

# Reduction of User Perceived Latency for a Dynamic and Personalized Site Using Web-Mining Techniques

Enrique Frías-Martínez and Vijay Karamcheti

Computer Science Department, Courant Institute of Mathematical Sciences  
New York University  
715 Broadway, New York, NY 10003 USA  
{frias, vijayk}@cs.nyu.edu

## ABSTRACT

Dynamic and personalized web-servers are becoming more important everyday, nevertheless they have an important drawback compared with their static counterparts: user perceived latency is higher because of the time needed to dynamically generate each page. Web-Mining techniques have been used to implement intelligent web services and to create more personalized environments for users. This paper proposes and develops a new web-mining based application: intelligent pre-generation of dynamic web pages. Our techniques reduce user perceived latency by pre-generating pages using models derived from past user behavior. In this paper we present and apply these ideas to a mid-sized personalized portal, *NYUHome*, which serves 44,000 members of the NYU community.

**KEY WORDS:** Web-Mining, Latency Reduction, Web Intelligence, Dynamic Servers, Personalized Servers

## 1. INTRODUCTION

Since dynamic and personalized web servers were introduced in the Internet [7], the number of users that access these kinds of servers and the overall percentage of traffic these servers are responsible for has increased every year. Nevertheless these servers have a very important drawback: the latency perceived by users is several times larger than the latency perceived in static content web servers. A large part of this latency arises from generating and assembling content personalized to user preferences.

Traditional solutions for reducing user perceived latency, like web caches, do not have the same benefits when used for dynamic and personalized content, because of the dynamic origin of the information [3]. To date, solutions to this problem have focused mainly on reusing the subdocuments that are part of the final page [2].

The web-mining based techniques described in this paper are a complementary solution to reducing user perceived latency. Our techniques enable pre-generation of pages, anticipating arrival of a user request. This means that when the request for the page arrives, the user does

not need to wait for the generation of the page, but rather only the time needed to send the page. This approach has the potential of making the response time of dynamic and personalized web-servers similar to their static counterparts. The decision of which page to pre-generate and when is made by user behavior models that capture user preferences and past behavior.

Two behavior models are presented in this paper. The first one, the Personalized Sequential Behavior Model, is used to predict the next page that a user will request once a session has been started. The second one, the Connection Time Model, is used to predict when a user is going to start a session.

The problem of predicting a user's next request is a common problem in Web-Mining. Many solutions have been proposed by different authors [9][12][11][5][13][14]. Nevertheless the problem of predicting when a user is going to start a session has not been as extensively studied. As far as we know, our algorithm is the first one that tries to tackle this problem.

The problem of reducing user perceived latency is important because it is related to user satisfaction. The reduction of latency implies a growth in user satisfaction. [15] showed that the problem of high latency cost e-commerce Web sites 4.35 billions annually in lost revenue. The same study also showed that decreasing the load time of a page by one second reduces the rate at which visitors abandon a site from 30% to 8%. These facts prove the usefulness of developing algorithms that are able to capture user behavior and use their predictions to pre-generate pages.

The techniques we propose have been implemented and evaluated in the context of traces collected from *NYUHome*, a dynamic and personalized web server that serves as the primary information portal for 44,000 users of the New York University. The average generation time for *NYUHome* has been observed to be 1.11 seconds. Our techniques have made it possible to reduce the average generation time to 0.68 seconds, almost a 40% reduction.

The rest of the paper is organized as follows. Section 2 presents *NYUHome* server and gives the main characteristics of its load. Section 3 introduces the algorithms used to capture user behavior: (1) prediction of next request and (2) connection pattern. Section 4

discusses the implementation of those algorithms in *NYUHome* and evaluates their benefits. Finally, in section 5 we summarize the article and discuss future work.

## 2. THE *NYUHome* WEB SITE

*NYUHome* is a site used widely by the New York University community and has more than 44,000 registered users. The site allows access to a wide variety of information ranging from news and stock information to research tools and web forums.

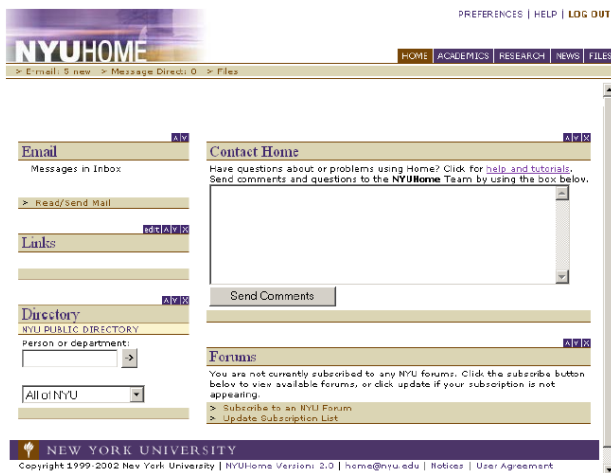
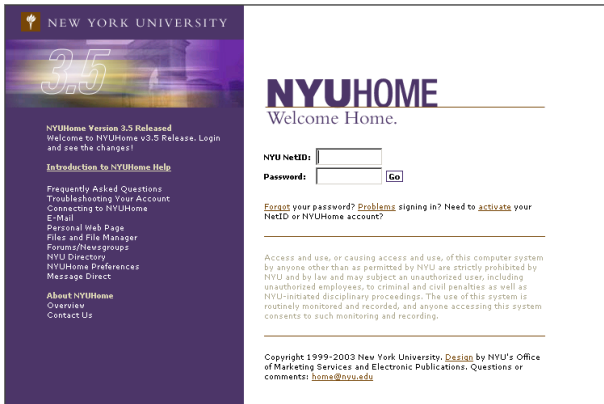


Fig. 1. (a) top: *NYUHome* Authentication Page, and (b) bottom: HOME tab of *NYUHome*.

Fig. 1(a) shows a snapshot of the authentication page of *NYUHome*. Once the user has been validated, the system generates the home tab, Fig. 1 (b).

As can be seen in the upper right corner of Fig. 1(b), *NYUHome* has 5 different tabs: HOME, ACADEMICS, RESEARCH, NEWS and FILES. Each one of these tabs has a set of channels that are presented by default in a two column layout. Each user can personalize the channels of each tab by choosing the layout. The channels of each tab include:

- HOME: Email, Directory, Links, Contact Home, Forums.

- ACADEMICS: Albert, Bookstore, Classes.
- RESEARCH: Directory, Search, Library.
- NEWS: Events, Finance, Horoscope, News, Sports, Weather.
- FILES: Web Page, Files.

A session with *NYUHome* starts with the authentication of the user. After that, the personal preferences of the user are used to construct each one of the tabs when requested. Depending on the channel, the content is obtained either by accessing the caches where the sharable channels are stored or by generating the channel from scratch for each user. The caches that store sharable channels (like Events, Horoscope, News, etc.) are updated periodically.

