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

ANEXOS

DIEGO ALEJANDRO LEON MEDINA

TUTOR: JOSÉ ANTONIO GIL SALINAS

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

CONTENIDO

1 ANEXOS

A continuación se presentan los conceptos específicos para el montaje de la arquitectura propuesta, en una primera parte se verán las técnicas utilizadas para la preparación de los ficheros de simulación, en la segunda se presenta la dinámica utilizada para lanzar las simulaciones contra la arquitectura y en la tercera parte se presenta la forma como se realizan los análisis de los resultados obtenidos en los experimentos.

1.1 TECNICAS UTILIZADAS PARA LA PREPARACIÓN DEL FICHERO DE SIMULACIÓN

A manera de ejemplo se va a explicar 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.

1.1.1 Archivos de log del proxy de la UPV

Estas trazas se utilizaran con algunas modificaciones como herramienta para las simulaciones con la arquitectura propuesta.

Las trazas recogidas por estos ficheros de log, han registrado desde su funcionamiento, los tiempos, tamaños, tipos, entre otros parámetros, de los objetos WEB que han sido servidos a los clientes. 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.

1.1.1.1 Ejemplo de petición http registrada en los proxies Squid de la UPV

La siguiente es una petición http realizada a través de los proxies de la universidad politécnica de valencia:

http://www.jerc.net/eivissa_p.jpg

Ilustración 1 Ejemplo de petición HTTP realizada a los proxies de la UPV

1.1.1.2 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 2 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.

1.1.2 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 en primera instancia el formato final que queremos conseguir para cada registro dentro de la traza, para en seguida enseñar a transformarla.

1.1.2.1 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=MD5_URL&Size=TamañoBytesObjeto

Ilustración 3 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).

1.1.3 Herramientas utilizadas para la transformación de trazas de los ficheros "Access.log" en un fichero de simulación

Para la transformación de los ficheros se hizo necesario generar una serie de herramientas en múltiples lenguajes de programación para conseguir extraer los campos necesarios del "Access.log" y convertirlos en una petición http para ser utilizada en las simulaciones contra la arquitectura.

Se ha creado para este fin un Shell script llamado "prepFicheroNav.sh" que es el encargado de reunir todas y cada una de las herramientas necesarias para tratar los ficheros de navegación de la UPV. Cada una de las herramientas diseñadas será comentada a grosso modo, dado que en su directorio/paquete se encuentra un fichero "Leeme.txt" que puede ser consultado en caso de requerir información adicional.

La siguiente es la pantalla inicial que aparece al lanzar el Shell script "prepFicheroNav.sh":

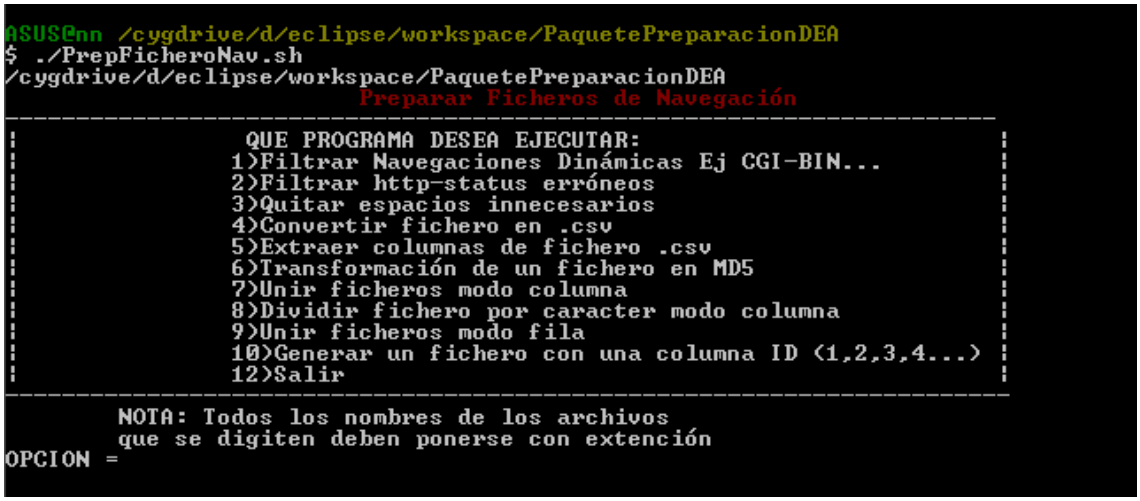

Ilustración 4 Menú principal del programa encargado de la transformación de los ficheros Access.log en ficheros de navegación para la arquitectura

Los siguientes son las opciones del menú del script "prepFicheroNav.sh", cada una de los subprogramas ha sido ordenada secuencialmente para preparar las navegaciones, con lo cual siguiendo en orden del 1 al 10, es posible partir del fichero "Access.log" y convertirlo en un fichero de navegación válido para ser utilizado contra la arquitectura.

1) Filtrar Navegaciones Dinámicas Ej CGI-BIN...
2) Filtrar http-status erróneos
3) Quitar espacios innecesarios
4) Convertir fichero en .csv
5) Extraer columnas de fichero .csv
6) Transformación de un fichero en MD5
7) Unir ficheros modo columna
8) Dividir fichero por carácter modo columna
9) Unir ficheros modo fila
10) Generar un fichero con una columna ID (1,2,3,4...)

A continuación se va a explicar la forma de utilizar cada uno de los subprogramas para preparar el fichero de navegación para la arquitectura.

1.1.3.1 Filtrar Navegaciones Dinámicas Ej CGI-BIN

El siguiente es la pantalla que se ejecuta con este subprograma encargado de realizar el filtrado de las navegaciones dinámicas de los ficheros de log del proxy de la UPV, el programa está desarrollado en el lenguaje de programación java y puede ser configurado a partir de otro fichero llamado "FILTRO.txt".

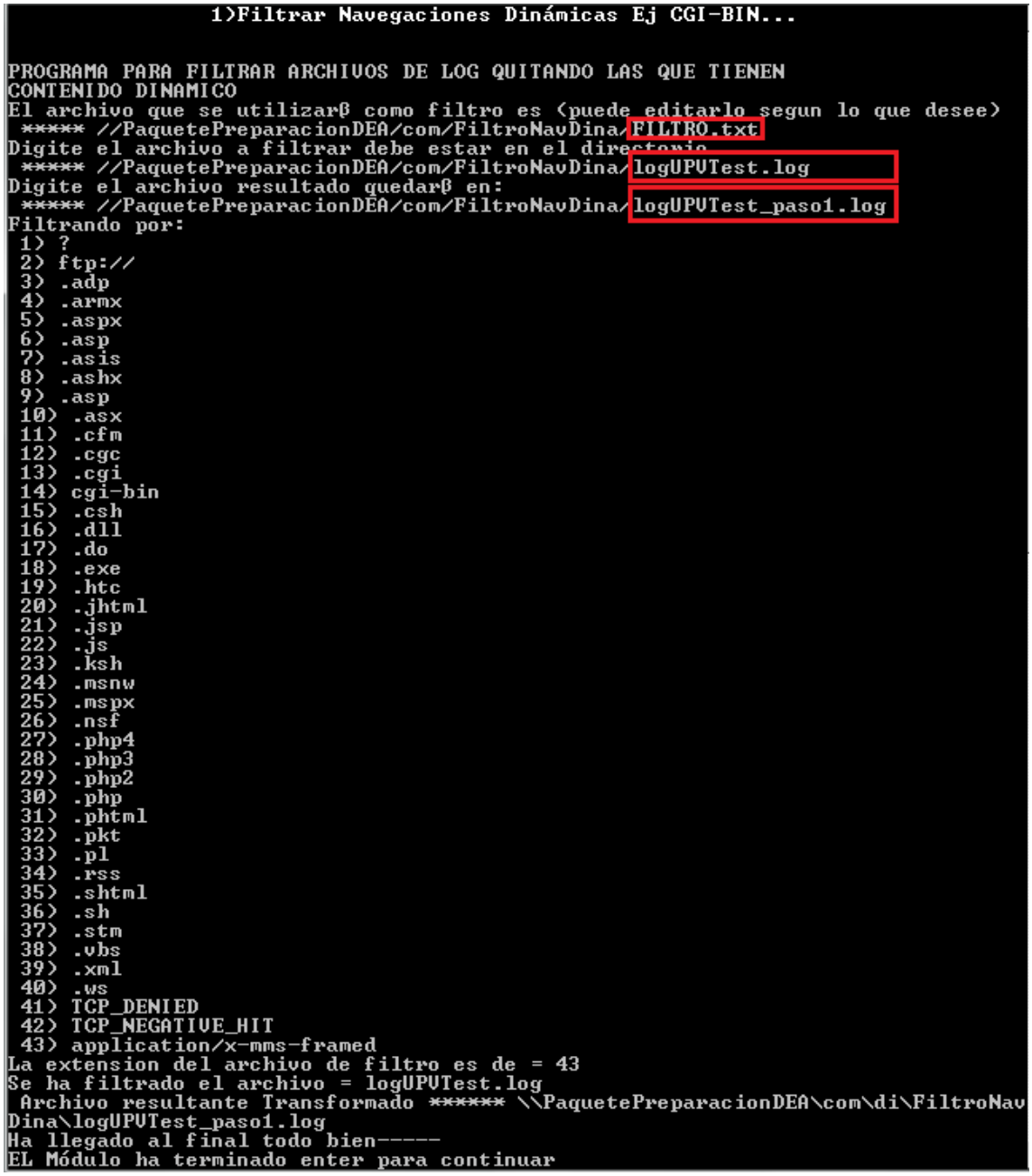

Ilustración 5 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Filtrar Navegaciones Dinámicas Ej CGI-BIN"

En rojo están resaltadas las partes fundamentales del script que son: el fichero que va a ser filtrado y el nombre que será utilizado en el fichero resultado. Adicionalmente se enseñan en pantalla todas las opciones del filtro configuradas en el fichero "FILTRO.txt"

La ejecución de esta opción del script hará que cualquier línea del log de la UPV que contenga alguna de las opciones del filtro sea eliminada.

