Migrating to a Web Filer

Karl L. Swartz | Network Appliance | TR 3012

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Table of Contents

[TR3012]

---

1. Preface to the Reader

Network Appliance continues to evolve its technology and products at a fast pace, with significant new features and performance enhancements introduced every nine months or less over the past three years. This paper reflects the nomenclature and product characteristics at the time of publication. In particular, current NetApp filers supersede the model numbers referenced in this paper.

2. Introduction

A Network Appliance filer (file server) running Data ONTAP™ can be configured for data access by means of multiple protocols. Either NFS or CIFS is included with the filer; the following protocols are available as optional, software-only upgrades.

NFS – Network File System, primarily for Unix clients
CIFS – Common Internet File System (formerly SMB), for Windows Networking clients
HTTP – HyperText Transfer Protocol, for Web browsers

With Data ONTAP release 4.0, HTTP protocol support is limited to the `GET` request, for retrieval of static files only. This represents the overwhelming majority of HTTP accesses for most web sites, so the performance advantages of a Web Filer can significantly improve a site's overall performance. Dynamic Web content (e.g., Common Gateway Interface or CGI scripts) can execute on other Web servers, the performance of which will be improved by virtue of not servicing the typically larger volume of `GET` requests.

This document describes the process of migrating a WWW (World Wide Web) site from a Unix-based server to a NetApp Web Filer. The popular Apache HTTP server is used for discussion purposes, and specific examples are provided. (An NT-based HTTP server would access any files on the filer using the CIFS protocol instead of NFS, and pathname syntax would be different. Otherwise, the changes described are functionally equivalent.)

3. The Basics

This section describes the simple steps needed to get a Web Filer up and running with static Web data.

Example Configuration

The examples start with a single Web server, a Unix system named `unixbox` running a recent release of the Apache HTTP server. The Domain Name Server (DNS) has a CNAME record (an alias) for `www` pointing to `unixbox`. Where fully-qualified domain names are needed, the domain `netapp.com` will be used. This is illustrated in Figure 1.

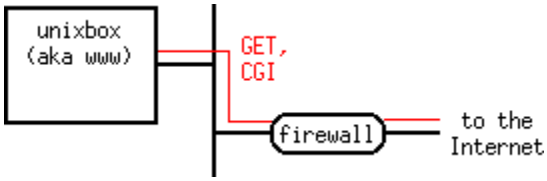

Figure 1. Pre-filer configuration

On `unixbox`, a directory `/usr/local/www` contains a subdirectory, `htdocs`, in which the Web site's HTML, GIF, and other static files reside. Another subdirectory, `cgi-bin`, contains any CGI scripts or binaries.

The second machine is a filer named `toaster`. The filer, with NFS and HTTP licensed and enabled, has a directory which is visible on `unixbox` as `/toaster/home/www`.

Copying Files

The first step is to copy `/usr/local/www/htdocs` to `/toaster/home/www/htdocs` so it is on the filer. Sites which are already using an NFS filer to provide faster, more reliable storage for their WWW files won't need to do anything in this step other than adjust these examples to reflect their own choice of directory name on the filer. Figure 2 shows the resulting configuration. Sites which have their WWW files on a local disk may want to migrate the entire set to the filer, which can be done as follows:

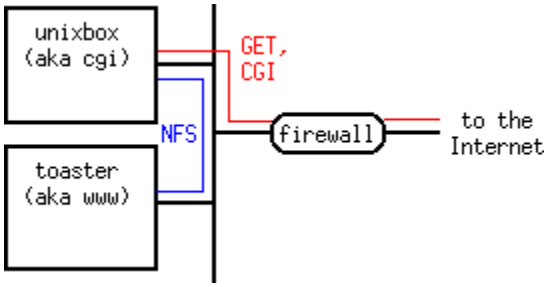

Figure 2. Filer added for NFS

```
cd /usr/local
tar cf - www | (cd /toaster/home; tar xvf -)
mv www www-OLD; ln -s /toaster/home/www www
```

If the HTTP server's log files are in this directory tree, you'll need to take a couple of additional steps. First, delete the copies of the logs on the filer, which on a busy site may be out of date before the copy completes. (The server will still be writing to the local log files even though they've been moved.) Then, notify the server that it should reconfigure itself. A common way of doing this is

```
kill -HUP `cat /usr/local/www/server/logs/httpd.pid`
```

The directory containing `httpd.pid` may vary -- `/var/run` is another common location. You'll need to find it, though, as Apache creates many child processes so it's hard to figure out which PID to use from the output of `ps`.

Trying It Out

You're now ready to try looking at your site's home page on the Web Filer, using the URL `http://toaster`

If this doesn't work, make sure HTTP is enabled on the filer and the root directory is correctly set. If you've installed the HTTP license key, here's what you need to do on the filer's console or via a telnet session to the filer:

```
options httpd.enable on
options httpd.rootdir /home/www/htdocs
```

These commands must be added to the filer's `/etc/rc` if they aren't already there, so that the filer will retain these option settings after a reboot.

