EVALUACIÓN DE SISTEMAS DE CACHE WEB PARTICIONADAS EN FUNCIÓN DEL TAMAÑO DE LOS OBJETOS

DIEGO ALEJANDRO LEÓN MEDINA

UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE INFORMÁTICA SISTEMAS Y COMPUTADORES
MASTER EN INGENIERÍA DE COMPUTADORES
2006-2007

EVALUACIÓN DE SISTEMAS DE CACHE WEB PARTICIONADAS EN FUNCIÓN DEL TAMAÑO DE LOS OBJETOS

DIEGO ALEJANDRO LEÓN MEDINA

TUTOR: JOSÉ ANTONIO GIL SALINAS

UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE INFORMÁTICA SISTEMAS Y COMPUTADORES
MASTER EN INGENIERÍA DE COMPUTADORES
2006-2007

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

INDICE DE TABLAS

INTRODUCCIÓN

La siguiente tesis es la respuesta a inquietudes sobre arquitectura de la web, busca de manera general iniciar la investigación sobre este tema, formulando una arquitectura propia que permita evaluar los sistemas de cache web particionado en función del tamaño de los objetos. Los objetos WEB son cacheados en los proxies (caches WEB) reemplazando objetos grandes por muchos objetos pequeños, esto hace que las tasas de aciertos se vean afectadas en función del tamaño de los objetos que cachean, en este documento se evalúa experimentalmente como una jerarquía de proxies puede mejorar las tasas de aciertos y las tasas de aciertos por byte comparada con una configuración que utilice un solo proxy.

Todos los planteamientos realizados han sido el resultado de largas reflexiones, planteamientos y experimentos, que se espera no sean tan densos para conservar el documento didáctico, comentando aspectos muy técnicos brevemente, se adjuntará toda la información, programas y herramientas utilizadas, para posibles ampliaciones y perfeccionamientos de la arquitectura y mejoras en la configuración de las caches web particionadas.

1.1 Objetivos

- Establecer un sistema de simulación para caches particionadas en función del tamaño de los objetos.
- Determinar si es posible aumentar las prestaciones de un sistema de cache web mediante la partición de la cache en función del tamaño de los objetos almacenados.
- Comprobar si el funcionamiento de dos caches WEB configuradas en jerarquía puede obtener mejores resultados que el uso de una sola cache WEB.
- En el caso que la partición de caches sea conveniente, se pretende formular una metodología para determinar cuál es el tamaño de particionamiento óptimo.

1.2 Justificación

Hoy en día en la era de la información los datos deben estar de la forma más rápida en cualquier sitio donde se soliciten, en ese punto, redes como internet hacen que todo este flujo vaya de un lado a otro a través de las tecnologías de la información. En estas tecnologías podemos aportar un grano de arena con estudios como este, en el que se busca mejorar de alguna forma las prestaciones de redes como internet intentando aprovechar los recursos existentes de la mejor manera posible.

Dado que se conoce muy poco sobre estudios similares, se pretende aportar esta arquitectura, como base para estudios posteriores. El propósito de esta tesina es de formar un nuevo conocimiento en el que sus resultados puedan ser de utilidad a los investigadores que trabajen en temas de arquitecturas de la web.

1.3 Resumen de la Metodología a utilizar

- Formular los conceptos básicos de arquitectura de la web, que permitan crear un marco referencial para la tesina.
- Consultar el estado del arte del particionamiento de las caches WEB, para extraer ideas generales que puedan ser aplicadas a esta tesis.
- Definir las partes y módulos que debe contener la arquitectura.
- Inferir las técnicas y herramientas necesarias para la puesta en marcha de la arquitectura.
- Establecer los requisitos para lanzar una simulación contra la arquitectura.
- Analizar el comportamiento de las simulaciones con la arquitectura escogida, examinando los aciertos y posibles mejoras frente a una simulación de referencia, para al final concluir cual puede ser la configuración de la cache web partida que puede obtener mejores resultados.
- Concluir qué datos son necesarios a ser medidos con la arquitectura y definir todo lo necesario para analizar los resultados.
- Deducir si el funcionamiento de dos caches WEB configuradas en jerarquía puede obtener mejores resultados que el uso de una sola cache WEB.
- Identificar las posibles líneas de investigación asumibles tomando como punto de partida los planteamientos de esta tesina.

2 MARCO TEÓRICO

2.1 Servidor(es) proxy

En el contexto de las redes informáticas, el término proxy hace referencia a un programa o dispositivo que realiza una acción en representación de otro. La finalidad más habitual es la del servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP controlando el acceso que realizan los clientes de la intranet a internet. De otro lado sirve como web cache al almacenar las peticiones de los clientes que más se repiten, evitando que futuras peticiones tengan que volver a pedir los objetos a las fuentes. [2]

2.2 Web Cache (Caché Web)

Es la definición de cache a nivel de la web, y como todas las caches su objetivo principal es agilizar el sistema evitándole trabajos innecesarios. Una caché web almacena copias de los objetos web que pasan por ella, de forma que subsiguientes peticiones pueden ser respondidas por el propio caché, si se cumplen ciertas condiciones.

Los siguientes son los tipos de caches web que existen, discriminadas por su funcionamiento.

2.2.1 Caches Agente Usuario (browser Caches)

Representan a las caches presentes en los navegadores WEB, constituyen a una cache local ubicada en el usuario final para cachear los objetos web en un punto más cercano al cliente que hace la petición de los objetos WEB.

2.2.2 Proxies Cache

Este tipo de caches se encarga de cachear peticiones de diferentes usuarios que realizan sus peticiones a través de servidores proxy, sus tasas de acierto son mucho mayores a las de las browser Caches, debido a que la gran cantidad de peticiones de objetos que manejan. [9]

Este tipo de caches en la World Wide Web está representada por los proxies, los clientes se conectan a Internet a través de estos proxies, los cuales al final proveen la respuesta, ya sea de su cache local o haciendo una nueva petición de información al servidor destino de la petición. Cabe anotar que no todos los proxies se utilizan como caches, hay unos que son utilizados como gateways que controlan el paso del tráfico a través de ellos. Es muy recomendable que los proxies sean puestos tan cerca como sea posible de los clientes.

La idea fundamental es la de servir documentos que sean pedidos más de una vez de la cache local del proxy a los clientes, sin tener que salir de la intranet, este es el caso ideal. Dado que los proxies afectan sustancialmente el tráfico de la red, típicamente en un entorno normal sobre el que se implementan Web Caches, se puede lograr reducir el tráfico entre un 30 a un 50 % [9].

2.3 Proxy Cache Jerárquicas

Este es un concepto lógico del concepto de Caching. Un grupo de caches se puede ver beneficiado por la cache que tiene cada uno de los proxies implementados, así si por ejemplo se solicita un determinado objeto X a una de las caches, ésta puede no tener el objeto en la cache pero, se hace posible que el objeto sea servido por alguna de las otras proxy-caches de la jerarquía [2].

2.3.1 Ventajas de la implementación de las Caches Jerárquicas

Éxitos adicionales por tener cacheados objetos en otras caches. Se puede esperar que el 10 % de las peticiones que pide una cache, las pueda encontrar como éxitos en sus vecinas [2].

Enrutamiento de la petición: Se puede configurar a través de los proxies la forma como pueden ser enviadas las peticiones a la red, dicho esto, se hace posible que las peticiones de los clientes se puedan clasificar y enrutar por el enlace que se desee.

2.4 Éxitos de la cache

Cuando hablamos de éxito de la cache se estará haciendo referencia a objetos web que han sido pedidos al proxy y éste los tenía en su Web cache.

2.5 Características de las Navegaciones de objetos WEB

Las siguientes son las principales características de las navegaciones de objetos WEB:

- Referenciados una vez: Objetos sólo requeridos una vez y que nunca podrán ser un acierto para la cache.
- Popularidad: Qué objetos son más o menos populares.
- Distribución por tamaño, a lo largo del tiempo de la simulación se accederán objetos WEB de todo tipo de tamaños.
- Localidad temporal: La posibilidad de requerir un objeto una segunda vez.

2.6 Objeto Web

Es una entidad que es intercambiada entre los clientes y los servidores web, es confundida por los usuarios habituales como las páginas web, pero en realidad un objeto

web hace referencia más al tipo de contenido que puede ser devuelto por los servidores web como, texto, imágenes, zips, etc. Los RFCs que definen los objetos web, hacen referencia a ellos en términos de entidades o recursos. En general se podrían encontrar dos clasificaciones generales para estos objetos dependiendo de su forma de generación: Objetos estáticos que vienen preparados por los servidores de antemano y son servidos a los clientes sin modificación alguna la segunda clasificación sería objetos dinámicos, que son objetos web servidos dependiendo de parámetros modelados por el cliente o el mismo servidor que son confeccionados en el mismo momento de la petición.

2.7 Métrica de las simulaciones

En las simulaciones que se van a realizar se van a tener en cuenta los siguientes conceptos, al momento de la obtención de resultados.

TCP_HIT

Indica que una copia válida del objeto se encuentra en la cache web.

Tráfico TCP_HIT (MB)

Indica el tráfico salvado por los aciertos obtenidos por la cache WEB.

TCP_MEM_HIT

Una copia del objeto estaba en la cache y estaba en memoria lo cual evitó que hubiera un acceso a disco.

TOTAL DEL EXITOS

Indica el total de objetos servidos exitosamente por la cache.

TOTAL HITS VOLUME

Indica el total de éxitos respecto a la cantidad de navegaciones que han sido procesadas por la WEB cache.

2.7.1 Tasa de éxitos (HR, Hit Ratio)

Indica el porcentaje de objetos web que son servidos por la cache. Es una medida que indica que tan eficiente es la cache que se está evaluando. Está descrita por la siguiente fórmula:

$$HR = \frac{\sum_{i=1}^{n} Hit_i}{\sum_{i=1}^{m} Request_j}$$

Esta será una de las medidas utilizadas como referencia en los análisis de los resultados.

2.7.2 Tasa de éxitos por Byte (BHR, Byte Hit Ratio)

Indica cuantos bytes del total de bytes requeridos han sido servidos exitosamente por la cache. Está descrita por la formula:

$$BHR = \frac{\sum_{i=1}^{n} Hit_{i(Bytes)}}{\sum_{i=1}^{m} Request_{i(Bytes)}}$$

Expresado de otra manera, corresponde al total del tráfico salvado por la cache, respecto de tráfico que ha pasado por ella.

2.8 Codificación MD5

De [[1]] MD5 es uno de los algoritmos de reducción criptográficos diseñados por el profesor Ronald Rivest del MIT (Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts). Fue desarrollado en 1991 como reemplazo del algoritmo MD4 después de que Hans Dobbertin descubriese su debilidad, está descrito completamente en el RFC 1321 .

Se enseña este concepto debido a que esta codificación será utilizada en una sección del planteamiento de la arquitectura.

Los resúmenes MD5 se utilizan extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de internet no se ha alterado. Comparando una suma MD5 publicada con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta MD5 para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del primer archivo. En los sistemas UNIX, el comando de md5sum es un ejemplo de tal herramienta. Además, también está implementado en el lenguaje de scripting PHP.

En sistemas UNIX y GNU/Linux se utiliza el algoritmo MD5 para cifrar las claves de los usuarios. En el disco se guarda el resultado del MD5 de la clave que se introduce al dar de alta un usuario, y cuando éste quiere entrar en el sistema se compara la entrada con la que hay guardada en el disco duro, si coinciden, es la misma clave y el usuario será autenticado. He ahí el problema de encontrar y generar colisiones de hash a voluntad.

El MD5 también se puede usar para comprobar que los correos electrónicos no han sido alterados usando claves públicas y privadas.

2.9 Formato del access.LOG

El siguiente es un resumen del formato del fichero de trazas "access.log" que es creado por defecto por los proxies squid, este resumen será utilizado más adelante para inferir alguna utilidad de sus campos:

PARÁMETROS	PARÁMETROS EJEMPLO
1) Indica los segundos pasados después de 1970 hasta la fecha de la navegación	1122674316.907
2) Cuanto tiempo se ocupo la caché Tiempo entre aceptar y terminar	4846
3) Dirección IP cliente que hace petición.	158.42.175.25
4) Resultado de la transacción	TCP_MISS/200
5) Cantidad de bytes entregados al cliente. (incluso cuando hay error)	807
6) Método utilizado en la petición	GET
7) URL requerida - accedida.	http://rad.msn.com/ADSAdClient31.txt
8) Operaciones de Búsqueda de identificación del cliente, generalmente apagado en el proxy Squid.	-
9) Un código que explica cómo fue ejecutada la petición, en el ejemplo se ha descargado directamente del servidor que contenía el objeto.	DIRECT/207.68.178.16
10) El tipo de contenido que se pide en el objeto.	text/html

Tabla 1 Resumen del formato del fichero de trazas del proxy "Access.log"

Hay que mantener la anterior tabla en mente para entender el procedimiento que necesita ser aplicado a este fichero para su utilización en la arquitectura.

2.10 ICP

ICP es el protocolo usado para la comunicación de los proxies Squid. El protocolo ICP está definido en dos RFC, el RFC 2186 que describe el protocolo en sí mismo y el RFC 2187 que describe su uso en las caches web jerárquicas. Para información adicional se puede consular en la bibliografía en la referencia [ii].

Lo que hace ICP con una cache en una jerarquía principalmente es localizar el objeto requerido por el cliente en las caches hermanas. Si la cache hermana no tiene el documento requerido, esta envía una petición a sus hermanas y estas contestan indicando "HIT" (Éxito) o "MISS" (Fallo), a partir de esa información la cache original decide como resolver el objeto WEB requerido. ICP está implementado sobre UDP.

3 ESTADO DEL ARTE

3.1 Herramientas para simulación de Navegaciones WEB y evaluación de prestaciones

- Web Polygraph

Ésta es la herramienta más importante utilizada actualmente para la evaluación de servidores proxy, Polygraph formula una filosofía análoga pero no igual al que se pretende ofrecer con la arquitectura que se plantea.

De [iii]Web Polygraph es una herramienta que consiste de clientes virtuales y servidores trabajando juntos con un fichero de simulación. Clientes (usualmente llamados robots), generan peticiones HTTP para simular objetos. Polygraph soporta los protocolos HTTP/1.0 y HTTP/1.1. Peticiones deben ser enviadas directamente a los servidores o a través de un intermediario (una proxy cache, un balanceador de carga, etc.). Cuando Polygraph se ejecuta mediciones y estadísticas se guardan en fichero de log para su análisis posterior. Polygraph genera trabajo de carga simétrica debido a que puede utilizar clientes reales o trazas de proxy. El equipo de Polygraph dice que el uso de trazas reales no le permite a Polygraph escalabilidad y flexibilidad, una traza real es tratada como un objeto constante.

La configuración de la herramienta web Polygraph es realizada a través de un lenguaje propio llamada PGL, el cual es usado para la configuración de experimentos con una carga de trabajo particular (por ejemplo comportamiento de los robots y servidores). Para enlazar robots y servidores para un proceso en particular, y para configuración adicional como puede ser el logging de la aplicación (como se configuran las trazas para que aparezcan registradas en los LOGs). Mucha de la configuración que hay que realizar con el polygraph se hace para un dominio específico.

Polygraph considera un proxy como una caja negra, Esto significa que todas las medidas son hechas en realidad por la herramienta. Cualquier otra aproximación no es real sobre el funcionamiento del polygraph, de momento no hay una interface estándar que permita probar proxy cache. Incluso se pudiera los resultados pueden ser incorrectos. El hecho que los proxies sean cajas negras para Polygraph significa que no es posible realizar mediciones sobre CPU o por ejemplo niveles de utilización de disco.

3.2 Políticas de WEB Caching

Existen tres tipos de políticas principales de caching: de umbral, en la que no todos los objetos se cachean, su cacheo depende de los parámetros que definen el umbral; adaptativa, en la que el umbral es variable dependiendo de lo que la cache considera que es una mejor configuración; por último está el cacheo de todos los objetos en donde todos los objetos son candidatos a ser cacheados [16].

3.3 Particionamiento de las caches WEB por tamaño de objeto cacheado

El particionamiento de la cache modifica los valores de umbral (threshold), para obtener mejores resultados para la WEB cache, para el caso del tamaño de los objetos el umbral se representaría por el tamaño máximo y mínimo de los objetos susceptibles a ser cacheados.

Para el particionamiento en cuanto al tamaño de los objetos [16] sugiere hacer otro tipo de partición respecto del tamaño de los objetos pequeños, guardando éstos en memoria principal para ser devueltos a los clientes de una forma más rápida, que si son buscados en disco.

Investigaciones anteriores han mostrado que el uso múltiples caches jerárquicas no necesariamente aporta mayores mejoras al uso de pocos niveles de cache, pero [17] ha demostrado experimentalmente que si es posible introducir mejoras en caches jerárquicas particionadas bajo ciertas condiciones. Adicionalmente infiere que las políticas de reemplazo de las caches aportan mejoras a las prestaciones (tasas de aciertos). Aquí también se ve muy claramente que el umbral escogido como tamaño para los objetos cacheados, es muy sensible en la consecución de mejoras de la performance de la cache.