1137279491.930 548 158.42.185.139 TCP_MISS/207 763 PROPFIND http://bay106.oe.hotmail.com/cgi-bin/hmdata/abel_gll@hotmail.com/folders/trAsH/ - DIRECT/65.54.161.253 text/xml

Ilustración 6 Ejemplo de línea del Access.log eliminada con el filtro de navegaciones dinámicas

Como pudo ser visto cgi-bin es el resultado de una navegación dinámica porque hace referencia a contenido servido en objetos que no pueden ser encontrados en la web, sino que son generados por servidores de aplicaciones. Dado que este proyecto en un futuro podría comparar resultados con navegaciones reales, entonces sólo interesa extraer de momento de los ficheros Access.log de la UPV, los objetos que puedan ser encontrados en la WEB (objetos estáticos).

1.1.3.2 Filtrar http-status erróneos

El siguiente paso a realizar en la transformación de los ficheros del proxy de la UPV es eliminar todos los errores de las navegaciones que corresponden en general a los códigos HTTP-status de la serie 400, 500 y 600, entre éstos y sólo a manera de ejemplo se encuentra el más común "404 Not Found". Esta opción puede ser configurada el fichero "FILTRO.txt", este fichero debe contener sólo los códigos que son aceptados como buenos, si se desean añadir más opciones al filtro se deben hacer con la notación "/200", que corresponde a la manera como vienen dadas en los ficheros log de la UPV. Este subprograma ha sido desarrollado en Java

La siguiente es la pantalla completa de una ejecución de esta opción del fichero "prepFicheroNav.sh".

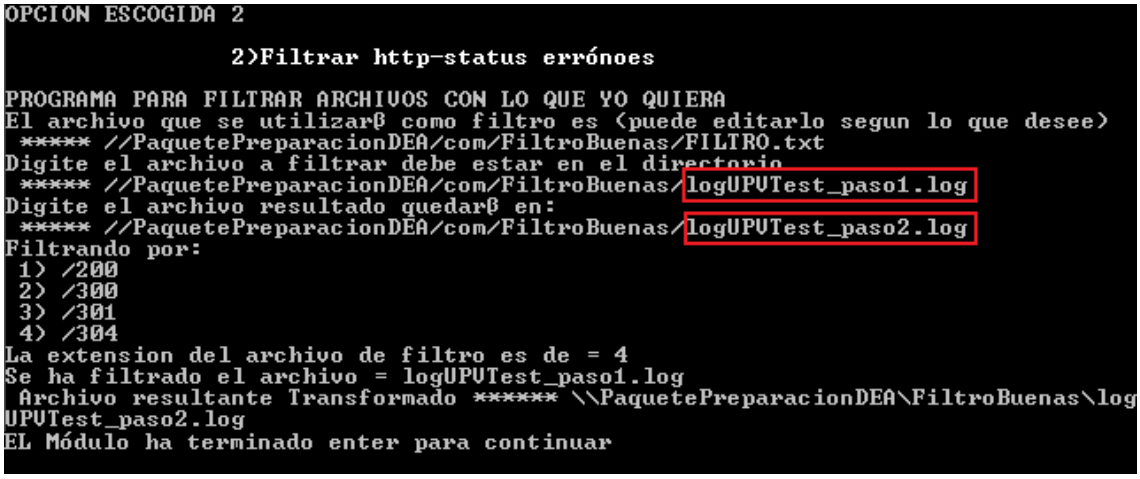

Ilustración 7 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Filtrar http-status erróneos"

En rojo están resaltados, el nombre del fichero al que vamos a cual se aplicará el filtro y el nombre que será puesto en el fichero donde será almacenado el resultado.

Este programa filtra de los ficheros Access.log de la UPV líneas como la siguiente:

1137279490.274 25 158.42.245.11 TCP_MISS/404 562 GET http://www.jerc.net/diada05_p.jpg - DIRECT/217.113.244.173 text/html

Ilustración 8 Ejemplo de línea del Access.log eliminada con el filtro de navegaciones con códigos http erróneos

En el ejemplo se observa como una línea que contenga ese valor es candidata a ser eliminada. Con ello se representa que el objeto .jpg que intentó descargarse no fue encontrado.

1.1.3.3 Quitar espacios innecesarios

La opción tres del script quitará los espacios innecesarios y formateará el fichero que estamos procesando a columnas separadas por un solo espacio. Esto permitirá más adelante, en la siguiente opción convertir el fichero Access.log de la UPV en un fichero CVS el cual es más procesable para extraer su información.

Este programa es un Shell script que utiliza el comando "sed"[1] La siguiente es la pantalla resultado de una ejecución de este subprograma:

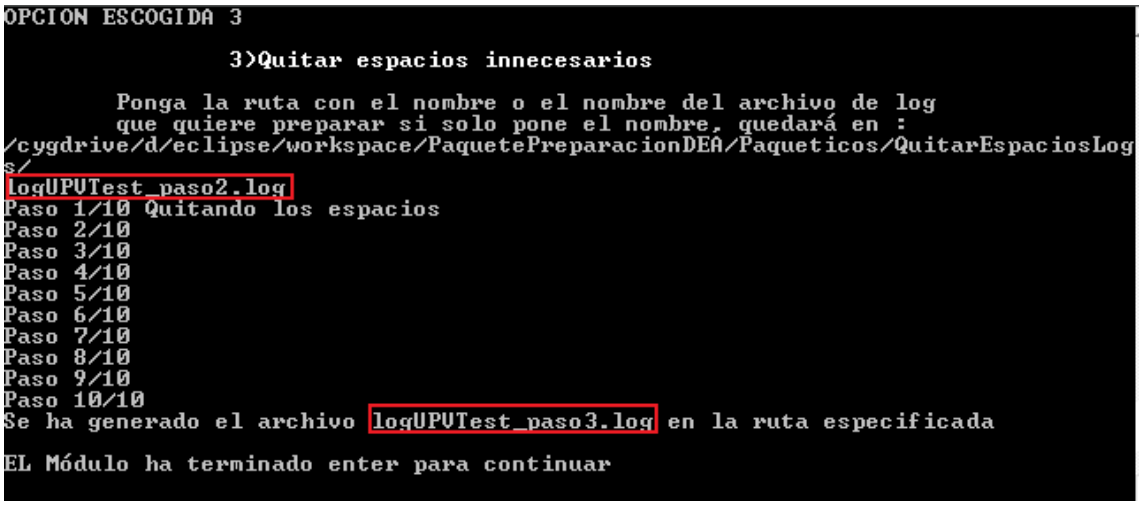

Ilustración 9 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Quitar espacios innecesarios"

La entrada es el resultado de la opción 2, de este se eliminarán durante ocho pasos los espacios no deseados, con esto se genera un nuevo fichero llamado logUPVTest_pase3.log que se continuará procesando en las siguiente opciones del script "prepFicheroNav.sh", en nuestro objetivo de conseguir un fichero con navegaciones que sirva para ser lanzado con la arquitectura propuesta en este documento.

1.1.3.4 Convertir fichero en .csv

Este tal vez constituye la opción más importante del script, porque toma el fichero generado en los pasos anteriores y lo convierte en un fichero .csv ("comma-separated values", Valores Separados por Coma), el cual es completamente manejable y analizable, según su extensión puede ser abierto con el Excel incluso o examinado con el lenguaje de programación AWK [2] .

Este subprograma fue desarrollado con el lenguaje de programación AWK, a continuación se presenta un ejemplo de una ejecución:

---

[1] Para información adicional se puede consultar el siguiente link: http://en.wikipedia.org/wiki/Sed

[2] Para más información sobre las capacidades de este lenguaje de programación el siguiente link tiene muy buena información al respecto: www.gnu.org/software/gawk/manual/gawk.html

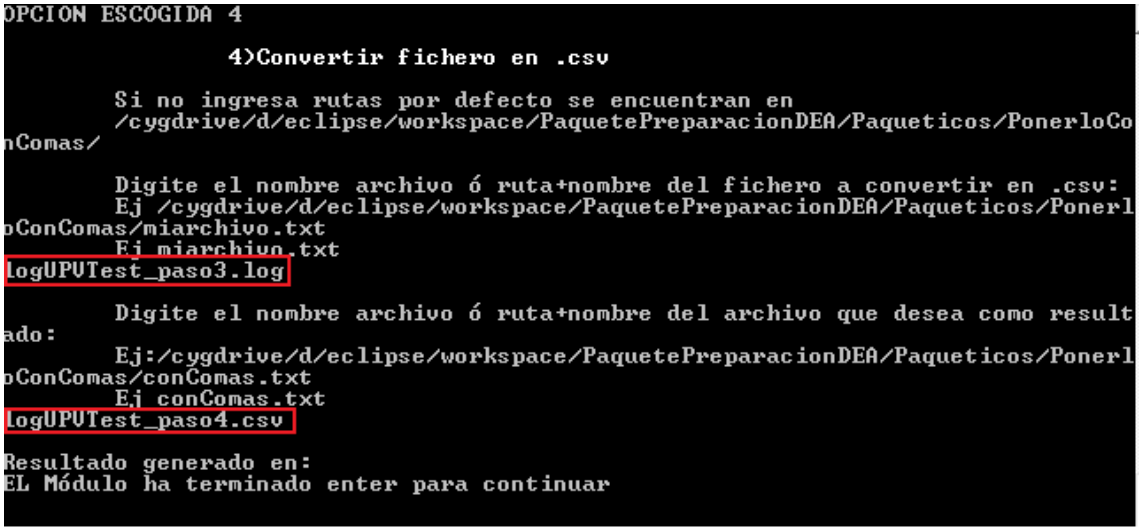

Ilustración 10 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Convertir fichero en .csv"

El fichero "LogUPVTest_paso3.log" es el fichero proveniente de los pasos anteriores que aún tiene la información del fichero "Access.log" del proxy de la UPV, filtrado de los datos indeseados para las simulaciones contra la arquitectura, el resultado de la ejecución se volcará sobre el fichero "LogUPVTest_paso4.csv".

