Quesada Sánchez, Carlos E.; Meneses, Esteban.
Políticas de reemplazo en la caché de web.
**Tecnología en Marcha.** Vol. 19-4 - 2006

# Políticas de reemplazo en la caché de web

*Carlos E. Quesada Sánchez*[1]
*Esteban Meneses*[2]

*La web es el mecanismo de comunicación más utilizado en la actualidad debido a su flexibilidad y a la oferta casi interminable de herramientas para navegarla. Esto hace que día con día se agreguen alrededor de un millón de páginas en ella [3]. De esta manera, es entonces la biblioteca más grande, con recursos textuales y de multimedia, que jamás se haya visto antes. Eso sí, es una biblioteca distribuida alrededor de todos los servidores que contienen esa información. Como fuente de consulta, es importante que la recuperación de los datos sea eficiente.*

## Palabras clave

*Web Caching*, *Web Proxy*, Políticas de Reemplazo, *Squid Web Proxy Cache*.

## Resumen

La web es el mecanismo de comunicación más utilizado en la actualidad debido a su flexibilidad y a la oferta casi interminable de herramientas para navegarla. Esto hace que día con día se agreguen alrededor de un millón de páginas en ella [3]. De esta manera, es entonces la biblioteca más grande, con recursos textuales y de multimedia, que jamás se haya visto antes. Eso sí, es una biblioteca distribuida alrededor de todos los servidores que contienen esa información. Como fuente de consulta, es importante que la recuperación de los datos sea eficiente. Para ello existe el Web Caching, técnica mediante la cual se almacenan temporalmente algunos datos de la web en los servidores locales, de manera que no haya que pedirlos al servidor remoto cada vez que un usuario los solicita. Empero, la cantidad de memoria disponible en los servidores locales para almacenar esa información es limitada: hay que decidir cuáles objetos de la web se almacenan y cuáles no. Esto da pie a varias políticas de reemplazo que se explorarán en este artículo. Mediante un experimento de peticiones reales de la Web, compararemos el desempeño de estas técnicas.

## Introducción

Para muchos, la web es el sistema de información más grande y dinámico que existe [5], para otros es la biblioteca de Alejandría de nuestro tiempo [2], pero quizás para la mayoría de usuarios sea solamente una herramienta útil en la búsqueda de información. Claro está, la utilidad también reside en la eficiencia con que los usuarios puedan obtener los recursos que andan buscando.

En la web existe un gran número de páginas que son visitadas desde todas partes del mundo, en ocasiones una misma página es visitada desde diferentes terminales dentro de una misma red u organización,

1 Escuela de Ingeniería en Computación del Instituto Tecnológico de Costa Rica.
Correo electrónico: *caquesada@ic-itcr.ac.cr*

2 Centro de Investigaciones en Computación del Instituto Tecnológico de Costa Rica.
Correo electrónico: *emeneses@ic-itcr.ac.cr*

provocando así que la misma solicitud se realice varias veces hasta el servidor origen, desperdiciando el ancho de banda con el que se cuenta. Este fenómeno es muy común, ya que dentro de la misma red las personas comparten intereses comunes y por tanto visitan páginas similares.

El *Web Caching* es una de las técnicas claves para mejorar la utilización de la web. Algunos productos de esta técnica están en la reducción de la carga de los servidores orígenes, la minimización del tráfico de la red y la disminución de la latencia que experimenta el usuario.

Los *cachés* de web actúan como intermediarios entre los clientes de la red y los servidores orígenes. Entonces, si un *browser* de la web necesita algún objeto, debe enviar una petición al servidor origen. De acuerdo con algunas condiciones, el servidor origen envía el objeto solicitado al cliente. Sin embargo, si un *caché* de web está presente, puede atrapar este objeto de tal forma que la siguiente vez que el objeto es requerido pueda ser recuperado localmente, sin necesidad de hacer una petición externa. La mayoría de las ocasiones este *caché* de web es implementado como un servidor *proxy*, que actúa como un intermediario en las peticiones web.

*Los cachés de web actúan como intermediarios entre los clientes de la red y los servidores orígenes. Entonces, si un browser de la web necesita algún objeto, debe enviar una petición al servidor origen.*

Sin embargo, la cantidad de memoria disponible por los *cachés* de web está limitada por el espacio en disco duro que puedan dedicar a almacenar información de la web. La buena administración de ese espacio depende de la política de reemplazo que se utilice. Este artículo explora los resultados de aplicar distintas políticas de reemplazo en un *caché* de web cuando se ejecutan peticiones reales de los usuarios.

