Dynamically configured mass virtual hosting

This document describes how to efficiently serve an arbitrary number of virtual hosts with Apache 1.3.

Topics

Motivation

The techniques described here are of interest if your `httpd.conf` contains many `<VirtualHost>` sections that are substantially the same, for example:

```
NameVirtualHost 111.22.33.44
<VirtualHost 111.22.33.44>
  ServerName www.customer-1.com
  DocumentRoot /www/hosts/www.customer-1.com/docs
  ScriptAlias /cgi-bin/ /www/hosts/www.customer-1.com/cgi-bin
</VirtualHost>
<VirtualHost 111.22.33.44>
  ServerName www.customer-2.com
  DocumentRoot /www/hosts/www.customer-2.com/docs
  ScriptAlias /cgi-bin/ /www/hosts/www.customer-2.com/cgi-bin
</VirtualHost>
# blah blah blah
<VirtualHost 111.22.33.44>
  ServerName www.customer-N.com
  DocumentRoot /www/hosts/www.customer-N.com/docs
  ScriptAlias /cgi-bin/ /www/hosts/www.customer-N.com/cgi-bin
</VirtualHost>
```

The basic idea is to replace all of the static `<VirtualHost>` configuration with a mechanism that works it out dynamically. This has a number of advantages:

1. Your configuration file is smaller so Apache starts faster and uses less memory.
2. Adding virtual hosts is simply a matter of creating the appropriate directories in the filesystem and entries in the DNS - you don't need to reconfigure or restart Apache.

The main disadvantage is that you cannot have a different log file for each virtual host; however if you have very many virtual hosts then doing this is dubious anyway because it eats file descriptors. It is better to log to a pipe or a fifo and arrange for the process at the other end to distribute the logs to the customers (it can also accumulate statistics, etc.).

Overview

A virtual host is defined by two pieces of information: its IP address, and the contents of the `Host:` header in the HTTP request. The dynamic mass virtual hosting technique is based on automatically inserting this information into the pathname of the file that is used to satisfy the request. This is done

most easily using mod_vhost_alias, but if you are using a version of Apache up to 1.3.6 then you must use mod_rewrite. Both of these modules are disabled by default; you must enable one of them when configuring and building Apache if you want to use this technique.

A couple of things need to be `faked' to make the dynamic virtual host look like a normal one. The most important is the server name which is used by Apache to generate self-referential URLs, etc. It is configured with the ServerName directive, and it is available to CGIs via the SERVER_NAME environment variable. The actual value used at run time is controlled by the UseCanonicalName setting. With UseCanonicalName Off the server name comes from the contents of the Host: header in the request. With UseCanonicalName DNS it comes from a reverse DNS lookup of the virtual host's IP address. The former setting is used for name-based dynamic virtual hosting, and the latter is used for IP-based hosting. If Apache cannot work out the server name because there is no Host: header or the DNS lookup fails then the value configured with ServerName is used instead.

The other thing to `fake' is the document root (configured with DocumentRoot and available to CGIs via the DOCUMENT_ROOT environment variable). In a normal configuration this setting is used by the core module when mapping URIs to filenames, but when the server is configured to do dynamic virtual hosting that job is taken over by another module (either mod_vhost_alias or mod_rewrite) which has a different way of doing the mapping. Neither of these modules is responsible for setting the DOCUMENT_ROOT environment variable so if any CGIs or SSI documents make use of it they will get a misleading value.

Simple dynamic virtual hosts

This extract from httpd.conf implements the virtual host arrangement outlined in the Motivation section above, but in a generic fashion using mod_vhost_alias.

```
# get the server name from the Host: header
UseCanonicalName Off

# this log format can be split per-virtual-host based on the first
field
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# include the server name in the filenames used to satisfy requests
VirtualDocumentRoot /www/hosts/%0/docs
VirtualScriptAlias /www/hosts/%0/cgi-bin
```

This configuration can be changed into an IP-based virtual hosting solution by just turning UseCanonicalName Off into UseCanonicalName DNS. The server name that is inserted into the filename is then derived from the IP address of the virtual host.

A virtually hosted homepages system

This is an adjustment of the above system tailored for an ISP's homepages server. Using a slightly more complicated configuration we can select substrings of the server name to use in the filename so that e.g. the documents for www.user.isp.com are found in /home/user/. It uses a single cgi-bin directory instead of one per virtual host.

```
# all the preliminary stuff is the same as above, then

# include part of the server name in the filenames
VirtualDocumentRoot /www/hosts/%2/docs
```

```
# single cgi-bin directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```

There are examples of more complicated `VirtualDocumentRoot` settings in the `mod_vhost_alias` documentation.

Using more than one virtual hosting system on the same server

With more complicated setups you can use Apache's normal `<VirtualHost>` directives to control the scope of the various virtual hosting configurations. For example, you could have one IP address for homepages customers and another for commercial customers with the following setup. This can of course be combined with conventional `<VirtualHost>` configuration sections.