Este es el último paso en el que se procesará el fichero "Access.log" filtrado con todos sus campos, a partir de aquí se extraerá sólo la información que se necesita para generar el fichero de simulaciones contra la arquitectura.

1.1.3.5 Extraer columnas de fichero .csv

La opción 5 del fichero "prepFicheroNav.sh" permite extraer cualquier columna del fichero .csv, esto permitirá extraer de un lado el campo URL del objeto (en la comuna 7 del fichero .csv) y el campo Bytes (ubicado en la columna 5 del fichero .csv, como se ha podido ver en secciones anteriores del documento). Para información adicional sobre cómo utilizar la utilidad referirse siempre a los ficheros Leeme.txt ubicados en los directorios de cada uno de los programas.

Para su información, este subprograma está escrito por la combinación de Shell script y de el lenguaje de programación AWK.

A continuación se presentan ejemplos sobre la manera de extraer el campo URL y Bytes del fichero .csv.

EXTRAER EL CAMPO BYTES:

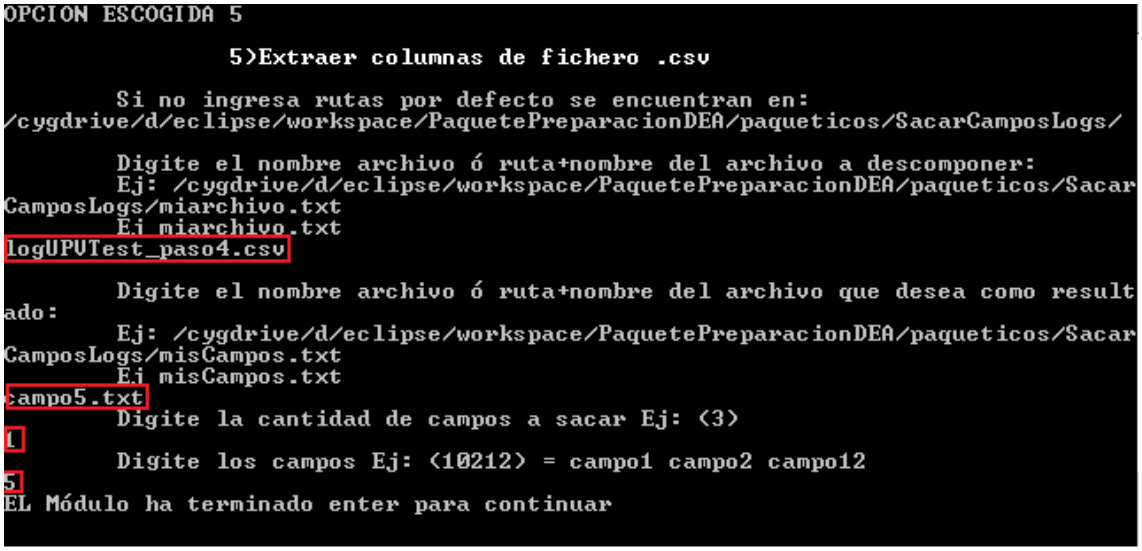

Ilustración 11 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Extraer columnas de fichero .csv", Extrayendo campo Bytes del Access.log de la UPV

En el fichero campo5.txt será un fichero con una columna de valores como los que se pueden ver a continuación:

25358
28420
31323
29322
…
…

Este es uno de los ficheros con los que se va a componer el fichero de simulación contra la arquitectura propuesta en este documento.

EXTRAER CAMPO URL

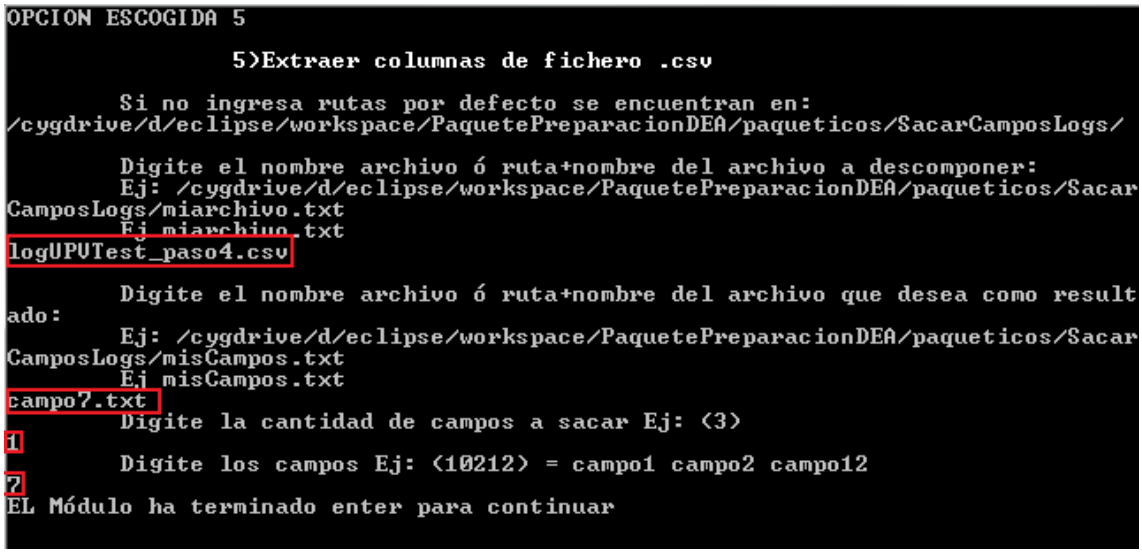

Ilustración 12 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Extraer columnas de fichero .csv", Extrayendo campo URL del Access.log de la UPV.

En el fichero ”campo7.txt” será un fichero con una columna de valores como los que se pueden apreciar a continuación:

```
http://www.jerc.net/eivissa_p.jpg
http://www.jerc.net/28juny05_p.jpg
http://www.jerc.net/dij_p.jpg
http://www.jerc.net/1maig_pX20copia.jpg
...
…
```

Este es el segundo fichero a ser usado para componer el fichero de simulación contra la arquitectura propuesta en este documento.

1.1.3.6 Transformación de un fichero en MD5

Una vez se tienen en un fichero (“campo7.txt”), las URLs de los objetos WEB que fueron visitados en la UPV, se hará la transformación MD5 de ese fichero, para facilitar la configuración final del fichero de simulaciones para la arquitectura dado que el MD5 tiene un tamaño fijo para cada uno de los valores transformados.

Adicionalmente a la transformación con MD5 este subprograma monta el resultado de la transformación con el prefijo y sufijo necesarios para construir las peticiones con el siguiente formato:

Prefijo: http://127.0.0.1:8080/lasletras/miServlet?URL=

Este prefijo corresponde a la URL del servidor que se encargará de servir las peticiones y como propiedad tendrá siempre en URL el valor en MD5 de la URL ejecutada.

Sufijo: &Size=

El sufijo corresponde a otra propiedad que describirá al tamaño del objeto solicitado al servidor, pero que de momento no tiene valor, porque éste será puesto en el siguiente paso del script “prepFicheroNav.sh”.

```
http://127.0.0.1:8080/lasletras/miServlet?URL=URL_con_MD5&Size=
```

Se ha utilizado una clave arbitraria para la transformación MD5, la siguiente es un ejemplo de una transformación realizada:

```
OPCION ESCOGIDA 6

                        6)Transformación de un fichero en MD5

rm: cannot remove `ConMd5.txt': No such file or directory
Key Utilizada en el MD5 = miproyectoeselmejor
Digite el nombre del archivo al cual le va a hacer el MD5 debe tenerlo en la car
peta \\PaquetePreparacionDEA\LogATrabajar\
\\PaquetePreparacionDEA\LogATrabajar\campo7.txt
Se estβ transformando el archivo \PaquetePreparacionDEA\LogATrabajar\campo7.txt
 Archivo resultante Transformado con MD5 dejado en****** \\PaquetePreparacionDEA
\Temporal\ConMd5.txt ******
EL Módulo ha terminado
```

Ilustración 13 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Transformación de un fichero en MD5"

En rojo se resalta el fichero que se desea transformar "campo7.txt" y como ejemplo se enseña a continuación como queda el resultado de la transformación en el fichero "ConMD5.txt"

URL del fichero campo7.txt	Resultado de la transformación MD5
http://www.jerc.net/eivissa_p.jpg http://www.jerc.net/28juny05_p.jpg http://www.jerc.net/dij_p.jpg http://www.jerc.net/1maig_pX20copia.jpg … …	http://127.0.0.1:8080/lasletras/miServlet?URL=65c7a9211be1517eff4c4e503f79b64a&Size= http://127.0.0.1:8080/lasletras/miServlet?URL=69d9bcd50d1f115a120500ac81904926&Size= http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size= http://127.0.0.1:8080/lasletras/miServlet?URL=2e411a34c0d38b8006af7ab418153de6&Size= …

En este punto de la ejecución del Shell script "prepFicheroNav.sh" se tiene casi montado el fichero a ser utilizado como entrada en las simulaciones de la arquitectura propuesta, tan solo faltando añadir el valor a la propiedad "Size" de cada uno de los peticiones de los objetos HTTP (Recordar, que esta información se encuentra en el fichero campo5.txt).

1.1.3.7 Unir ficheros modo columna

Este es el último paso para montar el fichero las peticiones http que serán utilizadas para lanzar las simulaciones contra la arquitectura. Como se ha visto en el paso anterior ya solo falta unir el campo5.txt del fichero "Access.log" al fichero "ConMD5.txt", para ello este programa se encarga de trabajar cualquier fichero de texto como si fuera una columna y unirlos con cualquier carácter que se desee en medio.

El programa está desarrollado en Java y tienen gran flexibilidad a la hora de unir en modo columna cualquier tipo de fichero.

