Available online at www.sciencedirect.com
SCIENCE DIRECT
Performance Evaluation 63 (2006) 988–1004
PERFORMANCE EVALUATION
An International Journal
www.elsevier.com/locate/peva

Web prefetching performance metrics: A survey

Josep Domènech*, José A. Gil, Julio Sahuquillo, Ana Pont

Department of Computer Engineering (DISCA), Universitat Politècnica de València, València, Spain

Received 30 March 2005; received in revised form 23 September 2005
Available online 5 December 2005

Abstract

Web prefetching techniques have been pointed out to be especially important to reduce perceived web latencies and, consequently, an important amount of work can be found in the open literature. But, in general, it is not possible to do a fair comparison among the proposed prefetching techniques due to three main reasons: (i) the underlying baseline system where prefetching is applied differs widely among the studies; (ii) the workload used in the presented experiments is not the same; (iii) different performance key metrics are used to evaluate their benefits.

This paper focuses on the third reason. Our main concern is to identify which are the meaningful indexes when studying the performance of different prefetching techniques. For this purpose, we propose a taxonomy based on three categories, which permits us to identify analogies and differences among the indexes commonly used. In order to check, in a more formal way, the relation between them, we run experiments and estimate statistically the correlation among a representative subset of those metrics. The statistical results help us to suggest which indexes should be selected when performing evaluation studies depending on the different elements in the considered web architecture.

The choice of the appropriate key metric is of paramount importance for a correct and representative study. As our experimental results show, depending on the metric used to check the system performance, results cannot only widely vary but also reach opposite conclusions.

Keywords: Web prefetching; Performance evaluation; Metrics taxonomy

1. Introduction

The international and global nature of the Internet makes it arduous to increase the system performance working in the network hardware and its interconnection elements because of the existing gap of time between technological advances in infrastructure and its practical use. As a consequence, research efforts have been concentrated on the web architecture and organization using and exporting techniques already learned and widely used in computer architecture to improve the performance.

Most of those techniques take advantage of the locality properties inherent to the web object accesses. In the Web, the locality has three different characteristics: temporal, spatial and geographical, which permits us to implement

* Corresponding author.
E-mail addresses: jodode@doctor.upv.es (J. Domènech), jagil@disca.upv.es (J.A. Gil), jsahuqui@disca.upv.es (J. Sahuquillo), apont@disca.upv.es (A. Pont).

0166-5316/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2005.11.001

efficiently caching, prefetching and replication techniques in order to increase performance. As a result, many efforts (commercial and research) applying those techniques in the web architecture have been carried out to improve performance. In this paper we focus on prefetching techniques, although some of the conclusions presented in this paper can also be adapted to include any web technique aimed to reduce the final user perceived latency.

Many research studies concentrate on the proposals of new prefetching techniques. Performance comparison studies among them cannot be fairly done because the proposed approaches are applied and tested in different baseline systems using also different workloads and conditions. In addition, the studies present different performance key metrics to evaluate their benefits.

In order to do fair performance comparison studies we need to tackle the mentioned drawbacks. To deal with the first one, we proposed in a previous work [12] a general experimental framework which permits the implementation of prefetching techniques in a flexible and easy way, under the same platform and real workloads; therefore, the same experimental conditions are considered.

In this paper, we address our work to tackle the second drawback. To this end, we analyze a large subset of key metrics and propose a taxonomy based on three main categories, which permits us to identify analogies and differences among the metrics commonly used and check experimentally their relation. As a consequence, we also introduce new byte based indexes as complementary metrics. This paper extends the work presented in [13] in several ways. We identify new analytical relations among the indexes, and a new metric is proposed to establish the analytical relation. We also include more experiments considering another prediction algorithm. Finally, references have been accordingly updated and discussed.

The remainder of this paper is organized as follows: Section 2 describes a generic web prefetching system in order to set the basic glossary of terms that will be used in the remaining sections; Section 3 presents the proposed taxonomy for prefetching performance metrics; Section 4 describes the experimental environment used to run the experiments; Section 5 discusses the relation between indexes from two different approaches, analytical and experimental, and finally, Section 6 presents some concluding remarks.

2. Generic prefetch architecture and basic glossary

We assume a generic web architecture composed of three main elements: clients, servers and proxies. Note that proxies act both as a client for the server and as a server for the client.

It is important to remark on the difference between the user and the client. The user is the person in front of a computer (or a similar device) demanding information, whereas the client is the software (i.e., the browser) with which the user interacts, that requests the demanded information to the appropriate server.

The main aim of prefetching techniques is to reduce the average latency perceived by the user. Several prefetching related techniques have been proposed focusing on different ideas to exploit the spatial locality of web objects using prefetching; for instance: some research studies propose clients to download objects prior to being requested by the user [17,26,16,21]; other studies propose to preprocess dynamic content [30]; some others concentrate on how to make pre-connections to the server [6]; and so on. All prefetching related techniques start predicting or trying to guess the next objects to which the client will access. This part of the prefetch is usually referred as the prediction engine. Then, the prediction results are submitted to the prefetching engine, which decides whether to prefetch or not such results, depending on other parameters; e.g., available bandwidth or server load. In this way, prefetched objects are a subset of the predicted objects. Notice that both the prediction engine and the prefetching engine can be found in the same element (client, proxy or server). We define below some basic variables that will be used in Section 3:

- *Predictions*: amount of objects predicted by the prediction engine.
- *Prefetchs*: amount of objects prefetched by the prefetching engine.
- *GoodPredictions*: amount of objects predicted that are subsequently demanded by the user.
- *BadPredictions*: those predictions that do not result in good predictions.
- *PrefetchHits*: amount of prefetched objects that are subsequently demanded by the user.
- *ObjectsNotUsed*: amount of prefetched objects never demanded by the user.
- *UserRequests*: amount of objects the user demands.

Fig. 1 shows graphically the relations between the variables defined above. A represents an object requested by the user but neither predicted nor prefetched. B represents a *GoodPrediction* that has not been prefetched, while C

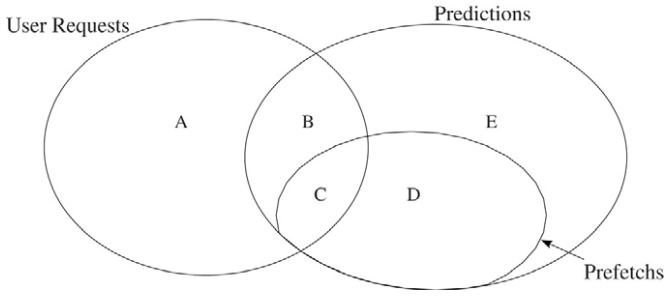

Fig. 1. Sets representing the relation between *UserRequests*, *Predictions* and *Prefetchs*.

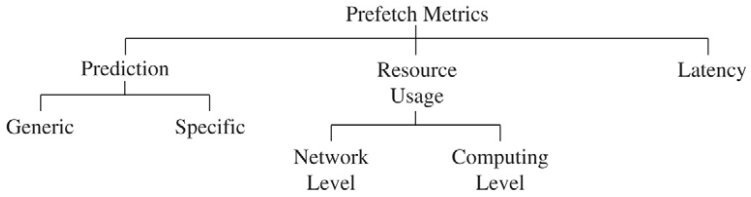

Fig. 2. Prefetching metrics taxonomy.

represents a *PrefetchHit*, i.e. a *GoodPrediction* that has been prefetched. D is an *ObjectNotUsed*, which is also a *BadPrediction*. Finally, E represents a *BadPrediction*.

Analogously, we can define byte related variables ($Predictions_B$, $Prefetchs_B$, and so on) by replacing the objects with the corresponding size in bytes in their definition.

3. Web performance indexes taxonomy

This section surveys the web performance indexes appearing in the open literature focusing on prefetch aspects. For the better understanding of the meaning of those indexes, we classify them into three main categories (see Fig. 2), attending to the system feature they evaluate:

- Category 1: prediction related indexes.
- Category 2: resource usage indexes.
- Category 3: end-to-end perceived latencies indexes.

