Universidad Carlos III de Madrid

Escuela Politécnica Superior

TESIS DOCTORAL

Propuestas arquitectónicas para servidores Web distribuidos con réplicas parciales

Junio de 2005

Autor: José Daniel García Sánchez

Directores: Jesús Carretero Pérez
Félix García Carballeira

Tesis Doctoral

Propuestas arquitectónicas para servidores Web distribuidos con réplicas parciales

Autor: José Daniel García Sánchez

Directores: Jesús Carretero Pérez
Félix García Carballeira

Presidente	D.	.....
Vocal	D.	.....
Vocal	D.	.....
Vocal	D.	.....
Secretario	D.	.....

Realizado el acto de defensa y la lectura de tesis en ..... el día ..... de ..... del año 2005.

Calificación: .....

El Presidente

El Secretario

Los Vocales

A mi esposa, ***Flor****,*
sin cuyo apoyo nunca
habría comenzado el doctorado.

A mis hijos, ***Carlos*** *y* ***María****.*
Por el tiempo que esta tesis les robó.

Resumen

En esta tesis se propone una arquitectura distribuida de servidor Web, basada en *switch* distribuido y replicación parcial de contenidos, que permite alcanzar una alta escalabilidad en cuanto a los volúmenes de datos manipulados sin que se produzca un deterioro en la fiabilidad del sistema resultante, haciendo posible además que la asignación de contenidos se adapte de forma dinámica a las necesidades del servicio.

Más concretamente, las propuestas realizadas en esta tesis incluyen:

- Una familia novedosa de soluciones arquitectónicas de *cluster Web con switch distribuido* que satisface los objetivos de replicación parcial y distribución dinámica de réplicas, mitigando las debilidades relativas a la fiabilidad.

- Un algoritmo de asignación de réplicas que hace que los elementos con mayor frecuencia de acceso sean replicados en más nodos servidores que los elementos con baja frecuencia de acceso.

- Una estrategia de replicación dinámica de contenidos que permite determinar cuando es necesario realizar la redistribución de contenidos y cómo debe realizarse dicha redistribución.

- La adaptación de tres políticas de asignación de peticiones al caso de replicación parcial de contenidos: la política estática circular, la política de asignación al nodo menos cargado y la política de distribución de peticiones dependiente de la localidad de referencias (LARD).

Las evaluaciones realizadas han demostrado que la fiabilidad de un *sistema Web basado en cluster* está limitada por la fiabilidad de su *switch* Web. Así mismo en esta tesis se demuestra que un sistema basado en replicación parcial que utilice un número relativamente bajo de réplicas ofrece una fiabilidad equivalente a la de un sistema totalmente replicado, aunque ofrece una capacidad de almacenamiento mucho mayor. Además, la replicación parcial no afecta negativamente al rendimiento global del sistema.

Abstract

In this thesis a new Web server distributed architecture is proposed. The proposed architecture is based on the usage of a distributed switch and the partial replication of contents, in such a way that a high scalability can be achieved regarding managed data volume and without a reliability loss of the resulting system. Besides, content allocation may be adapted to service needs.

The proposals presented in this thesis include:

- A new family of architecture solutions based on a Web cluster with distributed switch which satisfies the goals of partial replication and dynamic replica distribution reducing the reliability weaknesses.
- A replica allocation algorithm making the highly accessed elements to be replicated in more server nodes than the lowly accessed elements.
- A dynamic content replication strategy which allows to determine when content redistribution is needed and how this redistribution must be performed.
- The adaptation of three request dispatching policies to the case of partial content replication: round robin dispatching, less loaded node dispatching and locality aware request distribution dispatching (LARD).

Evaluations have proved that the reliability of a *cluster based Web system* is limited by its *Web switch* reliability. In the same way, this thesis shows that a partial replication based system using a relatively low number of replicas offers an equivalent reliability to that of a system based on full replication, while its storage capacity is much higher. Besides, partial replication does not affect in a negative way to the global system performance.

Agradecimientos

He de dar gracias a muchas personas por haber llegado hasta aquí, pero si a alguien debo en especial haber llegado a la finalización de esta tesis, es a mi esposa Flor. De ella recibí el más firme apoyo cuando decidí cambiar la carrera profesional de consultor por la carrera académica. Sin su comprensión y sus continuas palabras de ánimo no habría podido dedicarme a la tesis con la necesaria intensidad.

Sin los esfuerzos que hicieron mis padres para que estudiase una carrera universitaria, nunca habría llegado al punto en que me encuentro. Siempre me hicieron ver la importancia de una buena formación y nunca escatimaron esfuerzos de ningún tipo para ayudarme en todo lo que necesité.

Muchas personas han intervenido e mi formación académica a lo largo de los años. Este es el momento para tener un recuerdo para todos aquellos que me enseñaron algo. En especial, este es un momento en el que no puedo olvidar a Ángel Fernández, mi profesor de Matemáticas de E.G.B. de quien tan buenos recuerdos guardo y que me hizo disfrutar tanto con el aprendizaje de las Matemáticas. Así mismo, solamente puedo tener palabras de agradecimiento para todos y cada uno de los excelentes profesores que me dieron clase en la Facultad de Informática de la Universidad Politécnica de Madrid. Mi formación básica en Informática se la debo a ellos. Igualmente, he de agradecer a mis ex-compañeros de la Universidad Pontificia de Salamanca en Madrid el haberme hecho ver la informática desde un punto de vista muy práctico y conectado con la realidad empresarial. Especialmente debo a ellos mi formación en tecnologías orientadas a objetos que tan útil me fue durante mi actividad profesional en distintas organizaciones.

A mi formación han contribuido también distintas personas de las diversas empresas en las que he trabajado, me gustaría recordar especialmente a algunos de ellos como David Alarcón de la Asociación para la Investigación en Tecnología de Equipos Mineros (AITEMIN), Jesús Fernández Revelles, con el que coincidí en Fomento de Construcciones y Contratas y DMR Consulting, Ismael Gómez de Fomento de Construcciones y Contratas, Victor Merino de TOOL, Peter Kiesel y Doris Thyroff de Siemens Medizin Technik y José Daniel Barrado, David Brogeras, Dave Smith y Christian De Neef de DMR Consulting.

Con respecto a los trabajos específicos que me han llevado a la realización de esta tesis me gustaría dar las gracias de forma especial a mis directores, Jesús Carretero y Félix García, por su constante apoyo y motivación. Siempre que he necesitado su ayuda y consejo han tenido tiempo que dedicarme.

Así mismo, he de agradecer las ideas y ayuda del resto de mis compañeros del Grupo de Arquitectura de Computadores, Comunicaciones y Sistemas (ARCOS) de la Universidad Carlos III de Madrid. Todos (Alex, Chema, David, Javi, Luismi, Rosa, Sole, ...) me han ayu-

dado de alguna manera a lo largo de estos años. En especial he de expresar mi agradecimiento a José María Pérez Menor, que me ayudó con sus conocimientos sobre los mecanismos internos de las distintas versiones para servidor del sistema operativo Microsoft Windows y con sus siempre constructivas críticas. Mis compañeros del campus de Colmenarejo del Grupo de Inteligencia Artificial Aplicada (GIAA) me han proporcionado información muy valiosa sobre las técnicas a utilizar para realizar simulaciones de eventos discretos y el análisis de los resultados.

Por último, este trabajo nunca habría sido posible sin otros trabajos previos realizados por distintos autores. Es por ello que me gustaría extender mis agradecimientos a cada uno de los autores que aparecen citados en la bibliografía de esta tesis.

Colmenarejo, junio de 2005.

Parte de los trabajos de esta tesis han sido financiados por la *Comunidad Autónoma de Madrid* y el *Fondo Europeo de Desarrollo Regional de la Unión Europea* mediante el proyecto **Técnicas de aumento de prestaciones en clusters de servidores *Web* distribuidos y cooperativos (07T/0010/2003 1)** y por el *Ministerio de Educación y Ciencia* mediante el proyecto **Almacenamiento de altas prestaciones en entornos *Grid* (TIN2004-02156)**.

Índice general

Índice de figuras

Índice de Tablas

Índice de Algoritmos

Capítulo 1

Introducción

En este capítulo se realiza una introducción al trabajo realizado en esta tesis. Primeramente, se presenta una motivación del problema que se trata en el trabajo. Posteriormente, se plantean los objetivos que se pretende alcanzar. Finalmente, se describe la organización del resto del documento.

1.1 Motivación

Desde hace pocos años, Internet ha cambiado la manera en que los individuos trabajan y se relacionan entre sí. La comunicación entre individuos, compañías e instituciones es ahora fácil e instantánea a nivel mundial y, de igual manera, el acceso a una enorme cantidad de información es tan variada como el quehacer humano. La demanda de servicios basados en Web sigue creciendo y, como consecuencia de ello, los servidores Web se enfrentan cada vez a una mayor carga. Los problemas de rendimiento de las arquitecturas de servidores Web serán incluso mayores en el futuro debido al crecimiento en el número de usuarios que acceden a los servicios y al incremento en los volúmenes de información que las organizaciones necesitan poner a disposición de sus clientes.

El creciente desarrollo de las aplicaciones basadas en Web ha creado un enorme interés en el diseño y la construcción de software y hardware para este tipo de sistemas que permita satisfacer las necesidades de rendimiento a medida que se incrementa el número de usuarios que accede a los sistemas. Dado que la escalabilidad es una de las principales características requeridas para este tipo de sistemas, la construcción de servidores Web distribuidos se aprecia como una de las soluciones más efectivas para satisfacer esta necesidad.

Debido a la complejidad de la infraestructura de Internet, los problemas de rendimiento pueden aparecer en muchos puntos durante la interacción entre un cliente y un sistema basado en Web. Por ejemplo, podrían aparecer en la red debido a una congestión en los *routers*, o bien, podrían aparecer en el servidor de bases de datos por falta de capacidad.

No obstante, al tratarse Internet de una red sin administración centralizada, el sitio Web es el único componente que se encuentra bajo control directo del proveedor de contenidos. El resto de componentes escapan a su control. Es por esto que las únicas estrategias aplicables a corto plazo son aquellas que no requieren actuación sobre el resto de componentes (desde los protocolos utilizados, pasando por los servidores proxy hasta las clientes Web).

Por otra parte se estima que la contribución de los servidores Web a la latencia total de una transacción Web supera el 40 % [1]. Existen, al menos, dos razones que pueden hacer que este porcentaje se incremente en el futuro: la reducción de la latencia de red y el aumento de tamaño de los contenidos.

En primer lugar, el ancho de banda de las redes es cada vez mayor en comparación con la capacidad de los servidores. En 1997 Gilder [2] predijo un incremento anual en el ancho de banda superior al 100 %. Esta predicción parece confirmarse según algunos trabajos [3, 4, 5, 6].

Además, los sitios Web han ido evolucionando a lo largo del tiempo. Durante este proceso evolutivo se ha pasado de sitios que contenían principalmente texto a sitios que ofrecen contenidos en diversos formatos incluyendo imágenes, sonidos y vídeos. Esto hace que los contenidos que se sirven tengan cada vez un mayor tamaño, lo que hace que cada vez sea necesario más tiempo para recuperar dichos contenido de disco antes de enviarlos por la red.

Los servidores Web distribuidos ya sean geográficamente distribuidos [7] o localmente distribuidos [8] se presentan como una solución escalable al problema de aumento de prestaciones de servicios basados en Web.

Existen diversas soluciones que permiten la construcción de servidores Web distribuidos. Estas soluciones se basan, bien en la replicación total de contenidos (todos los contenidos se replican en todos los nodos servidores), bien en la distribución total de contenidos (todos los contenidos se distribuyen de forma que cada elemento se encuentra en un único nodo servidor).

Si todos los contenidos se encuentran replicados en todos los nodos servidores (lo que suele ser la alternativa más habitual), se dispone de sistemas altamente tolerantes a fallos y muy fiables. No obstante, este tipo de sistemas presenta limitaciones cuando el volumen de información que se desea publicar crece. De hecho, la adición de nuevos nodos servidores no permite escalar la capacidad de almacenamiento.

Si los contenidos se encuentran totalmente distribuidos, de forma que cada archivo se encuentre alojado en un único nodo servidor, el sistema resultante presenta una tolerancia a fallos mucho menor, así como una menor fiabilidad. Por otra parte, como cada archivo solamente tiene que almacenarse en un único nodo, la capacidad de almacenamiento conseguida con los mismos recursos es mucho mayor. Y lo que es más importante, la capacidad de almacenamiento es fácilmente escalable y se incrementa al incorporar nuevos nodos servidores al sistema.

Entre estas dos alternativas (la replicación total de contenidos y la distribución total), se puede situar una tercera, la replicación parcial, consistente en almacenar un cierto número de copias de cada elemento. Esta tercera alternativa constituye una parte fundamental de las propuestas presentadas en esta tesis.

1.2 Objetivos

A la vista de las necesidades de incremento de volumen de almacenamiento de forma que se mantenga un nivel aceptable de fiabilidad, en esta tesis se plantea como objetivo principal *el diseño de una arquitectura de servidor Web que se base en una replicación parcial de contenidos de forma que se pueda alcanzar una alta escalabilidad en cuanto a los volúmenes de datos manipulados sin que se produzca un deterioro en la fiabilidad del sistema resultante y permitiendo que la asignación de contenidos se adapte de forma dinámica a las necesidades*

del servicio.

La tesis estudiará las arquitecturas existentes en la actualidad y sus políticas de encaminamiento asociadas, con el objeto de determinar las posibles soluciones que permitan basarse en la replicación parcial de contenidos y la adaptación dinámica de los mismos a las necesidades del servicio.

Objetivos más específicos de esta tesis son:

- Estudiar las arquitecturas que son susceptibles de ser adaptadas a la situación de replicación parcial de contenidos.
- Realizar una propuesta de arquitecturas especialmente adaptadas para la replicación parcial de contenidos y la adaptación dinámica de los mismos.
- Estudiar el problema de la asignación de réplicas a los nodos servidores.
- Proponer un algoritmo de asignación de réplicas a nodos servidores basado en las probabilidades de acceso a los contenidos.
- Proponer una política de asignación de peticiones a los nodos servidores que tiene en cuenta que los contenidos se encuentran parcialmente replicados.

La tesis se ocupa del servicio de las peticiones estáticas, no incluyéndose el tratamiento de las peticiones dinámicas. Las peticiones estáticas son aquellas que no requieren un procesamiento especial por parte del servidor para construir la respuesta, sino que devuelven el contenido de un archivo que reside en dicho servidor. El tratamiento de las peticiones dinámicas se ha dejado como un trabajo futuro.

1.3 Organización del documento

El resto de este documento se ha organizado de la siguiente forma:

- El capítulo 2 presenta las arquitecturas de servidores Web distribuidos existentes en la actualidad y describe sus técnicas de encaminamiento asociadas.
- El capítulo 3 estudia las arquitecturas susceptibles de ser adaptadas a la situación de replicación parcial de contenidos y presenta las propuestas arquitectónicas realizadas.
- El capítulo 4 realiza una propuesta de las políticas de asignación de réplicas a nodos servidores y de asignación de peticiones.
- El capítulo 5 presenta los resultados de las evaluaciones realizadas.
- El capítulo 6 presenta las conclusiones del trabajo.
- Al final del documento se presentan las referencias bibliográficas utilizadas para la realización de esta tesis.

Capítulo 2

Estado de la cuestión

En este capítulo se presentan las arquitecturas de servidores Web distribuidos actualmente existentes y se describen sus técnicas de encaminamiento asociadas. Las arquitecturas de servidores Web distribuidos han aparecido en los últimos años como alternativa a otros mecanismos de escalado para servidores Web con el objetivo de resolver limitaciones derivadas del uso de un único computador. Cada una de las arquitecturas presentadas lleva asociados unos mecanismos específicos de encaminamiento de peticiones que permiten traspasar las peticiones de un componente a otro del servidor Web distribuido. Dichos mecanismos de encaminamiento también se estudian en este capítulo.

Por último, en este capítulo también se presentan las políticas existentes para la asignación de peticiones, que son las responsables de decidir qué elemento de un servidor Web distribuido es responsable de dar servicio a una petición determinada.

En la sección 2.1 se presenta las técnicas generales para escalar las prestaciones de un servidor Web. En la sección 2.2 se presentan las distintas arquitecturas distribuidas existentes en la actualidad, así como los mecanismos de encaminamiento de peticiones disponibles en cada una de las arquitecturas. En la sección 2.3 se realiza una presentación general de los distintos tipos de políticas de asignación de peticiones existentes. La sección 2.4 presenta las políticas de asignación de peticiones que no utilizan información de estado. La sección 2.5 presenta las políticas de asignación de peticiones que se basan exclusivamente en información del cliente que origina la petición. La sección 2.6 presenta las políticas de asignación de peticiones que se basan en información de estado del servidor. La sección 2.7 presenta políticas que son exclusivas de una arquitectura concreta (los sistemas Web distribuidos). Finalmente la sección 2.8 resume los contenidos del capítulo.

2.1 Escalabilidad de los servidores Web

Cuando es necesario mejorar el rendimiento de un servidor Web las dos primeras alternativas que se plantean son el escalado hardware y el escalado software. Ambas alternativas cumplen con la restricción de mantener el servidor Web alojado en un único nodo.

El escalado hardware consiste en la mejora de los recursos físicos del servidor. Esto puede conseguirse bien migrando el sistema a una nueva máquina con mayores prestaciones, o bien, añadiendo más recursos a la máquina, como podría ser la adición de más procesadores, discos o memoria. Esta solución suele ayudar a resolver problemas puntuales a corto plazo, pero

puesto que el número de recursos que se pueden alojar en una máquina es limitado, no se puede considerar su aplicación continuada a lo largo del tiempo.

El escalado software consiste en la mejora de los componentes software del servidor [8], bien sean estas mejoras, del sistema operativo o del propio servidor Web.

Una alternativa en este sentido es la mejora de un sistema operativo concreto como UNIX [9]. En [10] se propone la adición de nuevas características al sistema operativo como los contenedores de recursos (que permiten controlar la utilización de los recursos del núcleo por parte de las distintas tareas de un servidor) y en [11] se propone la mejora de las llamadas al sistema para el soporte de eventos (dividiendo la llamada al sistema `select()` en dos llamadas distintas).

Otra propuesta, es la modificación del sistema operativo [12, 13] de forma que se permita el envío directo de datos desde la caché del sistema de archivos a la red. Esta mejora se encuentra disponible en la Windows NT [14] y sus sucesores mediante la función `TransmitFile()`[15].

El sistema operativo también se puede mejorar introduciendo modificaciones que proporcionen un mecanismo unificado de gestión de búferes y cachés como IO-Lite [16]. El objetivo de dicho mecanismo es reducir las copias de datos mediante la unificación de búferes y cachés de los distintos subsistemas de entrada/salida.

Otra opción es la mejora del servidor Web. Se han propuesto diversas medidas [12, 13] para la mejora del servidor Web Apache [17] como el uso de una caché en el espacio de usuario, la aceleración de la generación de los logs o el uso de una caché para la traducción de URL's a nombres de archivo. Otro servidor Web, Flash [18], intenta evitar el bloqueo de sus hilos y procesos. Para ello Flash tiene una arquitectura denominada *arquitectura multiproceso dirigida por eventos* que desacopla la entrada/salida a disco del procesamiento de peticiones.

2.2 Arquitecturas distribuidas

La mejora de un servidor Web no resuelve el problema de la escalabilidad. Otra posible solución es el uso de una arquitectura distribuida, en la que que un conjunto de nodos ofrecen el servicio. De esta forma la escalabilidad se obtiene añadiendo nuevos nodos al sistema [8]. En todos estos sistemas, es necesario distribuir la carga entre los distintos nodos.

La Figura 2.1 muestra una arquitectura general. El servidor Web está compuesto por un conjunto de nodos que pueden recibir peticiones y procesarlas. El conjunto de todos los nodos ofrece una imagen única, de forma que los clientes no son conscientes de los nombres o las localizaciones de los nodos que componen el sistema Web. Desde el punto de vista de los clientes, se accede al sitio Web mediante una única interfaz virtual o nombre de sitio.

El sitio Web recibe peticiones realizadas por los clientes. Se debe observar que cuando un usuario realiza una petición (por ejemplo pinchando en un hiperenlace), esta petición suele traducirse en varias peticiones al servidor (una petición para cada objeto que forma la página). Cuando se realiza una petición (por ejemplo: `http://www.uc3m.es/index.html`), el cliente extrae el nombre del dominio (`www.uc3m.es`) y resuelve el nombre de dominio para encontrar la correspondiente dirección IP. El cliente establece una conexión TCP con la dirección IP y la utiliza para enviar la petición y recibir la respuesta. En caso de tratarse la versión 1.0 del protocolo HTTP [19], se utiliza una conexión nueva para cada petición. Por el contrario si se trata de la versión 1.1 [20], se utiliza la misma conexión.

Las arquitecturas distribuidas existentes [8] se pueden clasificar en tres familias:

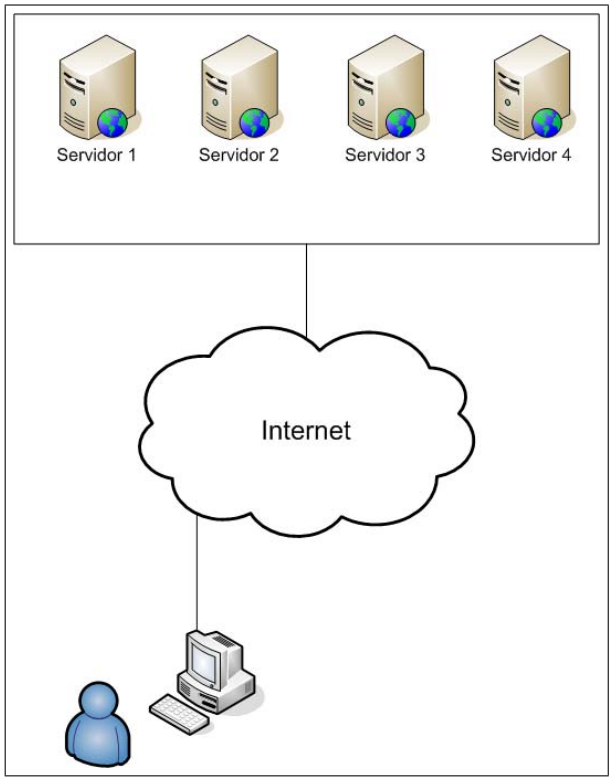

Figura 2.1: Arquitectura distribuida general de un servidor Web.

Sistemas Web basados en *cluster* La dirección IP de los nodos del *cluster* no es visible para los clientes. En lugar de esto, los clientes utilizan una dirección IP virtual que se corresponde con algún dispositivo de distribución situado entre los clientes y los servidores Web.

***Clusters* Web virtuales** La dirección IP de los nodos del *cluster* no es visible para los clientes. En su lugar, los clientes utilizan una dirección IP virtual que es compartida por todos los nodos servidores.

Sistemas Web distribuidos Cada nodo dispone de una dirección IP distinta que es pública y visible para todos los clientes. No existe en este caso ningún dispositivo intermedio, por lo que cualquier petición puede llegar a cualquier nodo.

Sea cual sea la arquitectura distribuida utilizada, es necesario realizar algún tipo de reparto de las peticiones. Esto requiere, por una parte, un mecanismo de asignación de peticiones, que decida qué nodo debe procesar cada petición. Por otra parte, es necesario un mecanismo de encaminamiento de peticiones que dirija la petición al nodo seleccionado.

2.2.1 Sistemas Web basados en *cluster*

Un *sistema Web basado en cluster* [21] o *cluster Web* está formado por un conjunto de nodos que se conoce públicamente a través de un único nombre y una dirección IP virtual. La

dirección IP virtual se corresponde con un único dispositivo frontal que actúa de interfaz entre Internet y los nodos servidores. Este elemento frontal se suele denominar *switch* Web. El *switch* Web recibe todas las peticiones dirigidas a la dirección IP virtual y las encamina a algún nodo servidor. La Figura 2.2 muestra la arquitectura general de un sistema Web basado en *cluster*.

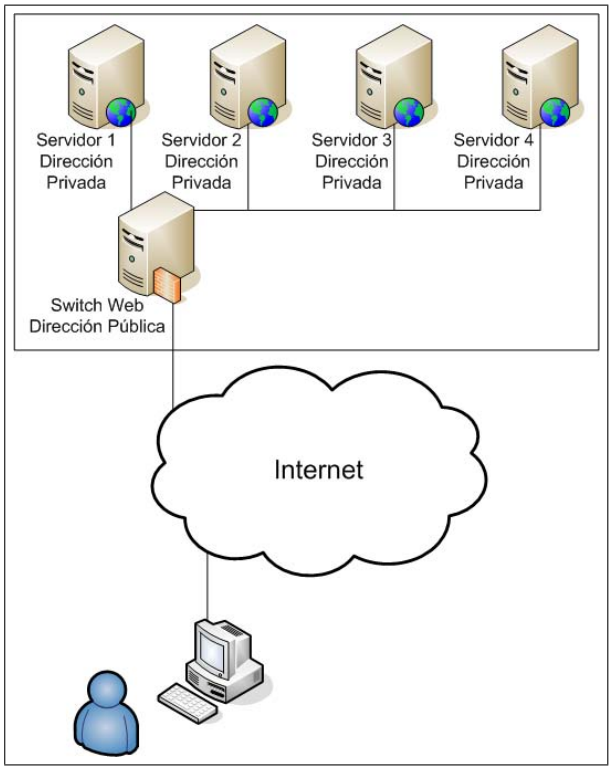

Figura 2.2: Arquitectura general de un sistema Web basado en *cluster*.

En un *cluster* Web el *switch* Web es el elemento que define el comportamiento del sistema. Dependiendo del modo en que el *switch* analiza y procesa las peticiones se tienen dos grupos de encaminamiento de peticiones.

El *encaminamiento independiente del contenido* se produce cuando el *switch* analiza las peticiones al nivel TCP (lo que equivale al nivel 4 del modelo OSI [22]). Este tipo de encaminamiento se realiza a un nivel de protocolo en el que no se pueden analizar las peticiones HTTP lo que tiene como ventaja un mayor nivel de eficiencia, pero con el coste de no poder tomar decisiones en función de la petición realizada.

El *encaminamiento dependiente del contenido* se produce cuando el *switch* analiza la petición HTTP antes de realizar el encaminamiento. Esto requiere que el *switch* establezca una conexión TCP completa con el cliente y examine el contenido completo de la petición HTTP antes de decidir el nodo al que se asigna la petición. Este encaminamiento es menos eficiente que el anterior, pero permite la aplicación de políticas mas complejas que tengan en cuenta los atributos de cada petición individual.

Independientemente del tipo de encaminamiento utilizado, los sistemas Web basados en *cluster* se pueden clasificar atendiendo al camino que sigue el flujo de respuesta desde el nodo

que sirve la petición hasta el cliente. Dependiendo del camino seguido se tienen arquitecturas unidireccionales o arquitecturas bidireccionales.

Las arquitecturas de flujo bidireccional son aquellas en que las respuestas del nodo servidor vuelven a pasar por el *switch* Web, que es el encargado de transmitir las respuestas a los clientes. Este tipo de arquitecturas son más sencillas pero menos eficientes, ya que el *switch* se tiene que encargar tanto de los paquetes entrantes como de los salientes.

En las arquitecturas de flujo unidireccional, los nodos que sirven las peticiones envían las respuestas directamente a los clientes. En este caso el *switch* Web solamente tiene que procesar las peticiones (pero no sus respuestas) con lo que se consigue un mayor grado de eficiencia. No obstante, estas arquitecturas requieren que los nodos dispongan de una conexión independiente para los paquetes de salida.

La tabla 2.1 ofrece una clasificación de las principales soluciones atendiendo al tipo de encaminamiento y los caminos seguidos por las respuestas.

	Encaminamiento	
	Independiente del contenido	**Dependiente del contenido**
Flujo bidireccional	Doble reescritura	Pasarela TCP Empalme TCP
Flujo unidireccional	Reescritura simple de paquete Encapsulado de paquetes Reenvío de paquetes	Cesión TCP

Tabla 2.1: Clasificación de arquitecturas para sistemas Web basados en *cluster*.

2.2.1.1 Doble Reescritura

En la doble reescritura de paquetes [23], la dirección privada de cada nodo es una dirección IP privada. El *switch* Web reescribe, tanto los paquetes entrantes como los paquetes salientes. Cuando un paquete llega desde el exterior al Web *switch*, éste cambia en la cabecera del paquete la dirección IP virtual (es decir, la dirección del *switch* Web) por la dirección del nodo seleccionado. Cuando un paquete llega al *switch* Web desde un nodo del *cluster*, éste cambia la dirección IP del nodo por la dirección IP virtual. La Figura 2.3 muestra el funcionamiento de un *switch* Web con doble reescritura.

2.2.1.2 Reescritura Simple

En la reescritura simple de paquetes [24], la dirección privada de cada nodo es también una dirección IP privada. En esta arquitectura, el *switch* Web realiza la reescritura de los paquetes entrantes. Cuando los nodos envían la respuesta, no lo hacen a través del *switch* Web. En vez de esto, el nodo reescribe los paquetes salientes sustituyendo su propia dirección IP por la dirección IP del *switch* Web. La Figura 2.4 muestra el funcionamiento de un *switch* Web con reescritura simple.

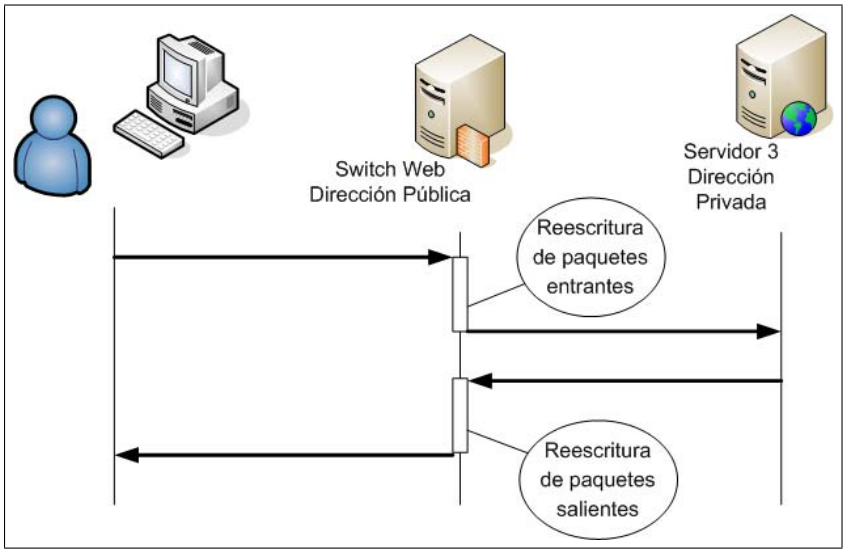

Figura 2.3: Flujo de paquetes en un *cluster* Web con doble reescritura.

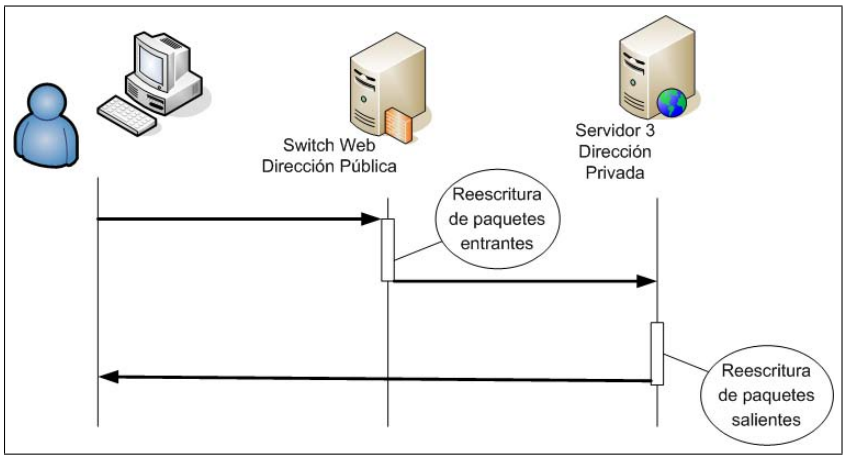

Figura 2.4: Flujo de paquetes en un *cluster* Web con reescritura simple.

2.2.1.3 Encapsulado de paquetes

El encapsulado de paquetes [25] crea por cada paquete recibido en el *switch* Web un nuevo paquete que tiene como información el paquete recibido. Cuando un nodo recibe un paquete encapsulado analiza la información del paquete interno y envía las respuestas directamente al cliente. La Figura 2.5 muestra el funcionamiento de un *switch* Web con encapsulado de paquetes.

2.2.1.4 Reenvío de paquetes

El reenvío de paquetes [26] se basa en el uso de direcciones privadas para los nodos al nivel MAC. Todos los nodos comparten con el *switch* Web la dirección IP virtual. Los paquetes llegan al *switch* Web que, los reenvía al nodo correspondiente reescribiendo la dirección MAC

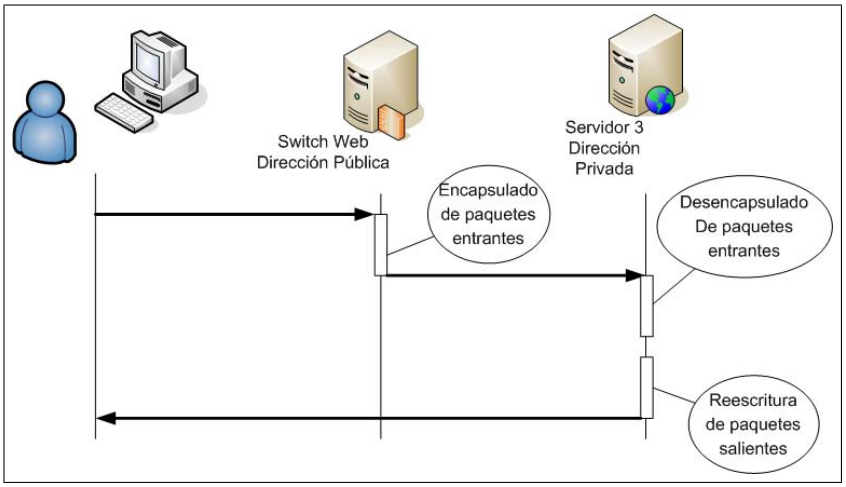

Figura 2.5: Flujo de paquetes en un *cluster* Web con encapsulado de paquetes.

en la dirección destino (nivel 2 del protocolo). Como todos los nodos tienen configurada la dirección IP virtual, reconocen los paquetes como suyos y pueden contestar directamente al cliente. La Figura 2.6 muestra el funcionamiento de un *switch* Web con reenvío de paquetes.

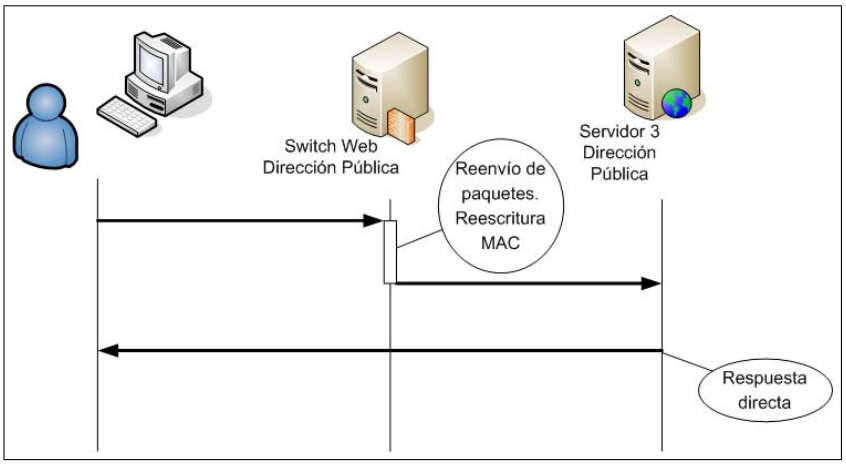

Figura 2.6: Flujo de paquetes en un *cluster* Web con reenvío de paquetes.

2.2.1.5 Pasarela TCP

El *switch* Web mantiene conexiones TCP persistentes con cada uno de los nodos servidores [27]. Cuando llega una petición HTTP al *switch*, éste retransmite la petición al nodo seleccionado. Cuando el nodo contesta al *switch* Web, éste también se encarga de retransmitir la respuesta al cliente. En esta solución toda la retransmisión se hace en la capa de aplicación (HTTP). La Figura 2.7 muestra el funcionamiento de un *switch* Web con pasarela TCP.

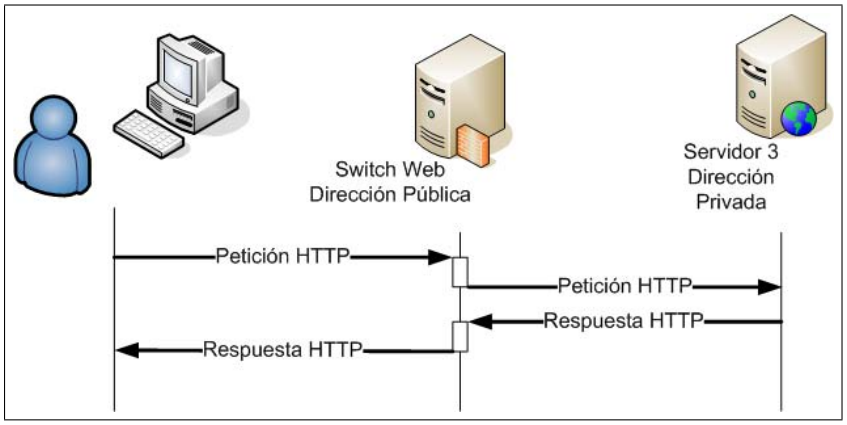

Figura 2.7: Flujo de paquetes en un *cluster* Web con pasarela TCP.

2.2.1.6 Empalme TCP

El empalme TCP supone una mejora sobre la pasarela TCP [28]. Para obtener una mejora de rendimiento, se evita que los datos tengan que llegar hasta el nivel de aplicación y se realiza su tratamiento en el nivel de red. Cuando se establece una conexión TCP entre un cliente y el *switch* Web, éste último selecciona un nodo servidor y trata todos los paquetes correspondientes a dicha conexión realizando sustituciones en las cabeceras IP y TCP. El empalme TCP se puede implementar mediante mecanismos hardware [29], la modificación del sistema operativo [30, 31, 32, 33, 34, 35, 36] o una biblioteca de funciones para sockets [37]. La Figura 2.8 muestra el funcionamiento de un *switch Web* con empalme TCP.

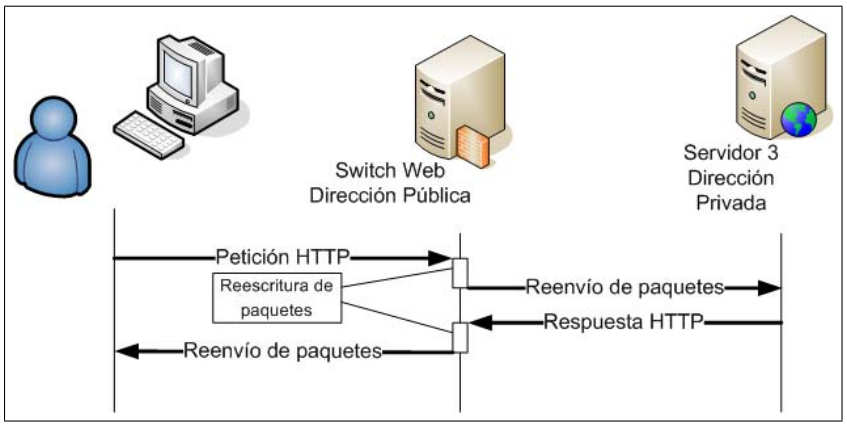

Figura 2.8: Flujo de paquetes en un *cluster* Web con empalme TCP.

2.2.1.7 Cesión de conexión TCP

El protocolo cesión de conexión TCP [38, 39, 40, 41, 42, 43] permite que el *switch* Web ceda a un nodo servidor la conexión con el cliente, de forma que el nodo servidor pueda contestar directamente. Cuando el *switch* Web recibe una petición, selecciona un nodo servidor y le envía la petición. Cuando llegan más paquetes al *switch* Web que se corresponden con la

misma conexión, éste simplemente reenvía los paquetes al nodo servidor responsable de esa conexión. Antes de enviar cualquier paquete a un nodo servidor, se reescriben los paquetes con la dirección del cliente. De esta forma, los nodos servidores pueden contestar directamente a los clientes, simplemente reescribiendo los paquetes con la dirección del *switch* Web. La Figura 2.9 muestra el funcionamiento de un *switch* Web con cesión de conexión TCP.

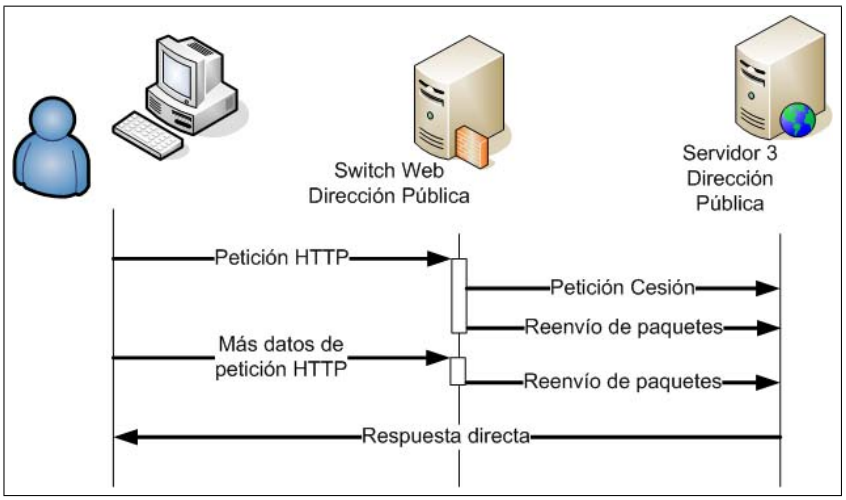

Figura 2.9: Flujo de paquetes en un *cluster* Web con cesión de conexión TCP.

2.2.2 *Clusters* Web virtuales

Un *cluster* Web virtual [44, 45, 46] está formado por un conjunto de nodos que tienen todos asignada la misma dirección IP. En este caso no existe ningún dispositivo central. Por el contrario, todas las peticiones llegan a todos los nodos del *cluster*. Cada nodo debe realizar un proceso de filtrado y decidir qué peticiones debe procesar y qué peticiones debe descartar. La Figura 2.10 muestra la arquitectura general de un *cluster* Web virtual.

Es necesario que un único nodo sirva cada petición. La solución más común suele ser que cada nodo evalúe una función *hash* sobre la dirección IP del cliente y su puerto para decidir si debe procesar o no la petición.

Para conseguir que todas las peticiones lleguen a todos los nodos de la red, se asigna la misma dirección IP a todos los nodos. Por tanto, el encaminamiento se realiza a nivel MAC.

Existen dos soluciones alternativas para la construcción de *clusters Web virtuales*: *unicast MAC* y *multicast MAC*.

Unicast MAC Todos los nodos se configuran con la misma dirección MAC. Esto implica que todos los nodos tienen que estar en la misma red y que no pueden comunicarse entre ellos.

Multicast MAC Todos los nodos se configuran con una misma dirección MAC para *multicast*, pero cada nodo tiene su propia dirección MAC *unicast* y su propia dirección IP. La dirección IP virtual se hace corresponder con la dirección MAC para *multicast*. Esta segunda alternativa sí permite la comunicación interna entre nodos.

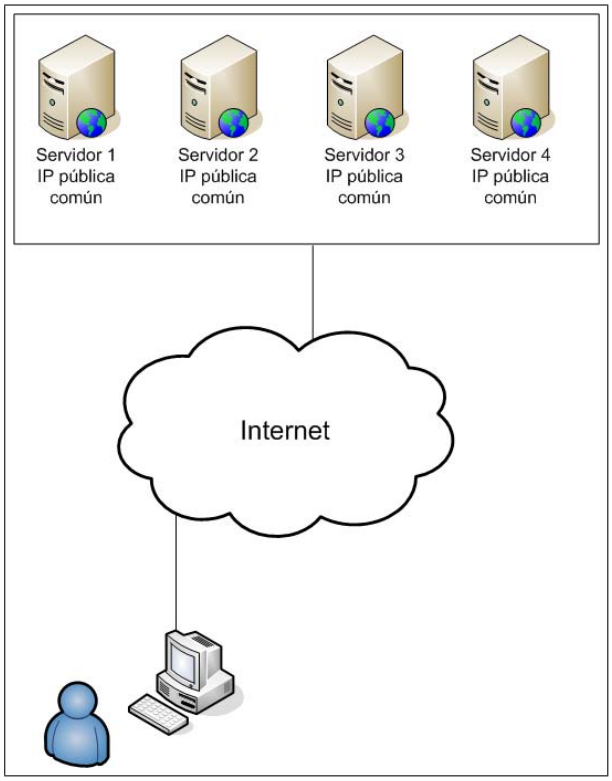

Figura 2.10: Arquitectura general de un *cluster* Web virtual.

2.2.3 Sistemas Web distribuidos

Un sistema Web distribuido está formado por un conjunto de nodos, en el que cada nodo tiene una dirección IP diferenciada y visible para los clientes. Cualquier petición puede llegar a cualquier nodo servidor. La Figura 2.11 muestra la arquitectura general de un sistema Web distribuido.

En general, el encaminamiento en un sistema Web distribuido se realiza mediante la resolución de nombre de dominio (DNS). Mediante DNS, se traduce un nombre de dominio en una dirección IP. Esto daría como resultado que siempre se encaminarían todas las peticiones al mismo nodo. Para evitar esto se puede utilizar la variante dinámica de DNS [47]. Este es un mecanismo muy simple que hace que cada vez que se realice una petición al servidor DNS este devuelva la dirección de algún nodo del sistema y un tiempo de validez para dicha asignación. La asignación dinámica puede ser cíclica [48, 49] o bien utilizar otra estrategia [50].

No obstante, la resolución dinámica de DNS es de una utilidad bastante limitada. Esto es debido al sistema de cachés de DNS existente en Internet. De hecho, diversas medidas [24, 51] muestran que el número de peticiones para los que se realiza una consulta al servidor de DNS autoritativo (el único sobre el que se puede tener control) es menor del 5 % del total.

Independientemente de la utilización o no de un mecanismo de encaminamiento mediante DNS dinámico, un sistema Web distribuido necesita un mecanismo adicional de encaminamiento que permita que un nodo pueda traspasar ciertas peticiones a otro nodo del sistema.

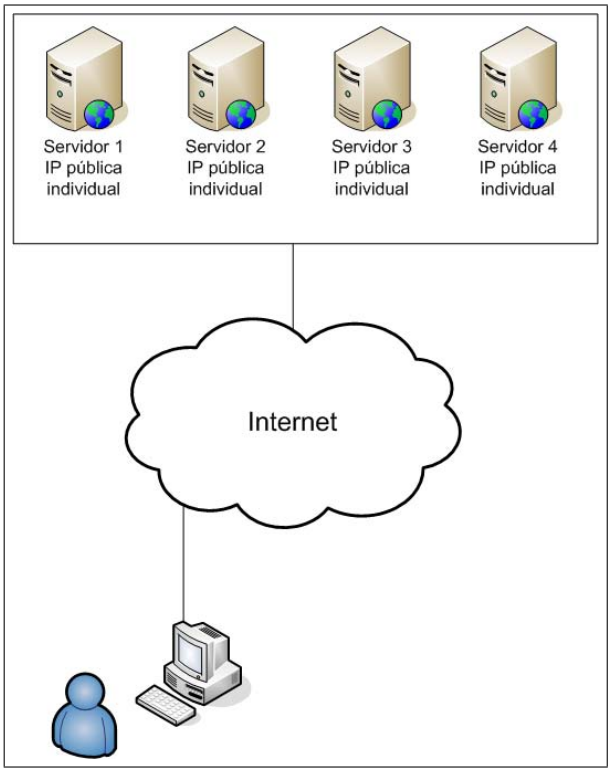

Figura 2.11: Arquitectura general de sistema Web distribuido.

El encaminamiento de peticiones se puede realizar con distintos mecanismos como la triangulación, la reescritura de URL o la redirección HTTP. Estos mecanismos se describen a continuación.

2.2.3.1 Triangulacion

La triangulación [48] sigue una idea bastante parecida al encapsulado de paquetes (véase encapsulado de paquetes en 2.2.1.3). Cuando un nodo decide redirigir la petición a otro nodo, encapsula sus paquetes y los reenvía al nodo seleccionado. Cuando llegan nuevos paquetes, éstos también se encapsulan y se reenvían al nuevo nodo. El nodo seleccionado contesta directamente al cliente, reescribiendo previamente los paquetes salientes. Obsérvese, que la triangulación puede implementarse en la capa TCP y ser por tanto independiente del contenido. La Figura 2.12 muestra el flujo de paquetes en un sistema Web con triangulación.

2.2.3.2 Reescritura de URL

La reescritura de URL [52, 53, 54] es especialmente útil cuando los contenidos son complejos y están compuestos de muchos elementos enlazados entre si. La idea básica consiste en hacer, que cuando se realiza una petición de un objeto (por ejemplo, una página Web) que contiene referencias a otros elementos, estas referencias se sustituyan dinámicamente por referencias a objetos en otros nodos. La Figura 2.13 muestra el flujo de peticiones en un sistema Web con reescritura de URL.

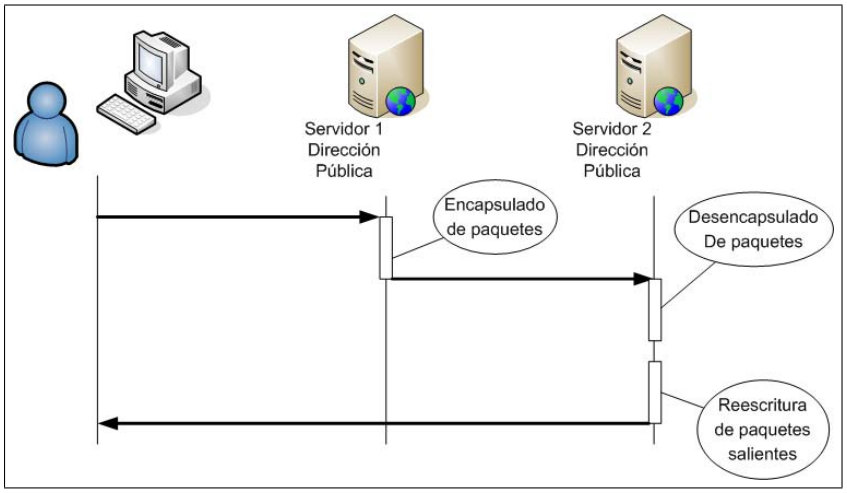

Figura 2.12: Flujo de paquetes en un sistema web distribuido con triangulación.

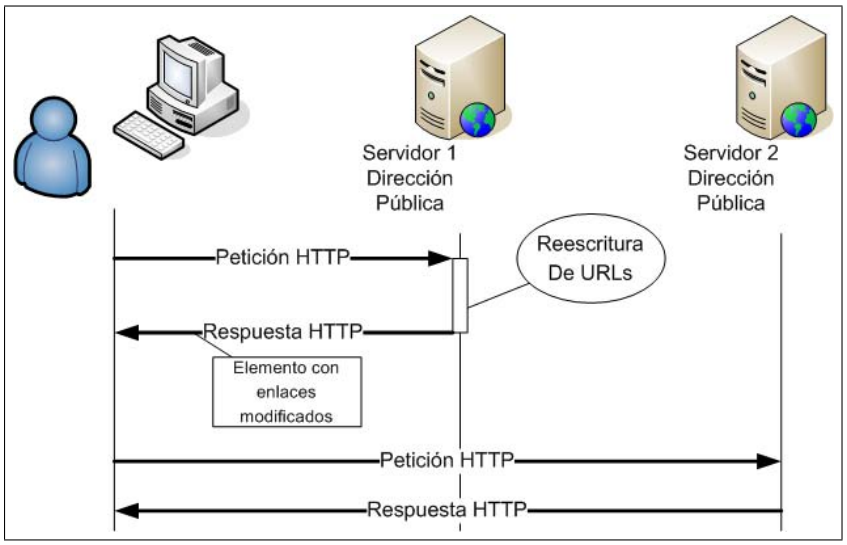

Figura 2.13: Flujo de paquetes en un sistema Web distribuido con reescritura de URL.

2.2.3.3 Redirección HTTP

La redirección HTTP es un mecanismo soportado por el protocolo HTTP (tanto en su versión 1.0 [19] como en su versión 1.1 [20]). Cuando un nodo decide redirigir una petición a otro nodo, contesta al cliente con un mensaje de redirección. Es responsabilidad del cliente enviar una nueva petición al nodo indicado. La Figura 2.14 muestra el flujo de peticiones en un sistema Web con redirección de peticiones HTTP.