3.4 Ideas generales sobre particionamiento de caches

- Partición por popularidad y prioridad

De [10]. En este artículo los autores proponen un algoritmo para realizar la partición por prioridad y popularidad ("QoS classes") de los objetos cacheados. Hablan que en los experimentos realizados se han dado cuenta que implementar muchas particiones de la cache resulta en tasas de aciertos inferiores que con pocas particiones, lo cual confirma a otro nivel lo concluido por [17]. El algoritmo propuesto hace la partición por prioridad y popularidad para conseguir mayores tasas de aciertos. Han inferido que los dos criterios también tienen grandes efectos sobre el funcionamiento y resultados de la cache web, y han conseguido mejorar las tasas de aciertos respecto una cache web sin particionar. El estudio se ha desarrollado utilizando simulaciones y éstas han demostrado que el nuevo algoritmo supera las desventajas de utilizar uno solo de los criterios en la partición de la WEB cache.

La partición con la mayor esperanza de conseguir mejores resultados se le asigna más espacio en la cache, por tanto, se proporciona mayor disponibilidad a las páginas WEB más importantes, redundando en mayores aciertos. (En otras palabras para objetos populares se garantiza una tasa de aciertos mayor). Los objetos impopulares serán reemplazados más pronto, permitiendo a los populares quedarse más tiempo disponibles en cache.

- Partición por paralelismo

En [11], se habla sobre el paralelismo como forma de partición de las caches, pero sin perder de mente las limitaciones que ésta puede producir para el sistema.

El paralelismo de nivel de instrucción (ILP) ha motivado el empleo de paralelismo de nivel de hilo (TLP) como una estrategia común para mejorar el funcionamiento de procesador.

TLP paradigmas como el procesamiento multihilo (SMT), el multiprocesamiento con los chips (CMP) y las combinaciones de ambos ofrecen la oportunidad de obtener rendimientos más altos. Sin embargo, ellos también tienen que afrontar el desafío de compartir los recursos de la arquitectura. De otro lado la evitación de cualquier control de recurso puede conducir a situaciones indeseadas donde un hilo monopoliza todos los recursos y daña otros hilos.

- Particionamiento por el uso que se le dan a la arquitectura

En [12], han inferido que es importante conocer exactamente que la clase de usos se le dará a la arquitectura para decidir correctamente qué mecanismo de partición dinámico es apropiado. Los autores han propuesto la partición de la cache como una alternativa para mejorar las políticas desahucio tradicionalmente usados en los niveles de cache compartida en arquitecturas modernas CMP: el rendimiento es mejorado a cargo de un coste razonable. Sin embargo, estos la nueva política presenta comportamientos diferentes dependiendo los usos que son dados a la arquitectura.

En [13], Este artículo investiga el problema de particionar una cache compartida entre múltiples ejecuciones de aplicaciones simultáneas. La política comúnmente usada LRU implícitamente divide un cache compartido en base a la demanda, dando más recursos del cache a los objetos que tienen una alta demanda y menos recursos de cache a los objetos que tiene una demanda baja.

Se presenta la partición de la cache por el concepto de utilidad UCP ("Utility based, Cache Partitioning"), se proveen diferentes formas de cachear los objetos dependiendo de la aplicación que se esté ejecutando sobre la cache, buscando soluciones diferentes para problemas heterogéneos. Comparan diferentes algoritmos de partición, obteniendo mejores resultados con UCP. Los conceptos de este artículo podrían ser utilizados para el particionamiento de las caches web.

- Particionamiento adaptativo

De [14]. Este trabajo explora una técnica de individualización de las caches, el estudio es aplicado a procesadores, pero podría ser exportado a las caches web. Los autores han distribuido los accesos en las diferentes particiones de cache habilitando a éstas a ser reprogramables en tamaño.

Una cache dedicada a una tarea en particular podría evitar colisiones con objetos de otras caches de otras tareas que se estén ejecutando. La parte complicada consiste en la

forma de definir la metodología correcta para realizar esa partición de la cache, ya que son recursos limitados.

- Particionamiento estático

De [15]. Este paper habla sobre una técnica de partición de la cache de manera estática para cada tarea, intentando buscar un máximo de predicción de lo que debería ser el parámetro usado en la partición para obtener mayores aciertos con la cache.

Esta técnica de partición minimiza el coste del caching porque predefine los parámetros a utilizar en las particiones, introduce mejoras en aciertos y prestaciones a partir de bajo coste del proceso de reemplazo y almacenaje.

4 METODOLOGÍA

Para el desarrollo de la metodología general se han utilizado métodos tanto empíricos como prácticos, para el planteamiento de la arquitectura que puede ser utilizada para simular las navegaciones WEB, la prueba y error han sido fundamentales para descartar muchas veces enfoques erróneos seleccionados y todo lo que será planteado es fruto de mucha reflexión y discernimiento

Lecturas de artículos, manuales, tutoriales, han permitido formar el conocimiento suficiente para hacer un planteamiento como el que se propone, adicionalmente la metodología es lo suficientemente especifica, para poder proveer una respuesta concreta.

Se va a definir todo lo necesario para la obtención del modelo de simulación de navegaciones web a través de proxies, para ello se describirán todos los elementos involucrados, definición, montaje, funcionamiento, y puesta en marcha de la arquitectura.

A partir de este punto se va a desarrollar esta metodología, para introducir el método científico en la definición de la arquitectura.

4.1 Procedimiento

Lo primero para definir es el inventario de todas las herramientas que pueden servir en la consecución de la configuración de la arquitectura, para ello es necesario pensar desde el sistema operativo en que se va a montar todo, hasta los posibles lenguajes de programación que se utilizarán, puede que la lista de herramientas sea grande al principio pero con el desarrollo de la arquitectura la lista se irá acotando.

Lo siguiente a realizar, será hacer el planteamiento funcional de lo que puede ser la arquitectura, para ello se intentará centrar al lector en el concepto más general de todos los módulos o partes que pueda constituir la arquitectura con su posible configuración y funcionamiento.

A continuación partiendo de lo que se habrá propuesto como arquitectura, se propondrá una configuración concreta que permita ser estudiada y configurada técnicamente en el desarrollo de la evaluación de la arquitectura de este documento.

A continuación se detallarán todos los aspectos técnicos que acarrea la arquitectura, exponiendo todas y cada una de las herramientas, técnicas, instrumentos, que serán necesarios para la puesta en marcha de la misma, se encuentran detalladas específicamente en los “Anexos” adjuntos al documento.

Por último una vez se haya conseguido probar que la arquitectura funciona, se puede disponer a evaluar la arquitectura propuesta.

4.2 Técnicas e instrumentos utilizados

El siguiente postulado muestra un inventario de todo lo necesario para la el planteamiento de la arquitectura, se describirán brevemente las herramientas hardware, software, lenguajes de programación y demás técnicas utilizadas.

4.2.1 Herramientas hardware y software utilizadas en la arquitectura

Las herramientas utilizadas para la arquitectura y su análisis fueron las siguientes:

Hardware

- Un ordenador con procesador compatible con Suse Linux o alguna distribución Linux. Aunque se estén trabajando con varios servidores proxy y servidores web, es posible realizar el montaje en un sólo ordenador.

Software

- Sistema Operativo Suse Linux 9.2 /Cygwin 1.5.25-14 [1]
- Servidor WEB Apache Tomcat versión 5.0.28
- Sun Microsystems JDK 1.4.0.12
- Cliente HTTP GNU Wget 1.11.3
- IDE Eclipse 3.2 [2]
- GNU NotePad ++ [3]
- Conversor Hora UTC [4]
- Servidor Proxy Squid v 2.5 [iv]

Lenguajes de Programación y Tecnologías programación utilizadas

- AWK
- Java J2SE
- Programación Batch
- Jsps (Java Server Pages)
- Servlets

Otros

- Access logs (trazas) de los proxies de la Universidad Politécnica de Valencia

[1] http://www.cygwin.com/
[2] www.eclipse.org
[3] notepad-plus.sourceforge.net
[4] http://software.ccschmidt.de/#UTC

4.3 Diseño funcional de la arquitectura

4.3.1 Notación general de la arquitectura planteada

Como punto de partida para ir introduciendo un poco la arquitectura que se ha escogido para la simulación de las navegaciones WEB a través de una jerarquía de servidores proxy, se presenta a continuación esquemáticamente como será notada la arquitectura, para no perder de vista en ningún momento los elementos que la componen.

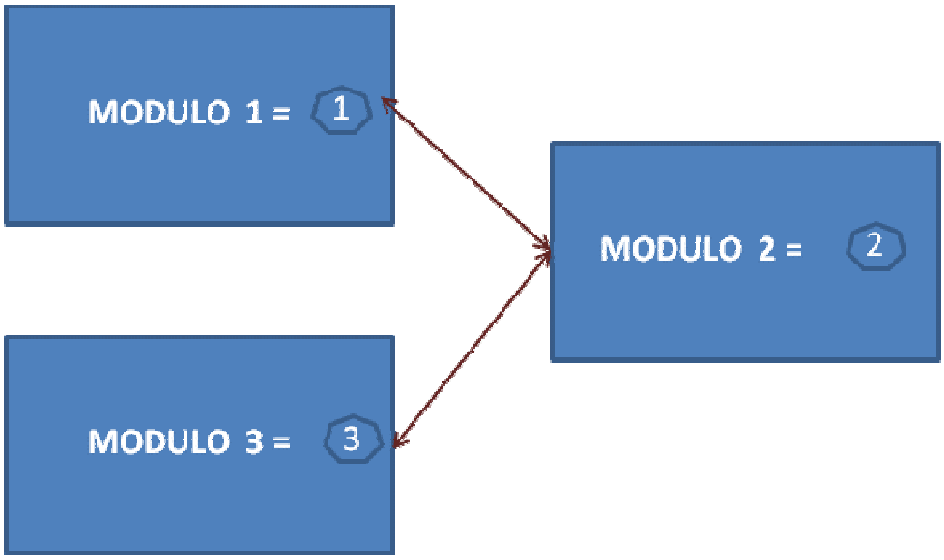

Ilustración 1 Notación General de la arquitectura

La notación que se está definiendo en esta sección esquematiza de manera general los módulos principales que componen la arquitectura; en ella se reconocen 3 módulos principales interconectados entre sí. Será muy importante tener siempre los módulos en mente para saber a qué pieza de la arquitectura se está refiriendo en cada una de las secciones e ilustraciones del documento que se tratarán más adelante.

4.3.1.1 Formas abreviadas para denotar los módulos de la arquitectura

Los heptágonos con su número, corresponden a una forma abreviada de denominar al módulo siendo siempre:

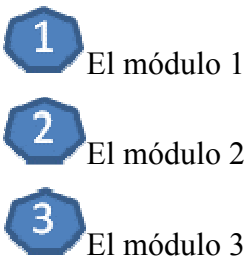

Ilustración 2 Notación general de los módulos de la arquitectura

4.3.2 Esquema general de la arquitectura

La arquitectura como se pudo ver en la sección del documento 4.3.1 consta de tres módulos, los cuales son se pueden apreciar en la siguiente ilustración:

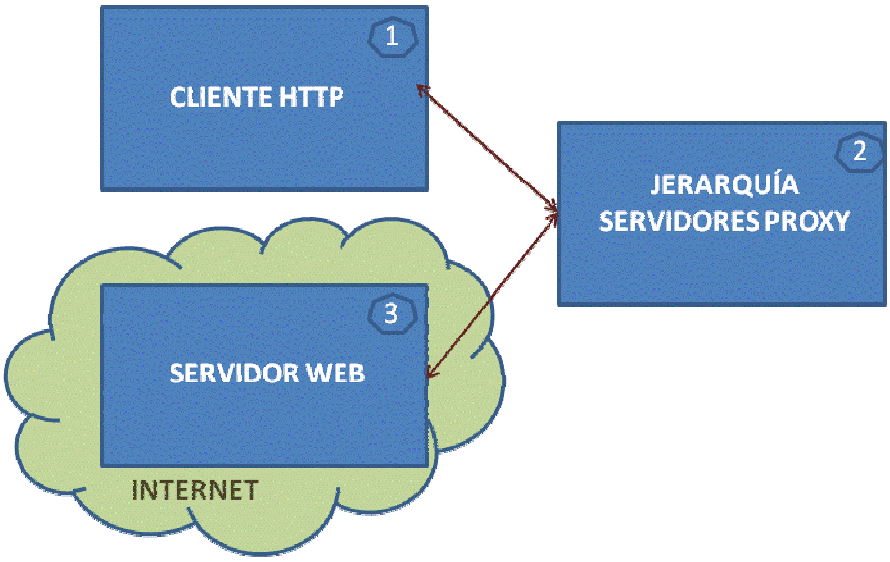

Ilustración 3 Esquema General de la Arquitectura

Ahora podemos reconocer que esos tres módulos descritos en la sección 4.3.1 tienen ahora un significado diferente, de manera general y para alguien que no posea mucho conocimiento de lo que el esquema representa, puede iniciar imaginándose como un usuario de internet, entonces sin saberlo, su papel es el de ser el Cliente HTTP (Módulo 1), de otro lado en casi toda intranet que accede a internet, por ejemplo la red de una universidad, se emplazan justo antes de internet, servidores proxy (Módulo 2) que se encargan de gestionar las peticiones de los clientes HTTP y por último se tiene a internet que al final es el sitio donde se encuentra la información que el cliente quiere (Módulo 3).

En resumen los 3 componentes fundamentales de la arquitectura serán: el primero de ellos el cliente Http que es el encargado de realizar las peticiones de los objetos web, el segundo la jerarquía de servidores proxy que se encargan de recibir las peticiones del cliente http y gestionarlas para al final devolver el objeto solicitado, por último tenemos Internet y como una sub parte de internet el tercer módulo el servidor web, será el encargado de generar los objetos que sean solicitados por el cliente.

Esta arquitectura está diseñada para simular las navegaciones WEB a través de una jerarquía de servidores proxy, para realizar el análisis del comportamiento de sus caches, a partir de ahora se van a ir explicando los módulos y su interacción.

4.3.3 Módulo 1 Cliente HTTP

- Cliente HTTP GNU Wget, es el encargado de realizar las peticiones de objetos WEB consecutivas.

4.3.4 Módulo 2 Jerarquía de servidores proxy

- Servidores proxy Squid versión 2.5, la cache WEB

4.3.5 Módulo 3 Servidor Web

- Servidor Apache Tomcat versión 5.0.28: Este servidor Web sigue las referencias oficiales para implementaciones de las tecnologías de Java Servlets y de Java Server Pages (JSPs). Sobre este servidor se montará la aplicación web que generará los objetos web necesarios para el cliente.
- JDK 1.4.0.12, es la base para el desarrollo de la aplicación web y para su ejecución sobre el servidor Tomcat.

4.3.6 Diseño software de la arquitectura

Sistema Operativo Suse Linux 9.2: Sobre este sistema operativo se ha montado la arquitectura (los tres módulos). Ha sido escogido debido a la fácil interfaz gráfica que ofrece, a continuación se presentan discriminados por software como están implementados los módulos de la arquitectura.

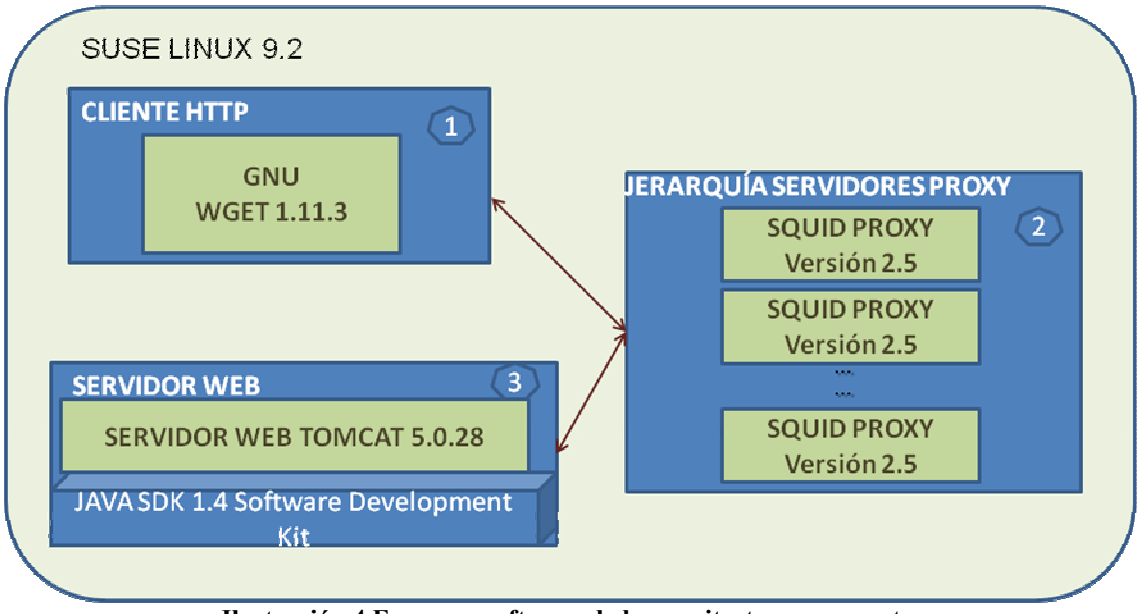

Ilustración 4 Esquema software de la arquitectura propuesta

Aparte de la distribución SUSE Linux, cualquier otra distribución puede ser utilizada siempre que permita compilar las versiones de los programas y aplicaciones utilizados, para algunos experimentos se ha comprobado que sobre "Cygwin" se puede hacer la implementación completa de la arquitectura.

