THE ANNALS OF “DUNAREA DE JOS” UNIVERSITY OF GALATI
FASCICLE III, 2009, Vol.32, No.2, ISSN 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

SOLUTION FOR AN IMPROVED WEB SERVER

George PECHERLE, Cornelia GYORODI*, Ovidiu BUKSA***, Stefan MOGYOROSI****

Professor PhD Eng.,PhD Assistant, ***Student*
Department of Computer Science, Faculty of Electrical Engineering and Information Technology, University of Oradea,
Str. Universitatii 1, 410087, Oradea, Romania
E-mail: cgyorodi@uoradea.ro, gpecherle@uoradea.ro, buksa_ovidiu@yahoo.com

Abstract: We want to present a solution with maximum performance from a web server, in terms of services that the server provides. We do not always know what tools to use or how to configure what we have in order to get what we need. Keeping the Internet-related services you provide in working condition can sometimes be a real challenge. And with the increasing demand in Internet services, we need to come up with solutions to problems that occur every day.

Keywords: web server, client, PHP, MySQL, ASP.NET, web application, Apache

1. INTRODUCTION

A server, in general, has one main purpose, to serve its clients to the best of its abilities. This depends very much on the software configuration of that server. The configuration has to be very well done in order for the server to accomplish its purpose. Without this, we cannot talk about having a reliable server that does what we want it to do.

A webserver has one main purpose: to host one or several websites and manage the clients' requests for each of the hosted sites.

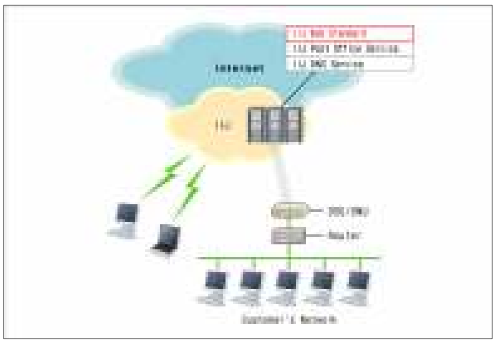
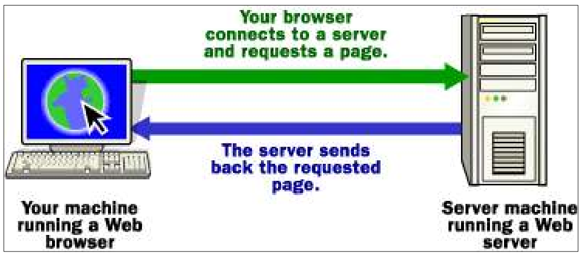


Fig. 1. Webserver, Internet and home or business client interconnectivity [6]

There are several protocols behind a webserver, but all of them can be traced to a single, main, concept: HTTP (HyperText Transfer Protocol). The process seems to be very complicated, but it is in fact very easy to understand.


Your browser connects to a server and requests a page.
The server sends back the requested page.
Your machine running a Web browser
Server machine running a Web server
Fig. 2. Client-server communication [5]

The address you type in your browser has several protocols embedded into it [1]:

http://www.google.com/index.html

- http: protocol for communication with the server;

- www.google.com: name assigned to an IP address of a machine, called name-server. This is mainly used for easing the way a user accesses a website. It is easier to remember a name instead of a combination of numbers.

- index.html: the name of the file contained in the server-side folder assigned for the name. This file can be any of the files in that folder, file that usually represents a page of the website. The file can also be contained in a subfolder of the main folder of the website

These three form together the URL (**U**niform **R**esource **L**ocator).

In general, all of the machines on the Internet can be categorized as two types: servers and clients. Those machines that provide services (like Web servers or FTP servers) to other machines are **servers**. And the machines that are used to connect to those services are **clients**. When you connect to Google at www.google.com to read a page, Google is providing a machine (probably a cluster of very large machines), for use on the Internet, to service your request. Google is providing a server. Your machine, on the other hand, is probably providing no services to anyone else on the Internet. Therefore, it is a user machine, also known as a client. It is possible and common for a machine to be both a server and a client, but for our purposes here you can think of most machines as one or the other [4].

