An In-Depth Discussion of Virtual Host Matching

The virtual host code was completely rewritten in **Apache 1.3**. This document attempts to explain exactly what Apache does when deciding what virtual host to serve a hit from. With the help of the new `NameVirtualHost` directive virtual host configuration should be a lot easier and safer than with versions prior to 1.3.

If you just want to *make it work* without understanding how, here are some examples[1].

Topics

Config File Parsing

There is a *main_server* which consists of all the definitions appearing outside of `<VirtualHost>` sections. There are virtual servers, called *vhosts*, which are defined by `<VirtualHost>` sections.

The directives `Listen`, `ServerName`, `ServerPath`, and `ServerAlias` can appear anywhere within the definition of a server. However, each appearance overrides the previous appearance (within that server).

The default value of the `Listen` field for main_server is 80. The main_server has no default `ServerPath`, or `ServerAlias`. The default `ServerName` is deduced from the servers IP address.

The main_server Listen directive has two functions. One function is to determine the default network port Apache will bind to. The second function is to specify the port number which is used in absolute URIs during redirects.

Unlike the main_server, vhost ports *do not* affect what ports Apache listens for connections on.

Each address appearing in the `VirtualHost` directive can have an optional port. If the port is unspecified it defaults to the value of the main_server's most recent `Listen` statement. The special port `*` indicates a wildcard that matches any port. Collectively the entire set of addresses (including multiple `A` record results from DNS lookups) are called the vhost's *address set*.

Unless a `NameVirtualHost` directive is used for a specific IP address the first vhost with that address is treated as an IP-based vhost. The IP address can also be the wildcard `*`.

If name-based vhosts should be used a `NameVirtualHost` directive *must* appear with the IP address set to be used for the name-based vhosts. In other words, you must specify the IP address that holds the hostname aliases (CNAMEs) for your name-based vhosts via a `NameVirtualHost` directive in your configuration file.

Multiple `NameVirtualHost` directives can be used each with a set of `VirtualHost` directives but only one `NameVirtualHost` directive should be used for each specific IP:port pair.

The ordering of `NameVirtualHost` and `VirtualHost` directives is not important which makes the following two examples identical (only the order of the `VirtualHost` directives for *one* address set is important, see below):

```
NameVirtualHost
111.22.33.44
```

```
<VirtualHost 111.22.33.44>
# server A
```

```
<VirtualHost 111.22.33.44>
# server A
...
</VirtualHost>
<VirtualHost 111.22.33.44>
# server B
...
</VirtualHost>

NameVirtualHost
111.22.33.55
<VirtualHost 111.22.33.55>
# server C
...
</VirtualHost>
<VirtualHost 111.22.33.55>
# server D
...
</VirtualHost>
```

```
</VirtualHost>
<VirtualHost 111.22.33.55>
# server C
...
</VirtualHost>
<VirtualHost 111.22.33.44>
# server B
...
</VirtualHost>
<VirtualHost 111.22.33.55>
# server D
...
</VirtualHost>

NameVirtualHost
111.22.33.44
NameVirtualHost
111.22.33.55
```

(To aid the readability of your configuration you should prefer the left variant.)

After parsing the `VirtualHost` directive, the vhost server is given a default `Listen` equal to the port assigned to the first name in its `VirtualHost` directive.

The complete list of names in the `VirtualHost` directive are treated just like a `ServerAlias` (but are not overridden by any `ServerAlias` statement) if all names resolve to the same address set. Note that subsequent `Listen` statements for this vhost will not affect the ports assigned in the address set.

During initialization a list for each IP address is generated and inserted into an hash table. If the IP address is used in a `NameVirtualHost` directive the list contains all name-based vhosts for the given IP address. If there are no vhosts defined for that address the `NameVirtualHost` directive is ignored and an error is logged. For an IP-based vhost the list in the hash table is empty.

Due to a fast hashing function the overhead of hashing an IP address during a request is minimal and almost not existent. Additionally the table is optimized for IP addresses which vary in the last octet.

For every vhost various default values are set. In particular:

1. If a vhost has no `ServerAdmin`, `ResourceConfig`, `AccessConfig`, `Timeout`, `KeepAliveTimeout`, `KeepAlive`, `MaxKeepAliveRequests`, or `SendBufferSize` directive then the respective value is inherited from the main_server. (That is, inherited from whatever the final setting of that value is in the main_server.)
2. The "lookup defaults" that define the default directory permissions for a vhost are merged with those of the main_server. This includes any per-directory configuration information for any module.
3. The per-server configs for each module from the main_server are merged into the vhost server.

Essentially, the main_server is treated as "defaults" or a "base" on which to build each vhost. But the positioning of these main_server definitions in the config file is largely irrelevant -- the entire config of the main_server has been parsed when this final merging occurs. So even if a main_server definition appears after a vhost definition it might affect the vhost definition.

If the main_server has no `ServerName` at this point, then the hostname of the machine that httpd is running on is used instead. We will call the *main_server address set* those IP addresses returned by a DNS lookup on the `ServerName` of the main_server.

For any undefined `ServerName` fields, a name-based vhost defaults to the address given first in the `VirtualHost` statement defining the vhost.

Any vhost that includes the magic `_default_` wildcard is given the same `ServerName` as the main_server.

Virtual Host Matching

The server determines which vhost to use for a request as follows:

Hash table lookup

When the connection is first made by a client, the IP address to which the client connected is looked up in the internal IP hash table.