The first category is the main one when comparing prediction algorithms performance and includes those indexes which quantify both the efficiency and the efficacy of the algorithm (e.g., precision). The second category quantifies the additional cost that prefetching incurs (e.g., traffic increase or processor time). This cost may become really high. For that reason, it must be taken into account when comparing prefetching techniques. Therefore, those indexes can be seen as complementary measures. Finally, the third category summarizes the performance achieved by the system from the user's point of view. Notice that prefetching techniques must take care of the cost increase because they can negatively impact on the overall system performance (traffic increase, user perceived latencies). Therefore, the three categories are closely related since, in order to achieve a good overall performance (Category 3), prefetching systems must trade off the aggressiveness of the algorithm (Category 1) and the cost increase due to prefetching (Category 2).

Different definitions for the same index can be found in the literature (e.g., precision) and this fact increases the heterogeneity of the research effort. In order to make this survey more readable, we only include the definition that we consider more precise and appropriate for evaluation purposes. In the cases where several names match the same definition, we select the most appropriate index name from our point of view. The goal of this section is not only to help the understanding of the indexes but also to discuss their usefulness, distinguishing those used for comparison purposes in any prefetching system from those applicable to a particular prefetching technique (i.e. specific). Specific indexes are only found in the Category 1 (Prediction), as shown in Fig. 2.

3.1. Prediction related indexes

This group includes those indexes aimed at quantifying the performance that the prediction algorithm provides. Prediction performance can be measured at different moments or in different elements of the architecture, for instance when (where) the algorithm makes the prediction and when (where) prefetching is applied in the real system. Thus, each index in this category has a *dual* index; e.g., we can refer to the precision of the algorithm and to the precision of the prefetch. Notice that those indexes measured when the prediction list is given do not take into account system latencies because the prediction algorithm works independently of the underlying network and the user's restrictions.

3.1.1. Generic prediction indexes

Precision (Pc). *Precision* measures the ratio of good predictions to the number of predictions [3,1,29,7,10] (see Eq. (1)). Precision, defined in this way, just evaluates the algorithm without considering physical system restrictions; e.g., cache, network or time restrictions; therefore, it can be seen as a theoretical index.

Some research studies refer to this index as *accuracy* [16,8,32,23,25,27,15] or *hit ratio* [5], while others use a probabilistic notation; e.g., $Pr(hit \mid match)$ in some models based on Markov chains like [28].

There are other research works that measure the impact on performance of the precision [16,23]. In these cases, the number of prefetched objects and prefetch hits are used instead of the number of predictions and good predictions respectively (see Eq. (2)).

$$Pc = \frac{GoodPredictions}{Predictions} \tag{1}$$

$$Pc = \frac{PrefetchHits}{Prefetchs} \tag{2}$$

Ibrahim and Xu [19] evaluate the prefetching performance through the *waste ratio*, which is defined as the percentage of undesired documents that are prefetched. As one can observe, this index is the complementary of the *precision*.

Recall (Rc). *Recall* measures the percentage of requested objects that were previously prefetched [1,7,29]. The *recall* quantifies the weight of the predicted (see Eq. (3)) or prefetched objects (see Eq. (4)) over the amount of objects requested by the user. Notice that this index only deals with the number of good predictions made but not with the total amount of predictions.

Some research works refer to this metric as *usefulness* [25,27,5], *hit ratio* [24,19] or *accuracy* [11,10]. *Predictability* has been employed in [30] to refer to the upper limit of the *recall*.

$$Rc = \frac{GoodPredictions}{UserRequests} \tag{3}$$

$$Rc = \frac{PrefetchHits}{UserRequests} \tag{4}$$

Applicability. Bonino et al. [3,4] define *applicability* as the ratio of the number of predictions to the number of requests. In [3] the authors incorrectly state that this index is a synonym of *recall*. Notice that this index can be obtained from the previous ones (i.e., dividing the *recall* by the *precision*); therefore no additional information is given. This is the main reason why no other work uses this index.

$$Applicability = \frac{Predictions}{UserRequests} \tag{5}$$

Precision alone or together with *recall* has been the most widely used index to evaluate the goodness of prediction algorithms. The time taken between the client request making and the response receiving consists of four main components; i.e., connection establishment time, request transference time, request processing time, and response transference time. Both *precision* and *recall* are closely related to the three first components. Therefore, the authors consider that a complete comparison study about prediction algorithms should also include byte related indexes to

quantify the last component. In this sense, analogously to the web proxy caching indexes (e.g., the *byte hit ratio*) [9], we propose the use of *byte precision* and *byte recall* as indexes to estimate the impact of prefetching on the time that the user wastes when waiting for the bytes of the requested objects.

Byte precision (Pc_B). *Byte precision* measures the percentage of predicted (or prefetched) bytes that are subsequently requested. It can be calculated by replacing the number of predicted objects with their size in bytes in Eq. (1).

Remark that, like *precision* does, *byte precision* quantifies how good the predictions are, but measured in bytes instead of in objects. An earlier approach to this index is the *Miss rate ratio* [2], described below. Bouras et al. [5] also mentioned that different results could be obtained when using the number of bytes instead of the number of objects.

$$Pc_B = \frac{GoodPredictions_B}{Predictions_B} \tag{6}$$

Byte recall (Rc_B). *Byte recall* measures the percentage of demanded bytes that were previously predicted (or prefetched). As mentioned above, this index quantifies how many accurate predictions are made, measured in transferred bytes.

This index becomes more helpful than the previously mentioned *recall*, when the transmission time is an important component of the overall user perceived latency.

$$Rc_B = \frac{GoodPredictions_B}{UserRequests_B} \tag{7}$$

3.1.2. Specific prediction indexes

Request savings. *Request savings* measures the number of times that a user request hits in the browser cache or the requested object is being prefetched, in percentage of the total number of user requests [17]. Furthermore, the request savings can be broken down into three groups depending on if they were previously prefetched, have been partially prefetched or cached as normal objects.

Notice that when prefetching is user-initiated the number of requests increases; therefore, this index makes sense when prefetching is proxy or server initiated. Fan et al. use this index in a prefetch system where the proxy pushes objects to the client. Nevertheless, this index could be also used when prefetching pushes objects from the server to the proxy or from the server to the client (with no intermediate proxies).

Miss rate ratio. Bestavros defines the *miss rate ratio* [2] as the ratio between the byte miss rate when the prefetch is employed to the byte miss rate when the prefetch is not employed, where the byte miss rate for a given client is the ratio of bytes not found in the client's cache to the total number of bytes accessed by that client.

As one can see, this index quantifies in which percentage the miss rate in the client cache drops due to the prefetching system. This is a specific index since it is only applicable to those systems storing the prefetched objects in the client's cache.

Probability of making a prediction. Pitkow and Pirolli [28] quantify the probability that the last accesses match the pattern prediction; in such a case the prefetch system computes the prediction outcomes. This index can be applied, for example, to those systems based in Markov models, but cannot be applied to a large subset of prefetching systems; e.g., the top-10 approaches [24]; so, it is classified as specific.

3.2. Resource usage

The benefits of prefetching are achieved at the expense of using additional resources. This overhead, as mentioned above, must be quantified because it can negatively impact on performance.

Although some prediction algorithms may require huge memory or processor time (e.g., high order Markov models), it is not the current general trend, where the main prefetching bottleneck is the network traffic. Therefore, we divide indexes in this category into two subgroups: network level and computing level.

3.2.1. Network level

Traffic increase (ΔTr_B). *Traffic increase* quantifies the increase of traffic (in bytes) due to unsuccessfully prefetched documents [24] (see Eq. (8)). It is also called *wasted bandwidth* [17], *extra bytes* [29], *network traffic* [27,5], and *bandwidth ratio* [2].

When using the prefetch, network traffic usually increases due to two side effects of the prefetch: objects not used and overhead. Objects not used waste network bandwidth because these objects are never requested by the user. On the other hand, the network traffic increases due to the prefetch related information interchange, called *Network overhead* by [16].