4.3.7 Protocolos utilizados en la arquitectura

Los siguientes son los principales protocolos que se usarán en la arquitectura cuando esté en funcionamiento:

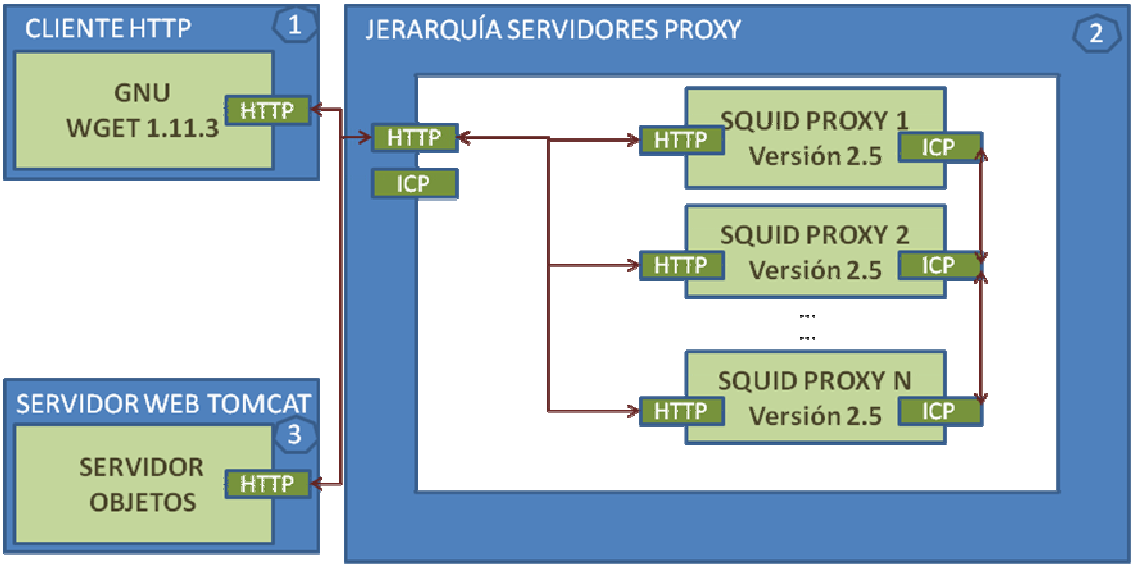

Ilustración 5 Principales protocolos utilizados en la arquitectura

En general la arquitectura utiliza los protocolos http para requerir y servir los objetos WEB y de otro lado está el protocolo ICP utilizando por los proxies para la comunicación con sus hermanos, hijos o padres.

Como puede ser visto, la arquitectura permitirá que una petición que realice el cliente HTTP, pueda ir a cualquiera de los servidores proxy configurados en la arquitectura, con lo cual funcionalmente la arquitectura es altamente escalable.

4.4 Elección de la arquitectura final a utilizar en el desarrollo de la evaluación de la arquitectura

Hasta este punto del diseño funcional de la arquitectura se ha hablado de manera genérica de los módulos y sus componentes, a partir de este punto se concretarán que tipo de configuración de servidores proxy será utilizada en el módulo 2 y todo lo requerido para poder realizar una simulación.

La siguiente son las configuraciones propuestas para el módulo 2 de la arquitectura las cuales se ampliaran y estudiarán en la "Evaluación de la arquitectura" de este documento.

4.4.1 Propuesta de configuración de la arquitectura servidores con proxies en jerarquía

La siguiente será la configuración del módulo 2 de la arquitectura en jerarquía, estará constituida por 2 servidores proxy squid a los que se les analizará el comportamiento de sus caches web.

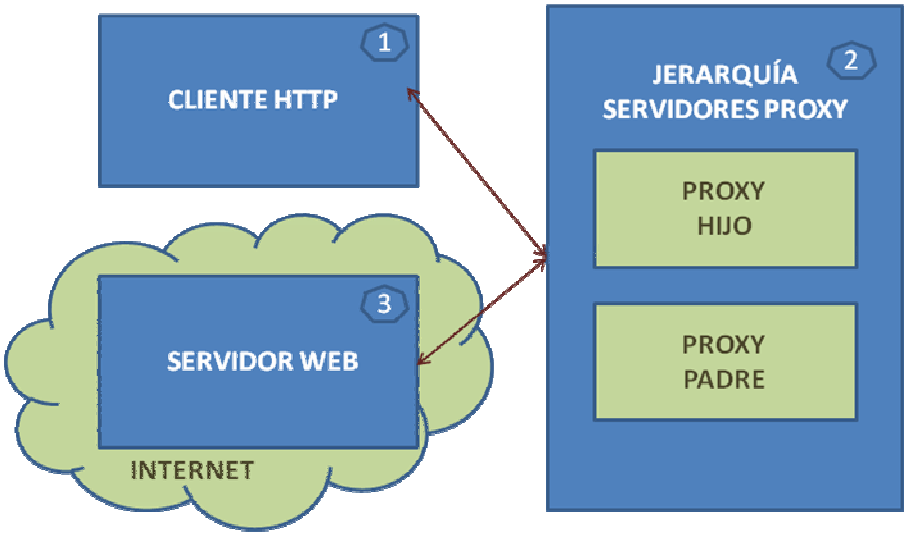

Ilustración 6 Esquema general de la arquitectura escogida en jerarquía de proxies a ser estudiada en la evaluación de la arquitectura

La anterior configuración permitirá que las caches web queden configuradas de la siguiente manera:

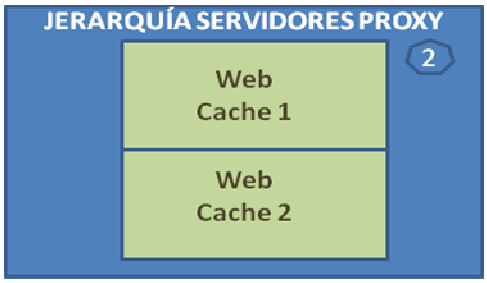

Ilustración 7 Propuesta de configuración de las caches WEB para las mediciones a realizar en la evaluación de la arquitectura

Dado que cada uno de los proxies tiene una cache web para los objetos que procesa, se va a utilizar ello, para evaluar si dos caches web pueden obtener mayores aciertos que una sola.

A continuación se expondrá la manera como se realizarán las peticiones contra esta configuración de servidores proxy, para entender la forma como funcionarán sus caches web.

4.4.1.1 Secuencia de petición de un objeto web, contra la configuración propuesta en jerarquía del módulo 2 de la arquitectura

Para la configuración propuesta, todas las peticiones realizadas por el cliente WEB serán hechas a través del proxy hijo, con lo cual en la siguiente ilustración expondrá la manera como se procesará una petición de un objeto WEB con la configuración padre-hijo de los servidores proxy.

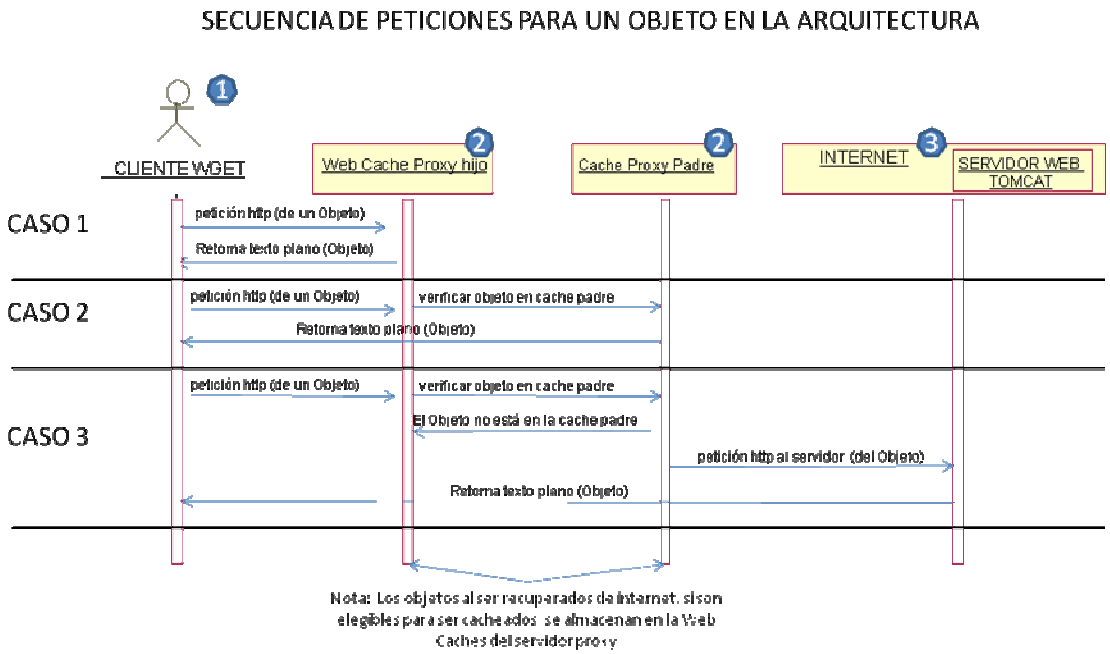

Ilustración 8 Secuencia de petición de un objeto web, contra la configuración de la arquitectura propuesta en jerarquía

En la "Ilustración 8" se observa la casuística seguida en la petición de un objeto web contra la configuración en jerarquía del módulo 2 de la arquitectura:

- Caso 1: El cliente HTTP (WGET) realizará una petición de un objeto web a la arquitectura a través del proxy hijo, con lo cual, si el proxy hijo tiene en su caché web el objeto, éste le será servido directamente sin necesitad de ir a internet.
- Caso2: El cliente HTTP (WGET) realizará una petición de un objeto web a la arquitectura a través del proxy hijo, si la cache web del hijo no tiene cacheado el objeto, el proxy hijo pregunta al padre si tiene guardado el objeto requerido, si el padre lo tiene guardado en su cache web, entonces lo enviará cliente HTTP a través del proxy hijo.
- Caso3: Para el último caso si el objeto no se encuentra cacheado en ninguna de las dos caches web, el principio de la secuencia de la petición será como se ha visto en el caso 2, pero el padre responderá al hijo que no tiene el objeto web requerido, entonces, se pedirá directamente el objeto a internet (servidor TOMCAT), si el objeto requerido es del tamaño aceptado por las caches web, se cacheará, este es el precepto fundamental que permitirá que se puedan hacer mediciones sobre el funcionamiento de las caches web.

4.4.1.2 Protocolos utilizados en la configuración de la arquitectura con proxies en jerarquía

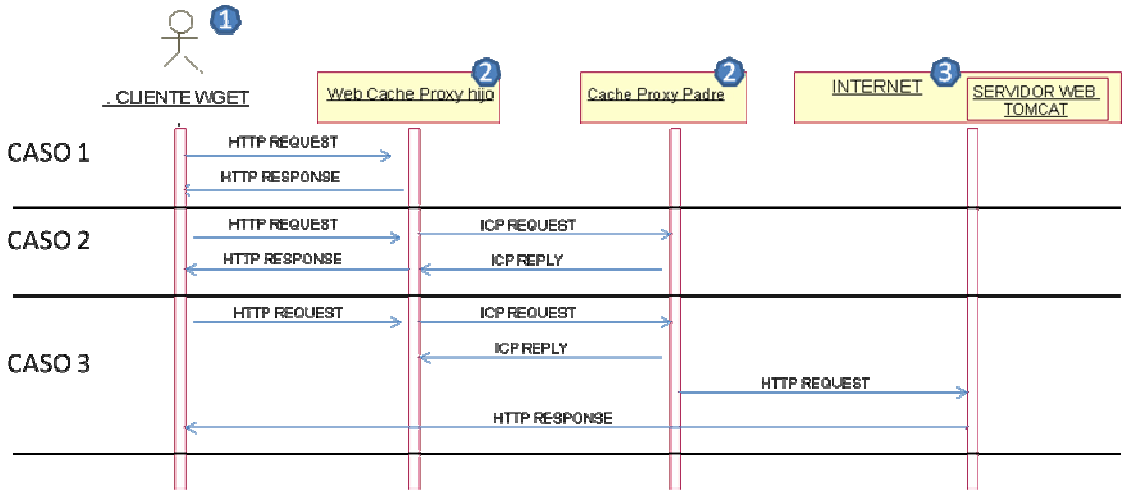

Ilustración 9 Principales protocolos utilizados en la jerarquía propuesta

En la ilustración anterior se observan los principales protocolos utilizados siguiendo mismos tres casos anteriormente explicados, en el caso 1 sólo se habrán realizado peticiones HTTP, para el caso dos la comunicación entre el proxy hijo y el cliente HTTP se hace mediante HTTP y la comunicación entre las caches se hace mediante ICP, por último, en el caso 3 las peticiones realizadas a internet se hacen por HTTP.

4.4.1.3 Aspectos técnicos de la propuesta

Se hace necesario tener configurados dos servidores proxy para el procesamiento de las peticiones HTTP de la arquitectura, para ello a continuación se enseñan con un ejemplo las principales propiedades [5] configuradas en el "squid.conf" del proxy Padre y del Proxy Hijo.

SQUID PADRE:

```
http_port 127.0.0.1:3128
icp_port 3130
cache_peer 127.0.0.1 sibling 3129 3131
maximum_object_size 20 KB
minimum_object_size 0 KB
memory_replacement_policy lru
cache_dir ufs /usr/local/squid1/var/cache 100 16 256
icp_access allow all
visible_hostname miproxyuno
unique_hostname padre
strip_query_terms off
```

[5] Se han configurado muchas más propiedades para el tunning de la arquitectura, pero las propiedades principales son las que se exponen; para información adicional del resto de propiedades, por favor consultar los ficheros "squid.conf" que se adjuntan al documento.

SQUID HIJO:

```
http_port 127.0.0.1:3129
icp_port 3131
cache_peer 127.0.0.1 parent 3128 3130
maximum_object_size 4096 KB
minimum_object_size 20 KB
memory_replacement_policy lru
cache_dir ufs /usr/local/squid2/var/cache 100 16 256
icp_access allow all
visible_hostname miproxydos
unique_hostname hijo
strip_query_terms off
```

De las anteriores propiedades infiere lo siguiente:

- El puerto HTTP que será utilizado por el proxy padre/hijo para escuchar las peticiones del cliente
- El puerto ICP que será utilizado para la comunicación con las caches de la jerarquía,
- El tamaño mínimo/máximo de los objetos que serán aceptados por la cache web. En el ejemplo de configuración del proxy padre se ve que este almacenará en su cache los objetos pequeños (La sección resaltada en amarillo 0-20 KB). De otro lado en la configuración expuesta para el proxy hijo se aprecia que éste almacenará los objetos de mayor tamaño. (Almacenará objetos desde 20-4096 KB). Entonces, se puede concluir que la jerarquía funcionará como sola caché que aceptará objetos de 0-4096 KB, este precepto será utilizado en el desarrollo de la evaluación de la arquitectura de este documento, dado que permite configurar la cache en jerarquía variando el tamaño máximo de objeto aceptado por el padre y el mínimo aceptado por el hijo.
- La política de reemplazo de objetos guardados en la cache, utilizada por el servidor proxy.
- El tamaño de la cache web. En este ejemplo las caches web tendrían un tamaño de 100 MB. La jerarquía de proxies padre-hijo se comportaría como si fuera una sola cache web de 200 MB.
- Se declara que se va a aceptar todas las peticiones ICP provenientes de otros proxies hermanos.
- Los nombres dados al proxy.
- Se declara que en los ficheros de las trazas del proxy se almacenen toda la URL de los objetos procesados.

