La Heterogeneidad de los Índices de Prestaciones de la Prebúsqueda Web

Josep Domènech, José A. Gil, Julio Sahuquillo, Johann Márquez y Ana Pont
Departamento de Informática de Sistemas y Computadores (DISCA)
Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, España
{jodode,jomarba}@doctor.upv.es ; {jagil@jsahuqui,apont@disca.upv.es}

Abstract

Web prefetching techniques are becoming important solutions to reduce the user perceived latency. Nevertheless, it is not possible to make a general fair comparison among the proposed techniques due to three main reasons: the underlying baseline system differs among the different studies; heterogenous workloads are used, and also different performance key metrics are considered.

This paper pursues to classify the most used indexes in the open literature, when studying the performance of the web prefetching techniques. For this purpose, we propose a three categories based taxonomy which identifies analogies and differences among the indexes. To evaluate the performance in an appropriate manner it is extremely important to suitably choose the indexes. This taxonomy suggest which indexes should be used to evaluate correctly the performance. The experiments show that depending on the chosen index, the obtained performance results can not only vary among them, but could also reach opposite conclusions.

Keywords: Web prefetching, Web performace evaluation, Taxonomy Metrics.

Resumen

Las técnicas de prebúsqueda en la Web se perfilan importantes para reducir la latencia percibida por el usuario. Sin embargo, no es posible realizar en general una comparación equitativa entre las técnicas propuestas debido principalmente a tres razones: el sistema subyacente difiere entre los estudios; se utilizan distintas cargas, y se cuantifican distintos índices de prestaciones.

Este trabajo persigue clasificar los índices utilizados, en la literatura abierta, cuando se estudian las prestaciones de las técnicas de prebúsqueda. Para ello, se propone una taxonomía basada en tres categorías que identifica analogías y diferencias entre los índices. Para realizar una correcta evaluación de prestaciones es de crucial importancia elegir adecuadamente los índices. La taxonomía sugiere qué índices deben utilizarse para evaluar correctamente las prestaciones y de qué forma se ha alcanzado la mejora del rendimiento. Experimentalmente se muestra que dependiendo del índice utilizado, los resultados de prestaciones obtenidos, pueden no sólo diferir ostensiblemente entre ellos, sino llegar a conclusiones opuestas.

Palabras claves: Prebúsqueda Web, Taxonomia de métricas, Evaluación de prestaciones Web

1. Introducción

La naturaleza internacional y global de Internet hace ardua la tarea de incrementar las prestaciones del sistema trabajando en el hardware de red y en sus elementos de interconexión. En consecuencia los mayores esfuerzos investigadores se han centrado en la arquitectura de la Web y su organización aplicando técnicas previamente utilizadas en la arquitectura de computadores. La mayoría de estas técnicas explotan la localidad inherente a los accesos a objetos Web. En la Web, la localidad tiene tres características: temporal, espacial y geográfica, lo cual permite implementar eficientemente técnicas de *caching*, prebúsqueda y replicación de contenido para mejorar las prestaciones.

Este artículo se centra en técnicas de prebúsqueda, aunque algunas de las conclusiones presentadas pueden extenderse al análisis general de las prestaciones de la Web. Muchos estudios de investigación abordan estas técnicas; aunque en general, estas propuestas son difícilmente comparables entre sí, debido a que se implementan sobre distintos sistemas subyacentes y proporcionan resultados bajo distintas cargas y condiciones. Además, cada estudio cuantifica distintos índices de prestaciones.

Para evaluar las prestaciones de manera equitativa, se deben solucionar las limitaciones mencionadas. Para abordar la primera de ellas, en un trabajo previo [13] se propuso un entorno experimental general, que facilita la implementación de técnicas de prebúsqueda de forma flexible, bajo la misma plataforma y cargas.

Para resolver la segunda limitación, el presente artículo, analiza un importante conjunto de métricas y se propone una taxonomía basada en tres categorías, que nos permite identificar analogías y diferencias entre ellas y constatar experimentalmente su relación. Además se proponen nuevos índices basados en byte como métricas complementarias.

2. Arquitectura Genérica de Prebúsqueda y Glosario Básico

Se asume una arquitectura genérica de Web compuesta por tres elementos principales: clientes, servidores y *proxies*. Es importante destacar la diferencia entre un cliente y un usuario. El usuario es la persona solicitando información en el computador, mientras que el cliente es el software (por ej., un navegador) que interactúa con el usuario solicitando la información demandada hacia el servidor apropiado. Puede observarse que los *proxies* actúan como clientes para el servidor y como servidores para el cliente.