La siguiente pantalla representa un ejemplo de la ejecución del subprograma:

```
OPCION ESCOGIDA 7

                7>Unir ficheros modo columna

PROGRAMA PARA UNIR CAMPOS
Digite el archivo con el campo1 a unir debe estar en el directorio
 ***** //PaquetePreparacionDEA/com/unir/ConMD5.txt
Digite el archivo con el campo2 a unir debe estar en el directorio
 ***** //PaquetePreparacionDEA/com/unir/campo5.txt
 ***** NOMBRE ARCHIVO RESULTADO =  //PaquetePreparacionDEA/com/unir/Simulacion.t
xt
Digite el String para poner en las posiciones  ııııCAMPO1????CAMPO2ıııı
ıııı =
???? =
ıııı =
Se ha unido el ConMD5.txtcampo5.txt
****
EL Módulo ha terminado
```

Ilustración 14 Pantalla de ejecución del script "prepFicheroNav.sh" opción "Unir ficheros modo columna"

Como su puede ver en Ilustración 14, el fichero que se genera llamado "**Simulación.txt**", es el fichero final que contiene todas las peticiones http extraídas del

fichero inicial “Access.log” del proxy de la UPV. Se ahora en adelante este fichero es un hito para el proyecto ya que constituye el elemento a poner a la entrada de la arquitectura. A partir de él se ejecutarán todas y cada una de las simulaciones que se deseen lanzar contra la arquitectura propuesta.

En el fichero “Simulación.txt” se tendrán peticiones http, como las que se pueden ver en la siguiente tabla:

http://127.0.0.1:8080/lasletras/miServlet?URL=65c7a9211be1517eff4c4e503f79b64a&Size=25358
http://127.0.0.1:8080/lasletras/miServlet?URL=69d9bcd50d1f115a120500ac81904926&Size=28420
http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=31323
http://127.0.0.1:8080/lasletras/miServlet?URL=2e411a34c0d38b8006af7ab418153de6&Size=29322

Estas son las URIs a ser usadas en las simulaciones, cada una representa a un objeto que ha sido consultado en la UPV y ha sido registrado en el “Access.log”.

1.1.3.8 Dividir fichero por carácter modo columna

Esta opción del script permite dividir un fichero de texto en dos partes, separadas por un carácter que sea común a cada una de sus líneas.

Esta utilidad ha sido desarrollada en AWK y ha sido pensada para trabajar con campos que vienen dentro del fichero “Access.log” del proxy de la UPV con el siguiente formato:

Ejemplo de algunos Campos de una petición registrada en el “Access.log” susceptibles a ser separados	**Resultado**
text/html	text html
TCP_MISS/404	TCP_MISS 404
DIRECT/217.113.244.173	DIRECT 217.113.244.173

Para esta utilidad no se presenta la pantalla de ejecución porque no se utilizará con un fin específico dentro de este documento.

1.1.3.9 Unir ficheros modo fila

Los ficheros “Access.log” del proxy de la UPV, se generan más o menos diariamente, con cual si se desea generar un fichero de Simulaciones para la arquitectura de varios días, será necesario unir muchos ficheros “Access.log” en uno solo; esto es lo que permite esta herramienta.

1.1.3.10 Generar un fichero con una columna ID (1,2,3,4...)

Una vez se extrae alguna información de los ficheros “Access.log” del proxy de la UPV, es posible que se requiera almacenar esta información en una base de datos, con lo cual puede ser necesario tener una columna con identificadores que mantenga ordenados los valores tal cual como fueron registrados originalmente por el proxy.

1.2 LANZANDO SIMULACIONES CONTRA LA ARQUITECTURA

La siguiente es la idea general de una simulación contra la arquitectura, a continuación se explicaran todas y cada una de las herramientas dispuestas para realizar las simulaciones:

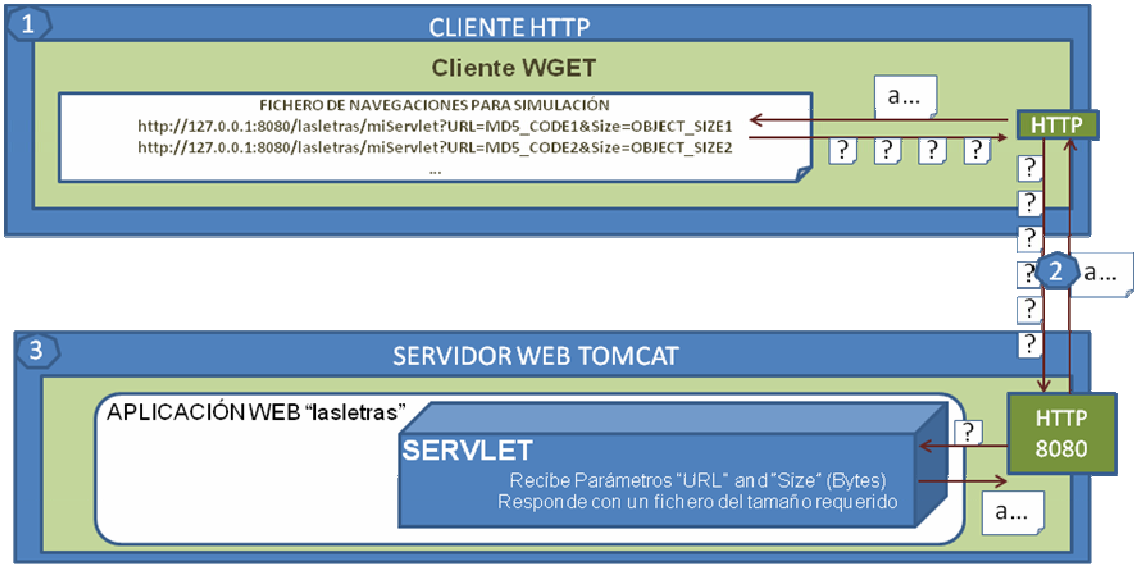

Ilustración 15 Esquema general de una simulación contra la arquitectura

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.

Se creó un shell script que fuera capaz de arrancar la arquitectura y ejecutar simulaciones de manera sencilla su nombre es "**ArrancadorSimulaciones.sh**", a continuación se presentan las opciones que permite el script en su menú principal:

```
ASUS@nn /usr/java/SIMULACIONES
$ ./ArrancadorSimulaciones.sh
                        MAKING-OF DEA
-------------------------------------------------------------------------
|                 QUE PROGRAMA DESEA EJECUTAR:                          |
|                1)Subir TOMCAT (Servidor de Objetos)                   |
|                2)Bajar TOMCAT                                         |
|                3)ENCENDER SQUID1 PADRE  HTTP(3128) ICP(3130)          |
|                4)ENCENDER SQUID2 HIJO   HTTP(3129) ICP(3131)          |
|                5)ENCENDER SQUID3 REFERENCIA HTTP(3127) ICP(3132)      |
|                6)LANZAR SIMULACION CON WGET A PADRE-HIJO              |
|                7)LANZAR SIMULACION CON WGET A REFERENCIA              |
|                8)Salir                                                |
-------------------------------------------------------------------------
        NOTA: Todos los nombres de los archivos
        que se digiten deben ponerse con extension
OPCION =
```

Ilustración 16 Menú principal del programa arrancador de la arquitectura y simulaciones

En los puntos siguientes se va a ver como se arrancan las simulaciones contra la arquitectura utilizando las opciones propuestas en el menú del script "ArrancadorSimulaciones.sh".

1.2.1 Encendiendo la arquitectura

Del menú visto en la sección 0 las 5 primeras opciones nos permiten encender la arquitectura, el resto de opciones está pensada para lanzar las simulaciones contra la arquitectura.

1.2.2 Arrancar el Servidor Web Tomcat con el JAVA servlet "lasletras"

Ejecutando el primer comando del menú se enseña la siguiente salida:

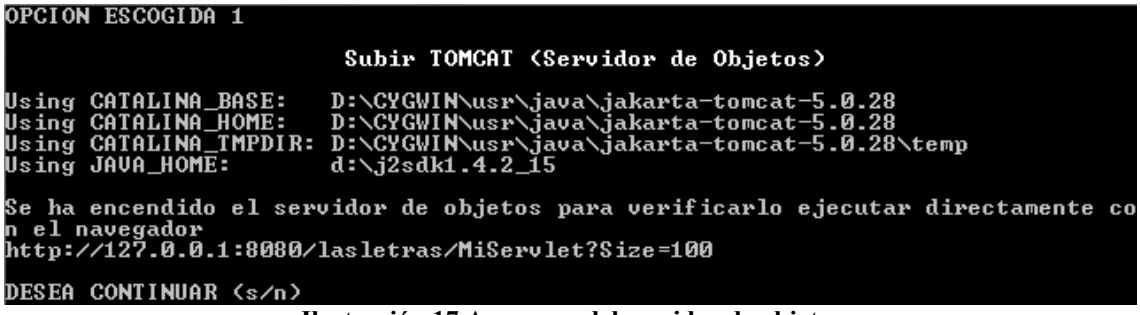

Ilustración 17 Arranque del servidor de objetos

Como se puede ver en la Ilustración 17 el arranque del servidor Tomcat muestra variables que tiene configuradas por defecto (CATALINA_BASE, CATALINA_HOME, CATALINA_TMPDIR, rutas donde se encuentran instalado el servidor Tomcat, que debe ser tenido muy en cuenta si se desea replicar la configuración de la arquitectura) de otro lado se enseña donde se encuentra el JAVA_HOME, esta variable para usuarios poco avanzados, indica donde se encuentra el JDK , el cual es utilizando por el servidor Tomcat para desplegar las aplicaciones web.

A continuación se da una explicación de cómo se puede probar en el navegador si el servidor web se ha iniciado correctamente. Ejecutando la URL *http://127.0.0.1:8080/lasletras/MiServlet?Size=100*.

Profundizando un poco más lo que se ha hecho ejecutando la primera opción del menú, es arrancar el servidor de objetos encargado de generar los objetos con los tamaños deseados, este servidor está pensado para servir objetos respondiendo a peticiones del tipo:

```
http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=3132300
```

Ilustración 18 Modelo de petición lanzado contra el servidor de objetos

Donde:

http://127.0.0.1:8080/lasletras/miServlet: es el servlet encargado de devolver los objetos. Entonces en el contexto web "lasletras" se encuentra montado un servlet programado en java y desplegado en el servidor Tomcat llamado "miServlet".

