Evaluación de políticas de reemplazo para una caché con distinción del tipo de contenido

F.J. González Cañete, E. Casilari, Alicia Triviño Cabrera
Departamento de Tecnología Electrónica. Universidad de Málaga
ETSI de Telecomunicación. Complejo Tecnológico. Campus de Teatinos.
29071 – Málaga
Teléfono: 952 13 71 76 Fax: 952 13 14 47
E-mail: fgc@uma.es

Resumen

En el presente artículo se ha realizado el estudio de una caché Web que únicamente recibe peticiones de objetos de tipo texto desde un switch de nivel 7 (L7). Este switch intercepta e interpreta las peticiones HTTP y puede ser configurado para que distribuya las peticiones de diferentes tipos de objeto entre diferentes proxies. Se ha realizado un estudio de las características de una traza HTTP de gran tamaño en términos del tipo de contenido, también se ha repetido el estudio teniendo en cuenta solamente el contenido de tipo texto. Este estudio nos permitirá deducir cuáles son las características más importantes para una caché de objetos de texto. Finalmente, se comparan varias políticas de reemplazo bajo diferentes configuraciones para obtener la que mejores resultados ofrece.

1. Introducción

Desde que se creó el *World Wide Web* (en adelante Web) ha sido necesario optimizar los recursos de los que se disponía, básicamente el ancho de banda, para que los tiempos de respuesta que apreciaban los usuarios fuera lo menor posible. Además, mediante una adecuada optimización, se puede reducir también la carga que soportan los servidores Web, con lo que no se convierten en cuellos de botella. Una de estas formas de optimización es el uso de cachés Web.

La forma más simple de arquitectura usando un *proxy* caché Web es la que se muestra en la Fig. 1 [1].

En esta arquitectura los usuarios están conectados a Internet a través de un *proxy*, de forma que los objetos que soliciten a los servidores Web pasan a través del *proxy*, el cual almacena una copia de los objetos que pasan a través de él. Si algún usuario solicita un objeto que ya fue solicitado por otro usuario, el *proxy* se lo devuelve directamente sin pedirlo al servidor Web original, con lo que el tiempo que tarda el usuario en obtener el objeto se reduce debido a que el *proxy* se encuentra más cerca que el servidor Web. Por otro lado también se reduce la carga de peticiones que soporta el servidor y se ahorra ancho de banda.

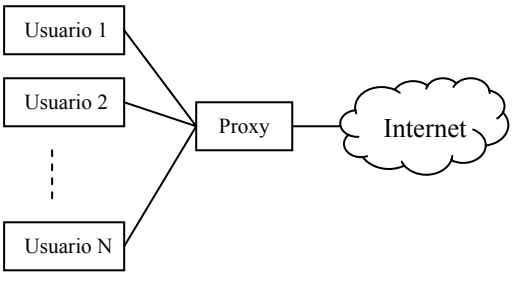

Fig. 1. Arquitectura de un *proxy* Web simple

Esta arquitectura tiene el problema de que todos los objetos son susceptibles de ser almacenados en la caché del *proxy*, es decir, se estarán guardando tanto imágenes como documentos HTML o poscript. Sin embargo, no todos los tipos de ficheros comparten las mismas características, tanto en tamaño como en repetitividad o popularidad. Para solucionar este problema se propone la arquitectura de la Fig. 2.

Con la arquitectura de la Fig. 2, los usuarios están conectados a Internet a través de un switch de nivel 7 (L7), que es un dispositivo de red que es capaz de diferenciar el tráfico de nivel 7 de la capa OSI y redireccionarlo en función de, entre otras características, el tipo de objeto. Esta diferenciación se realiza en función de la cabecera *content-type* de HTTP.

Con esta arquitectura se pueden usar diferentes *proxy* de forma que cada uno de ellos se ocupe de servir un tipo de objeto diferente (Imágenes, Texto, Video, Audio ,…) o bien usar un solo *proxy* que sea capaz de almacenar los objetos en colas diferentes en función de su tipo.