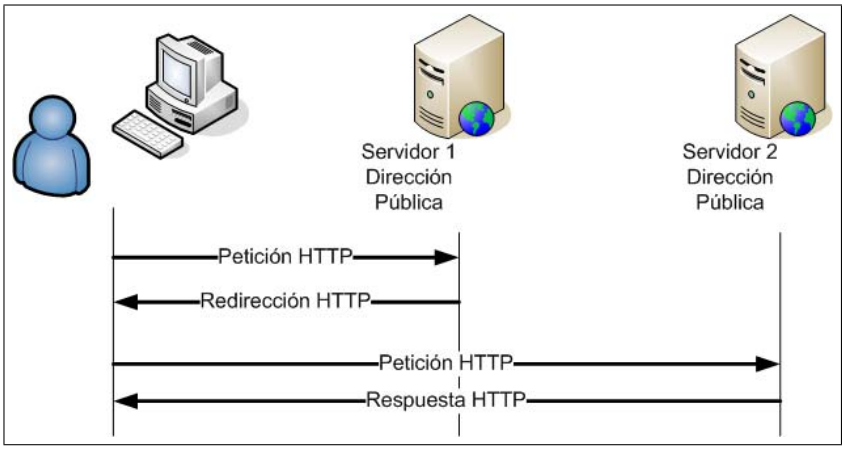

Figura 2.14: Flujo de paquetes en un sistema web distribuido con redirección HTTP.

2.3 Políticas de asignación de peticiones

Las políticas de asignación de peticiones tienen como misión decidir qué nodo debe procesar cada petición que llega al sistema. El resultado ideal de cualquier algoritmo de asignación de peticiones debería ser el equilibrio de carga (todos los nodos tienen la misma carga de trabajo). No obstante, en muchos casos, es suficiente con alcanzar el reparto de carga (se produce un reparto de peticiones entre los nodos) [55].

Cualquier algoritmo de asignación de peticiones debe cumplir los siguiente requisitos:

1. Debe ser de baja complejidad. Un retardo en el algoritmo deteriora el rendimiento del sistema.

2. El algoritmo debe ser computable haciendo uso exclusivamente de los estándares y protocolos actuales para Web.

3. La información de estado utilizada debe ser accesible para el nodo que evalúa el algoritmo.

El lugar de ejecución del algoritmo de asignación de peticiones varía dependiendo de la arquitectura del sistema.

En un sistema Web basado en *cluster* el lugar ideal para alojar el algoritmo de asignación de peticiones es el *switch* Web [8]. Esto se debe a que todas las peticiones pasan por el *switch* Web, que es el responsable de transferir la petición a algún nodo. Por tanto, es razonable que la selección del nodo al que se asigna la petición se realice en el mismo lugar donde se realiza la transferencia de dicha petición.

En un *cluster* Web virtual el algoritmo tiene que ser necesariamente de naturaleza distribuida, puesto que todas las peticiones llegan a todos los nodos. Es más, como la aceptación o rechazo de paquetes se hace en la capa MAC, no es posible utilizar la información de la petición HTTP. Es por esto, por lo que normalmente se utiliza una función *hash* [45] basada en la dirección del cliente (y posiblemente su puerto).

En un sistema Web distribuido el algoritmo puede ser distribuido o centralizado. No obstante siempre se debe tener en cuenta que la petición puede llegar inicialmente a cualquier nodo del sistema.

A continuación se presenta una clasificación de las políticas de asignación de peticiones.

Políticas sin información de estado Estas políticas no tienen en cuenta ninguna información de estado. La selección del nodo se realiza mediante alguna función que hace corresponder a cada petición un nodo.

Políticas basadas en información del cliente Estas políticas tienen en cuenta exclusivamente la información proveniente del cliente. La asignación del nodo se hace mediante una función que determina el nodo a partir de datos contenidos en la petición.

Políticas basadas en información del servidor Estas políticas tienen en cuenta la información del estado de carga de los nodos servidores para seleccionar el nodo al que se asigna la petición. El uso de estas políticas se basa en la toma de algún índice que indique el nivel de carga del nodo.

Políticas específicas para sistemas Web distribuidos Estas políticas tienen en cuenta detalles arquitectónicos que las hacen únicamente utilizables en sistemas Web distribuidos.

2.4 Políticas sin información de estado

Estas políticas no tienen en cuenta ninguna información de estado. La selección del nodo se realiza mediante alguna función que hace corresponder a cada petición un nodo. Esta selección se puede expresar como una función del número de petición que da como resultado el identificador del nodo seleccionado (véase expresión 2.1).

$$f : \mathbb{N} \longmapsto [0, n-1] \tag{2.1}$$

La asignación puede ser aleatoria estática (véase 2.4.1), circular estática (véase 2.4.2).

2.4.1 Asignación aleatoria estática

La asignación aleatoria estática utiliza una función que sigue una distribución estadística determinada. En el caso de que las capacidades de todos los nodos sean homogéneas, es usual utilizar una función que siga una distribución uniforme (véase ecuación 2.2). Esto puede conseguirse fácilmente utilizando un generador uniforme de números aleatorios.

$$P\left[f(i) = j\right] = \frac{1}{n} \qquad \forall j \in [0, n-1] \tag{2.2}$$

Si las capacidades de procesamiento de los nodos no son homogéneas se puede realizar una asignación de probabilidades basada en la capacidad relativa de cada nodo. Para ello se determina la capacidad relativa de cada nodo a partir de su capacidad absoluta C_j (véase

ecuación 2.3) y se utiliza dicha capacidad relativa como la tasa de peticiones que debe servir dicho nodo.

$$P\left[f\left(i\right)=j\right]=\frac{C_j}{\sum_{k=1}^{n}C_k}\qquad\forall j\in[0,n-1] \tag{2.3}$$

2.4.2 Asignación estática circular

Otra alternativa es el uso de una política de asignación circular (*round-robin*). La ecuación 2.4 muestra una función de asignación circular.

$$f\left(i\right)=i \bmod n \tag{2.4}$$

Para obtener una asignación de este tipo basta con guardar la última asignación. Si la última petición fue asignada al nodo i, la siguiente petición se asigna al nodo $(i+1) \mod n$.

En el caso de nodos heterogéneos, se puede modificar esta política incluyendo el uso de pesos [56]. A cada nodo se le asigna inicialmente un peso w_i^0 que indica la capacidad relativa del nodo, y se representa mediante w^* la suma de todos los pesos, que determina el período de asignación (véase ecuación 2.5).

$$\begin{aligned} w_j^0 &= \frac{C_j}{\min\limits_{k\in[0,n-1]}(C_k)} \\ w^* &= \sum_{j=0}^{n} w_j^0 \end{aligned} \tag{2.5}$$

Para la asignación de cada petición es necesario determinar el peso máximo m_p, y seleccionar el primero de los nodos de peso máximo s_p. Todos los pesos se mantienen, excepto el del nodo seleccionado que se decrementa en una unidad. Esta técnica de asignación se utiliza para las w^* primeras peticiones y luego las asignaciones se repiten cíclicamente (véase ecuación 2.6).

$$\begin{aligned} m_p &= \max_{j\in[0,n-1]}(w_j^p) \\ s_p &= j \Leftrightarrow w_j^p = m_p \wedge \nexists k < i \text{ que cumple } w_k^{p-1} = m_p \\ w_i^p &= \begin{cases} w_j^{p-1} - 1 & \Leftrightarrow j = s_p \\ w_j^{p-1} & \Leftrightarrow j \neq s_p \end{cases} \\ f\left(i\right) &= s_{i \bmod w^*} \end{aligned} \tag{2.6}$$

2.5 Políticas basadas en información del cliente

Estas políticas tienen en cuenta exclusivamente la información proveniente del cliente. La asignación del nodo se hace mediante una función que determina el nodo a partir de datos contenidos en la petición. La asignación pude ser por partición de clientes (véase 2.5.1), por partición de URL (véase 2.5.2), por partición de servicios (véase 2.5.3) o por tamaño de las respuestas (véase 2.5.4). En todos los casos la asignación se puede expresar como una función de la información de la petición P que da como resultado el identificador del nodo seleccionado (véase expresión 2.7).

$$f : P \longmapsto [0, n-1] \tag{2.7}$$

2.5.1 Asignación por partición de clientes

Los clientes pueden clasificarse en grupos y asignar cada grupo a un nodo. Esta clasificación puede realizarse mediante una función *hash* [45] o realizando una clasificación de los clientes efectivos [57].

La identificación de los clientes puede realizarse en la capa TCP. Esta identificación puede realizarse a partir de la dirección IP. No obstante, también puede tenerse en cuenta el puerto TCP.

La solución de clasificación mediante función *hash* tiene como ventaja el ser más general y se puede aplicar a distintas arquitecturas, incluyendo *clusters* Web virtuales. Una opción comúnmente utilizada es el uso de una función *hash* sobre el último byte de la dirección IP. Asumiendo que los valores de dicho byte estarán siempre distribuidos uniformemente, se puede utilizar como función el módulo sobre el número de nodos (véase ecuación 2.8).

$$f\left(p_i\right) = ultimobyte\left(ip\left(p_i\right)\right) \text{ mód } n \tag{2.8}$$

Por el contrario, la clasificación basada en los clientes efectivos realiza una partición de los clientes en función de la utilización pasada del servicio Web. Normalmente la extracción de la información se realiza a partir de los logs de los servidores, para realizar un agrupamiento de los clientes que distribuya de forma uniforme las peticiones entre los nodos servidores.

2.5.2 Asignación por partición de URL

La partición de URL necesita tener acceso a la petición HTTP completa. Esto permite que se pueda utilizar los datos de la petición de nivel de aplicación (en vez de utilizar únicamente los datos de nivel de TCP/IP). Una primera solución es dividir el árbol del sistema de archivos en n sub-árboles (de A_1 a A_n) que se alojan en cada uno de los n nodos servidores. Con esta solución, cuando llega una petición se analiza la URL solicitada y se asigna la petición al nodo que aloja el sub-árbol correspondiente (véase ecuación 2.9).

$$f\left(p_i\right) = j \Leftrightarrow url\left(p_i\right) \in A_j \tag{2.9}$$

No obstante, la partición del árbol plantea problemas en muchos casos. Para aplicarla, es necesario realizar un reparto equilibrado entre los nodos. Desafortunadamente, la mayoría de las peticiones en un servicio Web se concentran en una misma parte del árbol, por lo que se genera un importante desequilibrio en la asignación de peticiones.

Otra solución consiste, en aplicar una función *hash* sobre el la dirección URL para determinar el nodo al que se asigna la petición (véase ecuación 2.10).

$$f\left(p_i\right) = hash\left(url\left(p_i\right)\right) \tag{2.10}$$

Una función *hash* estándar tiene como principal inconveniente que casi todas las asignaciones se modifican cada vez que se modifica el número de nodos servidores activos. Esto puede tener un impacto muy negativo, ya que se generarían un gran número de fallos en caché. El algoritmo HRW [58] (*Highest Random Weight*) tiene como ventaja que el número de reasignaciones es mucho menor. La función de asignación selecciona el nodo para el que se obtiene un mayor peso W basado en la URL y el identificador del nodo i. El peso se obtiene como una generación de números pseudoaleatorios a partir del identificador del nodo y de la aplicación de una función de resumen $D()$ a la URL.

$$\begin{aligned} &f\left(p_i\right) = j \Leftrightarrow W\left(url\left(p_i\right), j\right) \geq W\left(url\left(p_i\right), k\right) \qquad \forall k \neq j \\ &W\left(u, i\right) = \left(1103515245 * \left(\left(1103515245 * i + 12345\right) \oplus D\left(u\right)\right) + 12345\right) \text{ mód } 2^{31} \end{aligned} \tag{2.11}$$

La asignación de nodo a partir de la URL también se puede realizar manteniendo una tabla de asignación, en la que se almacena el nodo en el que se encuentra almacenado cada objeto usando como clave su URL. En principio, este mecanismo podría presentar como inconveniente el presentar tiempos de búsqueda mayores que la aplicación de una función *hash*. Este tiempo se puede reducir notablemente si se combina el uso de estructuras de datos basadas en intentos (*LC-trie*) y la formalización de URL [59].

No obstante, todos los mecanismos de asignación por partición de URL mantienen el problema de no equilibrar la carga cuando las peticiones no son uniformes. Existen diversos estudios empíricos que demuestran que esto no es así [60, 61]. De hecho, un gran número de peticiones se concentran en un pequeño número de elementos [62].

2.5.3 Asignación por partición de servicios

La partición de servicios [63, 64] es de especial aplicación cuando los servicios proporcionados son heterogéneos. Este es el caso de los servicios Web que ofrecen distintos tipos de contenidos [54] (páginas HTML, fotografías, archivos multimedia, flujo de vídeo, contenidos dinámicos). En esta situación, se pueden usar nodos servidores especializados en ofrecer cierto tipo de contenido.

Una primera aproximación, es la partición estática (véase ecuación 2.12) en la que se asigna el nodo que sirve peticiones del tipo correspondiente. El tipo de petición se suele deducir por la extensión de la URL (todos los archivos con extensión `.jpg` son imágenes) o por el directorio en que se encuentra el elemento (todos los archivos en el directorio `cgi` son páginas dinámicas).

$$f\left(p_i\right) = j \Leftrightarrow tipo(p_i) = tipo(S_j) \tag{2.12}$$

Otra aproximación, es la clasificación de las peticiones por su consumo de recursos. La política CAP (*Client Aware Policy*) [27, 65] es una evolución de MC-RR [66] (*Multi Class Round Robin*). Clasifica las peticiones en cuatro categorías:

Estáticas Bajo esta categoría se incluyen las peticiones de elementos estáticos (como una página HTML o una fotografía), así como las peticiones dinámicas con muy pocas necesidades de procesamiento.

Ligadas a disco Bajo esta categoría se incluyen las peticiones que realizan un uso intensivo de disco (como una consulta compleja a base de datos).

Ligadas a CPU Bajo esta categoría se incluyen las peticiones que realizan un uso intensivo de la CPU (como aquellas que requieren cifrado de los contenidos).

Ligadas a disco y CPU Combinación de las dos anteriores.

El objetivo de la política CAP es mantener el equilibrio entre los distintos nodos servidores para cada una de las categorías. De esta forma, los conjuntos de peticiones (véase 2.13a) estáticas $P_{Estatica}$, peticiones ligadas a disco P_{Disco}, peticiones ligadas a CPU P_{CPU} y peticiones ligadas a disco y CPU P_{mixta} forman una partición sobre el conjunto de peticiones P. Las peticiones que llegan al servidor se pueden ver como una sucesión (véase 2.13b) de peticiones o como una sucesión independiente por cada clase de petición (véase 2.13c). La asignación de peticiones se puede hacer de forma cíclica independiente para cada clase de peticiones (véase 2.13d).

$$P = P_{Estatica} \cup P_{Disco} \cup P_{CPU} \cup P_{mixta} \tag{2.13a}$$

$$S = \{p_i\} : p_i \in P \tag{2.13b}$$

$$p_i = p_j^c : p_j^c \in P_c \tag{2.13c}$$

$$f\left(p_j^c\right) = j \bmod n \tag{2.13d}$$

2.5.4 Asignación de intervalos de tamaños a tareas con igual carga (SITA-E)

La asignación de intervalos de tamaños a tareas con igual carga (*Size Interval Task Assignment with Equal Load*) [67] está especialmente orientada a servicios Web eminentemente estáticos. Esta asignación parte de la premisa de que el tiempo de procesamiento es proporcional al tamaño del elemento servido. SITA-E considera que los tamaños de los archivos

siguen una distribución de Pareto limitada (véase ecuación 2.14). Dicha distribución está caracterizada por su variabilidad α, el menor tamaño posible k y el mayor tamaño posible p.

$$f(x) = \frac{\alpha k^{\alpha} x^{-\alpha-1}}{1 - \left(\frac{k}{p}\right)^{\alpha}} \qquad k \leq x \leq p \text{ y } 0 \leq \alpha \leq 2 \tag{2.14}$$

SITA-E define un conjunto de puntos de corte $x_0, x_1, \ldots, x_n$ (con $x_0 = k$ y $x_n = p$) en el rango de tamaños de archivo de forma que la carga se distribuya uniformemente entre todos los nodos. Es decir, que dichos puntos deben cumplir que la esperanza de la distribución en cada intervalo sea la misma (véase ecuación 2.15).

$$\begin{aligned} &E_i\{x\} = \int_{x_i}^{x_{i+1}} x f(x)\, dx \\ &E_0\{x\} = E_1\{x\} = \ldots = E_{n-1}\{x\} = \frac{E\{x\}}{n} \\ &\int_{x_0}^{x_1} x f(x)\, dx = \int_{x_1}^{x_2} x f(x)\, dx = \ldots \int_{x_{n-1}}^{x_n} x f(x)\, dx \end{aligned} \tag{2.15}$$

En el caso de la distribución limitada de Pareto, la asignación para los puntos de corte tiene solución analítica (véase ecuación 2.16).

$$x_i = \left(\frac{(h-i)}{h} k^{1-\alpha} + \frac{i}{h} p^{1-\alpha}\right)^{\frac{1}{1-\alpha}} \qquad \forall i = 0, \ldots, n \text{ y } \alpha \neq 1 \tag{2.16}$$

La asunción principal de SITA-E es que el tiempo de procesamiento es proporcional al tamaño del elemento solicitado. Esto es válido para el servicio de páginas estáticas. Pero esta asunción no puede mantenerse cuando se trata de páginas dinámicamente generadas. Incluso en el caso de elementos estáticos, SITA-E no tiene en cuenta el impacto de ciertas optimizaciones que pueden existir en el núcleo o en la pila TCP [15] y que pueden tener efecto especialmente para archivos pequeños.

2.6 Políticas basadas en información del servidor

Estas políticas tienen en cuenta la información del estado de carga de los nodos servidores para seleccionar el nodo al que se asigna la petición. El uso de estas políticas se basa en la toma de algún índice que indique el nivel de carga del nodo (utilización de CPU, memoria disponible, utilización de E/S, número de conexiones activas, etc.). La asignación puede expresarse como una función del conjunto de peticiones P y del conjunto de estados E que da como resultado el identificador del nodo seleccionado (véase expresión 2.17). La selección puede realizarse atendiendo al nodo menos cargado (véase 2.6.1), mediante una política cíclica por pesos evaluados dinámicamente (véase 2.6.2) o intentando que peticiones sucesivas de un mismo cliente sean servidas por un mismo nodo para incrementar los aciertos de caché (véase 2.6.3).

$$f : P \times E \longmapsto [0, n-1] \tag{2.17}$$

2.6.1 Asignación al nodo menos cargado

Existen diversas soluciones que asignan las peticiones al nodo menos cargado [68]. La política concreta dependerá del índice de carga seleccionado. Los índices de carga más utilizados son el número de conexiones activas o el tiempo de respuesta.

En general, esta familia de políticas se utiliza en *clusters* Web. Esto permite la implantación de la política en el *switch* Web.

En el caso de la asignación de peticiones basada en el número de conexiones activas, el índice utilizado es instantáneo. El *switch* Web conoce en todo momento cuantas conexiones existen con cada nodo servidor y realiza la asignación al nodo con menor número de conexiones abiertas. En este caso la función de asignación de peticiones $f()$ depende exclusivamente del tiempo y del número de conexiones activas c_i de cada nodo (véase ecuación 2.18).

$$f(t) = i \Leftrightarrow c_i(t) \leq c_j(t)\ \forall j \neq i \tag{2.18}$$

2.6.2 Asignación dinámica cíclica por pesos

La asignación dinámica cíclica por pesos [69] (*Weighted Round Robin*) es una variante de la asignación cíclica por pesos (véase 2.4.2) en la que los pesos se recalculan periódicamente en función de la carga de cada nodo.

Periódicamente, se calcula para cada nodo servidor un índice de métricas [70] de carga $L_i\,(t)$. Este índice se evalúa, para un período de tiempo $[t - \Delta t, t]$, a partir de un conjunto de métricas de carga. Las métricas de carga utilizadas pueden dividirse en tres categorías:

Métricas de entrada Son estimaciones del estado de los nodos servidores a partir de las peticiones que se reciben. Un ejemplo de este tipo de métricas, es el número de peticiones recibidas durante el período de medida.

Métricas de estado Son medidas del estado de un nodo servidor a partir de su consumo de recursos al final de un período de medida. Como ejemplo de este tipo de métricas se pueden citar, el número de procesos en ejecución, o la cantidad de memoria en uso.

Métricas de servicio Son medidas del rendimiento de un nodo servidor en cuanto al servicio de peticiones. Por ejemplo, se puede medir el tiempo medio necesario para servir cada petición durante el período de medida.

Dado un conjunto de c métricas de carga, se puede representar mediante $m_j^i\,(t)$ la j-ésima métrica sobre el i-ésimo servidor durante el período $[t - \Delta t, t]$. A cada métrica se le asigna un peso relativo r_j, que se considera un parámetro estático de configuración, de forma que se cumpla la ecuación 2.19.

$$\sum_{j=1}^{c} r_j = 1 \tag{2.19}$$

De esta forma, el índice de métricas de carga se determina como la suma de las métricas ponderadas por los factores de peso relativo (véase ecuación 2.20a). Los pesos a utilizar

durante el siguiente periodo serán los índices de carga, si la modificación es superior a un cierto umbral ϵ (véase ecuación 2.20b).

$$L_i\left(t\right) = \sum_{j=1}^{c} r_j m_j^i\left(t\right) \tag{2.20a}$$

$$w_i(t) = \begin{cases} w_i\left(t-\Delta t\right) & \text{si } \left|w_i\left(t-\Delta t\right) - L_i(t)\right| < \epsilon \\ L_i\left(t\right) & \text{en otro caso} \end{cases} \tag{2.20b}$$

2.6.3 Distribución de peticiones consciente de la localidad (LARD)

La política LARD (*Locality Aware Request Distribution*) [38, 71, 39] intenta asignar todas las peticiones de un cierto elemento al mismo nodo, siempre que éste se encuentre por debajo de un cierto umbral de carga. Cuando se supera dicho umbral, se selecciona otro nodo para las siguientes peticiones del mismo elemento. El objetivo de esta idea es conseguir el máximo número de aciertos en la caché de disco y del sistema de archivos.

Esta política se basa en tres métricas de coste: el coste de equilibrio c_e (véase ecuación 2.21a), el coste de localidad c_l (véase ecuación 2.21b) y el coste de remplazo c_r (véase ecuación 2.21c). Estas métricas utilizan como valores de referencia un valor de infrautilización r_i (valor de carga por debajo del cual el nodo está infrautilizado), un valor de sobrecarga r_s (valor por encima del cual el nodo ofrece tiempos de respuesta inaceptables) y un valor de fallo r_f (valor que representa el coste de un fallo en caché). El coste de equilibrio representa el retraso en el servicio de una petición provocado por el resto de peticiones en la cola del nodo. El coste de localidad representa el retraso en el servicio de una petición provocado por el acierto o fallo en la cache del nodo. El coste de remplazo representa el retraso futuro que se producirá por el remplazo de un elemento en la cache del nodo.

$$c_e\left(i\right) = \begin{cases} 0 & \text{si } carga\left(i\right) < r_i \\ \infty & \text{si } carga\left(i\right) > r_s \\ carga\left(i\right) - r_i & \text{en otro caso} \end{cases} \tag{2.21a}$$

$$c_l\left(e,i\right) = \begin{cases} 1 & \text{si } e \text{ en caché de } S_i \\ r_f & \text{en otro caso} \end{cases} \tag{2.21b}$$

$$c_r\left(e,i\right) = \begin{cases} 0 & \text{si } carga\left(i\right) < r_i \\ 0 & \text{si } e \text{ en caché de } S_i \\ r_f & \text{en otro caso} \end{cases} \tag{2.21c}$$

$$c\left(e,i\right) = c_e\left(i\right) + c_l\left(e,i\right) + c_r\left(e,i\right) \tag{2.21d}$$

El coste agregado c (véase ecuación 2.21d) es una métrica del coste de asignar una petición a cada uno de los nodos servidores. Cada petición se asigna al nodo para el que el coste agregado es menor (véase ecuación 2.22).

$$f(p) = i \Leftrightarrow c(p,i) \leq c(p,j) \, \forall j \neq i \tag{2.22}$$

2.7 Políticas específicas para sistemas Web distribuidos

Existen políticas que solamente son aplicables a la arquitectura de un sistema Web distribuido. Las políticas de esta familia varían desde las totalmente centralizadas (véase 2.7.1) hasta las totalmente distribuidas (véase 2.7.2).

2.7.1 Asignación centralizada para sistemas Web distribuidos

La idea de esta política es coordinar [50] el comportamiento de los nodos de un sistema Web distribuido con el encaminamiento realizado por una implementación dinámica del servicio de DNS de autoridad. Esto constituye una mejora sobre las implementaciones cíclicas del servicio de DNS [49, 48]. Dicho servicio realiza la asignación de nodos teniendo en cuenta la carga de cada nodo y la carga generada por cada dominio de clientes. No obstante, tal y como se mencionó anteriormente (véase 2.2.3), el encaminamiento mediante DNS dinámico tiene efecto sobre un pequeño porcentaje de las peticiones. Para reforzar el comportamiento de la política, cada nodo realiza redirecciones para las peticiones que no le corresponden. La Figura 2.15 muestra la estructura general de la política de asignación centralizada.

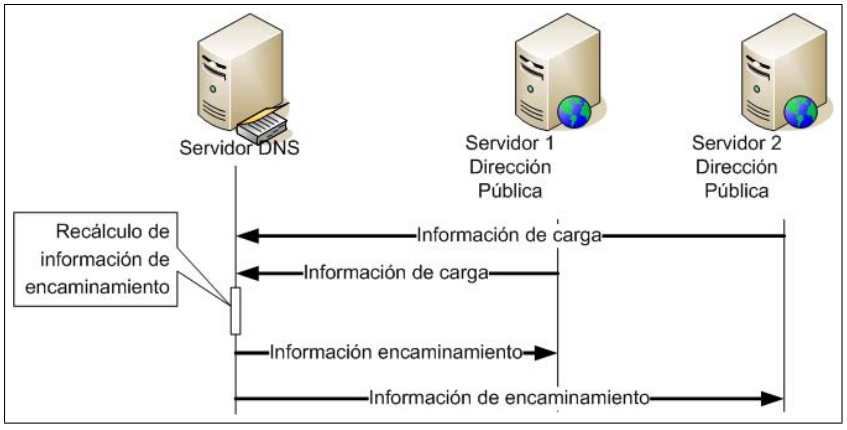

Figura 2.15: Estructura general de la política de asignación centralizada para sistemas Web distribuidos.

En cada nodo existe un servidor Web con un módulo de redirección que es responsable de decidir si una petición recibida debe redirigirse o no. Este módulo recibe una actualización periódica de la información de encaminamiento por parte del servicio de DNS.

Cada nodo tiene, además, un módulo de recogida de información de carga, que es responsable de recoger dos tipos de información:

Carga del nodo El módulo realiza un seguimiento de las condiciones de carga del nodo.

Carga de clientes El módulo analiza las peticiones recibidas (posiblemente a través del archivo de log), para determinar los dominios [57, 72] a los que pertenecen los clientes.

Los módulos de recogida de información local de cada nodo notifican periódicamente al servicio de DNS dinámico su estado de carga para que realice una fusión de los mismos y actualice las tablas de redirección, que serán, además propagadas a todos los nodos servidores.

2.7.2 Asignación distribuida para sistemas Web distribuidos

La asignación distribuida, al contrario que la centralizada, no requiere una coordinación [50] entre el servicio de DNS y los nodos servidores. Por el contrario, la política de asignación se implementa en los nodos servidores. La Figura 2.16 muestra la estructura general de la política de asignación distribuida.

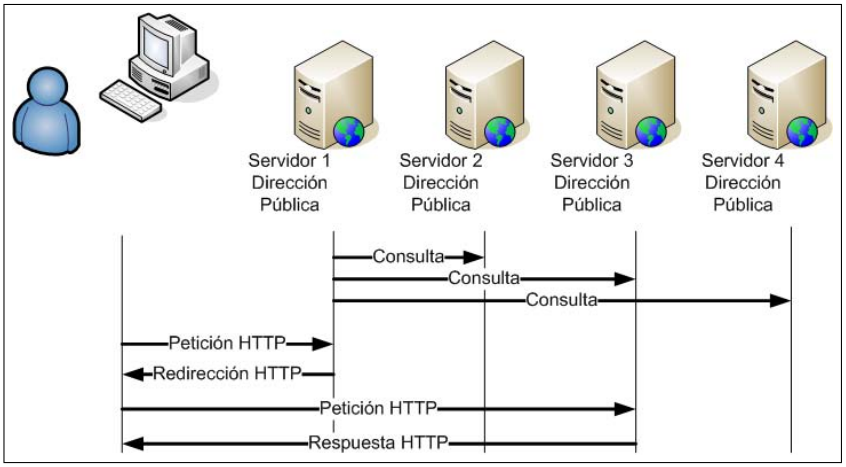

Figura 2.16: Estructura general de la política de asignación distribuida para sistemas Web distribuidos.

En cada nodo existe un módulo de redirección que es responsable de decidir si una petición debe redirigirse o no. Cada nodo tiene, además, un módulo de recogida de información de carga. Este módulo mantiene la información de carga local y, además, consulta periódicamente la información de carga de un subconjunto de nodos. El subconjunto de nodos puede variar desde un único nodo (el propio nodo local) hasta todos los nodos del sistema. No obstante, lo más habitual es el uso de un pequeño subconjunto de nodos seleccionados de forma aleatoria.

2.8 Resumen

Ante la necesidad de la mejora del rendimiento de un servidor Web, las dos primeras alternativas que se plantean son el escalado hardware y el escalado software. Ambas alternativas se centran en la mejora de las prestaciones manteniendo la restricción de mantener el servidor alojado en un único nodo.

Otra alternativa consiste en el uso de una arquitectura distribuida en la que un conjunto de nodos ofrecen el servicio. De este modo, se puede escalar el sistema mediante la adición de nuevos nodos al sistema. En este capítulo se ha presentado una clasificación de las arquitecturas distribuidas para servidores Web existentes en la actualidad: los *sistemas Web basados en cluster*, los *clusters Web virtuales* y los *sistemas Web distribuidos*.

Los *sistemas Web basados en cluster* se apoyan en la utilización de un nodo central o *switch Web* que se encarga de distribuir las peticiones entre los nodos para lo que se puede utilizar diversos mecanismos de encaminamiento. Dichos mecanismos pueden ser independientes del contenido (trabajando en el nivel TCP de la pila de protocolos) o dependientes del contenido (trabajando en el nivel HTTP de la pila de protocolos). Además el encaminamiento puede ser de flujo bidireccional (las respuestas también pasan por el *switch Web*) o de flujo unidireccional (los nodos servidores pueden contestar directamente a los clientes).

Los *clusters Web virtuales* comparten una misma dirección IP y no necesitan ningún nodo central para realizar el reparto. En este caso, todas la peticiones llegan a todos los nodos, y cada nodo es responsable de realizar un filtrado y decidir a qué peticiones debe atender. En esta arquitectura se hace necesario que el proceso de filtrado no permita que dos nodos decidan satisfacer una misma petición. Esto suele hacerse mediante la aplicación de algún algoritmo *hash* sobre la información del cliente que origina la petición.

Los *sistemas Web distribuidos* utilizan un conjunto de nodos servidores en los que cada nodo dispone de una dirección IP pública. Como primer mecanismo de distribución se suele incorporar el uso de un servidor DNS dinámico, aunque este mecanismo no es plenamente efectivo. Un segundo mecanismo de distribución de peticiones consiste en la posibilidad de que un nodo servidor pueda redirigir ciertas peticiones a otro nodo del *cluster*.

Un aspecto importante en todas las soluciones es la política utilizada para la asignación de peticiones a nodos servidores. Estas políticas deciden qué nodo servidor es responsable de dar servicio a cada petición. En este capítulo se ha realizado una clasificación de las políticas existentes en cuatro grupos: *políticas sin información de estado*, *políticas basadas en información del cliente*, *políticas basadas en información del servidor* y *políticas específicas para sistemas Web distribuidos*.

Todas las políticas de asignación de peticiones se basan en considerar que todos los nodos alojan todos los contenidos del sitio Web o que cada contenido se encuentra alojado en un único nodo. Por ejemplo, la asignación al nodo menos cargado selecciona el nodo con menos carga sin realizar ninguna otra consideración por lo que asume que cualquier petición puede ser servida por cualquier nodo servidor. En el otro extremo se puede considerar, por ejemplo, la asignación por partición de URL que realiza la selección del nodo servidor determinando, a partir de la URL, cuál es el nodo servidor que almacena el contenido.

Un objetivo de esta tesis es la replicación parcial de contenidos. Otro objetivo es la adaptación dinámica de la asignación de dichos contenidos a los nodos servidores. En el siguiente capítulo se estudia la compatibilidad de las arquitecturas existentes con los objetivos mencionados y se realizan las propuestas arquitectónicas que se consideran necesarias para alcanzar dichos objetivos. En el capítulo 4 se proponen las políticas de replicación de contenidos entre nodos, así como las políticas de asignación de peticiones que pueden utilizarse con las arquitecturas propuestas en el capítulo 3.

Capítulo 3

Propuestas arquitectónicas

En este capítulo se realizan propuestas de modificación de las arquitecturas distribuidas existentes que se han presentado en el capítulo anterior. Dichas propuestas se derivan de la necesidad de adaptar las arquitecturas para que sean compatibles (si esto es posible) con los objetivos de esta tesis (el uso de replicación parcial de contenidos y la adaptación dinámica de la asignación de los mismos a los nodos servidores). Tras realizar las propuestas pertinentes para cada familia de arquitecturas, se presenta **una nueva propuesta arquitectónica** de especial interés en presencia de réplicas parciales: el ***cluster Web con switch distribuido***.

Como ya se ha presentado en el capítulo 1 (véase sección 1.2), dos objetivos restringen el tipo de soluciones que se estudian en esta tesis. Por una parte, el uso de replicación parcial de contenidos, que hace que cada elemento se distribuya en un subconjunto de nodos servidores. Por otra parte, la asignación adaptativa de los contenidos a los nodos servidores, que revisa la asignación de los elementos dependiendo de las necesidades del sistema.

En una solución basada en replicación parcial cada elemento e_i se encuentra almacenado en un subconjunto del conjunto de nodos servidores. Este hecho tiene impactos sobre la arquitectura del sistema. La principal razón es que esta restricción elimina la libertad de distribuir una petición a cualquier nodo servidor. Además, se hace necesario el establecimiento de un mecanismo que permita determinar de forma eficiente los nodos que albergan un determinado recurso. En este capítulo se estudian las consecuencias de esta restricción sobre cada una de las arquitecturas existentes (*sistemas Web basados en cluster*, *clusters Web virtuales* y *sistemas Web distribuidos*).

La asignación de réplicas se puede realizar de forma estática (una asignación inicial que se mantiene sin alteraciones en el tiempo) o dinámica (la asignación de réplicas a nodos se modifica a lo largo del tiempo). Las políticas de asignación de réplicas, tanto estáticas como dinámicas, se discuten en el capítulo 4. Independientemente de la política seleccionada, el hecho de que la asignación de réplicas sea estática o dinámica ejerce un impacto sobre la arquitectura del sistema. Dicho impacto se estudia en este capítulo para cada posible arquitectura.

El esquema más simple de asignación de contenidos a los nodos servidores es el de naturaleza estática. En dicho enfoque, se determinan, en un momento inicial, el número de réplicas que se va a tener de cada elemento y el nodo servidor en que se va a alojar cada una de las réplicas. Desgraciadamente, la asignación estática de contenidos carece de información sobre el grado de utilización de los distintos elementos y no puede realizar ninguna acción de

adaptación a las necesidades del servicio.

Un esquema más avanzado es el de asignación dinámica de contenidos a los nodos servidores. En este caso, se revisa periódicamente la asignación de contenidos a los nodos para determinar el número de réplicas de cada recurso y el nodo en que debe alojarse dicha réplica. El objetivo perseguido es la adaptación a las necesidades del servicio.

El capítulo presenta las propuestas arquitectónicas para cada una de las arquitecturas existentes y concluye con una propuesta novedosa de solución arquitectónica. La sección 3.1 presenta las propuestas arquitectónicas para *sistemas Web basados en cluster*. La sección 3.2 justifica las razones que hacen imposible el uso de un *cluster Web virtual* en presencia de réplicas parciales. La sección 3.3 presenta las propuestas arquitectónicas para *sistemas Web distribuidos*. La seccion 3.4 realiza una comparación entre las propuestas de arquitecturas adaptadas. Tras esta comparación, la sección 3.5 presenta una propuesta novedosa: el ***cluster Web con switch distribuido***. Finalmente, la sección 3.6 realiza una presentación de las conclusiones relativas a las propuestas arquitectónicas.

3.1 Análisis de *sistemas Web basados en cluster*

En un *sistema Web basado en cluster* existe un nodo (*switch Web*) encargado del reparto de peticiones y un conjunto de nodos encargados de dar servicio a las peticiones recibidas.

El encaminamiento de peticiones más sencillo es el bidireccional. Este es el encaminamiento utilizado por los mecanismos pasarela TCP y empalme TCP. Las peticiones llegan al *switch Web* que las envía a un nodo servidor para su procesamiento. Posteriormente, el nodo servidor envía la respuesta al *switch Web* que es el responsable de contestar al cliente. La diferencia entre los dos mecanismos radica en que la pasarela TCP actúa en el nivel de aplicación y el empalme TCP actúa en niveles inferiores, requiriendo modificaciones en la pila TCP y proporcionando un mejor rendimiento.

Un *sistema Web basado en cluster* también puede utilizar encaminamiento unidireccional. Este el encaminamiento que se utiliza con el mecanismo de cesión TCP. La peticiones llegan al *switch Web* que las envía a un nodo servidor que, tras procesar la petición, contesta por un canal de salida distinto al de entrada. Este mecanismo requiere la modificación de la pila TCP tanto en el *switch Web* como en los nodos servidores.

En las siguientes secciones se comienza realizando un análisis de las restricciones sobre la arquitectura derivadas de los objetivos de esta tesis, para posteriormente pasar a presentar la familia de propuestas arquitectónicas para *sistemas Web basados en cluster*, dependiendo del tipo de replicación parcial (estática o dinámica) y del tipo de flujo de peticiones (unidireccional o bidireccional).

3.1.1 Restricciones sobre la arquitectura

En función de los objetivos establecidos en esta tesis, existen dos factores que imponen restricciones sobre la arquitectura general de un sistema Web basado en *cluster*. Por una parte la existencia de un mecanismo de replicación parcial y, por otra parte, la asignación adaptativa de contenidos.

3.1.1.1 Replicación parcial

Un objetivo fundamental del presente trabajo es el uso de estrategias de réplicas parciales. Esto quiere decir que un elemento e_i solo reside en un subconjunto de nodos servidores. Cuando llega una petición de un elemento, la asignación solo puede tener como resultado uno de los nodos que contienen dicho elemento (véase Figura 3.1).

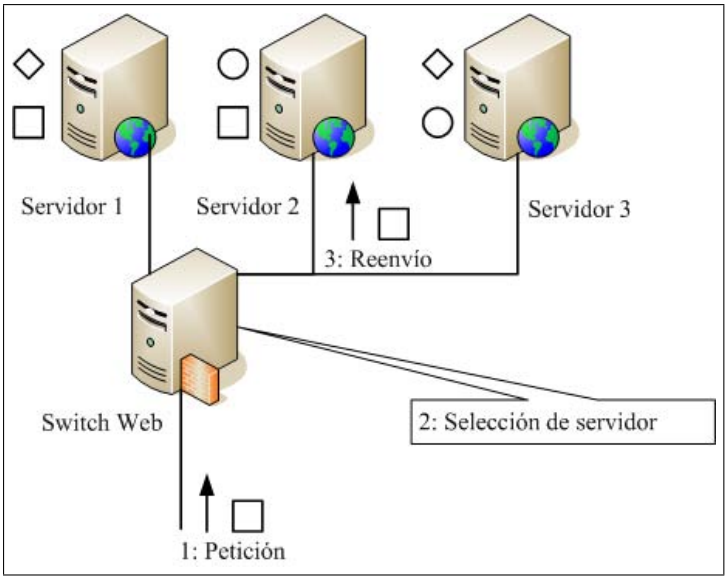

Figura 3.1: Encaminamiento de peticiones en *sistemas Web basados en cluster* con replicas parciales.

Por tanto, el *switch Web* debe mantener la información de la asignación de los contenidos a los nodos y realizar la asignación a partir de la información de la petición. Esto implica que el único tipo de encaminamiento aceptable es aquel que es dependiente del contenido.

La replicación parcial de contenidos implica que cada elemento se encuentra en un subconjunto de nodos servidores. Cuando llega una petición al *switch*, éste es responsable de seleccionar un nodo servidor en el que resida una copia del elemento y realizar el encaminamiento de la petición.

Es importante tener en cuenta que la existencia de réplicas parciales tiene impacto sobre las políticas de asignación de peticiones que se pueden utilizar. La política de asignación de peticiones debe ser capaz de determinar eficientemente el subconjunto de nodos que contienen una réplica. Esto requiere el uso de algún servicio de directorio en el que se mantenga dicha información [54]. Aunque un servicio de directorio genérico puede provocar retrasos importantes, se pueden construir estructuras de datos con bajo consumo de recursos y que impongan pequeños retrasos [29, 59]. Por ejemplo, las estructuras de datos propuestas por Luo et al. en [59] basadas en la formalización de URL y el uso de tablas hash multinivel permiten el almacenamiento de la información de 76000 archivos en una estructura de datos de unos 540KB y con un tiempo de consulta de 1.12 microsegundos.

En definitiva, la replicación parcial hace necesario:

- El uso de una arquitectura que permita un encaminamiento dependiente del contenido

(pasarela TCP, empalme TCP o cesión TCP). El encaminamiento independiente del contenido no es aplicable en el caso de replicación parcial, puesto que al no encontrarse los elementos en todos los nodos es preciso conocer el elemento solicitado antes de decidir el nodo servidor que se encargará de la petición.

- La definición de una política de asignación de contenidos a nodos, ya sea ésta estática o dinámica.
- El uso de una política de asignación de peticiones basada en una estructura de datos que mantenga información actualizada sobre la asignación de contenidos a nodos.

3.1.1.2 Asignación de contenidos adaptativa

Puesto que la asignación de contenidos debe adaptarse a las necesidades del servicio, es necesario una reasignación de los contenidos a los nodos dependiendo del estado de cada elemento del sistema. Esta reasignación debe hacerse en función de de las peticiones recibidas anteriormente y del estado de cada nodo servidor.

El *switch Web* puede determinar la información de estado necesaria realizando una monitorización de su actividad. En algunos casos puede ser necesario, además, la consulta de información de estado adicional sobre los nodos servidores. Esto último ocurre cuando se realiza un encaminamiento unidireccional, en el que el *switch Web* puede no ser consciente de algunos parámetros de las respuestas, ya que éstas no pasan por él, sino que son directamente enviadas por los nodos servidores a los clientes.

Una modificación sobre la arquitectura base de un *sistema Web basado en cluster* que puede mejorar el funcionamiento de una asignación de contenidos adaptativa es el uso de una red adicional de servicio, para realizar las redistribuciones de contenidos sin afectar a la red de datos principal del *cluster* (véase Figura 3.2).

3.1.2 Propuesta de soluciones arquitectónicas

Teniendo en cuenta las restricciones establecidas en esta tesis se proponen distintas alternativas de arquitectura para *sistemas Web basados en cluster*.

Las variaciones en las distintas arquitecturas presentadas vienen dadas por dos propiedades:

Tipo de asignación de réplicas Se distingue entre las necesidades arquitectónicas de la asignación estática de réplicas y la asignación dinámica de réplicas.

Tipo de flujo de encaminamiento Se distingue entre las necesidades derivadas del encaminamiento bidireccional y el encaminamiento unidireccional.

Estas variaciones dan lugar a cuatro posibles variantes arquitectónicas:

WC RE/FB *Cluster Web* con asignación estática de réplicas y encaminamiento con flujo bidireccional.

WC RE/FU *Cluster Web* con asignación estática de réplicas y encaminamiento con flujo unidireccional.

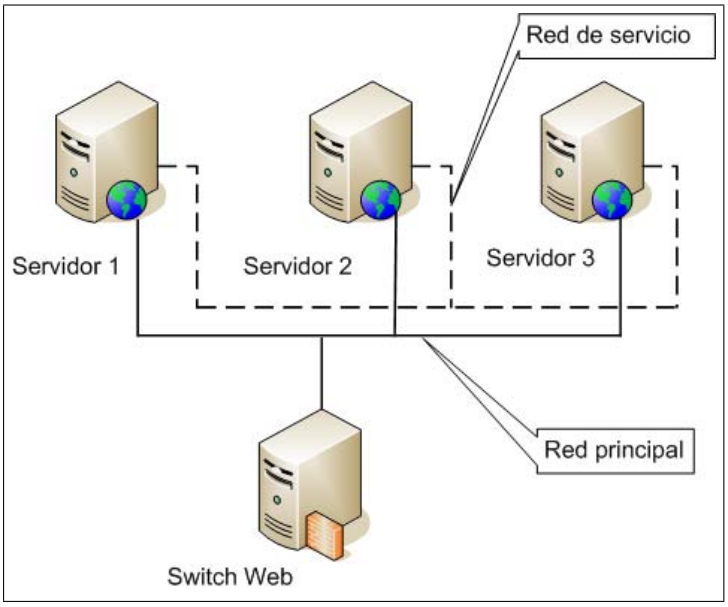

Figura 3.2: Arquitectura de un *sistema Web basado en cluster* con red de servicio para redistribución.

WC RD/FB *Cluster Web* con asignación dinámica de réplicas y encaminamiento con flujo bidireccional.

WC RD/FU *Cluster Web* con asignación dinámica de réplicas y encaminamiento con flujo unidireccional.

3.1.2.1 Propuesta arquitectónica WC RE/FB

La arquitectura más básica para el caso de un *sistema Web basado en cluster* es la que se basa en una asignación estática de réplicas y con encaminamiento de peticiones con flujo bidireccional (véase Figura 3.3).

La asignación estática de réplicas se discute con detalle en la sección 4.2. Cualquiera que sea la política de asignación utilizada, su aplicación se efectúa una única vez antes de la puesta en funcionamiento del sistema y no impone ningún requerimiento sobre la arquitectura del sistema. Al realizarse la asignación de réplicas antes de la puesta en funcionamiento, no es necesario el uso de una red de servicio.

El uso de encaminamiento de peticiones con flujo bidireccional permite que cada nodo servidor tenga una única interfaz de red por la que recibe las peticiones y envía las respuestas.

En el caso de que el encaminamiento se realice mediante pasarela TCP no es necesaria ninguna modificación en el sistema operativo del *switch Web*, ya que el encaminamiento se implementa mediante una aplicación HTTP que analiza las peticiones y realiza peticiones a los correspondientes nodos servidores. Es más, esta solución no plantea ningún problema con las conexiones de cliente persistentes o las peticiones en *pipeline* ya que puede realizar cada petición a un nodo servidor distinto.

Si el encaminamiento se realiza mediante empalme TCP es necesaria la modificación de la pila TCP del nodo correspondiente al *switch Web*. Esto se debe a que la optimización

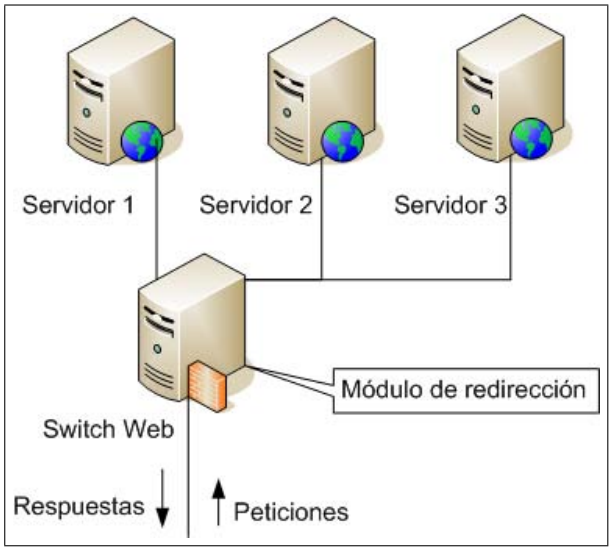

Figura 3.3: Arquitectura WC RE/FB un *sistema Web basado en cluster*.

ofrecida por el empalme TCP se consigue mediante el reenvío de paquetes desde el nivel de red. Además en este caso se debe tener en cuenta que si el cliente utiliza conexiones persistentes o envía peticiones en *pipeline* se debe realizar una selección de nodo servidor para cada petición.

Tanto si el encaminamiento se realiza mediante pasarela TCP como si se realiza mediante empalme TCP, la latencia asociada con la apertura y cierre de conexiones TCP se puede reducir si el *switch Web* mantiene un conjunto de conexiones permanentemente abiertas con cada nodo servidor.

Para que el *switch Web* pueda realizar sus funciones es necesaria la cooperación de varios módulos (véase Figura 3.4).

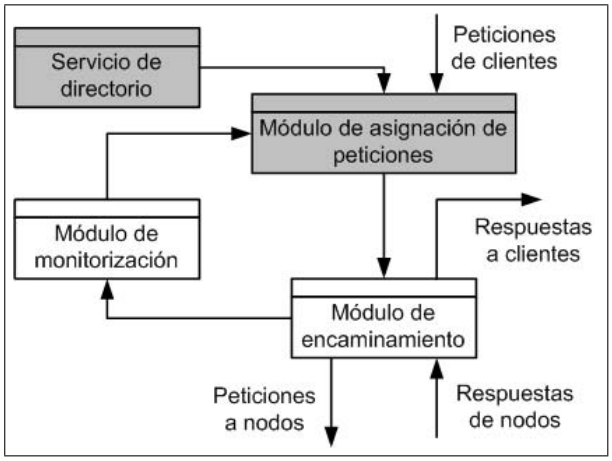

Figura 3.4: Módulos que intervienen en el procesamiento de peticiones en un sistema WC RE/FB.

A continuación se presenta una descripción de las funciones de cada módulo:

Módulo de asignación de peticiones Es el módulo encargado de recibir las peticiones y, tras analizarlas, asignar el servicio de cada petición a un nodo servidor. Las asignación de peticiones se realiza de acuerdo con la política de asignación de peticiones vigente. En cualquier caso, la asignación puede necesitar tener en cuenta la asignación de réplicas a nodos servidores (suministrada por el servicio de directorio) y la información de estado del *cluster* (suministrada por el módulo de monitorización).

Módulo de encaminamiento Recibe las peticiones que han sido asignadas por el módulo de asignación de peticiones con la correspondiente asignación del nodo servidor que debe procesarlas y realiza el encaminamiento bidireccional. Esto incluye el envío de peticiones a los nodos servidores así como la recepción de respuestas de los nodos servidores y su retransmisión a los clientes.

Servicio de directorio Mantiene la información de asignación de réplicas a nodos servidores y permite la consulta de dicha información por parte del módulo de asignación de peticiones.

Módulo de monitorización Recibe del módulo de encaminamiento la información de las peticiones enviadas a los nodos servidores así como de las respuestas recibidas. Con esta información construye la información de estado que suministra al módulo de asignación de peticiones.

3.1.2.2 Propuesta arquitectónica WC RE/FU

Esta arquitectura se basa en la asignación estática de réplicas y encaminamiento de peticiones con flujo unidireccional (véase Figura 3.5). Su diferencia con la arquitectura RE/FB es que el encaminamiento de peticiones sigue un flujo unidireccional en vez de bidireccional.

El uso de encaminamiento de peticiones con flujo unidireccional requiere que cada nodo servidor tenga dos interfaces de red:

- Una interfaz de red para la conexión con el *switch Web*. Esta es la interfaz de red utilizada para la recepción de las peticiones enviadas por el *switch Web* al nodo servidor.
- Una interfaz de red para la conexión con el canal de salida. Esta es la interfaz de red utilizada para el envío de las respuestas HTTP a los clientes que se realiza de forma directa y sin pasar por el *switch Web*.

Además, en este caso es necesario que tanto el *switch Web* como los nodos servidores ejecuten una versión modificada del sistema operativo que permita la implantación del protocolo de cesión TCP.

La estructura interna propuesta para el *switch Web* (véase Figura 3.6) es similar a la del *switch* de un cluster WC RE/FB, aunque con algunas variaciones, que se describen a continuación.

