Zend Platform™

White Paper:

Zend Download Server

By **Zend Technologies, Inc.**

May 2007

Zend
The *php* Company

Introduction

You are running a successful web site. You have probably found yourself in situations where your Apache web server was saturated by large number of clients. One of the most common reasons for this behavior is that while Apache is an excellent server for running your PHP applications, it doesn't scale when serving long downloads.

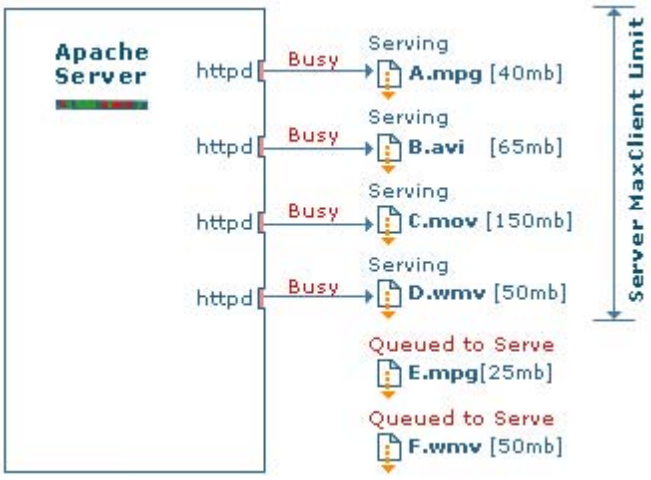
Long downloads can be caused by either transfers of large files, such as movies, or by users who have slow Internet connectivity. 60% of the world's Internet community is still using dial-up connections, and their speed is 56Kbps at best. Therefore, it is essential to serve large file downloads in an efficient way.

How do large file downloads affect your server's performance?

Before we answer this question, it is important to explain how the Apache web server works.

In order to serve HTTP requests, the Apache web server spawns many processes. These processes are usually called 'httpd' and each process can serve one HTTP request at a time. When a user requests a web page or requests a file download he generates an HTTP request. This request is handled by one of the Apache httpd processes from start to end. When there are too many incoming requests to the Apache web server, it spawns more httpd processes in order to serve these requests. The maximum number of concurrent httpd processes is configurable though limited by the amount of memory on the server. When the server reaches this maximum, it can't serve any more incoming HTTP requests. Users either wait a long time for the page to be loaded, or get messages like "gateway timeout" or "Server busy" if the server's incoming queue is full.

Saturated Apache Server



The difference between web page and file download serving

Serving HTML web pages is usually a short operation, with a few seconds of processing time. Therefore, under normal usage conditions it is hard to reach the point where all of the httpd clients are busy serving HTTP requests, which results in user denial or time outs.

With HTTP file downloads the situation is quite different. A file download can be a long operation that may take minutes to several hours depending on the file size and the end user's connection speed and quality.

During this time, there is a persistent connection between the httpd process that initially received the download request and the end user's download program (or web browser) until the file download is completed, and it is blocked from serving other HTTP requests. The remaining available httpd clients will serve all other incoming requests.

Your web server gets saturated

A typical server runs up to a maximum of a few hundred httpd processes. As the number of download requests increase and finally exceeds this number, the Apache server can't accept any more connections, and users who are trying to connect to the site will be rejected or experience long wait periods until they receive a response.

From the hardware point of view, it is hardly utilized even though the Apache web server reached its limits. File serving is mainly I/O, therefore the CPU and memory is mostly idle.

What is the common solution?

Companies for whom file downloads like movies, software products, and large audio files are important, service outage or poor performance are not an option. Therefore, in order to overcome this limitation, more hardware is purchased to maintain a desired quality of service, but because of the way Apache handles downloads, it ends up sitting idle. .

