

Performance-Tuning Apache 2.0

on Dell PowerEdge Servers



Apache 2.0, the latest version of the popular Apache Web server, comes with an expanded feature set that makes the server more cost-effective, reliable, and scalable. This article describes the new features, explores best practices when configuring Apache 2.0, and highlights methods for optimizing Apache 2.0 on Dell™ PowerEdge™ servers running Intel® Xeon® processors.

BY SIMON JANDRESKI

From its humble beginnings as a series of patches to the National Center for Supercomputing Applications (NCSA) Web server (hence the name: “A patchy server”), the Apache HTTP server has become the most popular Web server in use today. More than 63 percent of all Internet sites use some version of Apache to serve content to their site visitors.¹ Developed by the Apache Software Foundation, this open source, cross-platform HTTP server is standards-compliant, secure, and stable. With the increasing corporate acceptance of another open source project, the Linux® operating system (OS), the Apache Web server is gaining additional exposure that is only likely to grow, because Apache is typically bundled with Linux. Both products are well supported by the open source community.

The new Apache 2.0 version offers enhancements to core functions and to modules. This article explains the most important changes and helps administrators choose among configuration and compilation options to optimize performance.

Introducing enhancements to Apache

Apache 2.0 introduces the Apache Portable Runtime (APR), adds new Multi-Processing Modules (MPMs), and

modifies existing modules. Additionally, Apache 2.0 brings core enhancements such as a new Apache application programming interface (API), IP version 6 (IPv6) support, a new build system, and simplified configuration. The result is a Web server that has an expanded set of features, making it more portable, reliable, and efficient.

Apache Portable Runtime

APR is the primary component that allows Apache functionality to be supported by nearly any platform. Once the code has been written, it can be compiled using an ANSI C compiler and then run on any platform. Moreover, APR uses native OS calls whenever possible. Consequently, when Apache runs on the Microsoft® Windows® OS, it looks like a native Windows program, but on the UNIX® OS, it appears as a UNIX program. Because using native calls involves no emulation or OS subsystem translation, code runs faster, increasing performance and easing portability.

Multi-Processing Modules

MPMs control how Apache operates when dealing with multiple requests and are responsible for opening sockets and mapping those requests to threads and processes.

¹ Source: August 2002 Netcraft survey, <http://www.netcraft.com/survey>.



The APR and MPMs in Apache 2.0 allow the HTTP server to run much faster and more reliably on non-UNIX platforms, while giving UNIX users the ability to select either the preforked or threaded MPMs that suit their operating systems and needs. The new MPMs in Apache 2.0 do not use the POSIX® (Portable Operating System Interface) subsystem built into non-UNIX operating systems.

Module modifications

The Apache server has a reputation for providing very flexible ways to add new features to the core system. Following this tradition, Apache 2.0 is now almost entirely module-based. A core system provides basic functionality, but almost every other component of the server is actually a loadable module. Several Apache modules have been completely rewritten and many new modules have been added. Through its use of modules, the Apache core can be expanded with a great variety of features.

Understanding process-based, threaded, and hybrid servers

Before knowing how to optimize Apache 2.0, administrators must understand MPMs and their relationship to processes and threads. Apache 1.3 is a process-based server: it starts a parent process that, in turn, forks identical copies of itself, called children (see Figure 1). Each child process can serve a request independently of the others. This approach provides scalability; if one of the child processes misbehaves, it can be terminated without affecting the rest of the server. However, this increased stability comes with a performance hit: each child occupies additional memory, and the OS spends extra time in context switching—that is, assigning processor time to each child. Additionally, processes are isolated from each other, making process communication and data sharing cumbersome.

A threaded server resembles a process-based server, with one difference: threads can share memory and data with other threads (see Figure 1). Because the threads are part of the same process, no context switching occurs, increasing performance. However, a misbehaved thread can overwrite and corrupt data and code belonging to other threads, bringing the whole server down.

Apache 2.0 is a hybrid server, an approach taken because both threaded and process-based servers have their advantages and disadvantages. When configured as a hybrid server, Apache 2.0 can spawn different processes, each one of them containing multiple threads (see Figure 1).

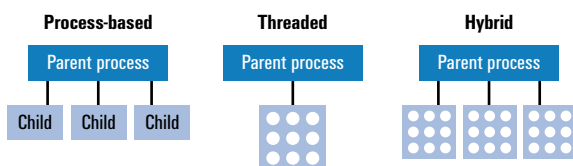


Figure 1. Process-based, threaded, and hybrid server models

Another way to increase Web site performance is to distribute the load among several servers by using a hardware load balancer.

