

Un Sistema de Web Caching para Acelerar el Acceso Remoto a Documentos

Víctor J. Sosa Sosa, Leandro Navarro

{vjsosa,leandro}@ac.upc.es

Universidad Politécnica de Cataluña (UPC)

Centro Nacional de Investigación y Desarrollo Tecnológico (Cenidet)

Resumen

En este artículo se describe una arquitectura de acceso y distribución de documentos generada a partir de varios estudios hechos a diferentes arquitecturas de caching en el Web. Se demuestra la factibilidad de definir una arquitectura de acceso a documentos, a través del uso del caching en el Web, que si bien no se enfoca a ofrecer la mejor eficiencia bajo un entorno específico de red, muestra flexibilidad a cambios. Con flexibilidad a cambios nos referimos a que si el entorno de red donde se desenvuelve esta arquitectura cambia (lo que trae como consecuencia cambios en políticas de validación de documentos y de comunicación entre caches), el sistema sigue presentando algún beneficio, evitando en cualquier momento la degradación del sistema causada por haber instalado el sistema de caching en el Web, cosa que ocurre en casi todos los sistemas de caching en el Web, lo cual los obliga a rediseñar su arquitectura y detener su funcionamiento.

1. Introducción

En este artículo describiremos una arquitectura de acceso a documentos en el Web la cual está soportada por un sistema de caching en el Web. A diferencia de otros trabajos, en este artículo no pretendemos sugerir que nuestra arquitectura será la mejor para cualquier circunstancia en la que se presente la red en un momento determinado (aunque bajo ciertos contextos pudiera ser la mejor), si no más bien, lo que pretendemos es sugerir una arquitectura que sea más versátil y que siga ofreciendo beneficios independientemente de los cambios que ocurran en su entorno. Otras arquitecturas propuestas dependen de las circunstancias del entorno para ofrecer algún beneficio, y en caso de cambios en el mismo, pueden ocurrir situaciones patológicas que degradan la eficiencia del sistema al grado que funcionaría mejor sin caches. En nuestra arquitectura, bajo ciertas configuraciones en el entorno, podemos hacer que su rendimiento sea el óptimo. Sin embargo, la ventaja adicional ante las demás es su flexibilidad ante los cambios, donde seguirá ofreciendo beneficios, de tal manera que el uso del cache no será un problema.

La idea detrás del Caching consiste en traer los datos lo más cerca que se pueda a sus consumidores, con el fin de minimizar los tiempos de acceso a los mismos. Un sistema de caches cooperativas consta de un grupo de proxy caches que comparten documentos almacenados entre ellos. En resumen, la construcción de un sistema de cache cooperativo requiere de varios aspectos a analizar entre los cuales podemos englobar 4 grandes rubros:

- **Organización** del sistema de caches **cooperativas**:
 - Jerárquica
 - Malla
 - Híbrida
- **Comunicación** entre **caches**, buscando eficiencia (mejor tasa de hits, menor consumo de ancho de banda). Podemos considerar 3 procesos que intervienen aquí: descubrimiento, obtención, y difusión.
 - *Descubrimiento*. ¿Cómo encuentran los documentos las caches?, 3 grandes enfoques:
 - Consulta exhaustiva. ICP (Inter-cache Communication Protocol).
 - Uso de Directorios. Intercambiados de manera:
 - Peer to peer

- En forma jerárquica
 - Hashing. Páginas encontradas en caches definidos por hash.
 - *Obtención.* ¿Cómo obtienen los documentos las caches?
 - Directo del servidor
 - A través de una copia en una jerarquía
 - A través de una copia en una malla
 - *Difusión.* ¿Es factible la transmisión de documentos iniciado por el servidor?. Asignación de documentos a caches elegidas por el servidor.
- **Consistencia.** Estrategias:
 - *Expire.* Manejo de fechas predefinidas de expiración de documentos.
 - *IMS.* Verificación por consistencia cada vez que ocurre un acierto (hit).
 - *IMS-Alex.* Se asigna tiempo de vida proporcional al tiempo transcurrido desde su última modificación.
 - *Invalidación.* El servidor al momento de una actualización de documento invalida las copias solicitadas previamente por las caches.
 - Variantes o combinación de las anteriores.
- Comportamiento de las **cargas de trabajo.** El comportamiento varía según el elemento que participa:
 - Clientes
 - Proxy-Caches
 - Servidores

Las bondades de tener una cache cooperativa se han analizado en [18][6] [1].

2. Definición de la Arquitectura

En base a nuestros resultados en [8][9][10], a estudios de cargas de trabajo en varios proxy-caches, y a trabajos anteriores en el análisis de eficiencia de los sistemas de cache cooperativo, hemos definido una arquitectura la cual presenta las siguientes características:

2.1 Tipo de Organización.

La organización que se propone en esta arquitectura es del tipo **Híbrida** (jerarquías a grandes distancias - latencias- definiéndola a 3 niveles, en donde cada nivel estará enlazado a una malla, figura 1). Resultados en [8][6][18] han demostrado que una estructura de cooperación híbrida ofrece mejores resultados en cuanto a tiempo de acceso a los documentos, y ancho de banda consumido en la red amplia (WAN), además de escalar de mejor manera. Una jerarquía a 3 niveles es un tipo de jerarquía actualmente aceptado por sus buenos resultados, ya que no enfatiza los costos del proceso store-and-forward [4][3] [7][5].

