Apache Server on Windows

Talk for ApacheCon Europe 2000. October 23, 2000. London, England.

Rich Bowen <rbowen@rcbowen.com>

This document, and all subsequent improvements thereto, will be available on my web site at http://www.rcbowen.com/imho/apache/

Introduction

Some folks use Windows because they just love it. Many of us use it because we have to - either there is some application that we have to run that only works on Windows, or we work for a company that requires that we run Windows. But the web server that comes with Windows NT, the Microsoft Internet Information Server, leaves much to be desired. And other servers available for NT are expensive, and don't offer anything extra that IIS does not.

Running Apache is a perfect solution for NT users. For those of us that are already familiar with Apache, there's nothing new to learn. For those of us that are in a multi-platform environment, we can run the same server on all of our servers, and have very few changes if we need to move something from one server to another. For those that are not yet familiar with Apache, there is very little that has to be learned to get started.

And, for users that like all of the ``conveniences'' of Windows, Apache has many of those as well - it has a handy Install Shield installation package, and runs as an NT service. And, with Daniel Lopez's wonderful Comanche package, you can even configure it from a GUI.

Disclaimer

I admit, I pretty much stopped using IIS after version 3. I have not tried version 4 or 5 or whatever they are on now. I am perfectly willing to accept that all of my reservations about IIS have been addressed in the latest version. I've already been thoroughly flamed for my comments in this paper, since presenting it in Orlando in March. You should take this as a ``how to'' for Apache, and not as an argument against using IIS, since that's not what I had in mind.

History

Before Apache was available on Windows, you had basically two choices. You could use IIS, which was pretty pathetic, especially in those days. Or you could use Netscape server. The Netscape servers were pretty good, because the folks that were working on them (some of them at least) were the same folks that worked on the NSCA server. But as they made the servers fancier and fancier, they become harder and harder to configure and use.

Finally, in October of 1997 (see http://www.apacheweek.com/issues/97-10-17#13b2) Apache 1.3 came out, with support for Windows NT and Windows 95. And there was much rejoicing. I started running Apache on my NT machines about 2 days after that.

Apache on Windows is ``entirely experimental.''

The Apache documentation contains the following warning:

```
Windows support is entirely experimental, and is recommended only for
experienced users.
```

This is quickly followed by:

```
Apache on NT has not yet been optimized for performance. Apache still
performs best, and is most reliable on Unix platforms.
```

And one can certainly not argue with those assertions. I'd take Apache on my Linux machines over Apache on NT any day. However, the reasonable thing to compare it to is not Apache on Unix, but IIS on NT, since that is really what the choice is.

Since late in 1997, when I started running Apache on NT, I've had a sneaking feeling that Apache was outperforming IIS, but I did not have any real evidence of this. And everything that I read about this seemed to say that I was way off base, and that IIS was much faster.

Back in December, 1999, I did some of my own testing. I was running Apache and IIS on the same server, for a number of reasons that I won't go into here. So, the hardware is identical, and the server load is identical. I did some benchmarking on a day immediately after Christmas, when hardly anyone was in the office - this is an internal server - so server load was not a factor. And I did the tests over a 100MB switched network, so network latency was not really an issue either. But, regardless of that, conditions were identical for each HTTP server.

Using a simple Perl program, I tested performance on the two servers. First, I just fetched a 1K HTML document. Next, I got a simple ``hello world'' Perl CGI program, to test CGI performance. The results were pleasing:

```
D:\Apachecon>perl benchmark.pl
GET

Benchmark: timing 2000 iterations of Apache, IIS...
    Apache: 34 wallclock secs ( 6.71 usr +  4.42 sys = 11.13 CPU)
       IIS: 31 wallclock secs ( 6.75 usr +  4.41 sys = 11.16 CPU)
CGI

Benchmark: timing 2000 iterations of Apache, IIS...
    Apache: 62 wallclock secs ( 6.16 usr +  4.13 sys = 10.28 CPU)
       IIS: 65 wallclock secs ( 6.31 usr +  4.02 sys = 10.33 CPU)

D:\Apachecon>perl benchmark.pl
GET

Benchmark: timing 2000 iterations of Apache, IIS...
    Apache: 34 wallclock secs ( 6.50 usr +  4.55 sys = 11.05 CPU)
       IIS: 31 wallclock secs ( 6.65 usr +  4.43 sys = 11.08 CPU)
CGI

Benchmark: timing 2000 iterations of Apache, IIS...
    Apache: 63 wallclock secs ( 5.73 usr +  3.82 sys =  9.54 CPU)
       IIS: 64 wallclock secs ( 6.15 usr +  3.97 sys = 10.12 CPU)
```

If you're not familiar with the way that the Perl Benchmark module does things, what those results mean is that Apache consistently outperformed IIS on the two simplest things that a web server does - serving HTML pages, and executing CGI programs. Yes, I ran more than just the two tests. These are just sample results. And clearly, a .03 second difference over 2000 iterations is not a huge difference, but it was gratifying to see that Apache consistently came out ahead.