If all of your data is static (only HTML, GIF, etc., files) and you have no user pages (URLs like `http://www.netapp.com/~user/`), all that's left to do is to change the DNS pointer for `www` to point to `toaster` instead of `unixbox`. Most sites will have at least a few CGI scripts or programs, image maps, or server-side includes, which will require some attention before putting the Web Filer into production.

4. Advanced Configuration Issues

While getting a Web Filer up with a set of static pages is fast and simple, dynamic pages require a bit more attention. This section discusses in detail the issues involved in porting a site with dynamic pages to a Web Filer environment, with an emphasis on obtaining the best performance so the filer's performance advantage is not wasted.

CGI Scripts

Network Appliance's filers are optimized for the sole task of serving files -- `GET` requests for HTTP, which account for the vast majority of most WWW activity. They cannot run user applications, and thus any CGI scripts or programs must be run elsewhere.
Pairing a general purpose application server with a Web Filer allows each device to focus on the WWW tasks it does best, providing faster response to the end-user while improving reliability. However, some effort is neccessary to implement a paired-server site.

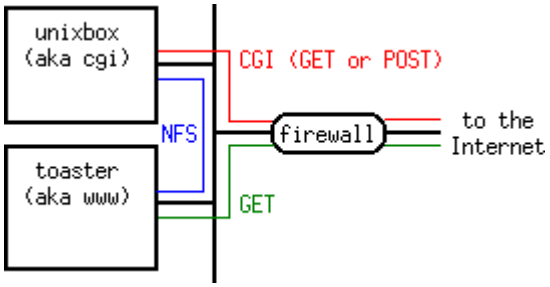

Figure 3. Paired servers

The home page is usually static, not a CGI, so the `www` name should end up pointing to the Web Filer. It's handy to add an alias for the application server so URLs aren't tied to a specific host, so we'll add a DNS entry `cgi` pointing to `unixbox`, with the goal of ending up with the configuration shown in Figure 3. The simplest way to have CGI accesses go to the proper place is to have the filer generate HTTP redirects. This can be done by adding the following line to `/etc/httpd.translations` on the filer:

```
Redirect /cgi-bin/* http://cgi.netapp.com/cgi-bin/*
```

While redirects have the appeal of simplicity, they're not without shortcomings. Most important, they only work for `GET`s -- `POST` requests cannot be redirected. Redirects also cause an additional HTTP transaction, which may cause a substantial additional delay for users accessing a server via high-latency connections. The most subtle problem is that after following a redirect, relative URLs are resolved relative to the URL from the redirect, not the URL of the original reference.

It's easiest to start with the problem of relative references. A hypothetical stock quotation URL `http://www.netapp.com/cgi-bin/stock-quote` generates the following HTML.

```
<head>
<title>Stock Quote for Network Appliance (NTAP)</title>
</head>
<body>
<background href="/background.gif">
<h1>Stock Quote for <a href="/">Network Appliance</a> (NTAP)</h1>
<b>28.5</b> (up 0.5) at 4:01 EDT on Sep 3, 1996
</body>
```

With a redirect, the links to the background and the home page will point to the CGI server, i.e.,

`http://cgi.netapp.com/background.gif`
and
`http://cgi.netapp.com/`
respectively, not to the filer which generated the redirect. If `unixbox`'s `/usr/local/www` is kept on the filer, this won't break anything since both Web servers will serve the same set of static files, though the advantages of using a Web Filer for static data will be lost. Additional redirects could be added on the CGI server to point non-CGI requests back to the filer, but this further exacerbates the problems for high-latency connections. If the CGI scripts can easily be changed, one could turn each one into an absolute URL, but a more efficient solution is to add a base URL into the generated HTML. The modified result would begin

```
<head>
<base href="http://www.netapp.com/">
...
```

This assumes that the relative URLs aren't themselves CGI references. If they are, then *not* adding the base URL will save unnecessary redirects. With a mix of CGI and static references, neither solution is perfect and so a hybrid is appropriate. If only one reference is to a CGI, one might convert that link to an absolute URL and add a base URL for the others.

Addressing the latency concerns of redirects from static files also requires some editing, but fortunately, this can be done fairly easily. The following Unix command and bit of Perl will do the trick:

```
perl -pi.bak -e \
's%( (href|action|src)="?)(/cgi-bin/)%${1}http://cgi.netapp.com$3%g' \
`find /toaster/home/www/htdocs -name '*.html' -print`
```

Since `POST` requests cannot be redirected, must either be converted to `GET` requests or all references to them must be modified to point to the CGI server. Fortunately, CGI references to other sites are almost always `GET`s, so the only `POST`s should come from local pages which can be modified as described above.

Image Maps

NCSA and other servers provide `imagemap` or a similar program to implement image maps -- images which produce different results when you select different parts of the image. This is simply another CGI program, which is handled as discussed in the previous section.