2.2 Tipo de Comunicación.

Como hemos mencionado antes, la comunicación en un sistema de cache cooperativo puede intervenir en varios procesos (descubrimiento, obtención, y difusión). A continuación se explica para cada proceso el método de comunicación que se utilizará.

- 1) Para el proceso de *Descubrimiento*: El proceso de descubrimiento, es el proceso que nos permite ubicar un documento en un área del sistema de cache cooperativo. Un mecanismo muy típico es utilizar la consulta exhaustiva, a través del uso de mensajes ICPs. Sin embargo, en nuestra arquitectura, no se aplicará la búsqueda exhaustiva, por el consumo de ancho de banda que ésta demanda (ver la forma de trabajar de ICP en [2]). Para reemplazar esto, se implementa un

mecanismo para la construcción de directorios (de metadatos). La construcción de estos directorios se hará de manera dinámica. Cuando un documento es solicitado al servidor o a una cache padre, inmediatamente después de emitir dicho documento (cierre de conexión fiable –tcp–), se emite un mensaje (multicast) indicando a los hijos de dicha cache padre o servidor los documentos que han sido solicitados por la cache solicitante. Esto con el fin de que niveles inferiores de la jerarquía se enteren de la ubicación de esos documentos, y lo tengan en cuenta para posteriores consultas. Las caches que recibirán esa información son únicamente las caches que pertenecen a esa rama de la jerarquía en niveles inferiores.

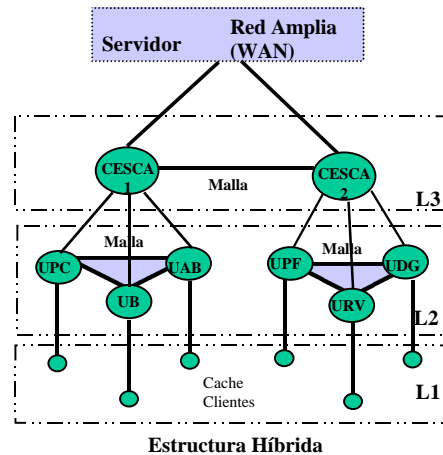


Figura 1. Un ejemplo de una arquitectura jerárquica en 3 niveles con ramas en malla (híbrida)

- 2) Para el proceso de *Obtención*: Los clientes obtienen los documentos a través de copias provenientes de una cache en una malla (hojas de un nivel de la jerarquía). Copias que han llegado a la malla siguiendo la ruta desde el servidor a la cache hoja atravesando los niveles de la jerarquía.
- 3) Para el proceso de *Difusión*: Existirá proceso de difusión a través del mecanismo de invalidación por difusión (pushing). Ésto ocurrirá cuando los documentos que han sido actualizados por el servidor sean emitidos dentro del mensaje de invalidación a las caches hijas. Este proceso se explica con detalle en la sección siguiente.

2.3 Mecanismo de Consistencia.

El mecanismo de consistencia será del tipo *Invalidación Multicast* a través de señalización de vida. Mediante este mecanismo de invalidación, cada minuto (valor obtenido de las evaluaciones hechas durante esta investigación) se emite una señal de vida (un paquete multicast dirigido a las caches hijas) que indica que la cache padre (o el servidor) está en línea con el sistema de cache cooperativa, y que todos sus compromisos adquiridos (contratos emitidos a las caches) siguen en pie. Si algún documento ha sido modificado en el servidor (en especial los documentos que han sido previamente solicitados por alguna cache), se emite un aviso (invalidación) a las caches hijas, a través del mensaje de señalización de vida, para que las caches que contienen dicha copia del documento la invaliden y la actualicen en posteriores solicitudes. Esa misma señal de vida es aprovechada para avisar a las caches que no contienen ese documento (que lo localizarán a través del uso de sus directorios –en la malla a la que pertenecen–) para que no intenten accederlo de una copia inválida. Si las caches hijas no recibieran la señal de vida por parte de las caches padres después de 5 minutos (5 intentos de señales por parte de la cache padre o servidor), asumirán que sus copias provenientes de dichas caches han sido invalidadas.

Una funcionalidad adicional en este mecanismo de invalidación es utilizar el mecanismo de pushing. Con el mecanismo de pushing activo, además de emitir el aviso de la invalidación a las caches hijo, se emitirá el documento o los documentos que han sido modificados, siempre y cuando se haya rebasado un umbral de popularidad basado en heurísticas, como se explica en la siguiente sección.

2.3.1 Estrategia de Difusión (Pushing) en el Mecanismo de Consistencia.

Como hemos mencionado en el mecanismo de consistencia a utilizar (invalidación multicast) tenemos una estrategia de difusión (pushing) que puede ser utilizada como una manera de que los servidores (o caches padres en una jerarquía) difundan los documentos más populares (que han sido actualizados por el servidor) a las caches, antes de ser nuevamente solicitados. En este método se utilizan heurísticas para decidir si una página es lo suficientemente popular como para difundirla (realizar el pushing). En esta arquitectura no se realiza una decisión global sobre difundir la página en todo el sistema de caches o no; en lugar de eso, cada cache y el servidor original toman su propia decisión de manera independiente, y así cada quien decide si realizar la difusión o no. Para ello, cada cache y el servidor original mantienen un contador de los accesos a cada página (X_p) el cual al principio es inicializado a 0, así como un *bit de difusión* el cual indicará si la página tiene que ser difundida o no. Si el *bit de difusión* está en 1, indicará que la cache tiene que difundir la página que ha sido actualizada al momento de emitir su señalización de vida a todas las caches hijas (dentro de la jerarquía que se define en nuestra arquitectura híbrida). La heurística de esta estrategia utiliza tres constantes positivas: ν, β, δ . Si una cache recibe una señal de invalidación para la página P, entonces se efectúa la siguiente operación: $X_p = X_p - \beta$; si la cache recibe una petición para P, entonces calculamos: $X_p = X_p + \delta$;