4.4.2 Propuesta de configuración de la arquitectura con servidor proxy de referencia

Adicionalmente a esta configuración de la jerarquía, se propone que en el "módulo 2" de la arquitectura se utilice un proxy con la configuración por defecto para las mediciones de referencia, de la siguiente manera:

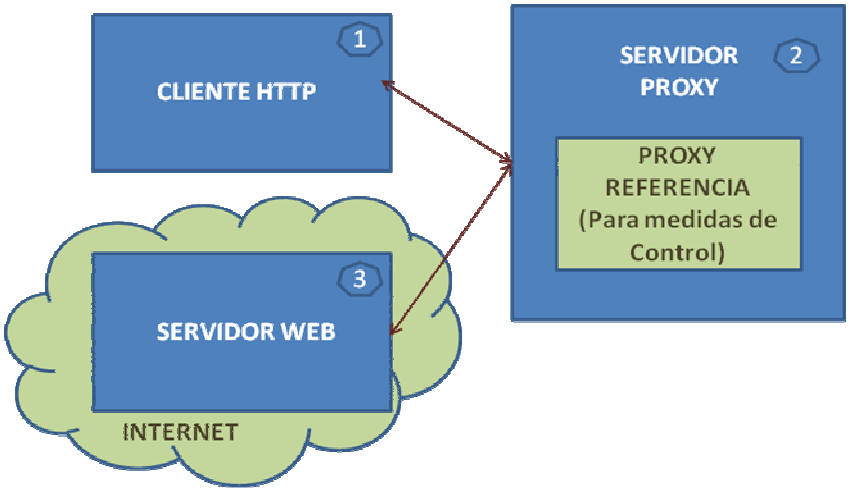

Ilustración 10 Esquema general de la arquitectura escogida de referencia a ser estudiada en la evaluación de la arquitectura

La "Ilustración 10" muestra que configuración de la arquitectura se usará para la realización de las medidas de control en el desarrollo de la evaluación de la arquitectura propuesta en este documento, compuesto de un proxy squid de referencia ("de referencia", con la configuración por defecto), para la realización de las mediciones de control, con el objetivo de comparar sus resultados con los obtenidos con la configuración en jerarquía de servidores proxy.

Esta configuración hará que la arquitectura trabaje en el módulo 2 como se puede ver en la siguiente ilustración:

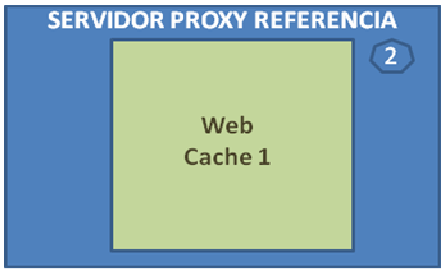

Ilustración 11 Propuesta de configuración de las caches WEB para las mediciones de control a realizar en la evaluación de la arquitectura

Al configurar el módulo 2 de la arquitectura de esa forma, las simulaciones ejecutadas evaluarán las prestaciones de esa cache web.

4.4.2.1 Secuencia de petición de un objeto web, contra la configuración propuesta como referencia del módulo 2 de la arquitectura

A continuación se explicará la secuencia de una petición de un objeto web a esta configuración de la arquitectura.

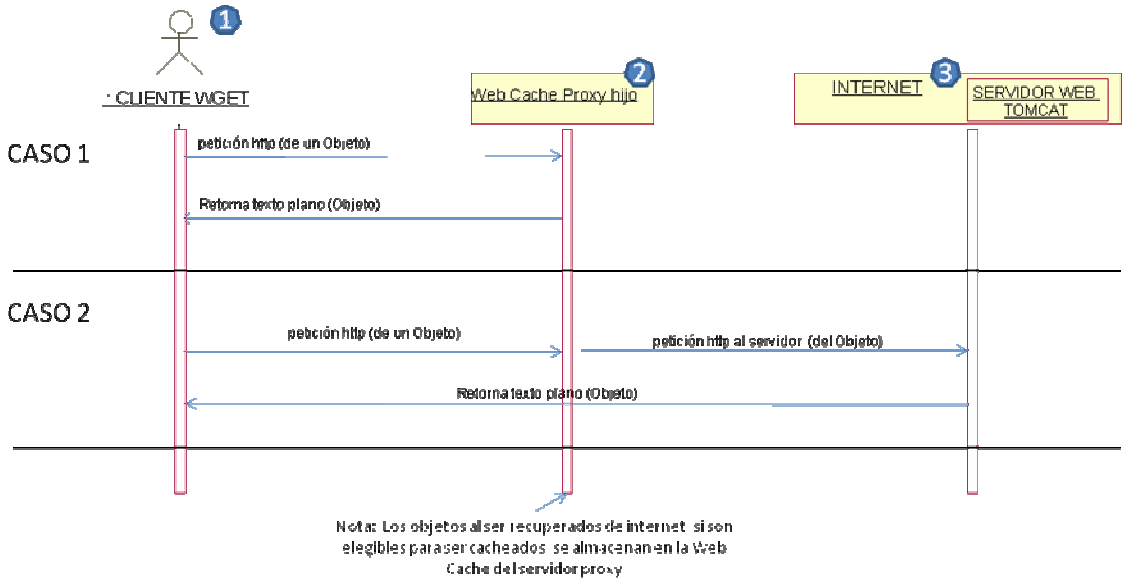

Ilustración 12 Secuencia de petición de un objeto web, contra la configuración de la arquitectura propuesta como referencia

En el caso uno se pide el objeto al proxy de referencia, si este tiene cacheado el objeto lo sirve al cliente, en el caso 2, si el objeto requerido por el cliente no está en la caché del proxy, lo pide al servidor de internet donde esté alojado el objeto y al final el objeto es enviado al cliente. Si el objeto requerido es del tamaño aceptado por la cache web entonces se cacheará.

4.4.2.2 Protocolos utilizados en la configuración de la arquitectura con proxies de referencia

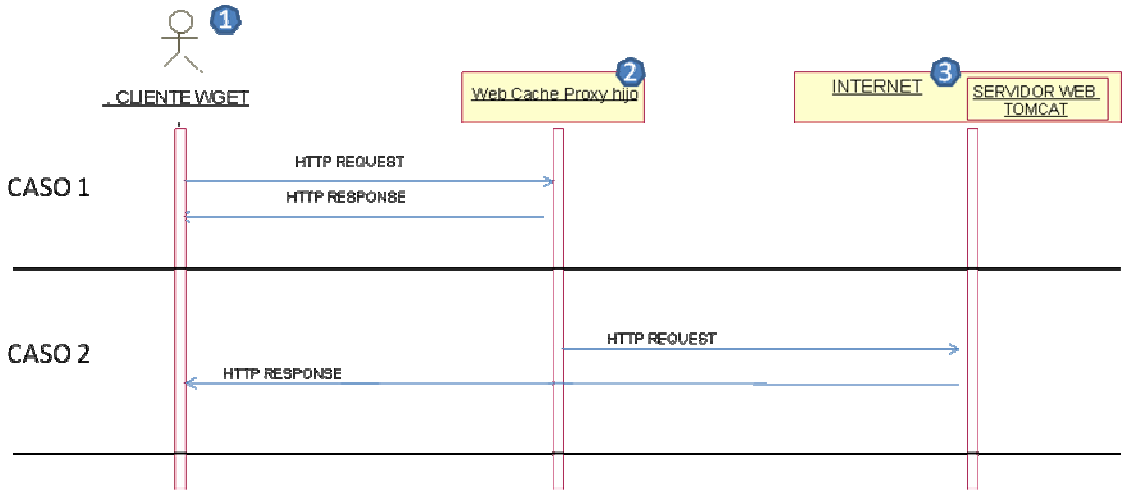

Ilustración Protocolos utilizados en la configuración de la arquitectura con proxies de referencia

Como puede ser visto en la secuencia de las peticiones, el protocolo predominante en la arquitectura configurada de esa forma es el HTTP.

4.4.2.3 Aspectos técnicos de la propuesta

A continuación se presentan las principales propiedades configuradas para el servidor proxy de referencia.

SQUID REFERENCIA

```
http_port 127.0.0.1:3127
icp_port 3132
minimum_object_size 0 KB
maximum_object_size 4096 KB
memory_replacement_policy lru
cache_dir ufs /usr/local/squid3/var/cache 200 16 256
visible_hostname miproxytres
unique_hostname normal
strip_query_terms off
```

De las anteriores propiedades infiere lo siguiente:

- El puerto HTTP que será utilizado por el proxy referencia para escuchar las peticiones del cliente. (debe ser diferente a los utilizados por para la configuración en jerarquía)
- El tamaño mínimo/máximo de los objetos que serán aceptados por la cache web. Aceptará objetos de un tamaño entre (0-4096 KB).
- La política de reemplazo de objetos guardados en la cache, utilizada por el servidor proxy.
- El tamaño de la cache web. En este ejemplo las cache web tendrá un tamaño de 200 MB
- Los nombres dados al proxy.
- Se declara que en los ficheros de las trazas del proxy se almacenen toda la URL de los objetos procesados.

4.5 Técnicas utilizadas para la preparación del fichero de simulación

Ahora que ya se tiene definidas las configuraciones de la arquitectura que serán evaluadas en el desarrollo de la evaluación de la arquitectura de este documento, lo siguiente será ver todo lo necesario para poner en funcionamiento la arquitectura.

A manera de ejemplo se va a explicar de manera general cómo se realiza el tratamiento de un fichero “Access.log” de la UPV el cuál será usado como punto de partida para la confección del fichero a ser ejecutado en las simulaciones contra la arquitectura planteada. Para información en detalle del procedimiento seguido puede consultar los “Anexos” adjuntos.

A continuación se enseña un ejemplo de una petición http que ha sido registrada dentro de estos ficheros de log de la UPV, para contrastarla con las que se usarán y registrarán en la arquitectura propuesta.

4.5.1 Ejemplo del fichero "Access.log" del proxy Squid de la UPV para una petición http

Un registro de un objeto consultado a través del proxy de la universidad politécnica de Valencia deja una traza como la enseñada a continuación:

```
1137279490.298 49 158.42.245.11 TCP_MISS/200 25358 GET
http://www.jerc.net/eivissa_p.jpg - DIRECT/217.113.244.173 image/jpeg
```

Ilustración 13 formato general del fichero de Access.log del servidor proxy squid.

El reto en este punto consiste en transformar este tipo de trazas del Access.log del squid en un tipo de petición http que pueda utilizarse como entrada para la arquitectura propuesta en este documento. Los dos campos principales a utilizar de la traza son los resaltados en color amarillo, dado que para hacer una petición contra la arquitectura necesitamos simular las diferentes peticiones HTTP que hay registradas con el tamaño de cada una de ellas.

4.5.2 Transformando las trazas del Access.log de la UPV en un fichero de peticiones http para ser utilizado en las simulaciones contra la arquitectura

Se hace fundamental tener las herramientas para transformar las trazas que se tienen en los ficheros "Access.log" de los proxies squid de la UPV, en un fichero que pueda ser utilizado para simular navegaciones HTTP contra la arquitectura propuesta, para ello a continuación se enseñará el formato de las peticiones a utilizar en las simulaciones.

4.5.2.1 Formato general de las peticiones http que serán usadas contra la arquitectura

El siguiente es el modelo de petición HTTP básico que se ejecutará contra la arquitectura propuesta:

```
http://127.0.0.1:8080/lasletras/miServlet?URL=URL&Size=TamañoBytesObjeto
```

Ilustración 14 Formato general de una petición HTTP a usar en la arquitectura

Para componer esta petición sólo se necesitan dos campos de cada una de las líneas registradas en el fichero "Access.log" de los Squid de la UPV, estos son la URI del objeto solicitado y el tamaño del objeto en Bytes (resaltado en amarillo).

4.6 Lanzando simulaciones contra la arquitectura

La siguiente es la idea general de una simulación contra la arquitectura, a continuación se muestra el procedimiento de una simulación:

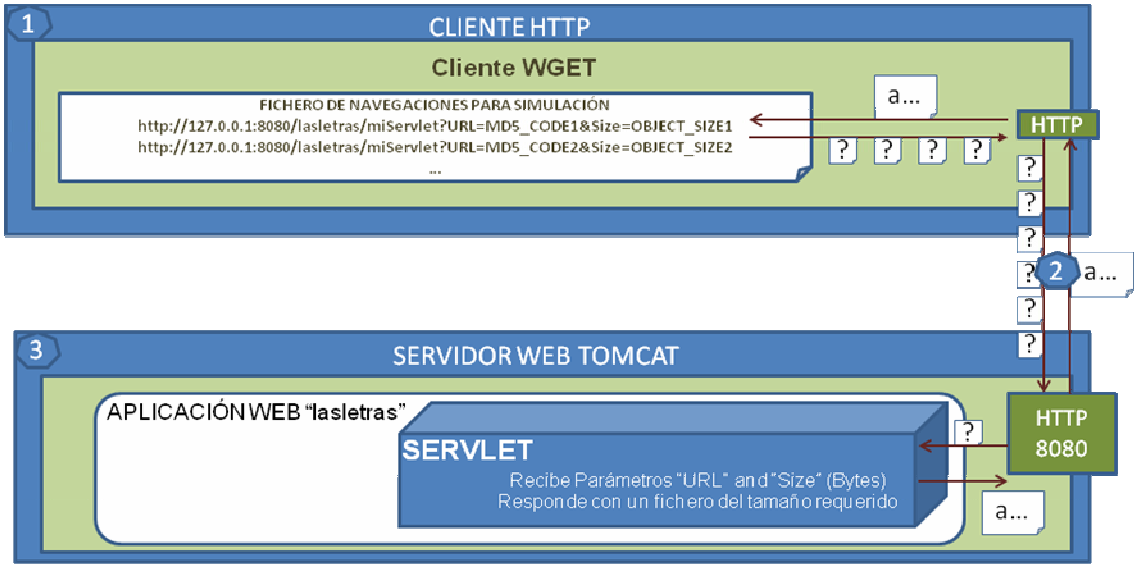

Ilustración 15 Esquema general de una simulación contra la arquitectura

Para lanzar una simulación contra la arquitectura final seleccionada, el cliente requerirá los objetos http al servidor Tomcat a través del módulo 2 de la arquitectura (jerarquía de servidores proxy) y los objetos resultantes van a volver por el mismo camino.

Mientras los objetos fluyen entre el cliente http y el servidor los proxies van cacheando los objetos que le son relevantes. Para información en detalle del lanzamiento de las simulaciones puede consultar los "Anexos" adjuntos.

4.7 Análisis de los resultados obtenidos por la arquitectura

Para el análisis de los resultados obtenidos en los logs del proxy se han evaluado muchas herramientas, dado que se necesita que la herramienta escogida sea lo suficientemente moldeable para personalizarla en medir sólo los datos que interesan, se ha escogido la herramienta "scalar.awk" creado por el profesor Yuri N. Forminov [6], este script ha sido modificado para una lectura más rápida de los resultados relevantes.

Para este documento solo va a ser necesario analizar el fichero de log del proxies llamado "Access.log".

4.7.1 Unidades utilizadas en los análisis de resultados de las simulaciones

Dado que los datos que se están midiendo pueden ser cantidades por ejemplo expresadas en miles, millones, porcentajes; es conveniente tener definidas unidades que permitan un análisis de los resultados más coherente y sencillo a los ojos del lector.

[6] SCALAR (Squid Cache Advanced Log Analyzer & Reporter) produces many detailed reports, such as: Time Based Load Statistic, Extensions Report, Content Report, Object Sizes Report, Request Methods Report, Squid & HTTP Result Codes Report and Cache Hierarchy Reports - most of reports are splitted on Requests, Traffic, Timeouts and Denies statistic. SCALAR is highly customizable tool/script written on AWK - all setting can be defined inside script header. SCALAR developed by Yuri N. Fominov.

Las siguientes serán las unidades a utilizar con cada una de los análisis que se hagan de la arquitectura[7]:

PARÁMETRO	UNIDAD
TCP_MEM_HIT	Kilo hits (K)
TCP_HITs	Kilo hits (K)
Hits Total	Kilo hits (K)
TOTAL HITS VOLUME	Porcentaje (%)
TCP_MEM_HIT	Megabyte (MB)
TCP_HIT	Megabyte (MB)
Saved Traffic	Megabyte (MB)
Saved Traffic	Porcentaje (%)

Tabla 2 Unidades utilizadas en los análisis de resultados

4.7.2 Errores en mediciones realizadas por la arquitectura

El error ha sido medido realizando 5 mediciones iguales en diferentes fracciones de tiempo contra las dos configuraciones de los proxy (Modulo 2 de la arquitectura) propuestas en el numeral "4.4" a ser evaluadas en el desarrollo de la evaluación de la arquitectura, el fichero de simulación utilizado es el mismo que será utilizado para el desarrollo de la evaluación de la arquitectura.

4.7.2.1 Error medido para la arquitectura configurada con jerarquía de proxies (Padre-Hijo)

El objetivo final de esta configuración es el de analizar cómo se comportan las dos caches web del proxy padre e hijo al realizar una simulación con el fichero extraído del proxy de la UPV.

A continuación se analizan los resultados obtenidos de 5 mediciones iguales sin alterar parámetros de configuración, cada valor consignado en la tabla representa la suma de los resultados obtenidos en la cache del padre + cache del hijo. Dicho de otro modo los resultados obtenidos en este punto, se entienden como los resultados obtenidos con la jerarquía de proxies.