La siguiente elección a tomar es qué política de reemplazo se deberá usar para cada tipo de objetos.

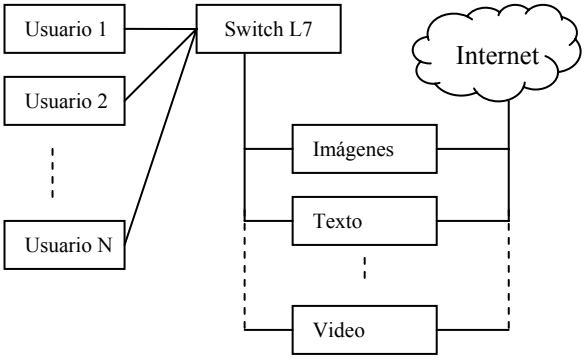

Fig. 2. Arquitectura usando un *switch* L7.

En el caso de que el espacio físico asignado para almacenar los objetos esté completo, la política de reemplazo es la encargada de decidir cuál o cuáles deben ser los objetos que deben eliminarse de la caché para hacer sitio y que entre uno nuevo. De entre las numerosas políticas de reemplazo que han sido propuestas, en este artículo se van evaluar las siguientes:

- LRU (*Least Recently Used*): Reemplaza el objeto que hace más tiempo que fue referenciado. Tiene la ventaja de su sencillez pero no tiene en cuenta la frecuencia de acceso a los documentos ni el tamaño de los mismos.

- LFU (*Least Frequently Used*): Reemplaza el objeto que menos veces haya sido referenciado, y en el caso de documentos con el mismo número de referencias, éstos se ordenan con política LRU. Tiene como inconveniente el hecho de que documentos que han sido muy referenciados pero que no vuelven a usarse, permanecen en la caché y no son reemplazados. Tampoco tiene en cuenta el tamaño de los documentos.

- LFU-DA (*LFU-Dynamic Aging*) [2]: Evita el problema de LFU considerando una variable con la edad de la caché. Cuando se inserta un nuevo documento o se referencia uno ya existente, se le añade el número de accesos al objeto menos referenciado presente en ese momento en la caché.

- GD-SIZE (*Greedy Dual-Size*) [3]: Usa una función de coste para valorar los objetos. En concreto la valoración (V(p)) de un objeto se estima mediante la Ec. 1:

$$V(p) = C(p)/S(p) \qquad \text{(Ec. 1)}$$

siendo $C(p)$ el coste de transmisión del objeto p y $S(p)$ el tamaño en bytes de p. Se reemplazaría de la caché aquel objeto con la menor valoración. Como funciones de coste se puede seleccionar la Ec. 2, con lo que sólo se tendría en cuenta el tamaño del objeto o la Ec. 3, con lo que se usaría como coste el número de paquetes que ocupa el objeto. Se ha modificado el tamaño de la MTU de 536 bytes del algoritmo original por 1460 bytes, que es la MTU más habitual en la actualidad. No tiene en cuenta la frecuencia de las referencias.

$$C(p) = 1 \qquad \text{(Ec. 2)}$$
$$C(p) = 2+S(p)/1460 \qquad \text{(Ec. 3)}$$

- GDSF (*Greedy-Dual Size with Frequency*) [4]: Es una modificación del algoritmo Greedy-Dual-Size para tener en cuenta la frecuencia de acceso de los objetos. La función de coste se muestra en Ec. 4:

$$V(p) = F(p) \cdot C(p)/S(p) \qquad \text{(Ec. 4)}$$

siendo F(p) el número de accesos al documento p, C(p) el coste y S(p) su tamaño. Como funciones de coste se han usado las ecuaciones 2 y 3.