El artículo está organizado de la siguiente manera. La sección 2 presenta una introducción al tema del *Web Caching*, mientras que la sección 3 trata el tema de las políticas de reemplazo. Los resultados son mostrados en la sección 4, donde se descubrirá qué técnica ofrece el mejor desempeño cuando se contrasten los diferentes modelos. Luego, la sección 5 presentará una discusión de los resultados y las conclusiones finales del artículo.

## Web Caching

El *Web Caching* es un método utilizado en Internet para acelerar la velocidad en que se obtienen las páginas web [4] ¡Error! No se encuentra el origen de la referencia., principalmente. El *Web Caching* se basa en el almacenamiento temporal de objetos web, normalmente páginas HTML, que son usados como las futuras respuestas de nuevas solicitudes durante la vida útil del objeto [7]. La figura 1 muestra un esquema del modelo.

El principal objetivo es mantener una copia temporal de algún objeto para evitar que la próxima vez que ese mismo objeto sea solicitado, este sea adelantado hasta el servidor que contiene el objeto; de esta manera, se reduce el tráfico y se aumenta la velocidad de respuesta al cliente.

En el ámbito de la web existen niveles donde un *caché* puede ser colocado; el primero de ellos se encuentra en la máquina del cliente, específicamente en el navegador de Internet, los cuales comúnmente proveen esta funcionalidad. Un segundo nivel de *caché* web es en la red local, en la cual el cliente es usuario. El tercer nivel de un *caché* se encuentra en un lugar mayor jerárquicamente hablando, ya que puede ser en el nivel organizacional o puede encontrarse en alguna clase de división regional. Los anteriores niveles de caché son los conocidos como del lado del cliente (*client-side* en inglés) ya que están directamente interconectados con el cliente que realiza las solicitudes. El cuarto nivel de *Web Caching* también llamado “acelerador web”, se encuentra del lado del servidor web (*server-side*); a este nivel se le conoce de esta forma

ya que evita que el servidor se encuentre ocupado, atendiendo solicitudes cuyos cambios son muy pocos, que con el tiempo estos serán mínimos, y dando espacio para que el servidor pueda realizar operaciones de mayor complejidad e importancia.

Las ventajas del *Web Caching* se pueden resumir en las siguientes:

- Decrece el consumo de ancho de banda, de manera que el tráfico de la red decrementa y minimiza la congestión de la red.
- Reduce la latencia de acceso debido a dos factores: los objetos frecuentemente accedidos son buscados en un *proxy* cercano y los objetos que no están en el *caché* se buscan en una red relativamente más rápida.
- Reduce el trabajo de carga de los servidores remotos.
- Si un servidor remoto no está disponible, una copia del objeto pedido puede ser recuperado del *proxy* local.

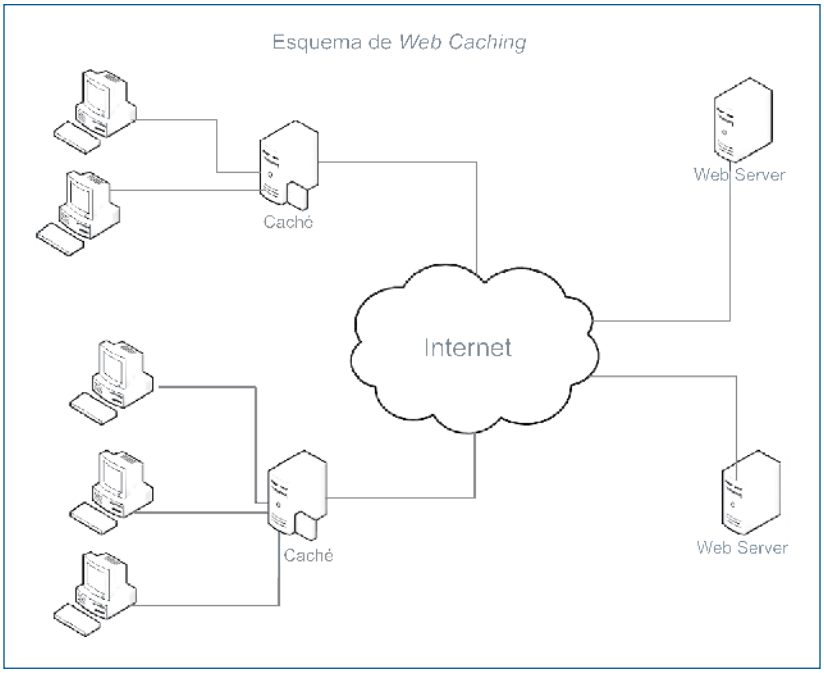


*Figura 1.* Esquema de *Web Caching*

- Provee una opción de monitorear el patrón de acceso a la web.