La principal diferencia en la estructura interna del *switch Web* con respecto a la arquitectura WC RE/FB es que en este caso el módulo de encaminamiento tiene que implementarse necesariamente en el nivel del núcleo del sistema operativo y que requiere la modificación

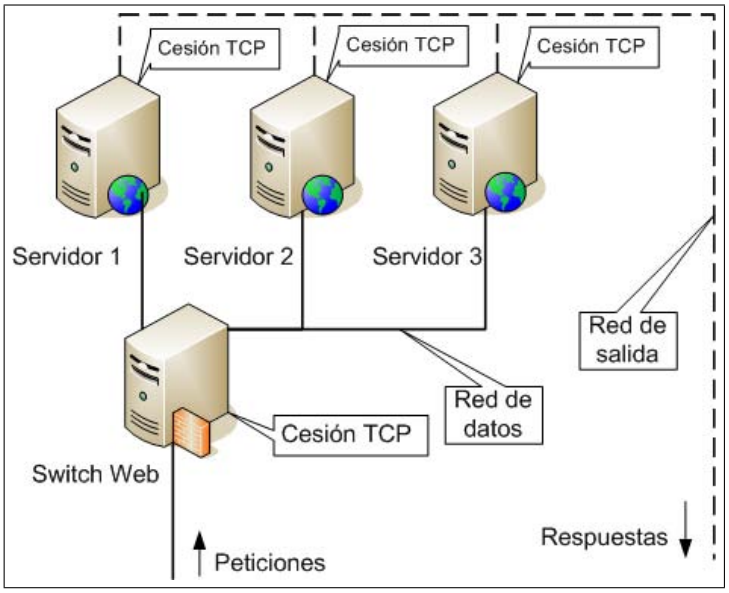

Figura 3.5: Arquitectura WC RE/FU un *sistema Web basado en cluster*.

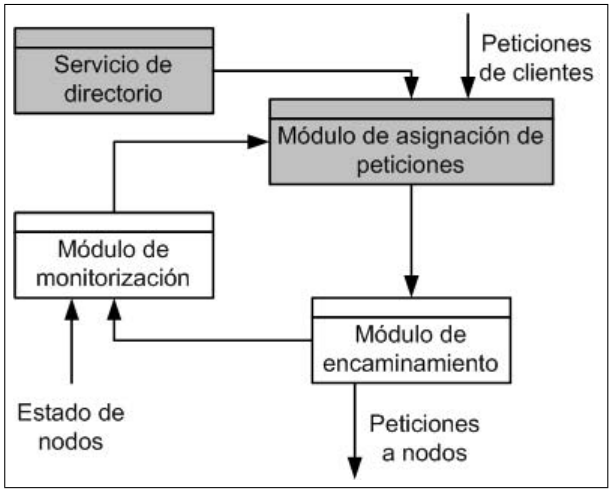

Figura 3.6: Módulos que intervienen en el procesamiento de peticiones en un sistema WC RE/FU.

de la pila TCP. Además dicho módulo de encaminamiento solamente se responsabiliza del encaminamiento de las peticiones y no de las respuestas. Esto se debe a que las respuestas las envían los nodos servidores directamente a los clientes.

Este último hecho hace que el módulo de monitorización no pueda obtener toda la información directamente del resto de módulos del *switch Web* y que tenga que obtener información de estado enviada por los nodos servidores, ya que las respuestas no pasan por el *switch*.

Por otra parte en este caso, no es suficiente con que los nodos servidores tengan alojado un servidor Web estándar. Además, de forma periódica, cada nodo servidor debe enviar información de estado al módulo de monitorización del *switch Web*.

3.1.2.3 Propuesta arquitectónica WC RD/FB

Esta arquitectura es la basada en la asignación dinámica de réplicas y encaminamiento de peticiones con flujo bidireccional (véase Figura 3.7). Su diferencia con respecto a la arquitectura RE/FB es que la asignación de elementos a los nodos se realiza de forma adaptativa.

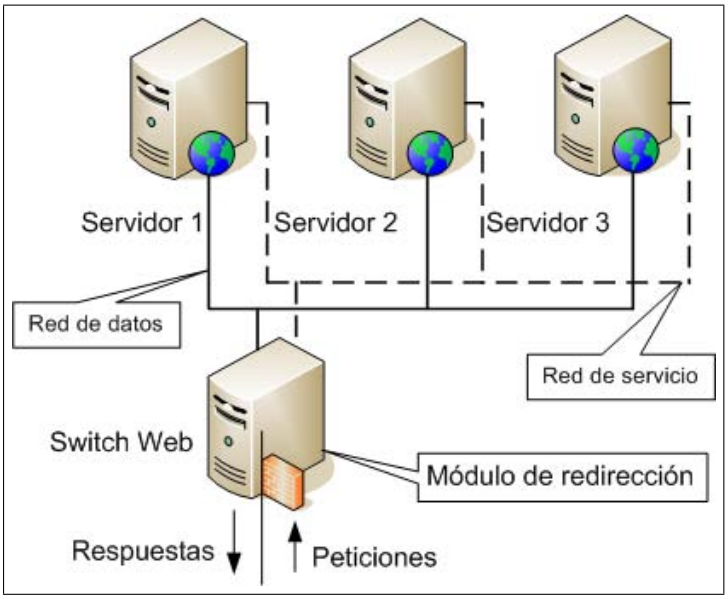

Figura 3.7: Arquitectura RD/FB un *sistema Web basado en cluster*.

La asignación dinámica de réplicas se discute en la sección 4.3. Independientemente de la política de asignación utilizada, la redistribución de réplicas se realiza de forma periódica e implica la transferencia de archivos entre los distintos nodos servidores, así como la actualización de la estructura de directorio del *switch Web*.

Se propone que la arquitectura del sistema utilice dos redes para mejorar las prestaciones del sistema:

- Una red de datos para el envío de peticiones desde el *switch* a los nodos servidores y el envío de respuestas desde los nodos servidores hasta el *switch*.
- Una red de servicio para las comunicaciones necesarias para la redistribución de contenidos entre los nodos servidores y la actualización de las estructuras de datos del servicio de directorio.

Esta arquitectura impone el requisito hardware de que tanto los nodos servidores como el *switch Web* deben estar conectados a dos redes. Por esta razón, se requiere que todos los nodos servidores estén equipados con dos interfaces de red. En el caso del *switch Web*, se hace necesaria la instalación de tres interfaces de red, pues además necesitará otra interfaz de red para su conexión a la red externa.

La estructura interna propuesta para el *switch Web* (véase Figura 3.8) incorpora en este caso los mecanismos necesarios para la adaptación dinámica de réplicas.

Es necesaria la incorporación de un nuevo módulo:

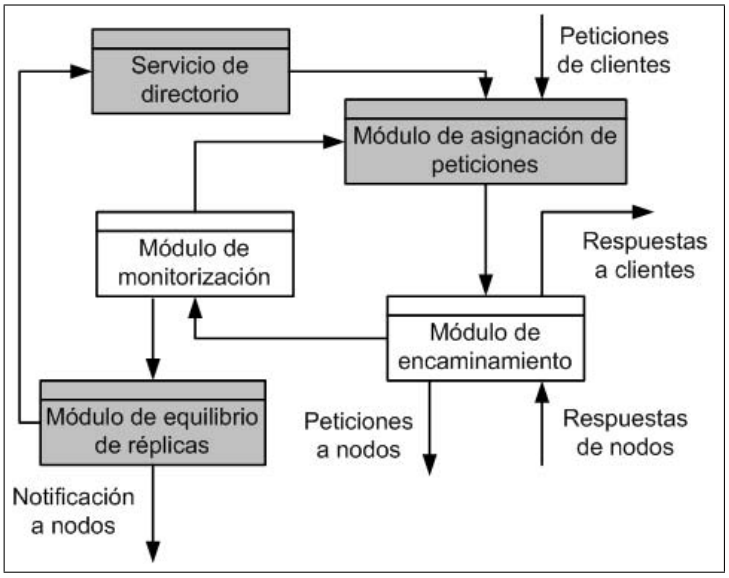

Figura 3.8: Módulos que intervienen en el procesamiento de peticiones en un sistema WC RD/FB.

Módulo de equilibrio de réplicas Este módulo es responsable de determinar las modificaciones en la distribución de réplicas entre los nodos servidores. De forma periódica determina los cambios que deben realizarse en dicha distribución y lo notifica a los nodos servidores y al servicio de directorio.

3.1.2.4 Propuesta arquitectónica WC RD/FU

Esta arquitectura se basa en la asignación dinámica de réplicas y el encaminamiento de peticiones con flujo unidireccional (véase Figura 3.9). Es la arquitectura más compleja, que combina características de las arquitecturas RE/FU y RD/FB.

Se propone que la arquitectura del sistema utilice tres redes:

- Una red de datos para el envío de peticiones desde el *switch* a los nodos servidores.
- Otra red de datos para el envío de respuestas desde los nodos servidores hasta los clientes.
- Una red de servicio para las comunicaciones necesarias para la redistribución de contenidos entre los nodos servidores y la actualización de las estructuras de datos del servicio de directorio.

Por tanto, y por el mismo razonamiento que en la sección anterior, será necesario equipar a los nodos servidores con tres interfaces de red y al *switch Web* con cuatro interfaces de red.

La estructura interna propuesta para el *switch Web* (véase Figura 3.10) tiene algunos aspectos derivados de la solución WC RE/FU y otros aspectos derivados de la solución WC RD/FB. Si bien la estructura general es bastante parecida a la solución WC RD/FB, la gestión del encaminamiento de peticiones se realiza al estilo de la solución WC RE/FU.

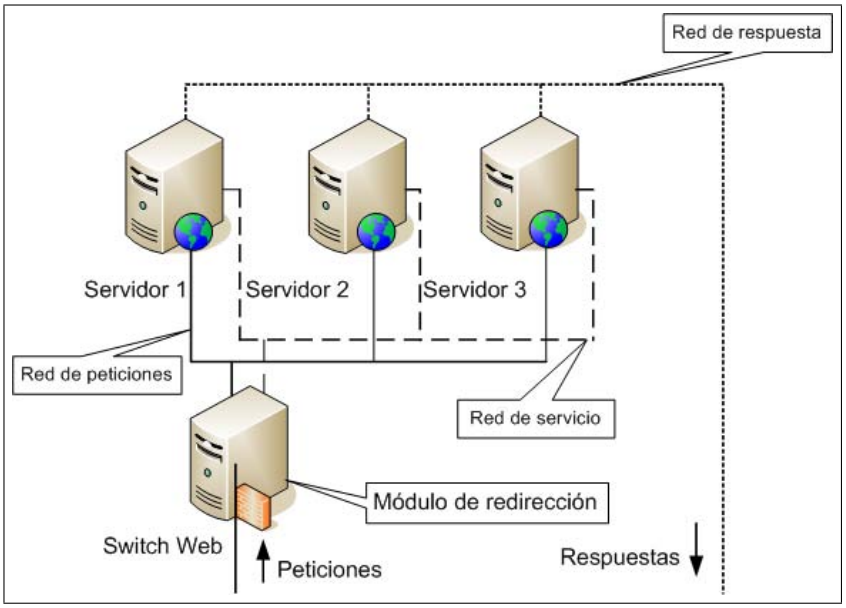

Figura 3.9: Arquitectura WC RD/FU un *sistema Web basado en cluster*.

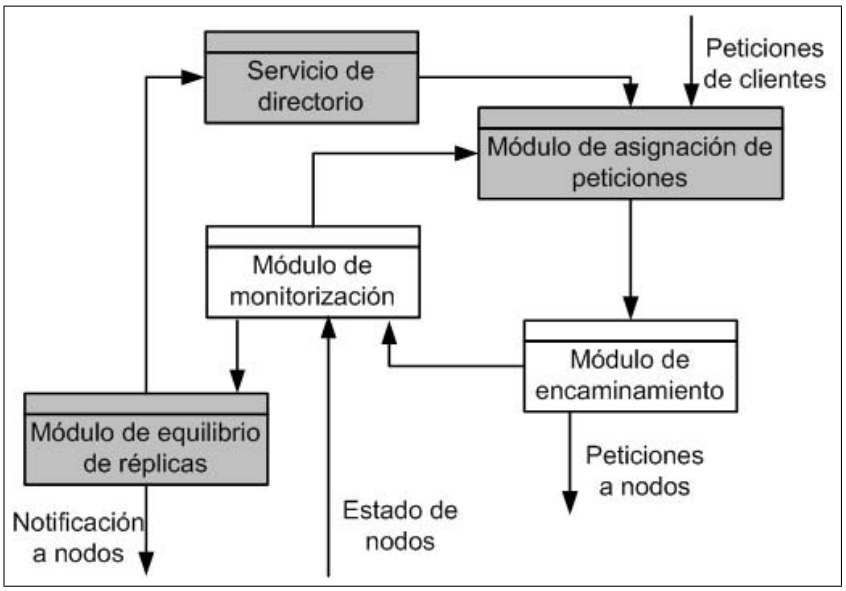

Figura 3.10: Módulos que intervienen en el procesamiento de peticiones en un sistema WC RD/FU.

Por una parte, se hace necesario que el módulo de encaminamiento se implemente en el nivel del núcleo del sistema operativo, al requerirse la modificación de la pila TCP. El encaminamiento solamente se responsabiliza de la transferencia de peticiones a los nodos servidores. Por esta misma razón, el módulo de monitorización debe recibir de los nodos servidores la información de estado de los mismos.

La adaptación dinámica de réplicas se consigue mediante el uso de un módulo de equilibrio

de réplicas que se encarga de determinar periódicamente la asignación de réplicas a nodos servidores.

Además, no es suficiente con que los nodos servidores tengan alojado un servidor Web estándar. De forma periódica, cada nodo servidor debe enviar información de estado al módulo de monitorización del *switch Web*.

3.1.3 Resumen de las propuestas sobre *sistemas Web basados en cluster*

Las arquitecturas de *sistemas Web basados en cluster* tienen, en general, la ventaja de tener un nodo central (el *switch Web*) que se encarga de coordinar el comportamiento del resto de nodos servidores. El uso de mecanismos de encaminamiento unidireccionales suele permitir descargar al *switch* de una parte del procesamiento necesario a cambio de incorporar dos líneas de comunicaciones separadas: una para la llegada de las peticiones y otra para la salida de las respuestas.

Ante la restricción de replicación parcial se impone una limitación sobre el conjunto de políticas de asignación de peticiones que se pueden seguir, ya que solamente son admisibles aquellas políticas que se basan en un servicio de directorio. No obstante, trabajos previos han demostrado que se pueden construir servicios de directorio altamente escalables y que imponen un retardo mínimo.

Cuando es necesaria una adaptación de la replicación al estado del sistema, la complejidad adicional introducida no es excesiva gracias a la existencia de un único punto central: el *switch Web*.

No obstante, la principal fuente de ventajas de esta familia de arquitecturas (el *switch Web*) es también su principal inconveniente. Por una parte, la existencia de dicho punto central supone un cuello de botella en el procesamiento de peticiones. Cuando la utilización del *switch* llega a su punto de saturación no es posible mejorar el rendimiento del sistema. Por otra parte, el *switch* supone un punto de fallo único. Si se produce una avería en dicho elemento el resto del sistema dejará de funcionar. Es decir, esta familia de arquitecturas ofrece una nula tolerancia a fallos.

La selección de una propuesta arquitectónica de entre las cuatro presentadas en esta sección depende las necesidades del servicio que se desee ofrecer y de la infraestructura disponible.

La propuesta más completa es la propuesta WC RD/FU que permite la adaptación dinámica de la distribución de réplicas entre los nodos y que utiliza flujo unidireccional (los mensajes de vuelta no vuelven a pasar por el *switch Web*). Esta alternativa requiere el uso de dos redes separadas para comunicar el *switch* con los nodos, así como de una separación entre la red de entrada y la red de salida al *cluster*. Además la solución implica que tanto el *switch* como los nodos servidores deben ejecutar una versión modificada del sistema operativo.

3.2 Análisis de *clusters Web virtuales*

En un *cluster Web virtual* todas las peticiones llegan a todos los nodos del *cluster*. Cada nodo servidor es responsable de realizar un filtrado de peticiones de forma que pueda identificar las peticiones que debe servir y descarte el resto.

Este mecanismo es de fácil implantación en el caso de replicación total. En dicho caso se puede realizar un filtrado por transformación de clave (*hashing*) sobre algunos campos de la petición HTTP.

También es fácil la implantación en el caso de distribución total. En tal caso, cada elemento se encuentra alojado en un único nodo, por lo que cada petición puede ser servida por un único nodo y todos los demás descartarán la petición.

No obstante, en el caso de replicación parcial existen varios nodos que alojan un determinado elemento. Para que fuese posible la utilización de un *cluster Web virtual* en presencia de replicación parcial sería necesario que un nodo servidor pudiese decidir si debe o no procesar una petición HTTP de un elemento que también puede estar alojado en otros nodos.

Dado que cada elemento se puede alojar en un conjunto distinto de nodos servidores, sería necesario disponer de una función *hash* distinta para cada elemento del sitio Web. Además, el conjunto de posibles resultados debería ser distinto para cada elemento. Es decir que si el elemento e_1 está alojado en los servidores s_2, s_6 y s_{13} la función *hash* solamente debe poder generar como resultado los valores 2, 6 y 13. El diseño de esta familia de funciones *hash* plantea problemas, ya que todas ellas (una para cada elemento) deberían generarse de forma automática

Otro obstáculo reside en la necesidad de incluir la URL solicitada en el proceso de filtrado. La URL es una cadena de texto relativamente larga que habría que analizar para determinar que función *hash* debería aplicarse. Además, este análisis debería realizarse para todas la peticiones. Dicho análisis es mucho más costoso que la simple aplicación de una función *hash* sencilla al último byte de una dirección IP.

Dado que el uso de réplicas parciales es el objetivo principal de esta tesis, se ha decidido no considerar el uso de soluciones basadas en *clusters Web virtuales*.

3.3 Análisis de *sistemas Web distribuidos*

En un *sistema Web distribuido* cualquier petición puede llegar a cualquier nodo del sistema, gracias al uso de DNS dinámico. No obstante no existe ninguna forma de asegurar que las peticiones se distribuirán entre los nodos de una determinada manera. Es más, es probable que las peticiones se distribuyan con un cierto grado de desequilibrio hacia algún nodo. Cada nodo puede realizar la redirección de algunas peticiones basándose en algún mecanismo de encaminamiento de peticiones para *sistemas Web distribuidos*.

3.3.1 Restricciones sobre la arquitectura

Teniendo en cuenta los objetivos de esta tesis, es necesario considerar el impacto que tienen las restricciones establecidas en dichos objetivos sobre un *sistema Web distribuido*. Por una parte, la replicación parcial impide que se asuma la existencia de todos los elementos en todos los nodos servidores. Por otra parte, la asignación adaptativa de contenidos hace que sea necesario transferir contenidos entre los distintos nodos.

3.3.1.1 Replicación parcial

Al estar los contenidos replicados de forma parcial, cada elemento e_i se encuentra almacenado en un subconjunto de nodos. La asignación de elementos a los nodos se puede realizar de forma

estática o dinámica. Cuando llega una petición a un nodo, el comportamiento dependerá de si el elemento solicitado se encuentra o no alojado en dicho nodo:

- Si el elemento solicitado se encuentra alojado en el nodo receptor de la petición, dicho nodo puede servirlo directamente o realizar la redirección (en caso de encontrarse muy cargado) a otro nodo que también contenga el elemento solicitado.
- Si el elemento solicitado no se encuentra alojado en el nodo receptor de la petición, el nodo puede redirigir la petición a otro nodo que contenga el elemento solicitado.

La replicación parcial implica que un determinado elemento puede no estar presente en un nodo servidor. Esto hace imposible el uso de un encaminamiento de peticiones basado en reescritura de URL, ya que dicho mecanismo asume que, al menos, los elementos primarios (los elementos que hacen referencia a otros elementos, como los archivos HTML) están replicados en todos los nodos servidores.

Como consecuencia, el encaminamiento de peticiones solamente se puede realizar mediante triangulación en el nivel HTTP o mediante redirección HTTP. Es importante tener en cuenta que la triangulación requiere la reescritura de los paquetes que forman las respuestas HTTP, mientras que en el caso de la redirección HTTP esto no es necesario.

Cada nodo servidor debe ser capaz de:

1. Determinar de una forma eficiente si tiene alojado un determinado elemento.
2. Determinar de una forma eficiente en qué nodos se encuentra alojado un determinado elemento.
3. Ser capaz de seleccionar un nodo al que redirigir una petición basándose en la información de carga global.
4. Realizar la recogida de su información de carga.
5. Notificar periódicamente de su información de carga al resto de nodos.

La determinación de la información de alojamiento de los elementos, se puede realizar mediante el uso de una estructura de datos basadas en la formalización de URL y el uso de tablas *hash* multinivel. En cuanto a la recogida y utilización de la información de carga, se puede usar información global de utilización del nodo servidor.

3.3.1.2 Asignación de contenidos adaptativa

Como ya se ha comentado anteriormente, la asignación estática de contenidos no tiene ningún impacto sobre la arquitectura del sistema, ya que tanto el número de réplicas de cada elemento como su asignación a nodos se realiza en una fase inicial y no se modifica a lo largo del tiempo. En este caso no existe ninguna adaptación a las condiciones de carga.

En el caso de asignación adaptativa a las condiciones de carga, la información necesaria debe ser recogida por cada nodo servidor y debe producirse intercambio de información entre los distintos nodos del sistema.

Un *sistema Web distribuido* tiene una arquitectura totalmente distribuida (no existe ningún nodo central), y por tanto cualquier reasignación de contenidos debe pasar previamente por el intercambio de información de carga entre los distintos nodos del sistema. En una fase posterior se puede efectuar la redistribución de contenidos entre los nodos.

Para que que el efecto de las comunicaciones debidas a la asignación adaptativa de contenidos tenga un impacto mínimo sobre el rendimiento del sistema, se puede utilizar una red de servicio para dichas comunicaciones.

3.3.2 Propuesta de soluciones arquitectónicas

En esta sección se presentan las distintas alternativas de arquitectura para *sistemas Web distribuidos* propuestas en esta tesis.

Las variaciones en las distintas arquitecturas presentadas son consecuencia de las posibilidades de asignación de réplicas. En este sentido se distingue entre las necesidades arquitectónicas de las asignación estática de réplicas y la asignación dinámica de réplicas.

En esta familia de arquitecturas no se considera la diferencia entre flujo unidireccional y bidireccional (tal y como se ha hecho en el caso de *sistemas Web basados en cluster*). Esto se debe a que en este caso, las peticiones llegan al nodo servidor, que debe enviar la respuesta al cliente. Esta simplificación se debe al hecho de que todos los nodos servidores de un *sistema Web distribuido* tienen un dirección IP distinta y visible desde el exterior.

Las variaciones en cuanto al tipo de asignación de réplicas dan lugar a dos posibles variantes arquitectónicas:

WD RE *Sistema Web distribuido* con asignación estática de réplicas.

WD RD *Sistema Web distribuido* con asignación dinámica de réplicas.

3.3.2.1 Propuesta arquitectónica WD RE

La arquitectura más básica para el caso de un *sistema Web distribuido* es la que se basa en la asignación estática de réplicas (véase Figura 3.11). En esta arquitectura el único requisito es que todos los nodos servidores tengan una dirección IP pública y visible desde el exterior. Además, de forma opcional, se puede utilizar un primer nivel de encaminamiento mediante DNS dinámico.

En cada nodo servidor existen varios módulos (véase Figura 3.12) que se encargan del procesamiento previo de las peticiones:

Filtro de existencia Determina si el elemento solicitado se encuentra en el nodo servidor. En caso de no encontrarse se pasa la petición al filtro de estado para la selección del nodo servidor al que debe encaminarse la petición.

Filtro de estado Determina si la petición la debe servir el nodo actual o debe efectuarse una redirección a otro nodo servidor. Para tomar la decisión tiene en cuenta las condiciones de estado de cada nodo. Si la petición la debe servir el nodo actual, ésta se traspasa al módulo de servicio de peticiones. En otro caso, se traspasa al módulo de encaminamiento.

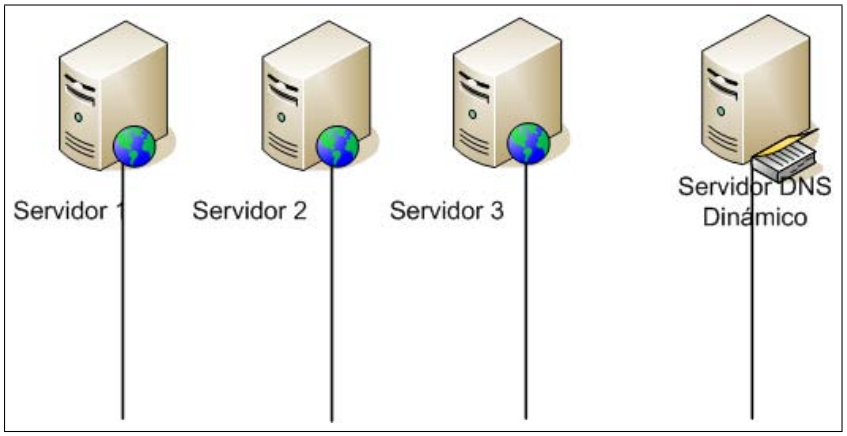

Figura 3.11: Arquitectura RE para un *sistema Web distribuido*.

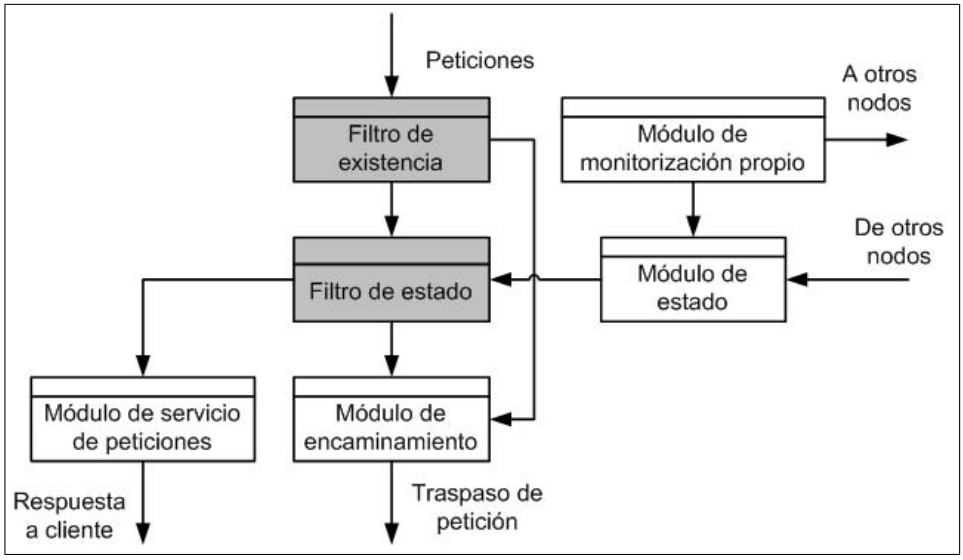

Figura 3.12: Módulos que intervienen en el procesamiento de peticiones en un nodo de un *sistema Web distribuido* RE.

Módulo de encaminamiento Realiza el encaminamiento de la petición a otro nodo utilizando el mecanismo de encaminamiento activo (triangulación o redirección).

Módulo de servicio de peticiones Realiza el servicio de la petición HTTP. Este módulo es normalmente el propio servidor Web.

Módulo de monitorización propio Recoge los datos de utilización del nodo servidor y periódicamente notifica al resto de nodos de la información recogida.

Módulo de estado Recibe periódicamente la información de estado de todos los módulos y realiza la fusión de la misma para su uso por el filtro de estado.

3.3.2.2 Propuesta arquitectónica WD RD

Esta arquitectura se basa en la asignación dinámica de réplicas (véase Figura 3.13). En esta arquitectura los nodos servidores deben poder comunicarse entre si, no solamente para intercambiar información de estado, sino que también deben poder realizar intercambio de archivos para que se pueda producir una adaptación de las réplicas al estado del sistema. Este mayor volumen de intercambio de información justifica el uso de una red de servicio.

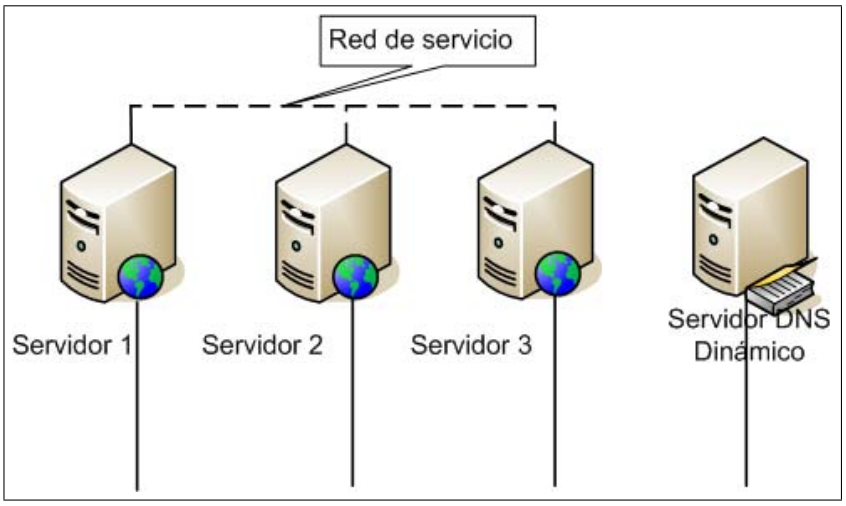

Figura 3.13: Arquitectura RD para un *sistema Web distribuido*.

En cada nodo servidor es necesario incorporar dos nuevos módulos (véase Figura 3.14) que se encarguen de la gestión de las réplicas:

Módulo de equilibrio de réplicas Se encarga de observar las peticiones que llegan al nodo y, en colaboración con el resto de nodos servidores, determinar qué elementos necesitan incrementar o decrementar el número de réplicas.

Módulo de intercambio de réplicas Se encarga de realizar las eliminaciones de réplicas y las transmisiones de réplicas a otros nodos servidores.

3.3.3 Resumen de las propuestas sobre *sistemas Web distribuidos*

Los *sistemas Web distribuidos* presentan, en general, una arquitectura muy flexible que permite incluso que el sistema se pueda distribuir geográficamente. Como contrapartida, suelen tener la necesidad de utilizar mecanismos de coordinación más complejos al tratarse de sistemas totalmente distribuidos y simétricos. Esto implica que las decisiones deben ser negociadas entre todos los nodos. Además se hace necesario que todos los nodos servidores que participan en el sistema posean una dirección IP pública, lo que puede limitar su escalabilidad.

Cuando se impone la restricción de replicación parcial de contenidos puede ocurrir que muchas peticiones lleguen inicialmente a un nodo servidor que no contenga el recurso solicitado y que deba, por tanto, producirse una redirección. Si, además, se impone la restricción de replicación adaptativa, se hace necesario el uso de una red de servicio separada que elimina

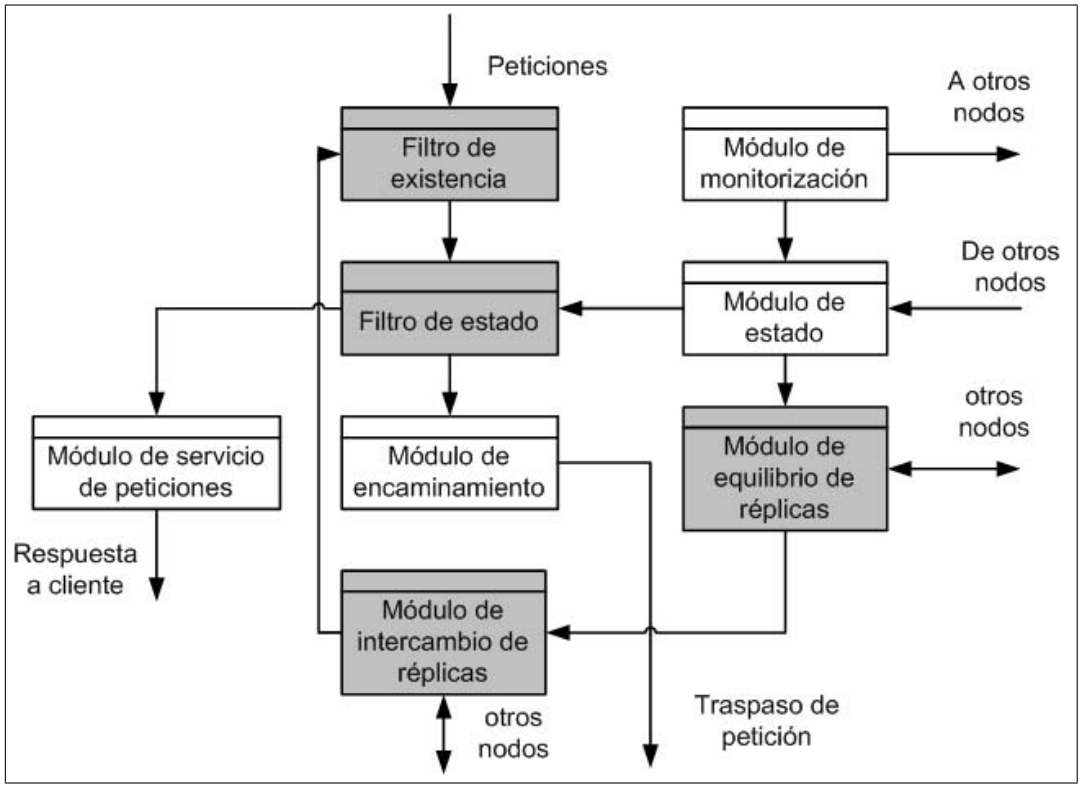

Figura 3.14: Módulos que intervienen en el procesamiento de peticiones en un nodo de un *sistema Web distribuido* RD.

la ventaja de la distribución geográfica, ya que en este caso sería necesario tener a todos los nodos servidores en una misma localización geográfica.

Si bien la replicación parcial y adaptativa puede no ser ventajosa en el uso de los *sistemas Web distribuidos* algunas de las ideas en las que se basan se pueden utilizar para una arquitectura mejorada de *clusters Web* que ofrezca una mayor tolerancia a fallos y disponibilidad.

La selección de una propuesta arquitectónica de entre las dos presentadas en esta sección depende las necesidades del servicio que se desee ofrecer y de la infraestructura disponible.

La propuesta más completa es la propuesta WD RD/FU que permite la adaptación dinámica de la distribución de réplicas entre los nodos. Esta alternativa requiere el uso de una red de servicio separada de la red de datos. No obstante, debe tenerse en cuenta que la incorporación de esta segunda red limita la distribución geográfica de los nodos.

3.4 Comparación de arquitecturas propuestas

En las secciones anteriores se ha realizado una propuesta de alternativas arquitectónicas para los *sistemas Web basados en cluster* (véase 3.1.2) y los *sistemas Web distribuidos* (véase 3.3.2). Así mismo, se ha justificado la imposibilidad de compatibilizar de una forma sencilla los objetivos de esta tesis con una arquitectura basada en un *cluster Web virtual* (véase 3.2).

La tabla 3.1 realiza una comparación de las distintas arquitecturas propuestas en las secciones anteriores.

	WC				WD	
	RE		RD		RE	RD
	FB	FU	FB	FU		
Asignación de réplicas E - Estática D - Dinámica	E	E	D	D	E	D
Flujo de peticiones U - Unidireccional B - Bidireccional	B	U	B	U	-	-
Interfaces de red por nodo servidor	1	2	2	3	1	2
Interfaces de red por *switch Web*	2	3	3	3	-	-
Modificación de Sistema Operativo en nodo servidor N - No, S - Si	N	S	N	S	S/N	S/N
Modificación de Sistema Operativo en *switch Web* N - No, S - Si,	S/N	S	S/N	S	-	-
Punto de fallo único N - No, S - Si	S	S	S	S	N	N
IP pública de nodo servidor N - No, S - Si	N	N	N	N	S	S

Tabla 3.1: Comparación de características de arquitecturas propuestas para *sistemas Web basados en cluster* y *sistemas Web distribuidos*.

Las arquitecturas de *sistemas Web basados en cluster* presentan algunos inconvenientes, como la necesidad de modificar el sistema operativo o su vulnerabilidad ante cierto tipo de fallos (un fallo en el *switch Web* afecta a todo el sistema). No obstante, estos sistemas pueden escalar fácilmente, siendo sencilla la incorporación de nuevos nodos servidores.

Los *sistemas Web distribuidos* son menos vulnerables al no existir un punto de fallo único. Sin embargo, la necesidad de que todos los nodos servidores dispongan de una dirección IP pública plantea un problema de escalabilidad (se debe disponer de una dirección IP pública distinta para cada nodo) y además puede no ser compatible con las directivas de seguridad de ciertas organizaciones.

Por esta última razón, en esta tesis se opta por los principios de las arquitecturas de *sistemas Web basados en cluster*. La vulnerabilidad de dichas arquitecturas ante cierto tipo de fallos se puede mitigar tomando algunas de las ideas utilizadas en los *sistemas Web distribuidos*. Esta combinación da lugar a la arquitectura propuesta en esta tesis: el ***cluster Web con switch distribuido***, que se presenta en la siguiente sección.

3.5 Nueva propuesta: *Cluster Web con switch distribuido*

Teniendo en cuenta lo expuesto en la sección anterior, la arquitectura propuesta para un *cluster Web con switch distribuido* (véase Figura 3.15) utiliza varios *switches Web* para realizar las asignaciones de peticiones. De esta forma se elimina el punto central de fallo del sistema. Para distribuir, dentro de lo posible, la llegada de peticiones a los distintos *switches* se puede utilizar como primer mecanismo, el encaminamiento mediante DNS dinámico. No obstante, no debe olvidarse la falta de garantías de dicho mecanismo (véase 2.2.3).

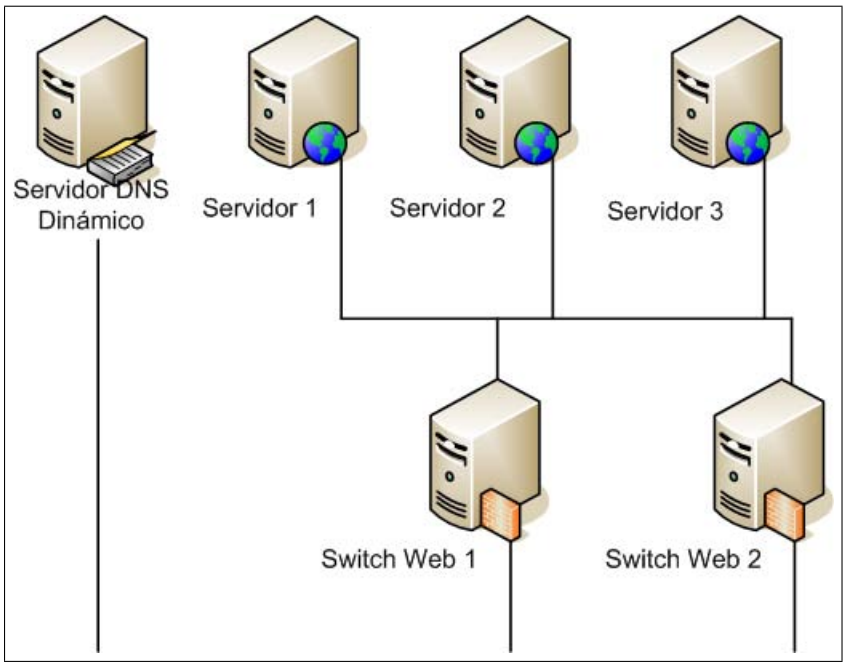

Figura 3.15: Arquitectura general de un *cluster Web con switch distribuido*.

Cuando una petición llega a un *switch*, éste puede pasarla a uno de los nodos servidores para su procesamiento. En los casos en que su estado aconseje traspasar la petición a otro *switch Web*, se puede utilizar un mecanismo de encaminamiento de *sistemas Web distribuidos* para traspasar la petición.

3.5.1 Restricciones sobre la arquitectura propuesta

Como en secciones anteriores, en esta sección se estudia el efecto de las dos restricciones establecidas en esta tesis: la replicación parcial de contenidos y la asignación adaptativa de réplicas.

3.5.1.1 Replicación parcial

Como ya se ha expresado anteriormente, la replicación parcial elimina la garantía de que un determinado elemento pueda estar alojado en todos los nodos servidores. La asignación de réplicas se puede realizar de forma estática o dinámica.

Independientemente del tipo de asignación de réplicas utilizada y de la correspondiente política de asignación, cada *switch Web* es el responsable de seleccionar un nodo servidor para cada petición recibida, de forma que en dicho nodo seleccionado resida una copia del elemento solicitado. Así mismo, el *switch* es responsable de realizar el encaminamiento de la petición. Al igual que en el caso de los *sistemas Web basados en cluster*, es posible utilizar un servicio de directorio basado en formalización de URL y tablas *hash* multinivel para acelerar la determinación del nodo servidor que debe recibir la petición.

Al igual que en los *sistemas Web basados en cluster*, el encaminamiento puede realizarse mediante pasarela TCP, empalme TCP o cesión TCP.

3.5.1.2 Asignación de contenidos adaptativa

En un *cluster Web con switch distribuido* la información del servicio de directorio se encuentra replicada en cada *switch*. Esto permite que el tiempo necesario para realizar el encaminamiento de cada petición sea mínimo.

Cuando es necesario adaptar la asignación de contenidos al estado del sistema, la información del servicio de directorio debe actualizarse después de cada adaptación. Esto requiere la actualización de la información en todos los nodos que actúan como *switch*.

Por otra parte, la información necesaria para realizar la adaptación de contenidos no se encuentra en este caso centralizada en un único punto ya que existen varios puntos de entrada al sistema. Por ello, antes de la adaptación es necesario que se efectúe una fusión de la información que posee cada *switch*.

3.5.2 Propuesta de soluciones arquitectónicas

En esta sección se presentan las distintas alternativas de arquitectura para *clusters Web con switch distribuido*.

Como en casos anteriores, las variaciones en las distintas arquitecturas presentadas son consecuencia de dos propiedades:

Tipo de asignación de réplicas Se distingue entre las necesidades arquitectónicas de la asignación estática de réplicas y la asignación dinámica de réplicas.

Tipo de flujo de encaminamiento Se distingue entre las necesidades derivadas del encaminamiento bidireccional y el encaminamiento unidireccional.

Estas variaciones dan lugar a cuatro posibles variantes arquitectónicas:

SD RE/FB *Cluster Web con switch distribuido* con asignación estática de réplicas y encaminamiento con flujo bidireccional.

SD RE/FU *Cluster Web con switch distribuido* con asignación estática de réplicas y encaminamiento con flujo unidireccional.

SD RD/FB *Cluster Web con switch distribuido* con asignación dinámica de réplicas y encaminamiento con flujo bidireccional.

SD RD/FU *Cluster Web con switch distribuido* con asignación dinámica de réplicas y encaminamiento con flujo unidireccional.

3.5.2.1 Propuesta arquitectónica SD RE/FB

La arquitectura más básica para el caso de un *cluster Web con switch distribuido* es la correspondiente a la asignación estática de réplicas y encaminamiento de peticiones con flujo bidireccional.

La estructura de módulos de un *switch Web* (véase Figura 3.16) es más compleja, incluso para el caso más simple de esta familia de arquitecturas, de lo que lo son en otras familias de arquitecturas. Esto se debe al hecho de que esta arquitectura incluye aspectos de los *clusters Web* y de los *sistemas Web distribuidos*.

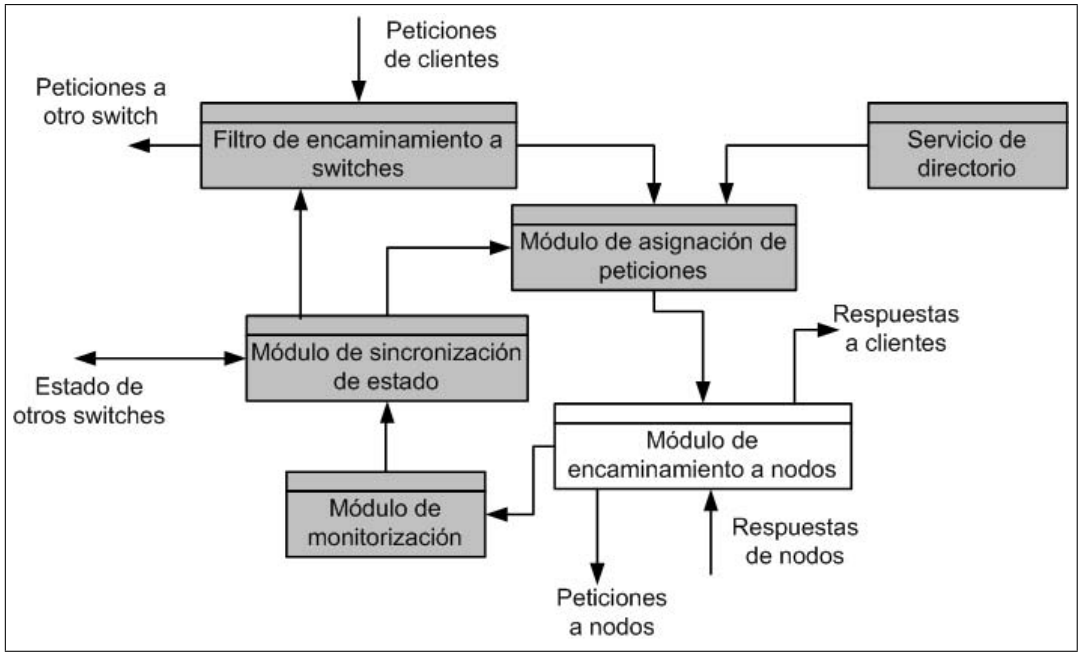

Figura 3.16: Módulos que intervienen en el procesamiento de peticiones en un sistema SD RE/FB.

Cada *switch* realiza una monitorización de su estado e intercambia su información de estado con la de los otros *switches*. De esta forma cada *switch* tiene la información necesaria para decidir cuándo debe traspasar una petición a otro *switch* y cuando puede servirla directamente.

Debe tenerse en cuenta que el traspaso de una petición a otro *switch* tiene un menor coste que el mantenimiento de la petición en el *switch* actual. Esto se debe a que el traspaso a otro *switch* se puede realizar sin tener en cuenta el contenido de la petición, lo que permite que se pueda realizar en el nivel TCP de la pila de protocolos y se pueda integrar fácilmente en el núcleo del sistema operativo. Sin embargo, la política de asignación de peticiones debe ser dependiente del contenido y esto requiere un mayor tiempo de procesamiento.

Cuando un *switch* decide encargarse del servicio de una petición realiza la selección de un nodo servidor de acuerdo con la política de asignación de peticiones y le envía la petición. Al igual que ocurre con los *clusters Web* es posible utilizar tanto encaminamiento mediante pasarela TCP como mediante empalme TCP. Además se puede reducir la latencia asociada a

la apertura y cierre de conexiones manteniendo un conjunto de conexiones permanentemente abiertas con cada nodo servidor.

A continuación se presenta una descripción de las funciones de cada módulo:

Filtro de encaminamiento a *switches* Se encarga de decidir si la petición recibida debe ser procesada por el *switch* actual o debe traspasarse a otro *switch*. Su fuente de información principal es la información de estado de los distintos *switches* que se sincronizan mediante el *módulo de sincronización de estado*.

Módulo de asignación de peticiones Es el módulo encargado de recibir las peticiones que han pasado con éxito el *filtro de encaminamiento a switches* y, tras analizarlas, asignar el servicio de cada petición a un nodo servidor. La asignación de peticiones se realiza de acuerdo con la política de asignación de peticiones vigente. En cualquier caso, la asignación puede necesitar tener en cuenta la asignación de réplicas a nodos servidores (suministrada por el *servicio de directorio*) y la información de estado del sistema (suministrada por el *módulo de sincronización de estado*).

Módulo de encaminamiento a nodos Recibe las peticiones que han sido asignadas por el *módulo de asignación de peticiones*, con la correspondiente asignación del nodo servidor que debe procesarlas, y realiza el encaminamiento bidireccional. Esto incluye el envío de peticiones a los nodos servidores, así como la recepción de respuestas de los nodos servidores y su retransmisión a los clientes.

Servicio de directorio Mantiene la información de asignación de réplicas a nodos servidores y permite la consulta de dicha información por parte del *módulo de asignación de peticiones*.

Módulo de monitorización Recibe del *módulo de encaminamiento a nodos* la información de las peticiones enviadas a los nodos servidores así como de las respuestas recibidas. Con esta información construye la información de estado que suministra al *módulo de sincronización de estado*.

Módulo de sincronización de estado Recibe la información de estado suministrada por el *módulo de monitorización* y se encarga de sincronizar la información de estado del *switch* con la de los otros *switches*. La información global de estado que mantiene, sirve de entrada al *módulo encaminamiento de switches* y al *módulo de asignación de peticiones*.

3.5.2.2 Propuesta arquitectónica SD RE/FU

Esta arquitectura (véase Figura 3.17) se basa en la asignación estática de réplicas y el encaminamiento de peticiones con flujo unidireccional. Su diferencia con la arquitectura SD RE/FB es que el encaminamiento de peticiones sigue un flujo unidireccional en vez de bidireccional.

En este caso es necesario que cada nodo servidor posea dos interfaces de red:

- Una interfaz de red para la conexión con la red que comunica los nodos servidores con los *switches Web*. Esta es la interfaz de red que se usa para la recepción de las peticiones enviadas por los *switches Web* al nodo servidor.

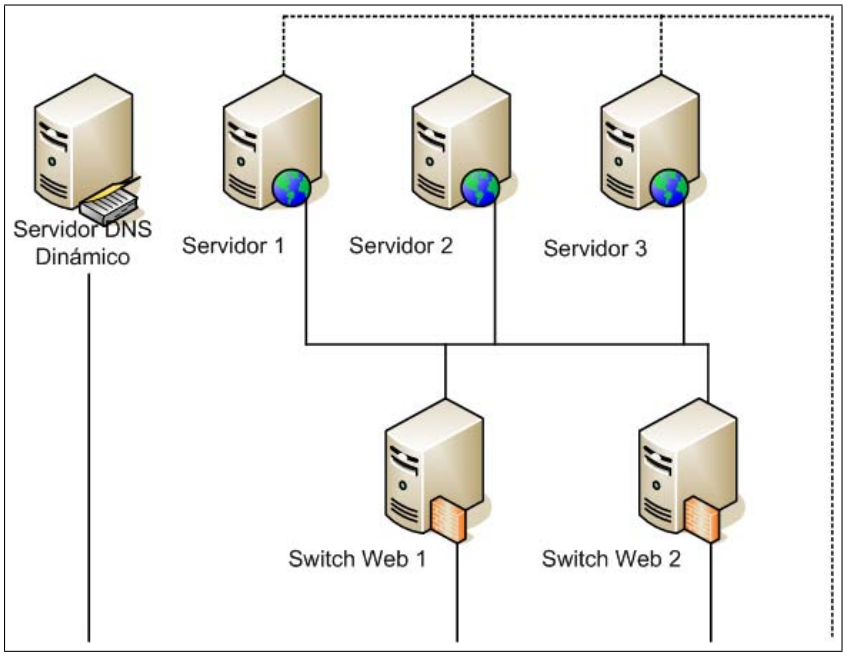

Figura 3.17: Arquitectura RE/FU de un *cluster Web con switch distribuido.*

- Una interfaz de red para la conexión con el canal de salida. Esta es la interfaz de red usada para el envío de las respuestas HTTP a los clientes, que se realiza de forma directa y sin pasar por el *switch Web.*

Además, se hace necesario que tanto el *switch Web* como los nodos servidores ejecuten una versión modificada del sistema operativo que permita la implantación del protocolo de cesión TCP.

La estructura interna para cada *switch Web* (véase Figura 3.18) es similar a la de un *switch Web* SD RE/FB con algunas variaciones.

La principal diferencia en la estructura interna de los *switches Web* con respecto a la arquitectura WC RE/FB es que en este caso el módulo de encaminamiento tiene que implementarse necesariamente en el nivel del núcleo del sistema operativo y que requiere la modificación de la pila TCP. Además dicho módulo de encaminamiento solamente se responsabiliza del encaminamiento de las peticiones, y no de las respuestas. Esto se debe a que las respuestas las envían los nodos servidores directamente a los clientes.

Este último hecho hace que el módulo de monitorización no pueda obtener toda la información directamente del resto de módulos del *switch Web* y que tenga que obtener información de estado enviada por los nodos servidores, ya que las respuestas no pasan por el *switch.*

3.5.2.3 Propuesta arquitectónica SD RD/FB

Esta arquitectura (véase Figura 3.19) se basa en la asignación dinámica de réplicas y el encaminamiento de peticiones con flujo bidireccional. Su diferencia con la arquitectura SD RE/FU es que la asignación de elementos a los nodos se realiza de forma adaptativa.