Several research studies fail to take into account that overhead [25,24,20]; therefore, their results cannot estimate prefetching costs accurately.

$$\Delta Tr_B = \frac{ObjectsNotUsed_B + NetworkOverhead_B + UserRequests_B}{UserRequests_B} \tag{8}$$

Extra control data per byte. Extra control data per byte quantifies the extra bytes transferred that are related to the prefetch information, averaged per each byte requested by the user. It is the *Network overhead* referred in [16], but quantified per requested bytes, and will be used below when relating *Traffic increase* to the prediction indexes.

$$ExtraControlData_B = \frac{NetworkOverhead_B}{UserRequests_B} \tag{9}$$

Object traffic increase (ΔTr_{ob}). *Object traffic increase* quantifies which percentage of the number of documents that clients get when using prefetching increases. Nanopoulos et al. [25] refer to this index as *network traffic* and Rabinovich [29] as *extra requests*.

As Eq. (10) shows, this index estimates the ratio of the amount of prefetched objects never used with respect to the total user's requests. It is analogous to the *traffic increase*, but it measures the overhead in number of objects.

$$\Delta Tr_{ob} = \frac{ObjectsNotUsed + UserRequests}{UserRequests} \tag{10}$$

3.2.2. Computing level

Server load ratio. Server load ratio is defined as the ratio between the number of requests for service when speculation is employed to the number of requests for service when speculation is not employed [2].

Space and time overhead. In addition to the server load, some research works discuss how the overhead impacts on performance. For instance, Duchamp [16] discusses the memory and processor time that the prefetch would need.

Prediction time. This index quantifies the time that the predictor takes to make a prediction. It is used in [15] to compare different predicting algorithms.

3.3. Latency related indexes

Indexes belonging to this category include those aimed at quantifying the end-to-end latencies; e.g., user or proxy related latencies. The main drawback of these indexes is that they include several time components, some of them difficult to quantify. Many times researchers do not detail which components they measure, although they use a typical index name; i.e., latency. Several names have been used instead; for instance, *access time* [26], *service time* [2] and *responsiveness* [20,31]. This situation is not the best for the research community, due to the fact that the different proposals cannot be fairly compared among them.

Through the different research works, latencies are measured both per page and per object. The *Latency per page* (L_p) is calculated by comparing the time between the browser's initiation of an HTML page GET and the browser's reception of the last byte of the last embedded image or object for that page [16]. Analogously, the *Latency per object*

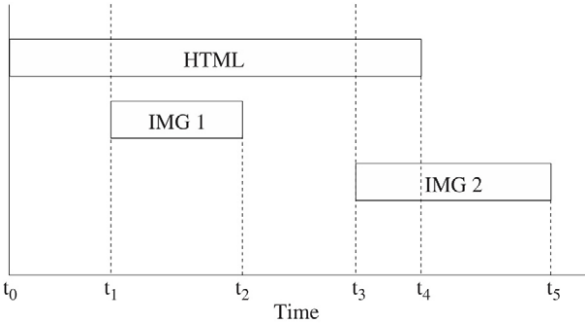

Fig. 3. Example where latency metrics behave differently when the prefetch hits on one of the images.

(L_{ob}) can be defined as the elapsed time since the browser requests an object until it receives the last byte of that object. In order to illustrate the benefits of prefetching, researchers calculate the ratio of the latency that prefetching achieves (either per page [17,16,23] or per object [22]) to the latency with no prefetching.

Unfortunately, some proposals that use *latency* when measuring performance do not specify which latency they are referring to; e.g., [20,2,5]. This fact can be misleading because both indexes do not perform in the same way. In order to illustrate this point we present the following example: a user requests an HTML object embedding two images (IMG1 and IMG2). As Fig. 3 shows, the transference of the HTML file starts at t_0 and ends at t_5. At times t_1 and t_3 the browser reads and processes the IMG tags of the embedded images. Then it starts the transferences, which end at t_2 and t_5 respectively. In this case, the cumulative L_{ob} is the sum of the time taken by the three transferences ($L_{ob} = t_4 - t_0 + t_2 - t_1 + t_5 - t_3$) where the *Latency per page* (L_p) is $t_5 - t_0$. If it is assumed that IMG1 was previously prefetched, no waiting time for such an object will be plotted; i.e. t_1 will match t_2, in such a way that it will reduce L_{ob} but not L_p, which will remain the same value.

To observe this feature in a real environment for a given client it is necessary that the object retrieving times are independent. Nevertheless, although times are not independent, both indexes have different values, as experimental results show.

Furthermore, the *Latency per object* measured when an object is downloaded by the web browser has two main components: (i) the queuing time, since the browser has a limited number of connections, and (ii) the request completion time, including the time of connection establishment (if needed) and the object transference. The first component is often ignored [5] and other research studies [2,21] do not specify whether the measured latency includes the queuing time. The first component should not be ignored because prefetch hits do not compete for a free connection so the queuing time of the remaining objects decreases and, consequently, their latency.

Note that all the stated relations between both latency related indexes are valid for any web latency reduction technique and not only for prefetching. In this sense, the definitions of these indexes can be adapted to include any web technique aimed at reducing the final user's perceived latency. The latency per page (or per object) ratio can be defined, in a wider way, as the ratio between the latency per page (or per object) when a latency reduction technique is used and the latency per page (or per object) when that technique is not used.

3.4. Summary: synonyms and experimental index category used

Through the proposed taxonomy, we have discussed the large variety of index names (synonyms) used to refer to the same metric. This heterogeneity can be observed through web performance studies appearing in the literature, a fact that adds extra difficulty to obtain a general view of the subject. Table 1 offers a scheme of the current situation. In addition, we also have found that the same index name has been used when measuring different variables (e.g., *accuracy* appears both for *precision* and for *recall*). As one can observe, the widest heterogeneity is present in the first category due to the wide range of prediction algorithms appearing in the literature and due to its importance in the prefetching systems, which are the main focus of this work.

Table 2 classifies a representative subset of research works that can be found in the open literature and shows the categories of indexes that are used in such works. As discussed above, a complete research work should include indexes belonging to the three categories. However, we only have observed this fact in 16% of the explored works, just

Table 1
Relation between the selected index name in this paper and those appearing in the literature

Category	Selected name	Literature	
		Name	References
1. Prediction	*Precision*	*Precision*	[3,1,29,8,10,4]
		Accuracy	[16,7,32,23,25,27,15]
		Pr(hit | match)	[28]
		Hit ratio	[5]
		Waste ratio	[19]
	Recall	*Recall*	[1,29,7]
		Usefulness	[25,27,5]
		Hit ratio	[24,19]
		Predictability	[30]
		Accuracy	[10,11]
	Applicability	*Applicability*	[3,4]
2. Resource usage	*Traffic increase*	*Traffic increase*	[26,16,24]
		Wasted bandwidth	[17]
		Bandwidth ratio	[2]
		Extra bytes	[29]
		Data transfer	[20]
		Network traffic	[27,5]
	Object traffic increase	*Network traffic*	[25]
		Extra requests	[29]
	Prediction time	*Prediction time*	[15]
3. Latency	*Latency per page*	*Latency*	[17,16,23]
		Responsiveness	[20,31]
	Latency per object	*Latency*	[21,22,5]
		Access time	[26]
		Service time ratio	[2]

Table 2
Relation between the research studies and the indexes used, grouped by category

References	Category			%
	1. Prediction	2. Resource	3. Latency	
[17,16,2,5]	X	X	X	16
[26,20]		X	X	8
[23]	X		X	4
[21,22,31]			X	12
[25,27,24,15]	X	X		16
[30,3,1,8,7,32,11,28,10,19,4]	X			44
TOTAL	80%	40%	40%	

four of the twenty-five works (row 1). Notice that the first column shows that 80% of the research works have at least one of the indexes belonging to the prediction category. Moreover, 44% of the studies only measure prediction metrics to evaluate the system performance. The second column states that 60% of the papers do not measure resource usage metrics. Finally, the third column shows that 60% of the research works do not quantify the end perceived latencies.