## Squid

*Squid* es un programa que hace caché de datos obtenidos en Internet [6][10]. Realiza este trabajo al aceptar peticiones de los objetos que los usuarios quieren descargar y realizar estas peticiones a la red en su nombre. *Squid* se conecta con el servidor correspondiente y pide el objeto solicitado por el usuario. Luego, de forma transparente, este objeto se entrega a la máquina cliente, pero al mismo tiempo, guarda una copia. La próxima vez que alguna máquina cliente de *Squid* solicite la misma página, *Squid* simplemente le transfiere su copia almacenada en memoria o disco, acelerando considerablemente la transferencia y ahorrando ancho de banda en la conexión a Internet.

Actualmente, *Squid* es capaz de hacer *proxy-caché* de los protocolos HTTP, FTP, GOPHER, SSL y WAIS. *Squid* también soporta SSL (*Secure Socket Layer*) con lo que también acelera las transacciones cifradas, y es capaz de ser configurado con amplios controles de acceso sobre las peticiones de usuarios. Al utilizar el protocolo de *caché* de Internet, *squid* puede ahorrar un considerable ancho de banda, mejorando la velocidad de acceso a Internet en estos protocolos. *Squid* es el resultado del esfuerzo de numerosas personas individuales de Internet y, al igual que el sistema operativo *Linux*, es gratis y tiene el código fuente disponible para poder modificarlo según las necesidades de cada persona o institución.

En la Escuela de Ingeniería de Computación en el Instituto Tecnológico de Costa Rica se utiliza el programa *Squid* como herramienta para mejorar el rendimiento del ancho de banda de red tanto interno como para el acceso a Intenet. En la escuela la configuración de *squid* es absolutamente transparente para los usuarios, ya que no es necesario "parametrizar" en los *browsers*

el uso del *proxy* porque este es detectado automáticamente; si se desea ver más acerca de esto, consultar [6].

## Políticas de reemplazo

Al hablar acerca de *web caching*, se mencionó que era un método de almacenamiento temporal. Al decir esto se deben responder muchas interrogantes: ¿dónde se almacenan los objetos?, ¿con cuál formato?, ¿cuál es el tamaño del almacenamiento? Pero aún más importante que estas preguntas anteriores, en el caso de las políticas de reemplazo sería: ¿cómo determinamos qué objetos entran y qué salen de la *caché* a la hora de responder una solicitud?

*La función principal de una política de reemplazo se basa en la forma de determinar cuál, objeto web será el siguiente en ser reemplazado o eliminado de la caché cuando se requiera.*

Una política de reemplazo existe en todas las formas de *cachés* que existen; este caso no es la excepción; para obtener un buen desempeño se debe afinar la *caché web* con alguna política de reemplazo que se ajuste a las necesidades del usuario final. Existen herramientas *web* llamadas *proxies*, que funcionan como un puente entre los clientes e Internet. Al utilizar una herramienta que puede ser configurada y afinada, es común para una aplicación *proxy* manejar soporte para *web caching*. Uno de los programas más utilizados para este propósito es *Squid* [2].

La función principal de una política de reemplazo se basa en la forma de determinar cuál, objeto web será el siguiente en ser reemplazado o eliminado de la *caché* cuando se requiera. Las políticas de reemplazo utilizan diferentes propiedades de los objetos web para determinar cuál es el más apto para ser eliminado; algunas de las propiedades más utilizadas son: edad (tiempo que ha permanecido en la *caché*), tamaño del objeto, cantidad de solicitudes, tipo de objeto, entre otros.

El principal objetivo al utilizar una *caché* es el minimizar dos parámetros: la latencia que tiene un documento para ser presentado al usuario final y reducir el tráfico de la red hacia el Internet. El *hit rate* y el *byte hit rate* son dos medidas muy utilizadas que describen la efectividad de un *proxy*. El *hit rate* representa el porcentaje de solicitudes que fueron atendidas gracias a que en *caché* se encontraba una copia del objeto solicitado. El *byte hit rate* es el porcentaje total de datos que fueron transmitidos a partir de la *caché*.

Si se quiere disminuir el número de peticiones a los servidores remotos, se debe incrementar el *hit rate*, mientras que si se quiere disminuir el ancho de banda requerido para las peticiones, entonces se debe aumentar el *byte hit rate* [1].

Existe una tendencia del comportamiento de una *caché* y es que las medidas de *hit rate* tanto como el *byte hit rate* rondan un rango del 30% - 50%, según muchos estudios realizados [1].