La arquitectura del sistema utiliza en este caso dos redes:

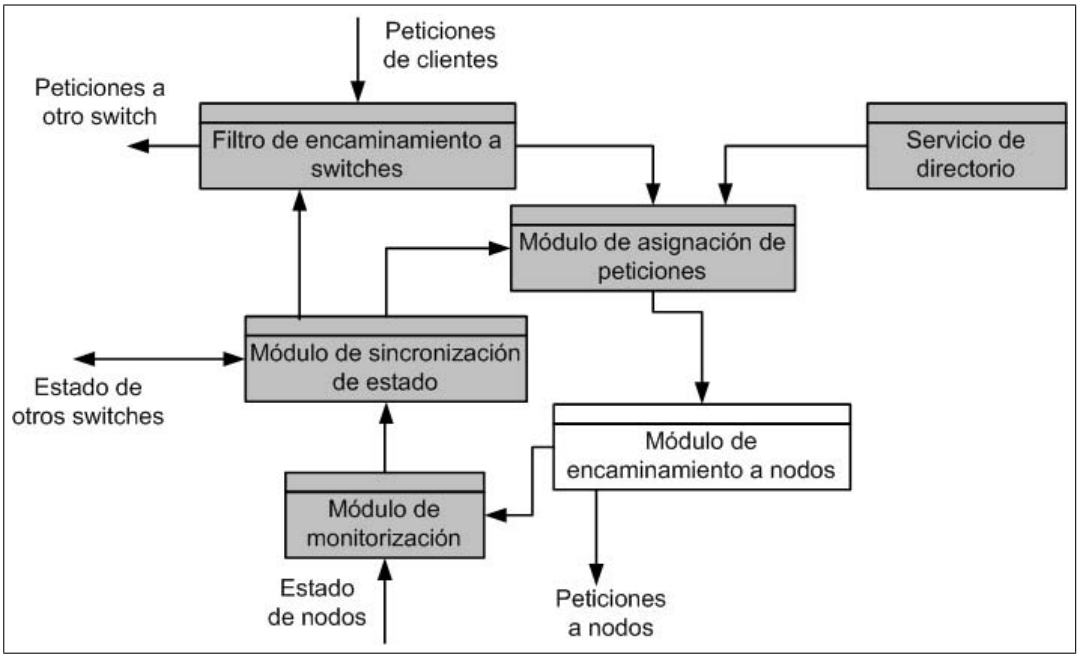

Figura 3.18: Módulos que intervienen en el procesamiento de peticiones en un sistema SD RE/FU.

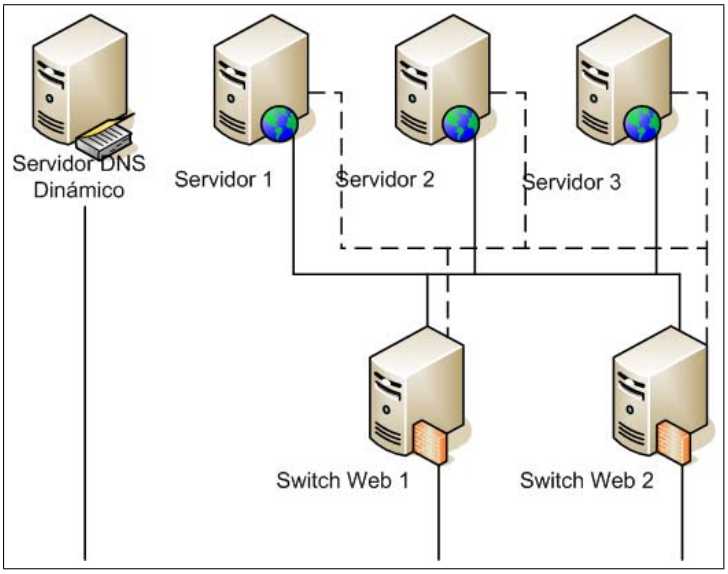

Figura 3.19: Arquitectura RD/FB de un *cluster Web con switch distribuido*.

- Una red de datos para el envío de peticiones desde los *switches* a los nodos servidores y el envío de respuestas desde los nodos servidores hasta los *switches*.

- Una red de servicio para las comunicaciones necesarias para la redistribución de contenidos entre los nodos servidores y la actualización de las estructuras de datos del

servicio de directorio.

Esta arquitectura impone el requisito hardware de que tanto los nodos servidores como los *switches Web* deben estar conectados a dos redes. Por esta razón, se requiere que todos los nodos servidores estén equipados con dos interfaces de red. En el caso del *switch Web*, se hace necesaria la instalación de tres interfaces de red, pues además necesitará otra interfaz de red para su conexión a la red externa.

La estructura interna de los *switches Web* (véase Figura 3.20) incorpora en este caso los mecanismos necesarios para la adaptación dinámica de réplicas.

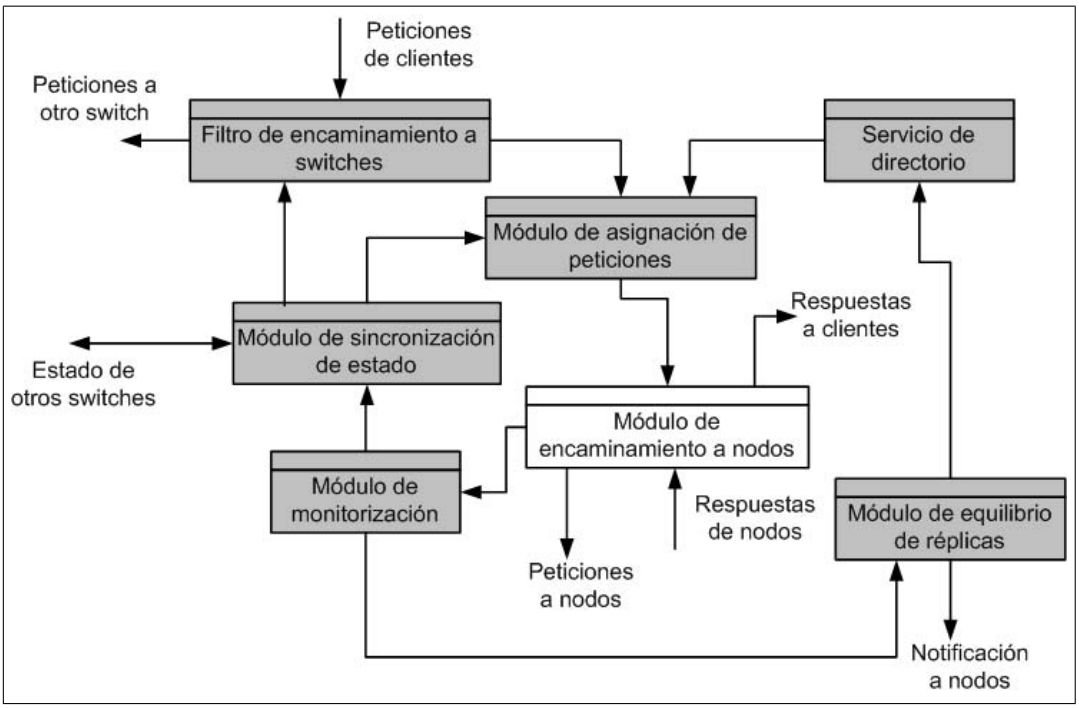

Figura 3.20: Módulos que intervienen en el procesamiento de peticiones en un sistema SD RD/FB.

En este caso es necesario la incorporación de un módulo adicional:

Módulo de equilibrio de réplicas Este módulo se ejecuta de forma periódica en un único *switch Web*. El módulo es responsable de determinar las modificaciones en la distribución de réplicas entre los nodos servidores. Cuando el módulo determina que es necesario realizar cambios en la distribución, lo notifica a los nodos servidores y a los *servicios de directorio* de todos los *switches Web*.

Es posible utilizar distintas estrategias para asegurar que en cada período un único *switch* ejecuta el módulo de equilibrio de réplicas. Una estrategia sencilla es seleccionar uno de los *switches* como *switch* primario y hacer que el módulo se ejecute siempre allí. También se puede decidir ejecutar el módulo cada vez en un *switch* distinto con algún criterio de reparto (como la asignación cíclica o la asignación al *switch* menos cargado).

3.5.2.4 Propuesta arquitectónica SD RD/FU

Esta arquitectura (véase Figura 3.21) se basa en la asignación dinámica de réplicas y el encaminamiento de peticiones con flujo unidireccional. Es la arquitectura más compleja y combina características de las arquitecturas SD RE/FU y SD RD/FB.

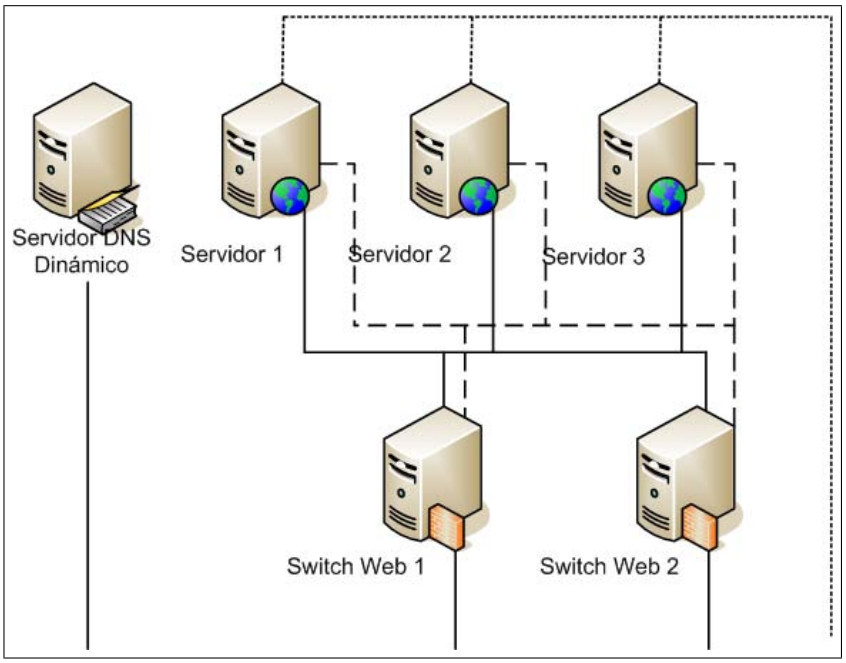

Figura 3.21: Arquitectura RD/FU de un *cluster Web con switch distribuido.*

La arquitectura del sistema utiliza tres redes:

- Una red de datos para el envío de peticiones desde los *switches* a los nodos servidores.
- Otra red de datos para el envío de respuestas desde los nodos servidores hasta los clientes.
- Una red de servicio para las comunicaciones necesarias para la redistribución de contenidos entre los nodos servidores y la actualización de las estructuras de datos de los servicios de directorio.

Por tanto, y por el mismo razonamiento que en la sección anterior, será necesario equipar a los nodos servidores con tres interfaces de red y a los *switches Web* con cuatro interfaces de red.

La estructura interna del *switch Web* (véase Figura 3.22) tiene algunos aspectos derivados de la solución SD RE/FU y otros aspectos derivados de la solución SD RD/FB.

Por una parte, se hace necesario que el *módulo de encaminamiento* se implemente en el nivel del núcleo del sistema operativo, al requerirse la modificación de la pila TCP. El encaminamiento solamente se responsabiliza de la transferencia de peticiones a los nodos servidores. Por esta misma razón, el *módulo de monitorización* debe recibir de los nodos servidores la información de estado de los mismos.

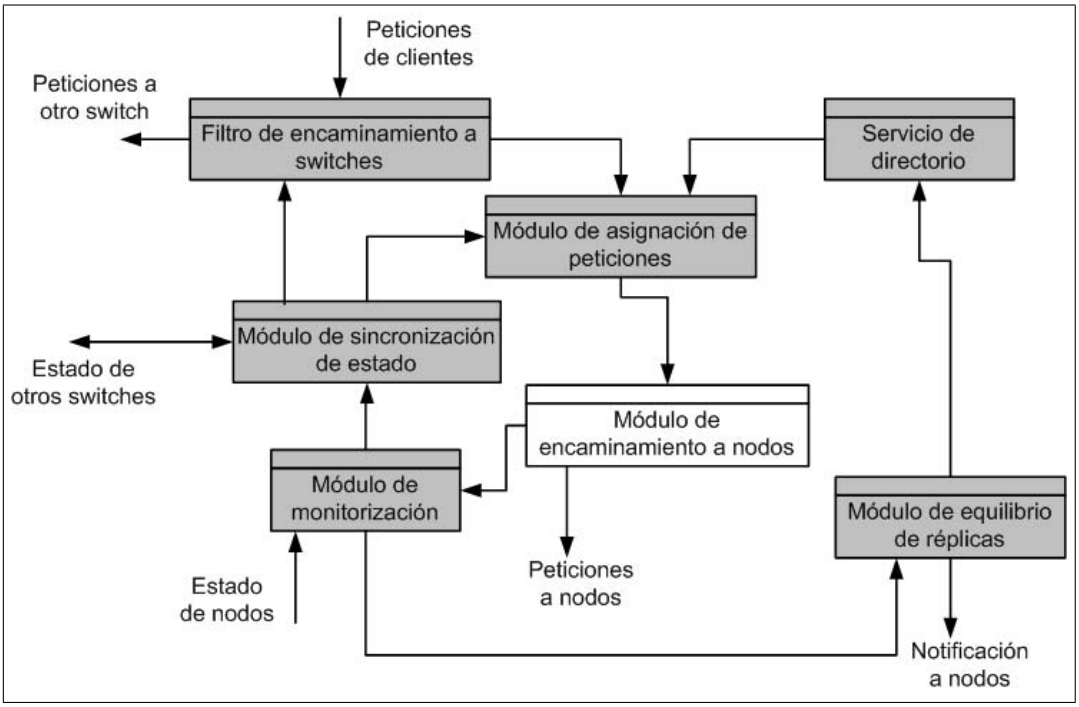

Figura 3.22: Módulos que intervienen en el procesamiento de peticiones en un sistema SD RD/FU.

La adaptación dinámica de réplicas se consigue mediante el uso de un módulo de equilibrio de réplicas que se encarga de determinar periódicamente la asignación de réplicas a nodos servidores y de coordinar la redistribución de réplicas y los contenidos de los servicios de directorio.

Además, no es suficiente que los nodos servidores tengan alojado un servidor Web estándar. De forma periódica, cada nodo servidor debe enviar información de estado al módulo de monitorización del *switch Web*.

3.5.3 Resumen de las propuestas arquitectónicas sobre *cluster Web con switch distribuido*

En esta sección se ha propuesto una nueva arquitectura: el *cluster Web con switch distribuido*. Dicha arquitectura combina ventajas de las arquitecturas que este trabajo ha identificado como más interesantes para el uso con replicación parcial y distribución dinámica de réplicas.

Las arquitecturas de *sistemas Web basados en cluster* presentan un cuello de botella en el procesamiento de peticiones debido a que todas las peticiones deben pasar por el *switch Web*. Además, en dichas arquitecturas el propio *switch Web* supone un serio problema desde el punto de vista de la tolerancia a fallos.

Las arquitecturas de *cluster Web con switch distribuido* solucionan este problema al ofrecer un conjunto de *switches* como punto de entrada al sistema. Esto permite distribuir la carga para incrementar la escalabilidad y además incrementa la tolerancia a fallos del sistema.

Por otra parte las arquitecturas basadas en *sistemas Web distribuidos* requieren que todos

los nodos servidores posean una dirección IP pública. Además, al utilizarse replicación parcial se corre el riesgo de que el número de redirecciones sea tan elevado que pueda afectar al rendimiento global del sistema.

Las arquitecturas de *cluster Web con switch distribuido* no se ven afectadas por este problema al requerirse únicamente que los *switches Web* posean dirección IP pública. Además, con estas arquitecturas, el primer nivel de redirección es independiente de la localización de las réplicas, al encontrarse una copia del servicio de directorio en cada *switch*. Esto hace que la única razón para que una petición sea redirigida a otro *switch* venga dada por la saturación del mismo.

3.6 Conclusiones

En este capítulo se han presentado diversas arquitecturas para servidores Web distribuidos y se ha estudiado su adecuación ante las restricciones de replicación parcial de contenidos y asignación dinámica de réplicas.

La arquitectura de *cluster Web virtual* se ha descartado por no ser adecuada para el uso con réplicas parciales de contenidos.

La primera contribución de este capítulo ha sido la propuesta de soluciones arquitectónicas de *sistemas Web basados en cluster* que son capaces de satisfacer los objetivos de replicación parcial de contenidos y distribución dinámica de réplicas.

La segunda contribución ha sido la propuesta de soluciones arquitectónicas de *sistemas Web distribuidos* que son capaces de satisfacer los objetivos de replicación parcial de contenidos y distribución dinámica de réplicas.

La comparación de ambas soluciones arquitectónicas sugiere que aunque ninguna de las dos soluciones está exenta de inconvenientes, es más deseable el uso de soluciones de *sistemas Web basados en cluster* realizando modificaciones que mitiguen su vulnerabilidad.

A partir de esta consideración se ha llegado a la tercera contribución de este capítulo: la propuesta de una nueva solución arquitectónica denominada *cluster Web con switch distribuido*.

En el siguiente capítulo se realiza un estudio de las estrategias que pueden utilizarse para la asignación de réplicas parciales a nodos servidores. Así mismo, en dicho capítulo se estudia qué políticas de asignación de peticiones se pueden utilizar de forma que sean compatibles con las arquitecturas propuestas en este capítulo.

Capítulo 4

Políticas de asignación de réplicas y de peticiones

En el capítulo anterior se ha realizado un estudio de diversas arquitecturas distribuidas que son compatibles con el uso de réplicas parciales. Así mismo, en dicho capítulo se ha propuesto una nueva arquitectura: el *cluster Web con switch distribuido*.

Independientemente de la arquitectura seleccionada, el uso de una estrategia de replicación parcial impone la necesidad de realizar la asignación de las réplicas de cada elemento a los distintos nodos. En este capítulo se proponen diversos algoritmos para la asignación de réplicas a nodos. El capítulo realiza primero una aproximación formal al problema definiendo la notación utilizada para formularlo. Posteriormente se realizan propuestas para la resolución del problema, comenzando con las soluciones a la asignación estática de réplicas, para continuar luego con las soluciones a la asignación dinámica.

Tanto en el caso de una arquitectura de *sistema Web basado en cluster* como en el de una arquitectura de *cluster Web con switch distribuido* es necesario, además, la definición de un mecanismo de asignación de peticiones a los nodos. En este capítulo se proponen distintas estrategias de asignación de peticiones que pueden utilizarse en presencia de replicación parcial. Primero se presenta una política que no utiliza información de estado, para pasar posteriormente a políticas que utilizan información de estado.

En este capítulo se realiza una presentación de las aportaciones realizadas tanto en lo relativo a la asignación de réplicas como en lo relativo a la asignación de peticiones. La sección 4.1 realiza una presentación del problema de la asignación de réplicas y define los modelos necesarios para la representación del mismo. La sección 4.2 presenta diversos algoritmos para la asignación estática de réplicas. En la sección 4.3 se tratan los aspectos relativos a la distribución dinámica de las réplicas. La sección 4.4 presenta el problema de la asignación de peticiones a los nodos servidores y propone soluciones al mismo. Finalmente, la sección 4.5 presenta un resumen de las contribuciones realizadas en este capítulo.

4.1 El problema de la asignación de réplicas

El problema de asignación de réplicas consiste en asignar las distintas réplicas de cada elemento a cada uno de los nodos servidores. En esta sección se presenta el modelo general (y

su notación) que se utilizará para definir las distintas variantes del problema y para expresar los correspondientes algoritmos que lo resuelven.

4.1.1 Modelo de sitio Web

Un sitio Web está formado por un conjunto de elementos o recursos, donde cada uno de ellos puede ser de un determinado tipo (página HTML, imagen, video, archivo de descarga, música, etc). Se puede definir el conjunto E como el conjunto de todos los elementos que forman el sitio Web:

$$E = \{e_1, e_2, \ldots, e_N\}$$

Una página Web está formada por un elemento primario (por ejemplo, una página HTML) que hace referencia a un conjunto de elementos secundarios (otros elementos incluidos en la página). Cuando un cliente solicita una página Web, primero solicita el elemento primario, y tras analizarlo y determinar los elementos secundarios asociados, solicita dichos elementos secundarios.

Por tanto, cada elemento del conjunto E puede ser un elemento primario o un elemento secundario. De esta forma, el conjunto E se puede expresar como la unión del conjunto de elementos primarios E^p y el conjunto de elementos secundarios E^s:

$$E = E^p \cup E^s$$

$$E^p = \{e_1^p, e_2^p, \ldots, e_n^p\}$$
$$E^s = \{e_1^s, e_2^s, \ldots, e_m^s\}$$

Existe una relación de dependencia entre los elementos secundarios y los elementos primarios que viene dada por el hecho de que en un elemento primario pueden estar incluidos un cierto número de elementos secundarios. Estas relaciones de dependencia entre elementos primarios y elementos secundarios se puede expresar mediante una matriz de inclusión I (véase ecuación 4.1).

$$I = (i_{ij}) \;\; |i_{ij} = \begin{cases} 1 & \text{si } e_i^p \text{ incluye } e_j^s \\ 0 & \text{en otro caso} \end{cases} \tag{4.1}$$

Con el apoyo de la matriz de inclusión se puede definir una página Web como un par formado por un elemento primario y un conjunto de elementos secundarios:

$$w_i = \langle e_i^p, E_i^s \rangle$$

$$E_i^s = \left\{ e_j^s \mid i_{ij} = 1 \right\}$$

Con lo que se puede definir el conjunto W de todas las páginas Web del sitio como:

$$W = \{w_1, w_2, \ldots, w_n\}$$

Cada elemento tiene asociado un tamaño t_i. La tabla 4.1 muestra el tamaño de distintos elementos y conjuntos.

Expresión	Valor	Significado
e_i	t_i	Tamaño de un elemento.
e_i^p	t_i^p	Tamaño de un elemento primario.
e_i^s	t_i^s	Tamaño de un elemento secundario.
E	$\sum_{k=1}^{N} t_k$	Tamaño del sitio Web.
E^p	$\sum_{k=1}^{N} t_k^p$	Tamaño de los elementos primarios.
E^s	$\sum_{k=1}^{M} t_k^s$	Tamaño de los elementos secundarios.
E_i^s	$\sum_{k=1}^{M} i_{ik} t_k^s$	Tamaño de los elementos secundarios de una página Web.
w_i	$t_i^p + \sum_{k=1}^{M} i_{ik} t_k^s$	Tamaño de una página Web.

Tabla 4.1: Tamaño de distintos elementos y conjuntos.

4.1.2 Modelo de servidor

Independientemente de la arquitectura utilizada, un servidor Web distribuido utiliza un conjunto de nodos servidores para alojar y proporcionar los contenidos de un sitio Web.

Sea S un conjunto de M nodos servidores:

$$S = \{s_1, s_2, \ldots, s_M\}$$

Cada nodo s_i tiene asociada una capacidad de almacenamiento c_i, que es el máximo volumen de almacenamiento que puede utilizar para alojar contenidos.

La asignación de contenidos a nodos se puede representar mediante la matriz de asignación A (véase ecuación 4.2).

$$A = (a_{ij}) \, | a_{ij} = \begin{cases} 1 & \text{si } e_i \text{ está asignado a } s_j \\ 0 & \text{en caso contrario} \end{cases} \tag{4.2}$$

Esta matriz se puede descomponer en dos submatrices: una para representar la asignación de elementos primarios (véase ecuación 4.3a) y otra para representar la asignación de elementos secundarios (véase ecuación 4.3b).

$$A^p = \left(a_{ij}^p\right) \, | a_{ij}^p = \begin{cases} 1 & \text{si } e_i^p \text{ está asignado a } s_j \\ 0 & \text{en otro caso} \end{cases} \tag{4.3a}$$

$$A^s = \left(a_{ij}^s\right) \, | a_{ij}^s = \begin{cases} 1 & \text{si } e_i^s \text{ está asignado a } s_j \\ 0 & \text{en otro caso} \end{cases} \tag{4.3b}$$

En general, el problema de la asignación consiste en encontrar los valores de una matriz A que represente la asignación de elementos a nodos. Como se verá en las próximas secciones, los valores de dicha matriz pueden estar sujetos a distintas restricciones lo que dará lugar a las soluciones que se proponen para dicho problema.

4.2 Propuestas de asignación estática de réplicas

En la asignación estática de réplicas la matriz A se determina antes de poner en marcha el sistema y sus valores se mantienen constantes a lo largo de la vida del sistema. En esta sección se proponen distintas soluciones al problema dependiendo de las restricciones que deban cumplirse.

Primeramente, se presentan las soluciones más simples en las que se supone que la capacidad de almacenamiento de los nodos no está restringida y que todos los elementos deben replicarse un mismo número de veces. Si bien esta familia de problemas no tiene un interés práctico, su solución permite introducir las estrategias a utilizar. Posteriormente, se proponen las soluciones para el caso en que la capacidad de almacenamiento está restringida y el número de réplicas es el mismo para todos los elementos. Por último, se elimina la restricción de igual número de réplicas y se propone un algoritmo que determina un número distinto de réplicas para cada elemento.

4.2.1 Propuestas de asignación para capacidad de almacenamiento no restringida

En esta sección se presentan algoritmos de asignación estática de réplicas que no tienen en cuenta la capacidad de almacenamiento de los nodos.

Por simplicidad, en este caso se considera que el número de réplicas por elemento es un parámetro del algoritmo y es fijo e igual para todos los elementos.

Por tanto, en este caso el problema a resolver consiste en realizar la asignación de N elementos primarios a un conjunto de M nodos servidores, de forma que cada elemento sea asignado a r nodos servidores. Es decir, el problema consiste encontrar un conjunto de valores a_{ij} que permiten que cumpla

$$\sum_{j=1}^{M} a_{ij} = r \ \forall i = 1, \ldots, N$$

4.2.1.1 Asignación cíclica inicial

Este algoritmo (véase algoritmo 1) realiza la asignación de las réplicas de forma cíclica en el conjunto de nodos, de forma que si el elemento e_k se replica en la secuencia de nodos s_i, s_{i+r}, el elemento e_{k+1} se replica en los nodos s_{i+1}, s_{i+r+1}. Para los subíndices de los nodos servidores se utiliza aritmética modular referida al número de nodos servidores.

El algoritmo no realiza ninguna distinción entre elementos primarios y secundarios. Para cada página Web realiza primero la distribución de su elemento primario, seguida de la distribución de todos sus elementos secundarios. Para mostrar el comportamiento del algoritmo se aplicará a un conjunto de páginas (véase tabla 4.2).

$k \leftarrow 1$
para todo $w_i \in W$
  {Replicar e_i^p en r nodos consecutivos a partir de s_k}
  para $l = 0, \ldots, r-1$
    $u \leftarrow 1 + (k + l - 1)$ mód M
    $a_{iu}^p \leftarrow 1$
  fin para
  $k \leftarrow 1 + k$ mód M
  para todo $e_j^s \in E_i^s$
    {Replicar e_j^s en r nodos consecutivos a partir de s_k}
    para $l = 0, \ldots, r-1$
      $u \leftarrow 1 + (k + l - 1)$ mód M
      $a_{iu}^s \leftarrow 1$
    fin para
    $k \leftarrow 1 + k$ mód M
  fin para
fin para

Algoritmo 1: Asignación cíclica inicial de réplicas.

Página Web	**Elemento primario**	**Elementos secundarios**
w_1	e_1^p	e_1^s , e_2^s
w_2	e_2^p	
w_3	e_3^p	e_3^s , e_4^s , e_5^s

Tabla 4.2: Conjunto de páginas ejemplo.

Aplicando la distribución cíclica inicial a este ejemplo, con un conjunto de ocho nodos servidores y un número de réplicas por elemento igual a tres, la distribución obtenida sería la de la tabla 4.3.

Servidor 1	e_1^p						e_4^s	e_5^s
Servidor 2	e_1^p	e_1^s						e_5^s
Servidor 3	e_1^p	e_1^s	e_2^s					
Servidor 4		e_1^s	e_2^s	e_2^p				
Servidor 5			e_2^s	e_2^p	e_3^p			
Servidor 6				e_2^p	e_3^p	e_3^s		
Servidor 7					e_3^p	e_3^s	e_4^s	
Servidor 8						e_3^s	e_4^s	e_5^s

Tabla 4.3: Ejemplo de distribución de réplicas utilizando el algoritmo de distribución cíclica inicial.

Una desventaja que plantea este tipo de distribución es que se da frecuentemente el caso de que un nodo servidor aloja varios elementos secundarios correspondientes a una misma página Web. Esto se debe a que la distancia entre el primer nodo de un elemento y el primer nodo del siguiente es de 1. Esto plantea un inconveniente desde el punto de vista de distribución de

las peticiones, ya que cuando un cliente accede a una página Web realiza una petición para cada uno de sus elementos secundarios. Dicho inconveniente se produce si el cliente pide los elementos en paralelo, que es lo que ocurre con la mayoría de los clientes modernos.

Si se define el siguiente conjunto:

$$N_i^s \equiv \{\text{conjunto de nodos en los que se aloja el elemento } e_i^s\}$$

Se pueden determinar los conjuntos (y sus correspondientes números de elementos) para las distintas páginas Web del ejemplo presentado (véase tabla 4.4).

Conjunto	**Elementos**	**Número de elementos**
N_1^s	$\{s_2, s_3, s_4\}$	3
N_2^s	$\{s_3, s_4, s_5\}$	3
$N_1^s \cup N_2^s$	$\{s_2, s_3, s_4, s_5\}$	4
$N_1^s \cap N_2^s$	$\{s_3, s_4\}$	2
N_3^s	$\{s_6, s_7, s_8\}$	3
N_4^s	$\{s_1, s_7, s_8\}$	3
N_5^s	$\{s_1, s_2, s_8\}$	3
$N_3^s \cup N_4^s \cup N_5^s$	$\{s_1, s_2, s_6, s_7, s_8\}$	5
$N_3^s \cap N_4^s$	$\{s_7, s_8\}$	2
$N_4^s \cap N_5^s$	$\{s_1, s_8\}$	2
$N_3^s \cap N_4^s \cap N_5^s$	$\{s_8\}$	1

Tabla 4.4: Conjuntos de nodos que alojan diversos elementos en el caso de distribución cíclica inicial.

En general el número de nodos que contiene al menos un elemento secundario de una página Web es

$$\text{mín}\,\{M, r + s - 1\}$$

donde M es el número de nodos servidores, r es el número de réplicas por elemento y s es el número de elementos secundarios de la página Web.

Así mismo, el número de nodos que contienen a todos los elementos secundarios de una página Web es

$$\text{máx}\,\{0, r - s + 1\}$$

Lo deseable sería que, para cada página Web, la primera cantidad fuese lo más alta posible y la segunda lo más baja posible.

4.2.1.2 Asignación cíclica final

Una alternativa para mitigar el problema identificado en la sección anterior es realizar una asignación cíclica que, dentro de la medida de lo posible, minimice el número de nodos servidores que contengan dos elementos consecutivos.

$k \leftarrow 1$
para $w_i \in W$
  {Replicar e_i^p en r nodos a partir de s_k}
  para $l = 0, \ldots, r-1$
    $u \leftarrow 1 + (k + l - 1)$ mód M
    $a_{iu}^p \leftarrow 1$
  fin para
  $k \leftarrow 1 + (k + r - 1)$ mód M
  para $e_j^s \in E_i^s$
    {Replicar e_j^s en r nodos a partir de s_k}
    para $l = 0, \ldots, r-1$
      $u \leftarrow 1 + (k + l - 1)$ mód M
      $a_{iu}^s \leftarrow 1$
    fin para
    $k \leftarrow 1 + (k + r - 1)$ mód M
  fin para
fin para

Algoritmo 2: Asignación cíclica final de réplicas.

El algoritmo de asignación cíclica final (véase algoritmo 2) realiza la asignación de las réplicas de forma cíclica en el conjunto de nodos, de forma que si el elemento e_k se replica en la secuencia de nodos s_i, s_{i+r-1}, el elemento e_{k+1} se replica en los nodos s_{i+r}, s_{i+2r-1}.

En este caso, el algoritmo no realiza tampoco distinción entre elementos primarios y secundarios. Por cada página Web realiza primero la distribución de su elemento primario, y después, la distribución de todos sus elementos secundarios. La tabla 4.5 muestra la aplicación del algoritmo de distribución al ejemplo de la tabla 4.2 para un conjunto de cuatro nodos servidores y tres réplicas por elemento.

Servidor 1	e_1^p		e_2^s			e_4^s	
Servidor 2	e_1^p			e_2^p		e_4^s	
Servidor 3	e_1^p			e_2^p			e_5^s
Servidor 4		e_1^s		e_2^p			e_5^s
Servidor 5		e_1^s			e_3^s		e_5^s
Servidor 6		e_1^s			e_3^s		
Servidor 7			e_2^s		e_3^s		
Servidor 8			e_2^s			e_4^s	

Tabla 4.5: Ejemplo de distribución de réplicas utilizando el algoritmo de distribución cíclica final.

Si se determinan los conjuntos (y sus correspondientes números de elementos) para las distintas páginas Web del ejemplo (véase tabla 4.6) se obtienen valores distintos del caso anterior.

El número de nodos que contiene al menos un elemento secundario de una página Web es

$$\text{mín}\,\{rs, M\}$$

Por otra parte, el número de nodos que contienen a todos los elementos de una página Web es

$$\begin{cases} 0 & \text{si } rs \leq M \\ \max\{0, r-(s-1)(M-r)\} & \text{en otro caso} \end{cases}$$

Conjunto	Elementos	Número de elementos
N_1^s	$\{s_4, s_5, s_6\}$	3
N_2^s	$\{s_1, s_7, s_8\}$	3
$N_1^s \cup N_2^s$	$\{s_1, s_4, s_5, s_6, s_7, s_8\}$	6
$N_1^s \cap N_2^s$	$\varnothing$	0
N_3^s	$\{s_5, s_6, s_7\}$	3
N_4^s	$\{s_1, s_2, s_8\}$	3
N_5^s	$\{s_3, s_4, s_5\}$	3
$N_3^s \cup N_4^s \cup N_5^s$	$\{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$	8
$N_3^s \cap N_4^s$	$\varnothing$	0
$N_4^s \cap N_5^s$	$\varnothing$	0
$N_3^s \cap N_4^s \cap N_5^s$	$\varnothing$	1

Tabla 4.6: Conjuntos de nodos que alojan diversos elementos en el caso de distribución cíclica final.

4.2.2 Propuestas de asignación equitativa para capacidad de almacenamiento restringida

Los algoritmos presentados en la sección anterior realizan el reparto de réplicas sin tener en cuenta la posible limitación de capacidad de almacenamiento. No obstante, suele ser habitual que el almacenamiento disponible en cada nodo servidor no sea ilimitado. De hecho, las restricciones de capacidad de almacenamiento son una de las razones que justifican el uso de replicación parcial en vez de replicación total.

En esta sección se estudia la asignación equitativa para capacidad de almacenamiento restringida. Es decir, que el número de réplicas que se realiza de cada elemento es el mismo. En la sección siguiente (véase 4.2.3) se presenta la asignación que utiliza distinto número de réplicas para distintos elementos.

Ahora el problema consiste en encontrar el número de réplicas r y la matriz de asignación A de forma que el número de réplicas de cada elemento sea r y que no se rebase la capacidad de almacenamiento de ningún nodo. Es decir:

$$\sum_{j=1}^{M} a_{ij} = r \quad \forall i \in [1, N] \tag{4.4a}$$

$$\sum_{i=1}^{N} a_{ij} t_i \leq c_j \quad \forall j \in [1, M] \tag{4.4b}$$

Si se realiza una suma de la familia de inecuaciones 4.4b se obtiene la restricción de la ecuación 4.5.

$$\sum_{j=1}^{M}\sum_{i=1}^{N} a_{ij}t_i \leq \sum_{j=1}^{M} c_j \tag{4.5}$$

Por otra parte, se tiene que

$$\sum_{j=1}^{M}\sum_{i=1}^{N} a_{ij}t_i = \sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_i j$$

Pero, aplicando la ecuación 4.4a

$$\sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_i j = \sum_{i=1}^{N} t_i r = r\sum_{i=1}^{N} t_i$$

Aplicando este resultado sobre la ecuación 4.5 se tiene

$$r\sum_{i=1}^{N} t_i \leq \sum_{j=1}^{M} c_j$$

De donde se obtiene una restricción (véase ecuación 4.6) sobre el número de réplicas en forma de cota superior.

$$r \leq \frac{\sum_{j=1}^{M} c_j}{\sum_{i=1}^{N} t_i} \tag{4.6}$$

En muchas ocasiones el número máximo de réplicas admisible puede ser menor que este valor. Esto se debe a dos razones. Por una parte, no es admisible distribuir una única réplica entre varios nodos servidores. Es decir la unidad de asignación es el archivo. Por otra parte, no tiene sentido que más de una réplica se asigne a un mismo nodo servidor (es decir, $\forall i, j : a_{ij} \in \{0, 1\}$).

Estas restricciones transforman el problema de la asignación en un problema de programación lineal entera que es computacionalmente costoso. No obstante, en la mayoría de los casos existen varios factores que pueden permitir simplificar dicho problema.

- El número de nodos servidores es muchísimo menor que el número de elementos a distribuir.
- El número de nodos servidores es un número relativamente cercano a 0.

- El número de réplicas es un número bastante cercano a la cota inferior de la ecuación 4.6.

Todo esto hace que una solución más practicable sea partir de la cota superior y realizar reducciones en dicho número si el algoritmo de asignación supera la capacidad máxima de algún nodo servidor. En las siguientes secciones se muestran diversas variantes para la asignación de réplicas.

4.2.2.1 Asignación equitativa cíclica inicial

Este algoritmo (véase algoritmo 3) es una modificación del algoritmo de asignación cíclica inicial, que tiene en cuenta las restricciones de capacidad de almacenamiento de los nodos servidores y asigna igual número de réplicas para todos los elementos.

$r \leftarrow 1$
$A \leftarrow 0, B \leftarrow 0$
mientras $r \leq M$ y $T\,(A + B) \leq C$
  $A \leftarrow A + B$
  $B \leftarrow 0$
  $k \leftarrow r$
  para todo $w_i \in W$
    $b^p_{ik} = 1$
    $k \leftarrow 1 + (k + 1)$ mód M
    para todo $e^s_j \in E^s_i$
      $b^s_{jk} \leftarrow 1$
      $k \leftarrow 1 + (k + 1)$ mód M
    fin para
  fin para
  $r \leftarrow r + 1$
fin mientras

Algoritmo 3: Asignación equitativa cíclica inicial de réplicas.

El algoritmo realiza asignaciones de todos los elementos siguiendo la estrategia cíclica inicial. Si después de una asignación de este tipo, queda sitio para realizar otra asignación de todos los elementos se vuelve a repetir el proceso. Cuando en una iteración no se puede alojar algún elemento (por sobrepasarse la capacidad máxima de un nodo), se anula la iteración completa.

Para simplificar la exposición del algoritmo, se utiliza en algunos puntos del mismo notación matricial. El algoritmo utiliza la matriz de asignación A, y una matriz auxiliar de asignación B. La matriz B se inicia en cada iteración con todos sus elementos a 0 y se utiliza para realizar las asignaciones de una iteración que se consolidan en la siguiente, exclusivamente si hay espacio disponible. El vector fila T contiene los tamaños de todos los elementos. Así mismo, el vector fila C contiene las capacidades de los nodos servidores.

Por tanto, el producto

$$T \cdot A$$

da lugar a un vector fila con el espacio ocupado en cada nodo. Si se considera que el operador $\leq$ realiza la comparación de dos vectores elemento a elemento, de forma que un vector es menor que otro, solamente si cada elemento del primer vector es menor que el correspondiente elemento del segundo vector, la condición para que una asignación A sea compatible con las capacidades de almacenamiento de los nodos es

$$T \cdot A \leq C$$

y la condición para que una asignación auxiliar B se pueda incorporar a la asignación A es

$$T \cdot (A + B) \leq C$$

Obsérvese, que en el algoritmo, y por conveniencia de notación, a veces se accede a la matriz B mediante sus submatrices B^p (matriz de asignación de elementos primarios) y B^s (matriz de asignación de elementos secundarios).

Para ilustrar el funcionamiento del algoritmo se ha tomado el ejemplo de la tabla 4.2 y se ha supuesto que dichos elementos tienen los tamaños de la tabla 4.7. Además, se ha supuesto que todos los nodos tienen una capacidad de 15000.

Elemento	**Tamaño**
e_1^p	100
e_1^s	2500
e_2^s	4000
e_2^p	150
e_3^p	175
e_3^s	1800
e_4^s	3200
e_5^s	10000

Tabla 4.7: Tamaños de los elementos utilizados en el ejemplo.

La tabla 4.8 muestra los elementos que se asignan a cada nodo servidor, así como el espacio ocupado.

Nodo	**Elementos**	**Ocupación**
s_1	e_1^p , e_5^s , e_4^s	13300
s_2	e_1^s , e_1^p , e_5^s	12600
$s3$	e_2^s , e_1^s , e_1^p	6600
$s4$	e_2^p , e_2^s , e_1^s	6650
$s5$	e_3^p , e_2^p , e_2^s	4325
$s6$	e_3^s , e_3^p , e_2^p	2125
$s7$	e_4^s, e_3^s , e_3^p	5175
$s8$	e_5^s , e_4^s , e_3^s	15000

Tabla 4.8: Ejemplo de resultado con asignación equitativa cíclica inicial.

Al igual que en el caso de la asignación cíclica inicial sin limitación de la capacidad de almacenamiento (véase 4.2.1.1), las réplicas correspondientes a distintos elementos secundarios de un mismo elemento primario quedan asignadas a conjuntos de nodos con intersección

no vacía. Para resolver este problema, otra alternativa es el uso de un esquema basado en asignación cíclica final como el de la siguiente sección.

4.2.2.2 Asignación equitativa cíclica final

Este algoritmo (véase algoritmo 4) es una modificación del algoritmo de asignación cíclica final, que tiene en cuenta las restricciones de capacidad de almacenamiento de los nodos servidores y asigna igual número de réplicas para todos los elementos.

El algoritmo realiza asignaciones de todos los elementos siguiendo la estrategia cíclica final. El primer paso es la selección de un número de réplicas tentativo (véase ecuación (4.6)). A partir del número de réplicas seleccionado se trata de realizar un asignación cíclica final, si en algún momento se excede la capacidad de algún nodo, se reduce el número de réplicas y se vuelve a repetir el proceso.

La tabla 4.9 muestra el resultado final de aplicar el algoritmo al conjunto de páginas utilizado como ejemplo. Como puede observarse en este caso se produce un menor grado de replicación y el aprovechamiento de la capacidad de almacenamiento de los nodos servidores es menor.

Nodo	**Elementos**	**Ocupación**
s_1	e_1^p , e_2^s , e_3^p , e_2^s	7475
s_2	e_1^p , e_2^s , e_3^p , e_2^s	7475
$s3$	e_1^p , e_2^s , e_3^p , e_2^s	7475
$s4$	e_1^p , e_2^s , e_3^p , e_2^s	7475
$s5$	e_1^s , e_2^p , e_1^s , e_3^s	14450
$s6$	e_1^s , e_2^p , e_1^s , e_3^s	14450
$s7$	e_1^s , e_2^p , e_1^s , e_3^s	14450
$s8$	e_1^s , e_2^p , e_1^s , e_3^s	14450

Tabla 4.9: Ejemplo de resultado con asignación equitativa cíclica final.

4.2.2.3 Asignación voraz de réplicas

Otra solución distinta para resolver el problema de la asignación de réplicas (véase algoritmo 5) es la utilización de un algoritmo voraz. En este caso la selección de elementos se hace basándose en el tamaño del elemento, realizando primero la asignación de los elementos de mayor tamaño. Además debe tenerse en cuenta la restricción de que no se puede asignar más de una réplica de un elemento a un mismo nodo.

Al igual que en el caso anterior, el algoritmo comienza intentando asignar el máximo número de réplicas posible. Si no se encuentra una solución para ese número de réplicas, se reduce dicho número y se vuelve a intentar encontrar una solución.

La tabla 4.10 muestra el resultado final de aplicar el algoritmo al conjunto de páginas utilizado como ejemplo. En el ejemplo puede observarse que el algoritmo tiende a generar espacios de ocupación más equilibrados.

$$r \leftarrow \frac{\sum_{i=1}^{M} c_i}{\sum_{j=1}^{N} t_j}$$

repetir
  $A \leftarrow 0$
  $k \leftarrow 1$
  para todo $w_i \in W$
    {Replicar e_i^p en r nodos a partir de s_k}
    para $l = 0, \ldots, r-1$
      $u \leftarrow 1 + (k + l - 1)$ mód M
      $a_{iu}^p \leftarrow 1$
      si $\sum_{i=1}^{N} a_{iu} t_i > c_u$ **entonces**
        $r \leftarrow r - 1$
        Pasar a siguiente iteración de **repetir**
      fin si
    fin para
    $k \leftarrow 1 + (k + r - 1)$ mód M
    para todo $e_j^s \in E_i^s$
      {Replicar e_j^s en r nodos a partir de s_k}
      para $l = 0, \ldots, r-1$
        $u \leftarrow 1 + (k + l - 1)$ mód M
        $a_{iu}^p \leftarrow 1$
        si $\sum_{i=1}^{N} a_{iu} t_i > c_u$ **entonces**
          $r \leftarrow r - 1$
          Pasar a siguiente iteración de **repetir**
        fin si
      fin para
      $k \leftarrow 1 + (k + r - 1)$ mód M
    fin para
  fin para
  salir
fin repetir

Algoritmo 4: Asignación equitativa cíclica final de réplicas.

4.2.3 Propuesta de asignación no equitativa para capacidad de almacenamiento restringida

Todos los algoritmos, presentados hasta el momento se basan en la asignación del mismo número de réplicas para todos los elementos. Esto sería apropiado si las peticiones de elementos se distribuyesen de forma uniforme para todos los elementos de un sitio Web. No obstante

$$r \leftarrow \frac{\sum_{i=1}^{M} c_i}{\frac{i=1}{N}} \sum_{j=1} t_j$$

Ordenar los elementos e_i en orden decreciente de tamaño
repetir
  para $i = 1$ to M
    $l_i \leftarrow c_i$
  fin para
  para todo $e_i \in E$
    Generar la lista $o = \langle o_1, \ldots, o_m \rangle$ con los índices de los servidores ordenados por orden decreciente de espacio disponible l_j
    si $\exists j \in \{o_1, \ldots, o_r\} \ | t_i > l_j$ **entonces**
      $r \leftarrow r - 1$
      $A \leftarrow 0$
      Pasar a siguiente iteración del bucle externo
    fin si
    $a_i j \leftarrow 1 \ \forall j = o_1, \ldots, o_r$
    $l_j \leftarrow l_j - t_i \ \forall j = o_1, \ldots, o_r$
  fin para
  salir
fin repetir

Algoritmo 5: Asignación voraz réplicas.

Nodo	**Elementos**	**Ocupación**
s_1	e_1^p , e_2^p , e_3^p , e_5^s	10425
s_2	e_1^p , e_2^p , e_3^p , e_5^s	10425
$s3$	e_1^p , e_2^p , e_3^p , e_5^s	10425
$s4$	e_1^p , e_2^p , e_3^p , e_5^s	10425
$s5$	e_1^s , e_2^2 , e_3^s , e_4^s	11500
$s6$	e_1^s , e_2^2 , e_3^s , e_4^s	11500
$s7$	e_1^s , e_2^2 , e_3^s , e_4^s	11500
$s8$	e_1^s , e_2^2 , e_3^s , e_4^s	11500

Tabla 4.10: Ejemplo de resultado con asignación equitativa voraz.

esto no suele ser así. En un sitio Web, unos elementos son más populares que otros, de forma que las peticiones recibidas sobre dichos elementos serán superiores en número.

En estos casos, puede ser más conveniente disponer de un mayor número de réplicas para los elementos más populares, y disponer de un menor número de réplicas para los elementos menos populares.

Ahora el problema consiste en encontrar los valores r_i y la matriz de asignación A, de forma que el número de réplicas de cada elemento e_i sea r_i y que no se rebase la capacidad

de almacenamiento c_j de ningún nodo. Es decir

$$\sum_{j=1}^{M} a_{ij} = r_i \ \forall i \in [1, N] \tag{4.7a}$$

$$\sum_{i=1}^{N} a_{ij} t_i \leq c_j \ \forall j \in [1, M] \tag{4.7b}$$

Si se realiza la suma de la familia de inecuaciones 4.7b se obtiene la restricción 4.8.

$$\sum_{j=1}^{M} \sum_{i=1}^{N} a_{ij} t_i \leq \sum_{j=1}^{M} c_j \tag{4.8}$$

Por otra parte, se tiene que

$$\sum_{j=1}^{M} \sum_{i=1}^{N} a_{ij} t_i = \sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_{ij}$$

Y aplicando la ecuación 4.7a

$$\sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_{ij} = \sum_{i=1}^{N} t_i r_i$$

Por tanto la restricción que debe mantenerse viene expresada por la inecuación 4.9.

$$\sum_{i=1}^{N} t_i r_i \leq \sum_{j=1}^{M} c_i \tag{4.9}$$

Si se conoce la probabilidad de que se realice una petición sobre un elemento, se puede realizar una asignación de réplicas que asigne un mayor número de réplicas para los elementos más populares y un menor número de réplicas para los elementos menos populares.

Sea p_i la probabilidad de que una petición haga referencia al elemento e_i. Se puede reordenar los elementos de tal forma que

$$p_i \geq p_{i+1}$$

Se puede realizar un reparto del espacio proporcional a la probabilidad y que tenga en cuenta las restricciones del problema.

La idea pasa por asignar inicialmente a cada elemento una cuota de participación en el almacenamiento global proporcional a su probabilidad asociada. De esta forma, los elementos

más populares tendrán más espacio disponible que los elementos menos populares. Por tanto, el número de réplicas inicial para cada elemento vendrá dado por

$$r_i = \frac{p_i \sum_{j=1}^{M} c_j}{t_i}$$

No obstante, el número de réplicas de un elemento no puede ser menor que 1 ni mayor que M, por lo que es necesario realizar ajustes del número de réplicas. Por tanto, se puede definir r_i^* tal y como se muestra en la ecuación 4.10.

$$r_i^* = \begin{cases} 1 & \text{si } r_i < 1 \\ M & \text{si } r_i > M \\ \lfloor r_i \rfloor & \text{si } 1 \leq r_i \leq M \end{cases} \tag{4.10}$$

Una vez realizado el ajuste del número de réplicas de cada elemento, se puede utilizar un algoritmo voraz de asignación de réplicas basado en probabilidades (véase algoritmo 6).

Determinar r_i^* $\forall i$ mediante la ecuación 4.10
Ordenar los elementos e_i en orden decreciente de tamaño
repetir
  $l_i \leftarrow c_i$ $\forall i$
  para todo $e_i \in E$
    Generar la lista $o = \langle o_1, \ldots, o_m \rangle$ con los índices de los servidores ordenados por orden decreciente de espacio disponible l_j
    si $\exists j \in \{o_1, \ldots, o_r\} \mid t_i > l_j$ **entonces**
      Seleccionar un r_k^* al que reducir el número de réplicas
      $r_k^* \leftarrow r_k^* - 1$
      $A \leftarrow 0$
      Pasar a siguiente iteración de bucle externo
    fin si
    $a_{ij} \leftarrow 1$ $\forall j = o_1, \ldots, o_{r_i^*}$
    $l_j \leftarrow l_j - t_i$ $\forall j = o_1, \ldots, o_{r_i^*}$
  fin para
  salir
fin repetir

Algoritmo 6: Asignación basada en probabilidades.

Un problema adicional a resolver es la selección del elemento sobre el que realizar reducción de réplicas en caso de que el algoritmo llegue a una situación en que la asignación no sea posible.

Para resolver este problema debe seleccionarse un elemento al que reducir el número de réplicas. La idea de la reducción del número de réplicas se basa en la determinación del valor

α_i (véase ecuación 4.11).

$$\alpha_i = \frac{r_i^*}{r_i} \tag{4.11}$$

Para aquellos elementos para los que r_i fuese menor que 1, r_i^* es exactamente 1 y por tanto

$$r_i < 1 \Rightarrow r_i^* = 1 \Rightarrow \alpha_i = \frac{r_i^*}{r_i} = \frac{1}{r_i} > 1$$

Para el resto de elementos, α_i expresa la proximidad entre los valores r_i y r_i^*. Un valor de α_i muy próximo a 1 indica que dicho elemento tiene asignados un número de réplicas muy próximo al que le correspondería por su probabilidad p_i. Por tanto, el mejor candidato para reducir réplicas, será aquel cuyo valor α_i sea más próximo a 1 sin sobrepasarlo.

Teniendo en cuenta este criterio, se puede refinar el algoritmo (véase algoritmo 7).

Determinar r_i^* $\forall i$ mediante la ecuación 4.10
Ordenar los elementos e_i en orden decreciente de tamaño
repetir
  $l_i \leftarrow c_i$ $\forall i$
  para todo $e_i \in E$
    Generar la lista $o = \langle o_1, \ldots, o_m \rangle$ con los índices de los servidores ordenados por orden decreciente de espacio disponible l_j
    si $\exists j \in \{o_1, \ldots, o_r\} \mid t_i > l_j$ **entonces**
      Determinar α_i $\forall i$ mediante la ecuación 4.11
      Seleccionar $k | \nexists q : \alpha_q \leq 1 \wedge \alpha_q > \alpha_k$
      $r_k^* \leftarrow r_k^* - 1$
      $A \leftarrow 0$
      Pasar a siguiente iteración de bucle externo
    fin si
    $a_{ij} \leftarrow 1$ $\forall j = o_1, \ldots, o_{r_i^*}$
    $l_j \leftarrow l_j - t_i$ $\forall j = o_1, \ldots, o_{r_i^*}$
  fin para
  salir
fin repetir

Algoritmo 7: Asignación basada en probabilidades refinada.