- GD* (*Greedy-Dual**) [5]: También es una modificación de Greedy-Dual-Size que tiene en cuenta la frecuencia de acceso a los documentos y la correlación temporal de referencias. Este algoritmo ecualiza la popularidad para no ocultar la correlación temporal de referencias usando un parámetro β en su función de coste (Ec. 5):

$$V(p) = (F(p) \cdot C(p)/S(p))^{1/\beta} \qquad \text{(Ec. 5)}$$

siendo el significado de cada uno de los parámetros iguales a los de GDSF, a excepción del parámetro β, que es un número entre cero y uno característico de cada muestra de tráfico.

En el presente artículo se estudia cuál es el algoritmo de reemplazo que mejores resultados ofrece para una caché que únicamente almacene objetos de texto Web, comparando varias formas de optimizar el rendimiento de la misma. En el resto del artículo se evaluará cuál es la política de reemplazo con la que mejor rendimiento se obtiene si únicamente se almacenan objetos de tipo texto, como pueden ser HTML, XML, o Java-script. Para ello se va a realizar un estudio de las características que presentan estos tipos de objetos como sus tamaños, popularidad o repetitividad.

2. Procesado y caracterización de las muestras

Para la evaluación del rendimiento de una caché basada en texto se ha usado una muestra que contiene peticiones HTTP de un *proxy* del proyecto IRCache [5] situado en el *Research Triangle Park* en Carolina del Norte (EEUU) durante los días 7 al 11 de junio del 2004. Esta muestra contiene información de cada una de las peticiones HTTP que se han realizado a través del *proxy*, como la hora a la que se realizó la petición, el tamaño del objeto, la URL solicitada, el tipo de objeto (*content-type*), el método de petición HTTP (GET, POST, …) y el código de respuesta del servidor.

Esta muestra ha sido preprocesada para eliminar en primer lugar aquellas peticiones que han sido generadas dinámicamente mediante CGI (*Common Gateway Interface*), ya que los objetos que se devuelven en este tipo de peticiones son únicos para cada petición y, por lo tanto, no tiene sentido almacenarlos en caché [6]. Para ello se han descartado aquellas peticiones que contienen la cadena ‘cgi’, ‘cgi-bin’ o ‘?’. También se han filtrado aquellas peticiones que contienen la cadena ‘:3128’, ya que se corresponde con el puerto por el que las cachés del proyecto IRCache se comunican para colaborar entre ellas, al ser una arquitectura jerárquica. Como códigos de respuesta “cacheables” se han considerado el 200 (OK), 203 (*Partial*), 206 (*Partial Content*), 300 (*Multiple Choices*), 301 (*Moved*) y 302 (*Redirect*). Para el caso de las peticiones que tienen como código de respuesta el 304 (*Not Modified*), el tamaño de objeto que aparece en las trazas no se corresponde con el tamaño real del objeto, sino con el tamaño de la respuesta dada por el servidor para

notificar al usuario que el objeto que tiene en su caché local es la misma

Tabla 1. Características de la muestra estudiada

Peticiones	4.040.036
Tamaño (GB)	40.4
Objetos distintos	1.713.903

Tabla 2. Características por *content-type*

	Referencias (%)	Bytes (%)
Aplicaciones	8,08	33,51
Audio	0,28	3,49
Imágenes	75,54	36,33
Texto	15,77	23,15
Video	0,12	3,19

que está en el servidor. Estos objetos han vuelto a ser pedidos al servidor Web original para poder averiguar su tamaño real. También se han vuelto a solicitar aquellos objetos de los que se desconocía su *content-type*. En la tabla 1 se resumen las características de la muestra después de ser preprocesada.

En la tabla 2 se muestra el porcentaje de referencias y de *bytes* de todos los tipos de objetos. Las imágenes son los objetos que más frecuentemente se solicitan, seguidos por el texto. Con respecto a la cantidad de tráfico generado, las imágenes y las aplicaciones, son las que más tráfico generan, seguido por el texto. De este análisis se aprecia que el texto es el segundo tipo de objeto más importante en la Web, después de las imágenes. Estos resultados son coherentes con los obtenidos en [8].