A server machine may provide one or more services on the Internet. For example, a server machine might have software running on it that allows it to act as a Web server, an e-mail server and an FTP (**F**ile **T**ransfer **P**rotocol) server. Clients that come to a server machine do so with a specific intent, so clients direct their requests to a specific software server running on the overall server machine. For example, if you are running a Web browser on your machine, it will most likely want to talk to the Web server on the server machine [4].

Web servers present a virtual hierarchy of information to browser programs. This is the *path* part of a URL. Although the top of the virtual hierarchy is rooted in the server-side file system hierarchy, the virtual hierarchy does not need be the same as the server-side file system hierarchy. Servers can usually be configured to map parts of the virtual hierarchy to different areas of the server-side file system, or to redirect the clients' requests to other servers. The URL may also identify a program that returns information in the form of a document. Information in the URL after the part that identifies the program is passed to the program as data that may be evaluated and used to determine the content of the document generated [5]. This is the case if you have a dynamic website written in a web-development languages (eg. Web-application written in PHP).

1.1. What do we need in order to have a working webserver?

First of all, you need a computer with a server-dedicated operating system, because server-dedicated operating systems have server software engines designed to optimize system resources consumption. The second thing you need is webserver software, like Apache or software that would know to answer a client-browser request for a particular HTML file. If you host on that server webpages that include server-side scripting, like ASP.Net or PHP, your webserver software needs the possibility of adding additional modules for this kind of pages, because the client-browser does not need to receive a code that it cannot interpret and display to the user. The client-browser will always receive a HTML code, but that code is generated by the interpreter on the server.

For web-applications that manage lots of data, it is recommended that you use a database. The database server is an independent server, from the software point of view. The only way of communication between the webserver and the database server is through the web application.

1.2. Where can I find this kind of software?

All you have to do is make a search on the Internet, because you can find plenty. The hard part is choosing the one that fits best, depending on your level of experience. If you have very little or no experience in this matter, we recommend to try a software that is user-friendly. To be more specific, you should get software that you can configure through a web browser or at least software that is easier for you to configure. Again, this is hard, because you need to try a wide variety of software in order to find the best one for you.

1.3 Comparison between 2 webserver software

We experimented with 2 programs that do what we previously presented. These are Apache, developed as open-source software by The Apache Software Foundation, and AbyssWebServer, developed by Aprelium. We wanted software that is easy to use and configure.

1.3.1. Installation

Both programs are very easy to install on any version of Windows. All you need to do is download the programs from the developer websites and install them.

1.3.2. Startup

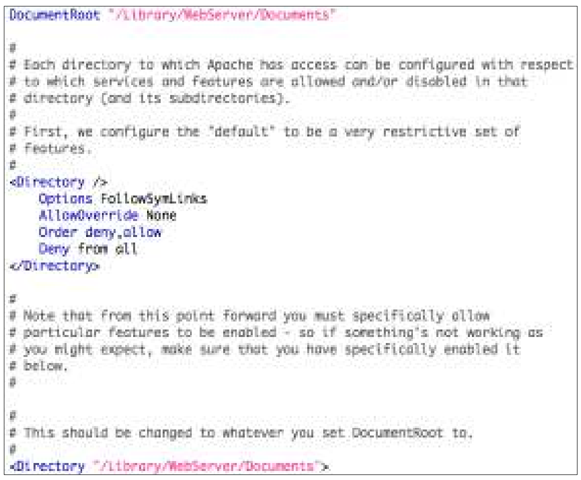
The first time you will run the software you installed, you will notice that for Apache (contained in the Wamp project), you have just one click to do in order to start the server. For Abyss, you are asked right after the installation is complete to start the program and a shortcut for the software is placed on your desktop (optional).

1.3.3. Configuration

The first thing you will notice when trying to configure Apache is that you do not have a menu or an interface that would help you configure the software. You are told that you need to edit a configuration file. In Abyss, you are presented a web interface that is very user friendly and help is just one click away. In Apache, help is all over the Internet and finding a help page that you can use easily is difficult. This is because Apache can be configured in more than one way.