## 2.1 WORKLOAD CHARACTERIZATION OF *NYUHome*

The paper by Shi and Karamcheti [10] presents an analysis of the workload encountered at the *NYUHome* server using instrumented server logs. In this section, we summarize the main results obtained in [10] to motivate the techniques described in the rest of the paper. A description of the log format used can be found in section 4. For a complete description of the workload characterization refer to [10].

- 82.85% of the sessions served by *NYUHome* have length one (only access the HOME tab). This occurs mainly when *NYUHome* is being used to read email, which is accessed through the HOME tab but served by another machine. 99% of the sessions have length five or smaller. This means that 16.15% of the sessions have a length between two and five.
- 90.1% of the requests across all sessions are for the HOME tab. The remaining 9.9% is divided as follows: ACADEMICS 5.1%, RESEARCH 1.4%, NEWS 2.3% and FILES 1.1%.
- Within a session, the minimum interval between two consecutive requests is three seconds. 90% of the requests in a session have request intervals of five seconds or longer. 50% of the requests have an interval of more than 100 seconds.
- The average processing time of each request is independent of the presented load. The *NYUHome* server is operating far below its planned capacity. The server utilization never exceeds 30%.
- The total latency observed by a user,  $T$ , can be modeled as:

$$T = T_p + T_c, \quad (1)$$

where  $T_c$  is the overhead needed to send the document from the server to the user, and  $T_p$  the processing overhead needed to generate and assemble the document.

$T_p$  itself can be approximated using a simple behavior model:

$$T_p = n_c t_c + n_g t_g + t_a N, \quad (2)$$

where  $n_c$  is the number of channels served from a cache,  $n_g$  the number of channels dynamically generated,  $t_c$  the time needed to obtain the content from a cache,  $t_g$  the time needed to generate a channel,  $t_a$  the time needed to assemble the channel in the document and  $N=n_c+n_g$ .

Table 1 presents the average value of  $T_p$  for each tab and the Total average, yielding the relationships:  $t_c+t_a=0.329$  and  $t_g+t_a=0.523$ . These values will be later used in Section 4.2.1.

Table 1. Average Generation time for each tab.

Tab	Average $T_p$
Total	1.11
HOME	1.14
ACADEMICS	0.66
RESEARCH	0.48
NEWS	1.92
FILES	2.07

The ideas that are going to be presented in the following sections are aimed at making  $T_p \rightarrow 0$  by pre-generating the pages before they are requested.

### 3. REDUCING USER OBSERVED LATENCY WITH WEB-MINING TECHNIQUES

In a dynamic and/or personalized web server the connection between a client and a server has the following steps: (1) request from the client, (2) generation of the page requested in the server according to the preferences defined by the user and (3) sending the generated page to the user. Each step has a latency associated with it. Fig. 2 presents the communication flow and the associated overheads. In this case  $T_c=T_{c1}+T_{c2}$  and the total latency seen by the user is  $T=T_p+T_c$ .

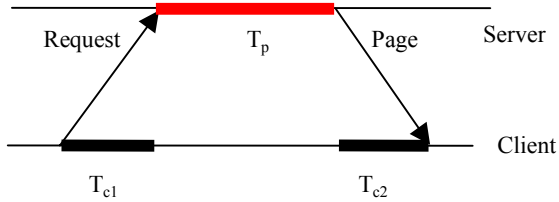


Fig. 2. Communication overheads between a client and a server.

In order to reduce  $T_p$  the ideal situation is expressed in Fig. 3 where the page requested has already been generated before the server receives the request. In this situation  $T=T_c$ . In order to pre-generate the page the server needs a behavior model that indicates which page is going to be requested in the next user click. The reduction is possible because the intelligent pre-generation system overlaps the time that a user spends on other activities,  $T_r$ , e.g. reading a previously requested page, with the time spent generating the next page.

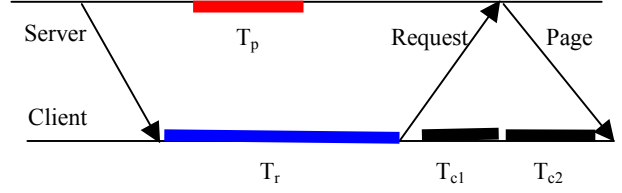


Fig. 3. Communication overheads between a client and a dynamic server with a prediction system.

The pre-generation of pages is based on two behavior models: (1) the Personalized Sequential Behavior Model, which predicts the next request of a user, and (2) the Connection Time Model, which predicts when a user will request the home page, in other words, when a user will start a session.

Both behavior models introduce additional load in the server. The extra load is caused by incorrect predictions because the server has to generate two pages: the predicted page, and the page requested. Fig. 4 presents an example of this situation.

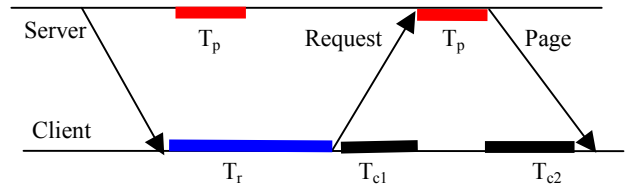


Fig. 4. Overhead introduced by an incorrect prediction.

In order to implement a pre-generation system, it is necessary to verify that the extra load introduced by both behavior models can be handled by the server. Also, in order to pre-generate the next request within a session, it is necessary to verify that there is enough time between two consecutive requests to pre-generate the page.

In the next two subsections the two algorithms needed to model user behavior are introduced.

#### 3.1 PERSONALIZED SEQUENTIAL BEHAVIOR MODEL (PSBM)

In [4] we presented a PSBM designed to predict user's next request. The model constructs rules which, unlike clustering [8][6] and association rules[1], capture sequentiality and temporality.

The basic idea of the PSBM is to create for each frequent user of the server a set of rules that have a support and a confidence bigger than a predetermined threshold. Each rule correlates an antecedent, the click stream of the user, with a consequent, which gives the predicted pages. A PSBM is defined by a set of parameters,  $|A|$ ,  $n$ ,  $|C|$ , and by the threshold of the support and confidence,  $\theta'$  and  $\sigma'$ . These parameters will be used in section 4.2.

In the case of *NYUHome*, the Personalized Sequential Behavior Model allows us to predict with high probability the next tab requested by a user.

### 3.2 CONNECTION TIME MODEL

In this section we describe a solution for the problem of predicting when a user is going to start a session.

Most personalized servers already keep information of each user, so it is natural to work with each user individually, instead of, for example, working with clusters of users. The algorithm will create a Connection Time Model for each user.