Un aspecto importante a tener en cuenta es la selección de las probabilidades p_i asociadas a cada uno de los elementos e_i. Cuando la asignación de réplicas es estática, no se tiene información sobre la probabilidad que tiene cada elementos de ser solicitado. De hecho la única información que se tiene sobre los elementos del sitio Web es la relativa a su estructura y a los tamaños de los elementos.

No obstante, a partir del tamaño de cada elemento se puede realizar una estimación de la probabilidad de petición. Diversos estudios empíricos [73, 74] demuestran que si se ordenan los

elementos en orden de popularidad decreciente, la probabilidad de acceso al i-esimo elemento sigue la distribución propuesta por George Zipf [75] (véase ecuación 4.12).

$$p(i) = \frac{k}{i} \tag{4.12}$$

Por la definición de probabilidad, debe verificarse que

$$\sum_{i=1}^{N} p(i) = 1$$

y por tanto

$$\sum_{i=1}^{N} \frac{k}{i} = k \sum_{i=1}^{N} \frac{1}{i} = 1$$

y el valor de k debe ser

$$k = \frac{1}{\sum_{i=1}^{N} \frac{1}{i}}$$

No obstante, no se tiene ningún conocimiento a priori sobre la popularidad de las páginas. Otros estudios [76, 77] demuestran que la popularidad de un elemento es mayor cuanto menor es su tamaño.

El algoritmo 8 muestra como puede generarse las probabilidades de acceso para cada uno de los elementos.

Ordenar los elementos e_i en orden decreciente de tamaño t_i
$k \leftarrow \frac{1}{\sum_{i=1}^{N} \frac{1}{i}}$
para $i = 1$ hasta N
  $p_i \leftarrow \frac{k}{i}$
fin para

Algoritmo 8: Asignación de probabilidades mediante distribución de Zipf.

Si se aplica este algoritmo al conjunto de páginas ejemplo presentado en las Tablas 4.2 y 4.7, la función de Zipf debe tener un valor de $k = 0{,}367936925$. La tabla 4.11 presenta los valores de p_i, r y r^* para el conjunto de páginas ejemplo.

Dada esta asignación de coeficientes, la ejecución del algoritmo, genera la asignación de réplicas que se muestra en la tabla 4.12.

Elemento	Tamaño	Orden	p_i	r	r^*
e_1^p	100	8	0,045992116	55,19053876	8
e_1^s	2500	4	0,091984231	4,415243101	4
e_2^s	4000	2	0,183968463	5,519053876	5
e_2^p	150	7	0,052562418	42,0499343	8
e_3^p	175	6	0,061322821	42,0499343	8
e_3^s	1800	5	0,073587385	4,905825668	4
e_4^s	3200	3	0,122645642	4,599211564	4
e_5^s	10000	1	0,367936925	4,415243101	4

Tabla 4.11: Asignación de probabilidades y valores de r y r^* para el conjunto de elementos del ejemplo.

Nodo	Elementos	Ocupación
s_1	e_5^s , e_2^s , e_3^p , e_2^p , e_1^p	675
s_2	e_5^s , e_3^p , e_2^p , e_1^p	4575
$s3$	e_5^s , e_3^p , e_2^p , e_1^p	7475
$s4$	e_5^s , e_3^p , e_2^p , e_1^p	4575
$s5$	e_2^s , e_4^s , e_1^s , e_3^s , e_3^p , e_2^p , e_1^p	3075
$s6$	e_2^s , e_4^s , e_1^s , e_3^s , e_3^p , e_2^p , e_1^p	3075
$s7$	e_2^s , e_4^s , e_1^s , e_3^s , e_3^p , e_2^p , e_1^p	3075
$s8$	e_2^s , e_4^s , e_1^s , e_3^s, e_3^p , e_2^p , e_1^p	3075

Tabla 4.12: Ejemplo de resultado con asignación no equitativa de réplicas.

4.3 Propuestas de asignación dinámica de réplicas

Todas las estrategias de asignación estática de réplicas establecen una asignación inicial de réplicas que no se modifica a lo largo del tiempo. Para cierto tipo de servicios, esto puede ser válido porque el patrón de acceso es estático. Pero es frecuente que el patrón de acceso a un servidor Web se modifique a lo largo del tiempo. Existen razones para ello:

- En muchos casos la información que se publica más recientemente es la más accedida por los usuarios. En un sitio Web que muestre las últimas noticias relacionadas por la organización, el acceso a dicha información implica peticiones de elementos distintos en cada momento.

- Existen ciertos servicios en los que el perfil de los usuarios se modifica con cierta periodicidad. En muchos sitios Web, el perfil de los usuarios que acceden durante los días laborables es muy distinto del perfil de los usuarios que acceden en fin de semana.

- El patrón de acceso se puede modificar ante la proximidad de un cierto evento. En el servicio Web de una universidad, la demanda de información sobre los contenidos de las asignaturas optativas se incrementa cuando se aproxima un periodo de matriculación.

- El patrón de acceso se puede modificar a lo largo de un mismo día. En muchos sitios Web el patrón de acceso es distinto por la mañana, al mediodía, por la tarde o por la

noche.

Todo esto hace que, en ciertos casos, pueda ser necesario modificar la distribución de réplicas entre nodos de forma periódica. Además, la adaptación dinámica de réplicas introduce una ventaja sobre la asignación estática de réplicas: en este caso no es necesario asumir que la frecuencia de peticiones para cada elemento sigue una determinada distribución teórica. En vez de esto, se puede realizar una recogida de la información y utilizar dicha información como entrada para la realización de la asignación de réplicas.

Dos problemas deben resolverse en relación con la asignación dinámica de réplicas. Por una parte, debe estimarse la probabilidad de acceso a cada elemento a partir del funcionamiento del sistema. Por otra parte, debe realizarse la redistribución de réplicas con un mínimo impacto sobre el rendimiento del sistema.

4.3.1 Frecuencia de peticiones

La estimación de la probabilidad de acceso a cada elemento se puede realizar a partir de las peticiones recibidas para cada elemento. Sea q_i el número de peticiones recibidas sobre el elemento e_i. La probabilidad de que una petición se realice sobre el elemento e_i se puede estimar como

$$\widehat{p_i} = \frac{q_i}{\sum_{i=1}^{N} q_i}$$

Por tanto la estimación de probabilidades no requiere ningún algoritmo complejo.

No obstante la recogida del número de peticiones que hacen referencia a cada elemento es distinta dependiendo de la familia de arquitectura utilizada.

En el caso de un *sistema Web basado en cluster* la recogida de información es sencilla puesto que todas las peticiones pasan por el *switch* Web y por tanto se puede realizar en ese punto dicha recogida de información.

En un *sistema Web distribuido*, cada nodo servidor recibe un conjunto de peticiones (que puede en algún caso traspasar a otro nodo). Por tanto, cada nodo debe recoger información sobre las peticiones efectivamente servidas por él y posteriormente es necesario realizar una fusión de la información de todos los nodos antes de realizar la asignación de elementos. Esta es una de las razones por las que en esta tesis no se ha dado preferencia al uso de este tipo de arquitecturas.

En un *cluster Web con switch distribuido*, cada *switch* recibe un conjunto de peticiones (que pueden ser traspasadas a otro *switch*). Por tanto, cada *switch* debe recoger información sobre las peticiones de las que se responsabiliza. Posteriormente, todos los *switches* menos uno (el que toma el rol de primario) traspasan la información recogida a uno de los *switches*, que es el encargado de realizar la asignación de elementos.

4.3.2 Asignación de elementos

En la asignación de elementos a los nodos se distinguen dos tipos: la asignación inicial y la asignación posterior.

La asignación inicial se realiza antes del arranque del sistema. Como en ese momento no se dispone de información sobre la frecuencia de acceso a cada elemento, se puede utilizar como estimador de la probabilidad de petición, una función de probabilidad a priori como las expuestas en la seccion 4.2.3.

La asignación posterior se realiza periódicamente o cuando el patrón de acceso ha cambiado sustancialmente. Para las asignaciones posteriores se puede utilizar como estimador de la probabilidad de petición, la frecuencia de petición.

En cualquier caso, el algoritmo a utilizar es el ya expuesto en la sección 4.2.3 (véase algoritmo 7).

Un problema que se plantea es la determinación del momento en que debe ejecutarse el algoritmo de asignación de réplicas. Si se opta por la ejecución periódica del algoritmo cada cierto tiempo, puede ocurrir que algunas veces el patrón de acceso se haya modificado sustancialmente mientras que otras veces los cambios sean mínimos.

Una solución a este problema es el uso del concepto de distancia entre dos vectores de probabilidad.

Sea $\widehat{p}_i^t$ el estimador de la probabilidad asociada al elemento e_i en el instante t. Se puede definir la distancia entre dos instantes de tiempo como

$$d\left(t, t+\Delta t\right) = \sum_{i=1}^{N} \left(\widehat{p}_i^t - \widehat{p}_i^{t+\Delta t}\right)^2$$

Utilizando el concepto de distancia se puede definir un umbral de redistribución u_r, de forma que la condición para que se ejecute el algoritmo de asignación de elementos sea

$$d\left(t, t+\Delta t\right) > u_r$$

No obstante, el problema del uso de la distancia euclídea es que puede dar mucha importancia a modificaciones en el vector de probabilidades que no generen modificaciones en el número de réplicas que se debe asignar a un determinado elemento. Otra solución que se puede aplicar a este problema es la utilización como criterio, las modificaciones en el número de réplicas de los elementos.

Sea $\bar{r}^*\left(t\right)$ el vector que define el número de réplicas que debe asignarse a cada elemento, evaluado en el momento t. El número de réplicas se determina mediante la ecuación (4.10). Sea $\bar{r}^*\left(t+\Delta t\right)$ el vector que define el número de réplicas que debe asignarse a cada elemento, evaluado en el momento $t+\Delta t$. La expresión

$$\sum_{i=1}^{N} \left| r_i^*\left(t\right) - r_i^*\left(t+\Delta t\right) \right|$$

representa el número de réplicas que deben eliminarse más el número de réplicas que deben añadirse como efecto de los cambios en los vectores de probabilidades.

El criterio que puede utilizarse para decidir si se realiza la redistribución será

$$\sum_{i=1}^{N} \left| r_i^*\left(t\right) - r_i^*\left(t+\Delta t\right) \right| > R$$

donde R es el número combinado de eliminaciones y adiciones de réplicas a partir del cual es necesario realizar redistribuciones.

La Figura 4.1 muestra la evolución de la función distancia a lo largo del tiempo. Cuando la función distancia rebasa el umbral establecido se realiza la redistribución de elementos entre los nodos servidores con lo que se reduce el valor de la función.

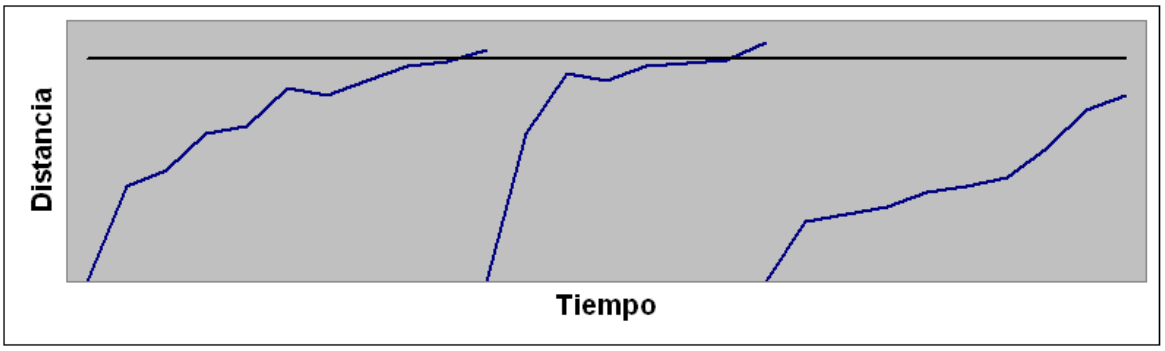

Figura 4.1: Evolución de la función distancia a lo largo del tiempo y momentos de redistribución.

4.3.3 Redistribución de elementos

Una vez se ha ejecutado el algoritmo de asignación de réplicas en algún momento posterior al inicial, es necesario realizar la distribución física de réplicas. Para determinar las acciones que deben realizarse se puede utilizar la matriz de asignación A.

Sea A la matriz de asignación antes de ejecutar el algoritmo de asignación de réplicas, y sea A' la matriz de asignación después de ejecutar dicho algoritmo. Se puede definir la matriz de transacciones T como

$$T = A - A'$$

Como tanto A como A' son matrices en que todos sus elementos pertenecen al conjunto $\{0, 1\}$, la matriz tendrá el siguiente significado

$$T = (t_{ij}) \mid \begin{cases} -1 & \text{Se debe copiar el elemento } e_i \text{ al nodo } s_j \\ 0 & \text{No se debe realizar ninguna operación} \\ 1 & \text{Se debe eliminar el elemento } e_i \text{ del nodo } s_j \end{cases}$$

Las operaciones de la matriz T no se pueden realizar en cualquier orden, porque existen dos restricciones:

- Por una parte no se pueden realizar primero todas las eliminaciones porque se corre el riesgo de borrar totalmente un elemento.
- Por otra parte no se pueden realizar primero todas las copias porque se podría requerir mas espacio del disponible.

Es más, siempre existen casos para el que cualquier transición entre el estado A y el estado A' tiene que pasar necesariamente por un estado intermedio que viola las restricciones del problema.

Por tanto, es necesario el uso de un espacio de almacenamiento para poder realizar las transiciones desde A hasta A'.

4.4 Propuestas de asignación de peticiones

Un servidor Web recibe de un conjunto de clientes una sucesión de peticiones z_i. De forma general el problema de la asignación de peticiones consiste en encontrar una función que asigne a cada petición z_i un nodo servidor s_i.

En el caso de los *sistemas Web basados en cluster* la asignación de peticiones se realiza en el *switch Web*. En el caso de la arquitectura propuesta en esta tesis (el *cluster Web con switch distribuido*) se hace necesario la implantación de la política de asignación de peticiones en cada *switch* del *cluster*.

Como ya se ha expuesto en el capítulo 2, la forma más sencilla de asignación de peticiones es aquella en que la función de asignación no depende de ninguna información externa. En esta categoría entran, por ejemplo, el uso de una política de asignación estática circular (véase 2.4.2), en la que la función de asignación es del tipo:

$$f : Z \mapsto [1, M]$$

$$f(z_i) = (i \text{ mód } M) + 1$$

Otras formas más complejas de asignación tienen en cuenta información del cliente (véase 2.5) o del servidor (véase 2.6). Si se considera que Y es el conjunto de información adicional, se tiene que de forma general, la función de asignación es

$$f : Z \times Y \mapsto [1, M]$$

En general, unas políticas de asignación de peticiones son apropiadas para el caso en que los contenidos están totalmente replicados y otras políticas de asignación son apropiadas para el caso en que los contenidos están totalmente distribuidos. La tabla 4.13, muestra qué políticas pueden ser utilizadas en el caso de replicación total y qué políticas pueden usarse en el caso de distribución total.

La *asignación aleatoria estática* hace corresponder aleatoriamente un nodo servidor con cada petición, por lo que asume que cualquier petición puede ser servida por cualquier nodo servidor. Esto hace que solamente sea compatible con la replicación total de contenidos.

La *asignación estática circular* selecciona, de forma rotatoria, un nodo servidor para cada petición, de forma que también asume que cualquier petición puede ser servida por cualquier nodo. Por la misma razón, solamente es compatible con la replicación total de contenidos.

En la *asignación por partición de clientes* todas las peticiones que provienen de un mismo cliente se asignan siempre a un mismo nodo servidor. Otra vez, se asume, de forma implícita, que todos los contenidos deben estar replicados en todos los nodos servidores.

Con la *asignación por partición de URL* se selecciona para cada URL un nodo servidor. Esto hace esta política especialmente apropiada para la distribución total de contenidos. Aunque en este caso no se impide la replicación total de contenidos, dicha estrategia no tiene

Política	**Replicación total**	**Distribución total**	**Replicación parcial**
Asignación aleatoria estática	✓	×	×
Asignación estática circular	✓	×	×
Asignación por partición de clientes	✓	×	×
Asignación por partición de URL	✓	✓	×
Asignación por partición de servicios	✓	✓	×
Asignación SITA-E	✓	✓	×
Asignación al nodo menos cargado	✓	×	×
Asignación dinámica cíclica por pesos	✓	×	×
Asignación LARD	✓	×	×

Tabla 4.13: Comparación de la idoneidad de distintos tipos de políticas de asignación de peticiones para modelos de replicación de contenidos.

mucho sentido con esta política porque las peticiones de un cierto elemento siempre irán dirigidas al mismo nodo.

La *asignación por partición de servicios* puede ser estática o dinámica. En el caso de partición de servicios estática, las peticiones se asignan a los nodos servidores dependiendo del tipo de contenido asociado al elemento solicitado. Esto hace que la única opción sea la distribución total de contenidos (todos los contenidos de un mismo tipo deben alojarse en un mismo nodo servidor). En el caso de la partición dinámica de servicios, como la política *Client Aware Policy* (véase 2.5.3), una determinada petición puede ser asignada a cualquier nodo servidor por lo que la única opción sería la replicación total.

La política *SITA-E* distribuye las peticiones dividiendo el conjunto de elementos en tantos subconjuntos como nodos servidores, atendiendo al tamaño de los elementos. Si se alojan los elementos de cada subconjunto en el correspondiente nodo servidor, se permite el uso de distribución total. No obstante, debe recordarse que esta política no es compatible con la replicación parcial porque, dado un elemento de un cierto tamaño, todas las peticiones que hagan referencia a dicho elemento serán siempre asignadas al mismo nodo servidor.

La *asignación al nodo menos cargado* selecciona siempre el nodo servidor con menor carga según alguna métrica, lo que una vez más se apoya en la asunción implícita de que todos los nodos servidores contienen una copia de todos los elementos que componen el sitio Web.

La *asignación dinámica cíclica por pesos* selecciona los nodos de forma circular aunque repitiendo más veces la selección de los nodos que tienen más capacidades disponibles. Como cada petición puede ser asignada a cualquier nodo servidor, esta política también se apoya en el hecho de que todos los contenidos se encuentran replicados en todos los nodos servidores.

Por último la política *LARD* también se apoya en la replicación total de contenidos, puesto que el conjunto de nodos servidores que dan servicio a un cierto elemento se amplia seleccionando al nodo que tiene menos carga en un cierto momento.

Como se ha expuesto, ninguna de estas políticas puede usarse directamente en el caso de replicación parcial de contenidos. Por una parte, las políticas orientadas al caso de replicación total determinan el nodo servidor asumiendo que todos los elementos se encuentran en todos los nodos. Por otra parte aquellas políticas apropiadas para contenidos totalmente distribuidos, determinan el nodo en el que se encuentra el contenido solicitado y asignan la

petición a dicho nodo.

Para que una política de asignación de peticiones pueda usarse en el caso en que los contenidos se encuentran parcialmente replicados, es necesario que incorpore a su proceso de decisión la restricción de seleccionar un nodo servidor que efectivamente contenga el elemento solicitado. Dicho de otra forma, es necesario adaptar la política de forma que solamente se tenga en cuenta para cada petición aquellos nodos que alojan el elemento solicitado.

En las siguientes secciones se proponen las adaptaciones necesarias para el uso, en caso de contenidos parcialmente replicados, de distintas políticas. Primeramente se propone la adaptación de la política estática circular (véase sección 4.4.1). Posteriormente se propone la adaptación de la política de asignación al nodo menos cargado (véase sección 4.4.2). Por último, se propone la adaptación de la política de asignación LARD (véase sección 4.4.3).

4.4.1 Asignación parcial estática circular

En esta sección se propone una adaptación de la política estática circular que tiene en cuenta la restricción de replicación parcial.

Al estar los contenidos parcialmente replicados, cada elemento se encontrará replicado en un subconjunto de nodos distinto. Sea S_i el conjunto de nodos donde está almacenado el elemento e_i. Entonces, el número de elementos del conjunto S_i, será

$$|S_i| = r_i$$

donde cada conjunto S_i es

$$S_i = \left\{ s_1^i, s_2^i, \ldots, s_{r_i}^i \right\}$$

Como el conjunto de nodos candidatos no es el mismo para cada elemento, no es posible utilizar un único índice para ir iterando por los distintos nodos servidores. Es necesario utilizar un índice k_i para cada elemento e_i. Dichos índices pueden iniciarse al primer servidor del conjunto asociado a dicho elemento

$$k_i \leftarrow 1 \ \forall i \in [1, N]$$

El algoritmo 9 muestra el sencillo proceso de asignación de peticiones en este caso.

Asignar la petición del elemento e_i al nodo $s_{k_i}^i$
$k_i \leftarrow k_i \text{ mód } r_i + 1$

Algoritmo 9: Asignación parcial estática circular.

No obstante, este algoritmo requiere el uso de N índices (tantos como elementos). Una solución para reducir el número de índices utilizados es emplear un índice para todos los elementos que se alojan en el mismo subconjunto de nodos servidores. De esta forma, el número de subconjuntos de nodos servidores no depende del número de elementos alojados, sino que depende exclusivamente del número de nodos servidores. Por tanto la iniciación de los índices pasa a ser

$$k_i \leftarrow 1 \ \forall i \in \left[1, 2^M - 1\right]$$

que se puede completar con la definición de un valor m_i asociado a cada elemento que indique el índice que debe utilizarse para dicho elemento

$$m_i = \sum_{j=1}^{M} 2^j a_{ij}$$

de forma que el índice a usar para el elemento e_i será k_{m_i}.

Nótese que si se utilizan todos los índices k_i, esta mejora solamente supone un ahorro cuando se cumple

$$2^M - 1 < N$$

esto quiere decir que para un conjunto de 10^5 elementos solamente se obtendría un ahorro cuando se utilizan menos de 16 nodos servidores. Sin embargo, en realidad no se utilizan todos los índices k_i, puesto que hay subconjuntos de S a los que no se asigna ningún elemento. Por tanto, se puede conseguir un ahorro sustancial de almacenamiento para los índices, si éstos se almacenan en alguna estructura de datos asociativa.

La iniciación de los índices queda finalmente como

$$k_i \leftarrow 1 \quad \forall i \in \bigcup_{j \in [1,N]} \{m_j\}$$

El algoritmo 10 muestra el proceso de asignación de peticiones en este caso.

Asignar la petición el elemento e_i al nodo $s^i_{k_{m_i}}$
$k_{m_i} \leftarrow k_{m_i}$ mód $r_i + 1$

Algoritmo 10: Asignación parcial estática circular con índices agrupados.

4.4.2 Asignación parcial al nodo menos cargado

En esta sección se propone una adaptación de la política de asignación de peticiones al nodo menos cargado que tiene en cuenta la restricción de la replicación parcial.

La asignación al nodo menos cargado se basa en la definición de alguna métrica de carga que pueda evaluarse para todos los nodos servidores. Sea $c_i(t)$ el valor de dicha métrica de carga para el nodo servidor s_i en el instante t.

En el caso de un sistema Web basado en *cluster*, la evaluación de dicha métrica de carga puede realizarse de forma sencilla en el propio *switch* si se utiliza información disponible en dicho elemento como, por ejemplo, el número de conexiones activas.

En el caso de un *cluster Web con switch distribuido*, cada *switch* dispone de la información relativa a las peticiones que ha gestionado, pero no existe ningún punto central donde toda la información pueda encontrarse agregada. Para solucionar este problema se propone el intercambio de información entre los *switches* de la contribución a la carga de los nodos, cada cierto tiempo. Como la contribución a la carga debida al *switch* que evalúa la petición está siempre más actualizada, se puede conceder más importancia a dicha información realizando una agregación de información como la presentada en la ecuación 4.13. En dicha

expresión se considera que $c_i^j(r)$ es la carga del nodo s_i debida al *switch* ws_j en el instante t, y que el último intercambio de información se ha producido en el instante t_{int}.

$$\widehat{c}_i^j(t) = \alpha c_i^j(t) + (1-\alpha)\sum_{\substack{k=1\\k\neq i}}^{M} c_k^j(t_{int}) \tag{4.13}$$

4.4.3 Asignación parcial LARD

En las secciones anteriores se han propuesto algoritmos de asignación de peticiones que no tienen en cuenta el hecho de que el tiempo de respuesta de un nodo servidor ante una petición depende de la localización o no del elemento solicitado en su memoria caché. En esta sección se propone una adaptación de la política de asignación de peticiones LARD (véase 2.6.3) que tiene en cuenta la restricción de la replicación parcial.

Aunque la política LARD ofrece buenos resultados, dicha política está diseñada para su utilización en *sistemas Web basados en cluster* en la que todos los elementos se encuentran replicados en todos los nodos. En el caso de replicación parcial de contenidos algunas adaptaciones son necesarias.

Por una parte, cuando se utiliza una asignación del número de réplicas basada en probabilidades de acceso, se da lugar a un número importante de elementos para los que se asigna una única réplica ($r_i = 1$). Por tanto para dichos elementos, no tiene sentido la aplicación del algoritmo, ya que todas las peticiones se asignan al mismo nodo.

Por otra parte, el algoritmo de asignación de peticiones debe considerar exclusivamente aquellos nodos que contienen el elemento solicitado.

Inicialmente (véase algoritmo 11) se construye una familia de conjuntos S_i que contienen los nodos servidores en los que se almacena el correspondiente elemento e_i. Además, se construye otra familia de conjuntos T_i, inicialmente asignados al conjunto vacío, y que mantendrán los nodos servidores que tienen asignados cada elemento. Por último se inicia el tiempo de modificación t_i^{mod} de cada conjunto T_i a 0.

```
para todo i = 1 hasta N
  t_i^mod ← 0
  S_i ← ∅
  T_i ← ∅
  para todo j = 1 hasta M
    si a_ij = 1 entonces
      S_i ← S_i ∪ {s_j}
    fin si
  fin para
fin para
```

Algoritmo 11: Iniciación de valores para algoritmo parcial LARD.

Cuando llega una petición (véase algoritmo 12), se determina el número de nodos servidores que pueden satisfacer la petición. Si solamente un nodo puede satisfacer la petición,

ésta se asigna a dicho nodo. En otro caso se estudia el conjunto T_i de nodos que actualmente tienen asignado el servicio del elemento e_i.

Si el conjunto T_i es vacío, se selecciona el nodo menos cargado de los que pueden satisfacer la petición para dar servicio a la misma y se incorpora dicho nodo al conjunto T_i

Si el conjunto T_i es no vacío, se determina cuál de los nodos incluidos en el mismo es el menos cargado (s_{sel}), y si comprueba si dicho nodo no está saturado. En este caso se asigna la petición al nodo seleccionado. Si todos los nodos han llegado a un punto de saturación, se incorpora un nuevo nodo al conjunto T_i y se le asigna la petición.

En cualquier caso, se hace necesario reducir el conjunto T_i cuando la carga debida a un elemento disminuye. Para conseguir esto, si transcurrido un cierto tiempo t^{rev} no se ha producido ningún cambio en el conjunto T_i, se elimina el nodo menos cargado de los existentes en T_i.

si $|S_i| = 1$ **entonces**
  Asignar la petición a s_1^i
sino
  si $T_i = \emptyset$ **entonces**
    $s_{sel} \leftarrow s_j^i \in S_i \mid c_j = \text{mín}_{k \in [1,r_i]} \{c_k\}$
    $T_i \leftarrow T_i \cup \{s_{sel}\}$
    Asignar la petición al nodo s_{sel}
  sino
    $s_{sel} \leftarrow t_j^i \in T_i \mid c_j = \text{mín}_{k \in [1,r_i]} \{c_k\}$
    si $|T_i| = r_i$ **entonces**
      Asignar la petición al nodo s_{sel}
    sino si $c_{sel} > C_{MAX}$ y $\exists k | c_k < C_{MIN}$ o $c_{sel} \geq 2C_{MAX}$ **entonces**
      $s'_{sel} \leftarrow s_j^i \in S_i \mid c_j = \text{mín}\, k \in [1, r_i] \{c_k\}$
      $T_i \leftarrow T_i \cup \{s'_{sel}\}$
      Asignar la petición al nodo s'_{sel}
    sino
      Asignar la petición al nodo s_{sel}
    fin si
    si $|S_i| > 1$ y $t - t_i^{mod} > t^{rev}$ **entonces**
      $s_{elim} \leftarrow t_j^i \in t_i \mid c_j = \text{máx}_{k \in [1,r_i]} \{c_k\}$
      $T_i \leftarrow T_i - \{s_{elim}\}$
    fin si
  fin si
  si Se ha modificado T_i **entonces**
    $t_i^{mod} \leftarrow t$
  fin si
fin si

Algoritmo 12: Asignación parcial LARD.

4.5 Resumen del capítulo

En este capítulo se ha presentado uno de los problemas asociados a esta tesis: la asignación de réplicas a nodos bajo la hipótesis de replicación parcial de contenidos. Se ha realizado una formulación del problema que permite representar el problema como la búsqueda de los valores para una matriz (la matriz de asignación) que determina la asignación de réplicas a nodos. El problema, en sus distintas variantes, consiste en la determinación de los valores de dicha matriz que cumplen determinadas restricciones.

La primera contribución realizada en este capítulo es la *definición de algoritmos que permiten realizar la asignación de las réplicas de los elementos de un sitio Web a un conjunto de nodos servidores, cuando el almacenamiento está limitado y se deben generar el mismo número de réplicas para todos los elementos.*

La segunda contribución de este capítulo es la *definición de algoritmos que permiten realizar la asignación de las réplicas de los elementos, de forma que los elementos con más frecuencia de acceso tengan un mayor número de réplicas y los elementos con baja frecuencia de acceso tengan pocas réplicas.*

La tercera contribución consiste en la *propuesta de una estrategia para la replicación dinámica de contenidos que permite determinar cuándo es necesario redistribuir contenidos y cómo se debe realizar dicha redistribución.*

Por último, la cuarta contribución consiste en la propuesta de un *algoritmo para la asignación de peticiones a nodos servidores en presencia de réplicas parciales de elementos.*

Capítulo 5

Evaluación

En este capítulo se presenta la evaluación de las ideas propuestas en la tesis. La evaluación se ha realizado atendiendo a tres factores fundamentales:

- La capacidad de almacenamiento de la solución.
- La fiabilidad ofrecida por la solución.
- El rendimiento de la solución en el procesamiento de peticiones.

Para los dos primeros factores (capacidad de almacenamiento y fiabilidad) se ha procedido a realizar una evaluación analítica. Para el rendimiento la evaluación se basa en un modelo de simulación que se basa estudios experimentales realizados sobre sitios Web con gran volumen de almacenamiento y alta carga en cuanto a número de usuarios, como fue el caso del sitio Web de los campeonatos mundiales de fútbol de 1998 en Francia.

La evaluación de la capacidad de almacenamiento se realiza comparando la capacidad de almacenamiento disponible cuando los contenidos están parcialmente replicados con los casos en que los contenidos están totalmente replicados o totalmente distribuidos.

En el caso de la fiabilidad no se puede considerar exclusivamente la estrategia de replicación, sino que además se debe tener en cuenta la arquitectura utilizada. Esto es así, debido a que la topología de la arquitectura tiene impacto sobre la fiabilidad global del sistema. Por esta razón, se ha evaluado la fiabilidad ofrecida por diversas arquitecturas comparando el caso de replicación parcial con los casos de replicación total y distribución total.

La evaluación del rendimiento se ha realizado considerando el impacto que tienen sobre el mismo las distintas estrategias de replicación y los distintos algoritmos de asignación de peticiones.

La sección 5.1 presenta las métricas de evaluación utilizadas en este capítulo. La sección 5.2 contiene la evaluación de la capacidad de almacenamiento conseguida mediante las distintas estrategias de replicación de contenidos. En la sección 5.3 se realiza la evaluación de la fiabilidad para las distintas arquitecturas y estrategias de replicación. En la sección 5.4 se realiza una comparación entre fiabilidad y capacidad de almacenamiento cuando se incrementa el número de réplicas por elemento. La sección 5.5 presenta la evaluación del rendimiento de la solución arquitectónica propuesta. Finalmente, en la sección 5.6 se presentan las conclusiones derivadas de las evaluaciones realizadas.

5.1 Métricas de evaluación

En esta tesis se evalúan tres factores para realizar la comparación de las ideas propuestas con respecto a las ya existentes. En primer lugar, se realiza la evaluación de la capacidad de almacenamiento para realizar una comparación de las distintas estrategias de replicación de contenidos. Posteriormente, se realiza la evaluación de la fiabilidad ofrecida para comparar el efecto sobre la fiabilidad de la selección de una determinada arquitectura y estrategia de replicación. Finalmente, se realiza una comparación del rendimiento ofrecido por la solución propuesta en esta tesis con respecto a otras soluciones ya existentes.

Antes de realizar la evaluación es necesario definir las métricas o indicadores asociados a cada factor evaluado. Esto se realiza en las secciones siguientes.

5.1.1 Métricas de capacidad de almacenamiento

Un aspecto importante, es la determinación de la capacidad efectiva de almacenamiento de la que dispone el servidor Web. Dicha capacidad viene determinada, además de por la capacidad de almacenamiento de cada nodo, por la estrategia general de replicación (replicación total, replicación parcial o distribución total).

Las métricas utilizadas son:

Volumen de almacenamiento utilizado Mide el volumen total de almacenamiento utilizado para alojar un determinado sitio Web. Es decir, el volumen de almacenamiento que efectivamente se está ocupando en todos los nodos servidores.

Tamaño máximo de sitio Web Mide el tamaño máximo que puede tener un sitio Web para poder ser alojado por el conjunto de nodos servidores.

Otro aspecto relacionado, que es interesante medir es el coste que tiene el incremento de capacidad de almacenamiento. Esta característica tiene un claro impacto sobre la escalabilidad del tamaño del sitio Web que se puede alojar.

La métrica que se utiliza para realizar esta medición es:

Eficacia del incremento de capacidad Mide la relación entre el incremento del tamaño máximo del sitio Web que se puede alojar y la adquisición de volumen de almacenamiento que es necesaria.

De forma general, le eficacia viene dada por la ecuación 5.1, donde ΔT es el incremento del tamaño máximo del sitio Web y $C_{adquirida}$ es la capacidad de almacenamiento que se ha adquirido (es decir, la capacidad de los discos que ha sido necesario adquirir).

$$E = \frac{\Delta T}{C_{adquirida}} \tag{5.1}$$

5.1.2 Métrica de fiabilidad

No todas las soluciones que se pueden utilizar para un servidor Web distribuido son igualmente fiables. La fiabilidad depende por una parte de la estructura de la arquitectura seleccionada, ya que la propia topología de la arquitectura puede incrementar o reducir la probabilidad de fallo. Por otra parte, la fiabilidad también se ve afectada por la estrategia de asignación de las réplicas a los nodos servidores.

La métrica utilizada en este caso es:

Fiabilidad Mide la probabilidad de que una petición de un elemento pueda completarse con éxito.

Si la probabilidad de que un sistema falle es P_f, su fiabilidad viene dada por la expresión 5.2 [78].

$$F = 1 - P_f \tag{5.2}$$

5.1.3 Métrica de rendimiento

El rendimiento de un servidor Web distribuido se puede medir con distinto nivel de detalle (globalmente o para cada nodo servidor) y utilizando distintas medidas (tiempo de respuesta, ocupación de procesador, ancho de banda).

No obstante, desde el punto de vista de la percepción que tienen los usuarios del sistema, se necesita medir el tiempo de espera de dichos usuarios ante una petición. Es más, el tiempo de respuesta de cada petición HTTP no da información exacta porque el usuario está interesado en que se satisfagan completamente todas las peticiones HTTP que componen una petición Web.

Para medir el rendimiento se ha utilizado el índice siguiente:

Tiempo de respuesta de peticiones Web Se define como el tiempo que un cliente percibe como tiempo necesario para obtener todos los elementos de una petición Web completa. Como una petición Web está formada por una petición de un elemento primario seguida de varias peticiones de elementos secundarios, se considera tiempo de respuesta el tiempo que transcurre desde que el cliente envía la petición del elemento primario hasta que recibe el último byte del último elemento secundario.

5.2 Evaluación de la capacidad de almacenamiento

La evaluación de la capacidad de almacenamiento permite comparar las distintas alternativas de alojamiento de contenidos entre nodos servidores, por lo que se evaluará esta característica para alojamiento totalmente replicado, alojamiento totalmente distribuido y alojamiento parcialmente replicado.

Obsérvese que no tiene sentido realizar distinción entre las distintas arquitecturas utilizadas, puesto que, en cuanto al alojamiento, todas utilizan un número de nodos servidores para almacenar los contenidos.

En todos los casos, el servidor Web se puede representar mediante el conjunto S ($S = \{s_1, s_2, \ldots, s_M\}$), donde cada servidor s_i tiene una capacidad c_i.

El sitio Web se puede representar mediante el conjunto E de elementos ($E = (e_1, e_2, \ldots, e_N)$), donde cada elemento e_i tiene un tamaño t_i. El tamaño total del sitio Web viene dado por la ecuación 5.3.

$$T = \sum_{i=1}^{N} t_i \tag{5.3}$$

Obsérvese, que para la evaluación de la capacidad de almacenamiento no es necesario establecer una distinción entre elementos primarios y secundarios.

La asignación de elementos a nodos servidores se puede representar mediante la matriz de asignación A:

$$A = (a_{ij}) \mid a_{ij} = \begin{cases} 1 & \text{si } e_i \text{ se aloja en } s_j \\ 0 & \text{si } e_i \text{ no se aloja en } s_j \end{cases}$$

De forma general, el volumen de ocupación V_k en el servidor s_k es:

$$V_k = \sum_{i=1}^{N} a_{ik} t_i$$

y el volumen de ocupación V utilizado para alojar un sitio Web será

$$V = \sum_{j=1}^{M} V_j = \sum_{j=1}^{M} \sum_{i=1}^{N} a_{ij} t_i$$

Esto permite calcular, de forma general, el volumen de almacenamiento utilizado (véase ecuación 5.4).

$$V = \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} t_i \tag{5.4}$$

5.2.1 Replicación total

En el caso de la replicación total todos los elementos están asignados a todos los nodos servidores. Es decir,

$$a_{ij} = 1 \begin{cases} \forall i \in [1, N] \\ \forall j \in [1, M] \end{cases}$$

y por tanto

$$\sum_{i=1}^{N} a_{ij} = N \qquad \sum_{j=1}^{M} a_{ij} = M$$

Si se sustituye en la ecuación 5.4,

$$V = \sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_{ij} = \sum_{i=1}^{N} M t_i = M \sum_{i=1}^{N} t_i = MT$$

En cuanto al tamaño máximo del sitio Web, se debe verificar que

$$V_k \leq c_k \ \forall k \in [1, M]$$

En caso de replicación total, la matriz de asignación tiene todos sus valores a 1, y el volumen de ocupación en un nodo servidor V_k es

$$V_k = \sum_{i=1}^{N} a_{ik} t_i = \sum_{i=1}^{N} t_i = T$$

por lo que la restricción que debe cumplirse es

$$T \leq c_k \ \forall k \in [1, M]$$

o lo que es lo mismo

$$T \leq \min_{k=1..M} \{c_k\}$$

Es decir, que el tamaño máximo de un sitio Web está limitado por el nodo servidor de menor capacidad de almacenamiento.

El aumento de tamaño máximo de almacenamiento no se puede conseguir en este caso añadiendo nuevos nodos servidores, porque esto no incrementaría nunca el valor de T.

Por tanto, la única forma de mejorar el tamaño máximo de almacenamiento es sustituyendo el disco de menor capacidad c_j por otro de mayor capacidad $c'_j = c_j + \Delta c_j$.

Si la nueva capacidad no rebasa el nuevo mínimo de capacidades, se tiene que

$$c'_j = c_j + \Delta c_j \leq \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} \quad \Rightarrow \quad \Delta T = \Delta c_j$$

Por otra parte, si la nueva capacidad rebasa el nuevo mínimo de capacidades, se tiene que

$$c'_j = c_j + \Delta c_j > \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} \quad \Rightarrow \quad \Delta T = \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} - c_j$$

Por tanto

$$\Delta T = \begin{cases} \Delta c_j & \text{si } \Delta c_j \leq \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} - c_j \\ \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} - c_j & \text{si } \Delta c_j > \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} - c_j \end{cases}$$

Es importante observar, que si existe más de un disco con la capacidad mínima

$$\exists j_1, j_2 \mid c_{j_1} = c_{j_2} = \min_{j \in [1,M]} \{c_j\} \quad \Rightarrow \quad \min_{\substack{i \in [1,M] \\ i \neq j}} \{c_i\} - c_j = 0$$

y por tanto

$$\Delta T = \begin{cases} \Delta c_j & \text{si } \Delta c_j \leq 0 \\ 0 & \text{si } \Delta c_j > 0 \end{cases}$$

Teniendo en cuenta que Δc_j no puede ser negativo, se concluye que en el caso de existir más de un disco de capacidad mínima no es posible incrementar la capacidad del sistema sustituyendo un único disco. La única solución en este caso es incrementar la capacidad de todos los discos de capacidad mínima.

Sea $M' \leq M$ el número de discos de capacidad mínima c y considérese los nodos servidores ordenados en orden creciente de capacidad. Es decir

$$c = c_k < \min_{i \in [M'+1, M]} \{c_i\} \ \ \forall k \in \left[1, M'\right]$$

Si se incrementa la capacidad de dichos discos en Δc de forma que dicho incremento no supere el nuevo mínimo

$$c + \Delta c \leq \min_{i \in [M'+1]} \{c_i\} \quad \Rightarrow \quad \Delta T = \Delta c$$

Si por el contrario el incremento se hace de forma que se supere el nuevo mínimo

$$c + \Delta c \leq \min_{i \in [M'+1]} \{c_i\} \quad \Rightarrow \quad \Delta T = \min_{i \in [M'+1]} \{c_i\} - c$$

Es decir, que si existe más de un disco de capacidad mínima es necesario incrementar la capacidad de todos los discos de capacidad mínima y que el límite máximo de incremento viene impuesto por el siguiente disco de capacidad mínima.

Una vez estudiados el volumen de ocupación y el tamaño máximo se pasará a estudiar la eficacia del incremento de capacidad. En este estudio es necesario distinguir el caso en que todos los nodos disponen de capacidades heterogéneas del caso en que las capacidades de los nodos son homogéneas.

Si se considera que los nodos tienen capacidades heterogéneas ($\exists i, j \mid c_i \neq c_j$), se puede optar entre incorporar un disco a un nodo servidor ya existente o sustituir un disco por otro de mayor capacidad.

Si se admite que se puede incorporar un nuevo disco a un nodo servidor ya existente, la eficacia del incremento de almacenamiento viene dada por

$$E = \frac{\Delta T}{\sum_{i=1}^{M} \Delta c_i}$$

En caso de no poder incorporar un nuevo disco a un nodo servidor ya existente, la única solución es sustituir un disco existente por otro de mayor capacidad. Por tanto la eficacia del incremento de almacenamiento viene dada por

$$E = \frac{\Delta T}{\sum_{i=1}^{M} \Delta c_i + \sum_{i \mid \Delta c_i > 0} c_i}$$

Si se considera que los nodos tienen capacidades homogéneas ($\forall i, j : c_i = c_j = c$), se debe incrementar por igual la capacidad de todos los nodos servidores en Δc, bien sea incorporando nuevos discos o sustituyendo los existentes.

Si se admite que se puede incorporar un nuevo disco a cada nodo servidor ya existente, la eficacia del incremento de almacenamiento viene dada por

$$E = \frac{\Delta T}{\sum_{i=1}^{M} \Delta c_i} = \frac{\Delta T}{M \Delta T} = \frac{1}{M}$$

Es decir que la eficacia del incremento es, en este caso, constante y se mantiene en $1/M$.

En caso de no poder incorporar un nuevo disco a un nodo servidor, se hace necesaria la sustitución de los discos existentes por discos de la nueva capacidad, por lo que la eficacia del incremento viene dada por

$$E = \frac{\Delta T}{M\,(c + \Delta T)} = \frac{1}{M} \frac{\Delta T}{c + \Delta T}$$

5.2.1.1 Ejemplo 1: Capacidades heterogéneas

En este ejemplo se considera un servidor compuesto por 12 nodos servidores con la configuración presentada en la tabla 5.1.

Nodo	Capacidad
s_1	50 GB
s_2	80 GB
s_3	80 GB
s_4	100 GB
s_5	100 GB
s_6	100 GB
s_7	200 GB
s_8	200 GB
s_9	200 GB
s_{10}	200 GB
s_{11}	200 GB
s_{12}	200 GB

Tabla 5.1: Ejemplo de configuración de nodos con capacidades de almacenamiento heterogéneas.

En este caso el tamaño máximo del sitio Web es

$$T = \min_{i \in [1,12]} \{c_i\} = 50\ GB$$

La figura Figura 5.1 muestra la eficacia del incremento de almacenamiento para distintos valores de incremento de tamaño máximo.

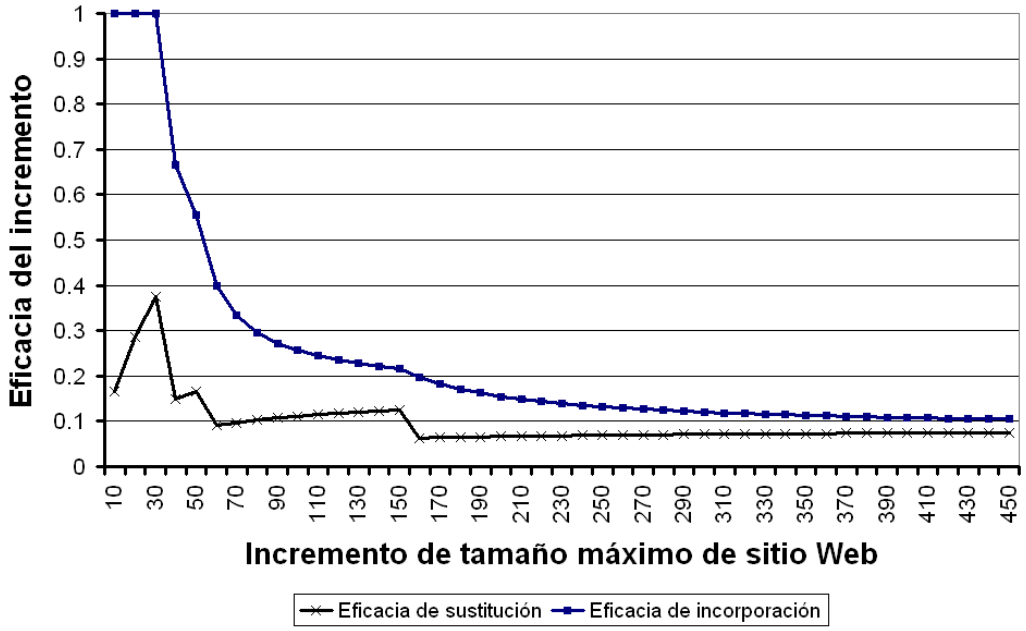

Figura 5.1: Eficacia del incremento de la capacidad en caso de alojamiento totalmente replicado y capacidades heterogéneas.

La eficacia del incremento es siempre mayor cuando se incorpora un nuevo disco a un nodo servidor que cuando se sustituye uno de los discos existentes por otro de mayor capacidad.

En el caso de la incorporación de nuevos discos sin retirar los discos ya existentes, la eficacia es completa cuando solamente es necesario realizar incorporaciones en un nodo. Esto se debe que el incremento en la capacidad del sistema es igual a la capacidad incrementada en dicho nodo. Cuando hay que realizar incorporaciones en más de un nodo servidor la eficacia decrece, ya que es necesario incorporar al sistema varias veces la capacidad de almacenamiento que se desea incrementar (por ejemplo, para conseguir un incremento de 100 GB es necesario incorporar 390 GB y para conseguir un incremento de 450 GB es necesario incorporar 4290 GB).

El caso extremo ocurre cuando la capacidad de almacenamiento a incorporar en cada nodo es muy grande en comparación con el almacenamiento ya disponible en dicho nodo. *En este caso, la eficacia se aproxima a* $1/M$ *(*$1/12$ *en el ejemplo).*

En el caso de la sustitución de los discos ya existentes en los nodos servidores, la eficacia es creciente en cada uno de los intervalos en los que se produce la sustitución de un mismo número de discos. En el ejemplo, para incrementos de hasta 30 GB se necesita sustituir un disco, para incrementos de hasta 50 GB se necesitan sustituir tres discos, para incrementos de hasta 150 GB se necesitan sustituir 6 discos, y para incrementos mayores es necesario sustituir los 12 discos. Dentro de cada intervalo la eficacia es creciente porque crece el porcentaje de capacidad incorporada con respecto al total de capacidad del nodo.

El caso extremo se produce cuando la capacidad incorporada es muy grande en comparación con la capacidad de almacenamiento que existía previamente. *En este caso, la eficacia se aproxima a* $1/M$ *(*$1/12$ *en el ejemplo).*

Si bien para pequeños incrementos, la eficacia de la incorporación de discos es muy superior a la eficacia de la sustitución, cuando el incremento de tamaño se hace mayor, las eficacias se van aproximando. En ambos casos, la eficacia tiende a aproximarse al valor $1/M$.

5.2.1.2 Ejemplo 2: Capacidades homogéneas

En este ejemplo se considera un sistema compuesto por 12 nodos servidores.

La Figura 5.2 muestra la eficacia del incremento de almacenamiento para distintos valores de incremento de tamaño máximo del sitio Web. Se muestra la evolución de eficacia de incorporación para los casos en que la capacidad inicial es 50, 100, 150 y 200 GB.

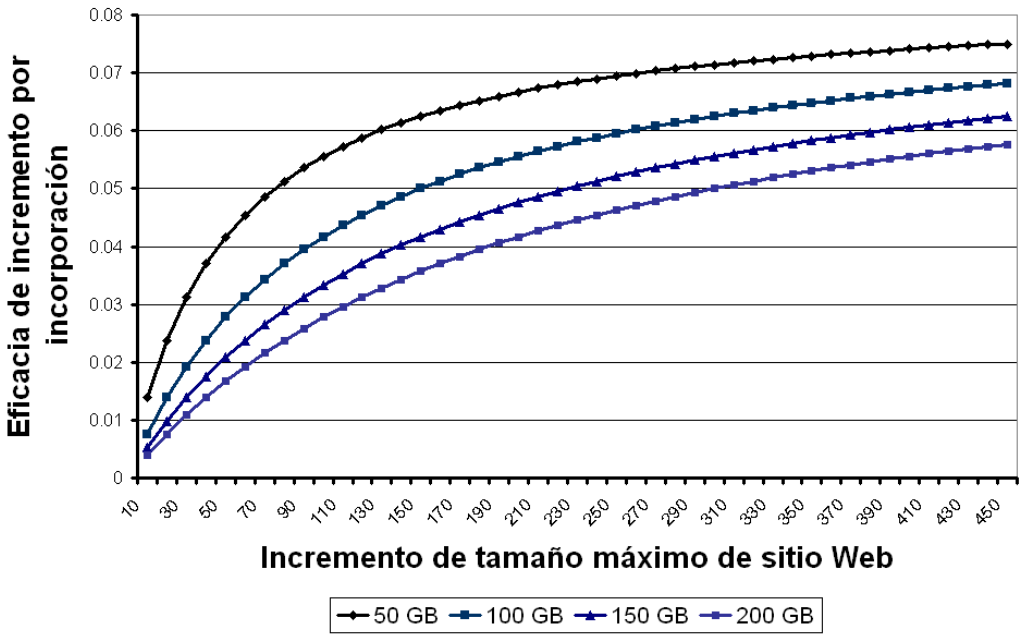

Figura 5.2: Eficacia del incremento de la capacidad en caso de alojamiento totalmente replicado y capacidades homogéneas en los nodos.

Como puede observarse, la eficacia del incremento es mayor cuanto mayor es el incremento de capacidad que se realiza. Cuando los incrementos son muy grandes la eficacia se aproxima al valor $1/M$ (1/12 en el ejemplo).

Es importante tener en cuenta que la eficacia del incremento, en el caso en que sea posible incorporar nuevos discos sin sustituir los existentes, se mantiene de forma constante en $1/M$*, independientemente de la capacidad incrementada. Por tanto, siempre es preferible realizar la incorporación de nuevos discos y la sustitución de discos queda como un mecanismo de aumento de la capacidad para aquellos casos en que no es posible realizar incorporación sin sustitución.*

Para este ejemplo no se ha considerado un tamaño máximo de disco, ya que a partir de este punto el incremento de capacidad no es posible. Es decir, si el mayor disco que se puede adquirir (ya sea por razones técnicas o económicas) es de 400 GB, éste será el tamaño del mayor sitio que se podrá albergar.

5.2.2 Distribución total

En el caso de la distribución total cada elemento está asignado a un único nodo servidor. Es decir,

$$\sum_{i=1}^{N} a_{ij} \leq N \ \forall j \in [1, N] \qquad \sum_{j=1}^{M} a_{ij} = 1 \ \forall i \in [1, M]$$

Si se sustituye en la ecuación 5.4,

$$V = \sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_{ij} = \sum_{i=1}^{N} t_i = T$$

Es decir, que el volumen ocupado es igual al espacio ocupado por el sitio Web.