4. Experimental environment

In this section we first describe the experimental framework used for the performance evaluation study. Secondly, we briefly present the prefetching algorithms considered in the evaluation and finally, we present the main characteristics of the workload employed.

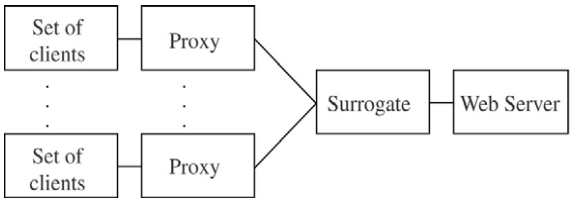

Fig. 4. Architecture of the simulation environment.

4.1. System architecture

In [12] we proposed and described in detail an experimental framework for testing web prefetching techniques. In this section we summarize the main features of such environment and how it has been extended to carry out the experiments presented in this paper.

The architecture is composed by three main parts (as shown in Fig. 4): the back end (server and surrogate), the front end (client) and optionally the proxy, which is not used in this paper. The framework implementation combines both real and simulated parts in order to provide flexibility and accuracy.

The back end part includes the web server and the surrogate server. The framework emulates a real surrogate, which is used to access to a real web server. Although the main goal of surrogates is to act as a cache for the most popular server responses, we use it as a predictor engine.

The front end, or client part, represents the users' behavior browsing the web with a prefetching enabled browser (like Mozilla [18]). To model the set of users that access concurrently to a given server, the simulator can be fed by using either real or synthetically generated traces. The simulator collects basic information for each request performed to the web server, and then writes it to a log file. By analyzing this log at post-simulation time, all performance metrics can be calculated.

The extension of the simulator presented in this paper concentrates on the client part in order to provide the additional information needed to evaluate the usefulness of the proposed indexes. In this context, the extension includes three main features:

- The concept of *page* is introduced. In this way, statistics per page can also be provided, i.e., the simulator estimates both the perceived latency for downloading each individual object and the perceived latency for downloading the whole page.
- New indexes are calculated. The proposed indexes are now implemented and calculated to check their usefulness.
- Users have a limited session time. After this time, client cache is emptied representing the expiration time of the cached objects.

4.2. Prefetching algorithms

In order to check the behavior of the indexes, the experiments were run using two different traces and two different prefetching algorithms: the *Dependency Graph* (DG) based on the algorithm proposed by Padmanabhan and Mogul [26] and the *Prediction by Partial Match* (PPM) based on the algorithm proposed by Palpanas and Mendelzon [27].

The DG prediction algorithm constructs a dependency graph that depicts the pattern of accesses to the server files. The graph has a node for every file that has ever been accessed. There is an arc from node A to B if and only if at some point in time a client accessed B within w accesses after A, where w is the *lookahead window* size. The weight on the arc is the ratio of the number of accesses to B within a window after A to the number of accesses to A itself. The prefetching aggressiveness is controlled by a *threshold* parameter applied to the arcs weight.

The PPM prediction algorithm uses Markov models of m orders to store previous contexts. Predictions are obtained from comparing the current context with each Markov model. Like DG, the prefetching algorithm controls the aggressiveness through a user defined *threshold*.

4.3. Workload description

The behavior pattern of the users was taken from logs collected during eight days, between May 5th and May 12th 2003, by a Squid proxy of the Polytechnic University of Valencia. From this log, accesses to two of the most popular

Table 3
Trace characteristics

Trace	A	B
Training accesses	251,271	148,213
Experiment accesses	65,866	37,655
HTMLs accesses	2269	1717
Users	300	132
Bytes transferred (kB)	188,600	68,428

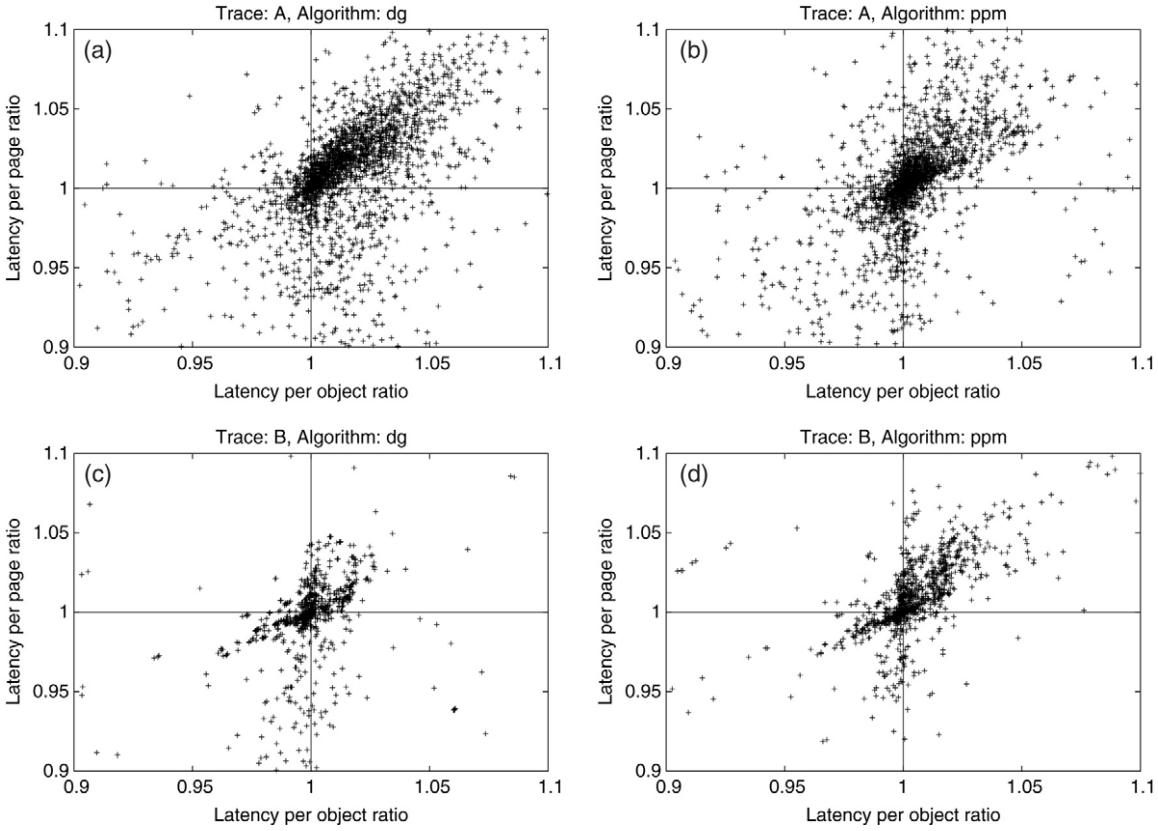

Fig. 5. *Latency per object* ratio as a function of *latency per page* ratio.

servers in this environment were taken and reproduced in the simulation, using the first week to train the prefetching algorithm and the following day to obtain the performance indexes. The first trace (A) contains accesses to a news web server, whereas the second one (B) includes the accesses to a student information web server. The main characteristics of the trace can be found in Table 3.

5. Relation between indexes

This section has three main goals. The first one is to illustrate the usefulness of the indexes introduced in this paper (i.e., *byte precision*, *byte recall, and extra control data per byte*) as well as the way in which they differ from the analogous classical ones. The second one is to study and show how indexes are related among them. Finally, the third goal is to identify the most meaningful indexes when evaluating the overall system performance. We use both empirical and analytical approaches to show how indexes are interrelated. When analytical relations were found, results for the empirical approach are not showed.