The algorithm we propose uses the concept of a burst of connections. A burst of connections is defined as a period of time when a user concentrates a large portion of the beginning of his sessions. A burst of connections is defined by two parameters: (1)  $M$ , the length of the burst measured in seconds and (2)  $N$ , the percentage of connections that should be included in that burst.

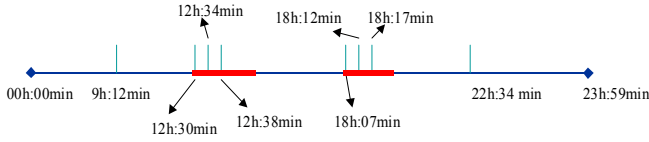


Fig. 5. Example of Burst of Connections.

Fig. 5 presents an example of burst of connections. The figure presents when the user requests the home page during a day. Let the burst of connections be defined with  $M=15 \text{ min.}$  and  $N=25\%$ . This means that a burst will concentrate at least 25% of the connections in a time interval spanning no more that 15 minutes. In this case, if the user has a total of 8 visits for the day, that means that a least 2 visits should be in the same 15 minute period. With this definition the user has two burst of connections: the first one goes from 12:30 to 12:45 and the second one from 18:07 to 18:22.

The set of burst of connections of each user will be generated using the algorithms presented in 3.2.1 and 3.2.2. We will use this information to predict when a user is most likely to request the home page in order to pre-generate this page before the predicted burst starts. As presented in Fig. 6, if the user does end up connecting to the site during the burst period, the request sees and effective  $T_p$  of 0.

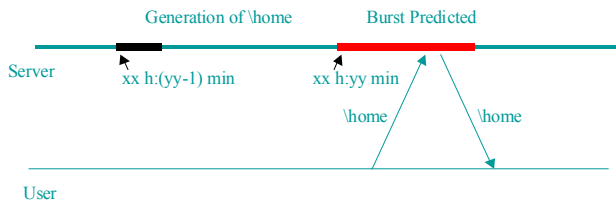


Fig. 6 Pre-generation of home.

The algorithm that constructs the Connection Time Model of each user is divided into two steps:

- “Slotting”&Clustering the set of home requests. Because users of different servers will have different connection patterns we propose different ways of

slotting (dividing into slots) and clustering the set of requests in order to better capture when a user is most likely to start a session. The selection of one of these techniques is server dependent.

- Detection of the Bursts of Connections for each cluster of each user.

The following subsections describe these two steps in more detail.

#### 3.2.1 SLOTTING&CLUSTERING THE SET OF HOME REQUESTS

In [4] we detailed how to transform a set of logs  $L$  expressed as,

$$L = \{L_1, \dots, L_{|L|}\}$$

$$L_i = (IP_i, LOGNAME_i, TIME_i, METHOD_i, URL_i, PROT_i, CODE_i, BYTES_i), \forall i / i = 1 \dots |L| . \quad (3)$$

into a set of sessions  $S$ ,

$$S = \{S_1, \dots, S_{|S|}\}, \quad (4)$$

where  $|L|$  is the number of log entries in  $L$  and  $|S|$  is the number of sessions of  $S$ . Each session is defined by a vector  $(USER\_ID, TIME, PAGES)$ , where  $USER\_ID$  identifies the user of the session,  $PAGES$  the set of pages requested and  $TIME$  expresses when the session started,

$$S_i = (USER\_ID_i, TIME_i, PAGES_i), i = 1 \dots |S|. \quad (5)$$

The set of different users of a system is given by,

$$USERS = \{USER_1, \dots, USER_{|USERS|}\}, \quad (6)$$

where  $|USERS|$  is the total number of different users of the log.

The set of sessions  $S$  will be grouped into  $|USERS|$  clusters, where each cluster  $S\_USER_i$  contains the connection times of the sessions of user  $i$ :

$$S\_USER_i = \cup (TIME_k \text{ of } S_k / USER\_ID_k \text{ of } S_k = USER_i), \quad (7)$$

$$\forall i = 1, \dots, |USERS|, \forall k = 1, \dots, |S|$$

The connection times of each  $S\_USER_i$  are going to be divided using a fixed slot time  $SLOT\_LENGTH$ . This slot time basically shows when we expect that the users are going to present repeated behavior:

$$S\_USER_i = \left\{ (SLOT_{i,1}, SECONDS_{i,1}), \dots, (SLOT_{i,k(i)}, SECONDS_{i,k(i)}) \right\}$$

$$SLOT_{i,k} = \text{int} \left( \frac{TIME_k}{SLOT\_LENGTH} \right), \forall k = 1, \dots, k(i) \quad (8)$$

$$SECONDS_{i,k} = \text{rest} \left( \frac{TIME_k}{SLOT\_LENGTH} \right), \forall k = 1, \dots, k(i)$$

where  $k(i)$  is the number of sessions of  $USER_i$ , and each tuple  $(SLOT, SECONDS)$  represents the start of a session. We define  $\text{int}$  as a function that returns the

integer part of the division, and *rest* as a function that returns the rest of the division.

We consider that the basic *SLOT\_LENGTH* is one day, because users will repeat their behavior on a daily basis. Nevertheless, in some cases other *SLOT\_LENGTH* may be useful, for example 12 hours.

The next step consists of clustering the set of connection times of each user  $S\_USER_i$  according to a cluster policy function  $P$ . The reason for clustering the connection times is to group the slots that we consider to have a common behavior.

$S\_USER\_C_i$  presents the result of clustering  $S\_USER_i$  according to the clustering policy function  $P$ , keeping only the *SECONDS* of connection in each cluster. Given  $p$  the number of clusters defined by the function  $P$ ,  $S\_USER\_C_i$  can be expressed as:

$$S\_USER\_C_i = \left\{ S\_USER\_C_{i,1}, \dots, S\_USER\_C_{i,p} \right\}, \quad \forall i = 1, \dots, |USERS|$$

$$S\_USER\_C_{i,k} = \left\{ \cup (SECONDS_j / P(SLOT_j, SECONDS_j) = P_k) \right\}, \quad \begin{matrix} k=1, \dots, p, \\ \forall j=1, \dots, k(i) \end{matrix} \quad (9)$$

$$P : (SLOT_j, SECONDS_j) \rightarrow \{P_1, \dots, P_p\}, \quad \forall j = 1, \dots, k(i)$$

In order to define the clustering policy function  $P$  it has to be considered that we can associate to *SLOT* more semantic than just a number, for example the day of the week. The definition of the possible clustering functions depend also on the *SLOT\_LENGTH* considered. For the most common case, in which *SLOT\_LENGTH* is defined as 1 day, some possible clustering functions  $P$  are:

