Introducción:

Nuestro trabajo trata sobre los balanceadores de carga y sus posibilidades, por lo que trataremos algunos balanceadores de carga distintos a los que hemos tratado en clase.

Balanceadores de carga:

Como hemos visto en la asignatura, un balanceador de carga es un sistema hardware o software que se pone al frente de un conjunto de servidores que atienden una aplicación y asigna las solicitudes que llegan de los clientes a los servidores finales usando algún algoritmo en concreto para evitar cuellos de botella.

Algunos de estos algoritmos son:

Round Robin: algoritmo basado en una cola de turnos. Adecuado si todos tienen una potencia similar.

Menor número de conexiones (Least Connections): Se reparte el trabajo según la utilización de cada una. Se evita la sobrecarga de los servidores que tengan más trabajo.

Ponderación: A cada servidor se le asigna un peso y el balanceador servirá tantas peticiones a cada uno de manera ponderada por ese peso. Es necesario ajustarlo adecuadamente con el rendimiento de cada máquina.

Prioridad: Similar a la ponderación pero con grupos de máquinas.

Tiempo de Respuesta: Se reparte el trabajo según los tiempos de respuesta de cada máquina.

Combinación: Se reparte el trabajo usando una mezcla de los algoritmos anteriores.

A continuación se nombran algunos de los balanceadores software más conocidos en la actualidad:

- **Linux Virtual Server → http://www.LinuxVirtualServer.org/**

Implementa los siguientes algoritmos:
- Planificación por Round-Robin (Round-Robin Scheduling).
- Planificación por Round-Robin ponderado (Weighted Round-Robin Scheduling).
- Planificación por menor número de conexiones (Least-Connection Scheduling).
- Planificación por menor número de conexiones ponderado (Weighted Least-Connection Scheduling).
- Planificación por menor número de conexiones local (Locality-Based Least-Connection Scheduling).

• Planificación por menor número de conexiones local con réplicas (Locality-Based Least-Connection with Replication Scheduling).
• Planificación por hashing de destino (Destination Hashing Scheduling).
• Planificación por hashing de origen (Source Hashing Scheduling).
• Planificación por menor retardo esperado (Shortest Expected Delay Scheduling).
• Planificación por servidores sin peticiones en espera (Never Queue Scheduling).

• **BalanceNG → http://www.inlab.de/balanceng/**

Implementa los siguientes algoritmos:
• Round Robin
• Simple Weighted Round Robin Random
• Weighted Random
• Client Address Hashing
• Least Session
• Least Bandwidth
• Least Resource based on agent supplied information.

Haproxy → http://haproxy.1wt.eu/

Implementa los siguientes algoritmos:
• Round Robin
• Static-rr
• Leastconn
• First
• Source
• Uri
• url_param
• hdr(<name>)
• rdp_cookie

Pen → http://siag.nu/pen/

Implementa los siguientes algoritmos:
• Round Robin
• Weight
• Last time
• Max number of clients

Crossroads Load Balancer → http://crossroads.e-tunity.com/

Implementa los siguientes algoritmos:
• Round Robin

- Least connections
- First available
- External Program
- Strict-Hashed-ip
- Lax-Hashed-ip
- Strict-Stored-ip (last time)
- Lax-Stored-ip
- Weight

Distributor load balancer → http://distributor.sourceforge.net/

Implementa los siguientes algoritmos:
- Round Robin
- Hash
- Httpcookie

No se describen los distintos balanceadores porque prácticamente tienen la misma funcionalidad exceptuando pequeños detalles que se pueden consultar en sus respectivos enlaces.

Escenario:

No hemos conseguido configurar Octopus correctamente ya que daba muchos fallos. Por eso, vamos a poner un ejemplo de cómo se supone que se configura y cómo lo hemos intentado nosotros. Suponemos un escenario con una máquina balanceadora y 4 servidores finales, siendo la máquina balanceadora una máquina con Octopus funcionando como balanceador.

Octopus nos permite establecer los siguientes algoritmos de balanceo:
– Least Load (LL)
– Least Connections (LC)
– Round Robin (RR)
– http URI Hashing (HASH)
– static URI Hashing (STATIC)

La instalación de Octopus requiere descargarse un tar.gz, descomprimirlo y realizar los pasos que pone en el archivo INSTALL que aparece una vez que hemos descomprimido el archivo descargable desde su página web. Los pasos son los siguientes:

◦ $ ./configure --disable-snmp (if compiling without SNMP support)
◦ $ make
◦ $ make install

Como se indica, si la librería SNMP está instalada hay que optar por los siguientes pasos:

- tar zxvf octopuslb-1.14.tar.gz
- $ cd octopuslb-1.14s
- $ ./configure (if you're compiling with SNMP support)

Para la configuración del balanceador hay que acceder al archivo /usr/local/etc/octopuslb.conf mediante nano o cualquier otro editor de Linux y configurarlo de la siguiente manera:

```
binding_port=80
algorithm=RR //Aquí pondremos el algoritmo que vayamos a usar
default_maxc=1000
shm_run_dir=/var/run/octopuslb
log_file=/var/log/octopuslb.log

[granja-nodo-1]
ip=192.168.2.001
port=80
maxc=1000
status=enabled
clone=false
[/]

[granja-nodo-2]
ip=192.168.2.002
port=80
maxc=1000
status=enabled
clone=false
[/]

[granja-nodo-3]
ip=192.168.2.003
port=80
maxc=1000
status=enabled
clone=false
[/]

[granja-nodo-4]
```

ip=192.168.2.004
port=80
maxc=1000
status=enabled
clone=false
[/]

Cada parámetro establece lo siguiente:

binding_port: El puerto de escucha que va a usar el balanceador.
algorithm: El algoritmo que se va a usar para balancear. Puede ser RR (Round Robin), LC (Least Connections), LL (Least Load), HASH HTTP URI y STATIC (muy similar al anterior).
default_maxc: define el número máximo de sesiones del servidor.
ip: la ip del servidor al que nos estamos refiriendo.
port: puerto de escucha del servidor.
maxc: número máximo de sesiones.
status: indica si el servidor está habilitado.
clone: indica si este servidor va a hacer de máquina de apoyo si algún servidor cae.

Después de esto, arrancamos el servicio mediante la siguiente orden:

octopuslb-server -c /usr/local/etc/octopuslb.conf