In case you'd like to repeat the tests for yourself, here's the code that I used:

```
use Benchmark;
use LWP::Simple;

print "GET\n\n";
timethese(2000, {
        'Apache' => 'get ("http://9.95.144.25/test.txt";;)',
        'IIS' => 'get ("http://9.95.144.25:90/test.txt";;)',
});

print "CGI\n\n";
timethese(2000, {
        'Apache' => 'get ("http://9.95.144.25/cgi-bin/test.pl";;)',
        'IIS' => 'get ("http://9.95.144.25:90/scripts/test.pl";;)',
});
```

Now, IIS advocates will say that CGI is not the way that you are supposed to do things on IIS. As of this writing, I don't yet know how to do stuff in ASP, so I can't say how it compares to mod_perl. However, the Microsoft claim has always been that IIS is substantially faster than Apache, and yet it can't even serve a HTML document faster, on its own native platform.

So, while it is certainly true that Apache performs better on Unix than on NT, it is no slouch on NT, and compares very well to its competition.

Yet Another Disclaimer

IIS 3. Apache 1.3.12. Perhaps it was not a fair comparison. It was what I had at my disposal at the time. Again, this is not intended to be a ``Apache is better than IIS" diatribe. I'm merely making the point that, for most applications, Apache is plenty fast enough, and compares well to the alternative.

The alternatives

Speaking of the competition, there are several alternatives, when it comes to choosing a HTTP server for NT. Apache compares well to each of them.

IIS (Microsoft)

Well, I've already mentioned IIS, so I won't belabor the point. There is very little advantage to bashing Microsoft. Sure, it's fun, but this is an Apache conference, so I'll try to limit my discussion to Apache. So, I'll be brief.

The obvious advantage of IIS is that you already have it. When you install NT, there's IIS, along for the ride. So it is, in a sense, free. In my huble opinion, the advantages end there. The two main complaints that I have against it are:

- Configuration: There are a very limited number of things that you can configure. The configuration GUI, while moderately easy to use, simply does not give you the range of configuration options that you get with Apache. Apache assumes that you might

want to configure everything, and makes it possible to do that.

- Authentication: IIS uses NT accounts to manage authenticated access to the web server. This might be a good idea on an Intranet (although, personally, I don't think so) but is a really bad idea on a web server out on the internet. Sure, you can lock down NT accounts so that they don't have access to stuff, but creating NT user accounts, and then giving people out on the internet those logins, seems like a recipe for getting hacked. And that's in addition to the nightmare of managing user accounts. *shudder* This is making life difficult for me on a customer project right now.

There are other things that irritate me about IIS, but, like I said, we're here to talk about Apache.

Netscape

According to NetCraft, the next in line is Netscape. I can't say much about Netscape, since I have not run it for 3 years and 2 versions. However, in the pre-Apache days, it was my choice over IIS on NT. This was mainly because of the ways that it handled CGI and authentication, which were much closer to the way that Apache did things.

Website (O'Reilly)

I mention WebSite because many of the people with whom I communicate via email lists say that it is a great server, and well worth the price. However, I have never used it myself, and so cannot comment more intelligently than that.

Others

And, of course, there are many other choices out there. WebServer Compare (http://webcompare.internet.com/) lists 23 HTTP servers that run on NT, and I'm aware of at least one that they missed. There's no shortage of choices.

Differences between Apache Unix and Apache NT

There are some differences between Apache on Unix and Apache on NT. There are some differences in the actual way that the code works, which I won't say much about, and then there are differences in the way that you configure and use the server, which is what this talk is really about.

Threading vs. forking

One of the biggest differences between Apache on Unix and Apache on NT is between threading and forking.

On Unix, Apache forks multiple child processes, each of which listens for and serves requests, maintaining contact with the parent process. After a while, a child will die off, and the parent process will fork a new process to take its place. There are a number of configuration directives that let you control this behavior - how many processes are forked, the maximum, and minimum, number of processes that can be going at any one time, how log

a process is allowed to live, and so on.