Simulación	TCP_MEM_HIT en K	TCP_HIT en K	Total Hits en K	TOTAL HITS VOLUME %	TCP_MEM_HIT MB	TCP_HIT MB	Saved Traffic MB	Saved Traffic %
1	113,732	98,56	212,292	50,4676	109,603	153,283	262,885	7,0929
2	113,713	98,576	212,289	50,4669	109,583	153,291	262,874	7,0926
3	113,75	98,563	212,313	50,4726	109,61	153,289	262,899	7,0933
4	113,768	98,553	212,321	50,4745	109,618	153,276	262,894	7,0932
5	113,743	98,577	212,32	50,4743	109,596	153,298	262,887	7,093
Promedio	113,7412	98,5658	212,307	50,4712	109,602	153,2874	262,8878	7,093
Desviación Estándar	0,018323755	0,01042593	0,015411	0,00366	0,013397761	0,008325	0,009523655	0
Error %	**0,008194632**	**0,00466262**	**0,006892**	**0,00164**	**0,005991661**	**0,003723**	**0,004259108**	**0**

Tabla 3 Error de la arquitectura propuesta con configuración en jerarquía

[7] La información que se va a mostrar en el fichero resultado es la que se considera de momento como relevante. Pero puede ser modificado para futuras investigaciones añadiendo o quitando datos.

De la tabla anterior se infieren los valores de error para cada una de las unidades utilizadas en las mediciones. Este error servirá para analizar los resultados y postulados de la "evaluación de la arquitectura" de este documento.

4.7.2.2 Error medido para la arquitectura configurada con el servidor proxy de referencia

Para el sistema de referencia se han realizado también 5 mediciones con los mismos parámetros de configuración y los resultados obtenidos son los siguientes:

Medición	TCP_MEM_HIT en K	TCP_HIT en K	Total Hits en K	TOTAL HITS	TCP_MEM_HIT MB	TCP_HIT MB	Saved Traffic MB	Saved Traffic
1	113,276	97,965	211,241	50,2178	78,408	113,844	192,252	5,1872
2	113,258	97,98	211,238	50,217	78,389	113,852	192,241	5,1869
3	113,295	97,967	211,262	50,2228	78,416	113,849	192,265	5,1875
4	113,313	97,957	211,27	50,2247	78,424	113,837	192,261	5,1874
5	113,288	97,981	211,269	50,2244	78,402	113,852	192,254	5,1872
Promedio	113,286	97,97	211,256	50,2213	78,4078	113,8468	192,2546	5,1873
Desviación Estándar	0,020603398	0,01029563	0,015411	0,00366	0,01338656	0,00638	0,0092358	0,0002
Error %	**0,00921412**	**0,00460435**	**0,006892**	**0,00164**	**0,005986652**	**0,002853**	**0,004130375**	**0,0001**

Tabla 4 Error de la arquitectura propuesta con configuración de referencia

Los resultados obtenidos dejan ver un error bastante bajo entre simulaciones que aporta un gran intervalo de confianza para cualquier tipo de simulación realizada con la arquitectura propuesta. Estos errores obtenidos se usarán como referencia para el análisis de los resultados de la "evaluación de la arquitectura".

5 EVALUACIÓN DE LA ARQUITECTURA PROPUESTA

Una vez ha sido elegida la arquitectura, se puede iniciar ahora a formular los caminos para testearla.

5.1 Metodología

Para de la evaluación de la arquitectura propuesta se intentará inferir experimentalmente si es posible que una configuración de la arquitectura en jerarquía, la cual fue planteada en la metodología general, puede proveer mejores resultados en determinadas condiciones que el sistema de referencia utilizado.

Pasa el desarrollo del problema planteado, se ha dejado un poco la parte empírica para ir a los hechos y buscar los conceptos de las simulaciones.

El gran concepto que se quiere demostrar aquí es que, utilizando datos de navegaciones reales se pueda inferir que un montaje en jerarquía de caches que realice discriminación de tamaño de los objetos pueda hacer que la cache obtenga mayores aciertos que una que le da igual reemplazar un objeto muy grande por muchos más de menor tamaño.

Las técnicas e instrumentos utilizados y unidades para el desarrollo de la evaluación de la arquitectura, han sido planteados previamente durante la metodología general.

La investigación y el planteamiento en lo que resta de documento, intenta explorar capturas de datos relacionadas (experimentos), que permitan facilitar el discernimiento de nuevos conceptos a partir de los casos de estudio analizados.

5.2 Procedimiento

Para el desarrollo de la evaluación de la arquitectura se van a definir una serie de casos de estudio (Experimentos) a ser ejecutados contra la arquitectura planteada en la metodología general, todos y cada uno de los casos intentan compara una configuración de jerarquía de proxies con una configuración de referencia que solo utilice un proxy.

La nomenclatura y nombres técnicos utilizados han sido previamente definidos cuando se precisó la arquitectura que se iba a utilizar.

En general en la evaluación de la arquitectura se van a definir tres juegos de pruebas contra la arquitectura, un juego contra la configuración de la arquitectura planteada como jerarquía y el otro juego de pruebas contra la arquitectura de referencia, cada uno compuesto de una serie de simulaciones las cuales se analizarán y se consignarán por separado.

Los dos parámetros básicos a analizar en los experimentos son:

- El tamaño de objeto WEB máximo y mínimo que aceptan las caches de la jerarquía, respecto al mismo parámetro con la arquitectura de referencia.

- La distribución de los tamaños de las caches web utilizadas en la jerarquía, comparada con en tamaño de la cache utilizada como referencia.

Una vez que realizados los experimentos lo siguiente es plasmar esos resultados en tablas que permitan inspeccionar los datos obtenidos por simple inspección. Para el análisis detallado de cada uno de los experimentos, se van a utilizar gráficas que enseñen las dos arquitecturas contrastadas buscando cuál en cada momento puede ser la que mejor haya obtenido los resultados.

Por último se responderá lo más concretamente posible el postulado planteado en la evaluación de la arquitectura, infiriendo en qué casos cada una de las configuraciones de la arquitectura dio mejores o peores resultados.

5.3 Diseño de la investigación

Dentro del desarrollo de la metodología general, se han propuesto 2 configuraciones de la arquitectura que va a ser desarrollada y estudiada en los siguientes experimentos propuestos, primero se hará un análisis de las características de la traza que será utilizada en las simulaciones para la toma de cada una de las mediciones, para entender la manera como están dispuestos los tamaños de los objetos a lo largo de la navegación extraída del proxy de la UPV.

5.3.1 Características de la traza del proxy de la UPV a utilizar en los experimentos

Dado que el estudio a realizar se basa en analizar cómo influye el tamaño de los objetos sobre la arquitectura, se hace necesario conocer en detalle cómo están distribuidos los tamaños de los objetos en el fichero que se usará para lanzar las simulaciones.

La siguiente tabla representa la distribución del número de peticiones y la cantidad de bytes en cada uno de los tamaños de objetos presentes en el fichero se simulación.

Tamaño de Objeto	# Peticiones	% Total	MB	%
0-0.1KB	601	0,14287	0,114	0,003
0.1-1.0KB	288131	68,4966	93,736	2,529
1-5KB	73744	17,531	181,940	4,909
5-10KB	23556	5,5999	165,823	4,474
10-50KB	28630	6,80613	565,647	15,26
50-100KB	3384	0,80447	231,194	6,238
100-500KB	1774	0,42173	314,095	8,475
0.5-1.0MB	323	0,07679	242,947	6,555
1-5MB	446	0,10603	844,088	22,77
5-10MB	46	0,01094	318,710	8,599
10-50MB	11	0,00262	173,044	4,669
50-100MB	3	0,00071	210,592	5,682
>100MB	1	0,00024	364,356	9,831
TOTAL	**420650**	**100**	**3706,286**	**100**

Tabla 5 Número de peticiones y cantidad de Bytes por tamaño de objeto presente en el fichero de simulación

De la “Tabla 5” se puede inferir que el total de peticiones procesado es de 420650 peticiones y que aproximadamente el tráfico total generado en la simulación será aproximadamente de 3,7 GB.

Las siguientes figuras enseñarán como están distribuidas las peticiones de objetos por el fichero de simulación (Tamaño de Objeto).

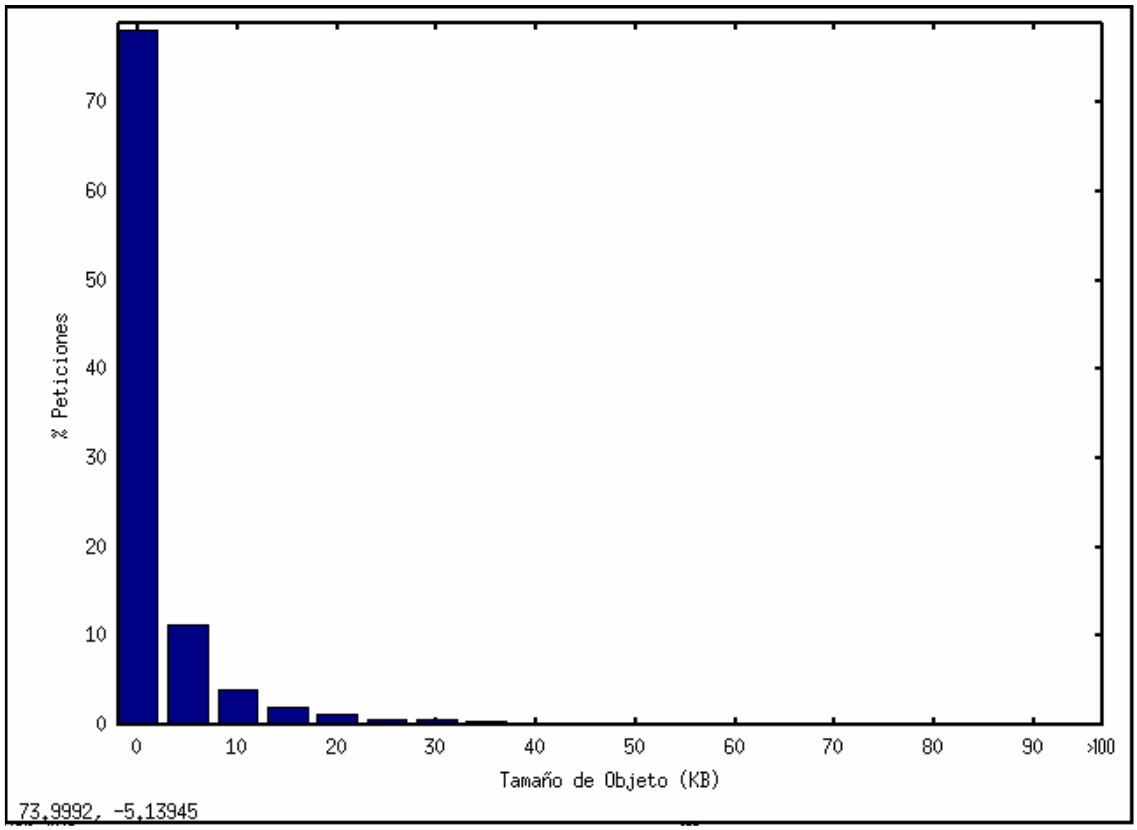

Ilustración 16 Densidad de peticiones respecto a los tamaños de objetos en el fichero de simulación

En la siguiente ilustración, se verá la densidad acumulada de las peticiones respecto al tamaño de los objetos, para ver desde otro punto de vista como se distribuyen por el fichero de simulación.

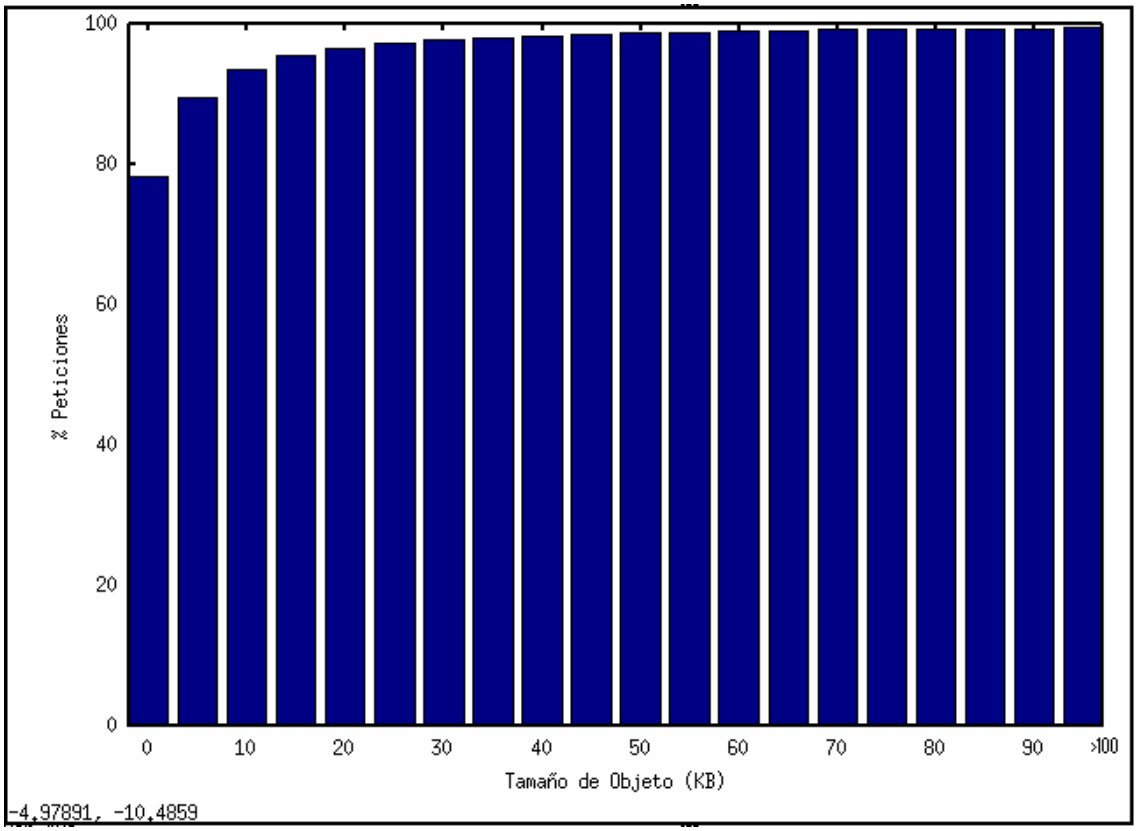

Ilustración 17 Densidad acumulada de peticiones respecto a los tamaños de objetos en el fichero de simulación

De la ilustración se puede inferir que la mayor cantidad de objetos se encuentra entre los 0,1 KB y los 10 KB, lo anterior también indica que la mayoría de los objetos son pequeños.

Las siguientes figuras enseñarán como están distribuidos los bytes dependiendo del tamaño de los objetos a través del fichero de simulación.

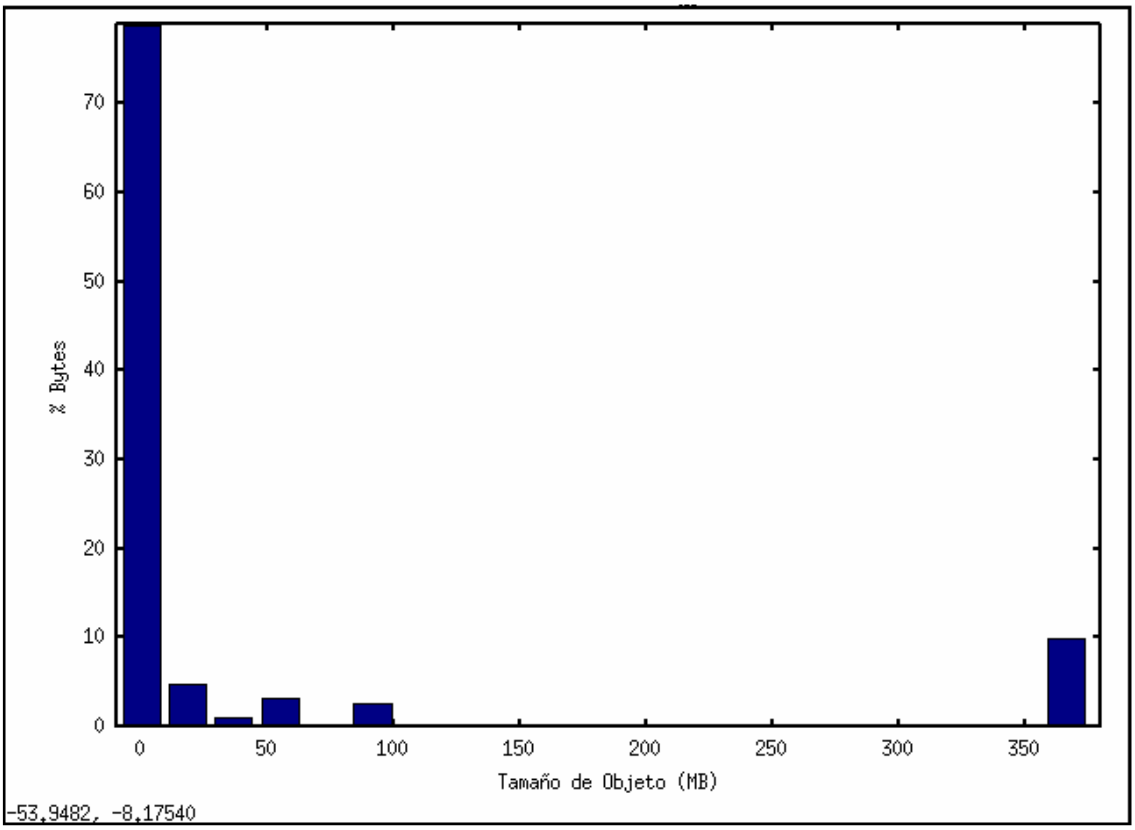

Ilustración 18 Densidad de Bytes respecto a los tamaños de objetos en el fichero de simulación

El siguiente ilustración corresponde a la densidad acumulada de la cantidad de bytes por el tamaño de los objetos WEB.

