New and improved – Apache 2

Rules of Succession

Whether they start now or in a few months, administrators will soon need to think about making the jump from Apache Server 1.3 to version 2. This article discusses important new features and provides decision making guidelines.

BY THOMAS GRAHAMMER

After two and a half years of development, the Apache Software Foundation [1] finally declared version 2.0.35 of the Apache web server [2] "fit for public use" in April [3]. When this issue went to print, 2.0.40 was the current version. But it seems that many administrators are still not prepared to risk updating.

For those of you still wavering, this article will point out some of the advantages the new web server offers, and those of you who have decided to go for the new version will certainly be interested in avoiding the traps.

What's new?

A whole bunch of new features in Apache 2.0 immediately catch the eye:

- Build system: The Build system now uses autoconf and libtool. The installation follows the usual pattern after expanding the archive: ./configure prefix = *Prefix*, without the "prefix" option the target path is "/usr/local/apache2". This step is followed by "make" and "make install".
  You will need an ANSI C compiler, such as GCC; Perl version 5.003 or better is optional. You can either include the modules you require in the executable like in Apache 1.3 ("--enable-*module*") or load them as Dynamic Shared Objects ("--enable-*module* = shared") at runtime. The new Apache Extension Tool "apxs" is used for creating DSOs.
- Configuration: The administrator modifies the "< *Prefix* > /conf/ httpd.conf" file to configure the program. This file simplifies many of the previously confusing configuration instructions or removes them entirely, as is the case for the "port" and "BindAddress" instructions.
  Now, you only need the "listen" instruction to set IP addresses and port numbers. The server name and the port number, used for redirection and recognizing virtual servers, can be configured using the "ServerName" instruction sometime in future.
- IPv6: Apache will use IPv6 Listening Sockets on systems where the portable runtime library (see the section on APRs) supports IPv6. Additionally, the configuration instructions "Listen", "NameVirtualHost" and "VirtualHost" can handle IPv6 addresses, for example "Listen [fe80::1]:8080".
- Modules: Mod_ssl is now officially part of the Apache package. Mod_proxy has been mostly reprogrammed and several functions have been placed in a number of Mod_*cache modules. The Mod_deflate compressor is new and may replace Mod_gzip (see the article on page 26).
- Filters: Apache modules can now be used to filter ingoing and outgoing data streams. You can thus filter CGI

John C.H. Grabill, www.visipix.com

script output using the "INCLUDES" filter contained in Mod_include, which allows you to execute server side includes.

- Multilingual: Apache stores error messages intended fro client browsers in multilingual SSI documents. The administrator can modify them to reflect corporate design.
- Multi-protocol support: Apache 2.0 provides a platform capable of multi-protocol support. (The documentation for this feature is unfortunately extremely thin and prevented us from ascertaining the practical advantages this feature may offer.)

Under the Hood: Yesterday and Today

Apache supported only the Unix operating system up to version 1.2. In contrast to Microsoft Windows, Unix operating systems are capable of copying processes (so called forking). Apache thus ran as a preforking server: When launched, the parent process creates a number of instances of its own process – as defined in the configuration file – and these process listen for HTTP requests. If the number of requests exceeds the number of processes available to receive them, additional instances of the original process are launched.

Apache version 1.3 was ported to Windows with a great deal of effort (and to Netware 5 and IBMs Transaction Processing Facility). To do so, the developers had to completely re-write the process engine. The so called thread version of Apache 1.3 is required for Windows, for Unix/Linux you can use the process variant.

To facilitate porting Apache 2.0 the developers abstracted the platform specific code segments from the remaining Apache code and placed it in the APR and MPP sources (see below). This kind of modularity also facilitates platform specific optimization.

The Apache Portable Runtime APR

Apache 2 no longer (directly) uses Posix interfaces in contrast to previous Apache versions. Poorly implemented or slow Posix libraries or emulations meant that the server did not perform well on non Unix operating systems.

The Apache Portable Runtime (APR) library introduced to replace Posix was programmed by Apache to place an abstraction layer between the operating system and Apache 2. The API provided by APR contains the basic functionality of a virtual operating system, including file and network I/O, memory management, as well as thread and process management. By preference the APR will always use native calls to the operating system. Additionally, the APR methods emulate the former Posix methods to facilitate porting older code to APR.

The achieved goal of APR Version 1.0 was to provide those functions required for Apache 2.0. There are plans to develop APR independently of Apache as a basis for platform independent program development and to make it available to interested programmers.

Multiprocessing Modules

Apache 2 uses special modules to abstract the code used to manage processes in threads in Version 1.3. It is these Multi Processing Modules' (MPMs) task to pass incoming HTTP requests to simple execution units, which will in turn process the request. The MPM in use specifies whether to use either processes or threads. This kind of modularization provides for a clearly