Si X_p rebasa un umbral ν ($X_p > \nu$) entonces el *bit de difusión* se inicia a 1, de lo contrario se le asigna un 0. Adicionalmente, las caches hijas emiten un mensaje a sus caches padres cada vez que la página es leída, esto con el fin de que el sistema de caches mantenga el estatus de las páginas leídas en toda la rama de la jerarquía. Los valores utilizados para estas constantes dentro de nuestra arquitectura son $\nu=8$, $\beta=1$, y $\delta=2$. La proporcionalidad que hemos puesto de quitar por cada invalidación 1 y añadir por cada petición un 2, se debe a que estudios de popularidad en los logs de algunos proxi-caches se acercan fuertemente a la distribución de Zipf. La distribución de Zipf señala que el número de peticiones a una página (R) está relacionado con su popularidad de la siguiente manera:

$$R(i) = \frac{\Omega}{i^\alpha}$$

donde el exponente α refleja el grado de diferencia de popularidad entre un documento y otro, y la constante Ω define una aproximación al número de peticiones para el documento más popular representado por $i = 1$. Para entender mejor lo anterior veamos el siguiente ejemplo. Supongamos que $\alpha = 1$ y que $\Omega = 100$, entonces el documento más popular (es decir con la posición $i = 1$) tendrá 100 peticiones, el documento con la popularidad siguiente ($i = 2$) tendrá 50, y el siguiente ($i = 3$) se aproximará a 30, y así sucesivamente, lo cual nos da una idea de que la popularidad de los documentos sigue una secuencia de 2 a 1 por cada posición en su índice de popularidad. Lo anterior ayudó a definir las constantes β y δ . Análisis hechos a varios logs de proxi-cache presentan un acercamiento muy fuerte a la distribución de Zipf. El valor de ν viene dado por estudios hechos en este trabajo, así como análisis heurísticos realizados en [19].

2.4 ¿Cómo se Obtuvo esta Arquitectura?

La definición de esta arquitectura de distribución y acceso a documentos se obtuvo en su parte sustancial gracias a la ayuda de los simuladores que han sido desarrollado durante este trabajo [10], los cuales

principalmente son extensiones al NS[17]. El proceso que ayudó a definir esta arquitectura se basó en un análisis comparativo y representativo de diversas arquitecturas de comunicación entre caches con los mecanismos más reconocidos actualmente para validar la consistencia de los documentos en las caches. Como es bien conocido en nuestros días las redes tienden a presentar crecimientos y modificaciones topológicas que hacen que los sistemas de cooperación entre caches sufran de algunos problemas de eficiencia al momento de utilizar un sistema de consistencia en particular, por esta razón, lo que se busca como resultado de esta arquitectura es tener un sistema cooperativo de caches en el Web que sea flexible a crecimientos y modificaciones en la red en que se implemente. El proceso fundamental que ayudó a la definición de la arquitectura que aquí proponemos, está basado en los análisis que realizamos en [8], el cual básicamente consistió en revisar (utilizando el simulador) las formas más típicas para conectar un sistema de caches cooperativas como son: jerarquía, malla e híbridas, utilizando varios algoritmos que permiten validar los documentos en una cache. Los algoritmos más conocidos y aceptados son: TTLA0, TTLA, M, PSM, APM [8]. Las configuraciones anteriores se aplicaron al simulador para que iniciara su proceso a partir de la reproducción de las cargas de trabajo que fueron obtenidas de los logs de las caches que se encuentran en el troncal de la red académica catalana que administra el Centro de Super Cómputo de Cataluña (CeSCa). El simulador detiene su proceso para cada configuración al momento en que termina de reproducir toda la carga de trabajo. Las cargas de trabajo utilizadas se pueden ver en la tabla 1.

Número de peticiones	3,089,592
Número de objetos	212,352
Número de clientes aproximado	11,765
Promedio de peticiones por segundo	21
Bytes transferidos	30GB
Duración	2 días

Tabla 1. Características de las trazas utilizadas en las simulaciones

A manera de resumen podemos comentar que los resultados obtenidos en [8] nos permitieron obtener algunas formas de comunicación entre caches y algoritmos de validación de consistencia que son flexibles a los cambios topológicos en una red. De esa manera, dichos resultados nos dieron pie a definir una forma de comunicación entre caches (más escalable) que al ser combinada con un mecanismo de consistencia basado en invalidación por multicast al cual le hemos hecho mejoras al utilizarlo también en el proceso de descubrimiento que se comenta en la sección 2.2) nos ha dado resultados muy buenos que han dado pie a la arquitectura que aquí se presenta y de la cual describiremos su modo de operación a continuación.