Para el tipo de objeto texto existen diferentes subtipos. La cantidad de cada uno de ellos y su influencia en el tráfico generado ha sido representada en la Tabla 3. En esta tabla puede apreciarse que la mayoría de las referencias y el tráfico generado se corresponde con las páginas HTML, seguidas por el texto plano y las hojas de estilo. Finalmente, el XML y los códigos JavaScript asociados a páginas Web como ficheros independientes influyen menos en el tráfico.

2.1. Tamaño de los objetos

El tamaño de los objetos y su variabilidad es uno de los parámetros que influyen en la efectividad de una caché. Una caché funcionará mejor cuanto más homogéneos sean los tamaños de los objetos que se albergan en ella, esto se debe a que cuando se produzca un reemplazo, se eliminarán objetos de tamaño similar y no se sustituirán muchos objetos pequeños por uno muy grande.

En la Fig 3 se ha representado la función de distribución de los tamaños tanto de todos los objetos de texto en conjunto como de los HTML, texto plano (PLAIN) y hojas de estilo (CSS) por separado. En esta figura se puede verificar que los objetos HTML (media: 16.622 bytes - mediana: 5.545) son mayores que los objetos de texto plano (media: 17.926 – mediana: 674) y las hojas de estilo (media: 5.539 – mediana: 3394)

Tabla 3. Características del texto

	Referencias (%)	Bytes (%)
html	56,67	64,14
plain	22,05	26,92
css	17,43	6,57
xml	2,69	1,67
javascript	0,49	0,23

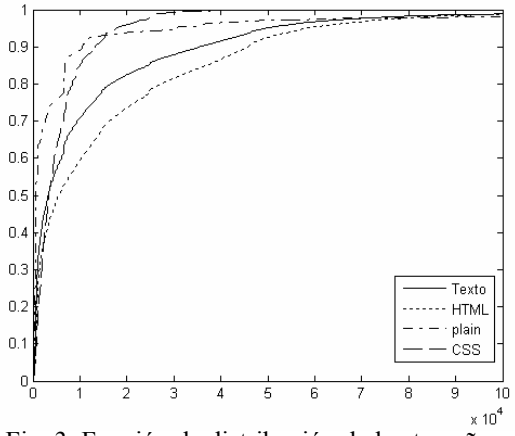

Fig. 3. Función de distribución de los tamaños

La variabilidad en los tamaños de los tipos de texto es muy alta y mayor que la media (desviación típica de 37.454 bytes para el HTML, 214.153 bytes para texto plano y de 6265 bytes para hojas de estilo), lo que puede sugerir un posible comportamiento *heavy-tailed* en los tamaños de los objetos. Para comprobar este posible comportamiento en la Fig 4 se ha representado la función de distribución complementaria en escala logarítmica para buscar un comportamiento lineal en la misma [9]. Efectivamente, para todos los tipos de texto en conjunto se aprecia comportamiento *heavy-tailed* aproximadamente desde 10^2 hasta $5*10^7$ bytes, desde 10^2 hasta $7*10^7$ bytes para el texto plano y desde 10^2 hasta 10^6 bytes para el HTML. Las hojas de estilo no tienen una variabilidad elevada, con lo que no presentan este tipo de comportamiento.

2.2. Popularidad

Otro aspecto influyente en la efectividad de una caché es la repetitividad de los documentos que maneja. Si un objeto es solicitado numerosas veces, será más probable que ya se encuentren en la caché cada vez que sea solicitado, con lo que se conseguirá un acierto en la caché.