```
UseCanonicalName Off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory /www/commercial>
  Options FollowSymLinks
  AllowOverride All
</Directory>

<Directory /www/homepages>
  Options FollowSymLinks
  AllowOverride None
</Directory>

<VirtualHost 111.22.33.44>
  ServerName www.commercial.isp.com

  CustomLog logs/access_log.commercial vcommon

  VirtualDocumentRoot /www/commercial/%0/docs
  VirtualScriptAlias /www/commercial/%0/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.45>
  ServerName www.homepages.isp.com

  CustomLog logs/access_log.homepages vcommon

  VirtualDocumentRoot /www/homepages/%0/docs
  ScriptAlias /cgi-bin/ /www/std-cgi/
</VirtualHost>
```

More efficient IP-based virtual hosting

After the first example I noted that it is easy to turn it into an IP-based virtual hosting setup. Unfortunately that configuration is not very efficient because it requires a DNS lookup for every request. This can be avoided by laying out the filesystem according to the IP addresses themselves rather than the corresponding names and changing the logging similarly. Apache will then usually not need to work out the server name and so incur a DNS lookup.

```
# get the server name from the reverse DNS of the IP address
```

```
UseCanonicalName DNS

# include the IP address in the logs so they may be split
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# include the IP address in the filenames
VirtualDocumentRootIP /www/hosts/%0/docs
VirtualScriptAliasIP /www/hosts/%0/cgi-bin
```

Using older versions of Apache

The examples above rely on `mod_vhost_alias` which appeared after version 1.3.6. If you are using a version of Apache without `mod_vhost_alias` then you can implement this technique with `mod_rewrite` as illustrated below, but only for Host:-header-based virtual hosts.

In addition there are some things to beware of with logging. Apache 1.3.6 is the first version to include the `%V` log format directive; in versions 1.3.0 - 1.3.3 the `%v` option did what `%V` does; version 1.3.4 has no equivalent. In all these versions of Apache the `UseCanonicalName` directive can appear in `.htaccess` files which means that customers can cause the wrong thing to be logged. Therefore the best thing to do is use the `%{Host}i` directive which logs the `Host:` header directly; note that this may include `:port` on the end which is not the case for `%V`.

Simple dynamic virtual hosts using `mod_rewrite`

This extract from `httpd.conf` does the same thing as the first example. The first half is very similar to the corresponding part above but with some changes for backward compatibility and to make the `mod_rewrite` part work properly; the second half configures `mod_rewrite` to do the actual work.

There are a couple of especially tricky bits: By default, `mod_rewrite` runs before the other URI translation modules (`mod_alias` etc.) so if they are used then `mod_rewrite` must be configured to accommodate them. Also, mome magic must be performed to do a per-dynamic-virtual-host equivalent of `ScriptAlias`.

```
# get the server name from the Host: header
UseCanonicalName Off

# splittable logs
LogFormat "%{Host}i %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

<Directory /www/hosts>
  # ExecCGI is needed here because we can't force
  # CGI execution in the way that ScriptAlias does
  Options FollowSymLinks ExecCGI
</Directory>

# now for the hard bit

RewriteEngine On

# a ServerName derived from a Host: header may be any case at all
RewriteMap lowercase int:tolower

## deal with normal documents first:
# allow Alias /icons/ to work - repeat for other aliases
RewriteCond %{REQUEST_URI} !^/icons/
```

```
# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/
# do the magic
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/docs/$1

## and now deal with CGIs - we have to force a MIME type
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/cgi-bin/$1
[T=application/x-httpd-cgi]

# that's it!
```

A homepages system using `mod_rewrite`

This does the same thing as the second example.

```
RewriteEngine on

RewriteMap lowercase int:tolower

# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/

# check the hostname is right so that the RewriteRule works
RewriteCond ${lowercase:%{SERVER_NAME}} ^www\.[a-z-]+\.isp\.com$

# concatenate the virtual host name onto the start of the URI
# the [C] means do the next rewrite on the result of this one
RewriteRule ^(.+) ${lowercase:%{SERVER_NAME}}$1 [C]

# now create the real file name
RewriteRule ^www\.([a-z-]+)\.isp\.com/(.*) /home/$1/$2

# define the global CGI directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```

Using a separate virtual host configuration file

This arrangement uses more advanced `mod_rewrite` features to get the translation from virtual host to document root from a separate configuration file. This provides more flexibility but requires more complicated configuration.

The `vhost.map` file contains something like this:

```
www.customer-1.com /www/customers/1
www.customer-2.com /www/customers/2
# ...
www.customer-N.com /www/customers/N
```

The `http.conf` contains this:

```
RewriteEngine on

RewriteMap lowercase int:tolower

# define the map file
RewriteMap vhost txt:/www/conf/vhost.map
```

```
# deal with aliases as above
RewriteCond %{REQUEST_URI} !^/icons/
RewriteCond %{REQUEST_URI} !^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
# this does the file-based remap
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^/(.*)$ %1/docs/$1

RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^/(.*)$ %1/cgi-bin/$1
```