Distribución de carga con tolerancia a fallos en Servidores Distribuidos Heterogéneos

Matias Borgeaud [†], **Laura Berger** [††]
Facultad de Ingeniería, Universidad Nacional de La Pampa, General Pico (6360), Argentina
{borgeaud,berger}@ing.unlpam.edu.ar

Abstract

The new digital information era has stated to us the need of higher computing capabilities, not just storage space but also processing. Even hardware update planning, there is a limiting factor: costs. In this sense, technology could solve the first problem (storage capacity) in a cost-effective way; however, there are many alternatives for the second problem (processing power). The aim of this work is in proposing a mechanism for load distribution among a set of heterogeneous distributed servers with fault-tolerance.

Keywords: load distribution, distributed servers, fault-tolerance.

Resumen

La nueva era digital de la información nos ha planteado la necesidad de mayores capacidades computacionales, no sólo de almacenamiento sino también de procesamiento. Aún planificando actualizaciones del hardware existe, como límite, el factor costo. En este sentido, la tecnología ha probado resolver el primer problema (almacenamiento) de formas económicas y eficaces; sin embargo, para el segundo problema (procesamiento) hay muchas alternativas. El objetivo de este trabajo es proponer un mecanismo de distribución de la carga de procesamiento en un conjunto de servidores heterogéneos distribuidos con tolerancia a fallos.

Palabras claves: distribución de carga, servidores distribuidos, tolerancia a fallos.

---

† Ingeniero en Sistemas por la Universidad Nacional de La Pampa, Argentina.
†† Analista Programador por la Universidad Nacional de La Pampa, Argentina.

1. INTRODUCCIÓN

Planteamos como escenario de estudio una organización que brinda un servicio dado en Internet, por ejemplo, un sitio Web. La organización dispone de un equipo para actuar como servidor. Para hacerlo lo más real posible y abarcar la mayor cantidad de casos especiales, suponemos además que se trata de un sitio Web dinámico que cuenta con el soporte de un servidor de base de datos detrás de la aplicación. Para este escenario consideramos, en un momento dado, un incremento importante en la cantidad habitual de visitas y la imposibilidad (técnica o por los costos) de una inmediata actualización del hardware. Esta situación lleva a una inevitable saturación del servicio por el "cuello de botella" que se genera con el exceso de requerimientos. Sin embargo, existe la posibilidad de distribuir el servicio en varios servidores interconectados[1], por ejemplo, los equipos del personal administrativo.

En un escenario así definido, se buscará entonces implementar un sistema, que permita distribuir el volumen de visitas incrementado, entre los equipos de la organización que se destinen para actuar como servidores, teniendo en cuenta que éstos:

- se encienden y apagan intermitentemente (de forma no programada);
- tienen diferentes capacidades;
- realizan otras tareas (sobre las cuales no puede estimarse la duración o exigencias de cómputo);
- tienen sistemas operativos distintos;
- pueden tener versiones diferentes del software para el servicio, etc.

De este modo, más allá de implementar simplemente un *repartidor*[2] de carga, se buscará realizar un balanceador de carga transparente que soporte heterogeneidad y que cuente con mecanismos de tolerancia a fallos para asegurar alta disponibilidad del servicio. Si bien puede decirse que este trabajo categoriza entre los escritos auto-contenidos, se recomienda consultar [1], [2] y [3] como bibliografía de soporte al tema en general y sobre el área de aplicación del mismo.

1.1. Terminología

En adelante utilizaremos los siguientes términos. En todos los casos, el significado es el mencionado a continuación.

- **Cliente**: un equipo o usuario que realiza peticiones de un servicio a un cierto servidor.
- **Servidor**: un equipo que se encarga de atender peticiones de un servicio, entregando respuestas o resultados a los clientes.
- **Balanceador de carga (LBS)**: un equipo que se encarga de repartir peticiones de un servicio, entre un conjunto de servidores capaces de atenderlas.

[1] Un escenario como el descrito, se conoce como *Farm* o *granja de servidores.*
[2] Habitualmente se conoce como *IP Sprayer*, *Load Dispatcher* o simplemente *Load Balancer Server* (LBS).

- **Granja de servidores**: el conjunto de servidores que trabajan en conjunto atendiendo las peticiones de servicio repartidas por el balanceador de carga, este último incluido.
- **Master**: el equipo responsable de la dirección IP de la granja de servidores.

2. CONCEPTOS PARA LA IMPLEMENTACIÓN

Para el balanceador de carga, prácticamente cualquier equipo es adecuado: no es necesaria una gran capacidad de memoria o procesamiento. Además podrá contar con cualquier sistema operativo.