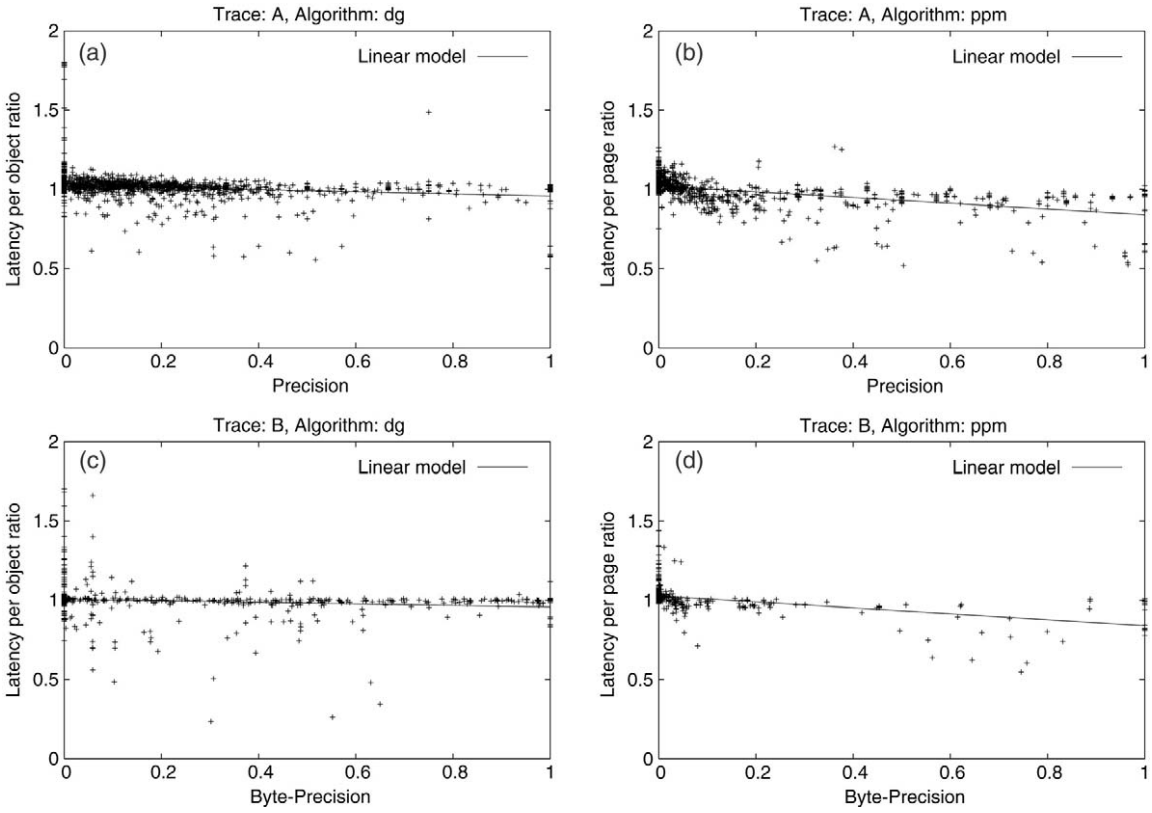

Fig. 6. Relation between precision and latency related indexes.

5.1. Empirical tests

The relations among indexes are checked by comparing them by pairs that could be potentially related through different prefetching scenarios. Pairs were selected from the most widely used indexes as discussed in Section 3: *precision*, *byte precision*, *recall*, *byte recall*, *latency per page* and *latency per object*.

In order to detect the possible relations, a four-graphic figure is plotted for each pair of indexes as a result of combining the two described algorithms using the two mentioned traces. Then, in those graphics where some sign of linear relation appears, we quantify it through the linear correlation coefficient. In order to observe how the indexes behave in a wide range of situations, the experiments were run ranging the *threshold* value of the prefetching algorithms from 0.1 to 0.9 (with a 0.1 increment). Points in the graphics refer to the performance indexes measured for every client in each experiment. We remark that the measured values of the prediction indexes refer to the prefetched objects instead of the predicted objects since they are closer to the system performance.

Fig. 5 plots the *Latency per object* ratio to the *Latency per page* ratio, both referring to different alternatives about how latency can be measured (as discussed in Section 3). The points refer to the performance index measured for every client in each experiment.

Notice that, depending on the way the latency is measured, it is possible to reach not only different but also opposite conclusions. Suppose that we take a point in the upper left side quarter and we consider the *Latency per object*; in such a case, the prefetching system outperforms the non-prefetching system. However, if we considered the *Latency per page*, we would conclude the opposite. The correlation coefficient between both latency ratios corresponding to the points plotted in Fig. 5 ranges from 0.55 to 0.77, i.e., the indexes present a certain linear correlation but it is far from being strong; so that *Latency per object* and *Latency per page* ratios cannot be directly comparable.

As a consequence, we suggest that studies should differentiate the use of both latency ratios because they are aimed at exploring the performance from different points of view. The *Latency per page* ratio evaluates the system performance from the user's point of view (since it measures the latency as perceived by the user) while the *latency*

Table 4
Linear correlation coefficient between prediction and latency indexes

Latency	Trace \ Alg	Precision				Recall			
		Pc		Pc_B		Rc		Rc_B	
		DG	PPM	DG	PPM	DG	PPM	DG	PPM
L_p	A	−0.3333	−0.5453	−0.3684	−0.6107	−0.4088	−0.5089	−0.5780	−0.7227
	B	−0.3495	−0.3436	−0.1533	−0.4929	−0.5037	−0.4149	−0.5309	−0.4579
L_{ob}	A	−0.2068	−0.2667	−0.2663	−0.2114	−0.2311	−0.6023	−0.2318	−0.3049
	B	−0.1292	−0.1564	−0.3855	−0.1944	−0.6685	−0.1083	−0.3780	−0.1024

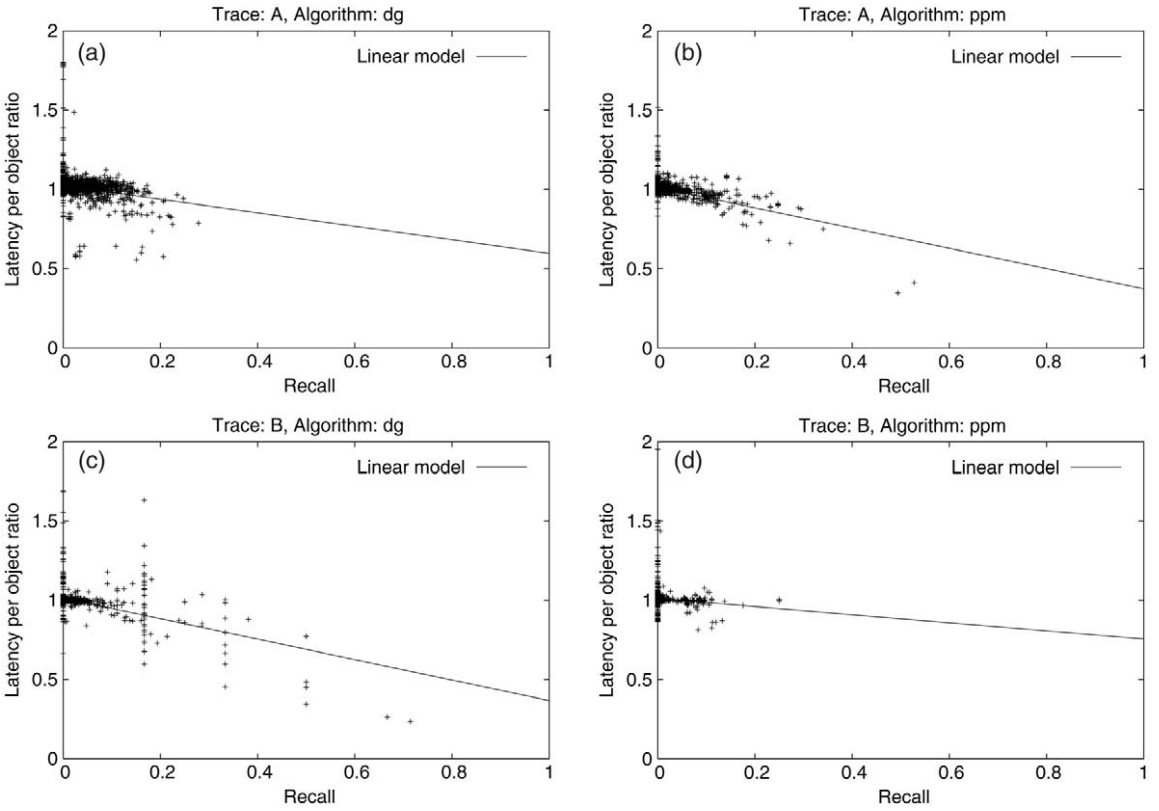

Fig. 7. *Recall* as a function of *latency per object* ratio.

per object ratio measures the performance from the point of view of HTTP. Therefore, it should be used when the meaning of a page is not so clear; for instance when we are measuring the latency in a proxy server.