A continuación se presentarán algunas de las políticas de reemplazo más conocidas en el ámbito del *web caching*:

- FIFO: este algoritmo significa, por sus siglas en inglés, que el primer objeto en entrar en la *caché* será al primer archivo en salir de la *caché*. Esta política toma como parámetro para eliminar un objeto la edad del objeto en la *caché*, mientras más tiempo tenga en la *caché*, mayor, es su probabilidad de ser eliminado. El reemplazo en esta política es un consecutivo lineal de los objetos web.
- RANDOM: realiza la función de reemplazo por medio de la selección aleatoria de un objeto, sin tomar en cuenta ninguna clase de parámetro ni propiedad del objeto que será eliminado en la *caché*. Esta política de reemplazo es estudiada por motivos académicos, pero, en realidad, su desempeño es bajo en comparación con otras políticas.

- LRU: la política de reemplazo LRU busca un mejor rendimiento de la *caché*, utilizando como parámetro de reemplazo el tiempo que ha permanecido un objeto sin haber sido solicitado de nuevo por algún cliente. Es decir, el archivo que menos ha sido utilizado recientemente es el candidato por ser reemplazo de la *caché*.
- LRU-MIN: es una variación que existe sobre la política de reemplazo LRU, donde lo que se busca que es los objetos que se mantendrán en la *caché* serán aquellos cuyo tamaño sea menor. De esta forma, son reemplazados objetos de tamaño mayor, liberando así más espacio en la *caché* para nuevos documentos. La teoría en la cual se basa este algoritmo para su selección es que a la hora de llegar un nuevo objeto cuyo tamaño es S, al iniciar, son removidos de la *caché* aquellos objetos cuyo tamaño sea menor que S; en caso de que esto no cumpla se divide en tamaño de S entre 2 y se itera sobre esto.

*Otra forma de implementación que ha sido muy utilizada en cuanto a las políticas de reemplazo es el usar un árbol de Heap como estructura de datos, para almacenar los objetos web; así, es por medio de una llave que es generada para cada objeto se selecciona el mejor candidato para ser reemplazado de la cache.*

Un detalle de las políticas de reemplazo que ya han sido mencionadas es que su implementación se realiza, en la mayoría de los casos, por medio de una lista enlazada, donde cada nodo es un objeto que existe en la *caché* en un determinado momento. Una manera sencilla para seleccionar el siguiente objeto para ser eliminado de la *caché* es manteniendo aquel que resulta ser el mejor candidato en uno de los extremos de la lista.

Otra forma de implementación que ha sido muy utilizada en cuanto a las políticas de reemplazo es el usar un árbol de Heap como estructura de datos, para almacenar los objetos web; así, es por medio de una llave que es generada para cada objeto se selecciona el mejor candidato para ser reemplazado de la cache. Lo que cambia en las siguientes políticas de reemplazo es la forma en que se genera la llave para cada objeto web.

- *Heap-LRU:* esta política de reemplazo utiliza el algoritmo LRU, con la diferencia de su implementación por medio de un árbol de Heap. El valor que se usa para la llave de un objeto es el tiempo desde el último acceso al objeto, eliminando así el menos empleado recientemente.
- *HEAP-LFUDA:* la política de reemplazo es una variante de LFU (Least Frecuently Used) la cual lleva un contador por cada documento dentro de la *caché*, lo que mantiene los documentos con mayor cantidad de accesos en *caché*. La variante utiliza una técnica de envejecimiento dinámico y agrega un factor de edad para evitar que ciertos documentos permanezcan en *caché* por tiempo indefinido ya que este es uno de los problemas que tiene por sí sola la política de LFU.
- *HEAP-GDSF:* el algoritmo de GDSF (Greedy-Dual Size Frequency) asigna una prioridad a aquellos documentos basándose en el costo de que el documento sea solicitado y este haya sido removido de la *caché* y también toma en cuenta el tamaño del documento en sí. Aquellos documentos que tengan la menor prioridad serán los primeros en ser removidos.

Para el ambiente de pruebas que se definió, fue necesario agregar ciertas políticas de reemplazo al *software* del Web Proxy Cache Squid, Versión 3.0, ya que este en su versión descargable desde Internet trae consigo solamente cuatro políticas de reemplazo, entre ellas LRU, Heap-LRU, Heap-LFUDA, Heap-GDSF. Por esta razón, fue necesario agregar las políticas de FIFO, Random y LRU-MIN. La versión descargable de estas políticas de reemplazo se encuentran en la página oficial del Proyecto SPREAD [10].