En cuanto al tamaño máximo que puede tener el sitio Web, se tiene que,

$$V_k \leq c_k \ \forall k \in [1, M] \quad \Rightarrow \quad \sum_{k=1}^{M} V_k \leq \sum_{k=1}^{M} c_k$$

Por otra parte,

$$\sum_{k=1}^{M} V_k = \sum_{k=1}^{M} \sum_{i=1}^{N} a_{ik} t_i = \sum_{i=1}^{N} t_i \sum_{k=1}^{M} a_{ik} = \sum_{i=1}^{N} t_i = T$$

Y la restricción a aplicar es

$$T \leq \sum_{k=1}^{M} c_k$$

Es decir, que el tamaño del sitio Web no debe sobrepasar la capacidad de almacenamiento agregada de todos los nodos servidores.

El aumento de la capacidad máxima de almacenamiento se puede conseguir, en este caso, de dos maneras: incrementando la capacidad de un nodo servidor ya existente o añadiendo un nuevo nodo servidor.

Una forma de incrementar la capacidad de un nodo servidor es agregando un disco adicional de capacidad ΔT (véase ecuación 5.5).

$$E = \frac{\Delta T}{\Delta T} = 1 \tag{5.5}$$

Si no es posible agregar un disco a un nodo ya existente, se puede incrementar la capacidad de un nodo, sustituyendo su disco de capacidad c_i por otro de capacidad $c_i + \Delta T$. No obstante debe tenerse en cuenta que puede existir un límite superior al tamaño de los discos que se pueden adquirir en un momento dado (ya sea por limitaciones técnicas o económicas). Esto

puede hacer que sea necesario sustituir varios discos. Si se considera que la capacidad agregada de los discos a sustituir es C, la eficacia del incremento vendrá dada por la ecuación 5.6.

$$E = \frac{\Delta T}{C + \Delta T} \tag{5.6}$$

Si todos los nodos servidores tienen discos con la capacidad máxima la única alternativa es añadir nuevos nodos servidores. Aparentemente la eficacia del incremento en este caso es

$$E = \frac{\Delta C}{\Delta C} = 1$$

pero debe tenerse en cuenta que además de adquirir un nuevo disco, es necesario adquirir todo el equipamiento del nodo servidor, lo que reduce drásticamente la eficacia de la operación. Por esta razón, esta es la última alternativa a considerar.

5.2.2.1 Ejemplo 1: Capacidades heterogéneas

En este caso se considera un sistema compuesto por 12 nodos servidores con la configuración presentada en la tabla 5.1.

En este caso el tamaño máximo del sitio Web es

$$T = \sum_{i=1}^{12} c_i = 1710\ GB$$

La eficacia del incremento en caso de incorporación de discos adicionales es siempre 1, tal y como se ha demostrado en la ecuación 5.5.

Para evaluar la eficacia del incremento en el caso de sustitución de discos se han considerado diversos tamaños máximos de discos que se pueden adquirir (entre 200 GB y 500 GB). La figura 5.3 muestra la evolución de la eficacia cuando aumenta el incremento del tamaño máximo.

Como puede observarse la eficacia del incremento de capacidad mejora cuanto mayor es la cantidad de capacidad incrementada. Por ejemplo, la eficacia de incrementar en 30 GB el tamaño máximo del sitio Web es 0,375, para un incremento de 50 GB es de 0,5 y para 150 GB es de 0,75.

Las curvas de eficacia son todas coincidentes para incrementos de hasta 150 GB. Este es el punto a partir del cual es necesario sustituir más de un disco cuando el tamaño máximo de disco es 200 GB (150 GB de incremento más 50 GB de tamaño del servidor s_1). Para dicho tamaño máximo (150 GB), los siguientes puntos de discontinuidad (en los que vuelve a bajar la eficacia son 270 GB y 390 GB.

Si se consideran discos con tamaño máximo de 300 GB, la caída de eficacia se produce para un incremento de 250 GB. Para discos con tamaño máximo de 400 GB, la caída se encuentra en un incremento de 350 GB.

Estas caídas en la eficacia del incremento se producen justo en los puntos en los que la limitación del tamaño máximo de disco hace que sea necesario realizar las sustituciones de discos en más nodos servidores.

Es interesante observar que la tendencia general es que la eficacia del incremento por sustitución se aproxime al valor 1 cuando crece el incremento del tamaño máximo del sitio Web.

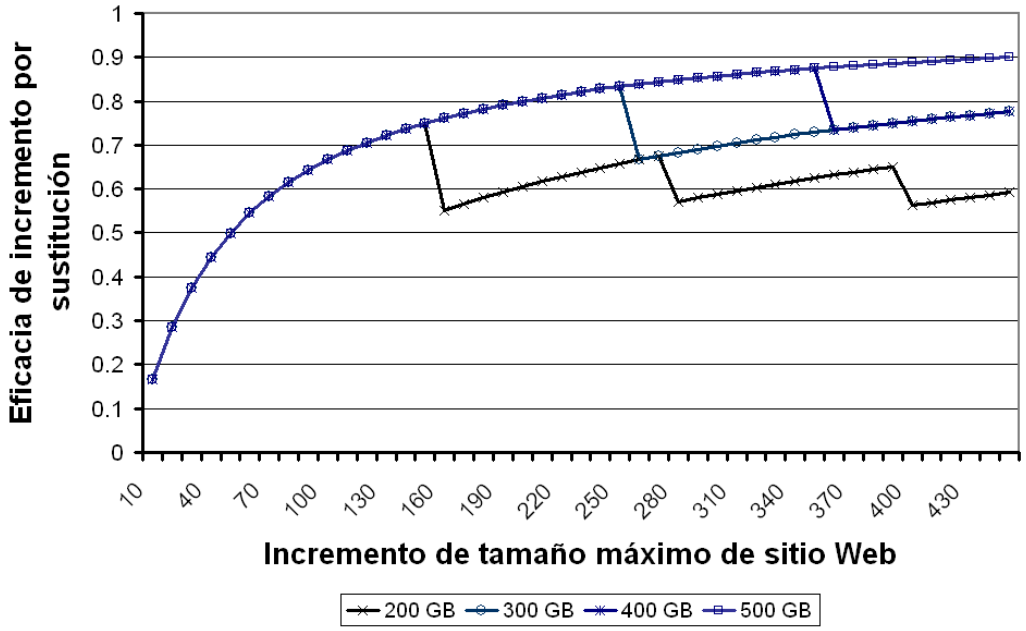

Figura 5.3: Coste del incremento de la capacidad en caso de alojamiento totalmente distribuido.

5.2.2.2 Ejemplo 2: Capacidad homogénea

En este ejemplo se considera un sistema compuesto por 12 nodos servidores, todos con una capacidad inicial de 150 GB.

El tamaño máximo del sitio Web alojado será

$$T = \sum_{i=1}^{M} c = Mc = 12 \cdot 150 = 1800\ GB$$

La eficacia de la incorporación de discos adicionales es siempre 1, tal y como se ha demostrado en la ecuación 5.5.

Para evaluar la eficacia del incremento en el caso de sustitución de discos se han considerado diversos tamaños máximos de discos que se puede adquirir (entre 200 GB y 500 GB). La Figura 5.4 muestra la evolución de la eficacia cuando aumenta el incremento del tamaño máximo.

Los resultados obtenidos son similares a los obtenidos en el ejemplo anterior (véase 5.2.2.1).

De forma general la eficacia crece con el incremento del tamaño máximo del sitio Web. Al igual que en el ejemplo anterior se producen caídas en la eficacia cuando es necesaria la sustitución de un mayor número de discos.

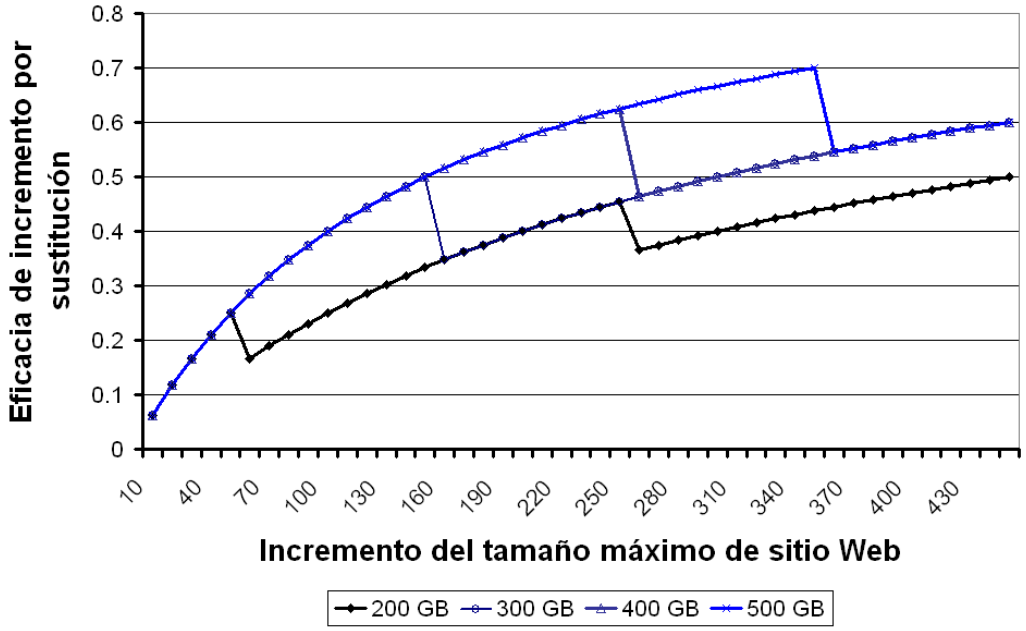

Figura 5.4: Coste del incremento de la capacidad en caso de alojamiento totalmente distribuido con capacidad inicialmente homogénea.

5.2.3 Replicación parcial

En el caso de la replicación parcial cada elemento está asignado a varios nodos servidores. Es decir,

$$\sum_{i=1}^{N} a_{ij} < N \ \forall j \in [1, M] \qquad \sum_{j=1}^{M} a_{ij} = r_i \ \forall i \in [1, N]$$

donde r_i es el número de réplicas para el elemento e_i, y se cumple

$$1 \leq r_i \leq M$$

Si se sustituye en la ecuación 5.4,

$$V = \sum_{i=1}^{N} t_i \sum_{j=1}^{M} a_{ij} = \sum_{i=1}^{N} r_i t_i$$

Obsérvese que el volumen de ocupación es menor que el que necesita el caso de replicación total, pero mayor que el que necesita la distribución total.

$$MT \leq \sum_{i=1}^{N} r_i t_i \leq T$$

Por otra parte el tamaño máximo que puede tener el sitio Web debe cumplir la restricción

$$\sum_{i=1}^{N} r_i t_i \leq \sum_{j=1}^{M} c_j$$

En el caso general en que se asigna un número de réplicas distinto a cada elemento, un incremento del espacio disponible (por ejemplo incrementando la capacidad de un nodo), da lugar a la posibilidad de incrementar el número de réplicas (véase 4.2.3).

Un caso particular de interés, es aquel en que se utilizan igual número de réplicas para todos los elementos ($r_i = r \ \forall i = 1, \ldots, N$). En dicho caso se simplifica tanto la expresión para el volumen de ocupación, como la restricción de tamaño máximo

$$V = \sum_{i=1}^{N} r t_i = r \sum_{i=1}^{N} t_i = rT$$

$$T \leq \frac{1}{r} \sum_{j=1}^{M} c_j$$

Al igual que en el caso de la distribución total, se puede conseguir incrementar la capacidad de almacenamiento por dos vías: realizando la sustitución de discos en algún nodo existente o incorporando nuevos nodos servidores.

El incremento de la capacidad de un nodo servidor se puede realizar agregando un disco adicional. Si se desea incrementar la capacidad global del servidor en ΔT, se debe incorporar un disco de capacidad $r\Delta T$. En este caso la eficacia del incremento es

$$E = \frac{\Delta T}{r\Delta T} = \frac{1}{r}$$

Si no es posible agregar un disco a un nodo ya existente, se puede incrementar la capacidad de un nodo, sustituyendo su su disco de capacidad c_i por otro de capacidad $c_i + r\Delta T$. Si se supera el tamaño máximo del disco que se puede adquirir, se hará necesaria la sustitución de varios discos de capacidad agregada C. En este caso la eficacia de la sustitución es

$$E = \frac{\Delta T}{C + r\Delta T}$$

Por último, y por el mismo razonamiento de la sección 5.2.2, la opción de incrementar la capacidad mediante la incorporación de nuevos nodos servidores debe considerarse como la última opción.

5.2.3.1 Ejemplo 1: Capacidades heterogéneas

En este caso se considera un sistema compuesto por 12 nodos servidores con la configuración de la tabla 5.1 y un mecanismo de asignación con igual número de réplicas a todos los elementos.

El tamaño máximo del sitio Web es

$$T = \frac{1}{r}\sum_{i=1}^{12} c_i = \frac{1}{r}1710\ GB$$

Obsérvese que el valor obtenido es una cota superior del tamaño máximo, ya que este puede ser menor debido a la propia distribución de tamaños individuales de los elementos. La Figura 5.5 muestra la evolución del tamaño máximo del sitio Web cuando se varía el número de réplicas por elemento. No se muestran en la gráfica los casos $r = 1$ (contenidos totalmente distribuidos) ni $r = 12$ (contenidos totalmente replicados).

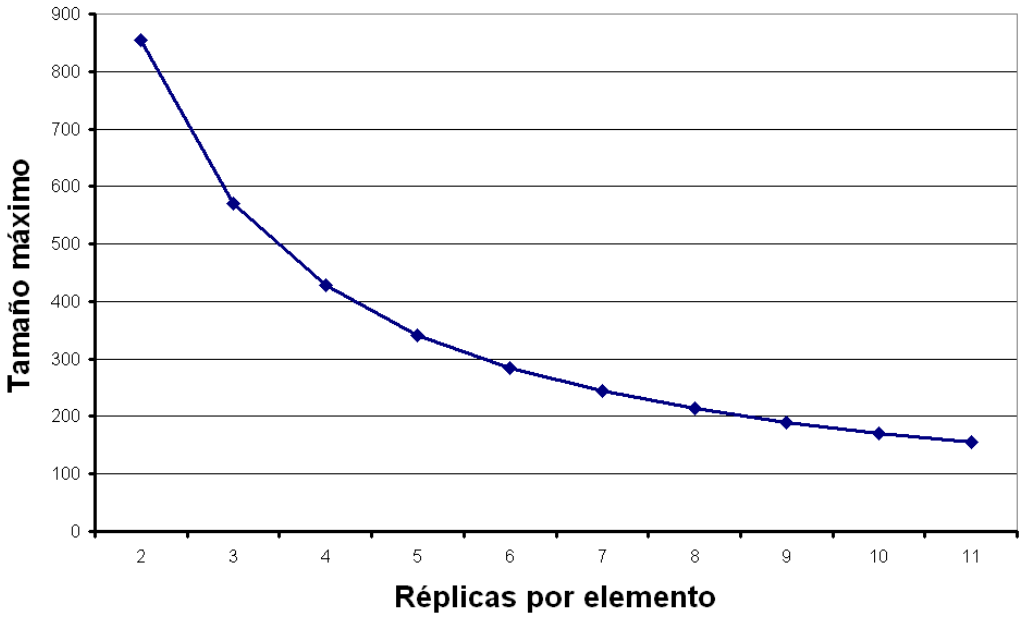

Figura 5.5: Tamaño máximo del sitio Web que se puede alojar con replicación parcial.

En caso de incorporar discos adicionales a los nodos servidores existentes, la eficacia del incremento de capacidad es

$$E = \frac{1}{r}$$

y en el caso de realizar sustitución de discos existentes, la eficacia del incremento de capacidad es

$$E = \frac{\Delta T}{C + r\Delta T}$$

La Figura 5.6 muestra la eficacia de la sustitución para los casos de 2, 3 y 4 réplicas por elemento, con tamaño máximo de disco de 300 GB.

Como puede observarse, la eficacia del incremento en caso de sustitución aumenta cuanto mayor es el volumen de incremento. Las discontinuidades se producen justo cuando es necesario incorporar un nuevo disco. Para cada caso, la eficacia siempre queda ligeramente por

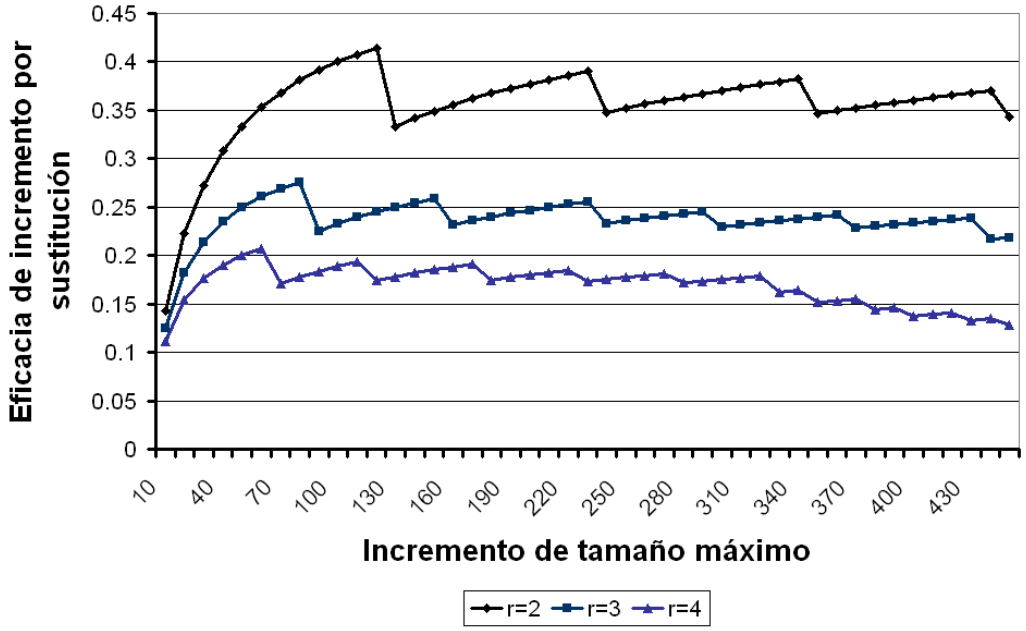

Figura 5.6: Coste del incremento de la capacidad en caso de alojamiento parcialmente replicado.

debajo del valor de eficacia $1/r$ que se obtendría si se realizase incorporación de nuevos discos (1/2, 1/3 y 1/4 respectivamente).

5.2.3.2 Ejemplo 2: Capacidades homogéneas

En este ejemplo se considera un sistema compuesto por 12 nodos servidores, todos con una capacidad inicial de 150 GB.

El tamaño máximo del sitio Web es

$$T = \frac{1}{r}\sum_{i=1}^{12} c_i = \frac{1}{r}1800\ GB$$

Al igual que en el caso anterior, el valor obtenido es una cota superior del tamaño máximo, debido a la propia distribución de tamaños individuales de los nodos.

En caso de incorporar discos adicionales a los nodos servidores existentes, la eficacia del incremento de capacidad es

$$E = \frac{1}{r}$$

y en el caso de realizar sustitución de discos existentes, la eficacia del incremento de capacidad es

$$E = \frac{\Delta T}{C + r\Delta T}$$

La Figura 5.7 muestra la eficacia de la sustitución para los casos de 2, 3 y 4 réplicas por elemento, con tamaño máximo de disco de 300 GB.

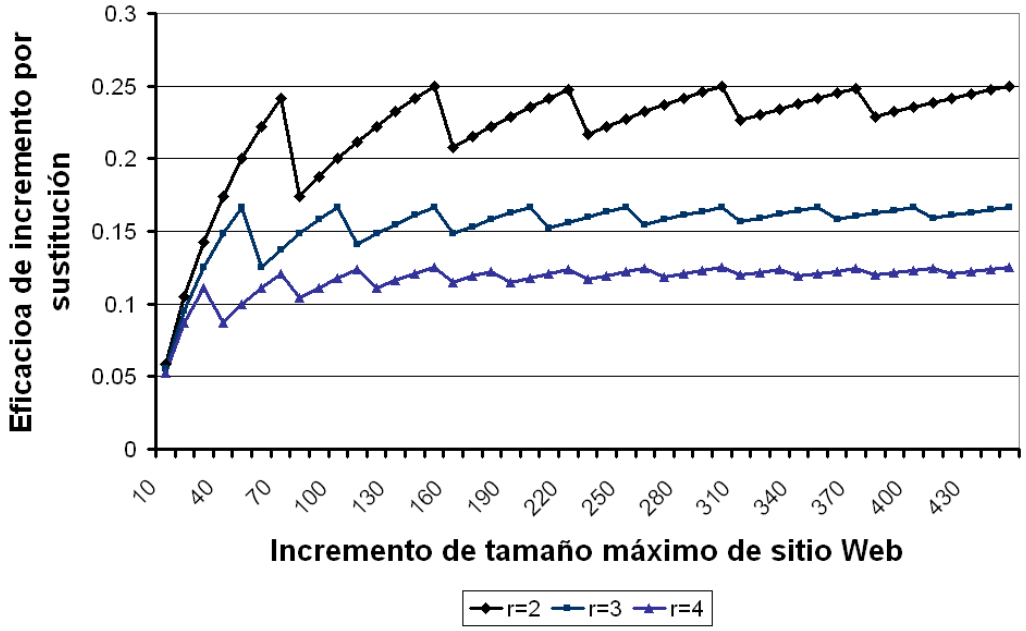

Figura 5.7: Coste del incremento de la capacidad en caso de alojamiento parcialmente replicado con capacidades iniciales homogéneas.

5.2.4 Comparación de la capacidad de almacenamiento

Las distintas estrategias de alojamiento de contenidos (replicación total, replicación parcial, distribución total) dan lugar a distintas necesidades de almacenamiento e imponen distintas restricciones sobre el tamaño máximo del sitio Web que puede ser alojado (véase tabla 5.2).

	Volumen	**Tamaño Máximo**
Replicación Total	MT	$T \leq \min_{k=1,\ldots,M} c_k$
Replicación Parcial	rT	$T \leq \frac{1}{r} \sum_{j=1}^{M} c_j$
Distribución total	T	$T \leq \sum_{j=1}^{M} c_j$

Tabla 5.2: Comparación del volumen de almacenamiento de un sitio Web y su tamaño máximo.

La estrategia que mayor volumen de ocupación genera es la replicación total y la que menor volumen genera es la distribución total. La replicación parcial varía entre ambos valores extremos dependiendo del índice de replicación.

5.2.4.1 Ejemplo 1: Capacidades heterogéneas

En este caso se considera un sistema compuesto por 12 nodos servidores con la configuración presentada en la tabla 5.1.

La Figura 5.8 muestra el tamaño máximo del sitio Web que puede alojarse en el sistema ejemplo, cuando se aplica una estrategia de replicación total, distribución total o replicación parcial (para distintos valores de r).

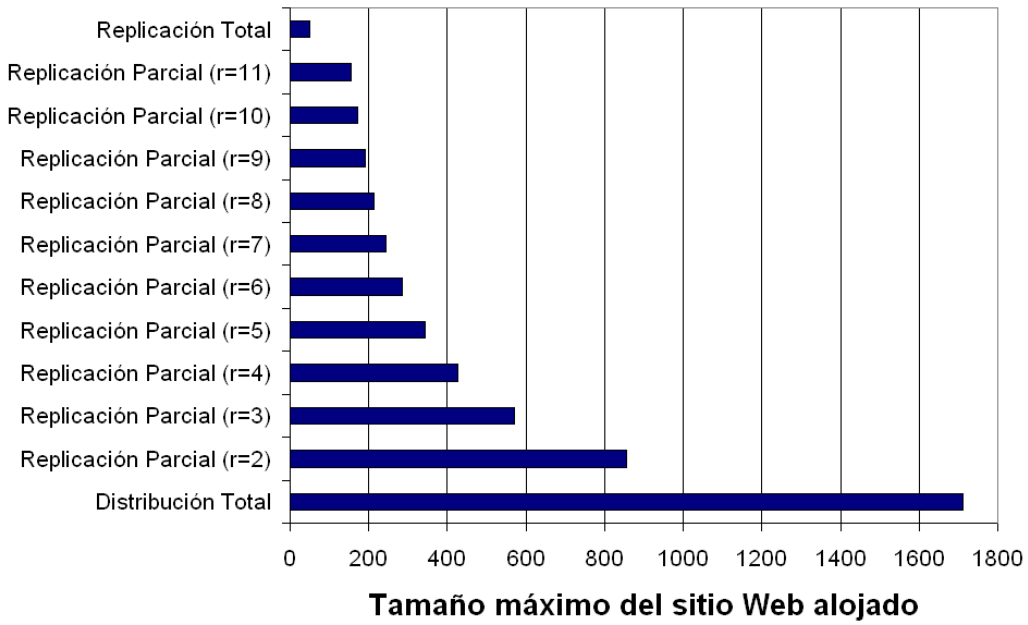

Figura 5.8: Tamaño máximo del sitio Web que puede alojarse dependiendo de la estrategia de replicación seleccionada para sistema con capacidades heterogéneas.

El tamaño máximo que se puede alcanzar es mucho mayor en el caso de la distribución total. En el otro extremo está la distribución total que ofrece un tamaño muchísimo menor. Las soluciones de replicación parcial son, desde este punto de vista, aproximaciones intermedias en cuanto al tamaño máximo que se puede alcanzar.

En cuanto a la eficacia del incremento cuando se realiza la incorporación de nuevos discos, la Figura 5.9 muestra su valor para las distintas estrategias de replicación. Se ha considerado en todos los casos que el tamaño de disco de mayor capacidad que se puede adquirir es de 300 GB.

En el caso de distribución total, la eficacia de la incorporación de espacio adicional de almacenamiento es siempre 1, puesto que todo el espacio incorporado se utiliza sin necesidad de almacenar copias adicionales de los elementos. Para los casos de replicación parcial con bajo número de réplicas por elemento ($r \leq 4$), la eficacia es siempre $1/r$. Es decir, que la eficacia depende exclusivamente del número de réplicas por elemento. Para $r > 4$, entra en juego el tamaño máximo del disco que se puede adquirir (300 GB en este caso), que hace que la eficacia, a partir de un cierto punto, pase a ser 0.

La Figura 5.10 muestra la evolución de la eficacia por sustitución para distintos incremen-

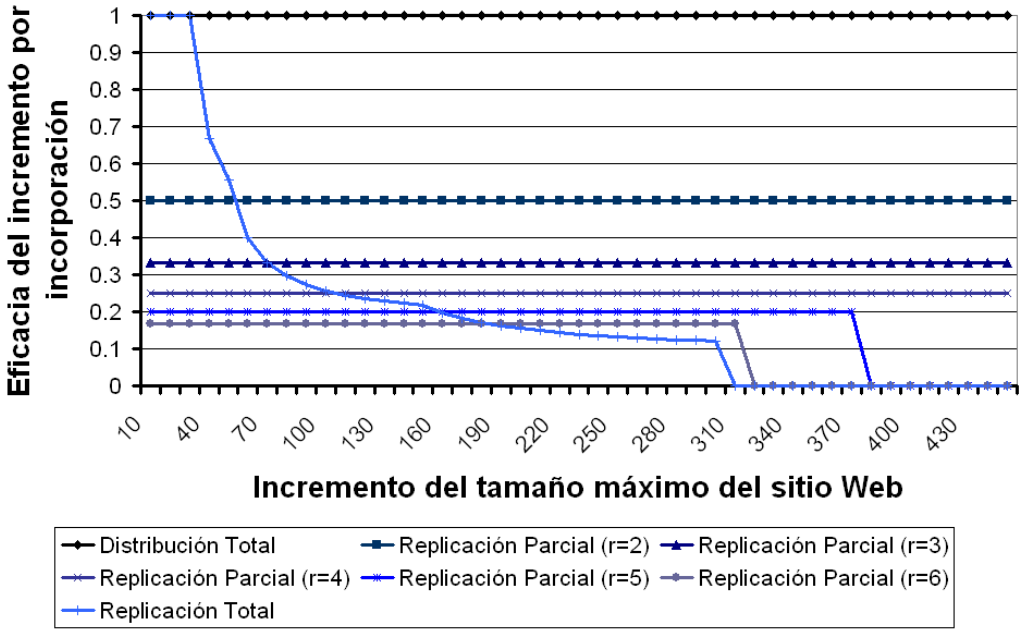

Figura 5.9: Eficacia del incremento de la capacidad por incorporación partiendo de capacidades iniciales heterogéneas para distintas estrategias de replicación.

tos del tamaño máximo del sitio Web. En todos los casos se ha considerado que el tamaño del disco de mayor capacidad que se podía adquirir era de 300 GB.

Puede observarse que la mayor eficacia se obtiene siempre en el caso de la distribución total, puesto que en este caso se produce un aprovechamiento máximo de los recursos.

Aunque la replicación total ofrece inicialmente mejores resultados que la replicación parcial, esto solamente ocurre para valores pequeños en los que se puede incrementar la capacidad del sistema totalmente replicado sustituyendo únicamente un disco. A partir de un cierto punto (entre los 40 y los 60 GB, en el ejemplo), las estrategias de replicación parcial presentan una eficacia superior a la estrategia de replicación total.

Por otra parte, la estrategia de replicación total no es en absoluto escalable. La limitación del tamaño máximo de disco hace que a partir de un cierto punto (250 GB en el ejemplo), la eficacia pase a ser 0, puesto que no es posible incrementar el espacio disponible.

5.2.4.2 Ejemplo 2: Capacidades homogéneas

En este ejemplo se considera un sistema compuesto por 12 nodos servidores, todos con una capacidad inicial de 150 GB.

La Figura 5.11 muestra el tamaño máximo del sitio Web que puede alojarse en el sistema ejemplo, cuando se aplica una estrategia de replicación total, distribución total o replicación parcial (para distintos valores de r).

El tamaño máximo que se puede alcanzar es mucho mayor en el caso de la distribución total. En el otro extremo está la distribución total que ofrece un tamaño muchísimo menor. Las soluciones de replicación parcial son, desde este punto de vista, aproximaciones intermedias

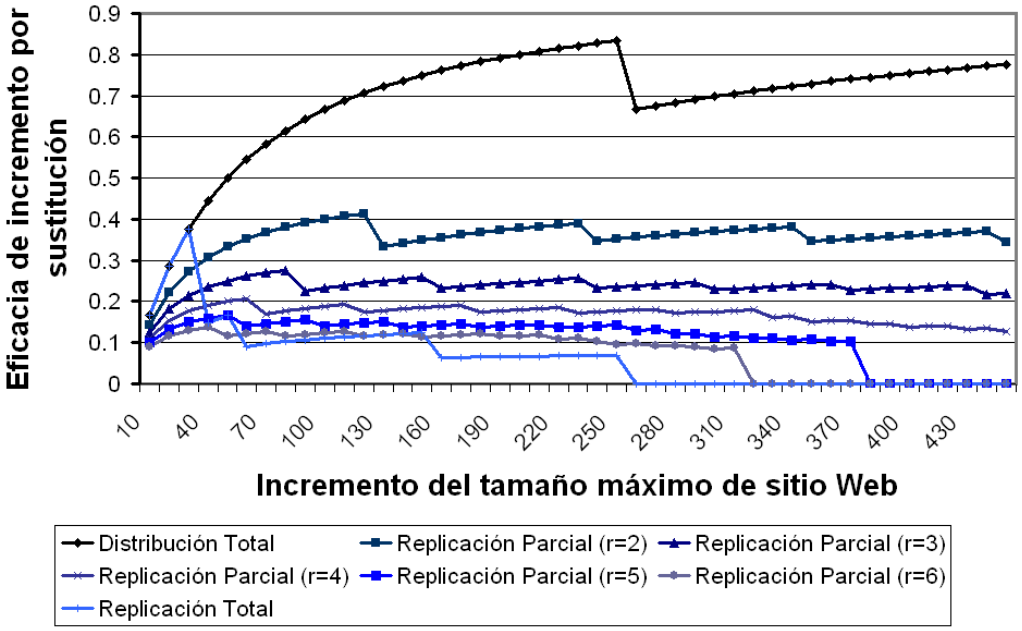

Figura 5.10: Eficacia del incremento de la capacidad por sustitución con capacidades iniciales heterogéneas para distintas estrategias de replicación.

en cuanto al tamaño máximo que se puede alcanzar.

La Figura 5.12 muestra la evaluación de la eficacia de incremento cuando se realiza la incorporación de nuevos discos. En todos los casos se ha considerado que el tamaño del disco de mayor capacidad que se puede adquirir es de 300 GB.

En todos los casos la eficacia se mantiene constante y depende exclusivamente de la estrategia de replicación de contenidos utilizada. En el caso de la replicación total, existe un límite impuesto por el tamaño máximo del disco, a partir del cual no es posible incorporar más volumen de almacenamiento.

La evaluación de la eficacia por sustitución se muestra en la Figura 5.13 para distintos incrementos del tamaño máximo del sitio Web. En todos los casos se ha considerado que el tamaño del disco de mayor capacidad que se puede adquirir es de 300 GB.

Puede observarse claramente, una vez más, que los resultados de la evaluación dejan claro la superioridad de la distribución total sobre el resto de alternativas en cuanto a eficacia del incremento. La estrategia menos eficaz vuelve a ser la replicación total. Las estrategias de replicación parcial se mantienen a mitad de camino.

5.2.4.3 Resumen de la comparación

Las evaluaciones realizadas muestran que si se comparan todas las estrategias de replicación, partiendo de un mismo conjunto de recursos de almacenamiento, la replicación total es la solución que da lugar a un menor tamaño máximo del sitio Web alojado. En el otro extremo se encuentra la distribución total. Otra vez, la replicación parcial varía entre ambos valores extremos dependiendo del índice de replicación.

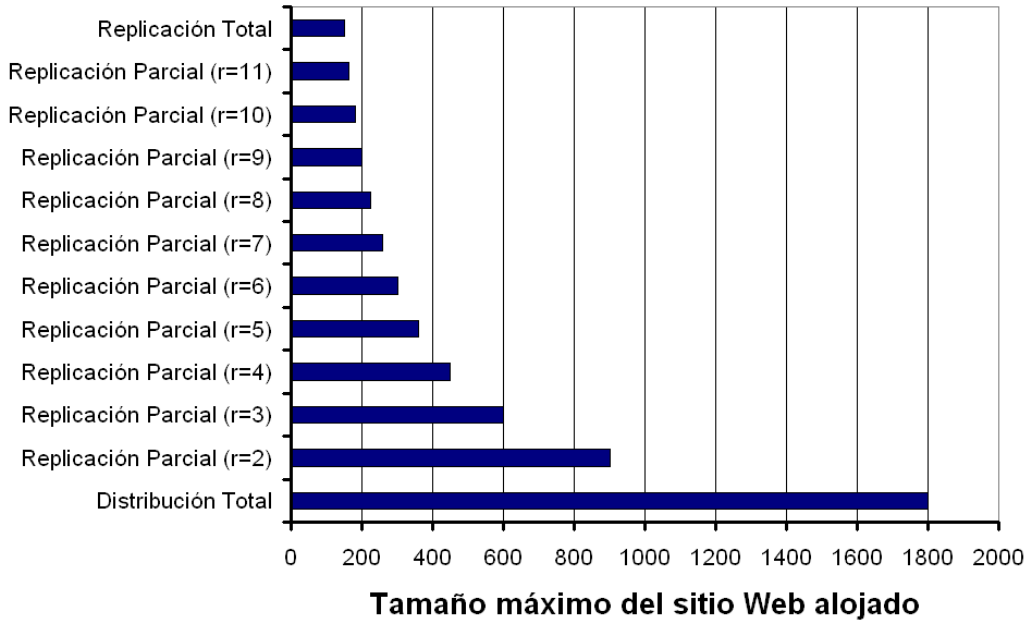

Figura 5.11: Tamaño máximo del sitio Web que puede alojarse dependiendo de la estrategia de replicación seleccionada para sistema con capacidades homogéneas.

Cuando se trata de incrementar el tamaño máximo del sitio Web que puede ser alojado, la replicación total es la estrategia más problemáticas. Por una parte, su coste es mayor que el de cualquier otra estrategia. Por otra parte, el tamaño máximo estará siempre limitado por el tamaño del disco más grande que se pueda adquirir y es independiente del número de nodos del servidor Web. En el otro extremo, la distribución total tiene costes relativamente bajos a la hora de incrementar la capacidad global. A medio camino se encuentran las soluciones de replicación parcial.

Si se prescinde de otras consideraciones (como la tolerancia a fallos o la fiabilidad), la solución más deseable desde el punto de vista de la capacidad de almacenamiento es la distribución total.

Cuando se realiza un incremento en la capacidad de un sitio Web se puede elegir entre tres alternativas:

- Incorporación de un disco adicional a un nodo servidor existente.
- Sustitución de un disco en un nodo servidor existente.
- Incorporación de un nuevo nodo servidor.

Desde el punto de vista del incremento de la capacidad de almacenamiento, la alternativa más eficaz es siempre la incorporación de un disco adicional. En ciertos casos, esta alternativa puede no ser viable. Esto sucede, por ejemplo, cuando todos los nodos servidores tienen conectados el máximo de discos que soporta su arquitectura.

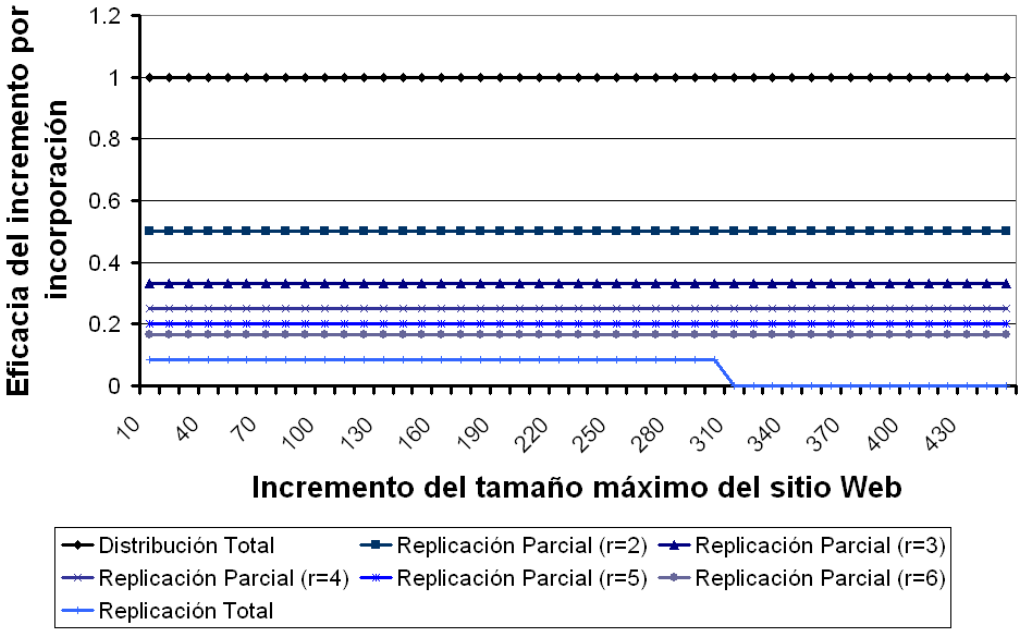

Figura 5.12: Eficacia del incremento de la capacidad por incorporación partiendo de capacidades iniciales homogéneas para distintas estrategias de replicación.

En dicho caso, se debe optar por la siguiente alternativa, en términos de eficacia: la sustitución de un disco existente. Para maximizar la eficacia, se debe optar siempre por sustituir el disco de menor capacidad. Otra vez, puede darse el caso de que esta alternativa tampoco sea viable. Esto ocurre cuando todos los discos de todos los nodos servidores son de la máxima capacidad que puede adquirirse (ya sea por razones técnicas o económicas).

La última alternativa a considerar será la incorporación de nuevos nodos servidores al sistema. En cuanto a esta alternativa, es solamente válida para estrategias parcialmente replicadas o totalmente distribuidas.

5.3 Evaluación de la fiabilidad

La fiabilidad de un servidor Web distribuido se puede evaluar como la probabilidad de que el servidor Web funcione correctamente. Es decir, la fiabilidad es la probabilidad de poder obtener un elemento del servidor Web distribuido. La fiabilidad de un servidor Web distribuido depende de la arquitectura utilizada y de la estrategia de replicación seleccionada.

La fiabilidad F_s de un sistema compuesto por N componentes conectados en serie [78] y cada uno con fiabilidad F_i viene dada por la ecuación 5.7.

$$F_s = \prod_{i=1}^{N} F_i \tag{5.7}$$

De la misma forma, la fiabilidad F_p de un sistema compuesto por N componentes conec-

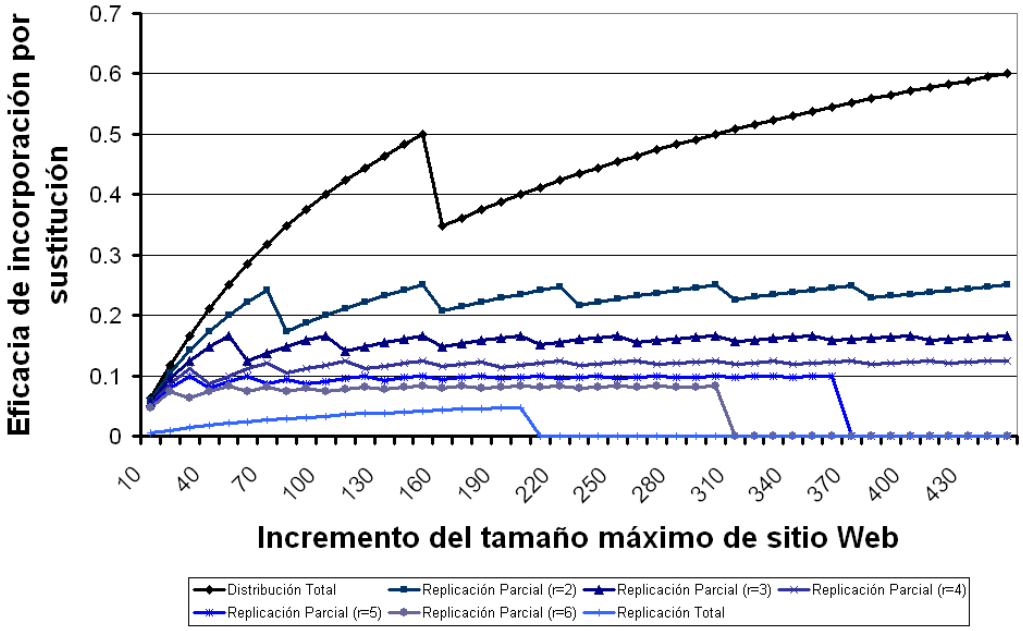

Figura 5.13: Eficacia del incremento de la capacidad por sustitución con capacidades iniciales homogéneas para distintas estrategias de replicación.

tados en paralelo [78] y cada uno con fiabilidad F_i viene dada por la ecuación 5.8.

$$F_p = 1 - \prod_{i=1}^{N} (1 - F_i) \tag{5.8}$$

En las siguientes secciones se evalúa la fiabilidad de las distintas opciones arquitectónicas y las distintas estrategias de replicación de contenidos.

Primeramente se evalúa la fiabilidad de un sistema Web basado en *cluster* por tratarse de la arquitectura base seleccionada en el Capítulo 3. Para esta arquitectura se estudia el impacto que tiene en la fiabilidad la utilización de las distintas estrategias de replicación de contenidos.

Posteriormente se realiza la evaluación de la fiabilidad de un *cluster* Web con *switch* distribuido, que es la propuesta arquitectónica que se presenta en esta tesis. De nuevo, se estudia el impacto que tiene en la fiabilidad la utilización de las distintas estrategias de replicación de contenidos.

5.3.1 Fiabilidad de Sistemas Web basados en *cluster*

En un sistema Web basado en *cluster* existen dos elementos importantes: el *switch* Web y los nodos servidores. En este caso el *switch* y los nodos servidores forman un sistema serie. Por tanto, la fiabilidad del sistema [78] mediante la ecuación ecuación 5.9.

$$F_{servidor} = F_{switch} F_{nodos} \tag{5.9}$$

La fiabilidad del *switch* es siempre la misma. Sin embargo, la fiabilidad del conjunto de nodos servidores depende de la estrategia de replicación utilizada.

5.3.1.1 Replicación total

Si todos los contenidos están totalmente replicados, el conjunto de nodos servidores se puede ver como un sistema paralelo redundante [78].

La fiabilidad de un conjunto de nodos se puede expresar en términos de la fiabilidad de cada nodo.

$$F_{nodos} = 1 - \prod_{i=1}^{M} (1 - F_{nodo_i})$$

La fiabilidad del servidor se obtiene sustituyendo en la ecuación 5.9, dando lugar a la ecuación 5.10.

$$F_{servidor} = F_{switch} \left(1 - \prod_{i=1}^{M} (1 - F_{nodo_i}) \right) \tag{5.10}$$

Si se asume que todos los nodos tienen la misma fiabilidad F, la fiabilidad del servidor vendrá dada por la ecuación 5.11.

$$F_{cluster} = F_{switch} \left(1 - (1 - F)^M \right) \tag{5.11}$$

5.3.1.2 Distribución total

Si todos los contenidos están totalmente distribuidos, la fiabilidad del sistema depende de la probabilidad de que un elemento esté alojado en un determinado nodo. Sea el suceso E_i

$E_i \equiv$ El elemento solicitado se encuentra en el nodo s_i

La probabilidad de que el elemento solicitado se encuentre en un determinado nodo s_i determina la contribución que la fiabilidad de cada nodo realiza a la fiabilidad del conjunto de nodos

$$F_{nodos} = 1 - \sum_{i=1}^{M} P(E_i) (1 - F_{nodo_i})$$

La fiabilidad del servidor se obtiene sustituyendo en la ecuación 5.9, dando lugar a la ecuación 5.12.

$$F_{servidor} = F_{switch} \left(1 - \sum_{i=1}^{M} P(E_i) (1 - F_{nodo_i}) \right) \tag{5.12}$$

Si se asume que la distribución de todos los elementos es uniforme se tiene que

$$P(E_i) = \frac{1}{M} \; \forall i \in [1, M]$$

Si además, se asume que todos los nodos servidores tienen la misma fiabilidad F, se tiene que

$$F_{nodos} = \left(1 - \sum_{i=1}^{M} \frac{1}{M}(1-F)\right) = (1-(1-F)) = F$$

La fiabilidad del servidor se obtiene sustituyendo en la ecuación 5.9, dando lugar a la ecuación 5.13.

$$F_{servidor} = F_{switch} F \tag{5.13}$$

Es decir, que en el caso de servidores Web totalmente distribuidos, la fiabilidad es independiente del número de nodos servidores y depende exclusivamente de la fiabilidad del switch y de la fiabilidad de los propios nodos.

5.3.1.3 Replicación parcial

Si los contenidos están parcialmente replicados, será necesario considerar el número de réplicas que se realiza de cada elemento. Si se tienen r réplicas de un elemento, distribuidas en M nodos, para determinar la probabilidad de que no se pueda acceder a dicho elemento se deben considerar todas las combinaciones de replicación y la probabilidad de que ocurra un fallo en todos los nodos en los que se encuentra replicado el elemento.

$$F_{nodos} = 1 - \sum_{\substack{i_1,i_2,\ldots,i_r=1 \\ i_1<i_2<\ldots<i_r}}^{M} P\left(\bigcap_{j=i_1}^{i_r} E_j\right) \prod_{j=i_1}^{i_r} \left(1 - F_{nodo_j}\right)$$

La fiabilidad del servidor se obtiene sustituyendo en la ecuación 5.9, dando lugar a la ecuación 5.14.

$$F_{servidor} = F_{switch}\left(1 - \sum_{\substack{i_1,i_2,\ldots,i_r=1 \\ i_1<i_2<\ldots<i_r}}^{M} P\left(\bigcap_{j=i_1}^{i_r} E_j\right) \prod_{j=i_1}^{i_r} \left(1 - F_{nodo_j}\right)\right) \tag{5.14}$$

Si se asume que la distribución de réplicas es uniforme entre todos los nodos servidores, se puede expresar la probabilidad de que las réplicas se encuentren en un determinado subconjunto de nodos como

$$P\left(\bigcap_{j=i_1}^{i_r} E_j\right) = \frac{1}{C_{M,r}} = \frac{1}{\binom{r}{M}}$$

Si además se asume que la fiabilidad de todos los nodos es idéntica e igual a F, se tiene que

$$F_{nodos} = 1 - \sum_{\substack{i_1,i_2,\ldots,i_r=1 \\ i_1<i_2<\ldots<i_r}}^{M} \frac{1}{\binom{r}{M}} \prod_{j=i_1}^{i_r} (1-F) = 1 - \frac{1}{\binom{r}{M}} \sum_{\substack{i_1,i_2,\ldots,i_r=1 \\ i_1<i_2<\ldots<i_r}}^{M} (1-F)^r$$

pero el número de sumas que hay en la expresión es exactamente $C_{M,r}$, por lo que

$$F_{nodos} = 1 - \frac{1}{\binom{r}{M}} \binom{r}{M} (1-F)^r = 1 - (1-F)^r$$

Por tanto, la fiabilidad del servidor viene dada por la ecuación 5.15.

$$F_{servidor} = F_{switch} \left(1 - (1-F)^r\right) \tag{5.15}$$

5.3.1.4 Comparación

La Figura 5.14 muestra la fiabilidad alcanzada por un sistema Web basado en *cluster* para distintos valores de fiabilidad. Para realizar la evaluación se ha considerado un sistema compuesto por 12 nodos y un *switch Web*, todos con la misma fiabilidad. La asignación de réplicas se ha supuesto equitativa. El caso del servidor totalmente replicado es por tanto aquél en el que número de réplicas es 12. El caso del servidor totalmente distribuido es aquél en que el número de réplicas es 1.

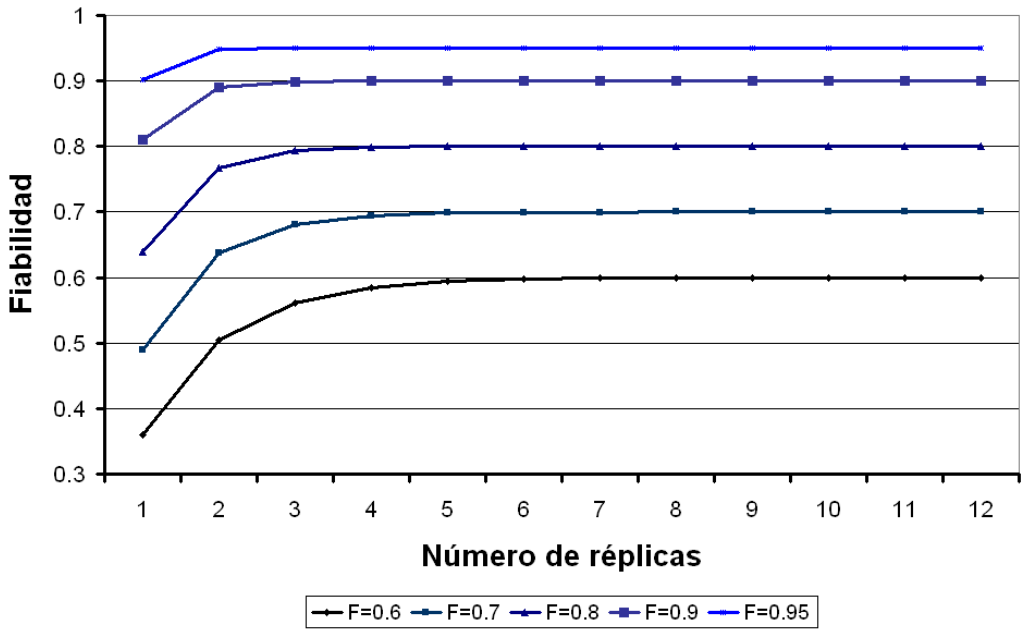

Figura 5.14: Fiabilidad de un sistema Web basado en *cluster* según el número de réplicas.

La primera observación que puede hacerse es que *la fiabilidad del sistema está limitada por la fiabilidad del switch Web y no puede ser mayor que ésta*. Esto es algo que se podía intuir, puesto que el *switch* Web supone un punto de fallo único (si el *switch* falla el sistema falla).

Otra observación interesante, es que *el número de réplicas tiene un gran impacto sobre la fiabilidad del sistema*. De hecho, *con un número de réplicas relativamente bajo se puede alcanzar una fiabilidad equivalente a la que tiene un sistema totalmente replicado.*

5.3.2 *Cluster* Web con *switch* distribuido

Un *cluster* Web con *switch* distribuido puede considerarse como la composición en serie de dos sistemas, por lo que la fiabilidad del *cluster* viene dada por el producto de fiabilidades (véase ecuación 5.16).

$$F_{web} = F_{switches} F_{nodos} \tag{5.16}$$

En dicha ecuación, $F_{switches}$ es la fiabilidad del conjunto de *switches* Web y F_{nodos} es la fiabilidad del conjunto de nodos servidores.

