LoadBalancingWithModProxy

(This was originally posted to cocoon-dev by PierFumagalli)

It came out recently on infrastructure at apache dot org a little "joke" about mod_jk versus mod_proxy, which is better, and so on...

Of course, I'm full-on promoting proxy, but my friends Noel J. Bergman and Glenn Nielsen had one question: how do I load balance a pool of backend servlet container using mod_proxy? This is a feature directly supported by mod_jk, but there doesn't seem to be an easy way to do this with more "standard" (or distributed in the core of Apache 2.0) modules.

Well, there is a way: go down and buy a Cisco load balancer, because if you need to have that kind of environment, I'm sure you're going also to have the money to buy one of those nice little gadgets.

But wait, what if (for example) you don't want to go down the hardware route? As I outlined in a page on the Cocoon Wiki available at ApacheModProxy, remember that also mod_rewrite is one *very* good friend.

And in case of load balancing, well, mod_proxy and mod_rewrite come together to offer a pretty optimal solution, and incredibly enough, available with only 5 lines of configurations in your "httpd.conf".

Let's point out that my description is based on the implementation provided by Jetty, as I am familiar with its configuration, but other servlet containers might just provide the same functionality.

The feature "required" is the ability to specify a well-known string in the session manger, so that each generated session will be identifiable by Apache. In Jetty this feature is called "WorkerName" and it is available on all session managers.

To configure it, simply mangle your "jetty.xml" in the following way:

```
<Call name="addWebApplication">
  ... put all the standard parameters to deploy your web application here...
  <!-- The context path spec. Which must be of the form / or /path/* -->
  <Arg>/</Arg>
  <!-- The Web application directory or WAR file. -->
  <Arg>
    <SystemProperty name="jetty.home" default="."/>/cocoon/
  </Arg>
  <!-- This will set the session manager worker name to "LB1" -->
  <Call name="getWebApplicationHandler">
    <Call name="getSessionManager">
      <Set name="workerName" type="string">LB1</Set>
    </Call>
  </Call>
</Call>
```

It is even easier in Tomcat (edit server.xml find the Engine element and add an attribute called jvmRoute. Set its value to LB1.):

```
<Engine name="MainEngine" defaultHost="localhost" jvmRoute="LB1">
```

Basically, for each backend servlet container, we're going to have specific worker names, in this example LB1, LB2, LB3, and so on...

The session ID generated by Jetty will then be something like this: jsessionid=XXXXXX.LB1, jsessionid=XXXXXX.LB2, jsessionid=XXXXXX.LB3, ...

Easy enough and recognizable.

Now, to the Apache configuration. The first thing we have to do is to create a "RewriteMap" file where all the different associations LBx -> hostname:port will be kept. For example, I call mine "balancing.conf", and here is what I have inside:

```
LB1  backend1.betaversion.org:8080
LB2  backend2.betaversion.org:8080
LB3  backend2.betaversion.org:8888
ALL  backend1.betaversion.org:8080|backend2.betaversion.org: 8080|backend2.betaversion.org:8888
```

For each "worker name", I associate the specific host:port where the worker is connected to via HTTP. Easy enough. The only thing to notice is the "ALL" entry, which is a list of all backend host:port entries, separated by pipe | characters. We need this for load balancing.

Now, in addition to all I've previously said on mod_proxy in the Wiki, this are the relevant bits to make sure that Apache 2.0 will load balance your web applications on all backend machnes, and that it will do so by also keeping session affinity.

These are the relevant things in my httpd.conf:

```
RewriteMap SERVERS rnd:/Library/Services/Apache/conf/servers.conf

<Location /webapp>
    RewriteEngine On

    RewriteCond "%{HTTP_COOKIE}"                "(^|;\s*)jsessionid=\w*\.(\w+)($|;)"
    RewriteRule "(.*)"                          "http://${SERVERS:%2}%{REQUEST_URI}"  [P,L]
    RewriteRule "^.*;jsessionid=\w*\.(\w+)($|;)"  "http://${SERVERS:$1}%{REQUEST_URI}"  [P,L]
    RewriteRule "(.*)"                          "http://${SERVERS:ALL}%{REQUEST_URI}" [P,L]
</Location>
```

These rewrite rules can be marginally improved like this (PMB):

```
    RewriteCond "%{HTTP_COOKIE}"                "(^|;\s*)jsessionid=\w*\.(\w+)($|;)"
    RewriteRule "(.*)"                          "http://${SERVERS:%2|${SERVERS:ALL}}%{REQUEST_URI}"  [P,L]

    RewriteRule "^.*;jsessionid=\w*\.(\w+)($|;)"  "http://${SERVERS:$1|${SERVERS:ALL}}%{REQUEST_URI}"  [P,L]

    RewriteRule "(.*)"                          "http://${SERVERS:ALL}%{REQUEST_URI}" [P,L]
```

This uses a reference to the `ALL` line in the server map as the default value. This is helpful when you've taken a downstream server out of the pool, but people have bookmarked a URL that included a session id pointing to the old server.

The first line (RewriteMap) will simply tell Apache to load a map called "SERVERS" of "rnd" type. Look up what that means in the mod_rewrite configuration, but it basically implies that when I'm going to do a lookup for the server associated with the "ALL" key, I will be given one of the three different backend entries, randomly selected (for load balancing).

Now, inside the <Location> tag, we enable the rewrite engine (first line), and we start prowling around for what we need to do:

The following 2 lines (RewriteCond and RewriteRule) will match a specific "jsessionid=XXXXX.YYY" in the cookies sent by the client, and will use the "YYY" value to look up in the map what host name to use for proxying the request.

Given our example, if the cookie looks like "jsessionid=XXXXX.LB2", the lookup will make it so that the request will be proxied to http://backend2.betaversion.org:8080/.....

Simple, and easy...

The second "RewriteRule" will do exactly the same of the first "RewriteCond/RewriteRule" set, but will actually act on the request URI. This works in case your client doesn't support cookies (or, Cocoon folks, replace jsessionid with continuationid, and you have the flow load balanced in a second!).

So, if a request comes in as /webapp/index.jsp;jsessionid=XXXX.LB3, it will be proxied to http://backend2.betaversion.org:8888/webapp/index.jsp;jsessionid=XXXX.LB3 ...

Simple and easy...

This was all fun for session stickiness (or continuation stickiness in the Cocoon world), but what about load balancing? That's where the third RewriteRule comes into place...

This rule simply takes any request not containing an appropriate session ID (cookie or URL) and will proxy it to ONE of the backend servers at random, therefore literally balancing your requests on a pool of backend servers.

Read the detailed documentation of mod_proxy and mod_rewrite, especially for the RewriteMap directive, and you'll see precisely what I mean...

Last thing, what happens in case of errors? Well, the "ErrorDocument" directive now helps us (again, as in the case of straight proxying). Just remember that configuration/redirection errors will have a status code of 502 (proxy error) and backend failures will have a status code of 503 (Bad Gateway)

Therefore, no fuss, mod_proxy is load balanced, and now my 25 Jetty instances are kicking ass

Tomorrow we're going to talk about auto-failover!