- Just one cluster that contains the beginning of every session,

$$S\_USER\_C_i = \left\{ S\_USER\_C_{i,1} \right\}, \quad \forall i = 1, \dots, |USERS|$$

$$S\_USER\_C_{i,1} = \cup (SECONDS_j / SLOT_j \subset \{ANY\_DAY\}) \quad (10)$$

$$P : (SLOT_j, SECONDS_j) \rightarrow \{ANY\_DAY\}, \quad \forall j = 1, \dots, k(i)$$

- Two clusters, one containing the sessions that are made on a week day and the other containing the sessions that are made during the weekend,

$$S\_USER\_C_i = \left\{ S\_USER\_C_{i,1}, S\_USER\_C_{i,2} \right\}$$

$$S\_USER\_C_{i,1} = \cup (SECONDS_j / SLOT_j \subset \{WEEKDAYS\}), \quad \forall j = 1, \dots, k(i)$$

$$S\_USER\_C_{i,2} = \cup (SECONDS_j / SLOT_j \subset \{WEEKEND\}), \quad \forall j = 1, \dots, k(i) \quad (11)$$

$$P : (SLOT_j, SECONDS_j) \rightarrow \{WEEKDAYS, WEEKEND\}, \quad \forall j = 1, \dots, k(i)$$

- Seven clusters, each one containing the connection times of the sessions made each day, as expressed in (12).

Fig. 7 presents an example of how the home requests of a user during the training log, each one indicated with  $\blacktriangledown$ , are first divided into slots, and then the set of slots are clustered according to  $P$ . In the case of Fig. 7,  $P$  has been defined as in (11), which defines two clusters.

$$S\_USER\_C_i = \left\{ S\_USER\_C_{i,1}, S\_USER\_C_{i,2}, S\_USER\_C_{i,3}, S\_USER\_C_{i,4}, \dots, S\_USER\_C_{i,7} \right\}$$

$$S\_USER\_C_{i,1} = \cup (SECONDS_j / SLOT_j \subset \{MON\}), \quad \forall j=1, \dots, k(i) \quad (12)$$

$$\dots$$

$$S\_USER\_C_{i,7} = \cup (SECONDS_j / SLOT_j \subset \{SUN\}), \quad \forall j=1, \dots, k(i)$$

$$P(SLOT_j, SECONDS_j) \rightarrow \{MON, TUE, WED, THU, FRI, SAT, SUN\}, \quad \forall j=1, \dots, k(i)$$

The decision of which *SLOT\_LENGTH* and which clustering policy function to choose will depend on the behavior of the users of a server, in other words, is server dependent. For example, considering a *SLOT\_LENGTH* of 1 day, if we have a system that is mainly used during the weekdays, (11) would probably provide the optimum solution. If we have a server that is equally accessed on any day of the week and that does not have a large number of users, (10) will be useful. When trying to obtain the connection model of a heavy loaded server that is equally accessed any day of the week, (12) will be the most appropriate.

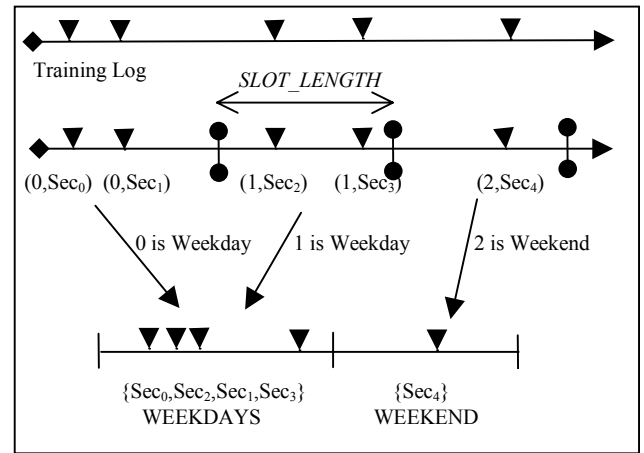


Fig. 7 Example of Slotting and Clustering of home requests.

### 3.2.2 BURST DETECTION ALGORITHM

The Burst Detection algorithm, given  $M$ , the length in seconds of the burst,  $N$ , the minimum percentage of home requests in a burst, and  $S\_USER\_C_i$ ,  $i=1, \dots, |USERS|$ , obtains the set of connection bursts for each user,  $USER\_BURST\_C_i$ ,  $i=1, \dots, |USERS|$ .

The input of the algorithm is:

$$S\_USER\_C_i = \left\{ S\_USER\_C_{i,1}, \dots, S\_USER\_C_{i,p} \right\}, \quad \forall i = 1, \dots, |USERS| \quad (13)$$

$$S\_USER\_C_{i,k} = \left\{ SECONDS_{i,k,1}, \dots, SECONDS_{i,k,e(i,k)} \right\}, \quad k = 1, \dots, p$$

where  $e(i,k)$  is the number of connections of user  $i$  in the cluster  $k$ . The output can be expressed as:

$$USER\_BURST\_C_i = \left\{ USER\_BURST\_C_{i,1}, \dots, USER\_BURST\_C_{i,p} \right\} \quad (14)$$

$$USER\_BURST\_C_{i,k} = \left\{ BURST_{i,k,1}, \dots, BURST_{i,k,b(i,k)} \right\}, \quad k = 1, \dots, p$$

where  $b(i,k)$  is the number of bursts of user  $i$  in cluster  $k$  and  $BURST_{i,k,b(i,k)}$  indicates when a burst starts for user  $i$  in cluster  $k$ . The length of the burst is defined by  $N$ . As

shown in Fig. 6, this information will be used for prediction purposes: the system will pre-generate the home page for user  $i$  in cluster  $k$  before each  $BURST_{i,k,j}$ ,  $j=1, \dots, b(i,k)$ .

Fig. 8 presents the algorithm used to obtain the burst of connections.

In this Section we have defined the concept of burst of connections using the parameters  $M$  and  $N$ . Nevertheless, the definition of the concept of burst can be done in different ways and with different parameters, in other words, is also application and server dependent.

```

Input : M,N,S_USER_Ci=1,...,|USERS|
Output: USER_BURST_Ci=1,...,|USERS|

function Obtain_Burst
  for each S_USER_Ci,i=1,...,|USERS|
    for each cluster S_USER_Ci,c,k=1,...,p
      USER_BURST_Ci,k = ∅
      for each SECONDSi,k,e,e=1,...,e(i,k)
        Counter=0
        for each SECONDSi,k,f,f=e,...,e(i,k)
          if SECONDSi,k,f ∈ [SECONDSi,k,e, SECONDSi,k,e+M]
            then Counter++ end if
          end for
          if (Counter/e(i,c) > N) then
            USER_BURST_Ci,c = USER_BURST_Ci,c ∪ SECONDSi,k,e
          end if
        end for
      end for
    end for
  end for

```

Fig. 8 Algorithm for Burst Detection.