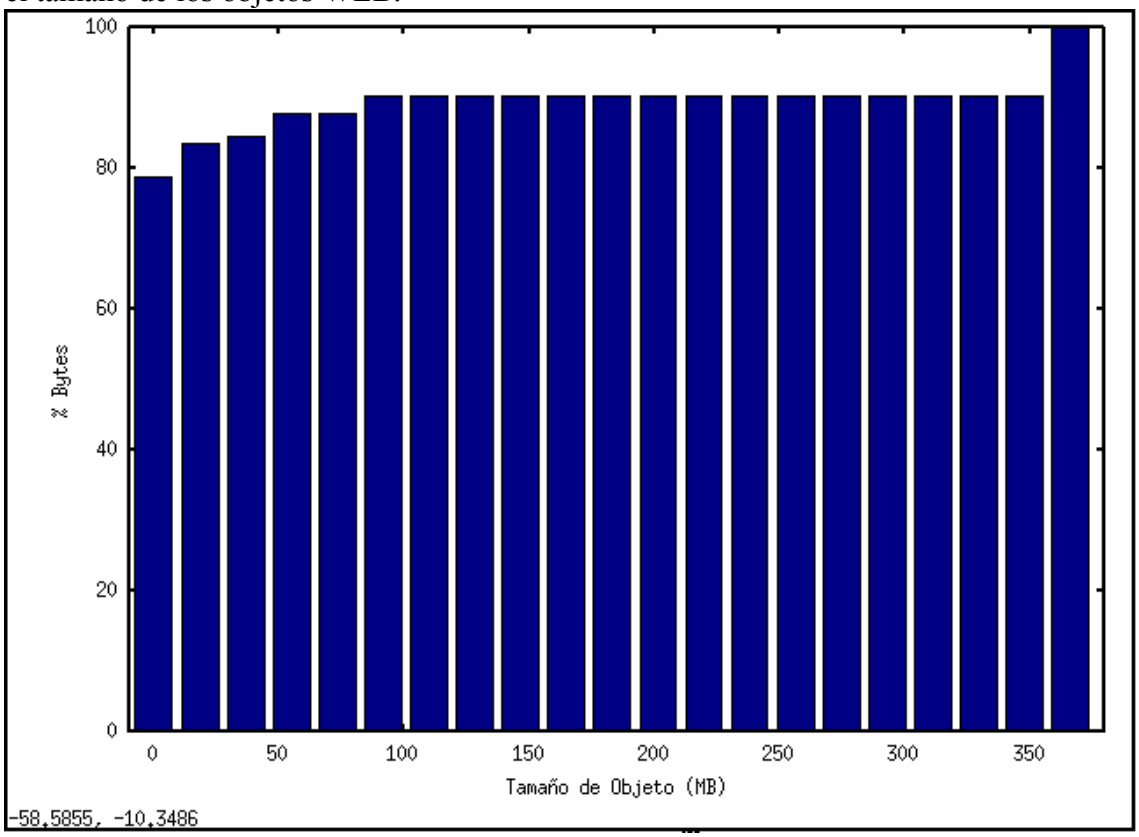

Ilustración 19 Densidad acumulada de Bytes respecto a los tamaños de objetos en el fichero de simulación

De las dos ilustraciones anteriores se puede inferir que las mayores cantidades de tráfico se encuentran para objetos que tienen un tamaño inferior a los 10 MB.

5.3.1.1 Estadísticas del fichero de simulación a utilizar

La siguiente tabla recoge algunos datos estadísticos del fichero de simulación que se utilizará para los experimentos.

Parámetro Estadístico	Valor
Media	293 Bytes
Promedio	9238,6 Bytes
Desviación Estándar	638810 Bytes
Máximo Tamaño de Objeto	364,356 MB
Varianza	4,08E+11
Moda	256 Bytes
Asimetría(skewness)	522,28
Curtosis(kurtosis)	305850

Tabla 6 Estadísticas del tamaño de los objetos a utilizar en la simulación

De la tabla anterior se puede inferir lo siguiente:

- Como puede ser visto en la media, la mitad de los tamaños de los objetos que hay en el fichero de simulación son menores de 293 Bytes, la otra mitad mayores. Confirmando lo visto anteriormente, que la traza está mayormente constituida por objetos pequeños.
- El valor máximo de objeto es bastante grande, con lo cual no es raro que el promedio difiera un poco de la media.
- El valor promedio de objeto presente en el fichero de simulación es aproximadamente 9 KB.
- El tamaño de objeto que más se repite es 256 bytes.
- El skewness de los datos es positivo, con lo cual la distribución de la mayoría los datos es hacia los tamaños pequeños, y cerca de los 522 Bytes.
- El valor de curtosis, hace discernir que los tamaños de objetos tienen una distribución leptocúrtica por su elevado grado de concentración de valores hacia un valor central.

Todos los anteriores valores confirman lo visto en la “Ilustración 16”. A continuación partiendo del conocimiento adquirido sobre el fichero de simulación que se va a usar, se configuran todos los experimentos a ejecutar contra la arquitectura.

5.3.2 Resumen de los Experimentos a realizar

La siguiente tabla muestra todas las medidas que se van a realizar sobre la arquitectura con los parámetros fundamentales que se varían de un experimento a otro.

	Contexto en que se realiza la medición	Servidor Proxy	Tamaño de la Web Cache en MB	Rango de tamaño de objetos permitidos por la cache web en KB
Experimento 1	JERARQUÍA DE PROXIES	Servidor Proxy Hijo	100	[0-X]
		Servidor Proxy Padre	100	[X-4096]
	MEDIDA DE CONTROL	Servidor Proxy Referencia	200	[0-4096]
Experimento 2	JERARQUÍA DE PROXIES	Servidor Proxy Hijo	100	[0-X]
		Servidor Proxy Padre	100	[X-50]
	MEDIDA DE CONTROL	Servidor Proxy Referencia	200	[0-50]
Experimento 3	JERARQUÍA DE PROXIES	Servidor Proxy Hijo	[50,100,150]	0-20
		Servidor Proxy Padre	[150,100,50]	20-4096
	JERARQUÍA DE PROXIES	Servidor Proxy Hijo	[50,100,150]	0-25
		Servidor Proxy Padre	[150,100,50]	25-4096
	JERARQUÍA DE PROXIES	Servidor Proxy Hijo	[50,100,150]	0-30
		Servidor Proxy Padre	[150,100,50]	30-4096
	MEDIDA DE CONTROL	Servidor Proxy Referencia	200	[0-4096]

Tabla 7 Resumen de los experimentos a realizar para el análisis de las jerarquías de caches web

En la Tabla 7 se ven los 3 experimentos que se proponen para evaluar, en los dos primeros experimentos la **X** indica el parámetro a iterar entre cada una de las mediciones realizadas en la toma de datos del experimento (desde ahora este parámetro se llamará "*Tamaño Objeto Máx/Min (Padre/hijo) cacheable (KB)*"). Para el experimento 3 iteraremos el tamaño de la caches de la configuración padre/hijo y el rango de tamaño de objetos permitidos por la cache WEB.

Nota: En todos los experimentos propuestos se cumplen los siguientes preceptos:

$$Tamaño\ Cache\ Padre + Tamaño\ Cache\ Hijo = Tamaño\ Cache\ Referencia$$

$$Tamaño\ Objetos\ Cacheables\ Proxy\ Padre + Tamaño\ Objetos\ Cacheables\ Proxy\ Hijo = Tamaño\ objetos\ Cacheables\ en\ Proxy\ Referencia$$

Lo que permitirá que se puedan comparar las jerarquía con el sistema de referencia.

5.3.3 Objetivo general a conseguir con los experimentos propuestos

Demostrar que alguno de los experimentos realizados con la jerarquía puede obtener mejores resultados que la arquitectura de referencia.

5.3.4 Experimento 1

Tamaño de las Web caches iguales para la jerarquía padre 100 Mega Bytes e hijo 100 Mega Bytes tamaño de los objetos [0-**X**-4096 Kilo Bytes] vs un solo proxy con tamaño de web cache de 200 MB y aceptando el mismo tamaño de objetos.

5.3.4.1 Resultados obtenidos

	Tamaño de Objetos Cacheables	TCP_MEM_ HIT en K	TCP_HIT en K	Total Hits en K	Tasa de aciertos	TCP_MEM_ HIT MB	TCP_HIT MB	Saved Traffic MB	Tasa de aciertos por byte
PADRE/HIJO	0-5-4096	125,636	93,754	219,39	**52,15499822**	112,051	145,749	257,801	**6,95631409**
	0-10-4096	122,882	93,842	216,724	**51,52121716**	113,398	148,763	262,162	**7,07398813**
	0-15-4096	120,127	94,453	214,58	**51,01152978**	113,597	148,983	262,58	**7,08526713**
	0-20-4096	114,392	99,035	213,427	**50,73743017**	109,559	154,157	263,715	**7,11589315**
	0-25-4096	113,774	98,549	212,323	**50,4749792**	109,71	153,175	262,885	**7,09349703**
	0-30-4096	112,733	98,502	211,235	**50,21633187**	112,764	149,552	262,316	**7,07814355**
	0-35-4096	112,349	97,87	210,219	**49,9748009**	112,211	148,366	260,577	**7,03121964**
	0-40-4096	110,71	98,987	209,697	**49,85070724**	111,682	147,838	259,52	**7,00269833**
	0-45-4096	110,954	98,294	209,248	**49,74396767**	111,16	147,19	258,351	**6,97115488**
MEDIDA CONTROL	**0-4096**	49,79	158,718	208,508	**49,56804945**	65,29	191,911	257,201	**6,94012412**

Tabla 8 Resultados experimento 1

Se realizaron 10 simulaciones variando el tamaño de los objetos cacheables por la arquitectura configurada en jerarquía. Los resultados obtenidos se analizarán después de la realización del experimento 2, debido a que serán contrastados.

5.3.5 Experimento 2

Tamaño de las Web caches iguales padre 100 Mega Bytes e hijo 100 Mega Bytes tamaño de los objetos [0-**X**-50 Kilo Bytes] vs un solo proxy con tamaño de web cache de 200 MB y aceptando el mismo tamaño de objetos.

5.3.5.1 Resultados obtenidos

	Tamaño de Objetos Cacheables	TCP_MEM_ HIT en K	TCP_HIT en K	Total Hits en K	Tasa de aciertos	TCP_MEM_ HIT MB	TCP_HIT MB	Saved Traffic MB	Tasa de aciertos por byte
PADRE/HIJO	0-5-50	126,784	94,71	221,494	**52,65517651**	89,261	132,505	221,766	**5,98397194**
	0-10-50	122,824	95,121	217,945	**51,81148223**	89,829	132,475	222,304	**5,99848894**
	0-20-50	114,82	99,308	214,128	**50,90407702**	83,826	134,089	217,915	**5,88005936**
	0-30-50	114,416	97,181	211,597	**50,30238916**	80,108	129,981	210,09	**5,66891527**
	0-40-50	111,741	99,335	209,69	**49,84904315**	80,86	120,499	201,358	**5,43329736**
MEDIDA CONTROL	**0-50**	91,027	129,814	220,841	**52,49994057**	67,189	157,252	224,441	**6,05615219**

Tabla 9 Resultados experimento 2

Para este experimento se realizaron 6 simulaciones utilizando límites de tamaño de objeto cacheable padre/hijo de 5, 10, 20, 30, 40 KB. Los resultados son diferentes respecto al experimento 1 como se verá a continuación.

5.3.5.2 Análisis de los resultados obtenidos en el experimento 1 y 2

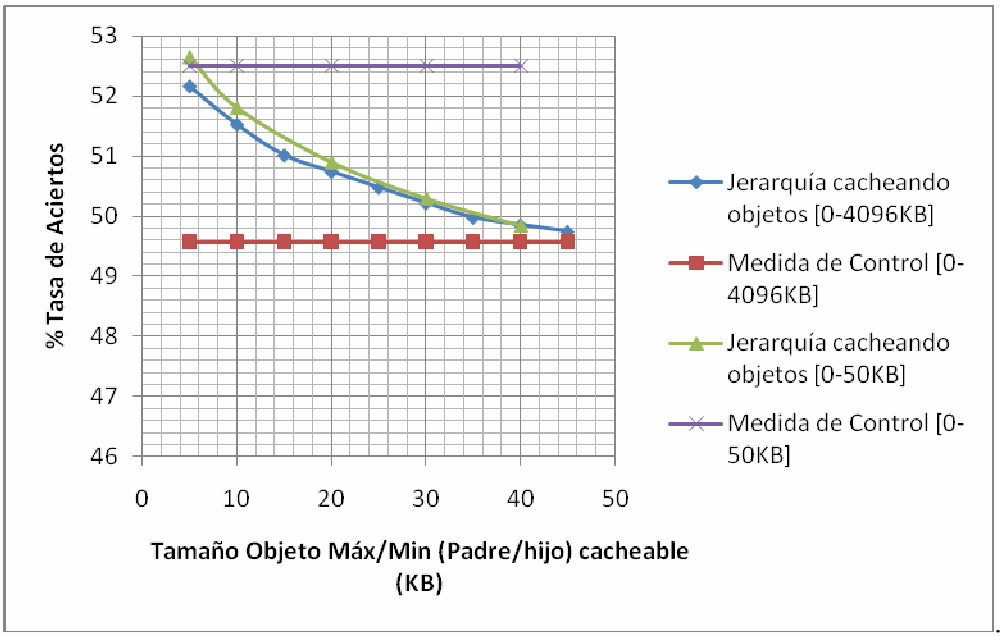

Ilustración 20 Experimento 1 y Experimento 2, "Tasa de aciertos" vs "Tamaño Límite de Objeto Máx/Min (Padre/Hijo) cacheable.

En la "Ilustración 20" se ve como la jerarquía con tamaños de objetos pequeños máx/Min cacheables obtiene mejores resultados que la medida de referencia, dicho de otra forma, la jerarquía consigue mayores éxitos que la arquitectura de referencia cuando usa límites de tamaño menores a 45 KB.

En azul se ve que para la configuración del padre aceptando objetos entre 0 - 5 KB y el hijo entre 5 y 4096 KB, se obtiene el mejor resultado del experimento. Lo anterior comprueba que la jerarquía evita que un objeto grande reemplace a más pequeños (el proxy padre se encarga de proteger a los objetos pequeños en su caché) consiguiendo así que esta pueda obtener mayores éxitos.

En verde para la jerarquía que sólo cachea objetos de tamaño entre [0-50KB] se observa que uno de los resultados con la jerarquía a conseguido mejorar la arquitectura de referencia. La tendencia del resultado obtenido con la jerarquía hace pensar que se pueden aumentar la tasa de aciertos un poco más.

Comparando las mediciones realizadas con las dos jerarquías se concluye que cuanto más grandes puedan ser los objetos permitidos a ser cacheados en la arquitectura de referencia, se podrán conseguir mejores configuraciones y mayor tasa de aciertos trabajando con jerarquías de proxies que discriminen los objetos por su tamaño.