En la Fig. 5 se ha representado en escala logarítmica el número de referencias del texto frente al ranking ordenado de referencias, es decir, primero el objeto más referenciado, después el segundo y así sucesivamente. Mediante esta representación se puede comprobar si los objetos de texto cumplen con la ley Zipf [10] [11], que establece que la probabilidad de que un objeto sea referenciado es inversamente proporcional a su popularidad (Ec. 5):

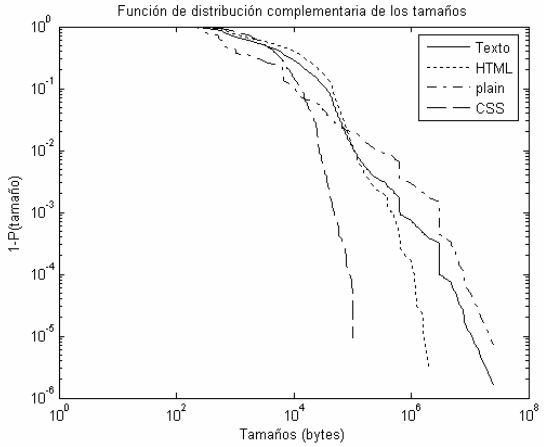

Fig. 4. Función de distribución complementaria de los tamaños

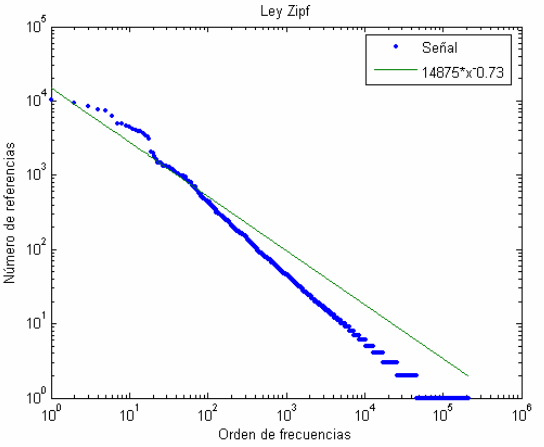

Fig. 5. Ley Zipf

$$P(i) = \frac{1}{i^{\alpha}} \quad \text{con } \alpha \text{ cercano a 1} \qquad \text{(Ec. 5)}$$

En la figura también se ha aproximado mediante una regresión de mínimos cuadrados la señal generada para obtener el valor de $\alpha = 0.73$.

2.3. Relación entre tamaño y repetitividad

Para capturar la posible relación que existe entre la popularidad y el tamaño de los objetos, se ha representado en la Fig. 6 la relación entre ambas para todos los objetos de texto.

En la figura se percibe que ambas características están correladas, cuanto menor es el tamaño de los documentos, mayor es el número de referencias que tienen. Esta característica puede favorecer el rendimiento en políticas de reemplazo que tengan en cuenta el tamaño de los documentos y que favorezcan a aquellos documentos que son más pequeños como ocurre con GD-SIZE, GDSF y GD*.

3. Comparativas de eficiencia

Como medida de la eficiencia de la caché se han usado las siguientes métricas:

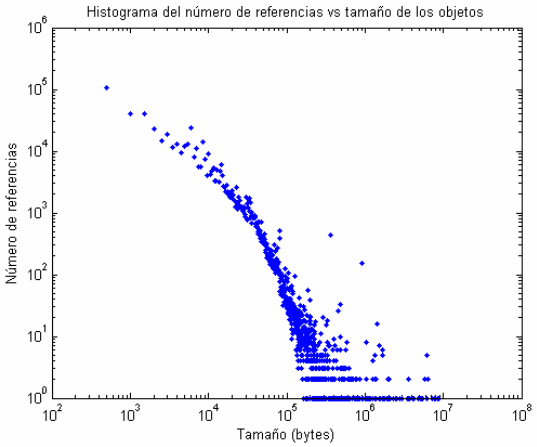

Fig. 6. Relación entre el número de referencias y el tamaño de los objetos

- *Hit Rate* (HR), definido como el cociente entre el número de aciertos en la caché y el número total de referencias.
- *Byte Hit Rate* (BHR), definido como el cociente entre las suma de los tamaños de los objetos que han producido un acierto en la caché y el tamaño total de todos los objetos.