### 3.2.3 DEFINING ELEMENTS OF THE CONNECTION TIME MODEL

The Connection Time Model has a set of parameters that should be selected in order to better capture user behavior. Those parameters are:

- *SLOT\_LENGTH* and clustering policy  $P$ . Depending on the behavior of the users, different slots and clusters will achieve better prediction rates.
- Burst definition:  $N$  and  $M$ . These values depend on how frequently a user accesses the site and how frequently the caches change. The base approach of generating a page at the beginning of a predicted burst guarantees that a page is at most  $M$  sec. old.
- Definition of Frequent Users. As stated in [4] not all users of a system are equally responsible for the load of the server. There is a core of users responsible for the majority of the load. One can trade off the space requirements of the algorithm against prediction accuracy by limiting its application to only these frequent users.

## 4. IMPLEMENTATION AND RESULT ANALYSIS

This section describes the implementation of the ideas presented for reducing latency in the *NYUHome* server.

### 4.1 LOG CHARACTERIZATION

The log file was collected from Friday, March 15th 2002 to Monday, April 1<sup>st</sup> 2002. The log file was obtained with an instrumented version of *NYUHome*, which made it possible to capture more information than standard logs. [10] presents a complete description of the modifications made to the server to capture the logs. Although [10] captures a great deal of information from each request, we are only going to use a subset of this information. For each request that the server receives, our log file stores:

$$\begin{aligned} &\langle \text{hash value of UserID} \rangle \langle \text{tabID} \rangle \\ &\langle \text{SessionKey} \rangle \langle \text{ArriveTime} \rangle \langle \text{Depart Time} \rangle \end{aligned} \quad (15)$$

where  $\langle \text{hash value of UserID} \rangle$  identifies the user that made the request,  $\langle \text{tabID} \rangle$  identifies the tab requested (HOME, ACADEMICS, RESEARCH, NEWS or FILES),  $\langle \text{SessionKey} \rangle$  identifies the session of the user,  $\langle \text{ArriveTime} \rangle$  gives in microseconds the arrival time of the request and  $\langle \text{Depart Time} \rangle$  the time at which the response was generated.

The log for the 18 day period has a size of 60M and contains about 920,000 requests. The time needed by the server to process each one of the requests, the processing overhead, is given by  $T_p$ ,

$$T_p = \langle \text{Depart Time} \rangle - \langle \text{Arrive Time} \rangle \quad (16)$$

Fig. 9 presents the number of home requests and the total number of requests during the 18 period day measured every thirty minutes.

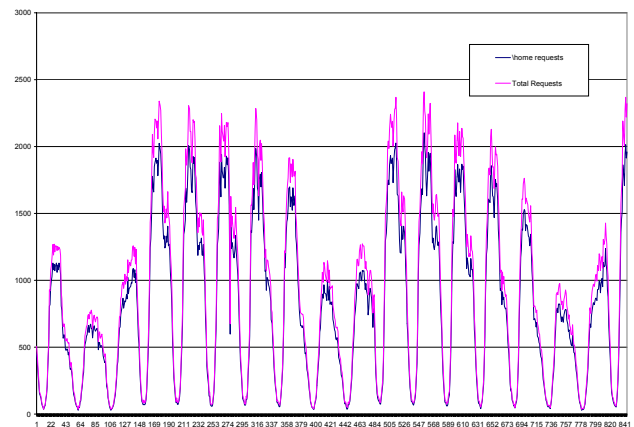


Fig. 9 Number of requests of *NYUHome* every thirty minutes.

As Stated in Section 2, the majority of the load is cause by home requests.

Each day is characterized by a peak which happen more or less between 1:00pm and 4:00pm which is in the middle of two local minimums which correspond to

3:00am to 5:00am of the same day and 3:00am to 5:00am of the following day.

In the first two days of the logs the number of visits is almost 50% less than on a regular Friday and Saturday. The reason for this is that the first three days of the log were the last days of Spring Break 2002, which as can be seen, definitely affects the load of the server. We hypothesize that the user behavior during these three days is very different from the regular behavior a user would display during the semester. In order to avoid polluting the behavior models with the atypical behavior of these three days we decided to discard the data corresponding to Friday March 15<sup>th</sup>, Saturday March 16<sup>th</sup> and Sunday March 17<sup>th</sup> 2002.

We consider the remaining 15 period day file representative of regular user behavior, which gives us around 800,000 requests to work with. From these valid requests, we take the first 75% as a training set of logs (600,000 requests), and the remaining 25% (200,000) as a testing log.

## 4.2 IMPACT OF PERSONALIZED SEQUENTIAL BEHAVIOR MODEL (PSBM)

In this section we present the benefits and costs associated with introducing the Personalized Sequential Behavior Model in *NYUHome*.

### 4.2.1 APPLICABILITY

Considering (2), which estimates the generation time of a tab, and the worst case scenario in which the non-cacheable channels have to be always generated from scratch, the generation time is 1.6 sec. for ACADEMICS (3\*0.53sec.), 1.6 sec. for RESEARCH (3\*0.53sec.), 1.06 sec. for FILES (2\*0.53sec.) and 2.65 sec. for NEWS (8\*0.33sec.). As mentioned in Section 2, in the worst case the request interval within a session is at least 3 seconds. This makes it possible to generate at least one page between the requests of two consecutive pages and also allows enough time to execute the prediction system. Taking into account that 90% of the requests have an interval bigger than five seconds, it is often possible to generate more than one page during the interval, which increases prediction accuracy and reduces latency.

Another question that has to be answered is how big will the benefit be after implementing the prediction system. Considering that only 9.9% of the requests can benefit from this mechanism (90.1% of the sessions have length 1), the average total generation time will not be reduced by more than 9.9%. Nevertheless, the mechanism will make it possible to significantly reduce the generation time of each tab.

### 4.2.2 BENEFITS FOR *NYUHome*

We have constructed a next request prediction system with the Personalized Sequential Behavior Model presented in [4] using the values  $|A|=I$ ,  $n=I$ ,  $|C|=I$ ,

$\theta'=1\%$  and  $\sigma'=5\%$ . More information about the model can be found in [4].

The total number of users presented in the training log was 26,714 from which only 11,665 produce a set of rules. The users are filtered because: (1) the user only accesses the home page, or (2) because the rules generated do not pass the  $\theta'=1\%$ ,  $\sigma'=5\%$  filter.

The total number of rules generated is 29,666, which gives an average of 2.5 rules per user. The total processing time of the training file was 4 min 30 seconds. Table 2 presents the results obtained by applying our PSBM to the testing log, considering only the requests of the users with a Prediction Time Model.

Table 2. Results Considering only requests of Users with a Next Request Prediction System.