El principal objetivo de las técnicas de prebúsqueda es reducir la latencia media percibida por el usuario. Algunas técnicas han propuesto ideas para beneficiarse de explotar la localidad espacial de los objetos Web proponiendo que los clientes descarguen objetos antes de que el usuario los requiera [17, 26, 16, 21]; otros estudios proponen preprocesar el contenido dinámico [30], y algunos otros se concentran en cómo establecer preconexiones con el servidor [7]. Todas las técnicas relacionadas con la prebúsqueda parten de una predicción de los siguientes objetos a los que el cliente accederá. A esta parte de la prebúsqueda se la denomina motor de predicción. A continuación, los resultados de la predicción se envían al motor de prebúsqueda, que decide si prebuscar o no esos objetos dependiendo de otros parámetros, por ej. ancho de banda disponible o nivel de carga en el servidor. De esta manera, los objetos prebuscados son un subconjunto de los objetos predichos. Nótese que ambos motores (predicción y prebúsqueda) pueden encontrarse en el mismo elemento (cliente, *proxy* o servidor), o bien estar distribuidos.

El hecho de utilizar distintos nombres para referirse a la misma métrica o bien el mismo nombre para referirse a distintas métricas, crea confusión y ambigüedad cuando se comparan distintos estudios. Este hecho se magnificaría todavía más si se tradujesen los índices al español. Como consecuencia a lo largo del trabajo se utilizarán los índices y variables tal y como aparecen en su versión original. A continuación, se definen algunas variables básicas que se utilizarán en la sección 3:

UserRequests: cantidad de objetos demandados por el usuario.

Predictions: cantidad de objetos predichos por el motor de predicción. *GoodPredictions (B)*: cantidad de predicciones que luego son demandadas por el usuario. *BadPredictions (E)*: aquellas predicciones no incluidas en *GoodPredictions*.

Prefetchs: cantidad de objetos prebuscados por el motor de prebúsqueda. *PrefetchHits (C)*: objetos prebuscados que son posteriormente demandados por el usuario. *ObjectsNotUsed (D)*: prebuscados nunca demandados por el usuario.

La figura 1 muestra la relación entre las variables definidas. A representa objetos solicitados por el usuario que no fueron ni predichos ni prebuscados. B representa las *GoodPredictions* que no han sido prebuscadas. C representa las *PrefetchHits* (*GoodPredictions* que han sido prebuscadas). D está formado por *ObjectsNotUsed*,

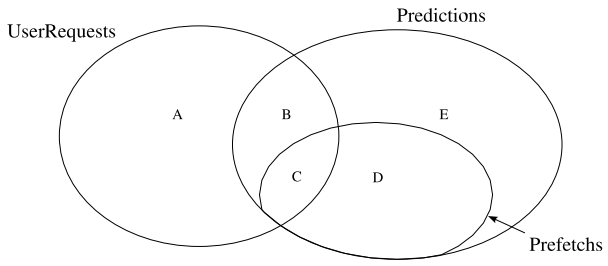

Figura 1: Relación de Variables Básicas de Prebúsqueda

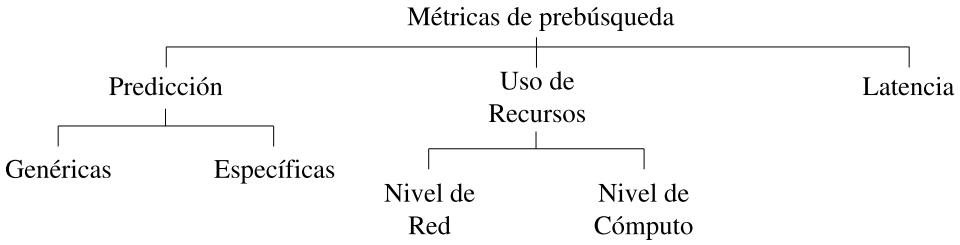

Figura 2: Taxonomía de Métricas de Prebúsqueda

incluidos también en *BadPredictions*. Finalmente E representa *BadPredictions*.