URL=0d46cb9a0ce4c2c07691acb45859fc60: Este parámetro para el servlet indica la URL de la petición que se está haciendo codificada mediante el algoritmo MD5[3].

Size=3132300: El parámetro "Size" indica el tamaño en bytes del objeto que se está solicitando.

- Evaluación de prestaciones del servidor de objetos:

Evaluación prestaciones del servlet lasletras para la generación de los objetos para las simulaciones:

A continuación se presenta el tiempo en que el servlet tarda en descargar 50000000 Bytes y almacenarlos en disco.

```
ASUS@nn ~
$ time wget http://127.0.0.1:8080/lasletras/miServlet?Size=50000000
--2008-07-29 23:05:38--  http://127.0.0.1:8080/lasletras/miServlet?Size=50000000

Connecting to 127.0.0.1:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: `miServlet@Size=50000000'

    [                <=>                 ] 50,000,000  13.9M/s   in 3.4s

2008-07-29 23:05:42 (14.0 MB/s) - `miServlet@Size=50000000' saved [50000000]


real    0m3.736s
user    0m0.202s
sys     0m0.998s
```

Ilustración 19 Evaluación de prestaciones del servidor de objetos

Como pudo observarse en la ejecución del comando el siguiente fichero ha sido creado en disco llamado "miServlet@Size=500000000" y su tamaño es de 50000000 bytes.

```
48832 -rw-r--r--   1 ASUS mkgroup 50000000 Jul 29 23:05 miServlet@Size=50000000
```

Ilustración 20 Fichero creado después de la ejecución de una petición http con el cliente Wget contra el servidor de objetos (Servet "lasletras").

Conclusión:

Se ha demostrado en este punto que el servidor de objetos funciona con autonomía del parámetro URL que se le ajuste. De otro lado se ha evaluado el máximo performance que puede ofrecer a la hora de servir objetos de gran tamaño y de otro lado se ha inferido la capacidad de ofrecer objetos a los clientes web del tamaño deseado.

1.2.3 Detener el servidor Tomcat

La segunda opción detiene la ejecución del servidor de objetos de la arquitectura, esto permite limpiar caches y matar el proceso encargado de la ejecución del servidor TOMCAT.

---

[3] Message-Digest Algorithm 5, RFC1321 - The MD5 Message-Digest Algorithm, es un RFC utilizando Genera un resumen de 128 bits del mensaje que se desee (En el caso de la arquitectura lo hace sobre la URL que se está procesando).

La siguiente es la pantalla que se muestra cuando el servidor es detenido:

Ilustración 21 Parada del servidor de objetos

Se resumen, en la pantalla como se ha detenido el servidor. Si se detuviera sin estar arrancado la pantalla mostraría una excepción java de “Connection refused”, típica cuando no se puede establecer conexión con un recurso.

1.2.4 Arrancando el servidor proxy squid padre

Con la tercera opción del menú se arranca el servidor proxy padre de la arquitectura, esta opción hace cuatro tareas fundamentales:

- Borrar los ficheros de log, que ya se hayan creado de ejecuciones anteriores, permitiendo arrancar una nueva simulación partiendo de 0.
- Borrar la cache de una simulación previa; cada nueva medición debe partir de una cache vacía para conseguir datos coherentes a lo largo de las simulaciones.
- Crear nuevos directorios para la memoria swap del proxy, esto permite que se cree la jerarquía de directorios que van a ser usados en la web cache del proxy.
- Arranque del servidor proxy de acuerdo con la configuración que se haya puesto en el fichero squid.conf[4] del proxy padre.

```
OPCION ESCOGIDA 3

               3)   ENCENDER SQUID1 PADRE  HTTP(3128) ICP(3130)

       Borrando los logs
2008/08/04 23:49:23| Creating Swap Directories
2008/08/04 23:49:26| Starting Squid Cache version 2.6.STABLE7 for i686-pc-cygwin
...
2008/08/04 23:49:26| Running on Windows Vista
2008/08/04 23:49:26| Process ID 1217468
2008/08/04 23:49:26| With 8192 file descriptors available
2008/08/04 23:49:26| Using poll for the IO loop
2008/08/04 23:49:26| Performing DNS Tests...
2008/08/04 23:49:26| Successful DNS name lookup tests...
2008/08/04 23:49:26| DNS Socket created at 0.0.0.0, port 53572, FD 4
2008/08/04 23:49:26| /etc/resolv.conf: (2) No such file or directory
2008/08/04 23:49:26| Adding DHCP nameserver 62.42.230.24 from Registry
2008/08/04 23:49:26| Adding DHCP nameserver 62.42.63.52 from Registry
2008/08/04 23:49:26| Adding DHCP nameserver 62.42.230.24 from Registry
2008/08/04 23:49:26| Adding DHCP nameserver 62.42.63.52 from Registry
2008/08/04 23:49:26| Unlinkd pipe opened on FD 10
2008/08/04 23:49:26| Swap maxSize 102400 KB, estimated 34133 objects
2008/08/04 23:49:26| Target number of buckets: 1706
2008/08/04 23:49:26| Using 8192 Store buckets
2008/08/04 23:49:26| Max Mem  size: 8192 KB
2008/08/04 23:49:26| Max Swap size: 102400 KB
2008/08/04 23:49:26| Rebuilding storage in /usr/local/squid1/var/cache (DIRTY)
2008/08/04 23:49:26| Using Least Load store dir selection
2008/08/04 23:49:26| Set Current Directory to /usr/local/squid1/var/cache
2008/08/04 23:49:26| Loaded Icons.
2008/08/04 23:49:26| Accepting proxy HTTP connections at 127.0.0.1, port 3128, F
D 11.
2008/08/04 23:49:26| commSetNoLinger: FD 12: (109) Protocol not available
2008/08/04 23:49:26| Accepting ICP messages at 0.0.0.0, port 3130, FD 12.
2008/08/04 23:49:26| WCCP Disabled.
2008/08/04 23:49:26| Configuring Sibling 127.0.0.1/3129/3131
2008/08/04 23:49:26| Ready to serve requests.
2008/08/04 23:49:28| Done scanning /usr/local/squid1/var/cache (0 entries)
2008/08/04 23:49:28| Finished rebuilding storage from disk.
2008/08/04 23:49:28|         0 Entries scanned
2008/08/04 23:49:28|         0 Invalid entries.
2008/08/04 23:49:28|         0 With invalid flags.
2008/08/04 23:49:28|         0 Objects loaded.
2008/08/04 23:49:28|         0 Objects expired.
2008/08/04 23:49:28|         0 Objects cancelled.
2008/08/04 23:49:28|         0 Duplicate URLs purged.
2008/08/04 23:49:28|         0 Swapfile clashes avoided.
2008/08/04 23:49:28|   Took 2.3 seconds (   0.0 objects/sec).
2008/08/04 23:49:28| Beginning Validation Procedure
2008/08/04 23:49:29|   Completed Validation Procedure
2008/08/04 23:49:29|   Validated 0 Entries
2008/08/04 23:49:29|   store_swap_size = 0k
2008/08/04 23:49:29| storeLateRelease: released 0 objects
```

Ilustración 22 Arranque del servidor proxy padre de la arquitectura

[4] "Squid.conf" es el fichero de configuración por defecto que utilizan los servidores proxy squid, este se encuentra por defecto en la siguiente ruta #<squid>/etc/Squid.conf

Como se pudo observar en la Ilustración 22, la opción 3 del script "ArrancadorSimulaciones.sh", arranca el servidor proxy padre, que funcionará recibiendo peticiones http en el puerto "3128" y comunicándose con los otros proxies de la jerarquía a través del puerto "3130".

Testeando el correcto arranque proxy padre:

Se puede probar si el proxy padre ha arrancado correctamente, utilizando algún cliente http que busque una URL o algún objeto a través del proxy.

Para ello se puede configurar el Mozilla Firefox[5] para que acceda a alguna dirección de internet a través del proxy padre.

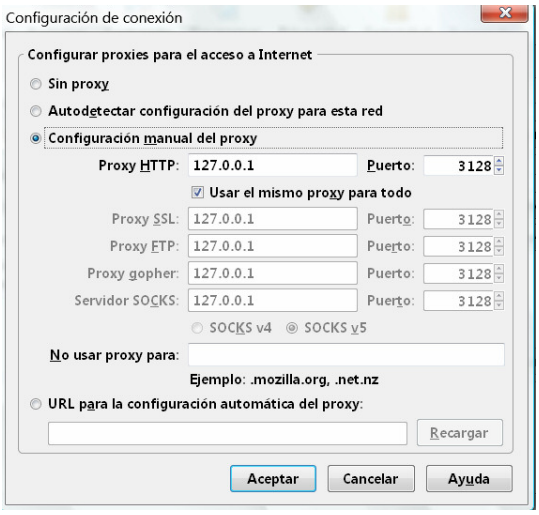

Ilustración 23 Configuración de la conexión del Mozilla Firefox para el test arranque correcto del servidor proxy padre

Luego se puede poner cualquier url que se desee y este elemento de la arquitectura devolverá la respuesta http con los objetos solicitados.

A continuación se van a probar dos peticiones objetos web a través del servidor proxy padre:

	Url del objeto	Tamaño Objeto (bytes)
URL 1	http://www.upv.es/images/p0.gif	4408
URL 2	http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=4408	4408
URL 3	http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=1000000	1000000

Resultado de las peticiones:

En el fichero de "store.log" del servidor proxy padre se puede observar que las dos peticiones de idéntico tamaño han sido cacheadas exitosamente por el servidor y la

[5] Navegador Web, www.mozilla.org/

petición que en tamaño era más grande que la aceptada por la web cache no ha sido cacheada, con lo cual se puede concluir que el servidor padre se encuentra funcionando correctamente.

