CHAPTER 1

Apache: The Number One Web Server

✦ ✦ ✦ ✦

In This Chapter

Understanding why Apache rocks

Boning up on Apache history

Thumbing through the feature set

Looking through Apache architecture

Sorting through the licensing options

✦ ✦ ✦ ✦

Welcome to Apache — the number one Web server in the world. If you are toying with the idea of running Apache, you are in the right place! This chapter introduces the Apache way of running a Web server.

More than 60 percent of the Web servers in the world use Apache, according to a prominent Web server survey company called Netcraft (`www.netcraft.co.uk/Survey/`). Netcraft publishes the Top Server statistics periodically. Table 1-1 shows the Top Server statistics that was available at the time of writing this chapter. If you want to put faces to the numbers, you can visit `www.apache.org/info/apache_users.html`.

Table 1-1
Top Server Statistics by Netcraft

Server	*Nov 2001*	*Percent*	*Dec 2001*	*Percent*
Apache	7750275	61.88	8588323	63.34
Microsoft IIS	3307207	26.40	3609428	26.62
iPlanet	431935	3.45	383078	2.83
Zeus	174052	3.45	172352	1.27

Apache Rocks On

What Apache has accomplished is simply amazing! Who knew that an open source Web server could consistently beat two major commercial competitors, Microsoft and Netscape as a Web server platform! Everyone has his or her own reason for why Apache is so popular. Here are mine:

- **Apache is a highly configurable Web Server with a modular design.** It is very easy to extend the capabilities of Apache Web server. Anyone with decent C or Perl programming expertise can write a module to perform a special function. This means that there are tons of Apache modules available for use.
- **Apache is a free, open source technology.** Being free is important but not as important as being open source.
- **Apache works great with Perl, PHP, and other scripting languages.** Most Web applications are still scripts. Perl excels in the script world and Apache makes using Perl a piece of cake with both CGI support and `mod_perl` support.
- **Apache runs on Linux and other Unix systems.** Linux used to be an underdog operating system, which has now found itself in enterprise computing arena. Linux and Apache go hand-in-hand in the enterprise world today. I believe Linux's acceptance in the business world has made Apache's entry into such territory easy. However, there are people who would argue that it was Apache's fame that made Linux find its way into the business world easier. Either way, Apache and Linux is a powerful combination. Other Unix systems such as FreeBSD and Solaris, and the new Mac OS X also play a great role in expanding Apache's user base horizon.
- **Apache also runs on Windows.** Although Apache will run much better on Windows platform with version 2.0, Apache was already in Windows market with Version 1.3.*x.* We will see a lot of Windows systems switching to Apache from Microsoft Internet Information Server (IIS) because Apache 2.0 architecture gives it the power it needed to compete natively.

Apache: The Beginning

Here is a bit of Apache history. In the early days of the Web, the National Center for Super Computing Applications (NCSA) created a Web server that became the number one Web server in early 1995. However, the primary developer of the NCSA Web server left NCSA about the same time, and the server project began to stall. In the meantime, people who were using the NCSA Web server began to exchange their own patches for the server and soon realized that a forum to manage the patches was necessary. The Apache Group was born. The group used the NCSA Web server code and gave birth to a new Web server called Apache. Originally derived from the core code of the NCSA Web server and a bunch of patches, the Apache server is now the talk of the Web server community. In three short years, it acquired the lead server role in the market.

The very first version (0.6.2) of publicly distributed Apache was released in April 1995. The 1.0 version was released on December 1, 1995. The Apache Group has expanded and incorporated as a nonprofit group. The group operates entirely via the Internet. However, the development of the Apache server is not limited in any way by the group. Anyone who has the know-how to participate in the development of the server or its component modules is welcome to do so, although the group is the final authority on what gets included in the standard distribution of what is known as the Apache Web server. This allows literally thousands of developers around the world to come up with new features, bug fixes, ports to new platforms, and more. When new code is submitted to the Apache Group, the group members investigate the details, perform tests, and do quality control checks. If they are satisfied, the code is integrated into the main Apache distribution.