TAB	C.P.R.	O.A.G.T.	A.G.T.w.P.S.	R.F.
ACADEMICS	0.8	0.66s	0.13 s	80%
RESEARCH	0.44	0.48s	0.25 s	47%
NEWS	0.77	1.92s	0.44 s	77%
FILES	0.66	2.07s	0.67 s	67%

In Table 2, the Correct Prediction Rate (C.P.R.) value is obtained by comparing the actual request of the user with the value predicted by the Personalized Sequential Behavior Model. Dividing the number of pages correctly predicted by the total number of pages requested we obtain the Correct Prediction Rate. The second column, O.A.G.T presents the Original Average Generation Time presented also in Table 1.

In order to obtain the Average Generation time with Prediction System (A.G.T.w.P.S.) we obtained the total generation time of the requests that were not predicted correctly. Dividing this amount by the total number of requests of that tab, we obtain the Average Generation Time with Prediction System. The Reduction Factor (R.F.) value is obtained by comparing the Average Generation Time with the Original Average Generation Time. Ideally the reduction factor should be equal to the Correct Prediction Rate, the differences arise from the fact that we are working with average values.

Ideally the training log should contain every user of the system. Nevertheless, in our case 15% of users in the testing log were not encountered in the training log or, were encountered, but did not generate a set of rules. Table 3 presents the same results as Table 2 but in this case, the requests of the users that do not have a model have also been considered.

Table 3. Results Considering All Requests of the Training Log.

TAB	C.P.R.	O.A.G.T.	A.G.T.w.P.S.	R.F.
ACADEMICS	0.69	0.66s	0.21 s	68%
RESEARCH	0.37	0.48s	0.28 s	41%
NEWS	0.73	1.92s	0.52 s	72%
FILES	0.53	2.07s	0.93 s	55%

Another possibility is pre-generating more than one page. Considering that 90% of the requests have a request interval of more than 5 seconds, it is possible to pre-generate two pages in these cases. As can be seen in

Table 4, this implies an increment in the Correct Prediction Rate.

Table 4. Results Considering a Two-Page Generation Policy and All Requests.

TAB	C.P.R.	O.A.G.T.	A.G.T.w.P.S.	R.F.
ACADEMICS	0.88	0.66s	0.082 s	87%
RESEARCH	0.61	0.48s	0.17 s	64%
NEWS	0.84	1.92s	0.28 s	85%
FILES	0.73	2.07s	0.52 s	74%

The fact that for 10% of the requests there is not enough time to pre-generate two pages, does not mean that the system can not be applied. For these requests the pre-generation of the pages will be partially completed after the request of the user.

#### 4.2.3 COSTS IN NYUHome

Considering the Sequential Behavior Model presented in the previous section, the increase in the number of pages generated for each tab due to incorrect predictions is: 17% for ACADEMICS, 46% for RESEARCH, 22% for NEWS and 26% for FILES. In absolute numbers, and for the testing log used, this means that 6,371 extra pages were generated due to incorrect prediction. Considering that 26,303 tabs were requested, without considering home requests, this means an increment of 24%. Considering the total load of the server, which includes home requests, the increment of the load is only 3%.

#### 4.2.4 SUMMARY OF IMPACT

Looking at the results in Table 3, we find that the Average Generation Time with Prediction System is reduced by at least 40% and the average reduction is 60%. We believe that the memory needed to store the Personalized Sequential Behavior Model is negligible: 2.5 rules per user, where each rule has basically one byte for the antecedent and one for the consequent. This is particularly our case, because personalized web sites already have a lot of information about each user[7]. The extra load introduced in the server increases the total load by 3%. Since the server is working far below its capacity [10], the extra load will not affect to user perceived latency.

### 4.3 IMPACT OF CONNECTION TIME MODEL

In this section we present the benefits and the costs of introducing a Connection Time Model in *NYUHome*.

#### 4.3.1 APPLICABILITY

Since 90% of the requests are for home potential benefits of a prediction mechanism which predicts session establishment are very significant. Basically, the reduction of the total average generation time will equal the reduction of the average generation of home. This also means that the extra load introduced by the prediction system can be very large. Because CPU utilization of the

*NYUHome* server never exceeds 30%, we consider that we can multiply by three the load without affecting user perceived latency.

#### 4.3.2 NYUHome CONNECTION TIME MODEL

The elements that define a Connection Time Model were introduced in section 3.2.3. In this section we identify which of those values produce a reduction of the average generation time of home while at the same time minimizing the extra load introduced.

##### 4.3.2.1 FREQUENT USER DEFINITION

The first step is how to define a frequent user. In this context, we define a frequent user as a user with a minimum number of visits in the training file. In the previous Section this definition was not as significant because only 10% of the requests were being handled.

We start with a Connection Time Model defined by a *SLOT\_LENGTH* of 1 day, *P* defined as in (10), *M=1* hour and *N=15%*. Keeping those values constant, we obtain a Connection Time Model defining frequent user as a user with at least 0, 5, 10 and 20 visits in the training log . Table 5 presents the characteristic of the Connection Time Models obtained.

Table 5. Characteristics of the Connection Models for different definitions of Frequent User.

Visits	Percentage of Users with Connection Model	Average # of Bursts per User	Processing Time
0	94%	4.5	5 min 30 sec
5	63%	5.7	4 min 10 sec
10	43%	6.3	3 min 40 sec
20	24%	6.7	3 min 10 sec
40	8%	8.2	3 min

This table presents, for the training file, the number of users for which bursts were found, the average number of bursts per user and the processing time necessary to obtain the Connection Time Model. As expected, when the number of visits required increases the number of users for which the condition holds decreases. Table 6 and Table 7 present the prediction accuracy results for these different models.

Table 6. Results of the Connection Models for different definitions of Frequent User.

Visits	Percentage of Pages Ready	Percentage of Pages Ready (Total)	A.G.T.	A.G.T. (Total)
0	23%	17%	0.78 sec	0.80 sec
5	23%	17%	0.76 sec	0.81 sec
10	23%	15%	0.75 sec	0.84 sec
20	22%	11%	0.74 sec	0.88 sec
40	20%	5%	0.71 sec.	0.94 sec

In Table 6, the Percentage of Pages Ready column expresses the percentage of home requests for whom the page is ready before the request. This value is obtained by dividing the number of pre-generate home tabs correctly predicted by the total number of home tabs requested by users that have a Connection Time Model.

The third column expresses the same value but comparing it with the total number of home request, considering both the requests made by users that do and do not have a Connection Time Model.

The Average Generation Time (A.G.T.) is obtained by adding the processing overhead, as expressed in (16), of the non predicted home requests of users with a Connection Time Model and dividing it by the total number of home requests made by users with a Connection Time Model. The fourth column (A.G.T. Total) expresses the same notion but considers both the home requests made by users with and without a Connection Time Model.