3. Taxonomía de Índices de Prestaciones de la Web

Esta sección ilustra los índices de prestaciones de la Web que aparecen en la literatura relacionados con la prebúsqueda. Para una mejor comprensión de estos índices, se clasifican en tres categorías (ver figura 2); atendiendo a la parte del sistema que evaluan

- Categoría 1: Índices relacionados con la predicción.
- Categoría 2: Índices de uso de recursos.
- Categoría 3: Índices de latencia percibida de extremo a extremo.

3.1. Índices Relacionados con la Predicción

Este grupo incluye aquellos índices orientados a cuantificar las prestaciones del algoritmo de predicción. Estas prestaciones pueden cuantificarse en distintos elementos de la arquitectura Web. Cada índice en esta categoría tiene un índice dual; por ejemplo, pudiéndose referirse a la precisión del algoritmo y a la precisión de la prebúsqueda. Nótese que si se cuantifican estos índices al proporcionar la lista de predicción no se tendrán en cuenta las latencias del sistema debido a que el algoritmo de predicción es independiente de las subcapas de red y los factores que afectan a dichas subcapas.

3.1.1. Índices Genéricos de Predicción

Precision (Pc): Mide la tasa de buenas predicciones sobre el número de predicciones [3, 1, 29, 8, 11] (ec. 1). Algunos estudios se refieren a este índice como *accuracy* [16, 9, 32, 23, 25, 27, 15] o *hit ratio* [5], mientras algunos de los modelos basados en cadenas de Markov utilizan la notación probabilística $Pr(hit \mid match)$ [28]. Otros trabajos miden el impacto de la precisión en las prestaciones [16, 23]. En estos casos, se utiliza el número de objetos y el número de aciertos prebuscados en lugar del número de predicciones y buenas predicciones respectivamente (ec. 2).

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

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

En [19] se evalúa el rendimiento de prebúsqueda mediante el índice *waste ratio*, que se define como el porcentaje de documentos prebuscados que no son posteriormente requeridos. Este índice es el complemento de *precision*.

Recall (Rc): Mide el porcentaje de objetos solicitados por el usuario que fueron previamente prebuscados [1, 8, 29]. El índice recall cuantifica el peso de los objetos predichos (ec. 3) o prebuscados (ec. 4) sobre la cantidad de objetos solicitados por el usuario.

Otros trabajos se refieren a esta métrica como *Usefulness* [25, 27, 5], *Hit Ratio* [24, 19] o *Accuracy* [10, 11]. *Predictability* se usa en [30] para referirse al límite superior del *recall*.

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

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

Applicability: En [4, 3] se define como la tasa del número de predicciones sobre el número de solicitudes (ec. 5). En [4] se considera de manera errónea este índice como sinónimo de *Recall*. Este índice puede obtenerse a partir de índices previos (dividiendo *recall* entre *precision*); por tanto, no proporciona información adicional.

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

El tiempo que transcurre desde que se hace la solicitud hasta su respuesta consta de cuatro componentes principales: tiempo de establecimiento de la conexión, tiempo de transferencia de la petición, tiempo de procesamiento de la petición y tiempo de transferencia de la respuesta.

Tradicionalmente, para evaluar la bondad de los algoritmos de predicción se ha utilizado el índice *precision* sólo o junto con *recall*. Ambos índices están estrechamente relacionados con los tres primeros componentes del tiempo de espera.Un buen estudio que compare algoritmos de predicción debería también incluir índices relacionados con byte para cuantificar este último componente. Análogamente a los índices de *web proxy caching* (por ej., *byte hit ratio*) [6], se propone el uso de byte precision y byte recall como índices para estimar el impacto de la prebúsqueda en el tiempo que el usuario pasa esperando los bytes de los objetos solicitados.

Byte Precision (Pc_B). Mide el porcentaje de bytes predichos (o prebuscados) que son posteriormente solicitados. Se calcula remplazando el número de objetos predichos por su tamaño en bytes en la ecuación 1.

Cabe destacar que, al igual que *precision, byte precision* cuantifica la bondad de las predicciones, pero estimadas sobre la base de bytes en lugar de número de objetos (ec. 6). Una aproximación a este índice es el Miss rate ratio [2]. En [5] también menciona que pueden obtenerse resultados diferentes usando el número de bytes en lugar del número de objetos.

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

Byte Recall (Rc_B). Mide el porcentaje de bytes solicitados que fueron previamente predichos (o prebuscados). Como se ha mencionado anteriormente, este índice estima cuántas predicciones correctas se realizaron, medidas en bytes transferidos (ec. 7). Este índice es más útil que recall cuando el tiempo de transferencia de la respuesta es un componente importante de la percepción total de la latencia.