Los equipos que actuarán como servidor podrán tener cualquier sistema operativo y software de servicio, cualquiera sea su versión. La única restricción es que sirvan el mismo servicio y contenido. Para asegurar lo último se puede usar un filesystem distribuido o programar una sincronización usando, por ejemplo, *rsync* [4], [5].

Dependiendo de la disponibilidad de equipo, un balanceador podría también actuar como servidor.

2.1. Conceptos adicionales

Una cuestión importante es la transparencia con la que opera el sistema. Usaremos una dirección IP pública para publicar el servicio. Detrás de ella, se ocultará el conjunto de servidores y el o los balanceadores de carga. Esta será la dirección que el cliente usará para acceder al servicio sin conocer que será redirigido a otro equipo.

La existencia de más de un balanceador de carga implica que en un determinado momento uno de ellos (y sólo uno) deberá estar activo repartiendo la carga. Usaremos el protocolo VRRP (*Virtual Router Redundancy Protocol*) para asignar una misma dirección IP virtual a todos los balanceadores –la IP pública designada para publicar el servicio-. El protocolo mantendrá un canal de control para asegurar la disponibilidad, así como de mecanismos de elección del responsable de esta dirección y con esto conseguiremos un mecanismo de tolerancia a fallos [6].

3. DESARROLLO DE LA PROPUESTA

Para la implementación del sistema propuesto, se optó por utilizar soluciones provistas para cada objetivo:

1. El software *pen* como balanceador de carga sobre GNU/Linux.
2. El software *vrrpd* como implementación del protocolo VRRPv2 según la rfc2338.
3. Una configuración de direcciones IP públicas/privada como la siguiente:
   a) un rango de direcciones IP privadas para los servidores (*host1*, *host2*, etc.);
   b) una dirección IP pública para publicar el servicio (*www.dummy.org*);

4. IMPLEMENTACIÓN

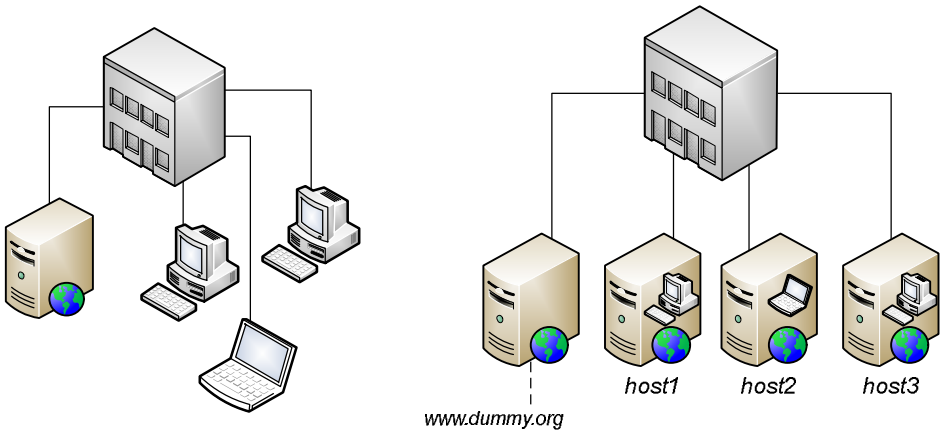
A continuación se detallarán aspectos sobre la implementación, la respectiva funcionalidad y los objetivos que se cubren con cada uno de los componentes de software, de los que se especifican las versiones utilizadas:

pen (versión 0.17.3) `ftp://siag.nu/pub/pen/`
vrrpd (versión 1.0) `http://sourceforge.net/projects/vrrpd/`
apache (versión 1.3.19) `http://httpd.apache.org/`
mysql (versión 3.11) `http://dev.mysql.com/downloads/`
php (versión 3.11) `http://www.php.net/`

Los paquetes correspondientes de *pen* y *vrrpd* se descargaron de los respectivos sitios oficiales. El resto del software –*apache*, *mysql* y *php*- se incluye en la mayoría de las distribuciones actuales de GNU/Linux y prácticamente no hay ninguna diferencia entre usar una versión u otra. Sólo por aspectos recomendables de seguridad, es altamente aconsejable utilizar las más recientes.

4.1. Servidores

El software de servicio se instaló en cada uno de los equipos destinados como servidor, junto con una copia de la aplicación –el sitio web de la empresa- y una copia de la base de datos ("proceso de replicación"). Las figuras 1a y 1b muestran el escenario descrito, antes y después de la instalación de los servidores, respectivamente:



(a) (b)

Figura 1: (a) Escenario antes de la instalación de los servidores y (b) después de la instalación.