URL EJECUTADA	Resultado de la petición en el fichero "store.log" del servidor proxy padre
URL 1	1217891363.008 **SWAPOUT** 00 00000007 03886F35733FC5BCAFE22B34F3A500FE 200 1217894974 1217410603 -1 image/gif **4408/4408** GET http://www.upv.es/noticias-upv/imagenes/verano_08.gif
URL 2	1217891459.713 **SWAPOUT** 00 00000008 229B4CBE480E7D465ADC62BEE255BDD3 200 1217895059 -1 -1 text/plain **4408/4408** GET http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=4408
URL 3	1217970526.605 **RELEASE** 00 00000000 3D1E2D8E42E4198626637EB5A2F84E44 200 1217974124 -1 -1 text/plain -1/1000000 GET http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=1000000

Tabla 1 Peticiones http de prueba contra el servidor proxy padre de la arquitectura

Como se puede ver en la Tabla 1, las dos peticiones de igual tamaño han sido servidas y correctamente cacheadas en el servidor padre, dado que su código de acción con los objetos fue "SWAPOUT"[6]. Para la URL 3 el objeto tiene el código de acción "RELEASE"[7] que indica que el objeto no ha sido cacheado en cache web del padre.

Conclusión:

Se ha demostrado en este punto que:

- El servidor proxy padre es capaz de recibir peticiones http y guardarlas correctamente en su cache web, si está habilitado para ello.
- Las peticiones http al servidor de objetos de la arquitectura también puedes ser atendidas y cacheadas por el servidor proxy padre.
- Los puertos configurados para este elemento de la arquitectura (proxy padre) funcionan correctamente.

Deteniendo el servidor proxy padre:

Una vez que hemos terminado de ejecutar una simulación o de testear algo con el servidor proxy padre, se puede detener la ejecución con la combinación de teclas "ctrl+c", a continuación se mostrará en pantalla:

Ilustración 24 Shutdown del servidor proxy padre de la arquitectura

[6] "SWAPOUT" Indica que el objeto fue correctamente cacheado en disco en al cache web.
[7] "RELEASE" Indica que el objeto fue removido de la cache, una vez ha sido servido al cliente http.

Como se puede observar, se detiene la ejecución satisfactoriamente y hace un resumen de cómo se ha detenido el servidor proxy.

1.2.5 Arrancando el servidor proxy squid hijo

Con la cuarta opción del menú se arranca el servidor proxy hijo de la arquitectura, esta opción hace cuatro tareas fundamentales, similares a las realizadas con la opción 3 del script de arranque:

- Borrar los ficheros de log, que ya se hayan creado de ejecuciones anteriores, permitiendo arrancar una nueva simulación partiendo de 0.
- Borrar la cache de una simulación previa; cada nueva medición debe partir de una cache vacía para conseguir datos coherentes a lo largo de las simulaciones.
- Crear nuevos directorios para la memoria swap del proxy, esto permite que se cree la jerarquía de directorios que van a ser usados en la web cache del proxy.
- Arranque del servidor proxy de acuerdo con la configuración que se haya puesto en el fichero squid.conf[8] del proxy hijo.

```
OPCION ESCOGIDA 4

               4)  ENCENDER SQUID2 HIJO   HTTP(3129) ICP(3131)

        Borrando los logs
2008/08/05 01:37:54| Creating Swap Directories
2008/08/05 01:37:57| Starting Squid Cache version 2.6.STABLE7 for i686-pc-cygwin
...
2008/08/05 01:37:57| Running on Windows Vista
2008/08/05 01:37:57| Process ID 1579584
2008/08/05 01:37:57| With 8192 file descriptors available
2008/08/05 01:37:57| Using poll for the IO loop
2008/08/05 01:37:57| Performing DNS Tests...
2008/08/05 01:37:57| Successful DNS name lookup tests...
2008/08/05 01:37:57| DNS Socket created at 0.0.0.0, port 55875, FD 4
2008/08/05 01:37:57| /etc/resolv.conf: (2) No such file or directory
2008/08/05 01:37:57| Adding DHCP nameserver 62.42.230.24 from Registry
2008/08/05 01:37:57| Adding DHCP nameserver 62.42.63.52 from Registry
2008/08/05 01:37:57| Adding DHCP nameserver 62.42.230.24 from Registry
2008/08/05 01:37:57| Adding DHCP nameserver 62.42.63.52 from Registry
2008/08/05 01:37:57| Unlinkd pipe opened on FD 9
2008/08/05 01:37:57| Swap maxSize 102400 KB, estimated 73 objects
2008/08/05 01:37:57| Target number of buckets: 3
2008/08/05 01:37:57| Using 8192 Store buckets
2008/08/05 01:37:57| Max Mem  size: 8192 KB
2008/08/05 01:37:57| Max Swap size: 102400 KB
2008/08/05 01:37:57| Rebuilding storage in /usr/local/squid2/var/cache (DIRTY)
2008/08/05 01:37:57| Using Least Load store dir selection
2008/08/05 01:37:57| Set Current Directory to /usr/local/squid2/var/cache
2008/08/05 01:37:57| Loaded Icons.
2008/08/05 01:37:57| Accepting proxy HTTP connections at 127.0.0.1, port 3129, F
D 10.
2008/08/05 01:37:57| commSetNoLinger: FD 11: (109) Protocol not available
2008/08/05 01:37:57| Accepting ICP messages at 0.0.0.0, port 3131, FD 11.
2008/08/05 01:37:57| WCCP Disabled.
2008/08/05 01:37:57| Configuring Parent 127.0.0.1/3128/3130
2008/08/05 01:37:57| Ready to serve requests.
2008/08/05 01:37:59| Done scanning /usr/local/squid2/var/cache (0 entries)
2008/08/05 01:37:59| Finished rebuilding storage from disk.
2008/08/05 01:37:59|         0 Entries scanned
2008/08/05 01:37:59|         0 Invalid entries.
2008/08/05 01:37:59|         0 With invalid flags.
2008/08/05 01:37:59|         0 Objects loaded.
2008/08/05 01:37:59|         0 Objects expired.
2008/08/05 01:37:59|         0 Objects cancelled.
2008/08/05 01:37:59|         0 Duplicate URLs purged.
2008/08/05 01:37:59|         0 Swapfile clashes avoided.
2008/08/05 01:37:59|   Took 2.2 seconds (   0.0 objects/sec).
2008/08/05 01:37:59| Beginning Validation Procedure
2008/08/05 01:37:59|   Completed Validation Procedure
2008/08/05 01:37:59|   Validated 0 Entries
2008/08/05 01:37:59|   store_swap_size = 0k
2008/08/05 01:38:00| storeLateRelease: released 0 objects
```

Ilustración 25 Arranque del servidor proxy hijo de la arquitectura

[8] "Squid.conf" es el fichero de configuración por defecto que utilizan los servidores proxy squid, este se encuentra por defecto en la siguiente ruta #<squid>/etc/Squid.conf

Como se observa en la Ilustración 25, la opción 4 del script "ArrancadorSimulaciones.sh", arranca el servidor proxy hijo, que funcionará recibiendo peticiones http en el puerto "3129" y comunicándose con los otros proxies de la jerarquía a través del puerto "3131".

Testeando el correcto arranque proxy padre:

Se puede probar si el proxy padre ha arrancado correctamente, utilizando algún cliente http que busque una URL o algún objeto a través del proxy.

Para ello se puede configurar la conexión del Mozilla Firefox[9] para que acceda a alguna dirección de internet a través del proxy padre.

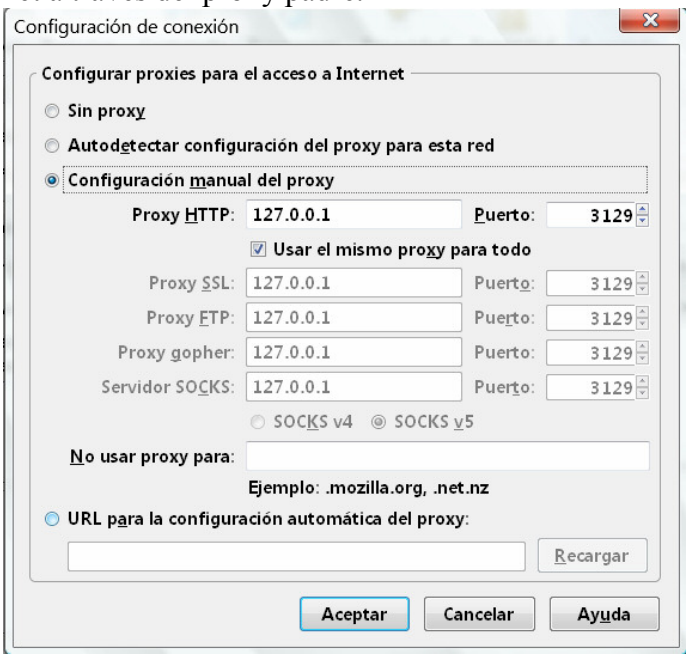

Ilustración 26 Configuración de la conexión del Mozilla Firefox para el test arranque correcto del servidor proxy hijo

Para probar que el proxy hijo funcione correctamente se debe realizar un test similar al realizado al proxy padre, pero esta vez los dos objetos que se testearán deben ser mayores en tamaño para que sean cacheadas por el servidor proxy hijo. No se debe perder nunca de vista que el proxy padre será el encargado de cachear objetos pequeños, mientras que el proxy hijo se encargará de cachear objetos grandes.