1.3.4. Performance comparison

There is a significant difference between the mentioned programs. This is mainly on the part handling the on-run configuration changes. While Abyss detects and requires the webserver administrator to restart the application before applying the changes made, Apache does not signal this, thus requiring the administrator to do this process manually.
Another aspect of the performance of these 2 programs is their method of reading their individual configuration files. Abyss is saving its configuration files in the form of a XML file. Apache has a configuration file in a form of a variable-XML file, thus more complex and, in our opinion, slower to read and find the required information. Apache can also save its configuration in a multi-file form. Accessing a single XML file is significantly faster than multiple files.
Running an efficient, fast webserver means that it has to be very easy to install, configure and maintain without having to work very hard and for a long period of time.



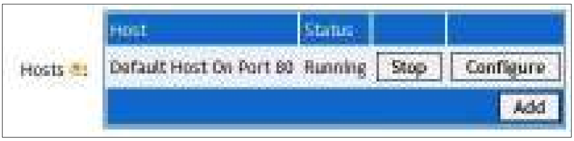
```
DocumentRoot "/Library/WebServer/Documents"

#
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/Library/WebServer/Documents">
```

Fig. 3. Configuration file for Apache [7]



Host	Status		
Default Host On Port 80	Running	Stop	Configure

Fig. 4. Graphical interface for AbyssWebServer configuration [3]

2. GENERAL ADVICE AND RECOMMENDED PACKAGES

The first thing you need to do is ask yourself what kind of pages will you host on your webserver. This is because resource consumption needs to be as low as possible. Webservers need to be able to handle thousands of simultaneous connections from all over the world. This is done by adding only the necessary features in your webserver. For example, if you are not going to host websites with a PHP application, you should not install the PHP feature, because it will only slow down your server and increase your server's response time, which is very important to be as little as possible.

Although not recommended because of some security issues, your database server application can be installed on the same machine. This will increase your response time between the database and the web application, but will leave your server vulnerable to security attacks. Of course, steps can be taken to prevent this from happening.

We recommend the following software for a optimal webserver, no matter the operating system installed on the machine:

- Abyss Web Server, with ASP.NET support[3]
- Preconfigured PHP package available on Abyss's developer website [3]
- MySQL server [8]

An example of this kind of implementation, using the previous stated packages is very simple.

For this, we are using AbyssWebServer version 2.5, with the preconfigured Aprelium PHP version 5.2.9-1, with a modified php.ini file (configuration file for the PHP script interpreter). The only modification made to this file is the memory within the PHP scripts can run. For development of PHP-based web applications, the memory we recommend is 128 MB, the maximum allowed. For public webserver, it is recommended to leave the php.ini file unmodified [9]. The based directory for your development website is not the default value, htdocs within the Abyss installation, but another directory on your local hard disk. This is mainly for easing the development process. The free version of Abyss only allows one host within the server. The commercial version of the program allows multiple websites to be hosted on the same server, each of them with their individual web address. This is called virtual hosting. Because we are using SQL-based databases, the MySQL server version 5.1 is needed. Within the MySQL server configuration, we recommend the default value for the database(s) files to be left in the MySQL installation folder. It is highly recommended that you set a password for your MySQL server because some, but not all, functions available in PHP require a basic security for communication with the database used for the application [9].

AbyssWebServer free edition has some advantages because it allows a single site to be hosted on the server, allowing securing access to certain folders within your web-application [3].

3. CONCLUSION

Making your own webhosting service can be very easy, but can also be a learning experience in what means to provide and maintain Internet-related services for you or others to use. The problems can occur every step of the way, but continuous maintenance can prevent this.

The example presented here is not intended to be a bad advertisement or a good one for the named applications. These applications were used as an example for the purpose of presenting the upsides and downsides of having a personal webserver.

4. REFERENCES

[1] Web Server information (Wikipedia website) http://en.wikipedia.org/wiki/Web_server
[2] Webopedia Web Server Description http://www.webopedia.com/term/w/web_server.html
[3] Aprelium Abyss Web Server Official Website http://www.aprelium.com/
[4] HowStuffWorks – how web servers work http://computer.howstuffworks.com/web-server.htm
[5] Web Server Concepts http://andrew-ford.com/stw/node124.html
[6] Internet Initiative Japan http://www.iij.ad.jp/en/index.html
[7] Apache Web Server Installation
[8] MySQL Query Analyzer – Improving SQL Query Performance http://www.mysql.com
[9] PHP: Hypertext Processor http://php.net/index.php