2.5 Modo General de Operación de la Arquitectura

El sistema de cache definido bajo la arquitectura propuesta operará de la siguiente forma: un cliente emite una petición a su cache-cliente (la cache que es primeramente contactada por un cliente la llamaremos cache-cliente), la cache-cliente verifica si puede resolver la petición, si es así, regresa el documento solicitado y la operación termina. De lo contrario, revisa (a través de un directorio) si alguna de las caches vecinas contiene el documento solicitado. Si el documento está contenido en una de las caches vecinas (caches conectadas en malla, no incluye el padre) entonces se remite la petición a dicha cache, de la cual se obtiene el documento para posteriormente entregárselo al cliente. Si ningún vecino contiene el documento, entonces la petición es remitida al padre el cual analiza su directorio para ver si alguno de sus vecinos contiene el documento, si no es así el proceso se repite recursivamente hasta encontrar el documento en una cache intermedia o en el servidor. Cada cache padre o servidor (si no se encuentra el documento en la jerarquía), mantiene una lista de (cache_solicitante, documento) la cual será emitida (vía multicast) inmediatamente después de cerrar la conexión de la petición, para dar a conocer entre sus hijos quién tiene el documento solicitado, de esta manera los hijos caches estarán enriqueciendo su directorio. Después de que dicha lista es emitida a las caches hijas, desaparece del servidor (o caches padres)

disminuyendo la carga en el control. Los directorios que mantiene cada cache intermedia refieren únicamente a los documentos que están almacenados en la malla a la cual pertenecen, y no a otros niveles. Esto ayuda a cercar el rango de acción de cada nivel de cache cooperativa (la idea de esta arquitectura es aplicar los mecanismos de cooperación y de validación por regiones –cercanas–), y no crear gasto de conexiones a muy largas distancias, los cuales han sido consideradas con alto grado de penalización en un sistema de cache cooperativa a gran escala [16].

Al momento que un documento es modificado por el servidor, el servidor mantiene una lista de documentos que son modificados para ser enviados al momento de emitir la señal de vida. Los servidores emiten una invalidación (sobrecargando la señal de vida) a su grupo multicast asociado, de esta manera, no tiene que llevar el control de todas las caches que han accedido los documentos involucrados. La distribución de esta invalidación, así como los documentos que viajan del servidor a las caches clientes, se hace a través de una jerarquía de cache que controla grupos multicast asociados. Las caches hijos se incorporarán al grupo multicast que define su padre. En la figura 2 se muestra un extracto de la arquitectura propuesta.

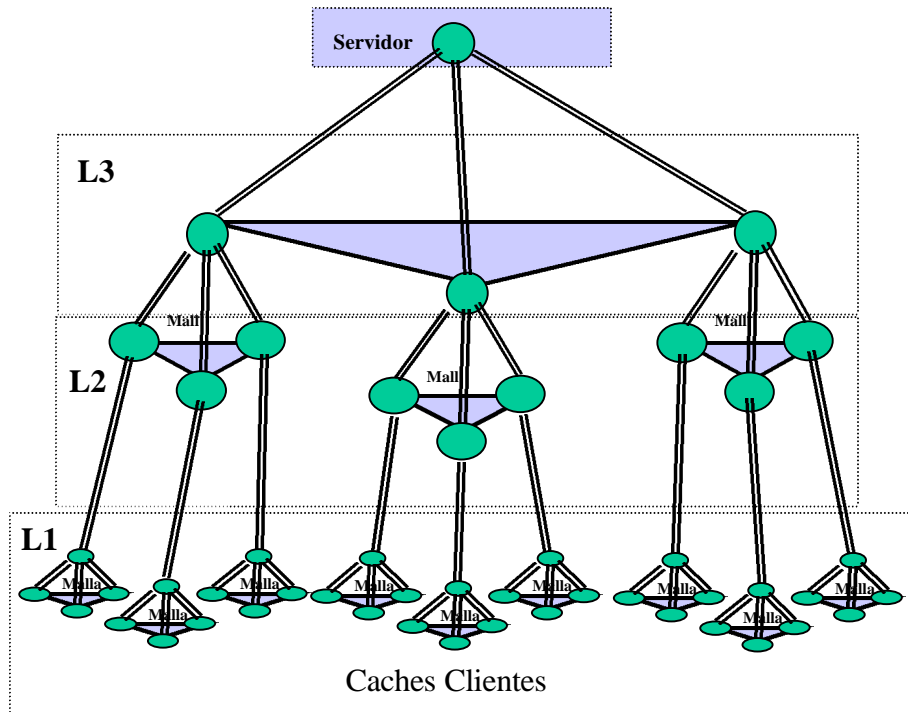


Figura 2. Arquitectura de distribución/acceso a documentos.

La figura 2 muestra con doble línea por dónde circulará la señal de vida que emite el servidor o caches padres (cada minuto vía multicast). La señal de vida es un indicador a las caches hijos de que el servidor o cache padre sigue conectado (vivo). Si una cache deja de recibir más de 5 señales de vida consecutivas entonces asumirá que los documentos que contiene de dicho servidor son inválidos. Esto da oportunidad de que alguno de los mensajes pueda perderse en el camino. Cada rama de caches que contenga documentos válidos que hayan sido solicitados por sus niveles inferiores retransmitirá esa señal recursivamente vía multicast. Si la funcionalidad de pushing está activa (*bit de difusión* = 1), cuando algún documento solicitado ha sido modificado por el servidor entonces será retransmitido sobrecargado en la señal de vida. La transmisión de ese o esos documentos se hará únicamente por las ramas en las que haya sido solicitado, es decir, se filtrarán los documentos en cada rama en los niveles inferiores según sea el caso.