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

3.1.2. Índices Específicos de Predicción

Request Savings. Mide el porcentaje de veces que una petición de usuario acierta en la *cache* del navegador (o el objeto solicitado esta siendo prebuscado) sobre el número total de peticiones de usuario [17]. Las peticiones pueden desglosarse en tres grupos dependiendo de si fueron previamente prebuscadas, parcialmente prebuscadas o almacenadas en el *cache* como objetos normales.

Nótese que cuando la prebúsqueda es iniciada por el usuario, el número de peticiones incrementa; por lo que sólo tiene sentido utilizar este índice cuando la prebúsqueda se inicia desde el proxy o desde el servidor. Fan *et al.* [17] usan este índice en un sistema de prebúsqueda donde el *proxy* "empuja" objetos hacia el cliente. Sin embargo, este índice puede también usarse cuando la prebúsqueda "empuja" objetos desde el servidor hacia el *proxy* o desde el servidor hacia el cliente (sin *proxies* intermedios).

Miss Rate Ratio. Bestavros [2] lo define como la razón entre *byte miss rate* utilizando prebúsqueda y *byte miss rate* sin utilizar prebúsqueda, donde *byte miss rate* es la tasa de bytes no encontrados en la *cache* del cliente respecto al total de bytes accedidos por el cliente. Este índice solo es aplicable a los sistemas que almacenan los objetos prebuscados en la *cache* del cliente.

Probability of Making a Prediction. En [28] se cuantifica la probabilidad de que los últimos accesos coincidan con un patrón de predicción; en este caso el sistema de prebúsqueda realiza la predicción. Este índice puede aplicarse a aquellos sistemas basados en modelos de Markov, pero no a una gran parte de sistemas de prebúsqueda, por ejemplo los basados en Top-10 [24]; por ello se clasifica como específico.

3.2. Índices de Uso de Recursos

Los beneficios de la prebúsqueda se alcanzan utilizando una cantidad adicional de recursos. Este sobrecoste debe cuantificarse porque puede impactar negativamente en las prestaciones.

Aunque algunos algoritmos de predicción pueden requerir mucha memoria o tiempo de procesador (por ej. modelos de Markov de orden elevado) no es la tendencia general actual, donde el principal cuello de botella en la prebúsqueda es el tráfico en la red. Por lo tanto, dividimos los índices de esta categoría en dos subgrupos; nivel de red y nivel de computo.

3.2.1. Nivel de Red

Traffic Increase ($\triangle Tr_B$.) Cuantifica el incremento de tráfico (en bytes) debido a los documentos prebuscados que posteriormente no se solicitan [24] (ec. 8). También se denomina *waste bandwidth* [17], *extra bytes* [29], *network traffic* [27, 5] y *bandwidth ratio* [2].

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

Cuando se usa prebúsqueda, el tráfico de la red se incrementa debido a dos efectos: predicciones incorrectas y sobrecarga. Las malas predicciones desperdician el ancho de banda porque se prebuscan objetos que el usuario no requiere posteriormente. Por otra parte, el tráfico de red se incrementa debido al intercambio de información adicional relacionada con el control de la prebúsqueda, denominado *network overhead* en [16].

Muchos estudios no contabilizan el *network overhead* [25, 24, 20], por lo que sus resultados no estiman el coste de la prebúsqueda de forma precisa.

Extra Control Data per Byte. Cuantifica los bytes extras transferidos relacionados con la información de prebúsqueda, promediado por cada byte pedido por el usuario (ec. 9). Se trata del *network overhead* referenciado en [16], pero promediado por bytes solicitados, y es útil para relacionar el incremento de tráfico con los índices de predicción [12].

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

Object Traffic Increase ($\triangle \mathrm{Tr}_{ob}$). Cuantifica el porcentaje de documentos de más que un cliente solicita cuando utiliza prebúsqueda. Nanopoulous *et al.* [25] se refieren a este índice como *network traffic* y Rabinovich [29] como *extra requests*.

Como muestra la ecuación 10, este índice estima la tasa de objetos prebuscados que no se utilizan respecto al total de objetos solicitados por el usuario. Es análogo al *traffic increase*, pero mide la sobrecarga en número de objetos.

$$\triangle Tr_{ob} = \frac{ObjectsNotUsed + UserRequests}{UserRequests} \quad (10)$$

3.2.2. Nivel de Cómputo

Server Load Ratio. Se define como la razón entre el número de solicitudes por tiempo de servicio cuando se aplica especulación y el número de solicitudes por tiempo de servicio sin aplicar especulación [2].