Posteriormente se configuró cada servidor para que inicie automáticamente el servicio Web Apache con soporte de lenguaje PHP y el motor de base de datos MySQL. En el sitio web de cada uno de los paquetes de software se puede encontrar información de instalación adicional.

Sólo por referencia, se nombró a los servidores *host1*, *host2*, etc.

4.2. Balanceadores de carga

Un equipo –al que referiremos como *lbs1*- se destinó exclusivamente para actuar como balanceador de carga. Así mismo, *host2* –uno de los servidores de la granja- se configuró como un segundo balanceador.

En estos dos equipos se instaló y configuró *pen* y *vrrpd* para iniciar automáticamente con las siguientes directivas (exactamente las mismas en ambos):

```
#!/bin/sh
#
# /etc/rc.d/rc.local:  Local system initialization script.
#
# Put any local setup commands in here:
#
(snip)
vrrpd –i eth0 –v 1 www.dummy.org
pen –r www.dummy.org:80 host1:80 host2:8080 host3:80
```

Listado 1. Script de inicio /etc/rc.d/rc.local para los balanceadores de carga

En el caso especial del *host2*, se tomó la precaución de cambiar el puerto por defecto para el servicio Web, para permitirle actuar tanto como servidor (puerto 8080) como balanceador de requerimientos Web (puerto 80). La figura 2 muestra el escenario final de la implementación, después de la instalación del software en los balanceadores de carga:

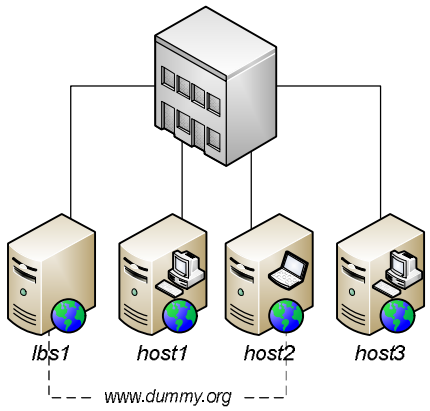


Figura 2: Escenario y topología de la red luego de la instalación de los balanceadores de carga.

En general, con la configuración de *pen* mostrada en el listado 1 (-r = *round robin allocation*), será suficiente para cualquier escenario. Sin embargo, existen otros modos de repartir la carga. En el sitio Web de *pen* se puede encontrar información adicional. Del mismo modo, se puede consultar información complementaria sobre *vrrpd* en el sitio web del software.

La tabla 1 muestra información cuantitativa acerca del total de requerimientos, total de errores (requerimientos no atendidos por sobrecarga), la tasa de error (TE) por sobrecarga (en %), cantidad de transacciones atendidas por segundo (TPS), promedio de transacciones por segundo (TPS), promedio de tiempos de respuesta (PTR) y el máximo tiempo de respuesta, a partir de pruebas de performance y stress sobre los escenarios sin y con mecanismos de distribución de carga. Los test fueron realizados con *ab* y *openload* simulando 15 usuarios simultáneos abriendo 300 conexiones concurrentes. En las figuras 3 y 4 se resumen estos resultados en gráficas comparativas.

Sin distribución de carga			
TPS	**PTPS**	**PTR**	**TE**
141,43	141,43	0,021	0
172	144,49	0,008	0
169,49	146,99	0,006	0
16,99	133,99	0,225	0
166	137,19	0,404	0
14,03	124,88	0,101	0
7,65	113,15	0,615	0
14,68	103,3	0,335	1
0,11	92,99	9,001	0
3,63	84,05	7,275	0
2,95	75,94	6,685	6
0,78	68,42	5,602	28
1	61,68	7,776	12
2,33	55,75	7,26	100
0,67	50,24	3,755	87

Total Requerimientos	1291
Total Errores	19
Máximo TR (segundos)	46,199

Con distribución de carga			
TPS	**PTPS**	**PTR**	**TE**
207	207	0,014	0
216,57	207,96	0,014	0
223,33	209,49	0,013	0
229,54	211,5	0,128	0
228,54	213,2	0,026	0
123,27	204,21	0,04	0
4,68	184,26	1,192	0
0,87	165,92	2,959	0
0,54	149,38	2,996	0
138,32	148,27	0,023	0
222,11	155,66	0,004	0
220,68	162,16	0,089	0
228,77	168,82	0,028	0
226,32	174,57	0,224	0
102,73	167,39	0,031	0

Total Requerimientos	2523
Total Errores	0
Máximo TR (segundos)	9,434

Tabla 1: Comparativa entre un escenario sin y con distribución de carga.