2.6 Análisis de la arquitectura

A continuación se muestran algunos análisis comparativos de la arquitectura que aquí definimos con algunas arquitecturas típicas en la actualidad (como las que se mencionan en [8]). La idea de este análisis es visualizar qué tanto es afectada nuestra arquitectura de cooperación al cambiar el contexto de validación de documentos.

Los escenarios de simulación se resumen en la tabla 2. Los análisis se han hecho utilizando las cargas de trabajo de la tabla 1.

Identificadores	Descripción
JTTLA0, JTTLA, JM, JPSM, JAPM	Arquitectura de cooperación jerárquica (J) con mecanismos de consistencia TTLA0, TTLA, M, PSM, APM
DTTLA0, DTTLA, DM, DPSM, DAPM	Arquitectura de cooperación distribuida (D) con mecanismos de consistencia TTLA0, TTLA, M, PSM, APM
H1TTLA0, H1TTLA, H1M, H1PSM, H1APM	Arquitectura de cooperación híbrida 1 (H1) con mecanismos de consistencia TTLA0, TTLA, M, PSM, APM
H2TTLA0, H2TTLA, H2M, H2PSM, H2APM	Arquitectura de cooperación híbrida 2 (H2) con mecanismos de consistencia TTLA0, TTLA, M, PSM, APM
ADDTTA0, ADDTTLA, ADDM, ADDPSM, ADDPM	Arquitectura para la Distribución de Documentos (ADD) propuesta con mecanismos de consistencia TTLA0, TTLA, M, PSM, APM

Tabla 2. Escenarios de prueba para las simulaciones

En lo que se refiere a tiempos de respuesta percibidos por los clientes, podemos ver en la figura 3 cómo los mecanismos de validación afectan a cada arquitectura de cooperación entre caches de diferente manera durante la reproducción de las cargas de trabajo bajo el ambiente de pruebas (distribución de caches en las que interviene el Cesca). Se ve que la arquitectura de cooperación distribuida bajo este contexto, es la que ofrece mejores resultados. Aunque también se puede percibir que ligeramente abajo la arquitectura que estamos proponiendo ADD (Arquitectura para la Distribución de Documentos) muestra mejores resultados que el resto, independientemente del algoritmo de validación que se esté utilizando. Un poco más abajo, y solo para algoritmos de validación TTLA (Adaptative-TTL) bajo una arquitectura de cooperación entre caches como la que se describe en H2, los tiempos de respuesta son muy aceptables, sin embargo si se modifica el algoritmo de validación, los tiempo de respuesta que perciben los clientes sufren un castigo considerable.

En lo que refiere al ancho de banda consumido tanto en los enlaces entre caches, como en los enlaces al exterior del sistema de cache cooperativa (WAN) podemos notar lo siguiente, en la figura 4 vemos como las arquitecturas de cooperación entre caches totalmente en malla (Distribuida) presentan un consumo masivo de los servicios de la red intra-caches, esto se debe considerar para el caso en que necesitamos algunos otros servicios de red que utilizan esos mismos enlaces. El sistema de cooperación jerárquica (comunicación solo padres e hijos) es la opción más viable para obtener un menor consumo de ancho de banda en los enlaces que conectan el sistema de cooperación entre caches debido a la nula interacción que existe entre las caches del mismo nivel en la jerarquía (independientemente del algoritmo de validación que se utilice), sin embargo, vemos que el consumo de ancho de banda hacia la red externa se dispara, lo que podría ocasionar casos patológicos en algunos tiempos de respuesta. De hecho, podemos ver en la figura 3 que el sistema jerárquico es el que aparece como el más lento.

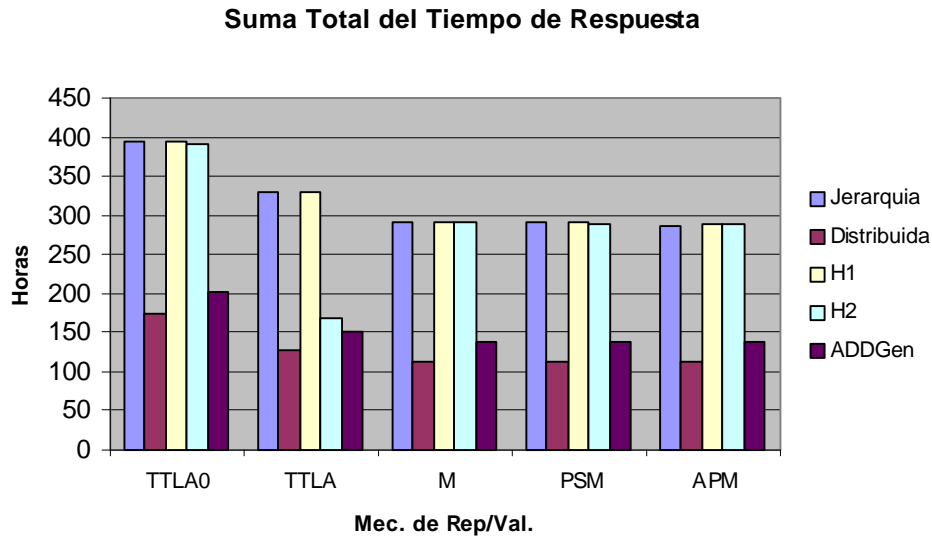


Figura 3. Suma total de los tiempos de respuesta percibidos por los clientes

En la figura 4, también podemos notar que la arquitectura ADD presenta consumos competitivos de ancho de banda tanto al interior del sistema de caches cooperativas como al exterior.