Space and Time Overhead. Además de la carga del servidor, otros trabajos tratan de observar cómo afectan otros aspectos de la sobrecarga a las prestaciones. Por ejemplo, Duchamp [16] hace referencia a la memoria y el tiempo de proceso que necesitaría la prebúsqueda.

Prediction Time. Cuantifica el tiempo que el predictor necesita para realizar predicciones. Es utilizado en [15] para comparar algoritmos de predicción.

3.3. Índices Relacionados con la Latencia

Los índices pertenecientes a esta categoría persiguen cuantificar las latencias de extremo a extremo, es decir, latencias relacionadas con el usuario o el *proxy*. La principal desventaja de estos índices es que incluyen varios componentes de tiempo, algunos de ellos difíciles de cuantificar. Muchas veces los investigadores no detallan qué componentes miden, aunque utilizan genéricamente como nombre de índice *latency*. También se han utilizado otros nombres como *access time* [26], *service time* [2], y *responsiveness* [20, 31]. Esta situación no es buena para la comunidad investigadora, porque las diferentes propuestas no pueden compararse entre sí de manera equitativa.

En los distintos trabajos de investigación, las latencias se miden tanto por página como por objeto. La *latencia por página* (L_p) se calcula como el tiempo transcurrido entre la petición (GET) de la página HTML por el cliente y la recepción del último byte del último objeto embebido en esta página [16]. Análogamente, la *latencia por objeto* (L_{ob}) puede definirse como el tiempo transcurrido desde que el usuario hizo la solicitud hasta que recibió el último byte de ese objeto.

Para ilustrar los beneficios de la prebúsqueda, los investigadores calculan la razón entre la latencia obtenida cuando se utiliza la prebúsqueda (ya sea por página [17, 16, 23] o por objeto [22]) y la latencia sin prebúsqueda. La mayoría de las propuestas que utilizan *latency* para cuantificar las prestaciones no especifican a cual de ellas se refiere. Esto puede ser engañoso ya que ambos índices no se comportan igual, como se muestra en la sección 4.

3.4. Resumen: Sinónimos y Categoría de Índices Experimentales Utilizados

A lo largo de la taxonomía propuesta, se analiza una amplia variedad de nombres de índices (sinónimos) utilizados para referirse a la misma métrica. Esta heterogeneidad puede observarse a través de distintos estudios que pueden encontrarse en la literatura, dificultando una visión general del tema. El cuadro 1 muestra un resumen de la situación actual, especificando los índices que se utilizan en los trabajos más representativos. Además, encontramos que el mismo nombre de índice se ha utilizado para cuantificar diferentes variables (por ej., *accuracy* es utilizado tanto para *precision* como para *recall*). Como puede observarse, la heterogeneidad más amplia aparece en la primera categoría debido al elevado número de algoritmos de predicción propuestos y a su importancia en los sistemas de prebúsqueda.

El cuadro 2 muestra las categorias de índices utilizados en los trabajos más representativos.

4. Resultados Experimentales

En esta sección se describe el entorno experimental utilizado, el algoritmo de prebúsqueda evaluado y, finalmente, los resultados obtenidos.

4.1. Arquitectura del Sistema

En [13] se propuso y describió con detalle un entorno experimental para diseñar y analizar técnicas de prebúsqueda en la Web. La arquitectura está formada por tres partes: El servidor Web, el *surrogate* o *proxy* inverso, el cliente y, opcionalmente, el *proxy*, que no se utiliza en este artículo (figura 3). Este entorno combina partes tanto simuladas como reales permitiendo flexibilidad y precision.

Cuadro 1: Relación entre Nombres de Índices Seleccionados en este Artículo y los de la Literatura.

