DNS based Load Balancing with Fault Tolerance

by

Kåre Presttun, Oslo, Norway
Kare@Presttun.org

Introduction

There are several load-balancing systems on the market. They range from local switch based load balancers to sophisticated appliances measuring server performance and distributing load according to available resources. Many sites also use DNS (Domain Name System) based load balancing using round-robin DNS. Some of the sophisticated techniques are rather expensive, while round-robin DNS does not give the fault tolerance one would like to have.

This brief note describes a way to achieve both load balancing and fault tolerance using purely DNS techniques. It can therefore be implemented with minimum cost.

Idea Outline

The traditional round-robin DNS load balancing is achieved by assigning the hostname, say www.example.com, one IP address per server one want to balance the load across. Each time the DNS server is queried it returns the IP addresses in a different sequence in a round-robin fashion. The free DNS server BIND (Berkley Internet Name Domain) which is maintained by ISC (Internet Software Consortium) supports this in addition to fixed sequence and random sequence (using `rrset-order`). Se a discussion about load balancing at the ISC site: http://www.isc.org/products/BIND/docs/bind-load-bal.html. If one balances over several hosts, and one of them is down, a certain part of the users will meet a dead service. This is normally not desirable.

The idea is to use the fault tolerance inherent in the DNS system (the idea came along while reading some white papers at http://www.radware.com/ and the Lisa-95 paper [3]). So the root servers delegate example.com to the DNS servers of example.com, say dns1.examle.com and dns2.examle.com. The DNS servers of example.com further delegate www.example.com as a sub-domain to DNS servers running on the web servers themselves (dns1www.expample.com and dns2www.example.com). These servers have a very short TTL set in the zone (typical 10s), and they only resolve the zone www.example.com to their own IP address.

What happens is that the client will try to resolve www.example.com via some name-server close to it which will locate dns1 and dns2 of the example.com domain. The query will be referred to the DNSs running on the web servers, each of which will only return their own IP address. If the server is down it will not answer queries and the fault tolerance mechanisms of DNS will result in another server being queried which will reply with his own IP address. In order for the A record replies to time out quickly from cache to allow for fall over to the other server, the TTL for the zones on the web servers themselves must be kept short.

Augmenting

Naturally there may be other reasons for a web server being down than the line is down or the server is down. The web daemon may be down or the database may be down etc. By augmenting the solution with a watchdog supervising these critical processes reliability may be improved. The watchdog could try to restart the supervised processes and if this fails it could close down the DNS daemon.

Applicability

The technique is applicable to any service like mail, web etc.

Example Zone Files

Say you have dns1.examle.com and dns2.examle.com running at 192.168.0.1 and 172.16.0.1 respectively. You have the two web servers you want to load-balance running on 192.168.0.10 and 172.16.0.10 respectively, and you have a mail-gateway running at 192.168.0.5. Here is what the zone files will look like:

```
;
; example.com zone
;
$TTL 804800   ; 7 days
; values from ripe-203 document
@       3600  SOA  dns1.example.com.  hostmaster.example.com. (
              2002081601 ; serial YYYYMMDDnn
              86400 ; refresh ( 24 hours)
              7200 ; retry ( 2 hours)
              3600000 ; expire (1000 hours)
              172800 ; minimum ( 2 days)
              )
              NS   dns1
              NS   dns2
              MX   10  mail
```

```
dns1            A    192.168.0.1
dns2            A    172.16.0.1
mail            A    192.168.0.5
www             NS   dns1www
                NS   dns2www
                MX   10  mail
dns1www         A    192.168.0.10
dns2www         A    172.16.0.10
localhost       A    127.0.0.1

;
; www.example.com zone on dns1www
;
$TTL 10         ; 10 s
@               SOA  dns1www.example.com.  hostmaster.example.com. (
                2002081601 ; serial YYYYMMDDnn
                86400 ; refresh ( 24 hours)
                7200 ; retry ( 2 hours )
                3600000 ; expire (1000 hours)
                10 ; minimum ( 10 s )
                )
                A    192.168.0.10
localhost       A    127.0.0.1

;
; www.example.com zone on dns2www
;
$TTL 10         ; 10 s
@               SOA  dns2www.example.com.  hostmaster.example.com. (
                2002081601 ; serial YYYYMMDDnn
                86400 ; refresh ( 24 hours)
                7200 ; retry ( 2 hours)
                3600000 ; expire (1000 hours)
                10 ; minimum ( 10 s )
                )
                A    172.16.0.10
localhost       A    127.0.0.1
```

And that should be it. The refresh, retry, and expire values in the www zone does not matter as there is no secondary server loading the zone.

Your Domain as Web Address

It is gaining popularity to have only the domain used as web address without the www in front. This can also be achieved with this technique but be careful. If you are load-balancing several servers located in one location do not use this technique! This requires your primary DNS servers to run on the web servers, and both (or more) will have different zone files and have to me masters. The zone files which allows web address with and without www looks like:

```
;
; example.com zone on dns1
;
$TTL 804800   ; 7 days
; values from ripe-203 document
@       3600  SOA  dns1.example.com.  hostmaster.example.com. (
              2002081601 ; serial YYYYMMDDnn
              86400 ; refresh ( 24 hours)
              7200 ; retry ( 2 hours)
              3600000 ; expire (1000 hours)
              172800 ; minimum ( 2 days)
              )
              NS   dns1
              NS   dns2
              MX   10  mail
          10  A    192.168.0.10
dns1          A    192.168.0.10
dns2          A    172.16.0.10
mail          A    192.168.0.5
www       10  A    192.168.0.10
              MX   10  mail
localhost     A    127.0.0.1

;
; example.com zone on dns2
;
$TTL 804800   ; 7 days
; values from ripe-203 document
@       3600  SOA  dns2.example.com.  hostmaster.example.com. (
              2002081601 ; serial YYYYMMDDnn
              86400 ; refresh ( 24 hours)
              7200 ; retry ( 2 hours)
              3600000 ; expire (1000 hours)
              172800 ; minimum ( 2 days)
              )
              NS   dns1
              NS   dns2
              MX   10  mail
          10  A    172.16.0.10
dns1          A    192.168.0.10
dns2          A    172.16.0.10
mail          A    192.168.0.5
www       10  A    172.16.0.10
              MX   10  mail
localhost     A    127.0.0.1
```

Security Issues

Running the DNS server on the same server as the web server gives an extra port into the web server and an extra possibility for compromising the web server. It also, and especially in the second example, gives the possibility for taking over your DNS servers via compromise of the web servers. Care should therefore be taken with respect to security. You may want to consider running BIND in a

chroot(). Information on how to do this is available from [5] and [6]. Rob Thomas has a web page with secure BIND template [6], highly recommended.

References

[1] BIND documentation: http://www.isc.org/products/BIND/
[2] Nicolai Langfeldt: DNS and BIND, QUE 2001, ISBN 0-7897-2273-9
[3] Roland J. Schemers: lbnamed: A Load Balancing Name Server in Perl, available at: http://www.stanford.edu/~riepel/lbnamed/
[4] RIPE recommended SOA values: http://www.ripe.net/ripe/docs/ripe-203.html
[5] BIND contributions: http://www.isc.org/products/BIND/contributions.html
[6] Rob Thomas, Secure BIND Template: http://www.cymru.com/Documents/secure-bind-template.htmll