Para distinguir entre la modificación de un objeto y la interrupción de la transferencia por parte del usuario, se comprueba si la diferencia de tamaño entre peticiones consecutivas a un mismo es menor del 5%. Si se cumple esta condición, se considera que el documento ha sido modificado, con lo que no se considera como un acierto, y en caso contrario se considera una cancelación [12].

Para calcular el parámetro β de la política de reemplazo GD*, en la Fig. 7 se ha representado en escala logarítmica la distribución de probabilidad de las distancias entre referencias de objetos con la misma frecuencia de acceso [5] y se ha aproximado mediante una regresión de mínimos cuadrados, con lo que se ha obtenido un valor de $\beta = 0.54$ según la Ec. 6 para objetos con igual frecuencia de acceso k=8 (con otros valores de k se obtienen valores similares).

$$T \propto \frac{1}{t^{\beta}} \qquad \text{para frecuencia k} \qquad \text{(Ec. 6)}$$

En un primer experimento se ha evaluado la eficiencia de cada uno de las políticas de reemplazo si se cachean todos los objetos de tipo texto. En las Fig. 8 y 9, se ha representado el HR y BHR en función del porcentaje del tamaño de la caché, siendo el 100% el tamaño total agregado de todos los objetos de texto distintos de la traza (3 GB), es decir, para el caso ideal en el que no hay que realizar ningún reemplazo debido a que hay suficiente espacio para almacenar todos los documentos. Para esta situación ideal el HR que se obtiene es del 47% y el BHR es del 45.3%.

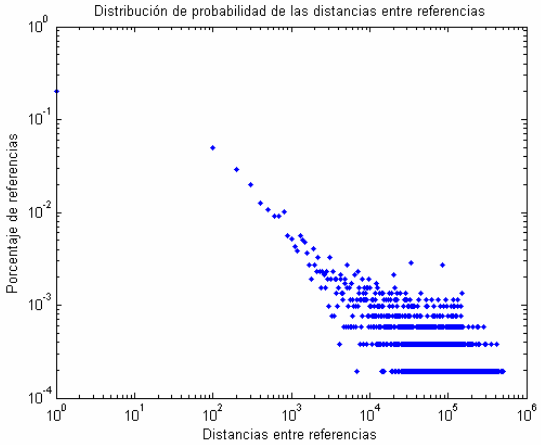

Fig. 7. Distribución de probabilidad de las distancias entre referencias

Con respecto al HR, las políticas que no consideran el tamaño de los objetos son las que peor resultado ofrecen (LRU, LFU y LFU-DA), ya que no aprovechan la relación entre el tamaño de los objetos y su repetitividad. La política GD-SIZE ofrece unos resultados ligeramente mejores, aunque los algoritmos más eficientes son GDSF y GD*. Como es de esperar se obtienen mejores resultados cuando se aplica una función de coste c(p)=1 con respecto a los obtenidos con la función de coste c(p)=packets, ya que lo que se pretende maximizar el número de aciertos en caché. Por tanto, las poíticas de reemplazo más eficientes para maximizar la tasa de acierto en caché y, por tanto, las más recomendables son GDSF(1) y GD*(1).

Para el BHR, las políticas sin función de coste tienen un comportamiento muy dispar. Mientras LRU y LFU obtienen los peores rendimientos, LFU-DA es la que mayor eficiencia obtiene, al igual que GDSF(packets), sobretodo para un tamaño de caché pequeña, que es donde mayor tasa de acierto presenta. Se da el caso de que la eficiencia para estas dos políticas de reemplazo es mejor para una caché del 2% que del 5%, y del orden del obtenido con una caché del 10% del tamaño total. Esto puede ser debido a que con un tamaño del 5% algún objeto grande ha sido reemplazado por otro, situación que no se ha dado para la caché del 2%, con lo que se ha conseguido una mayor tasa de acierto. El resto de políticas de reemplazo presentan un comportamiento peor, sobretodo las políticas con función de coste c(p)=1, resultando mejores las de coste c(p)=packets. Resumiendo, las mejores políticas de reemplazo para maximizar el BHR son LFU-DA y GDSF(packets) ya que mejoran en más de un 4% al la mejor del resto de políticas que es GD*(packets).