La política de reemplazo es responsable por determinar el orden en que los

documentos serán eliminados de la *caché* cuando *Squid* requiera de espacio para objetos nuevos. El *Proxy Squid* provee un API para que un usuario pueda agregar una nueva política de reemplazo, extendiendo así la funcionalidad del *Proxy*. El API que provee *Squid* está implementado de manera modular y permite al programador realizar las funciones básicas para las políticas de reemplazo; entre algunas de ellas están: agregar un objeto a la *caché*; eliminar un objeto de caché; "referenciar" un objeto; "de-referenciar" un objeto; iniciar un observador de los elementos; purgar un objeto; obtener el siguiente objeto para ser eliminado de la *caché*, entre otras funciones. Para mayor información, consultar [11]. Para hacer uso de esta interfaz, es necesario registrar la política de reemplazo que se está implementando en la configuración de *Squid* y recopilarlo.

*Se realizaron diferentes series de experimentos para poner en prueba el rendimiento de las diferentes políticas de reemplazo que han sido estudiadas en este artículo.*

## Experimentos

Se realizaron diferentes series de experimentos para poner en prueba el rendimiento de las diferentes políticas de reemplazo que han sido estudiadas en este artículo. Lo primero fue preparar un ambiente de prueba donde se ejecutarían cada una de las políticas y se tomarían los datos de cada una.

El ambiente de prueba utilizado posee las siguientes características:

- Red de Ingeniería en Computación del Instituto Tecnológico de Costa Rica
- Centro de Investigación en Computación
- Computadora Procesador Intel Pentium II 266, 128 MB de Ram, 4 GB de RAM, Sistema Operativo Debian Linux.
- Web Proxy Cache Squid, Versión 3.0

Para la realización de las pruebas, fue necesario tomar un conjunto de referencias de todo tipo de objetos web; para esto se tomó como base una bitácora de todas la direcciones web que fueron referenciadas por el Proxy Squid de la Escuela de Computación, en el período de marzo del 2005 hasta abril del 2005.

El experimento consistió en la simulación de la bitácora que se mencionó anteriormente, para cada una de las políticas de reemplazo; después de cada una de estas simulaciones, se recolectaron los datos para realizar análisis sobre los resultados más adelante. Al utilizar las direcciones de esta bitácora, se puede asegurar que las solicitudes que se realizarán para el experimento fueron en algún momento solicitudes reales que fueron hechas por uno o más usuarios de la red de Computación.

A continuación se presentarán algunas estadísticas importantes acerca de la bitácora que fue utilizada para los experimentos.

- Cantidad total de *urls* solicitados aproximadamente por prueba: 100 000
- Cantidad de *urls* diferentes solicitados: 27 616
- Aproximado de *bits* descargados desde Internet por prueba: 650 MB

### Experimento A

El primer experimento consistió en simular todas las peticiones contenidas en el *log* de la Escuela de Ingeniería en Computación, usando una *caché* de 50 MB en el servidor de *Squid*. Este experimento se repitió para cada una de las 7 políticas de reemplazo distintas que se habían incluido en el código de *Squid*.

El cuadro 2 muestra los resultados de estos experimentos. Para cada una de las políticas de reemplazo, se midió el número total de solicitudes, el total de *bits* transmitidos, el *hit rate*, el *byte hit rate* y el tiempo total requerido.

Como se puede notar, el número de peticiones no siempre es igual. Esto se debe a que muchas peticiones no lograron ser

Sitio más visitados:

*Cuadro 1.* Sitios más visitados

| Posición | Url | Cantidad de solicitudes | Posición | Url | Cantidad de solicitudes |
|---|---|---|---|---|---|
| 1. | www.msn.es | 8739 | 11. | img.blogdrive.com | 1216 |
| 2. | www.costarricense.cr | 6456 | 12. | www.navegalo.com | 980 |
| 3. | a1568.g.akamai.net | 4399 | 13. | www.nacion.com | 916 |
| 4. | origin.gfx2.hotmail.com. | 3687 | 14. | global.msads.net | 824 |
| 5. | us.i1.yimg.com | 2916 | 15. | latam.msn.com | 812 |
| 6. | sc.webmessenger.msn.com | 2781 | 16. | at.multimap.com | 760 |
| 7. | graphics.hotmail.com | 2035 | 17. | www.itcr.ac.cr | 746 |
| 8. | www.google.co.cr | 1489 | 18. | tcontent.e-messenger.net | 727 |
| 9. | gmail.google.com | 1396 | 19. | www03.quizyourfriends.com | 723 |
| 10. | vcontent.e-messenger.net | 1377 | 20. | java.sun.com | 688 |

*El tiempo tomado por cada una de las políticas varía de estrategia en estrategia y debería estar correlacionado con el byte hit rate.*

concretadas por problemas de estabilidad en la red. Sin embargo, el porcentaje de peticiones perdidas no supera el 1%. Esto tiene una incidencia también en el número de *bits* transmitidos.