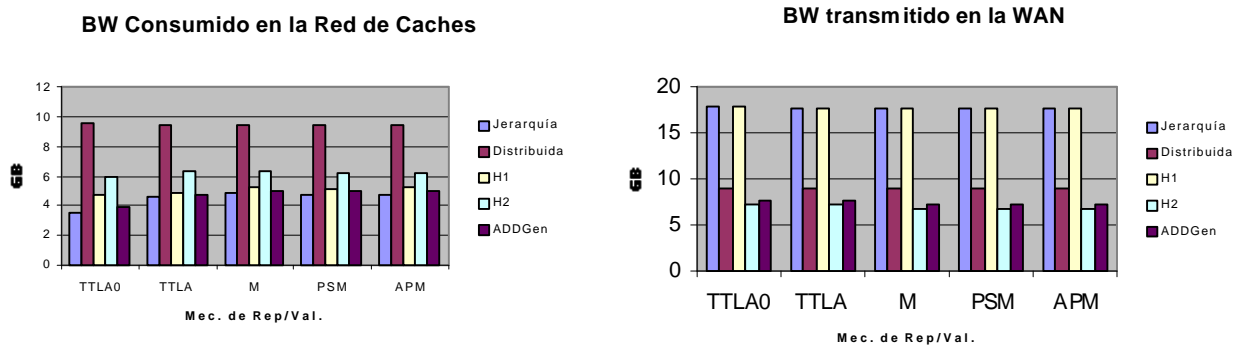


Figura 4. Consumo de ancho de banda (BW) en la red que interconecta las caches y en la WAN

Como ya hemos mencionado, los mecanismos de validación tiene como principal labor mantener la consistencia de los documentos. Es decir, procurar que los documentos que serán entregados a los clientes por las caches estén lo más actualizados con el servidor original. Dependiendo de los parámetros que se hayan dado a cada mecanismo de validación puede lograr mayor o menor grado de consistencia en los documentos. En la figura 5 comparamos el número de documentos que arribaron al cliente sin estar actualizados tomando como base los parámetros típicos que se utilizan en cada mecanismo de validación. Como podemos ver, si nuestro interés es obtener una máxima consistencia, el mecanismo que nos ofrece esa garantía en cualquier tipo de arquitectura es el mecanismo de TTLA0. El siguiente mecanismo que nos ofrece mejores resultados es el mecanismo de invalidación multicast con pushing selectivo (PSM). Esto, bajo cualquier sistema de cooperación entre caches que elijamos.