The Apache Feature List

One of the greatest features that Apache offers is that it runs on virtually all widely used computer platforms. At the beginning, Apache used to be primarily a Unix-based Web server, but that is no longer true. Apache not only runs on most (if not all) flavors of Unix, but it also runs on Windows 2000/NT/9*x* and many other desktop and server-class operating systems such as Amiga OS 3.*x* and OS/2.

Apache offers many other features including fancy directory indexing; directory aliasing; content negotiations; configurable HTTP error reporting; SetUID execution of CGI Programs; resource management for child processes; server-side image maps; URL rewriting; URL spell checking; and online manuals.

The other major features of Apache are:

✦ **Support for the latest HTTP 1.1 protocol:** Apache is one of the first Web servers to integrate the HTTP 1.1 protocol. It is fully compliant with the new HTTP 1.1 standard and at the same time it is backward compatible with HTTP 1.0. Apache is ready for all the great things that the new protocol has to offer.

For example, before HTTP 1.1, a Web browser had to wait for a response from the Web server before it could issue another request. With the emergence of HTTP 1.1, this is no longer the case. A Web browser can send requests in parallel, which saves bandwidth by not transmitting HTTP headers in each request. This is likely to provide a performance boost at the end-user side because files requested in parallel will appear faster on the browser.

✦ **Simple, yet powerful file-based configuration:** The Apache server does not come with a graphical user interface for administrators. It comes with single primary configuration file called `httpd.conf` that you can use to configure Apache to your liking. All you need is your favorite text editor. However, it is flexible enough to allow you spread out your virtual host configuration in multiple files so that a single `httpd.conf` does not become too cumbersome to manage with many virtual server configurations.

✦ **Support for CGI (Common Gateway Interface):** Apache supports CGI using the `mod_cgi` and `mod_cgid` modules. It is CGI 1.1 compliant and offers extended features such as custom environment variables and debugging support that are hard to find in other Web servers. See Chapter 12 for details.

✦ **Support for FastCGI:** Not everyone writes their CGI in Perl, so how can they make their CGI applications faster? Apache has a solution for that as well. Use the `mod_fcgi` module to implement a FastCGI environment within Apache and make your FastCGI applications blazing fast. See Chapter 14 for details.

✦ **Support for virtual hosts:** Apache is also one of the first Web servers to support both IP-based and named virtual hosts. See Chapter 6 for details.

✦ **Support for HTTP authentication:** Web-based basic authentication is supported in Apache. It is also ready for message-digest-based authentication, which is something the popular Web browsers have yet to implement. Apache can implement basic authentication using either standard password files, DBMs, SQL calls, or calls to external authentication programs. See Chapter 7 for details.

✦ **Integrated Perl:** Perl has become the de facto standard for CGI script programming. Apache is surely one of the factors that made Perl such a popular CGI programming language. Apache is now more Perl-friendly then ever before. Using its `mod_perl` module, you can load a Perl-based CGI script in memory and reuse it as many times as you want. This process removes the start-up penalties that are often associated with an interpreted language like Perl. See Chapter 16 for details.

✦ **Support for PHP scripting:** This scripting language has become very widely used and Apache provides great support for PHP using the `mod_php` module. See Chapter 15 for details.

✦ **Java Servlet support:** Java servlets and Java Server Pages (JSP) are becoming very commonplace in dynamic Web sites. You can run Java servlets using the award-wining Tomcat environment with Apache. See Chapter 17 for details.

✦ **Integrated Proxy server:** You can turn Apache into a caching (forward) proxy server. However, the current implementation of the optional proxy module does not support reverse proxy or the latest HTTP 1.1 protocol. There are plans for updating this module soon. See Chapter 10 for details.

✦ **Server status and customizable logs:** Apache gives you a great deal of flexibility in logging and monitoring the status of the server itself. Server status can be monitored via a Web browser. You can also customize your log files to your liking. See Chapter 8 for details.

✦ **Support for Server-Side Includes (SSI):** Apache offers set of server side includes that add a great deal of flexibility for the Web site developer. See Chapter 13 for details.