La estrategia LRU-MIN logró obtener el mayor *hit rate* con un 57,14%, seguido por la estrategia GDSF y después LFUDA. La estrategia RANDOM obtuvo el peor *hit rate*, lo cual es de esperar debido a que no hay ningún cálculo inteligente en el reemplazo que propone esta política.

Empero, LRU-MIN no fue la estrategia con el mejor *byte hit rate*, sino que lo fue LFUDA con un 26,97%, seguido por HEAP LRU y luego LRU. Este resultado es interesante, porque LFUDA no tuvo tantos aciertos como LRU-MIN, pero los aciertos que tuvo fueron de objetos más grandes, lo que redunda en una mejoría del *byte hit rate.*

Según lo que se discutió en la sección 2, la política de reemplazo LFUDA mejora el ancho de banda requerido para todas las peticiones web, mientras que LRU-MIN decrementa el número de peticiones que deben salir de la red local para poderse satisfacer.

El tiempo tomado por cada una de las políticas varía de estrategia en estrategia y debería estar correlacionado con el *byte hit rate*. Sin embargo, estos experimentos fueron hechos en diferentes momentos de saturación de la red, lo que no permite hacer una justa comparación entre las estrategias utilizando esta variable.

## Experimento B

El segundo experimento consistió en simular todos las peticiones en el *log* de la Escuela de Computación, utilizando solamente la política de reemplazo LFUDA, ya que es considerada como una, sino la mejor, política de reemplazo en cuanto a *caché*.

*Cuadro 2.* Resultados de políticas de Reemplazo (50 MB de caché)

| | Total solicitudes | Total bits | *Hit rate* | *Byte hit rate* | Tiempo (horas) |
|---|---|---|---|---|---|
| FIFO | 104248 | 654984K | 45,04% | 21,51% | 63,81 |
| RANDOM | 103850 | 647058K | 33,11% | 18,04% | 46,23 |
| LRU | 103612 | 649632K | 49,57% | 25,50% | 47,09 |
| GDSF | 103331 | 601770K | 51,37% | 23,63% | 47,65 |
| LFUDA | 103805 | 653584K | 50,98% | 26,97% | 46,18 |
| HEAP LRU | 103537 | 652810K | 48,31% | 26,41% | 37,81 |
| LRU-MIN | 103687 | 656665K | 57,14% | 22,74% | 40,00 |