4. Conclusiones

En el presente artículo se han obtenido a partir de unas muestras de tráfico de un *proxy* las principales características que tienen los objetos de texto que se encuentran en Internet, obteniéndose aquellos tipos que son los más frecuentes y que influyen de manera más notable en el tráfico.

Se ha caracterizado el tamaño de estos objetos deduciéndose que tienen un comportamiento *heavy-tailed*, tanto en general como en cada uno de los tipos más frecuentes a excepción de las hojas de estilo, que presentan poca variabilidad de su tamaño.

Se ha estudiado la popularidad de los objetos de tipo texto encontrándose que se ajusta a una ley exponencial de pendiente $\alpha = 0.73$, lo que favorece la eficiencia de una caché basada en frecuencias de acceso. También se ha encontrado que existe una clara correlación entre el tamaño de los objetos y su repetitividad, siendo los objetos más pequeños los que tiene más frecuencia de acceso, lo que favorece a las políticas de reemplazo que penalizan a los objetos más grandes.

Finalmente se ha evaluado mediante simulación la eficiencia de cada una de las políticas de reemplazo consideradas para las métricas de HR y BHR. Como mejores políticas de reemplazo para el HR se ha obtenido GDSF(1) y GD*(1). Para un mejor rendimiento del BHR se propone LFU-DA y GDSF(packets).

Como líneas futuras de investigación sugerimos separar los objetos de texto en varias colas, una para cada tipo de objeto entre los más accedidos, y evaluar las políticas de reemplazo para cada una de ellas por separado. Otra posible línea de investigación consistiría en aprovechar la correlación que existe entre el tamaño de los objetos y su repetitividad para restringir el tamaño de los documentos que entran en la caché.

Agradecimientos

Este trabajo ha sido parcialmente costeado por el proyecto de financiación pública Nº TEL2003-07953-C02-01.

Referencias

[1] A. Luotonen, K. Altis, "World-Wide WebProxies", *Proceedings First International Conference on the WWW*, 1994.

[2] M. Arlitt, C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", *IEEE/ACM Transactions on Networking*, Octubre 1997.

[3] P. Cao, "Cost-Aware WWW Proxy Caching Algorithms", *Proceedings USENIX Symposium on Internet Technologies and Systems,* Monterey, California, Diciembre, 1997.

[4] L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size Frequency Caching Policy", *Technical Report HP Labs HPL-98-69*, Noviembre, 1998.