Newer versions of Apache streamline this process by interpreting a `.map` file from the HTML directory in the server. A simple addition to `/etc/httpd.translations` on the filer redirects all map references to the CGI server.

```
Redirect /*.map?* http://cgi.netapp.com/*.map?*
```

If any maps contain relative URLs, they'll need to be pointed back to the filer. This can be done globally with a single Apache configuration directive:

```
ImapBase referrer
```

The same effect can be implemented on a per-map basis by adding

```
base referrer
```

to the map file.

Server-Side Includes

Server-Side Includes (SSIs) cannot be implemented on a filer. A common use of them seems to be to add hit counters to pages. A number of counters are available which use a CGI to generate an in-line graphic, often resembling an odometer. These do not require SSI capability. Other SSI applications will need to be modified to use some other technique.

As implemented in Apache, SSI directives appear as HTML comments if the server does not interpret them. Leaving them unmodified will cause no harm unless the SSI command itself contains sensitive information.

5. Other Hints and Optimizations

Log File Maintenance

The filer HTTP log is in `/etc/log/httpd.log`. Like Apache and other servers, the server never purges the log, so the webmaster must ensure that it does not fill the disk. The only procedural difference with a Web Filer is that there is no need to notify the server that the log has been cycled -- if the file has been renamed, the filer will start a new file the next time it tries to write to it.

The Common Logfile Format is used so existing scripts should work without modification. Two considerations are that the host (the first token in a log entry) is always an IP address and the date (the fourth token) is in GMT, not local time, and thus does not contain a timezone. The Perl script below will filter the log to expand IP addresses into fully-qualified domain names if desired, with the added benefit of only doing one DNS lookup per IP address, reducing name server load.

```
#!/usr/local/bin/perl

while (<>) {
    if (/^(\d+\.\d+\.\d+\.\d+) /) {
        $addr = pack('C4', split(/\./, $1));
        if (!defined($host{$addr})) {
            $name = gethostbyaddr($addr, 2);
            $host{$addr} = ($name ne '' ? $name : $1);
        }
        print "$host{$addr} $'";
    } else {
        print;
    }
}
```

Performance Tuning

File access times are available via the http log file, so updating access times in the filesystem is of little value and potentially generates a large number of otherwise unnecessary disk writes. These writes may adversely impact response time if they compete with disk reads to service HTTP requests. It therefore may be desireable to turn off these updates on a filer used solely as a Web Filer, which may be done using the following filer command:
`options no_atime_update on`
As with other filer options, this command must be added to the filer's `/etc/rc` so the filer will retain this change after a reboot.

Default Welcome Files for Directories

If a URL points to a directory, the server looks in that directory for a file named `index.html`. Apache allows this to be changed, so a site which prefers the CERN convention of `Welcome.html` can change the default with a configuration directive like
`DirectoryIndex Welcome.html`
The Web Filer equivalent is to add the following like to `/etc/httpd.translations`:
`Map */index.html */Welcome.html`

User Pages

Many sites allow users to create their own pages within their home directory space, without any intervention by the webmaster. This is done using the `UserDir` configuration directive. For example,

```
UserDir public_html
allows a user to create a file
~user/public_html/index.html
which is visible using the URL
http://www.netapp.com/~user/
```

If user directories are stored on a filer that's also providing web service, it's tempting to port this by adding the following rule in the filer's `/etc/httpd.translations` file:

```
# This doesn't work
Map /~* /home/*/public_html
```

The current security model of the Web Filer does not allow access to any files outside the *rootdir*, so this doesn't work. (In addition to the `Map` directive, symlinks are followed, but the final file must be within the directory tree rooted at *rootdir*.)

Future Enhancements

The following are features which are being considered for future releases of NetApp Web Filer Software. For more details on the availability of these features, please contact your NetApp sales representative.

Automatic directory listings
Password-protected pages
Virtual hosting
Virtual firewalls

6. Conclusions

A NetApp filer can be used to host all the data for one or more traditional Unix- or NT-based Web servers, using NFS or CIFS. Many sites may find it beneficial to go another step and allow browsers to directly access the static files on the NetApp filer via the HTTP protocol -- a Web Filer. This lightens the load for the traditional web servers, allowing them to apply more horsepower to the execution of dynamic content (CGIs, whether Perl or Java programs or more intensive applications such as database accesses). Implementing a Web Filer is relatively straightforward, as described in this document.

7. Footnotes

1. The examples were developed using Apache 0.8.6 though no changes are expected with 1.1.1, the current version as of this writing. Apache is derived from NCSA 1.3, so with the exception of the new image map support, most of the examples should apply to NCSA 1.3 as well. The NCSA server in turn inherited many features from CERN httpd, now known as W3C httpd, so many examples will work for that servers, too. (According to the Netcraft Survey, as of September 1, 1996, Apache was the most popular server software amongst Internet Web sites, with Apache and NCSA collectively accounting for over half of those Web sites.)
2. Perl is an interpreted programming language for system administration tasks and often used to implement CGI scripts. While it was developed on Unix, it has been widely ported, including a port to Windows NT. The examples work with either Perl version 4 or 5.

3. Updating the access time of a file means the system must write data that eventually goes to disk, even for read-only accesses.

NetApp®

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com