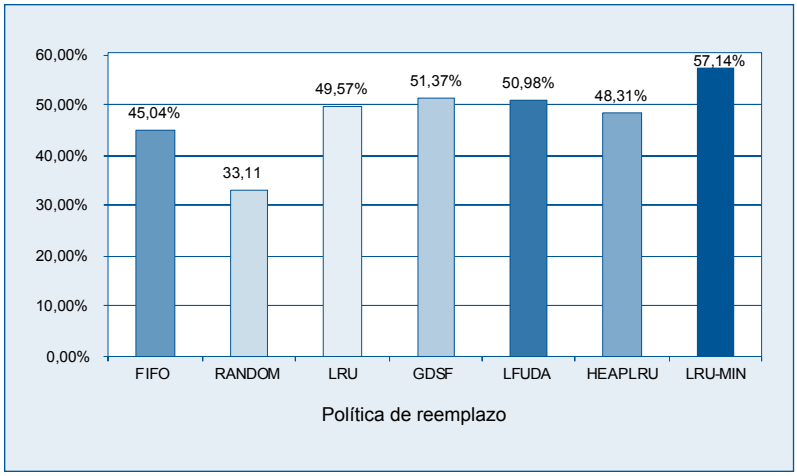


*Figura 2.* Hit rate - Cache 50 MB

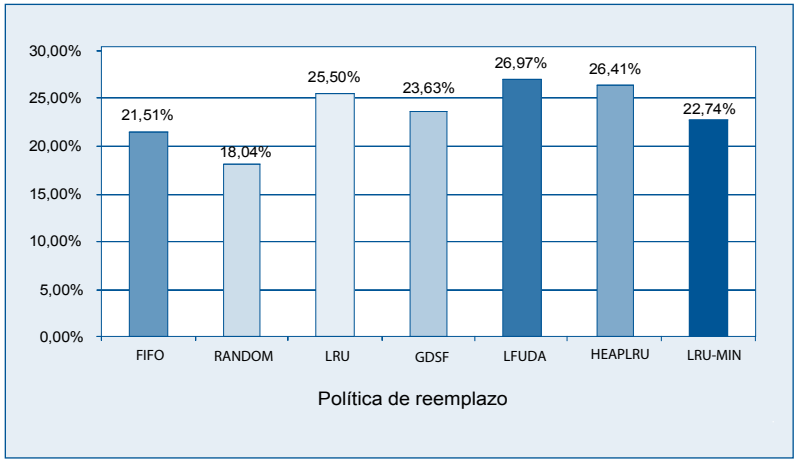


*Figura 3.* Byte hit rate - Cache 50 MB

En el experimento A se puede observar como la política LFUDA es prácticamente la mejor política de reemplazo entre todas las que fueron puestas a prueba. El experimento toma la política LFUDA; se realizan pruebas para diferentes tamaños de caché, entre ellos 10 MB, 30 MB, 50 MB, 70 MB; esto se hizo de esta manera para poder mostrar el comportamiento de una política de reemplazo con diferentes tamaños de cache que es uno de los parámetros más importantes al implementar un *proxy-caché*.

El cuadro 3 muestra los resultados de estos experimentos. Para cada una de las política de reemplazo LFUDA, se midió el número total de solicitudes, el total de *bits* transmitidos, el *hit rate*, el *byte hit rate* y el tiempo total requerido en cada uno de los tamaños de caché predefinidos.

## Conclusiones

Las políticas de reemplazo en la *caché* de web ofrecen una gama muy amplia de posibilidades. Son algoritmos que determinan cuál objeto debe sacrificarse para que los usuarios puedan satisfacer sus peticiones a la web de una forma más eficiente.

*Las políticas de reemplazo en la caché de web ofrecen una gama muy amplia de posibilidades.*

*Cuadro 3.* Política de reemplazo LFUDA

| | Total solicitudes | Total bits | *Hit rate* | *Byte Hit rate* | Tiempo (horas) |
|---|---|---|---|---|---|
| 10 MB | 107932 | 475529K | 35,08% | 24,02% | 73,72 |
| 30 MB | 96471 | 536303K | 47,96% | 28,64% | 64,26 |
| 50 MB | 103805 | 653584K | 50,98% | 26,97% | 46,18 |
| 70 MB | 107558 | 641486K | 52,04% | 29,67% | 63,24 |