Fig. 6 (graphs (a) and (b)) present an almost horizontal curve showing no apparent relation between precision and latency related indexes. The correlation coefficients showed in Table 4 quantify this negligible linear correlation, which confirms the visual appreciation. Graphs (c) and (d) show that there is a weak relation between the *byte precision* index and the latency related indexes. The remainder combinations of traces and algorithms are not shown because they are very similar.

Fig. 7 shows the relation between the *recall* and the *latency per object* ratio. As one can observe, depending on the workload and the prediction algorithm considered, a linear relation between the *recall* index and the *latency per object* ratio might exist. Some extra elements not gathered by the *recall*, like the queuing time and the object size heterogeneity, justify why the relation is not stronger. Fig. 8 shows that a certain correlation exists between the *recall* and the *latency per page* ratio, but it is far from being strong as the linear correlation coefficient shows in Table 4. The

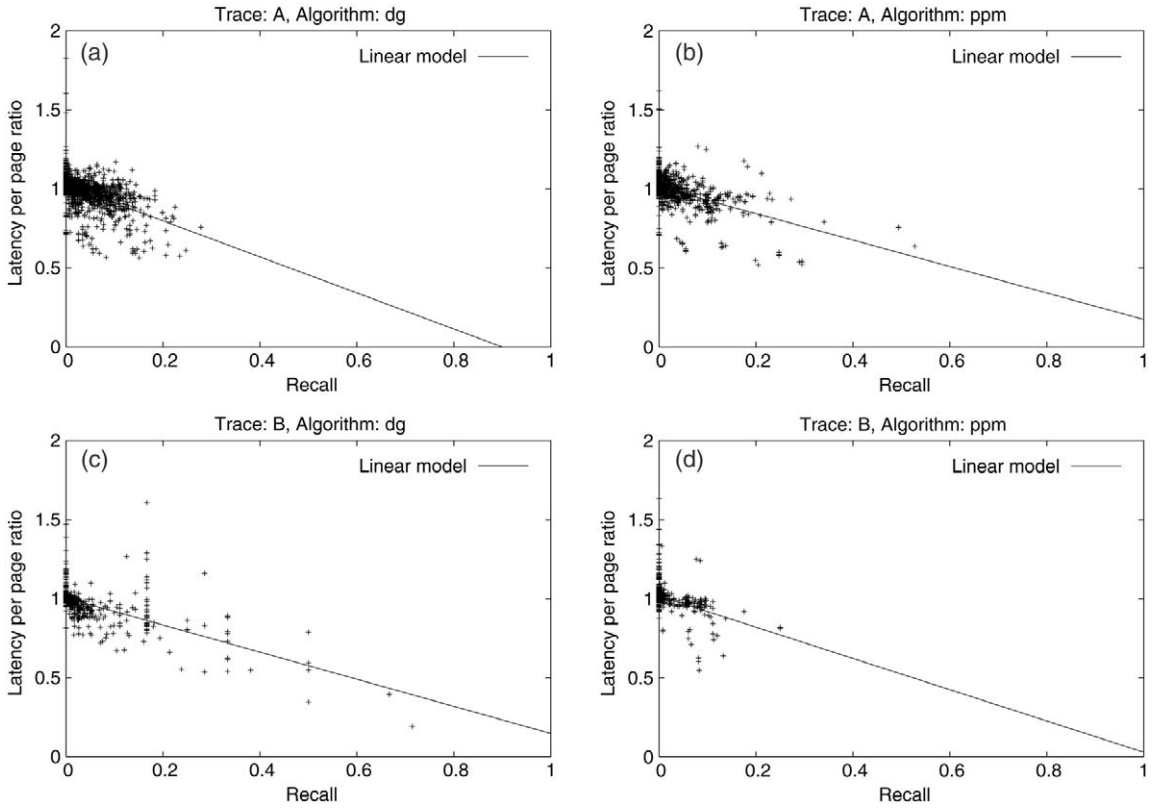

Fig. 8. *Recall* as a function of *latency per page* ratio.

latency per page ratio is not completely explained by the *recall* index since it involves more elements that affect it, like the simultaneous transference of objects (as explained in Section 3.3). These results corroborate the differences between both latencies observed in Fig. 5.

Figs. 9 and 10 show the relation between the *byte recall* and both latency indexes. The former presents a very weak correlation of the *byte recall* to the *latency per object* ratio; hence the fact that the *byte recall* does not explain the *latency per object* ratio. However, Fig. 10 and its associated correlation coefficient show that the *byte recall* is stronger correlated to the *latency per page* ratio than the *recall*, so this index explains better the user perceived latency. More details can be found in [14].

5.2. Analytical relations

In order to identify analytical relations among indexes we use the index definitions presented in Section 3. We found that the network level indexes can be obtained from the prediction related indexes. Eq. (11) shows that the *object traffic increase* depends on the *recall* (Rc) and the *precision* (Pc) achieved by the prefetching system. The equation is consistent with the meaning of both indexes: a decrease in the *precision* means a higher *object traffic increase*, whereas the higher the *recall* is, the lower the *object traffic increase* is.

$$\begin{aligned}\Delta Tr_{\text{ob}} &= \frac{ObjectsNotUsed + UserRequests}{UserRequests} \\ &= \frac{P - Pc \cdot P + \frac{Pc \cdot P}{Rc}}{\frac{Pc \cdot P}{Rc}} = 1 - Rc + \frac{Rc}{Pc}\end{aligned} \tag{11}$$

Eq. (12) shows the relation of the *traffic increase* index with those metrics related to the prediction; i.e., the *byte recall* (Rc_B), the *byte precision* (Pc_B) and the *extra control data per byte*. As this index is related to the number

