Introduction to web server

A **web server** can be referred to as either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet.

The most common use of Web servers is to host Web sites but there are other uses like data storage or for running enterprise applications.

Overview

The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and JavaScripts.

A client, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary memory, but this is not necessarily the case and depends on how the web server is implemented.

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Many generic web servers also support server-side scripting, e.g., Apache HTTP Server and PHP. This means that the behaviour of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to create HTML documents "on-the-fly" as opposed to returning fixed documents. This is referred to as dynamic and static content respectively. The former is primarily used for retrieving and/or modifying information from databases. The latter is, however, typically much faster and more easily cached.

Web servers are not always used for serving the world wide web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring and/or administrating the device in question. This usually means that

no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

History of web servers

The world's first web server.

In 1989 Tim Berners-Lee proposed to his employer CERN (European Organization for Nuclear Research) a new project, which had the goal of easing the exchange of information between scientists by using a hypertext system. As a result of the implementation of this project, in 1990 Berners-Lee wrote two programs:

- a browser called WorldWideWeb;
- the world's first web server, later known as CERN httpd, which ran on NeXTSTEP.

Between 1991 and 1994 the simplicity and effectiveness of early technologies used to surf and exchange data through the World Wide Web helped to port them to many different operating systems and spread their use among lots of different social groups of people, first in scientific organizations, then in universities and finally in industry.

In 1994 Tim Berners-Lee decided to constitute the World Wide Web Consortium (W3C) to regulate the further development of the many technologies involved (HTTP, HTML, etc.) through a standardization process.

Path translation

Web servers are able to map the path component of a Uniform Resource Locator (**URL**) into:

- a local file system resource (for static requests);
- an internal or external program name (for dynamic requests).

For a *static request* the URL path specified by the client is relative to the Web server's root directory.

Consider the following URL as it would be requested by a client:

http://www.example.com/path/file.html

The client's user agent will translate it into a connection to www.example.com with the following HTTP 1.1 request:

GET /path/file.html HTTP/1.1
Host: www.example.com

The Web server on www.example.com will append the given path to the path of its root directory. On an Apache server, this is commonly /home/www (On Unix machines, usually /var/www). The result is the local file system resource:

/home/www/**path/file.html**

The Web server then reads the file, if it exists and sends a response to the client's Web browser. The response will describe the content of the file and contain the file itself or an error message will return saying that the file does not exist or is unavailable.

Load limits

A Web server (program) has defined load limits, because it can handle only a limited number of concurrent client connections (usually between 2 and 80,000, by default between 500 and 1,000) per IP address (and TCP port) and it can serve only a certain maximum number of requests per second depending on:

- its own settings;

- the HTTP request type;
- content origin (static or dynamic);
- the fact that the served content is or is not cached;
- the hardware and software limitations of the OS where it is working;

When a Web server is near to or over its limits, it becomes unresponsive.

Kernel-mode and user-mode Web servers

A Web server can be either implemented into the OSkernel, or in user space (like other regular applications).

An in-kernel Web server (like TUX on GNU/Linux or Microsoft IIS on Windows) will usually work faster, because, as part of the system, it can directly use all the hardware resources it needs, such as non-paged memory, CPU time-slices, network adapters, or buffers.

Web servers that run in user-mode have to ask the system the permission to use more memory or more CPU resources. Not only do these requests to the kernel take time, but they are not always satisfied because the system reserves resources for its own usage and has the responsibility to share hardware resources with all the other running applications.

Also, applications cannot access the system's internal buffers, which causes useless buffer copies that create another handicap for user-mode web servers. As a consequence, the only way for a user-mode web server to match kernel-mode performance is to raise the quality of its code to much higher standards, similar to that of the code used in web servers that run in the kernel. This is a significant issue under Windows, where the user-mode overhead is about six times greater than that under Linux.[2]

Overload causes

At any time web servers can be overloaded because of:

- **Too much legitimate web traffic.** Thousands or even millions of clients connecting to the web site in a short interval, e.g., Slashdot effect;
- **Distributed Denial of Service** attacks;

- **Computer worms** that sometimes cause abnormal traffic because of millions of infected computers (not coordinated among them);
- **XSS viruses** can cause high traffic because of millions of infected browsers and/or Web servers;
- **Internet bots.** Traffic not filtered/limited on large web sites with very few resources (bandwidth, etc.);
- **Internet (network) slowdowns**, so that client requests are served more slowly and the number of connections increases so much that server limits are reached;
- **Web servers (computers) partial unavailability.** This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back-end (e.g., database) failures, etc.; in these cases the remaining web servers get too much traffic and become overloaded.

Overload symptoms

The symptoms of an overloaded Web server are:

- requests are served with (possibly long) delays (from 1 second to a few hundred seconds);
- 500, 502, 503, 504 HTTP errors are returned to clients (sometimes also unrelated 404 error or even 408 error may be returned);
- TCP connections are refused or reset (interrupted) before any content is sent to clients;
- in very rare cases, only partial contents are sent (but this behavior may well be considered a bug, even if it usually depends on unavailable system resources).

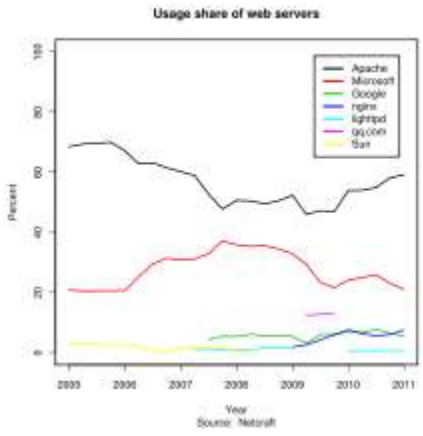
Anti-overload techniques

To partially overcome above load limits and to prevent overload, most popular Web sites use common techniques like:

- **managing network traffic**, by using:
    - **Firewalls** to block unwanted traffic coming from bad IP sources or having bad patterns;

    - **HTTP traffic managers** to drop, redirect or rewrite requests having bad HTTP patterns;
    - **Bandwidth management** and **traffic shaping**, in order to smooth down peaks in network usage;
- deploying **Web cache** techniques;
- using different domain names to serve different (static and dynamic) content by separate Web servers, i.e.:
    - http://images.example.com
    - http://www.example.com
- using different domain names and/or computers to separate big files from small and medium sized files; the idea is to be able to fully cache small and medium sized files and to efficiently serve big or huge (over 10 - 1000 MB) files by using different settings;
- using many Web servers (programs) per computer, each one bound to its own network card and IP address;
- using many Web servers (computers) that are grouped together so that they act or are seen as one big Web server (see also **Load balancer**);
- adding more hardware resources (i.e. RAM, disks) to each computer;
- tuning OS parameters for hardware capabilities and usage;
- using more efficient computer programs for Web servers, etc.;
- using other workarounds, especially if dynamic content is involved.

Market structure



Market share of major Web servers

For more details on HTTP server programs, see Category:Web server software.

Below is the most recent statistics of the market share of the top web servers on the internet by Netcraftsurvey in November 2010

Vendor	Product	Web Sites Hosted	Percent
Apache	Apache	148,085,963	59.36%
Microsoft	IIS	56,637,980	22.70%
Igor Sysoev	nginx	15,058,114	6.04%
Google	GWS	14,827,157	5.94%
lighttpd	lighttpd	2,070,300	0.83%