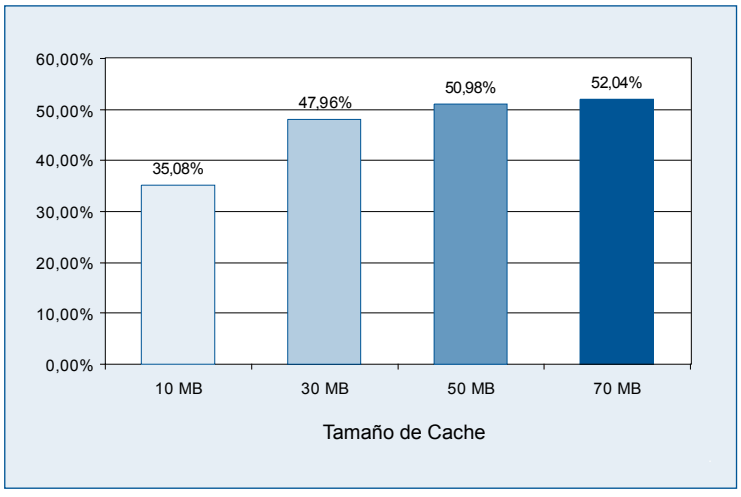


*Figura 4.* LFUDA - Hit rate

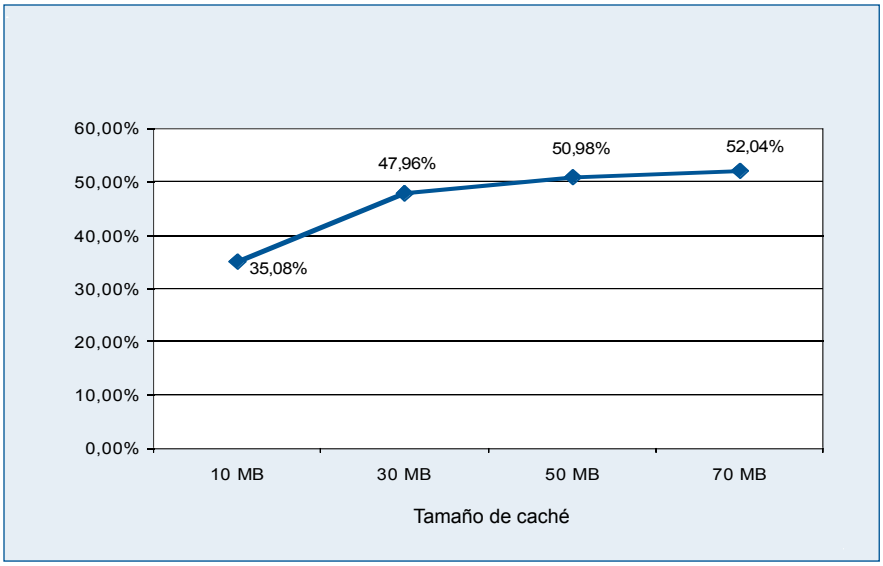


*Figura 5.* LFUDA - Hit rate

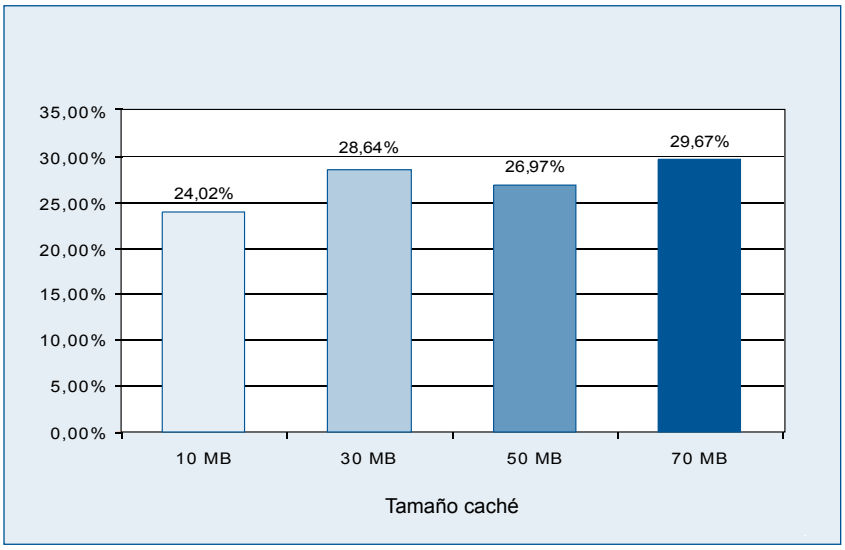


*Figura 6.* LFUDA - Byte hit rate

En los experimentos realizados con el *log* de la Escuela de Ingeniería en Computación, se determinó que la mejor estrategia para mejorar el *hit rate* es LRU-MIN, mientras que para favorecer el *byte hit rate* es LFUDA.

Utilizando la estrategia LFUDA, se logró determinar que a mayor tamaño del *caché*, mayor es el *hit rate*. Lo que indica que un tamaño de caché más grande favorece este rubro, aunque la curva converge alrededor de los 70 MB de *caché*.

## Bibliografía


Arlitt, Martin, Rich Friedrich y Tai Jin. *Performance evaluation of Web proxy cache replacement policies.* Proceedings of Performance Tools '98. Lecture Notes in Computer Science, 1998.

Castillo, Carlos. *Effective Web Crawling.* Tesis doctoral, Departamento de Ciencias de la Computación, Universidad de Chile. Noviembre, 2004.

Chakrabarti, Soumen. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan-Kaufmann, 2003.

Chandhok, Nikhil. *Web Distribution Systems: Caching and Replication***.** http://www.cse.wustl.edu/~jain/cis788-99/ftp/web_caching/ . November 1999.

Wang, Jia. *A Survey of Web Caching Schemes for the Internet***.** ACM Computer Communication Review, Vol. 29, N.° 5, octubre, 1999.

Wessels, Duane. *Squid: The Definitive Guide***.** Editorial O'Reilly, Estados Unidos, 2004.

Wessels, Duane. *Web Caching. Editorial O'Reilly, USA, 2001.*

*Sitio oficial de SQUID Web Proxy Cache: (http://www.squid-cache.org/).*

Sitio sobre Web Caching (http://www.web-caching.com/).

Sitio oficial de Spread: htp://www.ic-itcr.ac.cr/spread/).

Squid Developers. *Squid Programmers Guide: Chapter 12 - Storage Interface***.** (http://www.squid-cache.org/Doc/Prog-Guide/prog-guide-12,html).