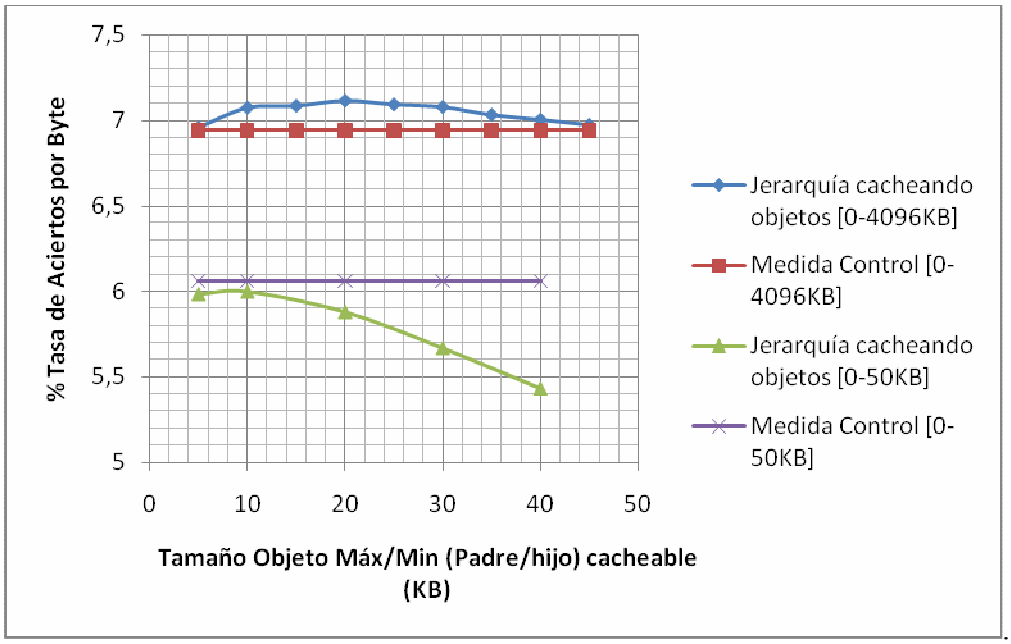

Ilustración 21 Experimento 1 y Experimento2, "Tasa de aciertos por Byte" vs "Tamaño Límite de Objeto Máx/Min (Padre/Hijo) cacheable

En azul se observa que en un límite de 20 KB la jerarquía que cachea objetos entre [0-4096KB] permite salvar mayor cantidad de tráfico con la arquitectura de referencia. De otro lado los límites pequeños también han influido en la obtención de mejores resultados con la jerarquía esto es para tamaños de objeto menores a 45 KB.

De otro lado se observa en verde que esa jerarquía no logra mejorar la cantidad de trafico salvado respecto de la arquitectura de referencia, aunque en la "Ilustración 20" se vio que una de las medidas obtuvo mayores éxitos, esto se debe a la distribución del tráfico de la traza utilizada que, para muchos objetos pequeños el proxy padre salva tráfico pero no tanto como el que se pierde al cachear solo objetos grandes en el proxy hijo.

5.3.6 Experimento 3

Tamaño de las Web caches diferentes (pero sumado padre/hijo igual a 200 MB) e iteración entre "Tamaño Límite de Objeto Máx/Min (Padre/Hijo) cacheable" para 20, 25 y 30 KB vs un solo proxy con tamaño de web cache de 200 MB

5.3.6.1 Resultados obtenidos

	Tamaños Caches Padre/hijo (MB)	TCP_MEM_ HIT en K	TCP_HIT en K	Total Hits en K	Tasa de aciertos	TCP_MEM_ HIT MB	TCP_HIT MB	Saved Traffic MB	Tasa de aciertos por byte
PADRE/HIJO	50-150	118,297	80,85	199,147	**47,34**	113,794	130,788	244,581	**6,6**
	75-125	116,201	91,307	207,508	**49,33**	111,188	144,634	255,822	**6,903**
Tamaño Máx/Mín	100-100	114,392	99,035	213,427	**50,74**	109,559	154,157	263,715	**7,116**
	112-88	114,379	100,595	214,974	**51,11**	109,294	155,606	264,9	**7,148**
	125-75	114,01	102,844	216,854	**51,55**	109,391	154,962	264,352	**7,133**
cacheable [20 KB]	150-50	112,509	106,842	219,351	**52,15**	101,312	160,789	262,101	**7,072**
PADRE/HIJO	50-150	117,322	80,957	198,279	**47,14**	113,418	128,793	242,211	**6,535**
	75-125	115,22	91,334	206,554	**49,1**	111,091	143,156	254,246	**6,86**
Tamaño Máx/Mín	100-100	113,774	98,549	212,323	**50,47**	109,71	153,175	262,885	**7,093**
	125-75	113,252	102,708	215,96	**51,34**	109,075	156,818	265,893	**7,174**
cacheable [25 KB]	150-50	112,209	106,526	218,735	**52**	108,659	155,097	263,756	**7,116**
PADRE/HIJO	50-150	116,511	80,828	197,339	**46,91**	115,863	124,576	240,439	**6,487**
	75-125	114,303	91,28	205,583	**48,87**	113,748	138,165	251,914	**6,797**
Tamaño Máx/Mín	100-100	112,733	98,502	211,235	**50,22**	112,764	149,552	262,316	**7,078**
	125-75	112,448	102,871	215,319	**51,19**	112,194	154,126	266,32	**7,186**
cacheable [30 KB]	150-50	111,492	106,55	218,042	**51,83**	111,566	153,675	265,24	**7,156**
MEDIDA CONTROL	200	49,79	158,718	208,508	**49,57**	65,29	191,911	257,201	**6,94**

Tabla 10 Resultados Experimento 3

Para este experimento se realizaron los juegos de mediciones y se ha hecho un planteamiento diferente para analizar el funcionamiento de las caches WEB pero sin desviarnos de nuestro objetivo de evidenciar que la jerarquía pueda dar mejores resultados que la arquitectura de referencia. A continuación se analizan los resultados vistos en la tabla anterior.

5.3.6.2 Análisis de los resultados obtenidos en el experimento

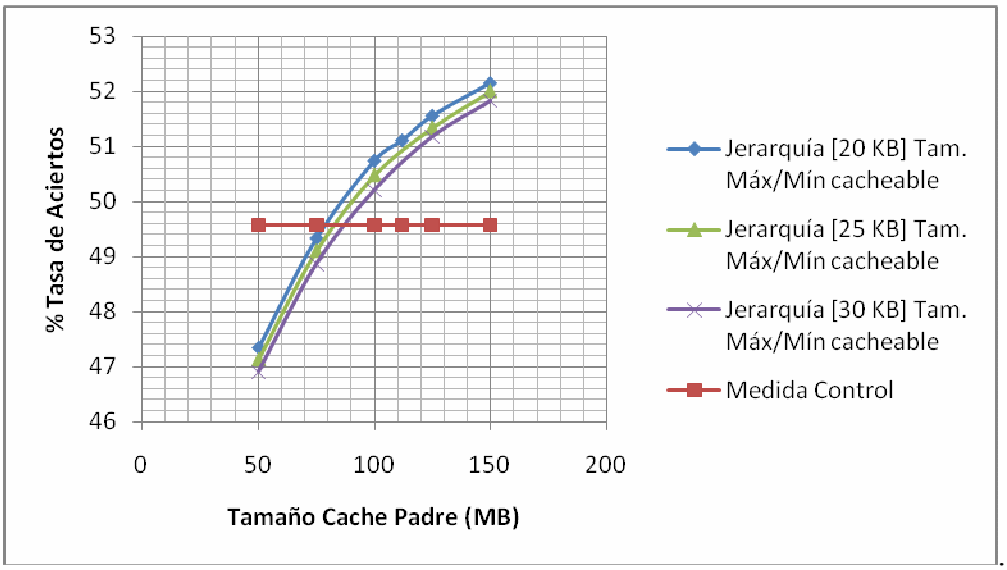

Ilustración 22 Experimento 3, "Tasa de aciertos" Vs Tamaño cache del Padre

El peor resultado obtenido de número de éxitos de la jerarquía ha sido cuando el tamaño de la cache del padre era de sólo 50 MB, lo que significa que se están perdiendo éxitos por remplazarse viejos objetos por nuevos objetos cuando la cache está llena.

El mejor resultado por la jerarquía es conseguido cuando el tamaño de la cache del padre es de 150 MB puesto que puede almacenar mayor cantidad de objetos y los objetos permanecen más tiempo en su cache.

La jerarquía se comporta mejor cuando la cache que almacena objetos pequeños es mayor que la que almacena objetos grandes.

En azul se puede ver que la jerarquía obtiene mayor tasa de aciertos cuando almacena objetos más pequeños [20 KB].

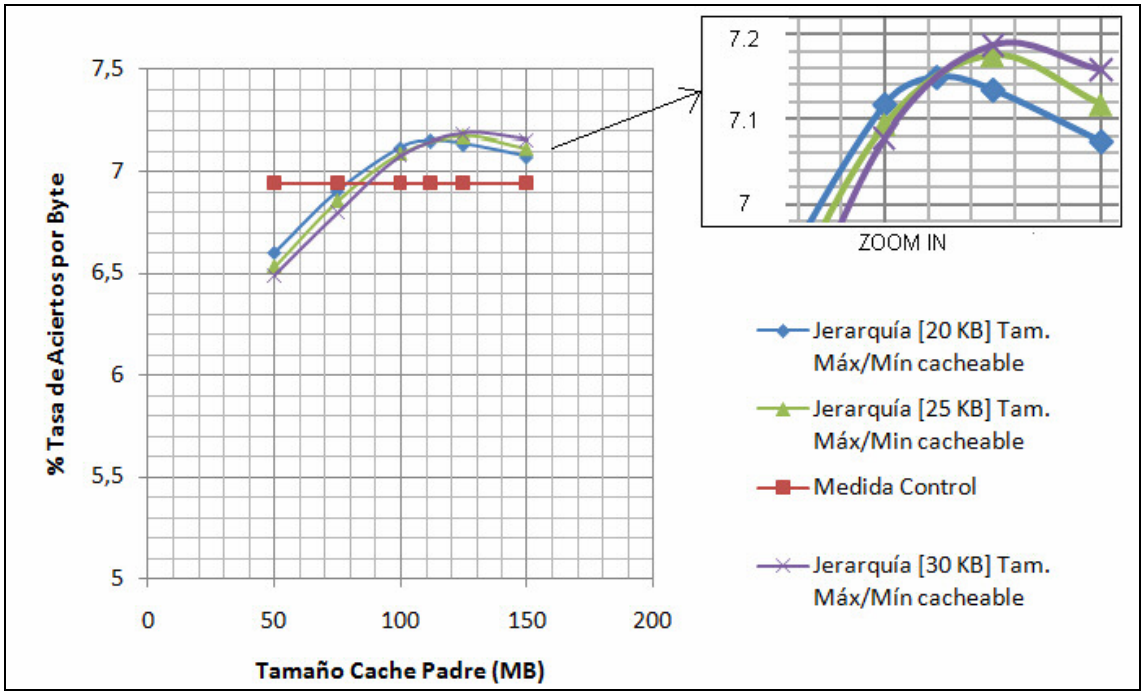

Ilustración 23 Experimento 3, "Tasa de aciertos por byte" vs Tamaño cache del Padre

En la "Ilustración 22" se observe que el peor resultado se daba usando una cache del padre de 50 MB, para el tráfico salvado sucede lo mismo que con la tasa de aciertos en cuanto que, sí es posible obtener mejores resultados con la jerarquía, pero se observa una baja en la tasa de aciertos por byte cerca de 125 MB de tamaño de cache padre.

5.3.7 Superposición de experimentos 1 y 3

A continuación se presentan gráficas en tres dimensiones que cruzan los resultados obtenidos para los experimentos 1 y 3 de manera que pueda inferirse cual puede ser la configuración ideal que se debe poner en la jerarquía, para obtener más resultados frente a la arquitectura de referencia funcionando con un solo proxy.

5.3.7.1 Tasa de aciertos

A continuación se muestran los cruces de los resultados de la tasa de aciertos para los experimentos 1 y 3, presentados desde las diferentes perspectivas, para poder apreciar de la mejor manera posible.

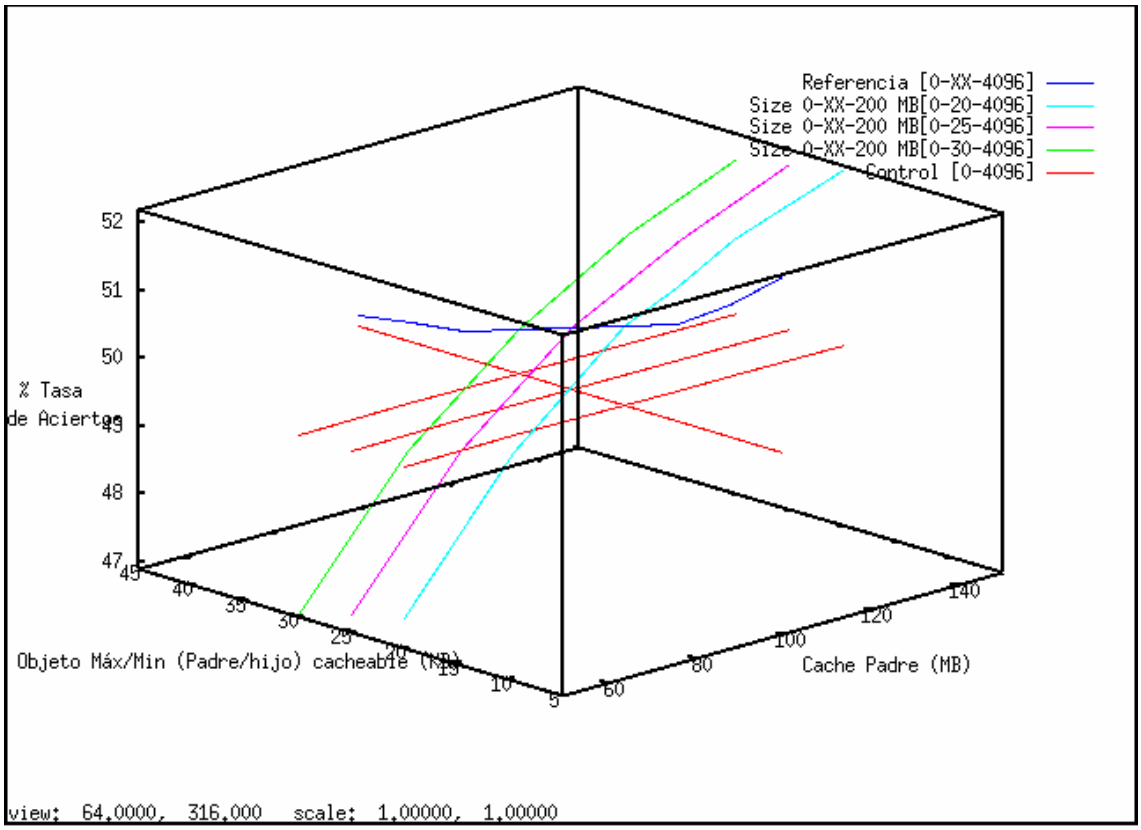

Ilustración 24 Cruce de tasa de aciertos experimentos 1 y 3 perspectiva (A)

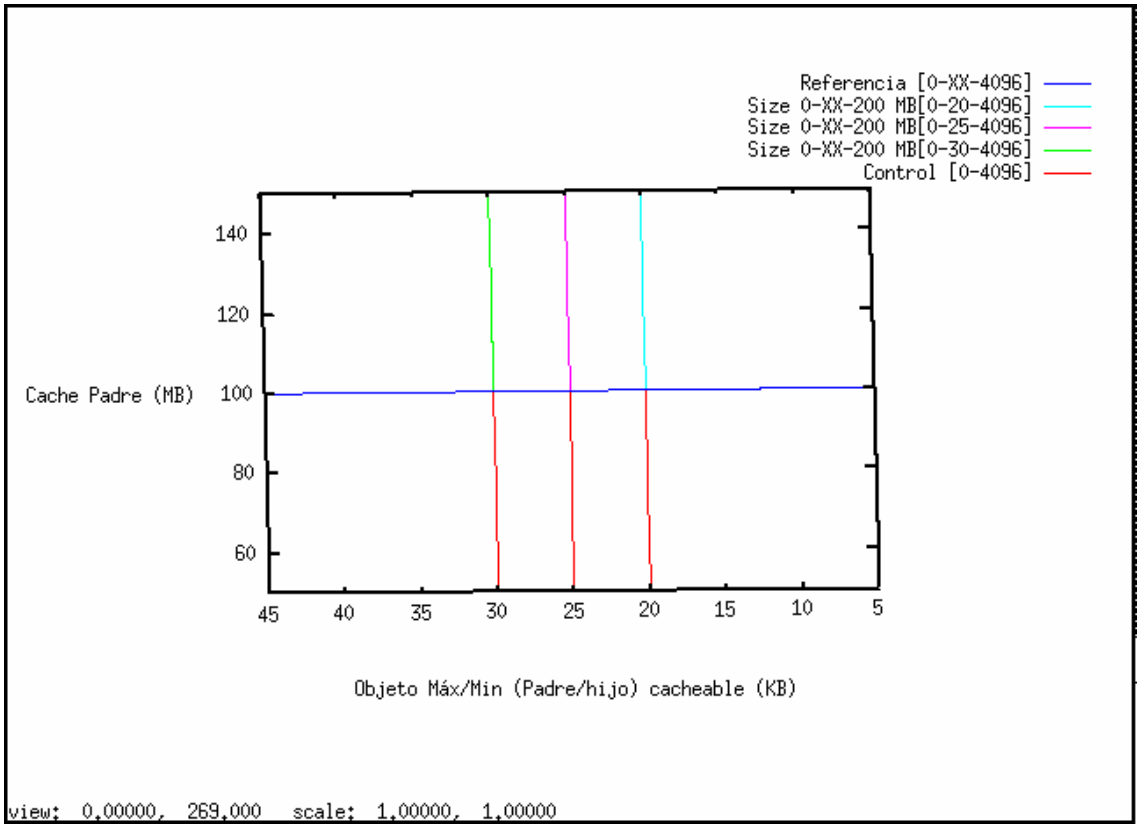

Ilustración 25 Cruce de tasa de aciertos experimento 1 y 3 perspectiva (B)

En las perspectiva (A) se puede ver que el mejor resultado se obtiene con un "Tamaño Límite de objeto Máx/Mín (Padre/Hijo) cacheable" de 20 KB y para la cache del padre

con un tamaño mayor, lo que garantiza un mayor protección al reemplazo de los objetos pequeños.

