead it introduces is not excessive. Fig. 3 shows that there is no significant difference between the throughputs achieved by the reference scheme and the fair multilevel queueing scheme. As expected, the throughput increases as the number of active clients (and therefore traffic) increases, until the server is fully utilised and no further increase is possible.

The main aim of the work is to reduce the response time for high priority requests without causing starvation for low priority requests. Fig. 4 shows that this aim has been achieved. First consider the response time for both high and low priority requests in the reference scheme (Simple Server): response time is proportional to how many requests are queued. As the traffic increases the buffer fills and the response time increases. When the processor becomes fully utilised there is a very sharp increase in the response time as the buffer fills more quickly. After this point the increase in response time tails off because the number of requests being dropped increases as can be seen in Fig. 5.

Contrast this with the response times obtained by the fair multilevel queueing scheme. The response time for the high priority requests is greatly reduced, while the response time for the low priority requests initially follows that for the reference scheme but with the rate of increase levelling out earlier. This improvement in the response times for both high and low priority requests is gained at the expense of the number of low priority requests that are dropped: they start to be dropped at a lower arrival rate and their number is always greater than the number of low priority requests dropped under the reference scheme. However, it is notable that no high priority requests are dropped using the fair multilevel queueing scheme. The final point worth noting from the results is that the response time is surprisingly insensitive to the number of consecutive high priority requests processed, with the biggest difference occurring when \( c \) is increased from one to two.

6. Investigating more complex client behaviour

The behaviour of the client defined in Specification 1 was identified as being too simplistic: in reality, interactive users of E-commerce systems are not willing to wait patiently for servers to respond, and if requests are rejected due to high loads they may use a different service. A new client was defined (Specification 9) that attempted to perform a transaction comprising \( k−1 \) browse requests followed by a single buy request. If no response is received after the client has waited a “reasonable” time, or if the server responds by rejecting a request, the transactional client might terminate (modelling a user moving to another site).
These are best illustrated by Fig. 6 which shows the number of high priority buy requests completed then more clients would reach their final buy request. With the fair server, the original client completes more buy requests, reasoning that if more browse requests were rejected before generating a buy request.

The behaviour of the transactional client was investigated in the same way as the original client, with significant differences noted. These are best illustrated by Fig. 6 which shows the number of high priority buy requests completed:

- With the simple server, the simple client completes fewer buy requests than the transactional client. This is due to the number of transactional clients terminating due to timeouts or rejected requests, reducing the pool of active clients and ultimately lessening the overload on the server. Consequently fewer requests from the transactional client are rejected, leading to more completions.
- With the fair server, the original client completes more buy requests than the transactional client. This is because the buy request is the final action of transactional clients, therefore many transactional clients will have terminated due to timeouts or rejected before generating a buy request.

This second point lead to investigating the notion of prioritising browse requests, reasoning that if more browse requests were completed then more clients would reach their final buy request.

7. Conclusions

The results obtained demonstrate that the performance of a system is dependent on the interactions between its components; if the behaviour of a component is varied then the performance of the system may exhibit unexpected changes. This type of emergent behaviour (Steels, 1990) has been studied extensively in other areas however the usual approach to performance analysis is to assume that traffic has given properties, and to analyse new protocols using that traffic. Although limited to a simulation study, this work suggests that this approach may not be best suited when the system being investigated is interactive in nature.

References