✦ **Support for Secured Socket Layer (SSL):** You can easily create an SSL Web site using OpenSSL and the `mod_ssl` module for Apache. See Chapter 19 for details.

Understanding Apache 2.0 Architecture

Apache Server 2.0 makes Apache a more flexible, more portable, and more scalable Web solution than ever before. The new 2.0 releases offer many improvements; the major improvements are discussed in the following sections.

Multiprocessing modules

The first major change in Apache 2.0 is the introduction of multiprocessing modules (MPMs). To understand why MPMs are created, you need to understand how Apache worked before. Apache Version 1.3 or earlier used a preforking architecture. In this architecture, an Apache parent process forked a set of child processes, which serviced the actual requests. The parent process simply monitored the children and spawned or killed child processes based on the amount of requests received. Unfortunately, this model didn’t work well under platforms that are not process-centric such as Windows. So, the Apache Group came up with the MPM-based solution.

Each MPM is responsible for starting the server processes and for servicing requests via child processes or threads depending on the MPM implementation. Several MPMs are available. They are discussed in the following sections.

The prefork MPM

The prefork MPM mimics the Apache 1.3 or earlier architecture, creating a pool of child processes to service requests. Each child process has a single thread. For example, if Apache starts 30 child processes, it can service 30 requests simultaneously.

If something goes wrong and the child process dies, only a single request is lost. The number of child processes is controlled using a minimum and maximum setting. When the number of requests increases, new child processes are added until the maximum is reached. Similarly, when the requests fall, any extra child processes are killed.

The threaded MPM

This MPM enables thread support in Apache 2.0. This is like the prefork MPM, but instead of each child process having a single thread, each child process is allowed to have a specified number of threads. Each thread within a child process can service a different request. If Apache starts 30 child processes where each child is allowed to have at maximum 10 threads, than Apache can service 30 × 10 = 300 requests simultaneously.

If something goes wrong with a thread, for example, an experimental module causes the thread to die, then the entire process dies. This means that all the requests being serviced by the threads within the child process will be lost. However, because requests are distributed among threads on separate child processes, it is likely that a child's death will take down at maximum of 1/*n* of all the total connection, where *n* presents the number of all simultaneous connections.

A process is added or removed by monitoring its spare-thread count. For example, if a process has less than the minimum number of spare threads, a new process is added. Similarly, when a process has a maximum number of idle threads, it killed.

All processes run under the same user and group ID assigned to Apache server.

Because threads are more resource efficient than processes, this MPM is very scalable.

The perchild MPM

This is also new in Apache 2.0. In this MPM model a set number of child processes are started with a specified number of threads. As request load increases the processes add new threads as needed. When request count reduces, processes shrink their thread counts using a minimum and maximum thread count setting.

The key difference between this module and the threaded MPM is that the process count is static and also each process can run using a different user and group ID. This makes it easy to run different virtual Web sites under different user and group IDs. See Chapter 6 for details.

The winnt MPM

This is the MPM for the Windows platform, including Windows 2000, Windows NT, and Window 9*x*. It is a multithreaded module. Using this module Apache will create a parent process and a child process. The child process creates all the threads that services the request. Also, this module now takes advantage of some Windows-only native function calls, which allows it to perform better than the earlier versions of Apache server on Windows platform.

Filtering I/O

Apache 2.0 now provides architecture for layered I/O. This means that one module's output can become another module's input. This filtering effect is very interesting. For example, the output produced by CGI scripts, which is processed by the `mod_cgi` module, can now be handed to the `mod_include` module responsible for SSIs. In other words, CGI scripts can produce output as SSI tags, which can be processed before the final output is sent to the Web browser. Many other applications of filtering I/O will be available in the future.

New CGI daemon

Because many of the MPM modules use threads, executing CGI scripts become cumbersome when a thread gets such a request. The `mod_cgi` module still works, but not optimally for threaded MPMs, so `mod_cgid` was added. The `mod_cgid` module creates a daemon process, which spawns CGI processes and interacts with threads more efficiently. Figure 1-1 shows how a CGI request for a script called `myscript.pl` is serviced.