On NT, there's no such thing as forking, and so this had to be handled differently. There are two Apache processes running on your NT machine. One of them is the parent process, and the other is the child process that actually handles requests. Within this child process, there are multiple threads, which can serve requests simultaneously. There can be a large number of threads at the same time, and Apache creates additional threads, as necessary, in much the same way that it forks new child processes on Unix.

Whether forking or threading is better, for some value of ``better'', is a discussion for another day. There are strong opinions on either side of the argument, and this is rather beyond the scope of what we're talking about.

General configuration tips

Unix and Windows refer to files differently. Windows has the notion of drive letters, which Unix systems don't have. And, by whatever twist of history, Unix uses forward slashes (/) to denote directories, while Windows/DOS uses back slashes (\).

The general rules are this:

- You don't have to specify the drive letter if the resource is on the same drive letter as your ServerRoot. So, if, for example, you have

  ```
  ServerRoot "c:/httpd"
  ```

  Then you do not need to specify drive letters in other directives

  ```
  Alias perldocs /perl/html
  ```

  But you can if you want to

  ```
  Alias perldocs C:/perl/html
  ```

  And, as with Apache on Unix, relative paths are assumed to be relative to the ServerRoot, and so don't require any disk path.

  ```
  PidFile logs/httpd.pid
  ```

- You should use quotes around any disk path with spaces in it. You should probably avoid using spaces in disk paths, on general principle, but Apache handles them correctly, if you feel the need to use them.

  There was a bug in earlier versions of Apache that caused problems when trying to run CGI programs from paths with spaces in them, which was especially irritating, because Apache installs in ``C:\Program Files\Apache Group\Apache'' That was when I developed the habit of installing in ``C:\httpd'', which has stuck with me, even though the bug has gone away since then.

- Use forward slashes, or back slashes - whichever makes you happiest. You may find some strange places where back slashes appear to not work for you, and you can avoid most of these by using forward slashes, but in most cases, you can use whatever

you're used to.

For example:

```
alias /perldocs c:\perl\html
```

worked, and

```
alias /perldocs /perl/html
```

worked, but

```
alias /perldocs \perl\html
```

did not. I did not really think that this was a bug, since ordinary users would probably not ever use that notation, so this probably does not actually matter to anyone.

NT-specific configuration directives

Because of the different ways that things are handled on Unix and NT, there are some configuration directives that are specific to NT, there are some other directives that don't mean anything on NT, and there are some directives for which the recommended values are different on NT, for whatever reason.

AccessConfig and ResourceConfig

In recent versions of Apache, the old 3-file configuration system has given way to just having one configuration file. There are files called srm.conf and access.conf, but they contain just comments. However, Apache still opens them up when the server starts, and looks for directives. You can tell Apache not to do this, by setting the AccessConfig and ResourceConfig directives to point at /dev/null on Unix. The equivalent of this on Windows is nul. Every directory contains an imaginary file called nul which acts like /dev/null - that is, it's just a bitbucket.

```
AccessConfig nul
ResourceConfig nul
```

`nul` can also be used for directives like AuthGroupFile, and anywhere else that you might use /dev/null on Unix

AccessFileName

The default value for this directive on Unix is .htaccess NT does not care much for filenames that start with a . so the recommended value for this on NT is htaccess. You can get away with using .htaccess if you like, but you might find that some editors will have trouble reading these files, and you might have difficulty actually naming a file that. When I tried to rename a textfile to .htaccess in Windows Explorer, I got a dialog box that said ``You must type a filename.''

LoadModule

Not really a difference here, just a comment. On Unix systems, the LoadModule directive allows you to load a module from an object file. The equivalent of a shared object (so) on

Unix, is a dll file on Windows, and so these directives will look a little different on Windows, but do exactly the same thing.

On Unix:

```
LoadModule status_module modules/mod_status.so
```

On Windows:

```
LoadModule status_module modules/ApacheModuleStatus.dll
```

MaxRequestsPerChild

There are actually mixed messages about this directive in the documentation. http://www.apache.org/docs/windows.html tells you what to set it to (0) and http://www.apache.org/docs/mod/core.html#maxrequestsperchild says that the directive ``has no effect on Win32." This might have been fixed by the time you read this.

What the directive means is how many requests a child should serve before that child exits. However, since on Windows, there is only one child, and it stays active for the length of the Apache process, it would be silly to have the process exit after any number of requests.

The default configuration file that ships with Apache for NT has this set to 0, which means never exit. You should leave it at that value.

MaxSpareServers, MinSpareServers

These directives refer to the number of child processes that should be left lying around idle before they are killed. This is of course meaningless on Windows, and so these directives have no effect on Windows.