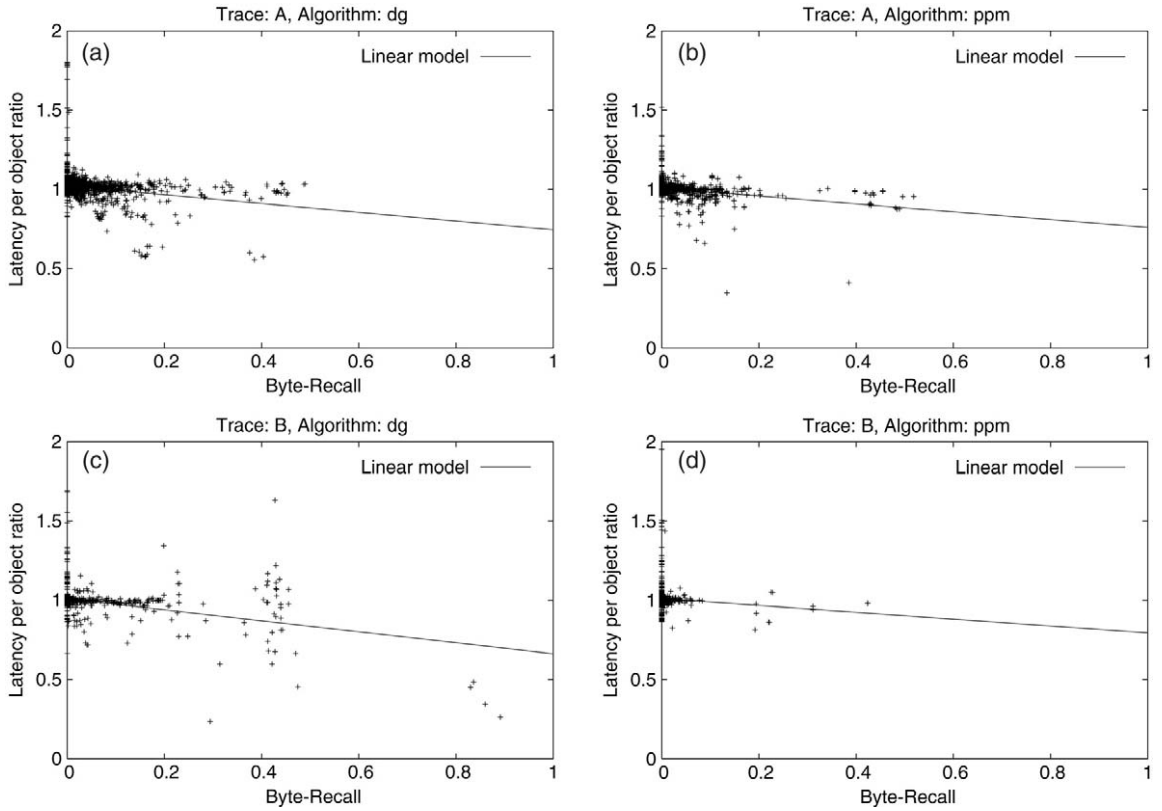

Fig. 9. *Byte Recall* as a function of *latency per object* ratio.

of bytes transferred, the main difference with the previous one lies in the fact that it depends on the *byte recall* and *byte precision* instead of on the *recall* and the *precision*. Notice that a new term appears with respect to Eq. (11) that refers to the *extra control data per byte*, so one can understand this term as the increase in network traffic due to the transference of data used to control the prefetching engine (e.g., prefetch hints) per each byte requested by the user.

$$\Delta Tr_B = \frac{ObjectsNotUsed_B + NetworkOverhead_B + UserRequests_B}{UserRequests_B} = \frac{P_B - Pc_B \cdot P_B + NO_B + \frac{Pc_B \cdot P_B}{Rc_B}}{\frac{Pc_B \cdot P_B}{Rc_B}}$$

$$= 1 - Rc_B + \frac{Rc_B}{Pc_B} + ExtraControlData_B \tag{12}$$

6. Conclusions

A large variety of key metrics have appeared in the open literature in order to evaluate the performance of web prefetching systems. This paper has analyzed this wide heterogeneity trying to (i) clarify both index definitions and how they are interrelated, and (ii) help when deciding which metric or index should be selected to evaluate the system performance.

We have proposed a taxonomy classifying the indexes related to prefetch in three main categories, according to the part of the system they pursue to evaluate: prediction, resource usage and latency. The goal of this taxonomy is not only to contribute to the understanding of the definition of the indexes. It is also aimed at analyzing analogies and differences between them, in order to identify which are the most useful metrics when carrying out performance studies.

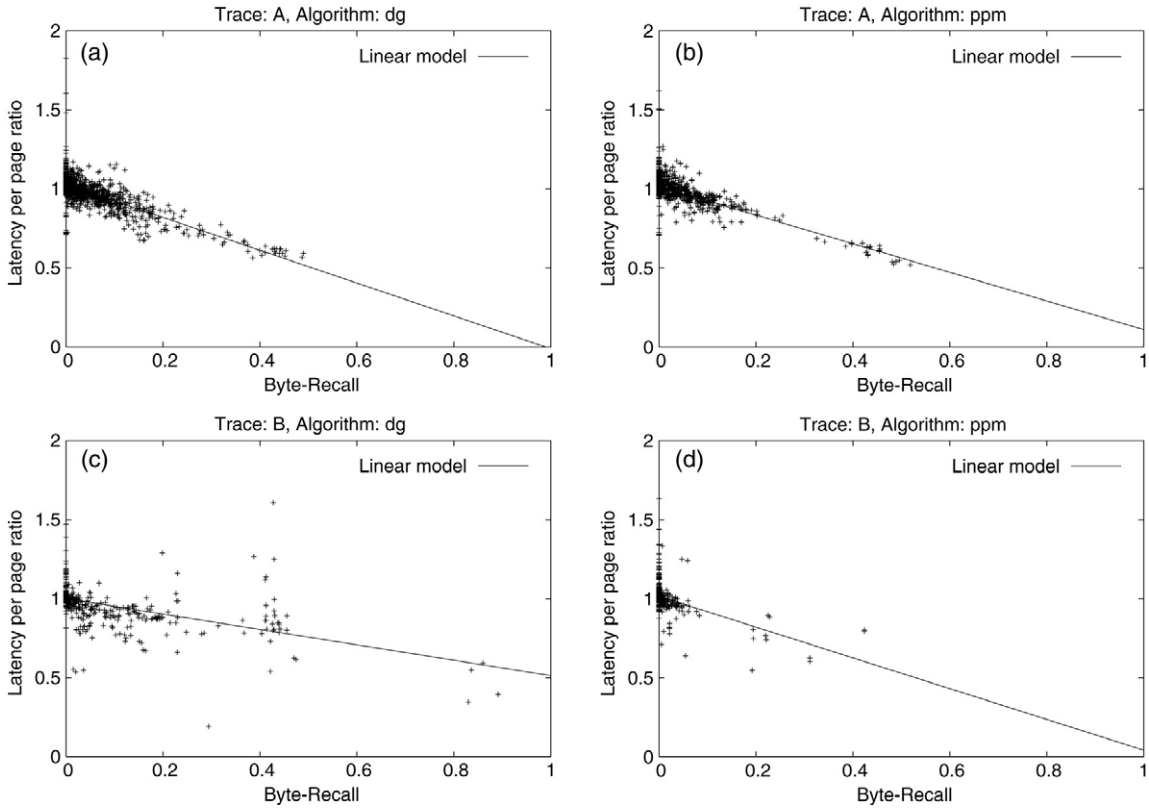

Fig. 10. *Byte Recall* as a function of *latency per page* ratio.

We have provided each metric or index with a definition (the one we considered more precise) from the large variety appeared in the literature. Then, looking for these definitions we observed certain analogies. We used both empirical and analytical approaches to show how indexes are interrelated. We found analytical relations between network level indexes and prediction related indexes. To check other possible relations among indexes, we ran experiments and calculated the statistical correlation among a representative set of them. From these results, we extract the main conclusions that follow:

– Performance studies should include at least latency related metrics. Depending on the goal of the performed study *latency per page* or *per object* is preferred. For instance, if the goal is to analyze the user's point of view, the latency per page must be included. However, when evaluating from the point of view of a proxy server, the *latency per page* makes no sense due to the lack of page concept. Consequently, the *latency per object* can be the most useful metric.
– Latencies cannot be used as the only metrics to check performance. Studies must analyze how the latencies reduction has been achieved for a given proposal. In this sense, resource usage indexes should be taken into account. *Traffic increase* is the one that provides more information; therefore, we suggest that performance studies should include at least this index.
– Our discussion has shown that is not recommendable to perform studies about the behavior of prefetching techniques just focusing on the algorithm point of view. Nevertheless, if some studies focus on this part, they should include *recall* and *byte recall* as performance metrics because they are the most correlated to the *latency per object* and *latency per page* respectively.

Acknowledgements

This work has been partially supported by Spanish Ministry of Education and Science and the European Investment Fund for Regional Development (FEDER) under grant TSI 2005-07876-C03-01.

References

[1] D. Albrecht, I. Zukerman, A. Nicholson, Pre-Sending documents on the WWW: A comparative study, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 1999.

[2] A. Bestavros, Using speculation to reduce server load and service time on the WWW, in: Proceedings of the 4th ACM International Conference on Information and Knowledge Management, Baltimore, USA, 1995.

[3] D. Bonino, F. Corno, G. Squillero, A real-time evolutionary algorithm for web prediction, in: Proceedings of the International Conference on Web Intelligence, Halifax, Canada, 2003.

[4] D. Bonino, F. Corno, G. Squillero, Dynamic prediction of web requests, in: Proceedings of the IEEE Congress on Evolutionary Computation, Canberra, Australia, 2003.