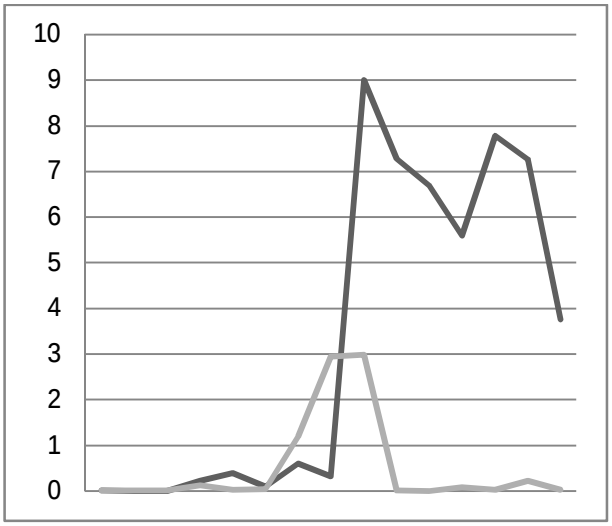


Figura 3: Evolución de la duración (en segundos) de un requerimiento promedio (la línea más oscura corresponde al escenario sin distribución de carga).

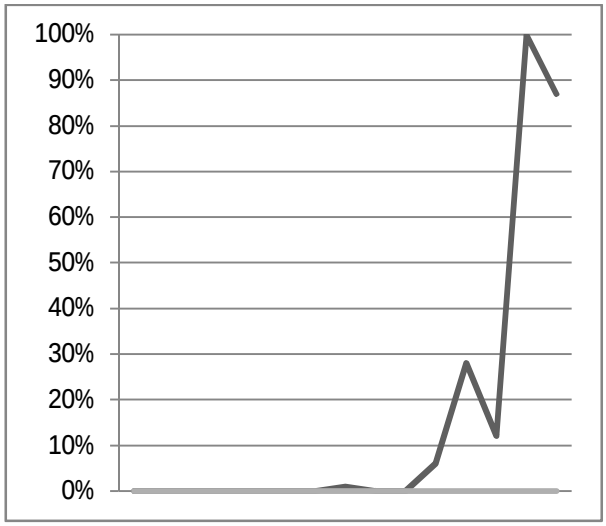


Figura 4: Evolución de la tasa de error (en %) (la línea más oscura corresponde al escenario sin distribución de carga).

5. CONCLUSIONES

Los resultados de la experiencia obtenidos al implementar los conceptos enunciados permitieron escalar un sistema para adaptarse al crecimiento de los requerimientos de un servicio que brinda.

Se logró obtener información precisa que justifica la propuesta: los tiempos de atención disminuyeron un 80% y la tasa de error se mantuvo en 0% con el mecanismo de distribución de carga propuesto (ver resultados en figuras 3 y 4). Esto, en cierta medida, alienta la distribución de carga frente a la actualización del equipo, sobre todo por el grado de escalabilidad que permite el mecanismo, adaptándose rápida y fácilmente al agregar un nuevo servidor.

Mostramos que la implementación del protocolo *vrrpd* en conjunto con *pen* elimina el concepto de "único punto de falla" en el balanceador de carga, dotando al sistema de tolerancia a fallos.

Se mencionaron algunos conceptos básicos de balance de carga. En este sentido se pueden ampliar las mejoras según los requerimientos de la organización en el futuro verificando los resultados obtenidos con otras alternativas de distribución de carga a la propuesta.

6. TRABAJOS FUTUROS

1. Se contemplarán algoritmos de distribución alternativos en futuras versiones.
2. Se compararán resultados obtenidos al usar distintos algoritmos de distribución.
3. Se implementarán scripts de inicialización automatizados para el balanceador de carga.
4. Se buscará consolidar el funcionamiento de la propuesta y simplificar el proceso de instalación y configuración de *pen* y *vrrpd*, sumándolos en un mismo paquete.

REFERENCIAS

[1] **Coulouris G., Dollimore J., Kindberg T.**: *Distributed Systems Concepts and Design*, 3rd ed. Addison-Wesley, (2003)

[2] **Juhász, Z., Kacsuk, P., Kranzlmüller, D.**: Distributed and Parallel Systems – Cluster and Grid Computing, Springer, (2005)

[3] **Rubin, H.**: *The design of a load balancing mechanism for distributed computer systems*. Technical Report CSD-87-362, University of California, Berkeley, (1987)

[4] **Tanembaum, A**.: *Distributed Systems – Principles and Paradigms*. Prentice Hall, c. 10, (2002)

[5] Sitio oficial de *rsync* – manpages. http://rsync.samba.org/ftp/rsync/nightly/rsync.html

[6] The Internet Society, RFC 3768. *Protocolo VRRPv2*. http://tools.ietf.org/html/rfc3768.