If the lookup fails (the IP address wasn't found) the request is served from the `_default_` vhost if there is such a vhost for the port to which the client sent the request. If there is no matching `_default_` vhost the request is served from the main_server.

If the IP address is not found in the hash table then the match against the port number may also result in an entry corresponding to a `NameVirtualHost *`, which is subsequently handled like other name-based vhosts.

If the lookup succeeded (a corresponding list for the IP address was found) the next step is to decide if we have to deal with an IP-based or a name-base vhost.

IP-based vhost

If the entry we found has an empty name list then we have found an IP-based vhost, no further actions are performed and the request is served from that vhost.

Name-based vhost

If the entry corresponds to a name-based vhost the name list contains one or more vhost structures. This list contains the vhosts in the same order as the `VirtualHost` directives appear in the config file.

The first vhost on this list (the first vhost in the config file with the specified IP address) has the highest priority and catches any request to an unknown server name or a request without a `Host:` header field.

If the client provided a `Host:` header field the list is searched for a matching vhost and the first hit on a `ServerName` or `ServerAlias` is taken and the request is served from that vhost. A `Host:` header field can contain a port number, but Apache always matches against the real port to which the client sent the request.

If the client submitted a HTTP/1.0 request without `Host:` header field we don't know to what server the client tried to connect and any existing `ServerPath` is matched against the URI from the request. The first matching path on the list is used and the request is served from that vhost.

If no matching vhost could be found the request is served from the first vhost with a matching port number that is on the list for the IP to which the client connected (as already mentioned before).

Persistent connections

The IP lookup described above is only done *once* for a particular TCP/IP session while the name

lookup is done on *every* request during a KeepAlive/persistent connection. In other words a client may request pages from different name-based vhosts during a single persistent connection.

Absolute URI

If the URI from the request is an absolute URI, and its hostname and port match the main server or one of the configured virtual hosts *and* match the address and port to which the client sent the request, then the scheme/hostname/port prefix is stripped off and the remaining relative URI is served by the corresponding main server or virtual host. If it does not match, then the URI remains untouched and the request is taken to be a proxy request.

Observations

- A name-based vhost can never interfere with an IP-base vhost and vice versa. IP-based vhosts can only be reached through an IP address of its own address set and never through any other address. The same applies to name-based vhosts, they can only be reached through an IP address of the corresponding address set which must be defined with a `NameVirtualHost` directive.
- `ServerAlias` and `ServerPath` checks are never performed for an IP-based vhost.
- The order of name-/IP-based, the `_default_` vhost and the `NameVirtualHost` directive within the config file is not important. Only the ordering of name-based vhosts for a specific address set is significant. The one name-based vhosts that comes first in the configuration file has the highest priority for its corresponding address set.
- For security reasons the port number given in a `Host:` header field is never used during the matching process. Apache always uses the real port to which the client sent the request.
- If a `ServerPath` directive exists which is a prefix of another `ServerPath` directive that appears later in the configuration file, then the former will always be matched and the latter will never be matched. (That is assuming that no `Host:` header field was available to disambiguate the two.)
- If two IP-based vhosts have an address in common, the vhost appearing first in the config file is always matched. Such a thing might happen inadvertently. The server will give a warning in the error logfile when it detects this.
- A `_default_` vhost catches a request only if there is no other vhost with a matching IP address *and* a matching port number for the request. The request is only caught if the port number to which the client sent the request matches the port number of your `_default_` vhost which is your standard `Listen` by default. A wildcard port can be specified (*i.e.*, `_default_:*`) to catch requests to any available port. This also applies to `NameVirtualHost *` vhosts.
- The main_server is only used to serve a request if the IP address and port number to which the client connected is unspecified and does not match any other vhost (including a `_default_` vhost). In other words the main_server only catches a request for an unspecified address/port combination (unless there is a `_default_` vhost which matches that port).
- A `_default_` vhost or the main_server is *never* matched for a request with an unknown or missing `Host:` header field if the client connected to an address (and port) which is used for name-based vhosts, *e.g.*, in a `NameVirtualHost` directive.
- You should never specify DNS names in `VirtualHost` directives because it will force your server to rely on DNS to boot. Furthermore it poses a security threat if you do not control the DNS for all the domains listed. There's more information [2] available on this and the next two topics.
- `ServerName` should always be set for each vhost. Otherwise A DNS lookup is required for each vhost.

Tips

In addition to the tips on the DNS Issues[3] page, here are some further tips:

- Place all main_server definitions before any `VirtualHost` definitions. (This is to aid the readability of the configuration -- the post-config merging process makes it non-obvious that definitions mixed in around virtual hosts might affect all virtual hosts.)
- Group corresponding `NameVirtualHost` and `VirtualHost` definitions in your configuration to ensure better readability.
- Avoid `ServerPaths` which are prefixes of other `ServerPaths`. If you cannot avoid this then you have to ensure that the longer (more specific) prefix vhost appears earlier in the configuration file than the shorter (less specific) prefix (*i.e.*, "ServerPath /abc" should appear after "ServerPath /abc/def").

URI References

[1] http://httpd.apache.org/docs-2.1/vhosts/examples.html
[2] http://httpd.apache.org/docs-2.1/dns-caveats.html
[3] http://httpd.apache.org/docs-2.1/dns-caveats.html#tips