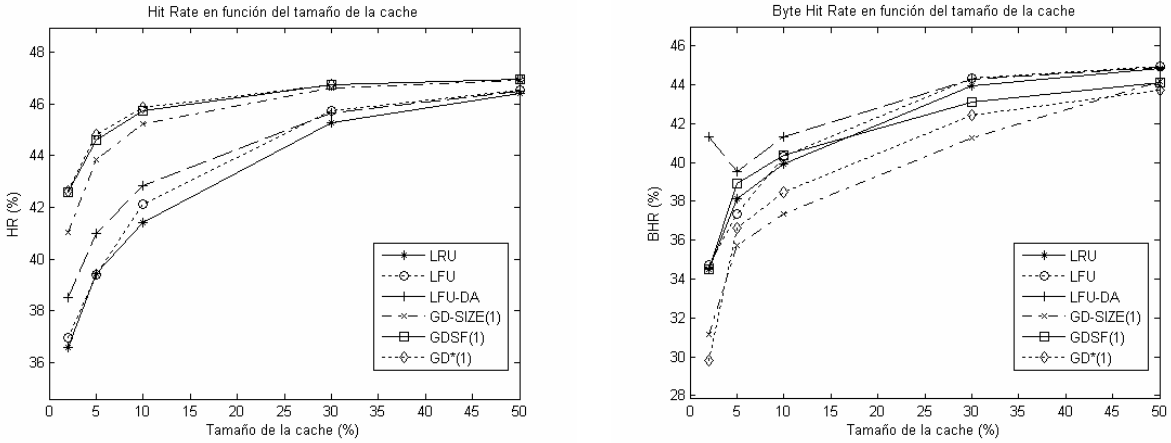

Fig. 8. HR y BHR para las políticas LRU, LFU, LFU-DA, GD-SIZE(1), GDSF(1) y GD*(1)

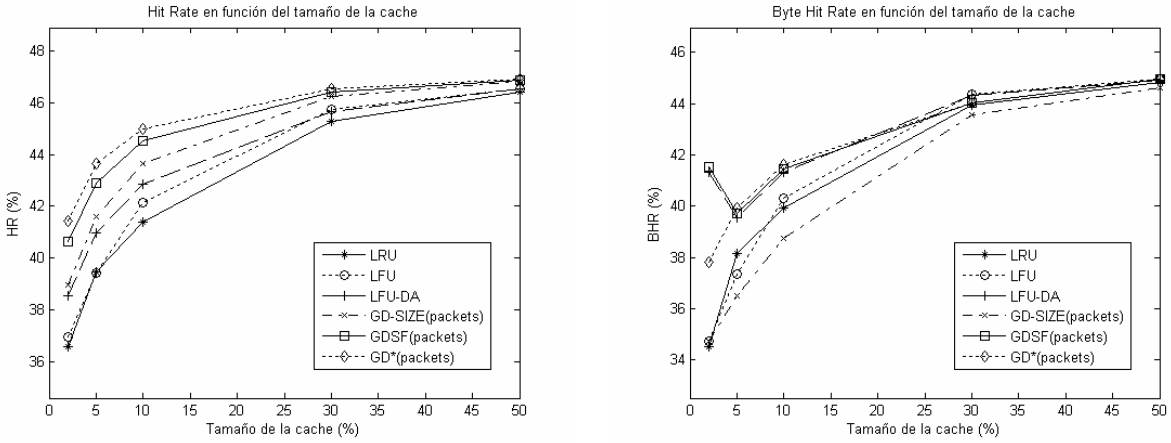

Fig. 9. HR y BHR para las políticas LRU, LFU, LFU-DA, GD-SIZE(packets), GDSF(packets) y GD*(packets)

[5] Jin, S., Bestabros, A., "GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams", Intl' Journal of Computer Communications, Vol. 24, No. 2, pp. 174-183, Febrero, 2001.

[6] Página del proyecto IRCaché con las muestras: http://www.ircaché.net/

[7] X. Zhang, "Cachability of Web Objects", Technical Report 2000-19, Agosto, 2000.

[8] M. Arlitt, C. Williamson, "Web server workload characterization: The search for invariants", *Proceedings of the ACM SIGMETRICS'96 Conference*, Philadelphia, PA, USA, Junio 1998.

[9] M.E. Crovella, A. Bestavros, "Self Similarity in the World Wide Web Traffic: Evidence and Possible Causes", *Proceedings IEEE/ACM Transactions on Networking*, Vol. 5, Nº 6, pp. 835-846, Diciembre 1997.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips, "On the Implications of Zipf's Law for Web Caching", *3rd International WWW Caching Workshop*, Junio, 1998.

[11] L. Breslau, P. Cao, P. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", IEEE Infocom, Vol XX, 1999.

[12] C. Lindemann, O.P. Waldhorst, "Evaluating the Impact of Different Documents Types on the Performance of Web Caché Replacement Schemes", *IEEE International Conference on Dependable Systems and Networks (DSN'02)*, Washington, Junio, 2002.