El conjunto de *switches* Web se puede ver como un sistema paralelo redundante. Esto es independiente del nivel de replicación de los contenidos, ya que cualquier *switch* puede redirigir cualquier petición a cualquier nodo servidor. Por tanto, si un sistema tiene P *switches*,

$$F_{switches} = 1 - \prod_{i=1}^{P} (1 - F_{switch_i})$$

La fiabilidad del conjunto de nodos será distinta dependiendo del nivel de replicación de los elementos.

5.3.2.1 Replicación total

En el caso en que los elementos estén totalmente replicados, la fiabilidad del conjunto de nodos será

$$F_{nodos} = 1 - \prod_{i=1}^{M} (1 - F_{nodo_i})$$

Y la fiabilidad del sistema vendrá dada por la ecuación 5.17.

$$F_{servidor} = \left(1 - \prod_{i=1}^{P} (1 - F_{switch_i})\right) \left(1 - \prod_{i=1}^{M} (1 - F_{nodo_i})\right) \tag{5.17}$$

Si se asume que la fiabilidad de todos los nodos es igual y toma el valor F_{nodo}, y que la fiabilidad de todos los *switches* es igual y toma el valor F_{switch}, la fiabilidad del sistema seguirá la ecuación 5.18.

$$F_{servidor} = \left(1 - (1 - F_{switch})^P\right) \left(1 - (1 - F_{nodo})^M\right) \tag{5.18}$$

5.3.2.2 Distribución total

En el caso en que los contenidos están totalmente distribuidos, la fiabilidad del conjunto de nodos depende de la probabilidad de que un elemento esté alojado en un determinado nodo.

$$F_{nodos} = 1 - \sum_{i=1}^{M} P(E_i) (1 - F_{nodo_i})$$

La fiabilidad del servidor se obtiene sustituyendo en la ecuación 5.16, dando lugar a la ecuación 5.19.

$$F_{servidor} = \left(1 - \prod_{i=1}^{P} \left(1 - F_{switch_i}\right)\right) \left(1 - \sum_{i=1}^{M} P\left(E_i\right) \left(1 - F_{nodo_i}\right)\right) \tag{5.19}$$

Si se asume que la distribución de todos los elementos entre los nodos es uniforme, se tiene que

$$P\left(E_i\right) = \frac{1}{M}$$

Si además se asume que todos los *switches* tienen la misma fiabilidad F_{switch}, y que todos los nodos servidores tienen la misma fiabilidad F_{nodo}, se tiene que la fiabilidad del servidor se puede expresar mediante la ecuación 5.20.

$$F_{servidor} = \left(1 - \left(1 - F_{switch}\right)^{P}\right) F_{nodo} \tag{5.20}$$

5.3.2.3 Replicación parcial

En el caso en que la distribución de contenidos sea parcial, se debe considerar el número de réplicas por elemento r. En este caso se deben considerar las posibles combinaciones de distribución de réplicas, así como la probabilidad de que fallen todos los nodos donde se encuentran distribuidas dichas réplicas. La fiabilidad del conjunto de nodos viene dada por la ecuación 5.21.

$$F_{nodos} = 1 - \sum_{\substack{i_1, i_2, \ldots, i_r = 1 \\ i_1 < i_s < \ldots < i_r}}^{M} P\left(\bigcap_{j=i_1}^{i_r} E_j\right) \prod_{j=i_1}^{i_r} \left(1 - F_{nodo_j}\right) \tag{5.21}$$

La fiabilidad del servidor se obtiene sustituyendo en la ecuación 5.16, dando lugar a la ecuación 5.22.

$$F_{servidor} = \left(1 - \prod_{i=1}^{p} \left(1 - F_{switch_i}\right)\right) \left(1 - \sum_{\substack{i_1, i_2, \ldots, i_r = 1 \\ i_1 < i_s < \ldots < i_r}}^{M} P\left(\bigcap_{j=i_1}^{i_r} E_j\right) \prod_{j=i_1}^{i_r} \left(1 - F_{nodo_j}\right)\right) \tag{5.22}$$

En el caso en que la distribución de todos los elementos sea uniforme y todos los nodos servidores tengan la misma fiabilidad F_{nodo} se tiene que

$$F_{nodos} = 1 - \left(1 - F_{nodo}\right)^{r}$$

Dando lugar a la ecuación 5.23 como expresión de la fiabilidad.

$$F_{servidor} = \left(1 - \left(1 - F_{switch}\right)^{P}\right) 1 - \left(1 - F_{nodo}\right)^{r} \tag{5.23}$$

5.3.2.4 Comparación

La Figura 5.15 muestra la fiabilidad alcanzada por un *cluster* Web con *switch* distribuido para distintos valores de fiabilidad. Para realizar la evaluación se ha considerado un sistema compuesto por 12 nodos servidores y 2 *switches* Web. En todos los casos se ha supuesto que la fiabilidad de los *switches* y de los nodos servidores es igual. El caso del servidor totalmente replicado viene representado por el valor de 12 réplicas. El caso del servidor totalmente distribuido está representado por el valor de 1 réplica.

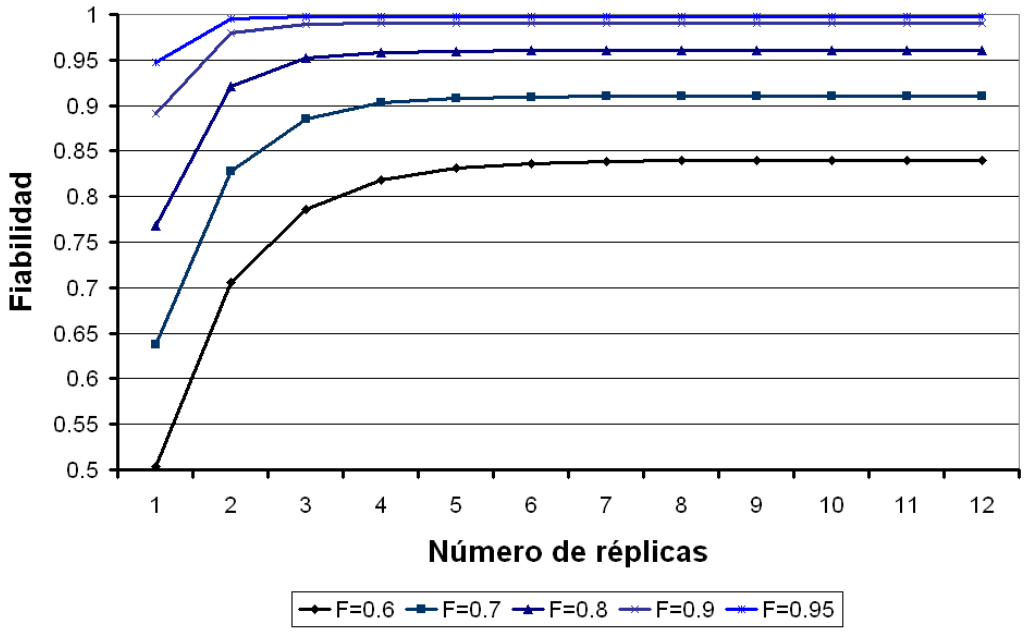

Figura 5.15: Fiabilidad de un *cluster* Web con *switch* distribuido de dos *switches* según el número de réplicas.

Como puede observarse, se puede conseguir una alta fiabilidad con un número relativamente bajo de réplicas. De hecho, un número muy alto de réplicas por elemento no mejora la fiabilidad del sistema.

Un aspecto a tener en cuenta en este tipo de sistemas es el incremento de fiabilidad que supone la adición de nuevos *switches*.

La fiabilidad del sistema (véase ecuación 5.16) se puede expresar siempre mediante la ecuación

$$F_{servidor} = F_{switches} F_{nodos}$$

En dicha ecuación F_{nodos} se mantiene constante para cada estrategia de replicación, por lo que mejora de fiabilidad δF se puede expresar como:

$$\delta F = \frac{F'_{switches} - F_{switches}}{F_{switches}} = \frac{\left(1 - (1 - F_{switch})^{P+\Delta P}\right) - \left(1 - (1 - F_{switch})^{P}\right)}{\left(1 - (1 - F_{switch})^{P}\right)} =$$

$$\delta F = \frac{-\left(1-F_{switch}\right)^{P+\Delta P} + \left(1-F_{switch}\right)^{P}}{\left(1-\left(1-F_{switch}\right)^{P}\right)} = \frac{\left(1-F_{switch}\right)^{P}\left(1-\left(1-F_{switch}\right)^{\Delta P}\right)}{1-\left(1-F_{switch}\right)^{P}}$$

Si se desea considerar la mejora de la fiabilidad que se consigue sobre el caso de un único *switch*, P será igual a 1 y por tanto

$$\delta F = \frac{1-F_{switch}}{1-\left(1-F_{switch}\right)}\left(1-\left(1-F_{switch}\right)^{\Delta P}\right) = \frac{1-F_{switch}}{F_{switch}}\left(1-\left(1-F_{switch}\right)^{\Delta P}\right)$$

Si además, se tiene en cuenta que la fiabilidad es una probabilidad ($0 \leq F_{switch} \leq 1$), se puede observar que

$$1-\left(1-F_{switch}\right)^{\Delta P} \leq 1$$

Y por tanto, cuando ΔP crece, δF se aproxima a su valor máximo

$$\delta F_{max} = \frac{1-F_{switch}}{F_{switch}}$$

La Figura 5.16 muestra la tasa de incremento de de fiabilidad con respecto al caso de un único *switch* Web cuando se incrementa el número de *switches*.

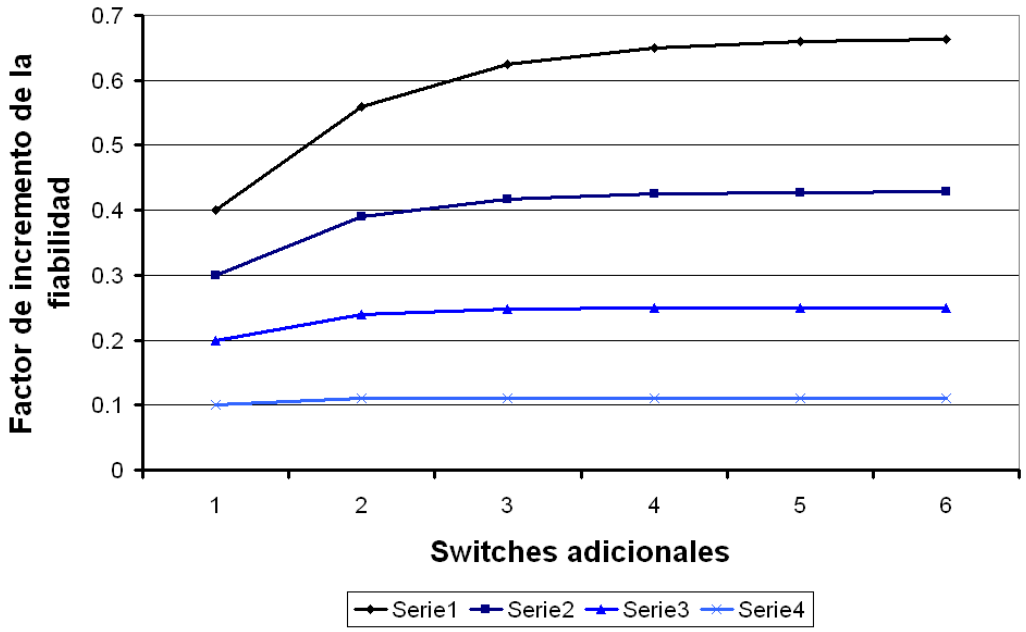

Figura 5.16: Factor de incremento de la fiabilidad con respecto al caso de un único *switch*.

Dos conclusiones interesantes pueden extraerse de esta última evaluación.

Por una parte, *la adición de los primeros switches tiene un gran impacto en la fiabilidad*. Posteriormente, cada incorporación de otro *switch* tiene impacto cada vez menor. Después del tercer o cuarto *switch* el efecto sobre la fiabilidad es prácticamente nulo.

Por otra parte, *el factor de incremento de la fiabilidad se ve muy influenciado por la fiabilidad individual de cada switch.* Así, con una fiabilidad individual de 0,6 se puede alcanzar un factor de incremento superior al 66 %. Con una fiabilidad individual de 0,8 se puede alcanzar un factor de incremento próximo al 25 %. Por último, con una fiabilidad individual de 0,9 apenas se alcanza un factor de incremento del 11 %.

5.3.3 Comparación de la fiabilidad

Para comparar la fiabilidad de las dos soluciones estudiadas, se ha evaluado la fiabilidad de un sistema Web basado en *cluster* y de un *cluster* Web con *switch* distribuido de 12 nodos servidores. En todos los casos se ha considerado que la fiabilidad de todos los nodos servidores y de todos los *switches* es 0,9. Para cada solución se ha evaluado la fiabilidad global para distintos números de réplicas por elementos. La Figura 5.17 muestra los resultados de la evaluación. En dicha gráfica, la curva de *1 switch* se corresponde con un sistema Web basado en *cluster* y el resto se corresponden con *clusters* Web con *switch* distribuido con distinto número de *switches*.

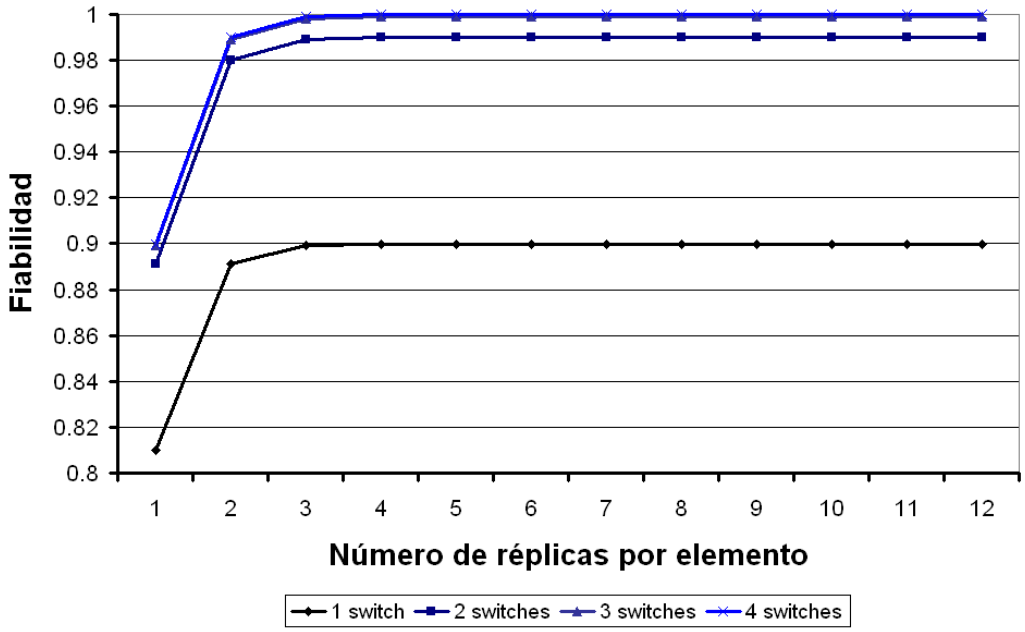

Figura 5.17: Comparación de la fiabilidad de un sistema Web basado en *cluster* y un *cluster* Web con *switch* distribuido.

Las evaluaciones muestran, una vez más, que *un factor determinante en el incremento de la fiabilidad de un sistema, es el número de réplicas por elemento.* De hecho, un número relativamente bajo de réplicas (3 o 4 en el ejemplo) hace que la fiabilidad global sea la misma que si el sistema estuviese totalmente replicado.

Además, de la evaluación, también se puede concluir que una solución de cluster Web con switch distribuido es siempre superior en fiabilidad a la solución que usa un único switch.

5.4 Relación entre capacidad de almacenamiento y fiabilidad

Las evaluaciones realizadas en las secciones anteriores sobre capacidad de almacenamiento y fiabilidad muestran que la capacidad de almacenamiento de un sistema parcialmente replicado es mayor cuanto menor es el número de réplicas por elemento. Por otra parte, cuando se reduce el número de réplicas también se reduce la fiabilidad del sistema. Debido a estas dos fuerzas contradictorias, el número de réplicas debe ser suficientemente alto como para ofrecer una alta fiabilidad y suficientemente bajo como para ofrecer una alta capacidad de almacenamiento.

Para tener en cuenta el impacto de ambas fuerzas, se ha evaluado la fiabilidad y la capacidad de almacenamiento en un *sistema Web basado en cluster* y en un *cluster Web con switch distribuido* (para 2 y 3 switches). Tanto la fiabilidad como la capacidad de almacenamiento se han evaluado para distintos números de réplicas por elemento. La figura 5.18 muestra los resultados obtenidos.

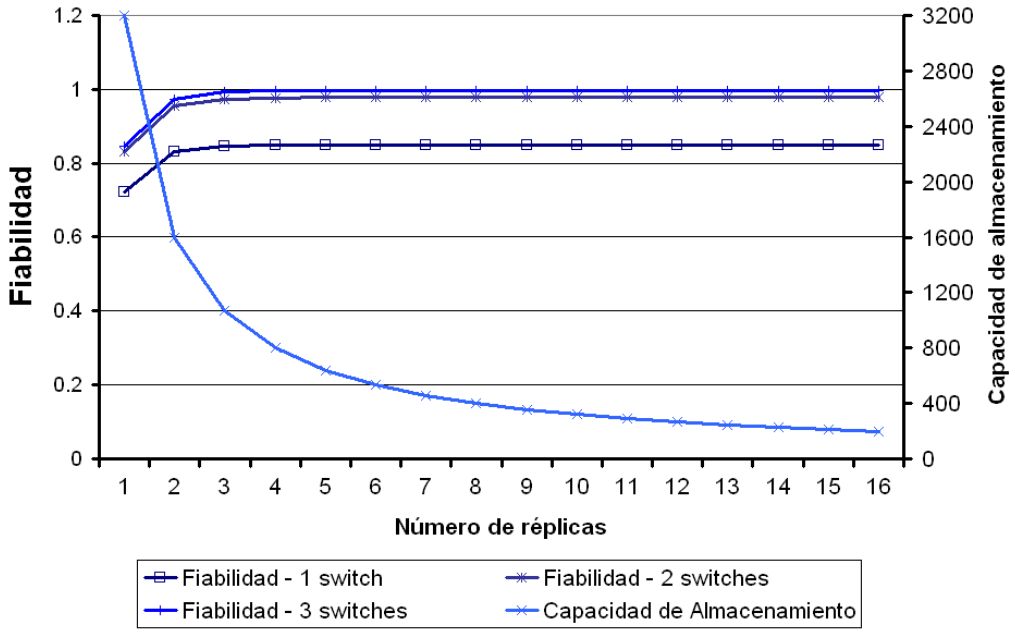

Figura 5.18: Fiabilidad y capacidad de almacenamiento obtenidas con distintos niveles de replicación parcial.

En todos los casos se ha considerado que el sistema está compuesto por 16 nodos servidores y que cada nodo servidor tiene una capacidad individual de almacenamiento de 200 GB. Para evaluar el efecto de la replicación parcial se han evaluado los casos de replicación total (16 réplicas), distribución total (1 réplica) y replicación parcial (entre 2 y 15 réplicas). Se ha considerado que tanto los nodos servidores como los *switches* tienen una fiabilidad del 85 %.

Los casos de replicación total o con un número muy alto de réplicas se corresponden con la parte derecha de la gráfica donde la fiabilidad es máxima y la capacidad de almacenamiento es mínima. En la parte izquierda de la gráfica se puede observar como la distribución total (número de réplicas igual a 1), ofrece capacidad de almacenamiento máxima y fiabilidad mínima. Cuando se aumenta el número de réplicas se incrementa la fiabilidad y se reduce la

capacidad de almacenamiento.

De forma general, la gráfica se puede utilizar como un instrumento de diseño de la arquitectura. Dependiendo de la fiabilidad y la capacidad de almacenamiento deseada se puede determinar el número de réplicas y el número de *switches* aconsejables.

Como ejemplo, considérese que los criterios para la selección de la arquitectura fuesen de al menos el 85 % y una capacidad de almacenamiento máxima. En el caso de utilizar un único *switch* sería necesario utilizar 10 réplicas por elemento, lo que daría lugar a un espacio disponible de 320 GB. Si se utilizan 2 o más *switches* se supera la fiabilidad objetivo utilizando 2 réplicas por elemento, lo que da lugar a un espacio disponible de 1600 GB. De esta manera se puede multiplicar por cinco el espacio disponible utilizando solamente dos *switches*.

Considérese ahora que el criterio de selección sea alcanzar una fiabilidad superior al 97 %. Si se utiliza un único *switch* es imposible alcanzar dicha fiabilidad (ni siquiera con replicación total). Si se utilizan 2 *switches* dicha fiabilidad se alcanzaría con 3 réplicas por elemento, dando lugar a un espacio disponible de 1066 GB. Si se utilizan 3 o más *switches*, se alcanzará la fiabilidad objetivo con 2 réplicas por elemento, obteniéndose un espacio disponible de 1600 GB.

Por último, considérese que el criterio de selección es la obtención de un espacio de almacenamiento superior a 1000 GB. Para ello sería necesario utilizar como máximo 3 réplicas por elemento. Una solución con un único *switch* ofrece una fiabilidad del 84,7 %. Con dos *switches* la fiabilidad pasa al 97,4 % y con 3 o más *switches* se supera el 99,3 % de fiabilidad. De esta manera, el número de *switches* se selecciona en función de la fiabilidad deseada.

Puede verse, por tanto, que existen dos maneras de utilizar la relación entre capacidad de almacenamiento y fiabilidad para guiar el diseño de la arquitectura:

- Fijar la fiabilidad mínima requerida y determinar el número de réplicas necesarias en función del número de *switches*.
- Fijar la capacidad de almacenamiento requerida (que a su vez fija el número de réplicas) y determinar la fiabilidad en función del número de *switches*.

5.5 Evaluación del rendimiento

En esta sección se presentan la evaluaciones del rendimiento realizadas para las distintas políticas de asignación de réplicas y las distintas políticas de asignación de peticiones. Para cada política de asignación de peticiones (asignación de peticiones cíclica, asignación al nodo menos cargado y asignación LARD) se ha realizado la evaluación del tiempo de respuesta de peticiones Web para distintos tipos de asignación de réplicas a nodos servidores.

Para realizar la evaluación del rendimiento se ha utilizado el modelo de simulación que se describe a continuación y que está basado en los datos recogidos en diversos estudios experimentales.

5.5.1 Modelo de simulación

Para la evaluación del rendimiento de las ideas propuestas en esta tesis se ha optado por el uso de un modelo de simulación que permita comparar distintas soluciones. El modelo de simulación utilizado se basa en modelos existentes para servidores Web. Desgraciadamente,

los modelos existentes no tienen en cuenta las implicaciones de la utilización de estrategias de replicación parcial de contenidos, por lo que ha sido necesaria su adaptación.

Además de presentar el modelo de simulación utilizado para las evaluaciones del rendimiento, en este capítulo se presenta un estudio de los valores apropiados para los parámetros de dicho modelo.

La sección 5.5.1.1 presenta los modelos de carga previamente existentes. La sección 5.5.1.2 presenta un modelo de simulación para servidores Web sin réplica completa. Por último, en la sección 5.5.1.3 se presenta la selección de valores realizada para los distintos parámetros del modelo.

5.5.1.1 Modelo de carga

El modelo de carga describe todos los parámetros que afectan a las peticiones que llegan a un servidor Web. Es conveniente en este punto definir algunos términos importantes para el modelo de carga:

Sesión Está formada por todas las interacciones que realiza un usuario con un sitio Web sin que medie entre ellas un tiempo de inactividad superior a un cierto umbral (definido por el propio sitio Web). Una sesión comienza cuando el usuario accede a una página Web del sitio y continúa con todas las peticiones que se realizan (por ejemplo pulsando sobre hiperenlaces) hasta que se abandona el sitio o se deja de enviar peticiones durante un cierto tiempo.

Petición Web Está formada por la petición de un elemento primario (normalmente una página HTML) y la petición de todos los elementos asociados a dicho elemento. Una petición Web puede comenzar con la solicitud de una página HTML y cuando el cliente reciba dicha página y determine sus elementos incluidos (una foto, una hoja de estilos, un applet en *java*, etc.) continuará con la solicitud de dichos elementos.

Petición HTTP Es cada petición individual que un cliente envía a un servidor Web para solicitar el envío de un elemento. De este modo, un programa cliente envía inicialmente una petición HTTP para el elemento primario y, posteriormente, una petición para cada elemento incluido en él.

Existen diversos modelos de carga para la generación de peticiones Web. En las siguientes secciones se presentan algunos de estos modelos de forma general así como los parámetros de los que dependen.

5.5.1.1.1 Modelo de Mah En 1997 Mah [79] propuso un modelo derivado de un estudio empírico que se puede utilizar para la generación de cargas para simulación. Dicho modelo presupone la presencia de varios servidores Web a los que accede un conjunto de clientes. La generación de la carga sigue los pasos siguientes:

1. Selección de un servidor de acuerdo con la distribución de popularidad de los servidores.
2. Determinación del número de peticiones de páginas Web que se deben pedir al servidor.
3. Para cada petición de página Web:

(a) Espera de un cierto tiempo (que simula el tiempo que el usuario tarda en pensar su siguiente acción).
(b) Selección de una página Web sobre la que se realizará la siguiente petición.
(c) Determinación del tamaño de la petición Web.
(d) Envío de la petición Web al servidor.
(e) Determinación del tamaño del tamaño del documento a ser servido.
(f) Envío del documento desde el servidor al cliente.
(g) Determinación del número de elementos adicionales contenidos en el documento.
(h) Determinación del tamaño de cada una de las peticiones para cada uno de dichos elementos.
(i) Envío de las peticiones al servidor.
(j) Determinación del tamaño de cada uno de los elementos adicionales.
(k) Envío de los elementos adicionales desde el servidor al cliente.

Por lo tanto, el modelo viene caracterizado por los parámetros que se describen en la tabla 5.3.

Parámetro	**Descripción**
Popularidad de Servidor	Sitio Web al que un usuario se conecta durante una sesión
Peticiones Web por sesión	Número de peticiones Web que un usuario solicita durante una sesión.
Tiempo de pensamiento	Tiempo que transcurre entre dos peticiones Web de un usuario. Modela el tiempo que el usuario tarda en *pensar* antes de seleccionar un hiperenlace.
Tamaño de la petición primaria	Tamaño de la petición HTTP que solicita un elemento primario (normalmente una página HTML).
Tamaño del elemento primario	Tamaño del elemento primario devuelto por el servidor en respuesta a la petición primaria (normalmente una página HTML).
Número de elementos secundarios	Número de elementos incluidos en el elemento primario (fotos, hojas de estilos, applets, etc.).
Tamaño de petición secundaria	Tamaño de la petición HTTP que solicita un elemento secundario.
Tamaño de elemento secundario	Tamaño del elemento secundario devuelto por el servidor en respuesta a una petición secundaria.

Tabla 5.3: Parámetros del modelo de carga de Mah.

Este modelo no caracteriza los tiempos entre llegadas de peticiones. El único tiempo caracterizado es el tiempo entre peticiones realizadas por los usuarios. Esto se debe a que los tiempos entre llegadas están influenciados por factores específicos a la infraestructura de red. El modelo de *Mah* pretende ser independiente de dicha infraestructura. Esta generalidad, tiene como desventaja que la aplicación del mencionado modelo a la simulación implica el uso de un modelo detallado que simule una infraestructura de red específica.

5.5.1.1.2 Modelo de Choi En 1999 Choi et al. [80] propusieron otro modelo basado en datos experimentales con el objetivo de generar cargas para simiulación. La generación de la carga sigue los pasos siguientes:

1. Determinación del tamaño de la petición HTTP.
2. Petición de un elemento primario.
3. Determinación del tamaño del elemento primario.
4. Envío del elemento primario al cliente.
5. Procesamiento del elemento principal durante un determinado tiempo.
6. Determinación del número de elementos incluidos en el elemento primario.
7. Determinación del tamaño de las peticiones HTTP.
8. Envío de las peticiones HTTP.
9. Determinación de los tiempos entre llegadas de objetos secundarios.
10. Determinación de los tamaños de los objetos secundarios.
11. Envío de los objetos secundarios.
12. Determinación del tiempo que el usuario dedica a ver el documento.

Por lo tanto, el modelo viene caracterizado por los parámetros que se describen en la tabla 5.4.

Este modelo no distingue entre peticiones de elementos primarios y de elementos secundarios. Los tamaños de todas las peticiones se modelan mediante un único parámetro. De hecho, aunque el tamaño de las peticiones HTTP fue estudiado por Choi et al., no fue utilizado en su generación de tráfico.

Al contrario que el modelo de Mah, este modelo si caracteriza los tiempos entre llegadas. Esto hace que no sea necesario apoyarse en un modelo detallado de la infraestructura de red.

5.5.1.2 Modelo para servidores sin réplica completa

Los modelos presentados en este capítulo [79, 80] están especialmente pensados para caracterizar el tráfico de un servidor Web que se aloja en una única máquina. Dichos modelos se pueden extender sin problemas a servidores Web distribuidos con réplica completa. Esto es así porque la selección del nodo que debe servir la petición no depende del elemento concreto que deba servirse.

Cuando se construye un modelo de un servidor Web distribuido en el que no hay replicación o la replicación es parcial (no todos los elementos se encuentran en todos los nodos) es necesario tener en cuenta algunos detalles adicionales.

En el capítulo 4 se presentó un modelo de sitio Web (véase 4.1.1). De acuerdo con dicho modelo, los parámetros que se proponen en esta tesis para la definición de un sitio Web son los presentados en la tabla 5.5.

Parámetro	Descripción
Tamaño de petición	Tamaño de una petición HTTP enviada al servidor. Este modelo utiliza el mismo parámetro tanto para las peticiones de objetos primarios como secundarios.
Tamaño de elemento primario	Tamaño del elemento primario devuelto por el servidor Web cuando un cliente realiza una petición primaria (normalmente una página HTML).
Tiempo de procesamiento primario	Tiempo que un cliente tarda en procesar un objeto primario y extraer las referencias a los objetos secundarios para generar las correspondientes peticiones.
Número de elementos secundarios	Número de elementos incluidos en el elemento primario (fotos, hojas de estilo, applets, etc.)
Tiempo entre llegadas de objetos secundarios	Tiempo entre llegadas al cliente de dos elementos secundarios.
Tamaño de elemento secundario	Tamaño del elemento secundario devuelto por el servidor en respuesta a una petición secundaria.
Tiempo de visión	Tiempo que el usuario dedica a ver el contenido antes de realizar una petición de un nuevo elemento primario.

Tabla 5.4: Parámetros del modelo de carga de Choi.

Parámetro	Descripción
Número de elementos primarios	Número de elementos que pueden ser solicitados por un usuario (normalmente páginas HTML).
Número de elementos incluidos	Número de elementos secundarios incluidos en un elemento primario.
Tamaño de elemento primario	Tamaño de un elemento primario que puede ser solicitado por un usuario.
Tamaño de elemento secundario	Tamaño de un elemento secundario.

Tabla 5.5: Parámetros del modelo de sitio Web.

En el mismo capítulo, también se presentó un modelo de servidor (véase 4.1.2). De acuerdo con dicho modelo, los parámetros que se proponen en esta tesis para la definición del modelo de servidor son los presentados en la tabla 5.6.

Parámetro	Descripción
Número de servidores	Número de nodos servidores en los que se pueden alojar contenidos.
Capacidad de servidor	Capacidad de almacenamiento de un nodo servidor.
Número de réplicas	Número mínimo de réplicas que deben mantenerse de un elemento.

Tabla 5.6: Parámetros del modelo de almacenamiento.

5.5.1.2.1 Modelo de peticiones Sea C el conjunto de T clientes que acceden a un sitio Web

$$C = \{c_1, c_2, \ldots, c_T\} \tag{5.24}$$

La actividad de un cliente viene determinada por una secuencia de sesiones y período de inactividad entre sesiones (véase Figura 5.19).

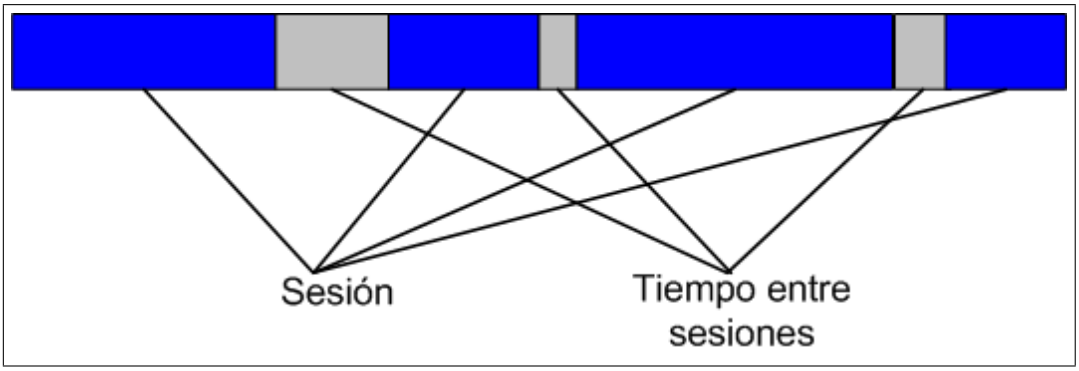

Figura 5.19: Actividad de un cliente a lo largo del tiempo.

De esta forma, la actividad Λ_i del cliente c_i viene definida por la secuencia de pares $\langle \Phi_j, \tau_j \rangle$

$$\Lambda_i = \langle \Phi_1, \tau_1 \rangle, \langle \Phi_2, \tau_2 \rangle, \ldots, \langle \Phi_j, \tau_j \rangle, \ldots$$

donde Φ_i es una sesión y τ_i es el tiempo que transcurre desde que finaliza la sesión Φ_i hasta que comienza la sesión Φ_{i+1}.

Durante una sesión un cliente solicita un número Q de elementos completos cada uno de los cuales está compuesto por un elemento primario y un conjunto de elementos secundarios. Cada una de la Q peticiones Web (véase Figura 5.20) se corresponde con una petición de un elemento primario ω_j, el análisis del objeto primario recibido durante un cierto periodo de tiempo ν_j, el envío y la recepción de las peticiones secundarias derivadas del objeto primario recibido y un tiempo de inactividad θ_j antes del envío de la siguiente petición.

$$\Phi_i = \langle \omega_1, \nu_1, \theta_1 \rangle, \langle \omega_2, \nu_2, \theta_2 \rangle, \ldots \langle \omega_Q, \nu_Q, \theta_Q \rangle \tag{5.25}$$

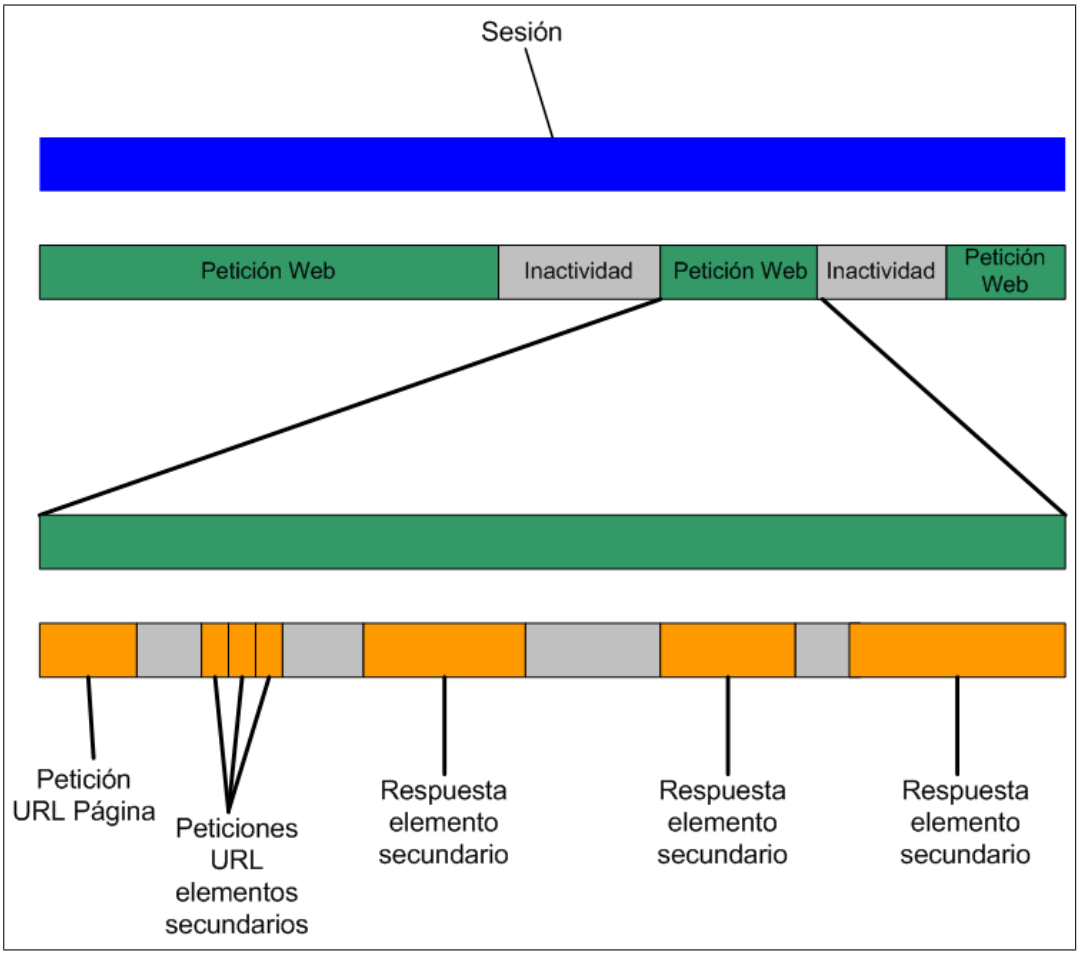

Figura 5.20: Modelo de peticiones de una sesión Web.

De acuerdo con todo lo expuesto, los parámetros que se proponen en esta tesis para la definición del modelo de peticiones son los que se presentan en la tabla 5.7.

5.5.1.3 Caracterización los parámetros del modelo

En esta sección se presenta la selección de valores realizada para cada uno de los parámetros del modelo, de modo que sea posible la simulación de servidores Web en los que los contenidos se encuentran distribuidos entre varios nodos servidores.

5.5.1.3.1 Número de elementos incluidos Cada elemento primario contiene un conjunto de elementos secundarios que es variable para cada elemento primario. Por tanto, el número de elementos incluidos se puede modelar como una variable aleatoria.

El número de elementos incluidos por un elemento primario se puede modelar bastante

Parámetro	Descripción
Tiempo entre sesiones	Tiempo que un cliente Web permanece inactivo entre la finalización de una sesión y el comienzo de la siguiente sesión.
Número de peticiones por sesión	Número de peticiones Web que un cliente realiza durante una sesión
Tiempo de inactividad	Tiempo que un cliente Web permanece inactivo entre al recepción de la respuesta a una petición y la solicitud de el envío de la siguiente petición
Tiempo de análisis	Tiempo que el cliente tarda en analizar el contenido de un elemento primario para determinar qué elementos secundarios debe pedir.

Tabla 5.7: Parámetros del modelo de peticiones.

bien mediante una distribución de Pareto [73] (véase ecuación 5.26).

$$f(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}} \qquad x \geq k \tag{5.26}$$

En [73] los estudios experimentales derivaron valores de $\alpha = 2{,}43$ y $k = 1$.

Diversos trabajos han utilizado valores similares para realizar simulaciones. En [7] se utilizan valores de $\alpha = 1{,}245$ y $k = 2$. En [50] se utilizan valores de $\alpha = 1{,}245$ y $k = 3$.En [66, 27] se utilizan valores para α que oscilan entre 1 y 1,5 con $k = 1$. En [65] se utilizan valores del $\alpha = 1{,}33$ y $k = 1$.

5.5.1.3.2 Tamaño de elementos primarios El tamaño de los elementos primarios se ha modelado [73] mediante una distribución lognormal para el cuerpo de la distribución (véase ecuación 5.27a) y una distribución de Pareto para la cola (véase ecuación 5.27b).

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{5.27a}$$

$$f(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}} \qquad x \geq k \tag{5.27b}$$

En [73] se obtuvieron valores experimentales de $\mu = 9{,}537$ y $\sigma = 1{,}318$ para el cuerpo de la distribución. En dicho experimento, se derivaron también valores de $\alpha = 1{,}1$ y $k = 133K$ con lo que el 93 % de los elementos se situan en el cuerpo de la distribución

En [77] se muestran diversos estudios experimentales en los que se estudian el modelo para la cola de la distribución. En todos ellos se determina que la cola de la distribución comienza a partir de los 10000 bytes con un valor aproximado para α de 1 (entre 0,93 y 1,33).

En [81] se realizaron experimentos de los que se derivaron valores de $\mu = 7{,}640$ con $\sigma = 1{,}705$ y $\mu = 7{,}881$ con $\sigma = 1{,}339$ para el cuerpo de la distribución. En los mismos

experimentos, se derivaron valores de $\alpha = 1{,}383$ con $k = 2{,}924$ y $\alpha = 1{,}177$ y $k = 3{,}558$ para la cola de la distribución.

En [66, 27, 65] se utilizan valores de $\mu = 7{,}64$ y $\sigma = 1{,}705$ para el cuerpo de la distribución y $\alpha = 1{,}383$ y $k = 2924$ para la cola de la distribución.

Todos estos trabajos no realizan una distinción entre el tamaño de los elementos primarios y los elementos secundarios. Sin embargo, en [7, 50] si se tiene en cuenta este hecho utilizándose los valores $\mu = 7{,}63$ y $\sigma = 1{,}001$ para el cuerpo de la distribución y $\alpha = 1$ y $k = 10240$ para la cola de la distribución.

5.5.1.3.3 Tamaño de los elementos secundarios Para modelar el tamaño de los elementos secundarios se ha utilizado una distribución lognormal (véase ecuación 5.28).

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{5.28}$$

En [7, 50] se utilizan los valores $\mu = 8{,}215$ y $\sigma = 1{,}46$.

5.5.1.3.4 Tiempo entre sesiones Las llegadas de sesiones se pueden describir bastante bien como procesos de Poisson [82]. Es importante tener en cuenta que el envío de paquetes de red no se puede describir bien como un proceso de Poisson [83]. Así mismo, ni las peticiones Web ni las peticiones HTTP individuales se pueden modelar tampoco como un proceso de Poisson. No obstante, el modelo utilizado en esta tesis no realiza una representación de las llegadas agregadas de peticiones al servidor (que si se podría representar como un proceso de Poisson). En vez de esto, se realiza una representación independiente del comportamiento de cada cliente para el que la hipótesis de proceso de Poisson no es válida.

El comportamiento de un usuario se ha modelado como la ejecución de sesiones separadas por periodos de inactividad o tiempo entre sesiones. El tiempo entre sesiones se puede modelar mediante una distribución de Pareto [73].

En [84] se utilizan valores de $\alpha = 1{,}4$ y $k = 20$.

5.5.1.3.5 Peticiones por sesión Las peticiones Web modelan el número de peticiones que un usuario realiza durante una sesión. La primera petición que un usuario realiza suele coincidir con la entrada en el sitio Web (ya sea introduciendo una dirección en el navegador o seleccionando un enlace en otro sitio Web). El resto de peticiones durante la sesión suelen coincidir con pulsaciones que el usuario realiza en enlaces internos del propio sitio.

El número de peticiones Web que un usuario realiza se puede modelar con bastante exactitud mediante una distribución gaussiana inversa [85] (véase ecuación 5.29).

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} e^{-\frac{\lambda(x-\mu)^2}{2\mu^2 x}} \tag{5.29}$$

En [85] el análisis experimental dio lugar a unos valores de $\widehat{\mu} = 3{,}86$ y $\widehat{var}(x) = 6{,}08$ (lo que equivale a $\widehat{\lambda} = 9{,}46$). Otro experimento en el mismo estudio dio lugar a unos valores de $\widehat{\mu} = 2{,}98$ y $\widehat{\lambda} = 6{,}24$.

En [7, 66, 27, 65, 50] se utilizan valores de $\mu = 3{,}86$ y $\lambda = 9{,}46$.

En definitiva, el número de peticiones Web por sesión suele tener una distribución sesgada hacia la izquierda. Esto se debe a que un número importante de visitantes suele solicitar una página inicial y abandonar el sitio o visitar unas pocas páginas. Las distribuciones que mejor se adaptan a los datos experimentales son la gausiana inversa y la lognormal.

5.5.1.3.6 Tiempo de inactividad Otro aspecto a considerar es la distribución del tiempo entre peticiones Web dentro de una misma sesión. Diversos estudios [86, 60] han modelado el tiempo entre peticiones Web (o tiempo que el usuario piensa antes de realizar la siguiente petición) mediante una distribución de Pareto (véase ecuación 5.30).

$$f(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}} \qquad x \geq k \tag{5.30}$$

En [60, 73] el análisis de datos experimentales deriva valores de $\widehat{k} = 1$ y $\widehat{\alpha} = 1{,}5$.

Diversos trabajos [7, 50] han utilizado los valores $k = 1$ y $\alpha = 1{,}4$. En otros trabajos [66, 27, 65] se ha utilizado los valores $k = 2$ y $\alpha = 1{,}4$.

En este trabajo se utilizará como modelo para el tiempo entre peticiones web la distribución de Pareto con $\alpha = 1{,}4$ y $k = 1$.

5.5.1.3.7 Tamaño de la petición HTTP Por cada petición Web se generan varias peticiones HTTP (una para el elemento primario y una por cada elemento secundario). El tamaño de las peticiones HTTP se ha modelado mediante una distribución lognormal (véase ecuación 5.31).

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{5.31}$$

En [80] los resultados experimentales derivaron valores de $\mu = 360{,}4$ y $\sigma = 106{,}5$.

En [87] los resultados experimentales derivaron valores de $\mu = 458{,}4$ y $\sigma = 199{,}79$.

En este trabajo se utilizará como modelo para el tamaño de las peticiones HTTP la distribución lognormal con $\mu = 360{,}4$ y $\sigma = 106{,}5$.

5.5.1.3.8 Parámetros a utilizar en la simulación Un modelo de sitio Web que distinga entre elementos primarios y secundarios permite realizar simulaciones para contenidos parcial o totalmente distribuidos. El tamaño del sitio Web viene determinado por el número de elementos primarios seleccionado.

La tabla 5.8 muestra un resumen de los parámetros que se utilizan para generar los contenidos del sitio Web a simular.

La generación del sitio Web viene determinada por el número de elementos primarios. Para cada elemento primario se genera dicho elemento y tantos elementos secundarios como se determine mediante la distribución del número de elementos secundarios. Este modelo permite generar sitios Web de diversos tamaños.

Muchos modelos de simulación utilizan un modelo de peticiones simplificado que no tiene en cuenta el comportamiento individual de cada cliente, utilizando un modelo del comportamiento agregado de todos los clientes. Sin embargo, si el modelo ha de incluir detalles sobre

Parámetro	**Distribución**
Número de elementos incluidos	Pareto ($\alpha = 2{,}43$, $k = 1$)
Tamaño de elementos primarios (cuerpo)	Lognormal ($\mu = 7{,}63$, $\sigma = 1{,}001$)
Tamaño de elementos primarios (cola)	Pareto ($\alpha = 1$, $k = 10240$)
Tamaño de elementos secundarios	Lognormal ($\mu = 8{,}215$, $\sigma = 1{,}46$)

Tabla 5.8: Parámetros de generación del sitio Web.

las relaciones entre los elementos primarios y secundarios, este tipo de modelos no contiene información suficiente para realizar simulaciones. Un modelo más detallado que tenga en cuenta el comportamiento de los clientes individuales si es compatible con la inclusión de detalles sobre las relaciones entre los elementos primarios y secundarios.

La tabla 5.9 muestra un resumen de los parámetros que se utilizan para generar las peticiones de los clientes.

Parámetro	**Distribución**
Tiempo entre sesiones	Pareto ($\alpha = 1{,}4$, $k = 20$)
Peticiones por sesión	Gaussiana inversa ($\mu = 2{,}86$, $\lambda = 9{,}46$)
Tiempo de inactividad	Pareto ($\alpha = 1{,}4$, $k = 1$)
Tamaño de petición	Lognormal ($\mu = 360{,}4$, $\sigma = 106{,}5$)
Tiempo de análisis	Weibull ($\alpha = 1{,}46$, $\beta = 0{,}382$)

Tabla 5.9: Parámetros de generación del sitio Web.

La generación de peticiones se basa en el análisis de tráfico de sitios Web con muy alta demanda de peticiones como el sitio Web de los campeonatos del mundo de fútbol de 1998 o el sitio Web de *America On Line* AOL.

5.5.2 Rendimiento de la arquitectura de *sistema Web basado en cluster*

En esta sección se presentan las evaluaciones del rendimiento realizadas para un arquitectura de *sistema Web basado en cluster*. Las evaluaciones se han realizado para las tres políticas de asignación de peticiones presentadas en la sección 4.4 (asignación estática circular, asignación al nodo menos cargado y asignación LARD).

En todos los casos se ha simulado el comportamiento de 800 clientes que realizan peticiones sobre un *sistema Web basado en cluster* compuesto por 16 nodos servidores. Cada nodo servidor tiene un espacio de almacenamiento de 200 GB.

En cada uno de los casos se ha comparado el uso de distintas alternativas de distribución de contenidos con el objeto de comprobar si la política de asignación de réplicas afecta al rendimiento de un *sistema Web basado en cluster*. Las alternativas evaluadas han sido:

RTOT Sistema totalmente replicado.

RPCI2 Replicación parcial cíclica inicial con dos réplicas por elemento.

RPCI4 Replicación parcial cíclica inicial con cuatro réplicas por elemento.

RPCF2 Replicación parcial cíclica final con dos réplicas por elemento.

RPCF4 Replicación parcial cíclica inicial con cuatro réplicas por elemento.

RNOEQ Replicación no equitativa basada en probabilidades de acceso.

5.5.2.1 Evaluación de la política de asignación de peticiones cíclica

Para evaluar la política de asignación de peticiones cíclica se han realizado las siguientes consideraciones:

- En el caso de un sistema totalmente replicado se utiliza un único índice global que itera de forma modular entre los 16 nodos servidores.
- En los casos de replicación cíclica se utiliza un índice para cada elemento alojado.
- En el caso de replicación no equitativa solamente se aplica la política a aquellos elementos que se alojan en más de un nodo servidor, ya que en otro caso solamente existe posibilidad de asignar la petición a un único nodo.

Como métrica de rendimiento se ha utilizado el tiempo medio de servicio de una petición Web. La Tabla 5.10 muestra los resultados obtenidos después de realizar 35 realizaciones de simulación para cada una de las alternativas de replicación de contenidos.

Replicación	**Media**	**Varianza**
RTOT	5,422234027	0,001745808
RPCI2	5,423221377	0,001727355
RPCF2	5,422842417	0,001739406
RPCI4	5,421925824	0,001722767
RPCF4	5,422525610	0,001755285
RNOEQ	5,422836699	0,001744325

Tabla 5.10: Resultados de simulación para el tiempo medio de servicio de petición Web de un *cluster Web* para la política de asignación de peticiones estática circular.

Para determinar si la diferencia en los tiempos medios de servicio de petición Web es significativa se ha realizado un test de análisis de la varianza obteniéndose los resultados que se muestran en la Tabla 5.11. El test se ha realizado para un valor de $\alpha = 0{,}05$.

F	0,004406713
Valor crítico de F	2,258342482
Probabilidad	0,999996127

Tabla 5.11: Resultado del test de análisis de la varianza para el tiempo medio de servicio de petición Web de un *cluster Web* con una política de asignación de peticiones estática circular con $\alpha = 0{,}05$.

Por tanto, no existe evidencia de que haya diferencia entre el tiempo medio de servicio de peticiones Web en el caso de asignación de peticiones estática circular cuando se modifica la política de asignación de contenidos.

5.5.2.2 Política de asignación al nodo menos cargado

Para evaluar la política de asignación de peticiones al nodo menos cargado se ha considerado que el *switch Web* utiliza como métrica de la carga de los nodos servidores el número de conexiones activas con cada uno de los nodos servidores. En el caso de replicación parcial, solamente se consideran los nodos servidores que efectivamente alojan el elemento solicitado. En el caso de replicación no equitativa, la política no se aplica a aquellos elementos que se alojan en un único nodo servidor, ya que en dicho caso no existe posibilidad de elegir el nodo a asignar a la petición.

Como métrica de rendimiento se ha utilizado el tiempo medio de servicio de una petición Web. La Tabla 5.12 muestra los resultados obtenidos después de realizar 35 realizaciones de simulación para cada una de las alternativas de replicación de contenidos.