ScriptInterpreterSource

On Unix, the first line of a script file, such as a shell script, or a Perl program, indicates what interpreter is to be used to run the program. That line will look something like:

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

or

```
#!/bin/sh
```

Windows has no concept of looking for the #! (``shebang") line, but does everything based on the file extension.

Apache lets you do things either way, as you like. If you are aiming for platform independence, it is very handy to be able to use the #! line, so that you can run your CGI programs on your Windows machine, and also on your Unix machine. But, if you are developing CGI programs only for Windows, then perhaps this does not matter to you, and you want it to execute based on the file extension.

The ScriptInterpreterSource directive tells Apache which one of these you prefer. The default value, script, tells Apache to look for a #! line to find out what interepreter to use to execute the program. Setting it to registry will cause Apache to look in the registry for the association between the file extension and a script interpreter to run it.

Tip: If you will be using Perl CGI programs, and want to maintain some level of portability between your Unix machines and your NT machines, you will want to install Perl in a location on your NT machine that is the same as on your Unix machines. For example, on my Linux machines, Perl is located at /usr/bin/perl and so every Perl program that I write begins with #!/usr/bin/perl So, when I install Perl on an NT machine, instead of installing it in the default location (which is c:\perl for ActivePerl (http://www.activestate.com/)) I install it in C:\usr so that the Perl executable is located at /usr/bin/perl. This allows me to write code on my Windows machine, and then move it, without changes, to my Linux machine, and have it run there. And vice versa.

Directives that don't work on NT (or, at least, work differently)

Many of the directives that work differently under NT are the ones you'd expect - notably the ones dealing with forking child processes. As discussed above, NT does not use fork, but uses threads, to accomplish what Unix-type systems use fork for.

There are some other directives that you might want to use, but which you will need to think about a little differently on NT.

Some of these are just my observations. Perhaps someone can correct me on these.

UserDir

Users on NT have user directories, but, as the comments in httpd.conf note,

```
      # Under Win32, we do not currently try to determine the home
directory of
      # a Windows login, so a format such as that below needs to be
used.  See
      # the UserDir documentation for details.
      #
      UserDir "C:/httpd/users/"
```

What this means is that you should just use UserDir in the normal way, except that you must specify an absolute path.

Under Unix, specifying a relative path causes Apache to append that path to the home directory of the user specified. That is, if the URL requested is http://www.rcbowen.com/~rbowen and UserDir is set to 'public_html', Apache will attempt to serve files from /home/rbowen/public_html.

However, on Win32, there's no nice way to figure out a user's home directory, so that syntax is unavailable.

XBitHack

XBitHack is an enormously useful directive that turns on SSI processing on files with the execute bit set. This allows a happy medium between changing filenames to something.shtml and having every .html file parsed for SSI directives.

Alas, NT does not have an execute bit to set. So, this wonderful directive does not work.

ServerType

Not that anyone ever uses the ServerType directive in real life, but it's worth mentioning that the inetd setting of this directive means absolutely nothing to NT, and so you should not use it. Of course, the Apache documentation now recommends that you not use the inetd setting anywhere else, either, so that's just as well.

Authentication

This is not a big difference, but it is worth mentioning. On Unix, the default encryption scheme used for the password files used for HTTP authentication is Unix crypt. On Windows, it is MD5.

In earlier versions of Apache on Windows, the password files were actually plaintext, and you will still find online documentation that says that this is still the case. Ignore it. Apache for Windows comes with a htpasswd.exe utility that works exactly like the htpasswd utility on Unix, for creating password files. Or you can use modules from the Perl HTTPD-User-Manage package to manage your password files.

Service vs. console

Apache on NT can be run in one of two modes - as a service, or as a console application.

Service

An NT service is a process that is started automatically when NT starts up, and runs in the background as long as NT is running. This is roughly the same as the way that Apache runs on a Unix machine, for all practical purposes. If you're going to run Apache on a production server, you need to run it as a service, so that it will start automatically when you boot your system.

There are a few different ways to install Apache as an NT service.

The easiest is to just select the Install Apache as Service (NT only) option in the Apache folder in your start menu. This installs Apache as an NT service called Apache.

If you want to install it with different service name, or to start with a different configuration than the default, you can install the service from the command line with command line options:

```
apache -i -n "service name"
```

Installs Apache with the service name service name.

```
apache -i -n "service name" -f "\httpd\alternative\httpd.conf"
```

Installs Apache with the service name service name, and it will run with configuration options set in the specified configuration file.