The Average Generation Time of a home tab using a pre-generation system is reduced at least by 30%, and in some cases up to 40%, when compared with the original 1.14 seconds of *NYUHome*.

Table 7. Results of the Connection Models for different definitions of Frequent User (II).

Visits	Correct Prediction	False Positives	Increment in Load
0	6%	94%	150%
5	8%	92%	130%
10	10%	90%	95%
20	12%	88%	49%
40	15%	85%	25%

In Table 7, Correct Prediction expresses the percentage of bursts for which the Connection Time Model accurately predicted that home was going to be requested by a user.

False Positives represents the percentage of home tabs that were pre-generated but not used, these pages are responsible for creating extra load.

Increment in Load gives the growth of the load compared to the original load of the server.

As expected, the Increment in Load is smaller when using a higher minimum for the definition of frequent user, because less home tabs are generated. The Correct Prediction rate increases when the Minimum Number of Visits increases. This is because the more we know about a user, the more accurate our predictions will be. Nevertheless it is not advisable to choose a high Minimum Number of Visits, because, although the average generation time remains more or less the same, the number of users that actually benefit from the prediction system is much smaller.

We have decided to choose 20 as the Minimum Number of Visits for a frequent user (on average, more than one visit during each day) because the number of users that benefit is 25% and because the Increase in Load is 49% with the Average Generation Time reduced by 36%. We have also decided to choose a *SLOT\_LENGTH* of 1 day because we consider that *NYUHome* users show a repeated behavior on a daily basis.

#### 4.3.2.2 CLUSTERING POLICY DEFINITION

Once the concept of frequent user has been defined we have to choose which clustering policy function  $P$  best

captures user connection patterns. Table 8 presents the characteristics of the Connection Time Models obtained with  $M=1$  hour and  $N=15\%$ , a minimum of 20 visits and  $P$  defined as in (10), (11) and (12), 1, 2 and 7 clusters respectively.

Table 8. Characteristics of the Connection Models for different Cluster definitions.

P	Percentage of Users with Connection Model	Average # of Bursts per User	Processing Time
(10)	24%	6.7	3 min 10 sec
(11)	29%	6.2 / 3.5	3 min 20 sec
(12)	29%	4.3/4.3/4.3/2.9/ 3.6 / 2.6 / 2.9	3 min 40 sec

The third column presents the average number of bursts of each user in each cluster. Table 9 and Table 10 present the results obtained using the testing file with the Connection Time Models presented in Table 8.

Table 9. Results of the Connection Models for different Cluster definitions.

P	Percentage of Pages Ready	Percentage of Pages Ready (Total)	A.G.T.	A.G.T. (Total)
(10)	22%	11%	0.74 sec	0.88 sec
(11)	20%	14%	0.73 sec	0.86 sec
(12)	22%	16%	0.72 sec	0.85 sec

Table 10. Results of the Connection Models for different Cluster definitions (II).

P	Correct Prediction	False Positives	Increment in Load
(10)	12%	88%	49%
(11)	21%	79%	39%
(12)	30%	70%	31%

As the results above illustrate,  $P$  defined as in (12) (one cluster for each day) best captures the Connection Time Models of the users. Although the Percentage of Pages Ready and the Average Generation Time remain more or less the same, the Correct Prediction is higher, which implies that the Increment in Load introduced in the system is smaller.

#### 4.3.2.3 DEFINITION OF $M$ AND $N$

All of the test undertaken up to this point have been conducted defining a connection burst with the parameters  $M=1$  hour,  $N=15\%$ . The definition of these parameters, especially  $M$ , depends on how frequently the caches of the pages change. Our model guarantees a page that is at most  $M$  seconds old. The value of  $M=1$  hour was selected because the characteristics of the channels of *NYUHome* require that the information is never more than 1 hour old.

Once  $M$  has been chosen,  $N$  has to be selected in relation with  $M$ . In general if  $M$  increases  $N$  should also increase. This makes it possible to capture connection bursts that are relevant. If the value of  $N$  is too high, the percentage of pages ready will decrease. Also, the growth of the load will be very small, because the number of pages that the system pre-generates is very small. If  $N$  is too low, the percentage of pages ready will increase but

the increment in load will be higher, because the number of pages pre-generated will also be higher.

Table 11 presents the characteristics of the Connection Time Model using a system with  $P$  defined as (12), a minimum number of 20 visits for each user in the training file,  $M=1\text{ hour}$  and the values 5%, 15% and 30% for  $N$ .

Table 11. Characteristics of the Connection Model for different definitions of  $N$ .

N	Percentage of Users with Connection Model	Average # of Bursts per User	Processing Time
5	30%	8.8/ 8.8 / 7.3 / 4.1/5.8 / 4.1 / 4.9	3 min 50 sec
15	29%	4.3/4.3 / 4.3 / 2.9 / 3.6 / 2.6 / 2.9	3 min 40 sec
30	29%	1.1/1.1 / 1.2 / 1.3 / 1.2 / 1.1 / 1.1	3 min 30 sec

The Connection Time Models obtained produce the results presented in Table 12 and 13.

Table 12. Results of the Connection Models for different definitions of  $N$ .

N	Percentage of Pages Ready	Percentage of Pages Ready (Total)	A.G.T.	A.G.T. (Total)
5	36%	26%	0.6 sec	0.76 sec
15	22%	16%	0.72 sec	0.85 sec
30	8%	5%	0.85 sec	0.94 sec

Table 13. Results of the Connection Models for different definitions of  $N$  (II).

N	Correct Prediction	False Positives	Increment in Load
5	30%	70%	47%
15	30%	70%	31%
30	26%	74%	12%

The data above show that a trade-off between the Increment in Load and the Percentage of Pages Ready is achieved with  $N=15\%$ . The results also show that the Correct Prediction does not depend on  $N$ , but rather on the clustering policy.

#### 4.3.2.4 OTHER VALUES OF $M$ AND $N$

We have also tested *NYUHome* with other  $M$  and  $N$  values. Table 14 presents the characteristics of the Connection Time Models defining  $P$  as in (12), a minimum number of visits of 20 and the values of  $M=30\text{ min.}/N=10\%$ ,  $M=1\text{ hour}/N=15\%$  and  $M=2\text{ hours}/N=20\%$  respectively.

Table 14. Characteristics of the Connection Models for different values of  $M$  and  $N$ .

M-N	Percentage of Users with Connection Model	Average # of Bursts per User	Processing Time
30min -10%	29%	5.3/5.3 / 5.2 / 4.6 / 4.1 / 2.9 / 3.2	4 min 50 sec
1h -15%	29%	4.3/4.3 / 4.3 / 2.9 / 3.6 / 2.6 / 2.9	3 min 40 sec
2h -20%	29%	4.2/4.2 / 4.1 / 2.7 / 3.4 / 2.7 / 2.7	4 min 30 sec

The Connection Time Models obtained produce the results presented in Tables 15 and 16. It is clear that the

percentage of users with a Connection Time Model mainly depends on the minimum number of visits required rather than on  $M$  and  $N$ .