En la perspectiva (B), de manera general se observan las medidas que se han realizado en los experimentos 1 (línea azul) y 3 (resto de líneas), se ha puesto esta perspectiva para no perder de vista cuales fueron los valores de partida.

5.3.7.2 Tasa de aciertos por byte

A continuación se muestran los cruces de los resultados de la tasa de aciertos por byte para los experimentos 1 y 3, presentados desde las diferentes perspectivas.

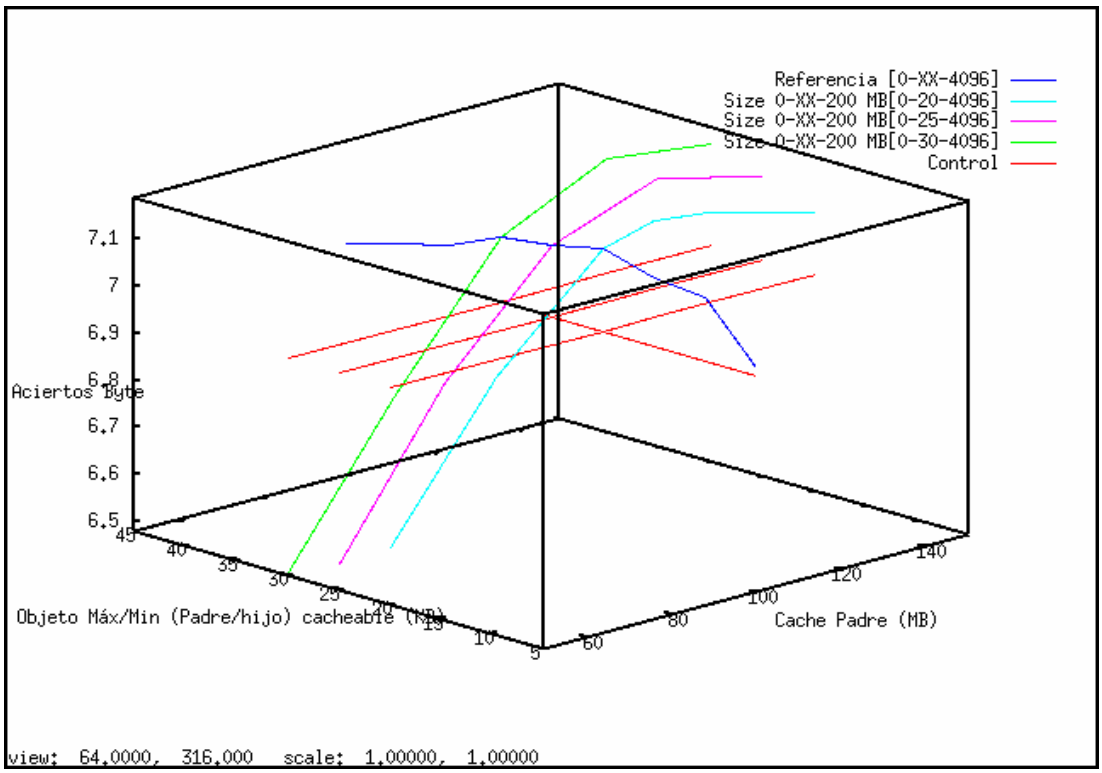

Ilustración 26 Cruce de tasa de aciertos por Byte experimento 1 y 3 perspectiva (A)

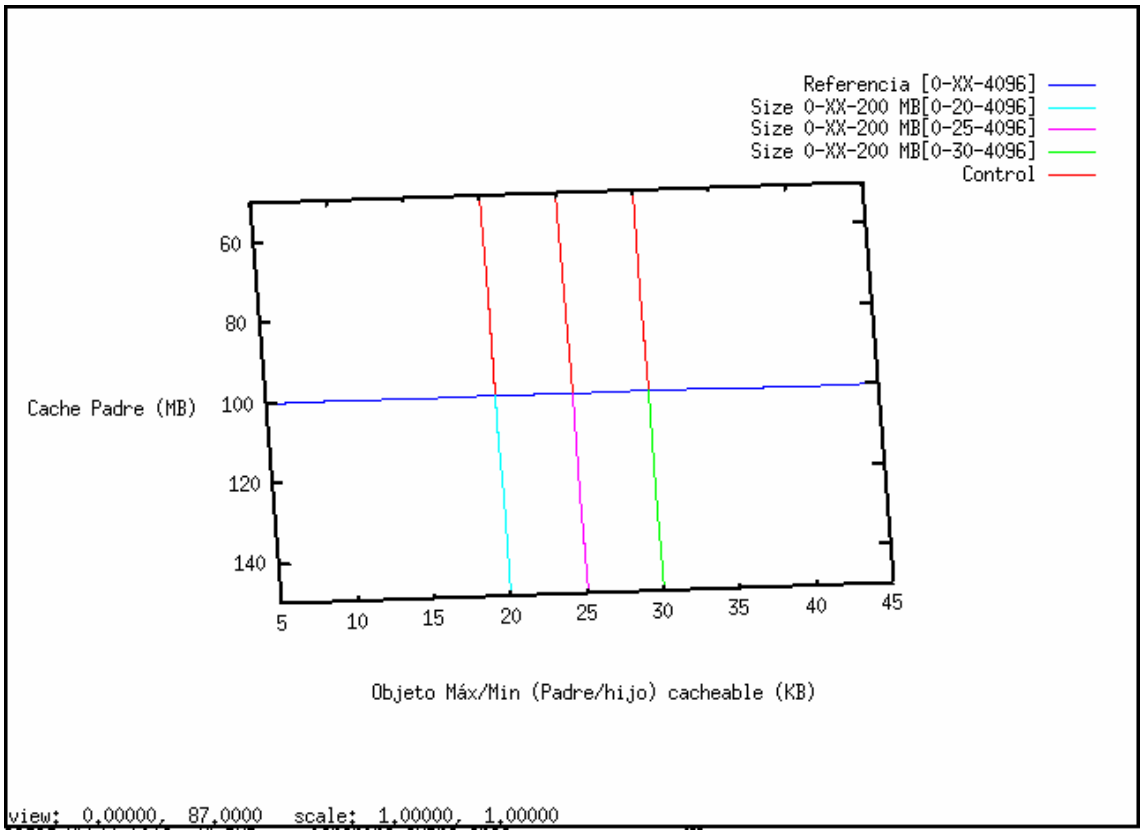

Ilustración 27 Cruce de tasa de aciertos por Byte experimento 1 y 3 perspectiva (B)

En las perspectiva (A), el mejor resultado se da para un tamaño de la cache padre mayor que la cache hijo, pero el máximo de tráfico salvado se ve cuando se utiliza un "Tamaño Límite de objeto Máx/Mín (Padre/Hijo) cacheable" de 30 KB. La tasa de aciertos por byte aumenta a medida que aumenta el tamaño de la cache del padre hasta alcanzar un máximo cerca de los 125 MB (tamaño cache padre) en donde inicia su decrecimiento, para todos los "Tamaños Límite de objeto Máx/Mín (Padre/Hijo) cacheable" medidos en los experimentos el comportamiento fue el mismo.

En la perspectiva (B), una vez más se observan las medidas que se han realizado en los experimentos 1 (línea azul) y 3 (resto de líneas), se ha puesto esta perspectiva para no perder de vista cuales fueron los valores de partida y confirmar que esta nueva gráfica 3D es consistente con los experimentos realizados.

6 FUTURAS LINEAS DE INVESTIGACIÓN

- Implementar en un futuro un particionamiento dinámico en función del tamaño de objeto, para evaluar si es posible conseguir aún mejores resultados.

- Se puede proponer estudiar cómo influye la elección del algoritmo de reemplazo de los proxies en los resultados obtenidos en la evaluación de la arquitectura propuesta, para poder así caracterizar nuevos comportamientos.

- Podría estudiarse a profundidad como influye el tráfico ICP entre los servidores proxy de la jerarquía en la configuración de la arquitectura, para analizar si es loable a implementarse realmente una configuración de una cache particionada.

- Nuevos tipos de particiones podrían ser probados con la arquitectura y la combinación de ellos podría ser probada en la búsqueda de nuevos y mejores métodos de WEB caching.

- Se hace posible desarrollar nuevos planteamientos de la arquitectura, todos los módulos que la componen pueden ser ampliados, debido a que funcionalmente y técnicamente son independientes.

- Es posible plantear muchas mejoras en el servidor de aplicaciones utilizado en este proyecto, puesto que virtualmente puede servir cualquier tipo de objeto web, así que un gran progreso en el futuro podría ser el extender el abanico de objetos a la mayoría presentes en el WEB 2.0. Haciendo de esta forma que las simulaciones sean lo más parecidas a las que se realizan en el mundo real.

- Aparte de las mediciones convencionales que pueden ser realizadas sobre los proxies, al utilizar proxies reales se podrían hacer estudios de rendimiento de CPU y de niveles de uso de disco de la caché.

CONCLUSIONES

- Se ha planteado una configuración que utiliza una jerarquía de proxies configurados como padre e hijo, que particiona la cache en función del tamaño de los objetos, la arquitectura ha sido probada y evaluada, exponiendo que sirve para realizar simulaciones.

- Se ha demostrado que es posible que una jerarquía de proxies pueda aumentar las prestaciones del sistema de caching en función del tamaño de los objetos almacenados, que un sólo servidor proxy, aprovechándose de la distribución que ésta puede hacer de los objetos WEB a cachear entre los miembros de la jerarquía, discriminándolos por tamaño y haciendo que objetos pequeños se cacheen con objetos pequeños y objetos grandes con objetos grandes. Todo lo anterior es posible bajo ciertas condiciones de configuración, como lo son: los tamaños de los caches utilizados en la jerarquía y el tamaño de objeto que puede almacenar cada cache. La configuración ideal vista en los experimentos realizados ubica el tamaño de la cache del padre en 150 MB y un tamaño de objeto máx/mín entre el padre y el hijo de 20 KB; el otro dato interesante es que esa partición no influye en un detrimento de la tasa de aciertos por byte, se consigue mejorar también en este aspecto, siendo el mejor tamaño de objeto máx/mín entre el padre y el hijo de 30 KB y el tamaño de la cache del padre de 150 MB.

- Una de las grandes ventajas de la arquitectura planteada radica en el hecho de utilizar un solo ordenador para implementarla. Para otros investigadores que tomen como punto de partida esta arquitectura, puede ser una gran ventaja ya que utiliza recursos muy comunes y económicos.

- La arquitectura planteada es altamente configurable puesto que cada uno de los módulos que la componen se puede adaptar de acuerdo al postulado que se desee plantear.

- La presente arquitectura permite realizar análisis de rendimiento de CPU y de niveles de utilización de discos de las caches, debido a que los proxies utilizados son instancias independientes al sistema de simulación, estos parámetros actualmente no son evaluables con otras herramientas existentes.

- La arquitectura es multiplataforma[8] y utiliza lenguajes de programación y software que son estándar y libres, por lo que para futuras ampliaciones esto puede ser una gran fortaleza.

- Se ha comprobado que se puede configurar la arquitectura dependiendo del tipo de estudio que se desee realizar con las caches web, en el documento se han planteado un par de configuraciones, pero las opciones las limitan los que se desee evaluar.

[8] SO basados en unix y Windows en combinación con Cygwin

- El uso de la CPU en todo momento durante las simulaciones no saturó el ordenador donde fueron realizadas.

BIBLIOGRAFÍA

[1] Rabinovich, M. (2002). *Web caching and replication*. Addison-Wesley.

[2] Wessels, D. *Web caching*. O'Reilly, 2001.

[3] C.-Y. Chiang, M. Liu and M. Muller, "Pyramid Set, Web Proxy Request Distribution, and their Applications", Technical Report.

[4] C.-Y. Chiang, "On Building Dynamic Web Caching Hierarchies", Ph.D. Dissertation, The Ohio State University, 2000.

[5] C.Williamson, "On Filter Effects inWeb Caching Hierarchies", ACMTransactions on Internet Technology, Vol. 2, No. 1, pp. 47-77, February 2002.

[6] M. Busari and C. Williamson, "ProWGen: A Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches", Computer Networks, Vol. 38, No. 6, pp. 779-794, June 2002.

[7] R. Doyle, J. Chase, S. Gadde, and A. Vahdat, "The Trickle-Down Effect: Web Caching and Server Request Distribution", Proceedings of the Web Caching. Workshop, Boston, MA, June 2001.

[8] N. Markatchev and C. Williamson, "WebTraff: A GUI for Web Proxy CacheWorkloadModeling and Analysis",IEEE MASCOTS, FortWorth, TX, October 2002.

[9] Duska, Bradley, David Marwood, and Michael Feely The Measured Access Characteristics of World-Wide-Web Client Proxy Caches, USENIX Symposium on Internet Technologies and Systems, December 1997.

[10] Wenying Feng, Yong Zhang. A general cache partition model for multiple QoS classes: algorithm and simulation. Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on. Volume 2, Issue , 28-30 Nov. 2005 Page(s): 544 - 549

[11] Miquel Moreto, Francisco J. Cazorla, Alex Ramirez, Mateo Valero. MLP-Aware Dynamic Cache Partitioning. Universidad Politecnica de Catalunya, Departamento de Arquitectura de Computadores. Centro Nacional de Supercomputacion. 16th International Conference on Parallel Architecture and Compilation Techniques. Posted to IEEE & CSDL on 4/3/2007 DOI 10.1109/L-CA.2007.3

[12] Miquel Moreto, Francisco J. Cazorla, Alex Ramirez and Mateo Valero. Explaining Dynamic Cache Partitioning Speed Ups. Universidad Politécnica de Cataluña, Computer Architecture Department, Spain. HiPEAC. European Network of Excellence on High-Performance Embedded Architecture and Compilation. IEEE Computer Architecture Letters, Vol. 6, 2007. Posted to IEEE & CSDL on 4/3/2007. DOI 10.1109/L-CA.2007.3

[13] Moinuddin K. Qureshi Yale N. Patt. Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches. Department of Electrical and Computer Engineering. The University of Texas at Austin.

[14] Peter Petrov and Alex Orailoglu, Member. Performance and Power Effectiveness in Embedded Processors—Customizable Partitioned Caches. IEEE transactions on computer-aided design of integrated circuits and systems, vol. 20, no. 11, noviembre 2001 pag.1309.

[15] David B. Kirk. Process Dependent Static Cache Partitioning for Real-Time Systems. Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 and IBM Systems Integration Division.

[16] Evangelos P. Markatos. “Main Memory Caching of Web Documents”.. Computer Architecture and VLSI Systems Group. Institute of Computer Science (ICS), Foundation for Research & Technology -- Hellas (FORTH). P.O.Box 1385

[17] C. Williamson, “On Filter Effects in Web Caching Hierarchies”, ACM Transactions on Internet Technology, 2002.

[18] F.J. González Cañete, E. Casilari, A. Triviño Cabrera. "Evaluación de Políticas de Control de Acceso en Cachés Web". Dpto. Tecnología Electrónica, Universidad de Málaga.Campus de Teatinos, 29071 Málaga

[19] Mudashiru Busari Carey Williamson, "Simulation Evaluation of a Heterogeneous Web Proxy Caching Hierarchy". Department of Computer Science,
University of Saskatchewan

---

[i] RFC 1321 (2008) http://www.faqs.org/rfcs/rfc1321.html
http://es.wikipedia.org/wiki/Algoritmo_MD5

[ii] RFC 2186 (2008)http://www.ircache.net/Cache/ICP/rfc2186.txt
RFC 2187 (2008)http://www.ircache.net/Cache/ICP/rfc2187.txt

[iii] IRCache (2007). Proxy-web traces.
http://www.ircache.net.

[iv] Squid, T. (2008). Squid web proxy cache.
http://www.squid-cache.org.