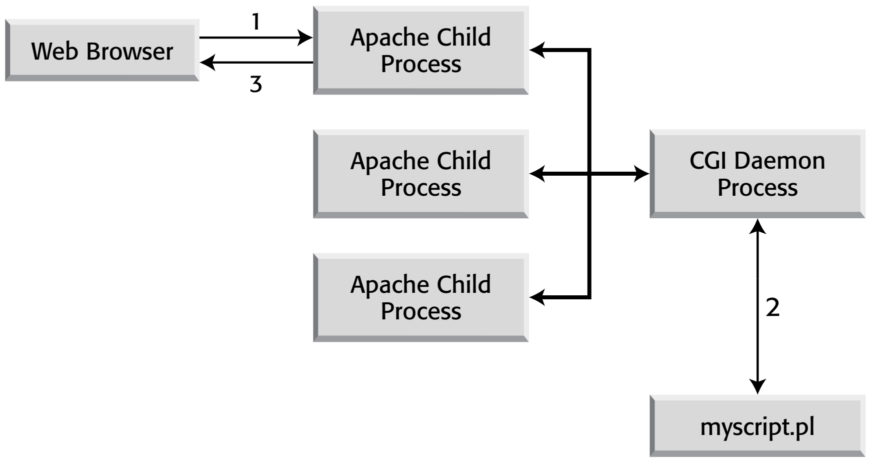

Figure 1-1: How the CGI daemon works with Apache child processes.

Here is how the CGI scripts are executed:

1. When the CGI request comes to a thread within a child process, it passes the request to the CGI daemon.

2. The CGI daemon spawns the CGI script and outputs the CGI script-generated data to the thread in the child process.
3. The thread returns the data to the Web browser.

When the main Apache server starts, it also starts the CGI daemon and establishes a socket connection. So, when a new child process is created, it inherits the socket connection and therefore does not have any need to create a connection to the CGI daemon for each request. The entire process improves CGI execution in the threaded environment.

Apache Portable Run-Time

In furtherance of the Apache Group's vision to create the most popular Web server in the world, it became clear that Apache's portability needed to be addressed in Apache 2.0. Prior to the current release, Apache had been dealing with portability internally, which made the code base less manageable. So, Apache Group introduced the Apache Portable Run-Time (APR). APR's purpose is to provide a single C interface to platform-specific functions so that code can be written once.

This enables Apache to run more natively on platforms such as Windows, BeOS, Amiga, and OS/2. Because of APR, many programs, such as ApacheBench, can run on these platforms.

Understanding the Apache License

Free software such as Apache, Perl (Practical Extraction and Reporting Language), and Linux (an *x*86-based Unix clone operating system) are getting a great deal of press because of Netscape's decision to make Netscape Communicator, one of the most popular Web browsers, available for free with its Mozilla project. Unfortunately, free software such as Apache, Perl, and Linux do not share the same licensing agreements, and so the media has created some confusion by associating these packages in the same licensing category.

All free software is intended to be, well, free for all. However, there are some legal restrictions that the individual software licenses enforce. For example, Linux, which is made free by GNU Public License (GPL), requires that any changes to Linux be made public. Apache, on the other hand, does not require that changes made to Apache be made public by anyone.

In short, think of Apache as free, copyrighted software published by the Apache Group. It is neither in the public domain nor is it shareware. Also note that Apache is not covered by GPL. The Apache Software License document is listed in Listing 1-1 for your convenience.

Listing 1-1: **Apache Software License**

```
/* ====================================================================
 * The Apache Software License, Version 1.1
 *
 * Copyright (c) 2000-2001 The Apache Software Foundation.  All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution,
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowledgment may appear in the software itself,
 *    if and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Apache" and "Apache Software Foundation" must
 *    not be used to endorse or promote products derived from this
 *    software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *    nor may "Apache" appear in their name, without prior written
 *    permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
```

Continued

Listing 1-1 *(continued)*

```
* ====================================================================
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation.  For more
* information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*
* Portions of this software are based upon public domain software
* originally written at the National Center for Supercomputing Applications,
* University of Illinois, Urbana-Champaign.
*/
```

✦ ✦ ✦