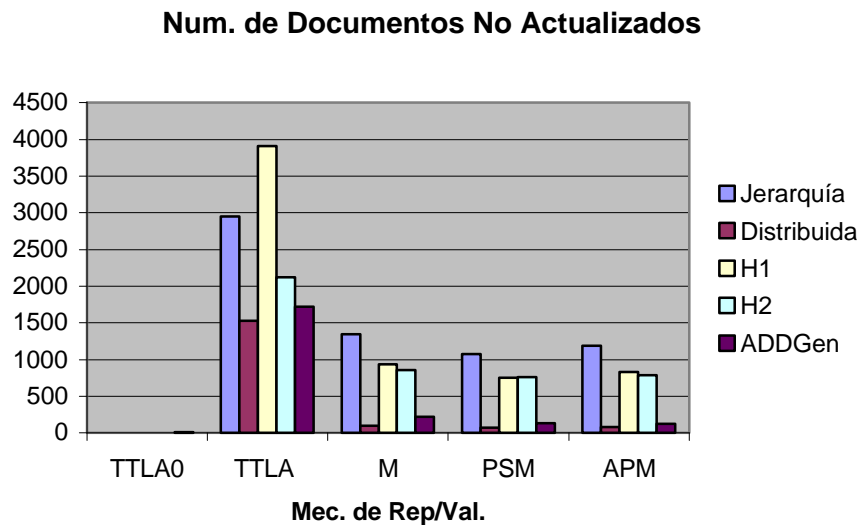


Figura 5. Número de documentos inconsistentes que recibieron los clientes

Ya hemos dicho que muchos trabajos relacionados con caches han evaluado y comparado las arquitecturas de cooperación entre caches por una parte, o los mecanismos de validación por otra. Normalmente el parámetro de comparación es ver qué tan bueno es uno con respecto al otro. Sin embargo, existe la polémica de saber el grado de beneficio (si lo hay) que producen las arquitecturas de caches cooperativas y sus mecanismos de validación comparado con un sistema que no utiliza caches, tomando como métrica de comparación la suma de los tiempos de respuesta obtenidos al reproducir las trazas en cada uno de los sistemas. En la figura 6 se grafica el grado de ganancia (Speedup) obtenido por cada arquitectura cooperativa y sus respectivo mecanismos de validación. La arquitectura que no utiliza caches la hemos llamado arquitectura de Proxies No Caches (PNC). Un Speedup igual a 1, nos indica que la arquitectura de caches cooperativa en cuestión y el mecanismo de validación que esté utilizando no está generando ganancia alguna, si no más bien, que el usar una arquitectura de caches cooperativas produciría el mismo efecto en cuanto a tiempos de respuesta que un sistema que no usa cache. Todos los valores arriba de 1 en el speedup representan el porcentaje de ganancia o beneficio. Vemos que los mecanismos de cooperación que presentan mejores resultados (algún beneficio) independientemente del cambio de algoritmos de validación, son el distribuido y el ADD (hasta beneficios superiores del 2,5 y 3,5), lo cual hace que tengamos una garantía de ganancia utilizando cualquiera de estas arquitecturas independientemente del algoritmo de validación que usemos.

En la figura 7 se presentan todas las arquitecturas revisadas en este trabajo contrastando los speedups con el consumo de ancho de banda que generan. La cercanía a la marca *ideal* representa mayor beneficio a menor costo. Las configuraciones ADDPSM, ADDAPM, DAPM, DTTLA y H2M son las mejores alternativas cuando el interés está puesto en la infraestructura de red. Se han hecho otros experimentos donde se escalan los sistemas de cooperación de cache (no incluidos en este artículo) en donde la arquitectura que mayor varía en los resultados es la arquitectura de cooperación distribuida en el sentido de los tiempos de respuesta y los speedups. El crecimiento en el número de caches impacta en los consumos de ancho de banda entre las caches que se utilizan en las arquitecturas totalmente distribuidas, lo que hace que se refleje en la disminución de la eficiencia de esta arquitectura cooperativa.

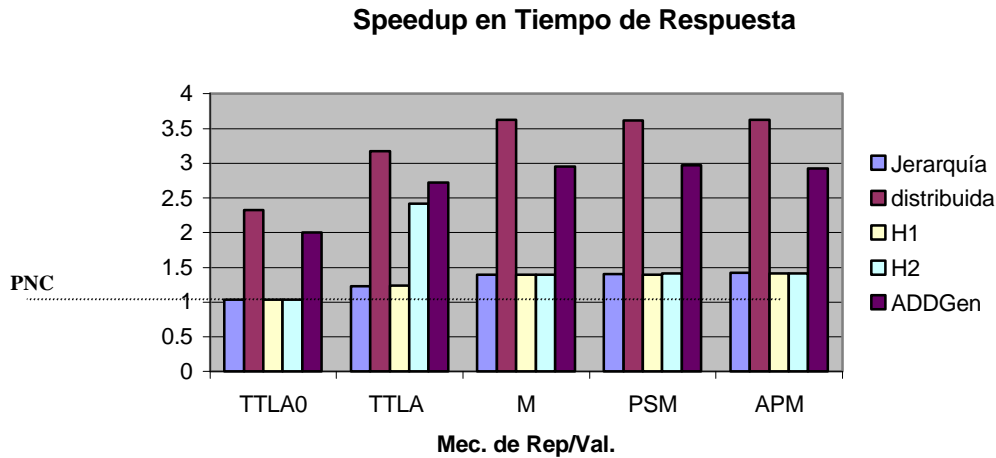


Figura 6. Speedup en la suma de tiempos de respuesta de las diferentes configuraciones con respecto a la arquitectura que no utiliza caches (PNC)