Predicción		
Precision	*Recall*	*Applicability*
Precision:[4, 1, 29, 9, 11, 3]	Recall:[1, 29, 8]	Applicability:[4, 3]
Accuracy:[16, 8, 32, 23, 25, 27, 15]	Usefulness:[25, 27, 5]	
Pr (Hit/match):[28]	Hit ratio:[24, 19]	
Hit ratio:[5]	Predictability:[30]	
Waste ratio:[19]	Accuracy:[11, 10]	
Uso de recursos		
Traffic Increase	*Object Traffic increase*	
Traffic increase:[26, 16, 24]	Network Traffic:[25]	
Wasted Bandwidth:[17]	Extra requests:[29]	
Bandwidth ratio:[2]	Prediction time:[15]	
Extra bytes:[29]		
Data transfer:[20]		
Network Traffic:[27, 5]		
Latencia		
Latency per page	*Latency per object*	
Latency:[17, 16, 23]	Latency:[21, 22, 5]	
Responsiveness:[20, 31]	Access time:[26]	
	Service time ratio:[2]	

Cuadro 2: Relación entre Estudios y Categorías Usadas

Referencias	*Predicción*	*Recursos*	*Latencia*	*%*
[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, 4, 1, 9, 8, 32, 10, 28, 11, 19, 3]	X			44
TOTAL	80 %	40 %	40 %	

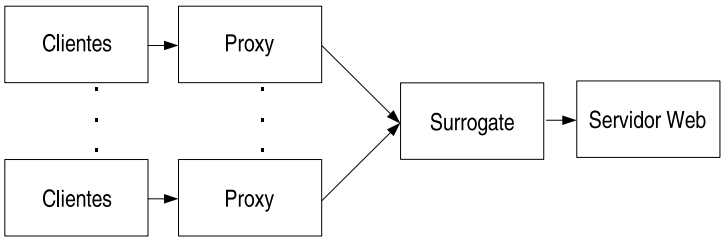

Figura 3: Arquitectura del Entorno de Simulación

Junto al servidor, el entorno emula un *proxy* inverso (*surrogate*), el cual se utiliza de pasarela para acceder a un servidor Web real. Aunque el principal objetivo de un *surrogate* es el de actuar como una cache de las respuestas más populares del servidor, en este entorno se utiliza como motor de predicciones, y sus predicciones son añadidas a la respuesta del servidor. El Cliente reproduce la conducta de usuarios que utilizan un navegador que implementa un motor de prebúsqueda (como por ej. Mozilla [18]). Para modelar un conjunto de usuarios accediendo concurrentemente a un servidor, el simulador puede usar tanto trazas reales como sintéticas. El simulador recoge información básica por cada petición realizada al servidor Web y la escribe en una traza que, posteriormente, permitirá calcular las métricas de rendimiento.

4.2. Algoritmo de Prebúsqueda

Para obtener los resultados experimentales, se ha implementado el algoritmo de prebúsqueda propuesto por Padmanabhan y Mogul [26]. Este algoritmo construye un grafo de dependencia que representa el patrón de accesos de usuario en un servidor Web. El grafo tiene un nodo por cada objeto web que ha sido accedido. Existe un arco del nodo A al nodo B siempre y cuando en algún momento un cliente haya accedido a B dentro de los w accesos posteriores al acceso de A, donde w es el tamaño de la *ventana de visión*. El peso del arco es la razón entre el número de accesos a B dentro de la ventana después de A y el número de accesos a A. La agresividad de la prebúsqueda se controla por un *umbral*, de manera que los objetos con una probabilidad mayor que este valor de ser solicitados en los siguientes k accesos son prebuscados. En los experimentos tomamos $k = 4$ y valores de *umbral* entre 0.1 y 0.9.

El sistema se configuró para simular accesos de usuario a *Marca* (www.marca.es). Se utilizó una traza recogida durante una semana (alrededor de 145.000 accesos) para entrenar el motor de predicción mientras que la traza del día siguiente (aproximadamente 35.000 accesos) se usó para obtener los resultados de la simulación. Cada simulación reproduce el comportamiento de 250 clientes aproximadamente. Los puntos trazados en la figura 4 hacen referencia a los índices de prestaciones medidos para cada cliente en cada experimento, que hacen unos 2250 puntos en total. El ancho de banda considerado para cada simulación está entre 48 kbps y 400 kbps. Debido a las limitaciones de espacio en el artículo, solo se presentan resultados de 200 kbps.

4.3. Resultados Obtenidos

La figura 4 muestra la *latency per object ratio* como funcion de la *latency per page ratio*, ambos refiriéndose a diferentes alternativas de cómo puede ser medida la latencia (como se mencionó en la sección 3.3). Nótese que dependiendo de la forma en que se mida la latencia es posible obtener no sólo resultados distintos sino también opuestos. Si se toma un punto del cuadrante superior izquierdo y se considera la *Latency per object* el sistema con prebúsqueda no mejora el rendimiento del sistema sin prebúsqueda. Sin embargo si se considera la *Latency per page*, podemos concluir lo contrario.