Choosing among MPMs during configuration

The most important decision during the configuration process of Apache 2.0 is deciding which MPMs to run. When compiling from source code, administrators must choose MPMs during Apache configuration and compile them into the server. (For binary files, such as an RPM™ [Red Hat® Program Manager] file, the software distributor will have preselected and precompiled the MPM.) Because some MPMs on UNIX use threads and some do not, Apache will perform better if admin-

istrators choose an MPM at configuration time and compile it into Apache. Administrators specify the MPMs to be compiled into the server by including the argument `-with-mpm=NAME` when running the `configure` script, where `NAME` is the name of the desired MPM. After compiling the server, administrators can list all compiled modules, including MPMs, by issuing the command `httpd -l`. For the UNIX platform, the three primary MPMs provided with Apache 2.0 are `prefork`, `worker`, and `perchild`.

The prefork MPM

Administrators should select the prefork MPM to implement a process-based Web server. Although process-based servers may have somewhat slower performance, they offer stability and compatibility with modules that do not support threads. To improve performance, the server's parent process forks several children and keeps them available to answer requests. When a request comes in, the server assigns it to a child process. If no child process is free, the server creates a new child and adds it to the pool. However, the creation of a child causes a delay. When configuring this MPM, administrators can set limits on how many children are created at startup, the maximum number of children, and so forth. This MPM makes for a stable Apache server, but comes with a performance and memory penalty because the size of individual processes can be significant.

The worker MPM

To implement a hybrid server, administrators can choose the worker MPM, which offers better reliability, robustness, and scalability than the prefork MPM. The parent process creates a specified number of child processes at startup, each of them in turn containing several threads. Only one thread per child listens to the network, which simplifies the code and reduces interprocess cooperation, improving performance. Although this MPM is stable and performs better than

prefork, because it is threaded all of the modules used with it must be completely thread-safe. Most Apache 1.3 modules are not thread-safe, and because the worker MPM is not backward compatible, administrators using this MPM cannot also use Apache 1.3 modules with Apache 2.0. Compiling with the prefork MPM, however, does allow administrators to use Apache 1.3 modules.

The perchild MPM

In situations that require Apache processes to be run under different user IDs for security and performance reasons, administrators can compile a hybrid server using the perchild MPM. Internet service providers (ISPs) typically use this MPM for virtual hosting. When perchild starts, it creates a specific number of processes, each with a specific number of threads and a specific user ID. If the load on the server increases, instead of creating a new process, it uses one of the existing processes to create a new thread. This MPM is the most scalable but also the least reliable.

Tuning Apache 2.0 and the server hardware for better performance

Apache administrators must balance the Web server's performance with its scalability. Performance can suffer as the load on the server increases with more visitors and if the proper steps were not taken beforehand to accommodate greater workloads.

Although most Linux distributions include a binary installation of Apache, for performance and security reasons Apache should be compiled from source code using the `configure` script. This way, administrators can include only the options needed and tailor Apache for their particular environments, because unneeded features can slow performance. MPMs should be compiled into the server for optimum performance. When possible, threads should be used to increase performance.

To further decrease the response time of Apache, administrators can increase the OS settings that limit the number of processes and threads. In Linux 2.2.x kernels, this process requires increasing the value of `NR_TASKS`, which specifies the maximum number of simultaneous processes supported by the OS, and recompiling the kernel. In 2.4.x kernels, the value for `/proc/sys/kernel/threads-max`, which defines the maximum number of threads, should be increased. Using 2.4.x kernels is recommended because this kernel series supports `sendfile` system calls; these calls improve performance by eliminating separate read and send mechanics.

Hardware considerations

The underlying hardware significantly affects Web server performance. A Web server should never need to swap, because swapping increases the latency of each request. When running Apache on a Dell™ PowerEdge™ server, administrators should ensure the

server has enough RAM. They should also obtain fast CPUs, disks, and network cards.

Setting directives

When configuring Apache, administrators should set the `MaxClients` directive, which specifies the maximum number of server processes (simultaneously connected clients), to a reasonably high number; this action prevents the server from spawning so many children that it starts swapping. The `HostnameLookups` directive should be set to off, because Domain Name System (DNS) lookups add latency to every request. If DNS information is needed in the server log files for generating reports, administrators can postprocess the log file using the `logresolve` command. Ideally, this postprocessing should be performed somewhere other than the production Web machine.

If using `mod_include` and server-side includes (SSI), administrators should configure Apache so that it does not parse every HTML file, which consumes extra processing time. Instead, the server should parse only a specific file extension (the ones where SSI will be placed, something other than the most common file extension), such as `.shtml`.

Administrators can set `AllowOverride None` on all directories so that Apache does not check the per-directory configuration files, which are usually named `.htaccess`. Another way of increasing performance is to disable the `mod_status` module and ensure that `ExtendedStatus` is set to off. These settings collect statistics about the server, connections, and requests, which slows Apache performance.

Content negotiation, used to automatically provide content to a client based on browser capabilities and language preferences, should be avoided to achieve high performance. However, if content negotiation is necessary, administrators should use a type-map file, which provides better performance than using the `MultiViews` option.