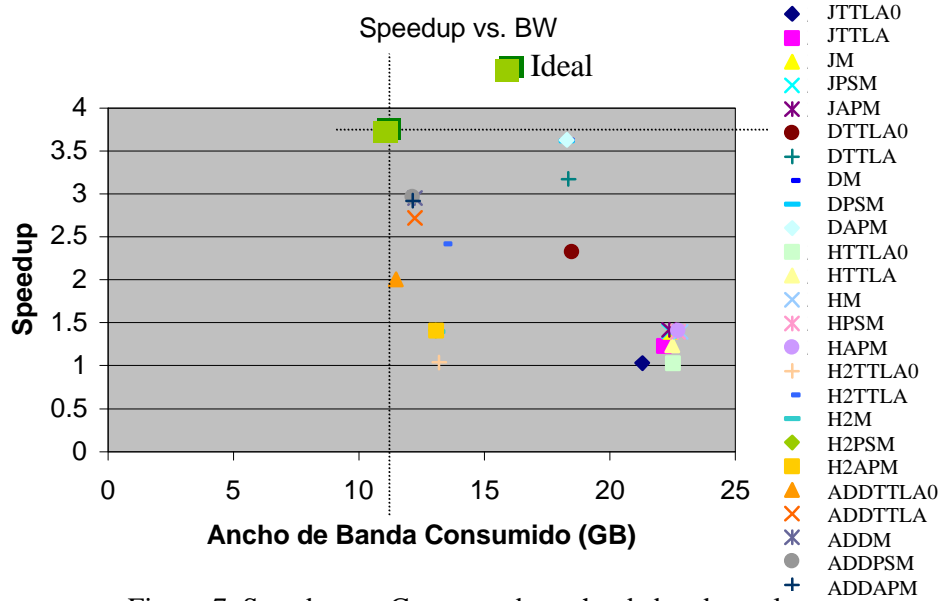


Figura 7. Speedup vs. Consumo de ancho de banda total

3 Conclusiones.

En este artículo se presenta una arquitectura de distribución de documentos la cual presenta la característica de ser flexible. Es decir, que puedan variar los parámetros del entorno y que aún se obtengan beneficios de la misma sin penalizar demasiado los recursos de la red. La mecánica a seguir fue la evaluación de sistemas cooperativos de caches (clásicos), combinados con los mecanismos de validación de documentos que mejores resultados han mostrado en la literatura. Con estas evaluaciones hemos encontrado que una arquitectura del tipo ADD, como la que se ha descrito en este capítulo puede ofrecer dichas prestaciones. La arquitectura está basada principalmente en un sistema de cooperación híbrido, con un sistema de búsqueda de documentos basado en directorios, y con la cual podemos definir como mecanismo de validación en casos óptimos los mecanismos de validación por invalidación multicast con señalización de vida. En caso de que exista la necesidad de modificar el mecanismos de validación por

requerimientos en los cambios de los recursos del sistema de caching, tendremos la confianza de que a pesar de ello seguiremos obteniendo beneficios del sistema de caching, cosa importante que aporta nuestra arquitectura.

4. Referencias y bibliografía.

- [1] S.G. Dykes, C.L., K.A. Robbins, and C.L. Jeffery, "A Viability Analysis of Cooperative Proxy Caching", in Proc. of the IEEE Infocom 2001
- [2] <ftp://ftp.rediris.es/docs/rfc/21xx/2186> "Internet Cache Protocol (ICP) version2"
- [3] Korea National Cache, <http://cache.kaist.ac.kr>
- [4] National Laboratory for Applied Network Research (NLNR), "Ircache project", available on line at <http://ircache.nlanr.net/>
- [5] Red Académica Española (Red Iris) Disponible en: <http://www.rediris.es/>
- [6] P. Rodriguez, C. Spanner, and E. W. Biersack, "Web Caching Architectures: Hierarchical and Distributed Caching". 4th International Web Caching Workshop, San Diego, USA. 31st March –2nd April, 1999.
- [7] N. G. Smith, "The UK national Web Cache – The state of the art", Computer Networks and ISDN System, 28:1407-1414, 1996
- [8] V.J. Sosa, L. Navarro, "Influence of the Document Replication/Validation Methods on Cooperative Web Proxy Caching Architectures". Communication Network and Distributed Systems Modeling and Simulation Conference CNDS'02 dentro del WMC'02. Pags. 238-245. ISBN: 1-56555-244-X. San Antonio, Tx. USA.
- [9] V.J. Sosa, L. Navarro, "Influencia de los Mecanismos de Replicación/Validación de documentos en las arquitecturas de Caches Cooperativas". para el Workshop de Sistemas Distribuidos y Paralelismo (WSDP'01). Semana de Computación Chilena, Nov. 2001. Chile. CD-ROM.
- [10] V.J.Sosa, L. Navarro, "A New Environment for Web Caching and Replication Study", para el Workshop de Sistemas Distribuidos y Paralelismo (WSDP'00). Semana de Computación Chilena 2000. Universidad de Santiago. Nov. 13-18 2000. Chile. ISBN 956-7069-53-0. CD-ROM.
- [11] V.J. Sosa, L. Navarro, "Proxycizer2ns: A Suitable Combination for Web Caching and Replication Study", para el Simposio Español de Informática Distribuida SEID00, pags. 197-205. ISBN: 84-8158-163-1. Sep. 25-27, 2000, Orense, Galicia, España.
- [12] V. J. Sosa, L. Navarro, O. Ardaiz, "Gestor local para el acceso y distribución de documentos Web a bibliotecas digitales: Biblioteca proxy". Congreso Internacional de Investigación en Ciencias Computacionales, CIICC'99. Pags. 24-31. Septiembre 1999, Cancún, México.
- [13] V. J. Sosa, L. Navarro, O. Ardaiz, "Document Distribution to Internet Digital Libraries", Revista Computación y Sistema, México, pags. 169-173. ISSN: 1405-5546. D.F., Abril 99. (Artículo del SEID99 seleccionado para la revista).
- [14] V. J. Sosa, L. Navarro, O. Ardaiz, "Document Distribution to Internet Digital Libraries", para el Simposio Español de Informática Distribuida SEID99, Santiago de Compostela, Galicia, Feb. 99, Pags. 351-357. ISBN: 86-8408-060-9
- [15] V.J.Sosa, "Evaluación de Una Jerarquía de Caches en Internet", Simposio Internacional de Computación, CIC'98. México, D.F. Nov. 98. Pags. 370-380. ISBN: 970-18-1916-0
- [16] R. Tewari, M. Dahlin, H. M. Yin, and J. S. Kay, "Design considerations for distributed caching on the internet", in Proc. of the Int'l. Conf. On Distributed Computing Systems (ICDS'99).
- [17] Virtual InterNetwork Testbed Project. Disponible en: <http://netweb.usc.edu/vint/>
- [18] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative Web Proxy caching", in Proc. of the 17th ACM Symp. On Operating Systems Principles, Dec. 1999.
- [19] H. Yu, L. Breslau, and S. Shenker, "A Scalable Web Cache Consistency Architecture". SIGCOMM99. volume 29, number 4, October 1999.
<http://www.acm.org/sigs/sigcomm/sigcomm99/papers/session5-1.html>