Consecuentemente, se sugiere que los estudios deben diferenciar el uso de ambos índices porque cada uno cuantifica la latencia desde diferentes puntos de vista. La *latency per page* evalúa el sistema desde el punto de vista del usuario (ya que mide la latencia percibida por el usuario) mientras que la *latency per object* mide el rendimiento desde el punto de vista del protocolo. Por lo tanto, esta última deberá usarse cuando lo que significa una página no esté muy claro, por ejemplo en un *proxy*. En [14] se muestran más experimentos que apoyan esta idea, como también relaciones entre otros índices.

5. Conclusiones

Para evaluar el rendimiento de los sistemas de prebúsqueda en la Web se ha venido utilizando una amplísima variedad de métricas. Este artículo analiza la heterogeneidad existente tratando de clarificar las definiciones de índices y de ver cómo están relacionados.

Se ha propuesto una taxonomía, que clasifica los índices relacionados con la prebúsqueda en tres categorías de acuerdo con la parte del sistema que están destinados a evaluar. Para cada métrica o índice se proporciona una definición (la que se considera más precisa) entre una gran variedad encontrada en la literatura. El análisis realizado permite extraer principalmente las siguientes conclusiones:

- Los estudios de prestaciones deberían incluir al menos métricas relacionadas con latencia. Aunque depende del objetivo del estudio, es preferible la latencia por página.

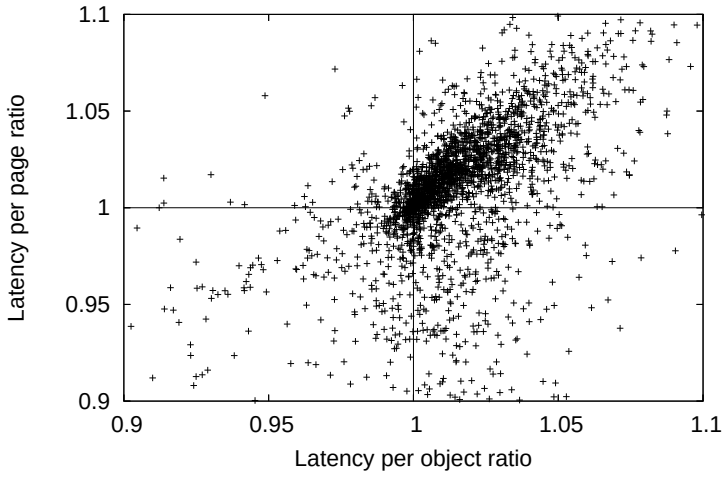

Figura 4: *Latency per Page Ratio* vs. *Latency per Object Ratio*

- Las latencias por sí solas no pueden ser utilizadas como métricas para analizar el rendimiento, debiéndose tener en cuenta los recursos del sistema empleados, por lo que se deben utilizar también índices de esta categoría.
- En caso de que un estudio se centre únicamente en las prestaciones del algoritmo, éste debería incluir *byte precision* y *byte recall* como métricas de prestaciones por estar fuertemente relacionadas con la *latency per object* y *latency per page*.

Agradecimientos

Este trabajo fue realizado con el apoyo parcial de los siguientes programas:

- Ministerio de Educación y Ciencia (España) y el Fondo Europeo de Desarrollo Regional (FEDER) bajo la beca TSI 2005-07876-C03-01.
- Programa Alβan, Programa de becas de alto nivel de la Unión Europea para América Latina, beca nº E04D031142BO.

Referencias

[1] David W. Albrecht, Ingrid Zukerman, and Ann E. Nicholson. Pre-sending documents on the www: A comparative study. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[2] Azer 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] Dario Bonino, Fulvio Corno, and Giovanni Squillero. Dynamic prediction of web requests. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Canberra, Australia, 2003.

[4] Dario Bonino, Fulvio Corno, and Giovanni Squillero. A real-time evolutionary algorithm for web prediction. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003.

[5] Christos Bouras, Agisilaos Konidaris, and Dionysios Kostoulas. Predictive prefetching on the web and its potential impact in the wide area. *World Wide Web*, 7(2):143–179, 2004.

[6] L. Cherkasova and G. Ciardo. Characterizing temporal locality and its impact on web server performance, 2000.