Ahora, siguiendo el mismo procedimiento realizado con el proxy padre, se van a probar dos peticiones objetos web a través del servidor proxy hijo:

	Url del objeto	Tamaño Objeto (bytes)
URL 1	http://www.etsmre.upv.es/Secretaria/JPG/LogoUPV.jpg	118828
URL 2	http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=118828	118828

[9] Navegador Web, www.mozilla.org/

URL 3	http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=5000000	5000000

Resultado de las peticiones:

En el fichero de "store.log" del servidor proxy hijo se puede observar que las dos peticiones de idéntico tamaño han sido cacheadas exitosamente por el servidor y la petición que en tamaño era más grande que la aceptada por la web cache no ha sido cacheada, con lo cual se puede concluir que el servidor padre se encuentra funcionando correctamente. La siguiente tabla resume estos resultados:

URL EJECUTADA	Resultado de la petición en el fichero "store.log" del servidor proxy padre
URL 1	1217894428.511 **SWAPOUT** 00 00000000 A402B966719AB11E0467BCF86B05C398 200 1217898039 1129026398 -1 image/jpeg **118828/118828** GET http://www.etsmre.upv.es/Secretaria/JPG/LogoUPV.jpg
URL 2	1217894585.561 **SWAPOUT** 00 00000001 C97E83EDB12A216EFD36D93072F7FF03 200 1217898185 -1 -1 text/plain **118828/118828** GET http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=118828
URL 3	1217894676.400 **RELEASE** 00 00000002 33103A66F8CE0E4D671ED92A73161AB5 200 1217898223 -1 -1 text/plain -1/5000000 GET http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=5000000

Tabla 2 Peticiones http de prueba contra el servidor proxy hijo de la arquitectura

Como se puede ver en la Tabla 2, las dos peticiones de igual tamaño han sido servidas y correctamente y cacheadas en el servidor hijo, dado que su código de acción con los objetos fue "SWAPOUT"[10], para la URL 3 se tiene un código de acción del servidor hijo de "RELEASE"[11] lo cual indica que el objeto no ha sido aceptado en la cache y una vez servido ha sido descartado. Es de notar la el correcto funcionamiento de las peticiones realizadas al servidor de objetos de la arquitectura, los resultado son completamente satisfactorios.

Conclusión

En este punto se ha demostrado lo siguiente:

- Es posible arrancar el servidor proxy hijo que hace parte de la arquitectura planteada, por medio del script general llamado "ArrancadorSimulaciones.sh"
- De otro lado se ha visto que su configuración de puertos es la correcta para el tratamiento de las peticiones http.
- Se pudo evaluar que la cache web del servidor proxy hijo acepta solo objetos del tamaño indicado en su configuración.
- Se ha demostrado que peticiones realizadas al servidor de objetos utilizando como pasarela el servidor proxy hijo funcionan correctamente.

Deteniendo el servidor proxy hijo:

Una vez que hemos terminado de ejecutar una simulación o de testear algo con el servidor proxy hijo, se puede parar la ejecución con la combinación de teclas "ctrl+c", lo siguiente imagen ilustrará lo que será mostrado en pantalla:

[10] "SWAPOUT" Indica que el objeto fue correctamente cacheado en disco en al cache web.
[11] "RELEASE" Indica que el objeto fue removido de la cache, una vez ha sido servido al cliente http.

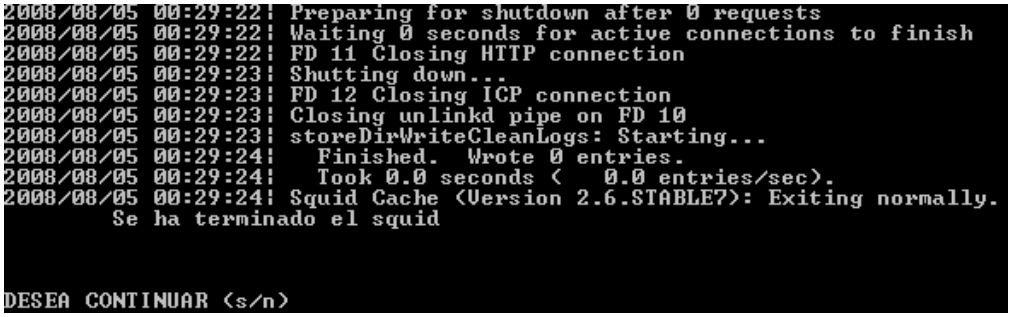

Ilustración 27 Shutdown del servidor proxy hijo de la arquitectura

Como se puede observar, se detiene la ejecución satisfactoriamente y hace un resumen de cómo se ha detenido el servidor proxy.

1.2.6 Arrancando el servidor proxy de referencia (configuración por defecto)

Con la quinta opción del menú se arranca el servidor proxy de referencia que será usado para realizar la toma de mediciones de referencia propuestas en la arquitectura, esta opción hace cuatro tareas fundamentales, similares a las realizadas con la opción 3 y 4 del script de arranque:

- Borrar los ficheros de log, que ya se hayan creado de ejecuciones anteriores, permitiendo arrancar una nueva simulación partiendo de 0.
- Borrar la cache de una simulación previa; cada nueva medición debe partir de una cache vacía para conseguir datos coherentes a lo largo de las simulaciones.
- Crear nuevos directorios para la memoria swap del proxy, esto permite que se cree la jerarquía de directorios que van a ser usados en la web cache del proxy.
- Arranque del servidor proxy de acuerdo con la configuración que se haya puesto en el fichero squid.conf[12] del proxy de referencia.

[12] "Squid.conf" es el fichero de configuración por defecto que utilizan los servidores proxy squid, este se encuentra por defecto en la siguiente ruta #<squid>/etc/Squid.conf

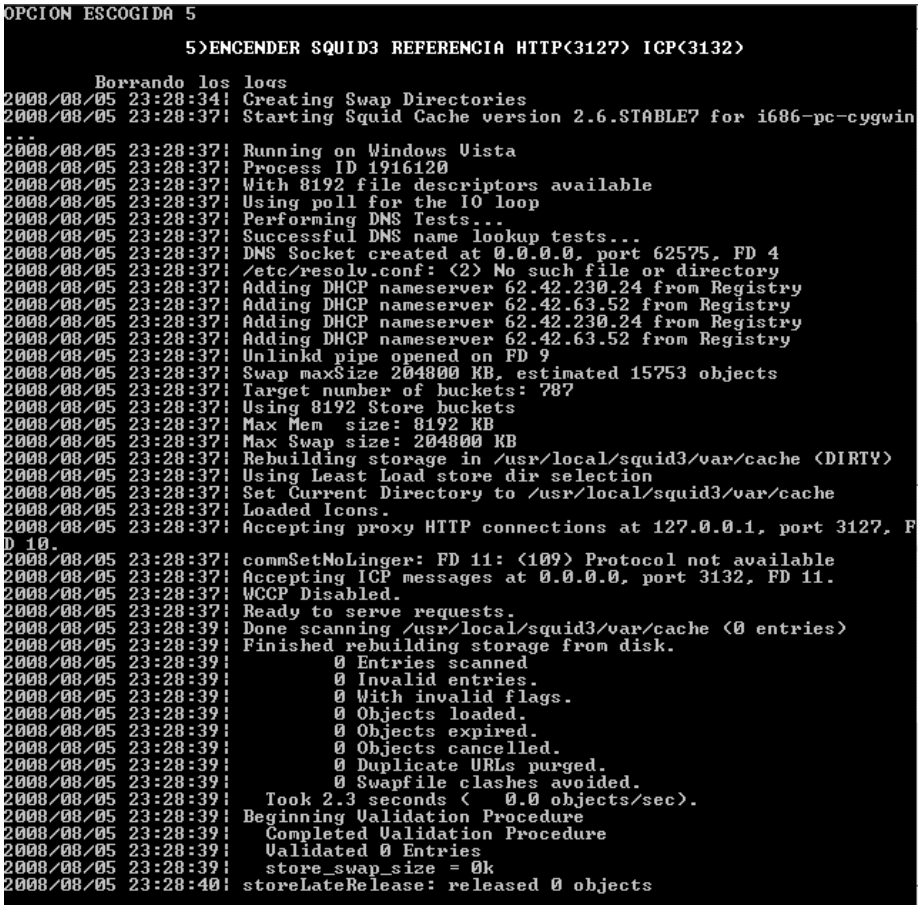

Ilustración 28 Arranque del servidor proxy de referencia de la arquitectura

Como se observa en la Ilustración 28, la opción 5 del script "ArrancadorSimulaciones.sh", arranca el servidor proxy de referencia de la arquitectura, que funcionará recibiendo peticiones http en el puerto "3127", sólo a manera de información se ha configurado el puerto "3132" como puerto ICT, pero dado que este proxy funcionará independientemente y no hará parte de una jerarquía de servidores proxies, entonces esta información no es muy relevante.

Testeando el correcto arranque del servidor proxy de referencia:

Similar a las pruebas realizadas para los otros servidores proxy, se va a probar en esta ocasión si el proxy de referencia ha arrancado y funciona correctamente, utilizando algún cliente http que busque una URL o algún objeto a través del proxy.

Para ello se puede configurar la conexión del Mozilla Firefox[13] para que acceda a alguna dirección de internet a través del proxy de referencia, como se puede ver a continuación:

[13] Navegador Web, www.mozilla.org/

Ilustración 29 Configuración de la conexión del Mozilla Firefox para el test arranque correcto del servidor proxy de referencia

Se van a realizar tres peticiones http contra el servidor proxy utilizado de referencia en la arquitectura, siguiendo el esquema planteado para los otros servidores proxy de la arquitectura las peticiones a realizar son las siguientes:

	Url del objeto	Tamaño Objeto (bytes)
URL 1	http://www.iad.upv.es/mapa2.jpg	265436
URL 2	http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=265436	265436
URL 3	http://127.0.0.1:8080/lasletras/miServlet?URL=0d46cb9a0ce4c2c07691acb45859fc60&Size=5000000	5000000

De las tres URLs planteadas se espera que las dos primeras sean cacheadas correctamente en la cache del servidor proxy, pero el último se espera que no sea cacheado.