With bigger values of  $M$ , the Percentage of Pages Ready will generally increase and the Increment in Load will decrease, but the information provided by the pre-generated pages will be older.

Table 15. Results of the Connection Models for different values of  $M$  and  $N$ .

M-M	Percentage of Pages Ready	Percentage of Pages Ready (Total)	A.G.T.	A.G.T. (Total)
30min -10%	15%	11%	0.79 sec	0.89 sec
1h -15%	22%	16%	0.72 sec	0.85 sec
2h -20%	33%	23%	0.62 sec	0.78 sec

Table 16. Results of the Connection Models for different values of  $M$  and  $N$ .

M -N	Correct Prediction	False Positives	Increment in Load
30min- 10%	17%	83%	36%
1h-15%	30%	70%	31%
2 h-20%	45%	55%	29%

### 4.3.3 SUMMARY OF IMPACT

The three basic elements necessary to deciding the characteristics of the Connection Time Model are the Increment in Load, the Percentage of Pages Ready and the staleness of the information provided ( $M$ ). Taking into account that we have considered a value of  $M=1\text{ hour}$  and a *SLOT\_LENGTH* of 1 day, a good Connection Time Model for *NYUHome* is given by  $P$  defined as (12) (each day of the week one cluster), a minimum number of visits per user of 20 and  $N=15\%$ . With this Connection Time Model the Percentage of Pages Ready is 22% and the Increment in Load is only 31%. This achieves an Average Generation Time of 0.72 sec., compared with the original 1.14 sec. generation time of the home tab.

As with the Personalized Sequential Behavior Model, the memory needed to store the models is negligible when compared to the amount of per-user information already present in such systems[7].

### 4.4 GLOBAL RESULTS

The combination of a Personalized Sequential Behavior Model and a Connection Time Model provides a complete pre-generation system.

The Total Average generation time using these two systems is reduced from 1.11 (Table 1) to 0.68 (38% reduction) for the users with a Connection Time Model and a Personalized Sequential Behavior Model:

$$\begin{aligned}
 &0.72s \cdot 0.91(\text{HOME}) + 0.13s \cdot 0.051(\text{ACADEMICS}) \\
 &+0.25 \cdot 0.014(\text{RESEARCH}) + 0.44 \cdot 0.023(\text{NEWS}) \\
 &+0.67 \cdot 0.011(\text{FILES}) = 0.68s
 \end{aligned}
 \tag{17}$$

The total load increment is 34%, 31% because of the Connection Time Model and 3% of the Personalized Sequential Behavior Model. We consider that because the

server utilization never exceeds 30%[10], *NYUHome* can handle this increment without affecting user latency.

For servers in which home requests are not as important as in *NYUHome*, the reduction of the generation time will be higher because the Personalized Sequential Behavior Model has better prediction rates than the Connection Time Model.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach to the problem of reducing user perceived latency in a dynamic and personalized web server. The system pre-generates the pages about to be requested using models that capture user behavior. This way, when the user actually requests the page, latency is determined only by the time needed to send the page from the client to the server. This makes it possible for dynamic servers to have latency similar to that perceived by users of static servers. We have proposed two algorithms for each of the problems that needed to be solved: (1) how to predict the next request of a user once the session has already started, and (2) how to predict the first request of a session.

We have applied these ideas to a dynamic and personalized web server. We have shown how to select the parameters that define the designed algorithms. The results were very satisfactory, at least a 38% reduction of the average generation time was achieved while increasing the load of the server by 34%.

Another interesting observation from this work is that relatively simple prediction models (with no generalization capabilities) can prove surprisingly effective in capturing user behavior and be the basis for the development of intelligent web applications.

We are continuing our work by designing a more formal approach to generating Connection Time Models. Currently we are focussing on solving the problem with Support Vector Machines. The results obtained to date are very promising.

## 6. ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Department of Research grant EX2001-46775835.

## REFERENCES

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, USA (1993)
2. Challenger, J., Iyengar, A., Dantzig, P.: A scalable system for consistently caching dynamic web data, Proceedings of Infocom'99, Mar. 1999
3. Doyle, R., Chase, J., Gadde, S., Vanhat, A.: The trickle-down effect: Web caching and server request distribution. Proc. of the 6th International Workshop on Web Caching and Content Distribution (WCW'01), June 2001.
4. Frias-Martinez, E., Karamcheti, V.: A Prediction Model for User Access Sequences. In: Fourth Workshop on Knowledge Discovery in the Web, WEBKDD'2002, Edmonton, Canada, (2002) 53-62
5. Joshi, A., Joshi, K., Krishnapuram, R.: On mining Web Access Logs. In: Proceedings of the ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, (2000) 63-69
6. Krishnapuram, R., Joshi, A., Nasraoui, O., Yi, L.: Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. In IEEE Transactions on Fuzzy Systems, Vol. 9 (4), (2001) 595-608
7. Manber, U., Patel, A., Robinson, J.: Experience with Personalization on Yahoo!. In: Communications of the ACM, Vol. 43, No. 8 (2000) 35-39
8. Mobasher, B., Cooley, R.: Automatic Personalization Based on Web Usage Mining. In Communications of the ACM, Vol 43:8 (2000) 142-151
9. Nanopoulos, A., Katsaros, D., Manolopoulos, Y.: Effective Prediction of Web-user Accesses: A Data Mining Approach. In: Proceeding of the WEBKDD 2001 Workshop, San Francisco, CA (2001)
10. Shi, W., Wright, R., Collins, E., Karamcheti, V.: Workload Characterization of a Personalized Web Site and Its Implications for Dynamic Content Caching. In: Proceedings of the 7th International Conference on Web Content Caching and Distribution (WCW'02), Boulder, CO (2002)
11. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. In SIGKDD Explorations (2000)
12. Su, Z., Yang, Q., Zhang, H.: A prediction system for multimedia pre-fetching on the Internet. In: Proceedings of the ACM Multimedia Conference 2000, ACM (2000)
13. Yang, Q., Tian-Yi, I., Zhang, H.: Mining High-Quality Cases for Hypertext Prediction and Prefetching. In ICCBR 2002, LNAI 2080, Springer-Verlag (2001) 744-755
14. Yang, Q., Zhang, H., Li, I., Lu, Y.: Mining Web Logs to Improve Web Caching and Prefetching. In: N. Zhong et al (Eds.): WI 2001, LNAI 2198, Springer-Verlag Berlin Heidelberg (2001) 483-492
15. Zona Research, <http://www.zonaresearch.com/infopress/99-jun30.htm>