To prevent disk access of frequently accessed files, administrators can map those files to memory using the `mod_file_cache` module. The list of files to map is specified as arguments to the global `MMapFile` directive.

By optimizing Apache 2.0

build and configuration

settings, administrators

can make more efficient

use of the hardware

on which Apache runs.

Another directive, `CacheFile`, takes a list of files and caches the file descriptors, saving time and resources.

Load testing

After configuring the Apache server with both compile-time and run-time optimizations, administrators should load test the Web site. The Apache distribution includes a load-testing tool called



ApacheBench (ab). The following command, for example, will request the `http://www.companyxyz.com` home page 5,000 times using 50 concurrent clients at any given time:

```
usr/local/apache2/bin/ab -n 5000 -c 50 \
http://www.companyxyz.com/
```

Completing the test may take some time, but the results are easy to understand and can be used to fine-tune performance. Administrators can also use the Flood load-testing utility, available from the Apache Software Foundation.

Load balancing

Another way to increase Web site performance is to distribute the load among several servers by using a hardware load balancer, such as the Dell PowerEdge Load Balancing Server-BIG-IP® Powered. This device distributes the network and HTTP traffic across several servers, making a Web site look as though it were being served by a single server to outside clients.

Administrators can also use a software load balancer, such as the Apache server itself. Another option is to provide one or several separate image servers that serve large image files and other static content. A kernel-based HTTP accelerator, such as TUX, can be configured with Apache to further speed the serving of static pages.

Proxy support

A forward proxy, or proxy, accepts a request for a Web site from one or more internal clients, contacts a remote Web server, and returns the responses. A reverse proxy can be used to load balance Web sites. Placed in front of other servers, a reverse proxy provides a unified front to back-end Web servers. For example, if a client requests `http://www.companyxyz.com`, the request goes to a reverse proxy server answering for that URL. The reverse proxy server then contacts one of the back-end servers to serve the request to the client. The client communicates only

Proxy support for Apache has been completely rewritten to comply with the most recent HTTP standard.

with the reverse proxy server. Proxy support for Apache has been completely rewritten to comply with the most recent HTTP standard. To enable proxy support, administrators pass the `-with-proxy` argument to the `configure` script during the compilation. The `proxy_module` should be listed in the Apache configuration file:

```
LoadModule proxy_module modules/mod_proxy.so
```

Two directives can be used with this module: `ProxyPass` and `ProxyPassReverse`. As shown in Figure 2, the Dell PowerEdge 2650 server can be configured to act as a reverse proxy and redirect all dynamic content and Secure Sockets Layer (SSL) traffic to Dell PowerEdge 6650 back-end servers while allowing the PowerEdge 2650 to serve all the static pages and images. Additionally, administrators can cache some content on the PowerEdge 2650 by enabling caching support in Apache. To do that, Apache should be built using the `configure` script with the `-enable-cache`, `-enable-disk-cache`, and `-enable-mem-cache` arguments and then loading and configuring the appropriate modules. The fastest way to serve content is to use caching instead of repeatedly serving static pages.

Future performance gains

The Apache 2.0 Web server is a powerful and popular HTTP server with numerous configuration options and many applications in the field. By optimizing Apache 2.0 build and configuration settings, administrators can make more efficient use of the hardware on which Apache runs. Experimenting with different configuration options and using the plentiful documentation available for Apache can lead to further performance gains. 🤖

Simon Jandreski (simon_jandreski@dell.com) is a network engineer and advisor in the Latin America Enterprise Expert Center at Dell. His interests include UNIX and Linux installation, management, and performance tuning. Additionally, he troubleshoots, optimizes, and administers Apache Web servers. Simon has a degree in Computer Science and RHCE, SCNA, SCNA, MCSE, MCSE+I, MCSA, CCNA, and A+ certifications.

FOR MORE INFORMATION

Apache HTTP server project: <http://httpd.apache.org>
 ApacheBench (ab): <http://httpd.apache.org/docs/programs/ab.html>
 Dell PowerEdge servers: <http://www.dell.com/servers>

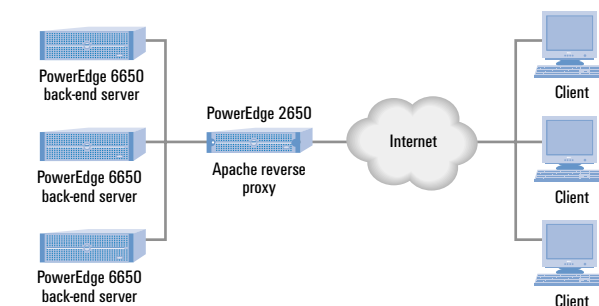


Figure 2. Dell PowerEdge 2650 server configured to act as a reverse proxy