[7] Edith Cohen and Haim Kaplan. Prefetching the means for document transfer: A new approach for reducing web latency. In *Proceedings of the IEEE INFOCOM '00 Conference*, Tel Aviv, Israel, 2000.

[8] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Efficient algorithms for predicting requests to web servers. In *Proceedings of the IEEE INFOCOM '99 Conference*, New York, USA, 1999.

[9] Carlos Cunha and Carlos F.B. Jaccoud. Determining www user's next access and its application to pre-fetching. In *Proceedings of the Second IEEE Symposium on Computers and Communications*, Alexandria, Egypt, 1997.

[10] Brian D. Davison. Predicting web actions from html content. In *Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, College Park, USA, 2002.

[11] Brian D. Davison. Learning web request patterns. In *Web Dynamics - Adapting to Change in Content, Size, Topology and Use*, pages 435–460. Springer, 2004.

[12] Josep Domènech, José A. Gil, Julio Sahuquillo, and Ana Pont. Web prefetching performance metrics: A survey. *To be published in Performance Evaluation*, 2006.

[13] Josep Domènech, Ana Pont, Julio Sahuquillo, and José A. Gil. An experimental framework for testing web prefetching techniques. In *Proceedings of the 30th EUROMICRO Conference 2004*, pages 214–221, Rennes, France, 2004. IEEE Computer Society.

[14] Josep Domènech, Julio Sahuquillo, José A. Gil, and Ana Pont. About the heterogeneity of web prefetching performance key metrics. In *Proceedings of the IFIP International Conference on Intelligence in Communication Systems (INTELLCOMM 2004)*, Bangkok, Thailand, 2004.

[15] Xing Dongshan and Shen Junyi. A new markov model for web access prediction. *Computing in Science and Engineering*, 4(6):34–39, 2002.

[16] Dan Duchamp. Prefetching hyperlinks. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, Boulder, USA, 1999.

[17] Li Fan, Pei Cao, Wei Lin, and Quinn 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*, pages 178–187, Atlanta, USA, 1999.

[18] Darin Fisher and Gagin Saksena. Link prefetching in mozilla: A server driven approach. In *Proceedings of the 8th International Workshop on Web Content Caching and Distribution (WCW 2003)*, New York, USA, 2003.

[19] Tamer I. Ibrahim and Cheng-Zhong Xu. Neural nets based predictive prefetching to tolerate www latency. In *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, Taipei, Taiwan, 2000.

[20] Javed I. Khan and Qingping Tao. Exploiting webspace organization for accelerating web prefetching. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003.

[21] Ravi Kokku, Praveen Yalagandula, Arun Venkataramani, and Michael Dahlin. Nps: A non-interfering deployable web prefetching system. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Palo Alto, USA, 2003.

[22] Thomas M. Kroeger, Darrell D.E. Long, and Jeffrey C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems*, Monterey, USA, 1997.

[23] Tong Sau Loon and Vaduvur Bharghavan. Alleviating the latency reduction and bandwidth problems in www browsing. In *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems*, Monterey, USA, 1997.

[24] Evangelos Markatos and Catherine Chronaki. A top-10 approach to prefetching on the web. In *Proceedings of the INET' 98*, Geneva, Switzerland, 1998.

[25] Alexandros Nanopoulos, Dimitrios Katsaros, and Yannis Manolopoulos. Effective prediction of web-user accesses: A data mining approach. In *Proceedings of the Third International Workshop WEBKDD – Mining Web Log Data Across All Customers Touch Points*, San Francisco, USA, 2001.

[26] VenkataÑ. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve World Wide Web latency. In *Proceedings of the ACM SIGCOMM '96 Conference*, Stanford University, USA, 1996.

[27] Themistoklis Palpanas and Alberto Mendelzon. Web prefetching using partial match prediction. In *Proceedings of the 4th International Web Caching Workshop*, San Diego, USA, 1999.

[28] James Pitkow and Peter 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] Michael Rabinovich and Oliver Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[30] Stuart Schechter, Murali Krishnan, and Michael D. Smith. Using path profiles to predict http requests. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[31] Qingping Tao. *Impact of Webspace Organization and User Interaction Behavior on a Prefetching Proxy*. PhD thesis, Kent State University, 2002.

[32] Ingrid Zukerman, David W. Albrecht, and Ann E. Nicholson. Predicting users' requests on the WWW. In *Proceedings of the 7th International Conference on User Modeling*, Banff, Canada, 1999.