[5] C. Bouras, A. Konidaris, D. Kostoulas, Predictive prefetching on the web and its potential impact in the wide area, in: World Wide Web: Internet and Web Information Systems, vol. 7, Kluwer Academic Publishers, The Netherlands, 2004.

[6] E. Cohen, H. Kaplan, Prefetching the means for document transfer: A new approach for reducing web latency, in: Proceedings of the IEEE INFOCOM, Tel Aviv, Israel, 2000.

[7] E. Cohen, B. Krishnamurthy, J. Rexford, Efficient algorithms for predicting requests to web servers, in: Proceedings of the Conference on Computer and Communications, New York, USA, 1999.

[8] C.R. Cunha, C.F.B. Jaccoud, Determining WWW user's next access and its applications to pre-fetching, in: Proceedings of the 2nd International Symposium on Computers and Communication, Alexandria, Egypt, 1997.

[9] L. Cherkasova, G. Ciardo, Characterizing temporal locality and its impact on web server performance, in: Proceedings of the 9th International Conference on Computer Communication and Networks, Las Vegas, USA, 2000.

[10] B.D. Davison, Learning Web Request Patterns, in: Web Dynamics: Adapting to Change in Content, Size, Topology and Use, Springer, 2004.

[11] B.D. Davison, Predicting web actions from HTML content, in: Proceedings of the 13th ACM Conference on Hypertext and Hypermedia, College Park, USA, 2002.

[12] J. Domènech, A. Pont, J. Sahuquillo, J.A. Gil, An experimental framework for testing web prefetching techniques, in: Proceedings of the 30th Euromicro Conference, Rennes, France, 2004.

[13] J. Domènech, J. Sahuquillo, J.A. Gil, A. Pont, About the heterogeneity of web pre-fetching performance key metrics, in: Proceedings of the IFIP International Conference on Intelligence in Communication Systems, Bangkok, Thailand, 2004.

[14] J. Domènech, J. Sahuquillo, A. Pont, J.A. Gil, Design keys to adapt web prefetching algorithms to environment conditions, in: Proceedings of the 1st International Conference on Communication System Software and Middleware, New Delhi, India, 2006.

[15] X. Dongshan, S. Junyi, A new markov model for web access prediction, Computing In Science and Engineering 4 (6) (2002).

[16] D. Duchamp, Prefetching hyperlinks, in: Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, USA,1999.

[17] L. Fan, P. Cao, Q. Jacobson, Web prefetching between low-bandwidth clients and proxies: potential and performance, in: Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Atlanta, USA, 1999.

[18] D. Fisher, G. Saksena, Link prefetching in Mozilla: A server-driven approach, in: Proceedings of the 8th International Workshop on Web Content Caching and Distribution, New York, USA, 2003.

[19] T. Ibrahim, C.Z. Xu, Neural nets based predictive pre-fetching to tolerate WWW latency, in: Proceedings of the 20th IEEE International Conference on Distributed Computing Systems, Taipei, Taiwan, 2000.

[20] J.I. Khan, Q. Tao, Exploiting webspace organization for accelerating web prefetching, in: Proceedings of the International Conference on Web Intelligence, Halifax, Canada, 2003.

[21] R. Kokku, P. Yalagandula, A. Venkataramani, M. Dahlin, NPS: A non-interfering deployable web prefetching system, in: Proceedings of the USENIX Symposium on Internet Technologies and Systems, San Antonio, USA, 2003.

[22] T.M. Kroeger, D.D.E. Long, J.C. Mogul, Exploring the bounds of web latency reduction from caching and prefetching, in: Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, USA, 1997.

[23] T.S. Loon, V. Bharghavan, Alleviating the latency and bandwidth problems in WWW browsing, in: Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, USA, 1997.

[24] E.P. Markatos, C.E. Chronaki, A top-10 approach to prefetching the web, in: Proceedings of the INET' 98, Geneva, Switzerland, 1998.

[25] A. Nanopoulos, D. Katsaros, Y. Manopoulos, Effective prediction of web-user accesses: A data mining approach, in: Proceedings of Workshop on Mining Log Data across All Customer Touchpoints, San Francisco, USA, 2001.

[26] V. Padmanabhan, J.C. Mogul, Using predictive prefetching to improve World Wide Web latency, in: Proceedings of the ACM SIGCOMM '96 Conference, Palo Alto, USA, 1996.

[27] T. Palpanas, A. Mendelzon, Web prefetching using partial match prediction, in: Proceedings of the 4th International Web Caching Workshop, San Diego, USA, 1999.

[28] J. Pitkow, P. Pirolli, Mining longest repeating subsequences to predict World Wide Web surfing, in: Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, USA, 1999.

[29] M. Rabinovich, O. Spatscheck, Web Caching and Replication, Addison-Wesley, Boston, USA, 2002.

[30] S. Schechter, M. Krishnan, M.D. Smith, Using path profiles to predict HTTP requests, in: Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, 1998.

[31] Q. Tao, Impact of Webspace Organization and User Interaction Behavior on a Prefetching Proxy, Ph.D. Thesis, Kent State University, USA, 2002.

[32] I. Zukerman, D. Albrecht, A. Nicholson, Predicting users' requests on the WWW, in: Proceedings of the 7th International Conference on User Modeling, Banff, Canada, 1999.

Josep Domènech received the B.S. degree in Computer Science in 2000, the M.S. degree in Computer Science in 2002 and the M.S. degree in Multimedia Applications in 2005 from the Polytechnic University of València (UPV), and the B.S. degree in Business from the University of València in 2004.

He obtained the Best University student award given by the Valencian Local Government (Spain) in 2001. He is currently working toward the Ph.D. degree in the Architecture Research Group of the Department of Computer Engineering of the UPV since 2003. His research interests include Internet architecture, web prefetching and user characterization.

José A. Gil is an assistant professor of the computers architecture and technology area of the Polytechnic University of Valencia, Spain. He teaches at the Computer Engineering School where he is also member of the Staff.

Professor Gil obtained his B.S., M.S. and Ph.D. degrees from the Polytechnic University of Valencia. His current interests include topics related with multimedia systems, web systems, proxy cache and distributed shared memory systems. He is joint author of several books on the subject of computer architecture and he has published numerous articles about industrial local area networks, computer evaluation and modelling, proxy cache systems and web systems. It has participated in numerous investigation projects financed by the Spanish Government and the Local Government of the Comunidad Valenciana and in development projects for different companies and city councils.

Julio Sahuquillo received his B.S., M.S., and Ph.D. degrees in Computer Science from the Polytechnic University of Valencia (UPV), in València, Spain. Since 2002 he is an associate professor at the Computer Engineering Department at the Polytechnic University of Valencia. His current research topics have included multiprocessor systems, cache design, instruction-level parallelism, and power dissipation. Recently, an important part of his research has concentrated on the web performance field, including proxy caching, web prefetching, and web workload characterization.

Ana Pont-Sanjuán received her M.S. degree in Computer Science in 1987 and a Ph.D. in Computer Engineering in 1995, both from Polytechnic University of Valencia. She joined the Computer Engineering Department in the UPV in 1987 where currently she is a full professor of Computer Architecture. Since 1998 until 2004 she was the head of the Computer Science High School in the UPV. Her research interest include multiprocessor architecture, memory hierarchy design and performance evaluation, web and internet architecture, proxy caching techniques, CDNs and communication networks.

Professor Ana Pont-Sanjuán has published a substantial number of papers in international Journals and Conferences on Computer Architecture, Networks and Performance Evaluation. She has been reviewer for several journals and regularly participates in the technical program committees of international scientific conferences.

She also has participated in a large number of research projects financed by the Spanish Government and Local Valencian Government. Currently she leads the Web Architecture Research group at the UPV where she has been advisor in several Ph.D. Thesis and currently she is directly tutoring six Ph.D. more.

Since January 2005 she is the Chairperson of the IFIP TC6 Working Group 6.9: Communication Systems for Developing Countries.