Figure 1: Useful information on upgrading to 2.0

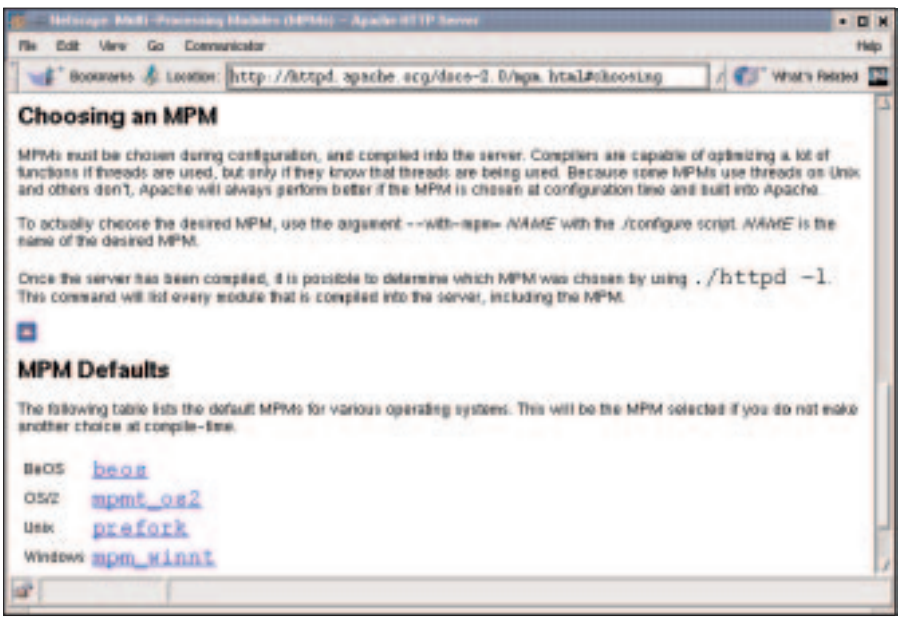

Figure 2: The Apache MPM site

structured Apache 2 source code and offers several other advantages. The Apache developers expressly allow MPMs to use operating system specific code. MPMs of this type can only be used on one operating system, but their performance will be far superior – and this is particularly evident on non-Unix operating systems. Bill Stoddard, an Apache developer, achieved a performance boost of 50 percent for static websites on Windows.

MPMs on Unix

There are several MPMs available for Unix. Each one of them has a different approach to how the web server deals with incoming HTTP requests. Webmasters can therefore choose the variant best suited to their applications by linking the MPM in to the Apache binary (see also "Ready-Made MPMs in Apache 2.0"). Threads use fewer resources than processes on Unix operating systems – and on Linux. On the other hand, a process based approach does provide better stability, as a faulty thread can bring down its parent process.

Update or Install from Scratch?

Since there have been some serious changes to the architecture of the Apache web server, you will definitely want to install from scratch [5], rather than attempting an update. After trying both approaches, I discovered that version 2.0 of the Apache attempts to re-use the original 1.3 modules if you perform an update. Due to changes in the base technology 1.3 modules cannot be used without some modifications.

You will also be unable to re-use the configuration files – particularly "httpd.conf" – as many configuration instructions have been simplified or just removed.

Is it Worth Changing?

Changing versions means a lot of work – you will need to install your Apache from scratch, including completely re-configuring all the settings. The standard version 1.3 modules are available in Apache 2.0 (see figure 1), but existing third party modules will not be available for the time being.

Administrators will have to decide for themselves whether upgrading is worthwhile. If you really use the features we just discussed, such as the Perchild MPM, you should probably go for the upgrade, despite the work involved. If the features merely appeal to you, or if you think they are a waste of time, you might prefer to stick the old adage: Never change a running Apache. ■

Ready-Made MPMs in Apache 2.0

Prefork (default for Unix platforms)

This MPM implements typical Apache 1.3 behavior in Apache 2.0. In this case, a parent process will create a pool of child processes for the incoming HTTP requests. The options "MinSpareServers" and "MaxSpareServers" are used to set the lower and upper limits for the child process pool.

If the number of free processes drops below the number defined in "MinSpareServers", then new processes are launched, if the number exceeds "MaxSpareServers", Apache will remove processes from memory. Since each process will handle only one request, an error in one process will drop only one connection to the server. This is a great advantage if you work with dynamically generated pages.

Threaded

This MPM is similar to Prefork, the main difference being that every Apache 2 process can run multiple threads. The configuration option "ThreadsPerChild" is used to specify how many. If Threaded is used, the httpd keeps count of the number of unused threads remaining.

The "MinSpareThreads" and "MaxSpareThreads" directives stipulate the number of unused threads that can occur before creating new processes or removing processes from memory.

This MPM thus uses multiple processes and threads – a genuine enhancement in comparison to Apache 1.3.

Dexter

The Dexter MPM also uses both processes and threads. In contrast to Threaded, the number of processes is clearly defined, whereas the number of threads per process depends on the current server load. The configuration statement "NumServers" specifies the number of processes to be created on launching the web server, the number of threads is defined in "StartThreads". "MinSpareThreads" and "MaxSpareThreads" define how the number of threads will be adjusted to reflect changing loads on the server.

Perchild

Perchild is based on the Dexter MPM, but adds an additional function that web hosters will appreciate. Just like Dexter, Perchild also uses a set number of processes that spawn threads. In order to run multiple virtual hosts with different privileges, Perchild assigns user and group IDs to the processes.

The "ChildPerUserID" directive specifies how many processes will run under a specific user ID.

While the "AssignUserID" statement within the context of a "VirtualHost" assigns a user ID to a specific process.

Modules for Other Operating Systems

There are additional MPMs for Windows, OS/2 and BeOS, which are automatically linked for the respective platforms by the build system.

INFO

[1] Apache Software Foundation: *http://www.apache.org*

[2] Apache Project: *http://httpd.apache.org*

[3] Release List: *http://www.apacheweek.com/features/ap2#rh*

[4] List of Apache 2 Modules: *http://httpd.apache.org/docs-2.0/mod*

[5] Installation: *http://httpd.apache.org/docs-2.0/install.html*

THE AUTHOR

Thomas Grahammer has a university degree in Computer Science and is in charge of software development at a Munich based software company. He is also a freelance software developer, with in-depth knowledge of databases and Apache, and holds professional qualifications from SuSE and PHP