Replicación	**Media**	**Varianza**
RTOT	5,422833918	0,001730142
RPCI2	5,422803377	0,001729813
RPCF2	5,422275413	0,001743461
RPCI4	5,422364230	0,001747777
RPCF4	5,423339211	0,001705529
RNOEQ	5,422798874	0,001732924

Tabla 5.12: Resultados de simulación para el tiempo medio de servicio de petición Web de un *cluster Web* para la política de asignación de peticiones al nodo menos cargado.

Para determinar si la diferencia en los tiempos medios de servicio de petición Web es significativa se ha realizado un test de análisis de la varianza obteniéndose los resultados que se muestran en la Tabla 5.13. El test se ha realizado para un valor de $\alpha = 0{,}05$.

F	0,002960304
Valor crítico de F	2,258342482
Probabilidad	0,999998564

Tabla 5.13: Resultado del test de análisis de la varianza para el tiempo medio de servicio de petición Web de un *cluster Web* con una política de asignación de peticiones al nodo menos cargado con $\alpha = 0{,}05$.

Por tanto, no existe evidencia de que haya diferencia entre el tiempo medio de servicio de peticiones Web en el caso de asignación de peticiones al nodo menos cargado cuando se modifica la política de asignación de contenidos.

5.5.2.3 Política de asignación LARD

Para evaluar la política de asignación de peticiones LARD se ha considerado que el *switch Web* utiliza como métrica de la carga de los nodos servidores el número de conexiones activas

con cada uno de dichos nodos. En el caso de replicación parcial, solamente se consideran los nodos servidores que efectivamente alojan el elemento solicitado. En el caso de replicación no equitativa, la política no se aplica a aquellos elementos que se alojan en un único nodo servidor, ya que en dicho caso no existe posibilidad de elegir el nodo a asignar a la petición.

Como métrica de rendimiento se ha utilizado el tiempo medio de servicio de una petición Web. La Tabla 5.14 muestra los resultados obtenidos después de realizar 35 realizaciones de simulación para cada una de las alternativas de replicación de contenidos.

Replicación	**Media**	**Varianza**
RTOT	5,423566866	0,001743385
RPCI2	5,423960405	0,001705924
RPCF2	5,423018666	0,001728655
RPCI4	5,423745110	0,001735874
RPCF4	5,423814965	0,001739495
RNOEQ	5,423986327	0,001698838

Tabla 5.14: Resultados de simulación para el tiempo medio de servicio de petición Web de un *cluster Web* para la política de asignación de peticiones LARD.

Para determinar si las diferencias son significativas se ha realizado un test de análisis de la varianza obteniéndose los resultados que se muestran en la Tabla 5.15. El test se ha realizado para un valor de $\alpha = 0{,}05$.

F	0,00261706
Valor crítico de F	2,258342482
Probabilidad	0,999998944

Tabla 5.15: Resultado del test de análisis de la varianza para el tiempo medio de servicio de petición Web de un *cluster Web* con una política de asignación de peticiones LARD con $\alpha = 0{,}05$.

Por tanto, no existe evidencia de que haya diferencia entre el tiempo medio de servicio de peticiones Web en el caso de asignación de peticiones LARD cuando se modifica la política de asignación de contenidos.

5.5.3 Rendimiento de la arquitectura de *cluster Web con switch distribuido*

En esta sección se presentan las evaluaciones del rendimiento realizadas para un arquitectura de *cluster Web con switch distribuido*. Las evaluaciones se han realizado para las tres políticas de asignación de peticiones presentadas en la sección 4.4 (asignación estática circular, asignación al nodo menos cargado y asignación LARD).

En todos los casos se ha simulado el comportamiento de 800 clientes que realizan peticiones sobre un *cluster Web con switch distribuido* compuesto por 16 nodos servidores y 3 *switches Web*. Cada nodo servidor tiene un espacio de almacenamiento de 200 GB.

En cada uno de los casos se ha comparado el uso de distintas alternativas de distribución de contenidos con el objeto de comprobar si la política de asignación de réplicas afecta al

rendimiento de un *sistema Web basado en cluster*. Las alternativas evaluadas han sido:

RTOT Sistema totalmente replicado.

RPCI2 Replicación parcial cíclica inicial con dos réplicas por elemento.

RPCI4 Replicación parcial cíclica inicial con cuatro réplicas por elemento.

RPCF2 Replicación parcial cíclica final con dos réplicas por elemento.

RPCF4 Replicación parcial cíclica inicial con cuatro réplicas por elemento.

RNOEQ Replicación no equitativa basada en probabilidades de acceso.

5.5.3.1 Evaluación de la política de asignación de peticiones cíclica

Para evaluar la política de asignación de peticiones cíclica se han realizado las siguientes consideraciones:

- En el caso de un sistema totalmente replicado se utiliza un único índice global que itera de forma modular entre los 16 nodos servidores. Los índices utilizados por cada uno de los *switches Web* son independientes.
- En los casos de replicación cíclica se utiliza un índice para cada elemento alojado. Los índices usados por cada uno de los *switches* son independientes.
- En el caso de replicación no equitativa solamente se aplica la política a aquellos elementos que se alojan en más de un nodo servidor, ya que en otro caso solamente existe posibilidad de asignar la petición a un único nodo.

Como métrica de rendimiento se ha utilizado el tiempo medio de servicio de una petición Web. La Tabla 5.16 muestra los resultados obtenidos después de realizar 35 realizaciones de simulación para cada una de las alternativas de replicación de contenidos.

Replicación	**Media**	**Varianza**
RTOT	5,436501603	0,003001719
RPCI2	5,436968535	0,003015402
RPCF2	5,436536060	0,002974351
RPCI4	5,436148605	0,003007950
RPCF4	5,436385993	0,003007949
RNOEQ	5,436293051	0,003026502

Tabla 5.16: Resultados de simulación para el tiempo medio de servicio de petición Web de un *cluster Web con switch distribuido* para la política de asignación de peticiones estática circular.

Para determinar si la diferencia en los tiempos medios de servicio de petición Web es significativa se ha realizado un test de análisis de la varianza obteniéndose los resultados que se muestran en la Tabla 5.17. El test se ha realizado para un valor de $\alpha = 0{,}05$.

Por tanto, no existe evidencia de que haya diferencia entre el tiempo medio de servicio de peticiones Web en el caso de asignación de peticiones estática circular cuando se modifica la política de asignación de contenidos.

F	0,000921172
Valor crítico de F	2,258342482
Probabilidad	0,999999922

Tabla 5.17: Resultado del test de análisis de la varianza para el tiempo medio de servicio de petición Web de un *cluster Web con switch distribuido* con una política de asignación de peticiones estática circular con $\alpha = 0{,}05$.

5.5.3.2 Política de asignación al nodo menos cargado

Para evaluar la política de asignación de peticiones al nodo menos cargado se ha considerado que cada *switch Web* utiliza como métrica de la carga de los nodos servidores el número de conexiones activas con cada uno de dichos nodos servidores. En el caso de replicación parcial, solamente se consideran los nodos servidores que efectivamente alojan el elemento solicitado. En el caso de replicación no equitativa, la política no se aplica a aquellos elementos que se alojan en un único nodo servidor, ya que en dicho caso no existe posibilidad de elegir el nodo a asignar a la petición.

Como métrica de rendimiento se ha utilizado el tiempo medio de servicio de una petición Web. La Tabla 5.18 muestra los resultados obtenidos después de realizar 35 realizaciones de simulación para cada una de las alternativas de replicación de contenidos.

Replicación	**Media**	**Varianza**
RTOT	5,436960812	0,002982179
RPCI2	5,436458507	0,002966236
RPCF2	5,436746731	0,002983863
RPCI4	5,436064798	0,003056556
RPCF4	5,436678541	0,002988681
RNOEQ	5,436390141	0,002995563

Tabla 5.18: Resultados de simulación para el tiempo medio de servicio de petición Web de un *cluster Web con switch distribuido* para la política de asignación de peticiones al nodo menos cargado.

Para determinar si la diferencia en los tiempos medios de servicio de petición Web es significativa se ha realizado un test de análisis de la varianza obteniéndose los resultados que se muestran en la Tabla 5.19. El test se ha realizado para un valor de $\alpha = 0{,}05$.

F	0,00115285
Valor crítico de F	2,258342482
Probabilidad	0,999999864

Tabla 5.19: Resultado del test de análisis de la varianza para el tiempo medio de servicio de petición Web de un *cluster Web con switch distribuido* con una política de asignación de peticiones al nodo menos cargado con $\alpha = 0{,}05$.

Por tanto, no existe evidencia de que haya diferencia entre el tiempo medio de servicio de peticiones Web en el caso de asignación de peticiones al nodo menos cargado cuando se

modifica la política de asignación de contenidos.

5.5.3.3 Política de asignación LARD

Para evaluar la política de asignación de peticiones LARD se ha considerado que cada *switch Web* utiliza como métrica de la carga de los nodos servidores el número de conexiones activas con cada uno de dichos nodos. En el caso de replicación parcial, solamente se consideran los nodos servidores que efectivamente alojan el elemento solicitado. En el caso de replicación no equitativa, la política no se aplica a aquellos elementos que se alojan en un único nodo servidor, ya que en dicho caso no existe posibilidad de elegir el nodo a asignar a la petición.

Como métrica de rendimiento se ha utilizado el tiempo medio de servicio de una petición Web. La Tabla 5.20 muestra los resultados obtenidos después de realizar 35 realizaciones de simulación para cada una de las alternativas de replicación de contenidos.

Replicación	**Media**	**Varianza**
RTOT	5,437371853	0,002958586
RPCI2	5,436919128	0,003030427
RPCF2	5,436589203	0,003005580
RPCI4	5,436711705	0,003072764
RPCF4	5,436961603	0,002942660
RNOEQ	5,437461929	0,003018680

Tabla 5.20: Resultados de simulación para el tiempo medio de servicio de petición Web de un *cluster Web con switch distribuido* para la política de asignación de peticiones LARD.

Para determinar si las diferencias son significativas se ha realizado un test de análisis de la varianza obteniéndose los resultados que se muestran en la Tabla 5.21. El test se ha realizado para un valor de $\alpha = 0{,}05$.

F	0,001424554
Valor crítico de F	2,258342482
Probabilidad	0,999999769

Tabla 5.21: Resultado del test de análisis de la varianza para el tiempo medio de servicio de petición Web de un *cluster Web con switch distribuido* con una política de asignación de peticiones LARD con $\alpha = 0{,}05$.

Por tanto, no existe evidencia de que haya diferencia entre el tiempo medio de servicio de peticiones Web en el caso de asignación de peticiones LARD cuando se modifica la política de asignación de contenidos.

5.5.4 Comparación del rendimiento

Las evaluaciones realizadas demuestran que, para los casos evaluados, no existe diferencia significativa en el tiempo de servicio de las peticiones Web al modificar la política de asignación de peticiones. En otras palabras, no existe ninguna evidencia que sugiera que el uso de una estrategia de replicación parcial pueda provocar una pérdida de rendimiento sobre el uso

de una estrategia basada en replicación total. Esto se ha demostrado tanto para el caso de un *sistema Web basado en cluster*, como para el caso de un *cluster Web con switch distribuido*.

No obstante, se podría argumentar que el test de análisis de la varianza es un test bastante conservador y que el hecho de no haber podido descartar la hipótesis de igualdad de medias, no implica necesariamente que dichas medias sean iguales.

Incluso si se admite que las medias no son iguales, las diferencias entre el mejor y el peor caso no supera el 0,03 % para ninguna de las situaciones evaluadas. Teniendo en cuenta las ventajas proporcionadas por los sistemas de replicación parcial (aumento del espacio de almacenamiento disponible y mejora de la fiabilidad), dicha pérdida se considera más que aceptable.

5.6 Conclusiones derivadas de la evaluación

En este capítulo se ha realizado una evaluación de las ideas propuestas en esta tesis con respecto a tres características de los servidores Web distribuidos: la capacidad de almacenamiento, la fiabilidad y el rendimiento.

La evaluación de la capacidad se ha realizado de forma analítica y se ha demostrado que el volumen de almacenamiento requerido es mucho mayor en el caso de la replicación total que en los otros casos. En el otro extremo se encuentra la distribución total donde el volumen de almacenamiento necesario es el menor de todas las estrategias estudiadas. Una desventaja adicional de la replicación total es que el tamaño del sitio Web más grande que se puede alojar viene limitado por el tamaño del dispositivo de almacenamiento de mayor capacidad que se puede adquirir. De aquí, se concluye que las estrategias de replicación total no permiten escalar en volumen de almacenamiento. La estrategia totalmente distribuida es la que mejor permite escalar el volumen de almacenamiento.

En este capítulo se ha definido el concepto de eficacia de incremento para medir la relación entre el incremento del tamaño máximo del sitio Web y el volumen de almacenamiento que es necesario adquirir. Se han distinguido por separado los casos de incorporación de nuevos dispositivos de almacenamiento a los nodos existente y de sustitución de dichos dispositivos de almacenamiento.

Utilizando esta métrica se ha demostrado que la eficacia de incorporación de nuevos dispositivos es siempre superior en el caso de distribución total que en el de replicación parcial. Así mismo, se ha demostrado que ante la opción de incorporar un nuevo dispositivo de almacenamiento o sustituir uno de los dispositivos por otro de mayor capacidad, desde el punto de vista de la capacidad de almacenamiento, es siempre más eficaz incorporar un dispositivo nuevo.

También se ha demostrado que la eficacia decrece cuando se incrementa el número de réplicas. Por tanto, desde el punto de vista del volumen de almacenamiento, la alternativa más interesante es la utilización de un sistema totalmente distribuido o parcialmente replicado con un número pequeño de réplicas por elemento.

La evaluación de la fiabilidad de los *sistemas Web basados en cluster* y de los *cluster Web con switch distribuido* se ha realizado teniendo en cuenta las distintas estrategias de replicación de contenidos. Esta evaluación se ha realizado de forma analítica y se ha derivado la expresión de la fiabilidad para cada caso.

Se ha demostrado que la fiabilidad de un *sistema Web basado en cluster* está limitada

por la fiabilidad del *switch* Web y no puede ser mayor que esta última. Así mismo, se ha demostrado que en un *sistema Web basado en cluster*, el número de réplicas tiene un gran impacto sobre la fiabilidad del sistema: con un número de réplicas relativamente bajo (4 o 5) se puede alcanzar una fiabilidad equivalente a la de un sistema totalmente replicado.

En el caso de un *cluster Web basado en switch distribuido*, el número de réplicas tambien ejerce un importante impacto sobre la fiabilidad del sistema. También en este caso se puede alcanzar una fiabilidad equivalente a la de un sistema totalmente replicado cuando se utilizan 4 o 5 réplicas por elemento. Además en este capítulo también se ha considerado el incremento de fiabilidad que se obtiene al incrementar el número de *switches* de un *cluster Web basado en switch distribuido*. Para ello se ha determinado el factor de incremento de la fiabilidad por cada *switch* adicional. Del estudio del mencionado factor, se ha concluido que la fiabilidad se incrementa notablemente cuando se añaden el primer y el segundo *switch*. Sin embargo a partir de este punto, la mejora de la fiabilidad que se obtiene es marginal.

Una vez estudiadas por separado la capacidad de almacenamiento y la fiabilidad se ha estudiado la relación existente entre ambas. La relación entre ambos parámetros puede utilizarse para guiar el diseño de la arquitectura de un servidor Web distribuido ya que una vez fijado uno de ellos (el mínimo nivel de fiabilidad requerido o la mínima capacidad de almacenamiento necesaria), se puede determinar las distintas combinaciones de número de réplicas y número de *switches* que satisfacen los requerimientos.

La evaluación del rendimiento se ha realizado a partir de un modelo de simulación basado en estudios experimentales. Dichas evaluaciones se han realizado tanto para los *sistemas Web basados en cluster* como para los *cluster Web con switch distribuido*. En ambos casos se ha visto que no existe evidencia para rechazar la hipótesis de que el tiempo de servicio de las peticiones Web no depende de la política de asignación de contenidos. Incluso se se admite que existen diferencias, éstas nunca superan el 0,03 %.

Todo lo expuesto se puede resumir en la siguiente afirmación: **un sistema parcialmente replicado ofrece mayor capacidad de almacenamiento manteniendo la fiabilidad y sin pérdida de rendimiento**.

Capítulo 6

Conclusiones

En este capítulo se presentan las conclusiones derivadas de los trabajos realizados en esta tesis. Así mismo, se definen las principales líneas de trabajo futuro.

La sección 6.1 resume la motivación de esta tesis así como los objetivos que se han cumplido con su realización. La sección 6.2 presenta un resumen de las principales aportaciones realizadas en cuanto a nuevas propuestas. La sección 6.3 discute los resultados obtenidos en la evaluación y las conclusiones que se derivan de dichos resultados. Finalmente, la sección 6.4 enumera las líneas de trabajo futuro.

6.1 Motivación y objetivos

El creciente desarrollo de Internet y de la distribución de contenidos a través de Web ha creado un enorme interés en el diseño y la construcción de software y hardware que permita satisfacer las necesidades de rendimiento a medida que se incrementa el número de usuarios de los sistemas. En otras palabras, se hace necesario el diseño de sistemas que permitan dar servicios Web y que sean escalables. Dicha escalabilidad es necesaria tanto desde el punto de vista del número de usuarios del servicio, como del volumen de contenidos que se hace disponible a dichos usuarios.

Ante la necesidad de escalabilidad, los servidores Web distribuidos aparecen como una solución efectiva. Existen diversas soluciones que son escalables en cuanto al número de usuarios del servicio.

Dichas soluciones suelen basarse en la replicación total de los contenidos. Es decir, cada archivo (página HTML, fotografía, vídeo, etc.) se almacena en todos los nodos servidores del sistema. Este tipo de soluciones presentan una alta fiabilidad, como demuestran las evaluaciones de fiabilidad realizadas en esta tesis. Sin embargo, el principal problema que se presenta en este caso es la falta de escalabilidad en cuanto al volumen de almacenamiento, al estar dicho volumen limitado por la capacidad de los discos que se pueden adquirir en cada momento.

Algunas soluciones se basan en la distribución total de contenidos. Es decir, cada archivo se almacena en un único nodo servidor. Este tipo de soluciones presenta una gran capacidad de almacenamiento. Además, el volumen de almacenamiento se puede escalar de una forma muy sencilla: basta con añadir un nuevo nodo servidor para incrementar la capacidad del

sistema. El principal punto débil de este tipo de sistemas es la fiabilidad, ya que ésta es mucho más baja que la de un sistema totalmente replicado.

En esta tesis se ha presentado una solución que:

- Se basa en la replicación parcial de contenidos.
- Permite que la asignación de contenidos se adapte de forma dinámica a las necesidades del servicio.

En la siguiente sección se presenta las propuestas realizadas para alcanzar dichos objetivos.

6.2 Propuestas realizadas

En esta sección se presentan las propuestas que se han realizado en esta tesis. Las propuestas que se presentan se han clasificado en tres bloques:

- Propuestas relativas a la arquitectura del sistema.
- Propuestas relativas a las políticas de asignación de réplicas a los nodos servidores.
- Propuestas relativas a las políticas de asignación de peticiones.

6.2.1 Propuestas relativas a la arquitectura de los sistemas

Considerando las arquitecturas Web distribuidas existentes en la actualidad, en esta tesis se ha realizado un estudio de idoneidad para el uso de réplicas parciales y asignación dinámica de contenidos. El estudio de las diversas arquitecturas estudiadas revela que:

- Los *sistemas Web basados en cluster* requieren adaptaciones, pero se pueden adecuar a los objetivos establecidos en esta tesis.
- Los *cluster Web virtuales* son incompatibles con la replicación parcial de contenidos.
- Los *sistemas Web distribuidos* se pueden adecuar a los objetivos establecidos con ciertas adaptaciones. No obstante, estos sistemas presentan algunos inconvenientes en cuanto a la seguridad de los sistemas y el número de direcciones públicas de red que se pueden necesitar cuando se escala el sistema.

De las arquitecturas estudiadas la que mejor se adapta los objetivos de la tesis es la de los *sistemas Web basados en cluster*. Aunque esta arquitectura se puede utilizar directamente, presenta una desventaja desde el punto de vista de la fiabilidad. Posee un componente (el *switch Web*) que supone un punto de fallo único. En otras palabras, si el *switch* Web falla, el sistema falla por completo.

Para mitigar este problema, en esta tesis se ha propuesto una arquitectura basada en los *sistemas Web basados en cluster* que toma algunas ideas de los *sistemas Web distribuidos*. Esta propuesta arquitectónica consiste en la incorporación de varios *switches Web* que realizan el reparto de las peticiones de forma coordinada.

En definitiva, las contribuciones que se han realizado en esta tesis relativas a la arquitectura de los sistemas son:

1. La propuesta de una familia de soluciones arquitectónicas de *sistemas Web basados en cluster* capaces de satisfacer los objetivos de replicación parcial y distribución dinámica de réplicas.

2. La propuesta de una familia de soluciones arquitectónicas novedosa de *cluster Web con switch distribuido* capaz de satisfacer los objetivos planteados.

6.2.2 Propuestas relativas a la asignación de réplicas

Un problema derivado de la replicación parcial de contenidos es la necesidad de decidir cómo se realiza la asignación de los contenidos a los nodos servidores.

La asignación de réplicas se puede realizar de forma equitativa o no equitativa. En la asignación de réplicas equitativa, se genera el mismo número de réplicas para todos los elementos del sitio Web. En la asignación no equitativa se generan más réplicas de los elementos que son accedidos con mayor frecuencia.

En esta tesis se han propuesto tres algoritmos de asignación equitativa de réplicas: *asignación equitativa cíclica inicial*, *asignación equitativa cíclica final* y *asignación equitativa voraz*. Así mismo, se ha propuesto un algoritmo de *asignación no equitativa de réplicas basado en las probabilidades de acceso a los elementos*.

La distribución dinámica de réplicas permite que cuando se modifique el patrón de acceso de los usuarios, sea posible realizar una redistribución de las réplicas. Esta redistribución se basará en las frecuencias de acceso a los elementos registradas durante un período de tiempo. Para determinar cuando es necesaria la redistribución, en esta tesis se ha propuesto un criterio basado en el número de réplicas que se deben añadir y/o eliminar.

En resumen, las contribuciones que se han realizado en esta tesis relativas a la asignación de réplicas son:

1. La propuesta de algoritmos que permiten realizar la asignación de réplicas de los elementos de un sitio Web a un conjunto de nodos servidores, cuando la capacidad de almacenamiento está limitada y se debe generar el mismo número de réplicas para cada elemento.

2. La propuesta de un algoritmo de asignación de réplicas que hace que los elementos con mayor frecuencia de acceso sean replicados en más nodos servidores que los elementos con baja frecuencia de acceso.

3. La propuesta de una estrategia para la replicación dinámica de contenidos que permite determinar cuando es necesario realizar la redistribución de contenidos y cómo se debe realizar dicha redistribución.

6.2.3 Propuestas relativas a la asignación de peticiones

La política de asignación de peticiones determina qué nodo servidor recibe la responsabilidad de dar servicio a cada petición. La replicación parcial de contenidos implica que la política de asignación de peticiones debe ser consciente de la asignación de réplicas a nodos. Esto se puede resolver mediante la utilización de un servicio de directorio que permita identificar el conjunto de nodos en los que se encuentra alojado un elemento.

En esta tesis se ha propuesto la adaptación de dos políticas clásicas de asignación de peticiones (asignación estática circular y asignación al nodo menos cargado). Además se ha propuesto la adaptación de una política que tiene en cuenta el comportamiento de las cachés de los servidores Web, dando lugar a la *asignación parcial LARD* (distribución de peticiones dependiente de la localidad).

La principal contribución realizada en esta tesis en cuanto a la asignación de peticiones ha sido precisamente la propuesta de la política de asignación parcial LARD que tiene en cuenta el efecto de la caché de cada servidor Web.

6.3 Resultados obtenidos de la evaluación

En esta sección se presentan las conclusiones derivadas de las evaluaciones realizadas. Por una parte se presentan las conclusiones derivadas de la evaluación de la capacidad de almacenamiento. Por otra parte se presentan las conclusiones derivadas de la evaluación de la fiabilidad. Finalmente se presentan las conclusiones derivadas de la evaluación del rendimiento de las distintas alternativas.

6.3.1 Capacidad de almacenamiento

La capacidad de almacenamiento ofrecida por un servidor Web distribuido depende de la estrategia de replicación seleccionada. En esta tesis se ha evaluado la capacidad de almacenamiento para las distintas estrategias de replicación (replicación total, distribución total y replicación parcial).

En la evaluación se ha determinado el volumen de almacenamiento para alojar un sitio Web para las distintas estrategias. Se ha demostrado que el volumen de almacenamiento es mucho mayor en el caso de la replicación total que en los otros casos. En el otro extremo se encuentra la distribución total donde el volumen de almacenamiento necesario es el menor de todas las estrategias estudiadas. Una desventaja adicional de la replicación total es que el tamaño del sitio Web más grande que se puede alojar viene limitado por el tamaño del dispositivo de almacenamiento de mayor capacidad que se puede adquirir. De aquí, se concluye que las estrategias de replicación total no permiten escalar en volumen de almacenamiento. La estrategia totalmente distribuida es la que mejor permite escalar el volumen de almacenamiento.

En esta tesis se ha definido el concepto de eficacia de incremento para medir la relación entre el incremento del tamaño máximo del sitio Web y el volumen de almacenamiento que es necesario adquirir. Se han distinguido por separado los casos de incorporación de nuevos dispositivos de almacenamiento a los nodos existente y de sustitución de dichos dispositivos de almacenamiento.

Utilizando esta métrica se ha demostrado que la eficacia de incorporación de nuevos dispositivos es siempre superior en el caso de distribución total que en el de replicación parcial. Así mismo, se ha demostrado que ante la opción de incorporar un nuevo dispositivo de almacenamiento o sustituir uno de los dispositivos por otro de mayor capacidad, desde el punto de vista de la capacidad de almacenamiento, es siempre más eficaz incorporar un dispositivo nuevo.

Por último, también se ha demostrado que la eficacia decrece cuando se incrementa el número de réplicas. De hecho, de forma general la eficacia de un sistema parcialmente replicado con r réplicas por elemento es siempre menor que $1/r$.

Por tanto, desde el punto de vista del volumen de almacenamiento, la alternativa más interesante es la utilización de un sistema totalmente distribuido o parcialmente replicado con un número pequeño de réplicas por elemento.

6.3.2 Fiabilidad

La fiabilidad de un servidor Web distribuido no depende exclusivamente de la estrategia de replicación. Además también depende de la topología definida por la arquitectura seleccionada.

En esta tesis se ha evaluado la fiabilidad de los *sistemas Web basados en cluster*, así como de la arquitectura que se propone en la misma: los *cluster Web con switch distribuido*. Para cada una de estas dos arquitecturas se ha considerado el impacto que tiene la estrategia de replicación de contenidos en la fiabilidad.

En la evaluación se ha derivado la expresión de la fiabilidad para un *sistema Web basado en cluster*. Del estudio de dicha expresión se ha derivado que la fiabilidad de un *sistema Web basado en cluster* está limitada por la fiabilidad del *switch* Web y no puede ser mayor que esta última.

Así mismo, se ha demostrado que en un *sistema Web basado en cluster*, el número de réplicas tiene un gran impacto sobre la fiabilidad del sistema: con un número de réplicas relativamente bajo (4 o 5) se puede alcanzar una fiabilidad equivalente a la de un sistema totalmente replicado.

En el caso de un *cluster Web basado en switch distribuido*, el número de réplicas tambien ejerce un importante impacto sobre la fiabilidad del sistema. También en este caso se puede alcanzar una fiabilidad equivalente a la de un sistema totalmente replicado cuando se utilizan 4 o 5 réplicas por elemento.

En esta tesis también se ha considerado el incremento de fiabilidad que se obtiene al incrementar el número de *switches* de un *cluster Web basado en switch distribuido*. Para ello se ha determinado el factor de incremento de la fiabilidad por cada *switch* adicional. Del estudio del mencionado factor, se ha concluido que la fiabilidad se incrementa notablemente cuando se añaden el primer y el segundo *switch*. Sin embargo a partir de este punto, la mejora de la fiabilidad que se obtiene es marginal.

6.3.3 Rendimiento

La evaluación del rendimiento se ha realizado a partir de un modelo de simulación basado en estudios experimentales. Dichas evaluaciones se han realizado tanto para los *sistemas Web basados en cluster* como para los *cluster Web con switch distribuido*. En ambos casos se ha visto que no existe evidencia para rechazar la hipótesis de que el tiempo de servicio de las peticiones Web no depende de la política de asignación de contenidos. Incluso si se admite que existen diferencias, éstas nunca superan el 0,03 %.

6.4 Trabajo futuro

En esta tesis se ha considerado exclusivamente el servicio de peticiones estáticas, no teniéndose en cuenta las peticiones dinámicas en las que el servidor Web ejecuta algún tipo de código para generar la respuesta. Se deja para un futuro trabajo la integración del servicio de peticiones dinámicas en una arquitectura de *cluster Web basado en switch distribuido*.

Por otra parte, se han realizado propuestas relativas a la asignación dinámica de réplicas a los nodos servidores, de forma que se pueda realizar una redistribución de los contenidos para permitir la adaptación a las necesidades de carga. Como trabajo futuro se plantea la realización de estudios que permitan determinar el efecto que la asignación dinámica tiene sobre el rendimiento del sistema. Este estudio tendrá como objetivo determinar la rentabilidad que puede ofrecer la inversión en una red de servicio adicional para la redistribución de contenidos.

Todas las arquitecturas presentadas y propuestas en esta tesis pertenecen a la categoría de arquitecturas localmente distribuidas en la que todos los elementos del sistema se encuentran en una única distribución geográfica. Otra alternativa, que impone un conjunto distinto de restricciones, es la de los sistemas geográficamente distribuidos en la que los elementos del sistema pueden encontrarse en varias localizaciones geográficas. En un trabajo futuro se pretende estudiar cómo se pueden adaptar las ideas propuestas en esta tesis a los sistemas geográficamente distribuidos.

Una alternativa que no se ha explorado en esta tesis es la utilización de técnicas no deterministas para la asignación de réplicas a los nodos servidores. El uso de algoritmos genéticos puede ser especialmente interesante en el caso de la asignación no equitativa de réplicas, ya que con esta técnica se podrían introducir en la evaluación de las soluciones criterios que permitan que los distintos elementos secundarios que corresponden a un mismo elemento primario se almacenen (si esto es posible) en conjuntos disjuntos de nodos servidores.

Bibliografía

[1] Huitema, Christian: *Network versus servers issues in end-to-end performance.* Keynote speech at the Performance and Architecture of Web Servers Workshop (Santa Clara, CA), Junio 2000. http://www.huitema.net/talks/server-and-networks.ppt.

[2] Gilder, George: *Fiber keeps its promise: Get ready, bandwidth will triple each year for the next 25.* Forbes, Abril 1997.

[3] Gray, Jim y Phrashant Shenoy: *Rules of thumb in data engineering.* En *Proceedings of the 16th IEEE International Conference on Data Engineering*, páginas 3–10. IEEE, Febrero 2000, ISBN 0-7695-0506-6. http://research.microsoft.com/ gray/papers/$\mathrm{MS}_T R_9 9_1 00_R rules_o f_T humb_i n_D ata_E ngineering.pdf$.

[4] Coffman, K. G. y Andrew M. Odlyzko: *Growth of internet.* En Kaminow, Ivan y Tingye Li (editores): *Optical Fiber Telecommunications IV B: Systems and Impairments*, páginas 17–56. Academic Press, Abril 2002, ISBN 0123951739. http://www.dtc.umn.edu/ odlyzko/doc/oft.internet.growth.pdf.

[5] Coffman, K. G. y Andrew M. Odlyzko: *Internet growth: Is there a "Moore's law" for data traffic?* En Abello, James, Panos M. Pardalos y Mauricio G. C. Resende (editores): *Handbook of Massive Data Sets*, páginas 47–93. Kluwer Academic Press, Abril 2002, ISBN 1402004893. http://www.dtc.umn.edu/ odlyzko/doc/internet.moore.pdf.

[6] Odlyzko, Andrew M.: *Internet traffic growth: Sources and implications.* En *Optical Transmission Systems and Equipement for WDM Networking II. Proceedings of the SPIE - The International Society for Optical Engineering*, volumen 5247, páginas 1–15, Orlando, FL, Septiembre 2003. SPIE. International Society for Optical Engineers. http://www.dtc.umn.edu/ odlyzko/doc/itcom.internet.growth.pdf.

[7] Cardellini, Valeria, Michele Colajanni y Philip S. Yu: *Geographic load balancing for scalable distributed Web systems.* En *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'00)*, páginas 20–27, San Francisco, CA, EEUU Agosto 2000. IEEE, ISBN 0-7695-0728-X.

[8] Cardellini, Valeria, Emiliano Casalicchio, Michele Colajanni y Philip S. Yu: *The state of the art in locally distributed Web-server systems.* ACM Computing Surveys, 34(2):263–311, Junio 2002, ISSN 0360-0300.

[9] Banga, Gaurav, Peter Druschel y Jeffrey C. Mogul: *Better operating system features for faster network servers*. Performance Evaluation Review, 26(3):23–30, Deciembre 1998, ISSN 0163-5999. http://www.cs.rice.edu/ gaurav/papers/wisp98.ps.

[10] Banga, Gaurav, Peter Druschel y Jeffrey C. Mogul: *Resource containers: A new facility for resource management in server systems*. Operating Systems Review, Special Issue, Winter 1998:45–58, 1999, ISSN 0163-5980. http://www.cs.rice.edu/ druschel/osdi99rc.ps.gz.

[11] Banga, Gaurav, Jeffrey C. Mogul y Peter Druschel: *A scalable and explicit event delivery mechanism for UNIX*. En *Proceedings of the USENIX 1999 Annual Technical Conference*, páginas 253–265, Monterrey, CA, Junio 1999. USENIX Association, ISBN 1-880446-33-2. http://www.cs.rice.edu/ gaurav/papers/usenix99.ps.

[12] Hu, Yiming, Ashwini Nanda y Qing Yang: *Measurement, analysis and performance improvement of the Apache Web server*. En *Proceedings of 18th IEEE International Performance, Computing and Communications Conference (IPCCC'99)*, páginas 261–267, Scottsdale, AZ, Febrero 1999. IEEE, ISBN 0-7803-5258-0. http://www.ececs.uc.edu/ oscar/papers/IPCCC99.pdf.

[13] Hu, Yiming, Ashwini Nanda y Qing Yang: *Measurement, analysis and performance improvement of the Apache Web Server*. International Journal of Computers and their Applications, 8(4):217–231, Deciembre 2001, ISSN 1076-5204.

[14] Hu, James C., Irfan Pyarali y Douglas C. Schmidt: *Measuring the impact of event dispatching and concurrency models on Web server performance over high-speed networks*. En *Conference Record of IEEE Global Telecomunications Conference (GLOBECOM 97)*, volumen 3, páginas 1924–1931, Phoenix, AZ, Noviembre 1997. IEEE, ISBN 0-7803-4198-8. http://www.cs.wustl.edu/ schmidt/PDF/globalinternet.pdf.

[15] Microsoft Developer Network: *TransmitFile*. http://msdn.microsoft.com/library/en-us/winsock/winsock/transmitfile$_2$.*asp*.

[16] Pai, Vivek S., Peter Druschel y Willy Zwaenepoel: *IO-Lite: A unified I/O buffering and caching system*. ACM Transactions on Computer Systems, 18(1):37–66, Febrero 2000, ISSN 0734-2071.

[17] Apache Software Foundation. http://www.apache.org.

[18] Pai, Vivek S., Peter Druschel y Willy Zwaenepoel: *Flash: An efficient and portable Web server*. En *Proceedings of the USENIX 1999 Annual Technical Conference*, páginas 199–212, Monterey, CA, Junio 1999. USENIX Association, ISBN 1-880446-33-2. http://www.usenix.org/events/usenix99/full$_p$*apers/pai/pai.pdf*.

[19] Berners-Lee, T., R. Fielding y H. Frystyk: *Hypertext Transfer Protocol - HTTP/1.0. RFC 1945*. Internet Engineering Task Force, Mayo 1996. http://www.w3.org/Protocols/rfc1945/rfc1945.

[20] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach y T. Berners-Lee: *Hypertext Transfer Protocol - HTTP/1.1. RFC 2616*. Internet Engineering Task Force, Junio 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[21] Schroeder, Trevor, Steve Goddard y Byrav Ramamurthy: *Scalable web server clustering technologies*. IEEE Network, 14(3):38–45, Mayo 2000, ISSN 0890-8044. http://www.comsoc.org/ni/public/2000/may/index.html.

[22] ISO: *Open Systems Interconnection - Basic Reference Model. ISO-IEC 7498-1*, spanish2 edición, 1994.

[23] Srisuresh, P. y K. Egevang: *Traditional IP Network Address Translator (Traditional NAT). RFC 3022*. Internet Engineering Task Force, Enero 2001. http://www.faqs.org/rfcs/rfc3022.html.

[24] Dias, Daniel M., William Kish, Rajat Mukherjee y Renu Tewari: *A scalable highly available Web server*. En *Proceedings of the IEEE International Computer Conference (COMPCON'96)*, páginas 85–92, Santa Clara, CA, Febrero 1996. IEEE, ISBN 0-8186-7414-8.

[25] Perkins, C.: *IP Encapsulation within IP. RFC 2003*. Internet Engineering Task Force, Octubre 1996. http://www.faqs.org/rfcs/rfc2003.html.

[26] Bourke, Tony: *Server Load Balancing*. O'Reilly, Agosto 2001, ISBN 0-596-00050-2.

[27] Casalicchio, Emiliano y Michele Colajanni: *A client-aware dispatching algorithm for web clusters providing multiple services*. En *Proceedings of the tenth international conference on World Wide Web*, páginas 535–544, Hong Kong, Mayo 2001. ACM Press, ISBN 1-58113-348-0. http://www10.org/cdrom/papers/pdf/p434.pdf.

[28] Andreolini, Mauro, Emiliano Casalicchio, Michele Colajanni y Marco Mambelli: *A cluster-based web system providing differentiated and guaranteed services*. Cluster Computing, 7(1):7–19, Enero 2004, ISSN 1386-7857.

[29] Apostolopoulos, G., D. Aubespin, V. Peris, P. Pradham y D. Saha: *Design, implementation and performance of a content-based switch*. En *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, volumen 3, páginas 1117–1126, Tel Aviv, Israel, Abril 2000. IEEE, ISBN 0-7803-5880-5. http://www.ecs.umass.edu/ece/wolf/courses/ECE697J/papers/web$_s$*witch.PDF*.

[30] Maltz, D. A. y P. Bhagwat: *TCP splice for application layer proxy performance*. Journal of High Speed Networks, 8(3):225–240, Junio 1999, ISSN 0926-6801.

[31] Cohen, A., S. Rangarajan y H. Slye: *On the performance of TCP splicing for URL-aware redirection*. En *Proceedings of USENIX'99: 2nd Symposium on Internet Technologies and Systems*, páginas 117–125, Boulder, CO, Octubre 1999. USENIX, ISBN 1-880446-33-2.

[32] Spatscheck, O., J. S. Hansen, J. H. Hartman y L. L. Peterson: *Optimizing TCP forwarder performance.* IEEE/ACM Transactions on Networking, 8(2):146–157, Abril 2000, ISSN 1063-6692.

[33] Adhya, Saibal K., Saugata Das-Purkayastha y Sumit Ganguly: *Asymmetric splice: optimizing TCP forwarder performance for the HTTP/1.1 protocol.* En *Proceedings of the 15th International Conference on Computer Communication*, volumen 1, páginas 239–251, Mumbai, India, Agosto 2002. International Council for Computer Communication, ISBN 1-891365-08-8.

[34] Kobayashi, Masayoshi y Tutomu Murase: *Asymetric TCP splicing for content-based switches.* En *Proceedings fo the IEEE International Conference on Commnications (ICC 2002)*, volumen 2, páginas 1321–1326, New York, NY, EEUU Abril 2002. IEEE, ISBN 0-7803-7400-2.

[35] Kobayashi, Masayoshi y Tutomu Murase: *Efficient support for pipelined requests in content-based switches using asymmetric TCP splicing.* IEICE Transactions on Communications, E86-B(6):1812–1820, Junio 2003, ISSN 0916-8516.

[36] Li, Chang, Gang Peng, Kartik Gopalan y Tzi cker Chiueh: *Performance guarantees for cluster-based internet services.* En Lee, S., S. Sekiguchi, S. Matsuoka y M. Sato (editores): *Proceedings of the Third IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2003)*, páginas 276–283, Tokyo, Japón Mayo 2003. IEEE, ISBN 0-7695-1919-9.

[37] Rosu, Marcel Catalin y Daniela Rosu: *An evaluation of TCP splice benefits in web proxy servers.* En *Proceedings of the eleventh international conference on World Wide Web*, páginas 13–24, Honolulu, Hawaii, Mayo 2002. ACM Press, ISBN 1-58113-449-5.

[38] Pai, Vivek S., Mohit Aron, Gaurov Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel y Erich Nahum: *Locality-aware request distribution in cluster-based network servers.* ACM SIGPLAN Notices, 33(11):205–216, Noviembre 1998, ISSN 0362-1340.

[39] Aron, M., D. Sanders, P. Druschel y W. Zwaenepoel: *Scalable content-aware request distribution in cluster-based network servers.* En *Proceedings of the 2000 USENIX Annual Technical Conference*, páginas 232–336, San Diego, CA, Junio 2000. USENIX, ISBN 1-880446-22-7.

[40] Tang, Wenting, Ludmila Cherkasova, Lance Russell y Matt W. Mutka: *Modular TCP handoff design in STREAMS-based TCP/IP implementation.* En Lorenz, P. (editor): *Proceedings of the First International Conference on Networking*, volumen 2094 de *Lecture Notes in Computer Science*, páginas 71–81, Colmar, Francia Julio 2001. Springer-Verlag, ISBN 3-540-42303-6.

[41] Kokku, R., R. Rajamony, L. Alvisi y H. Vin: *Half-pipe anchoring: an efficient technique for multiple connection handoff.* En *Proceedings 10th IEEE International Conference on Network Protocols*, páginas 12–21, Paris, Francia Noviembre 2002. IEEE, ISBN 0-7695-1856-7.

[42] Cecchet, E.: *Whoops!: a clustered web cache for DSM systems using memory mapped networks*. En Wagner, R. (editor): *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, páginas 806–811, Viena, Austria, Julio 2002. IEEE, ISBN 0-7695-1588-6.

[43] Kerdlapanan, D. y A. Khunkitti: *Content-based load balancing with multicast and TCP-handoff*. En *Proceedings of the 2003 International Symposium on Circuits and Systems (ISCAS '03)*, volumen 2, páginas 348–351, Bangkok, Tailandia Mayo 2003. IEEE, ISBN 0-7803-7761-3.

[44] Microsoft: *Network Load Balancing Technical Overview*, Enero 2000. http://www.microsoft.com/windows2000/techinfo/howitworks/cluster/nlb.asp.

[45] Vaidya, Sujit y Kenneth J. Christensen: *A single system image server cluster using duplicated MAC and IP addresses*. En *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN 2001)*, páginas 206–214, Tampa, FL, EEUU Noviembre 2001. IEEE, ISBN 0-7695-1321-2.

[46] Microsoft: *Technical Overview of Windows Server 2003 Clustering Services*, Deciembre 2003. http://www.microsoft.com/windowsserver2003/techinfo/overview/clustering.mspx.

[47] Brisco, T.: *DNS Support for Load Balancing. RFC 1794*. Internet Engineering Task Force, Abril 1995. ftp://ftp.is.co.za/rfc/rfc1794.txt.

[48] Aversa, Luis y Azer Bestavros: *Load balancing a cluster of web servers: using distributed packet rewriting*. En *Conference Proceedings of the 2000 IEEE International Performance, Computing, and Communications Conference (IPCCC 2000)*, páginas 24–29, Phoenix, AZ, EEUU Febrero 2000. IEEE, ISBN 0-7803-5979-8.

[49] Andresen, Daniel, Tao Yang y Oscar H. Ibarra: *Toward a scalable distributed www server on workstation clusters*. Journal on Parallel and Distributed Computing, 42(1):91–100, Abril 1997, ISSN 0743-7315.

[50] Cardellini, Valeria: *Request redirection algorithms for distributed web systems*. IEEE Transactions on Parallel and Distributed Systems, 14(4):355–368, Abril 2003, ISSN 1045-9219.

[51] Colajanni, Michele, Philip S. Yu y Daniel M. Dias: *Analysis of task assignment policies in scalable distributed web-server systems*. IEEE Transactions on Parallel and Distributed Systems, 9(6):585–600, Junio 1998, ISSN 1045-9219.

[52] Baker, Scott M. y Bongki Moon: *Distributed cooperative web servers*. Computer Networks, 31(11–16):1215–1229, Mayo 1999, ISSN 1389-1286.

[53] Li, Quanzhong y Bongki Moon: *Distributed cooperative Apache Web server*. En *Proceedings of the tenth international conference on World Wide Web*, páginas 555–564, Hong Kong, Mayo 2001. ACM Press, ISBN 1-58113-348-0.

[54] Carretero, Jesús, Javier Fernández y José M. Pérez: *Dynamic distribution and allocation of web objects*. En *Proceedings of the International Conference on Information Systems, Analysis and Synthesis (ISAS'2001)*, volumen XII, Orlando, FL, EEUU Julio 2001.

[55] Colajanni, Michele: *Emerging internet-based services: new frontiers for performance models and applications*. En *Proceedings of the First International Conference on the Quantitative Evaluation of Systems*, páginas 122–123. IEEE Computer Society, Septiembre 2004, ISBN 0-7695-2185-1.

[56] Linux Virtual Server: *Virtual server scheduling algorithms*, Mayo 2001. http://www.linuxvirtualserver.org/docs/scheduling.html.

[57] Krishnamurthy, Balachander y Jia Wang: *On network-aware clustering of web clients*. Computer Communication Review, 30(4):97–110, Octubre 2000, ISSN 0146-4833.

[58] Thaler, David G. y Chinya V. Ravishankar: *Using name-based mappings to increase hit rates*. IEEE/ACM Transactions on Networking, 6(1):1–14, Febrero 1998, ISSN 1063-6692.

[59] Luo, Mon Yen, Chun Wei Tseng y Chu Sing Yang: *URL formalization: An efficient technique to speedup content-aware switching*. IEEE Communication Letters, 6(12):553–555, Deciembre 2002, ISSN 1089-7798.

[60] Crovella, Mark E. y Azer Bestavros: *Self-similarity in world wide web traffic: evidence and possible causes*. IEEE/ACM Transactions on Networking, 5(6):835–846, Deciembre 1997, ISSN 1063-6692.

[61] Arlitt, Martin: *Characterizing web user sessions*. Performance Evaluation Review, 28(2):50–56, Septiembre 2000, ISSN 0163-5999.

[62] Arlitt, Martin y Tai Jin: *A workload characterization study of the 1998 world cup web site*. IEEE Network, 14(3):30–37, Mayo 2000, ISSN 0890-8044.

[63] Yang, Chu Sing y Mon Yen Luo: *Efficient content placement and management on cluster-based web servers*. En Hong, J. W. y R. Weihmayer (editores): *Proceedings of the Network Operations and management Symposium (NOMS 2000)*, páginas 463–476, Honolulu, HI, EEUU Abril 2000. IEEE/IFIP, ISBN 0-7803-5928-3.

[64] Yang, Chu Sing y Mon Yen Luo: *A content placement and management system for distributed web-server systems*. En *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, páginas 691–698, Taipei, Taiwan, Abril 2000. IEEE, ISBN 0-7965-0601-1.

[65] Casalicchio, Emiliano, Valeria Cardellini y Michele Colajanni: *Content-aware dispatching algorithms for cluster-based web servers*. Cluster Computing, 5(1):65–74, Enero 2002, ISSN 1386-7857.

[66] Casalicchio, Emiliano y Michele Colajanni: *Scalable web clusters with static and dynamic contents*. En *Proceedings IEEE International Conference on Cluster Computing*, páginas 170–177, Chemnitz, Alemania Noviembre 2000. IEEE, ISBN 0-7695-0896-0.

[67] Harchol-Balter, Mor, Mark E. Crovella y Cristina D. Martu: *On chosing a task assignment policy for a distributed server system.* Jorunal of parallel and Distributed Computing, 59(2):204–228, Noviembre 1999, ISSN 0743-7315.

[68] Teo, Yong Meng y Rassul Ayani: *Comparison of load balancing strategies on cluster-based web servers.* Simulation, 77(5–6):185–195, Noviembre 2001, ISSN 0037-5497.

[69] Hunt, Guerney D.H., Germán S. Goldszmidt, Richard P. King y Rajat Mukherjee: *Network dispatcher: a connection router for scalable internet services.* Computer Networks and ISDN Systems, 30(1–7):347–357, Abril 1998, ISSN 0169-7552.

[70] Goldszmidt, Germán S. y Guerney D.H. Hunt: *NetDispatcher: A TCP connection router.* Research report RC 20854, IBM Reasearch, T.J. Watson Research Center, New York, Mayo 1997. http://www.cs.princeton.edu/courses/archive/fall03/cs518/papers/networkdispatch.pdf.

[71] Aron, Mohit, Peter Druschel y Willy Zwaenepoel: *Efficient support for P-HTTP in cluster-based web servers.* En *Proceedings of the 1999 USENIX Annual Technical Conference*, páginas 185–198, Monterey, CA, EEUU Junio 1999. USENIX, ISBN 1-880446-33-2. http://www.usenix.org/publications/library/proceedings/usenix99/full$_p$*apers*/*aron*/*aron.pdf*.

[72] Krishnamurthy, Balachander, Richard Liston y Michael Rabinovich: *DEW: DNS-enhanced web for faster content delivery.* En *Proceedings of the twelfth international conference on World Wide Web*, páginas 310–320, Budapest, Hungría Mayo 2003. ACM Press, ISBN 1-58113-680-3.

[73] Barford, Paul y Mark Crovella: *Generating representative web workloads for network and server performance evaluation.* Performance Evaluation Review, 26(1):151–160, Junio 1998, ISSN 0163-5999.

[74] Almeida, Virgilio, Azer Bestavros, Mark Crovella y Adriana de Oliveira: *Characterizing reference locality in the www.* En *Proceedings of the 1996 International Conference on Parallel and Distributed Information Systems*, páginas 92–103, Deciembre 1996.

[75] Zipf, George K.: *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.

[76] Breslau, Lee, Pei Cao, Li Fan, Graham Phillips y Scott Shenker: *Web caching and Zipf-like distributions: Evidence and implications.* En *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM '99)*, volumen 1, páginas 126–134, New Yor, NY, EEUU Marzo 1999. ieee, ISBN 0-7803-5417-6.

[77] Arlitt, Martin F. y Carey L. Williamson: *Internet web servers: Workload characterization and performance implications.* IEEE/ACM Transactions on Networking, 5(5):631–645, Octubre 1997, ISSN 1063-6692.

[78] Trivedi, Kishor S.: *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Wiley, spanish2 edición, 2002, ISBN 0-471-33341-7.

[79] Mah, Bruce A.: *An empirical model of HTTP network traffic.* En *Proceedings of the Conference on Computer Communications (INFOCOM'97)*, volumen 2, páginas 592–600, Kobe, Japón 1997. ISBN 0-8186-7780-5.

[80] Choi, Hyoung Kee y John O. Limb: *A behavioral model of web traffic.* En *Proceedings Seventh International Conference on Network Protocols (ICNP'99)*, páginas 327–334, Toronto, Ontario, Canada, Octubre 1999. IEEE, ISBN 0-7695-0412-4.

[81] Crovella, Mark E.: *Performance characteristics of the world wide web.* En Haring, Günter, Martin Reiser y Christophe Lindemann (editores): *Whitebook on Performance Evaluation*, volumen 1769 de *Lecture Notes in Computer Science*, páginas 219–323. Springer-Verlag, Enero 2000, ISBN 3-540-67193-5.

[82] Floyd, Sally y Vern Paxson: *Difficulties in simulating the internet.* IEEE/ACM Transactions on Networking, 9(4):392–403, Agosto 2001, ISSN 1063-6692.

[83] Paxson, Vern y Sally Floyd: *Wide area traffic: the failure of Poisson modeling.* IEEE/ACM Transactions on Networking, 3(3):226–244, Junio 1995, ISSN 1063-6692.

[84] Wallerich, Jörg: *Design and implementation of a www workload generator for the ns-2 network simulator*, Noviembre 2001. http://www.net.in.tum.de/~jw/nsweb/doku/.

[85] Huberman, Bernardo A., Peter L. T. Pirolli, James E. Pitkow y Rajan M. Lukose: *Strong regularities in world wide web surfing.* Science, 280(5360):95–97, Abril 1998, ISSN 0036-8075.

[86] Deng, Shuang: *Empirical model of www document arrivals at access link.* En *IEEE International Conference on Communications (ICC'96)*, volumen 3, páginas 1797–1802, Dalas, TX, EEUU Junio 1996. IEEE, ISBN 0-7803-3250-4.

[87] Walters, Laurens O.: *A web browsing workload for simulation*, Junio 2004. http://pubs.cs.uct.ac.za/archive/00000137/.