Resultado de las peticiones:

Analizado de nuevo el log del proxy se puede decir que los siguientes son los resultados a las peticiones de objetos web realizados:

URL EJECUTADA	Resultado de la petición en el fichero "store.log" del servidor proxy padre
URL 1	1218234564.008 **SWAPOUT** 00 00000000 AA215A510FB304A93986DAFBBFA14077 200 1218238168 1196726492 -1 image/jpeg **265436/265436** GET http://www.iad.upv.es/mapa2.jpg
URL 2	1218234697.850 **SWAPOUT** 00 00000001 0E5AC4753FA3339EE971D2F89F56BE92 200 1218238297 -1 -1 text/plain **-1/265436** GET http://127.0.0.1:8080/lasletras/miServlet?URL=11111111111111111111111111111111&Size=265436
URL 3	1218234770.283 **RELEASE** 00 00000002 7488C69DAA12F8BEF05F85FB3BCADEAA 200 1218238307 -1 -1 text/plain -1/**5000000** GET http://127.0.0.1:8080/lasletras/miServlet?URL=22222222222222222222222222222222&Size=5000000

Tabla 3 Peticiones http de prueba contra el servidor proxy de referencia de la arquitectura

Con el último experimento realizado se ha podido comprobar que el servidor proxy de referencia funciona correctamente, ha cacheado las dos primeras peticiones e ignorado en cuanto a cache se refiere la tercera petición de un objeto web.

Conclusiones:

- Se ha inferido que el servidor proxy de referencia es capaz de procesar peticiones http, a través del puerto del local host seleccionado.
- Se ha inferido que el servidor proxy de referencia puede procesar peticiones http del servidor de objetos web de la arquitectura y cachearlas y rechazarlas según sea el caso.
- Los scripts de arranque funcionan y proveen una forma sencilla de lanzar nuevas simulaciones contra este elemento de la arquitectura.

1.2.7 Lanzar Simulaciones contra la arquitectura propuesta

La sexta opción de menú del Shell script "ArrancadorSimulaciones.sh", lanzará finalmente las simulaciones contra la arquitectura propuesta, (será necesario tener funcionando las opciones 1,3,4; Servidor Tomcat preparado para recibir peticiones, Proxy Squid Padre funcionando, Proxy Squid Hijo encendido) una vez se tiene todo funcionando y las caches web vacías se puede lanzar la simulación utilizando el fichero de simulación ("Simulacion.txt") inferido en la sección "1.1.3.7 Unir ficheros modo columna".

Este subprograma ha sido realizado con Shell script y utiliza el comando wget con pudo ser visto en la sección del documento "**¡Error! No se encuentra el origen de la referencia. ¡Error! No se encuentra el origen de la referencia.**", la siguiente es una pantalla de ejemplo de una simulación contra la arquitectura propuesta:

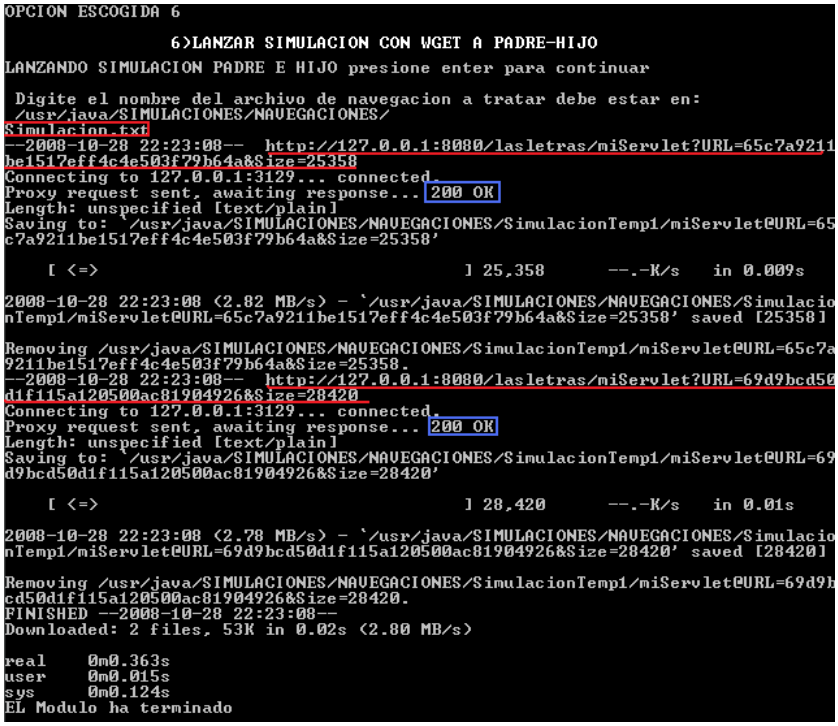

Ilustración 30 Ejemplo de simulación contra la jerarquía de proxies

Resaltado en rojo se observa que la simulación se va a realizar a partir del fichero "Simulacion.txt", que para el ejemplo, solo tendrá las siguientes peticiones http:

```
http://127.0.0.1:8080/lasletras/miServlet?URL=65c7a9211be1517eff4c4e503f79b64a&Size=25358
http://127.0.0.1:8080/lasletras/miServlet?URL=69d9bcd50d1f115a120500ac81904926&Size=28420
```

Las dos peticiones ejecutadas han sido correctamente descargadas por el cliente http de la arquitectura, tienen un HTTP estatus de 200, que indica que el objeto web ha sido descargado correctamente. Esta pequeña simulación habrá dejado trazas susceptibles de análisis profundos, los límites de las simulaciones en este punto solo están en las posibilidades de configuración del proxies squid.

1.2.8 Lanzar Simulaciones contra el proxy de referencia

Esta es la última opción del Shell script "ArrancadorSimulaciones.sh" permite lanzar una simulación contra el proxy de referencia, todos los resultados obtenidos serán contrastados contra los conseguidos en las simulaciones realizadas contra la jerarquía de proxies, esto permitirá creación de conocimiento, el gran objetivo a conseguir en la evaluación de la arquitectura propuesta.

Este subprograma es un Shell script que al igual que el anterior utiliza el cliente http wget para realizar las navegaciones. Como prerrequisito para una simulación será necesario tener arrancado el servidor Tomcat y el proxy de referencia La siguiente es la pantalla ejemplo de una simulación realizada contra el proxy de referencia:

```
OPCION ESCOGIDA 7

                7>LANZAR SIMULACION CON WGET A REFERENCIA

chmod: changing permissions of `wget22.sh': Permission denied
LANZANDO SIMULACION NORMAL UN SOLO PROXY

 Digite el nombre del archivo de navegacion a tratar debe estar en:
 /usr/java/SIMULACIONES/NAVEGACIONES/
Simulacion.txt
--2008-10-28 23:35:15--  http://127.0.0.1:8080/lasletras/miServlet?URL=65c7a9211
be1517eff4c4e503f79b64a&Size=25358
Connecting to 127.0.0.1:3127... connected.
Proxy request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: `/usr/java/SIMULACIONES/NAVEGACIONES/SimulacionTemp1/miServlet@URL=65
c7a9211be1517eff4c4e503f79b64a&Size=25358'

    [ <=>                                   ] 25,358      --.-K/s   in 0.004s

2008-10-28 23:35:15 (5.75 MB/s) - `/usr/java/SIMULACIONES/NAVEGACIONES/Simulacio
nTemp1/miServlet@URL=65c7a9211be1517eff4c4e503f79b64a&Size=25358' saved [25358]

Removing /usr/java/SIMULACIONES/NAVEGACIONES/SimulacionTemp1/miServlet@URL=65c7a
9211be1517eff4c4e503f79b64a&Size=25358.
--2008-10-28 23:35:15--  http://127.0.0.1:8080/lasletras/miServlet?URL=69d9bcd50
d1f115a120500ac81904926&Size=28420
Connecting to 127.0.0.1:3127... connected.
Proxy request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: `/usr/java/SIMULACIONES/NAVEGACIONES/SimulacionTemp1/miServlet@URL=69
d9bcd50d1f115a120500ac81904926&Size=28420'

    [ <=>                                   ] 28,420      --.-K/s   in 0.004s

2008-10-28 23:35:15 (7.30 MB/s) - `/usr/java/SIMULACIONES/NAVEGACIONES/Simulacio
nTemp1/miServlet@URL=69d9bcd50d1f115a120500ac81904926&Size=28420' saved [28420]

Removing /usr/java/SIMULACIONES/NAVEGACIONES/SimulacionTemp1/miServlet@URL=69d9b
cd50d1f115a120500ac81904926&Size=28420.
FINISHED --2008-10-28 23:35:15--
Downloaded: 2 files, 53K in 0.008s (6.48 MB/s)

real    0m0.296s
user    0m0.031s
sys     0m0.124s
EL Modulo ha terminado
```

Ilustración 31 Ejemplo de simulación contra el proxy de referencia

Como se pudo apreciar en la ilustración anterior, se han requerido dos objetos web al servidor Tomcat a través del servidor proxy de referencia, los cuales han sido servidos correctamente, dado que como se puede ver en azul su HTTP estatus es de 200 ok

En este punto ya se sabe como lanzar simulaciones contra la arquitectura, ahora el siguiente reto consiste en saber cómo analizar los resultados obtenidos en cada una de las simulaciones realizadas.

1.3 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 [14], este script ha sido modificado para una lectura más rápida de los resultados relevantes.

1.3.1 Script "checkAccessLog.sh"

Este script llama al programa al script "scalar.awk", pero se encarga de filtrar la información relevante [15] volcando los resultados en un fichero que se llamará "shortAnalysisResult.xls".

Para este documento solo va a ser necesario analizar el fichero de log del proxies llamado "Access.log", a continuación se va a poner un ejemplo del resultado obtenido para el análisis de una de las simulaciones realizadas:

TCP_MEM_HIT	TCP_MEM_HIT MB	TCP_HIT	TCP_HIT MB	Total Hits =	TOTAL MISSES	TOTAL HITS VOLUM	Saved Traffic MB:	Saved Traffic %
117.727K	80.066 MB	79.995K	82.543 MB	197.722K	221.503K	47.16 %	162.608 MB	4.87

Ilustración 32 Ejemplo del análisis del fichero "Access.log" de los proxies de la arquitectura

Como se puede ver en el análisis de resultados en el fichero .xls, se registran los valores de éxitos, tráfico salvado y peticiones realizadas en el proxy durante el tiempo del "Access.log". Toda esta información será bastante útil para la evaluación de la arquitectura propuesta planteada en este documento.

1.3.2 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.

---

[14] 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.

[15] De momento 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.

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

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 4 Unidades utilizadas en los análisis de resultados