To uninstall the Apache service,

```
apache -u -n "service name"
```

You can start and stop the Apache service in a number of ways. There is a services dialog that can be reached from the NT control panel, where you can press start and stop buttons.

You can send signals to the Apache service from the command line with the commands:

```
apache -n "service name" -k start
apache -n "service name" -k restart
apache -n "service name" -k shutdown
```

Or, you can use the NT net command.

```
net start apache
net stop apache
```

Console

An NT service, as the name suggests, runs on NT. Win95 and Win98 don't have this sort of thing. Thus, Apache has to run as a console application on Win9x. That means that you open a DOS window and invoke the Apache executable from the command line, and it runs in that console window, which must be left open for the duration of the Apache process. This also means that you must be logged onto the machine in order for Apache to be running - it does not start automatically on a reboot, even if you have it in your ``Startup'' folder, unless you first log on.

To stop or restart the Apache process, you need to open up another DOS window, and type

```
apache -k shutdown
```

or

```
apache -k restart
```

Simply closing the window in which Apache is running will cause Apache to exit immediately, without cleaning up after itself. If you do that, the next time you start Apache, it will complain about the pid file not being cleaned up:

```
[Tue Jan 04 21:15:32 2000] [warn] pid file c:/httpd/logs/httpd.pid
overwritten -- Unclean shutdown of previous Apache run?
```

This is probably not a big deal, since you're unlikely to be running production services on a Win9x machine anyway.

But there's no GUI!

One of the common complaints about Apache, and about Unix in general, is that configuration is just too darned difficult. What this usually means is one of two things. Either it is referring to the fact that every stinkin' application has a different configuration file format, or the fact that most of them don't have a nice GUI for configuration. Or both.

Now, while many of us are die-hard Unix bigots, and are quite content with the notion that if software was hard to write, it should be hard to use, these are no longer acceptable notions,

particularly in the NT world, where people are quite content to use an inferior product, if it is easier to use. One does not have to look any farther than NT itself for evidence of this.

Fortunately, there is Comanche. Hopefully some of you were able to attend Daniel Lopez Ridruejo's. Comanche is a GUI for configuring Apache. But not only Apache -- any application that has a text configuration file. There are currently Comanche plug-ins that let you configure Comanche and Samba, and it is fairly simple to write extensions for it to configure anything else.

You can get Comanche at http://comanche.com.dtu.dk/, or just track down Daniel while you're here, and ask him about it.

Using modules on NT Apache

Apache modules on NT are implemented as DLL files, so you can just load the ones that you want, and leave out the ones that you don't, and you don't have to recompile Apache to do this. That's nice. You can load a module with the LoadModule directive.

```
LoadModule speling_module modules/ApacheModuleSpeling.dll
```

The difficulty with modules on Windows is that Windows systems, as a rule, don't come equipped with a compiler. So adding non-standar modules is rather more difficult on Windows systems. For most users, the modules that come with the product are the ones that you're stuck with.

mod_perl

mod_perl, in case you're not familiar with it, is a module that loads the Perl interpreter into memory at server startup, so that Perl CGI programs don't have to pay that startup cost every time they are run. Additionally, mod_perl lets you write Apache modules in Perl.

For more information on mod_perl, see http://perl.apache.org/ or attend one of the many talks about mod_perl.

Installing mod_perl on NT is less than obvious. Fortunately, some kind person (Randy Kobes) has done this work for you, and you can just grab the binaries, and go from there. You can get those binaries at ftp://theoryx5.uwinnipeg.ca/pub/other/ These include Perl from ActiveState, and Apache binaries. There is some manual setup that has to be done, but that should not be too much of a hardship.

The future for NT Apache?

Please attend Bill Stoddard's talk immediately following this one.

Additional information

There's very little information online about Apache on Windows. The main reason for this is that Apache on NT is really not very much different from Apache on Unix, and the documentation covers the differences adequately. However, I suspect that the other reason is that there are not nearly as many people running Apache on NT as on Unix, and those that

would be running it on NT are being scared off by that comment that Apache on Windows is ``entirely experimental."

You'll find a document at http://www.apache.org/docs/windows.html that covers some of the Windows-specific things, and the rest, such as configuration directives, are found at their appropriate places throughout the documentation.

There's really not one central place where Apache on NT is specifically discussed, at least that I've found. Of course, there's this document. While I would not presume to claim that it is complete or authoritative, I think that it covers the main points adequately. Please let me know (rbowen@rcbowen.com) if I've left anything out, or made any glaring errors.