The Zend Solution

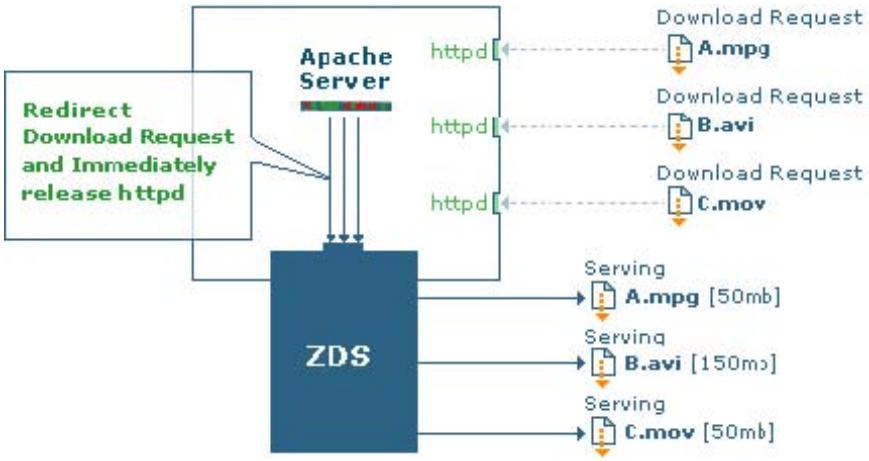
Zend Download Server (ZDS) is a PHP plug-in that efficiently serves large downloads over HTTP protocol. ZDS takes over downloads and leaves the Apache server to handle online requests. When a file download request arrives to the Apache server, ZDS immediately takes over this request, frees up the httpd process and handles the download until it is completed. This approach allows the Apache server to continue to handle PHP requests very efficiently, and ZDS to efficiently handle file downloads.

The key advantages of using the Zend Download Server are:

- Very efficient handling of large downloads
- Preventing clients with slow connections from saturating your server
- Full support for sending files from PHP scripts, including files outside of the document root
- Maximize utilization of your server's resources

The ROI (Return On Investment) of the ZDS is extremely fast, as it will quickly save you huge amounts of hosting and hardware dollars As it allows you to maximize your machine's CPU and network throughput. ZDS makes sure that when you need to buy a new server, it is because the platform you are currently using reached its limits, and can't be pushed further.

Download With ZDS



Zend Download Server Operation Modes

The Zend Download Server supports two modes of operation (both can be used together or separately according to your needs)

Transparent Mode

In your web server's configuration file you map the file types (*.mpeg, *.avi) you want to be sent through ZDS to PHP, and whenever there is a request for a file of this type, ZDS will automatically handle the request. It can be easily done by adding the following lines to your httpd.conf file: *AddType application/x-httpd-php .mpeg*

And you should also add it to */usr/local/Zend/etc/zend_mime_types.ini* in order for the file to be handled by ZDS properly.

For example, if you wish to serve video mpeg files via ZDS then you should add the line "*video/mpeg mpeg*" to *zend_mime_types.ini.*

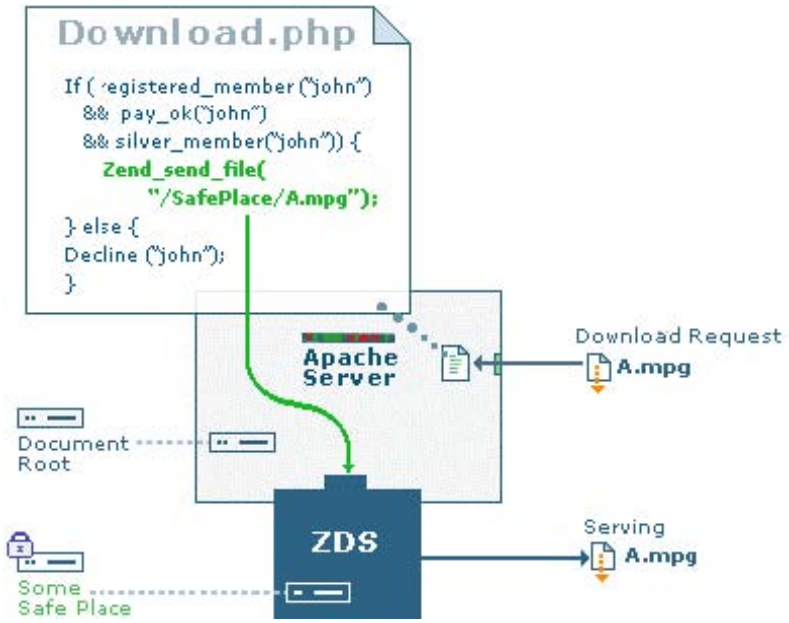
The MIME types file format is:

```
application/octet-stream tgz exe
application/z-gzip       gz
video/mpeg               mpeg
```

Secure Programmable Mode

In programmable mode you will initiate a download from within your PHP script. ZDS provides a PHP function called *zend_send_file(filename)* in order to achieve this functionality. Calling *zend_send_file()* will start the file download immediately and will terminate your PHP script's execution, freeing up the Apache process to handle the next incoming request.

Secure Programmable Mode



i It allows you to serve files that aren't under your web server's document root, which is a very important security feature. For example, many sites choose to store files that belong to members' areas out of the document root to protect them from unauthorized access. In order to achieve this functionality all you need to do is to call this simple API.

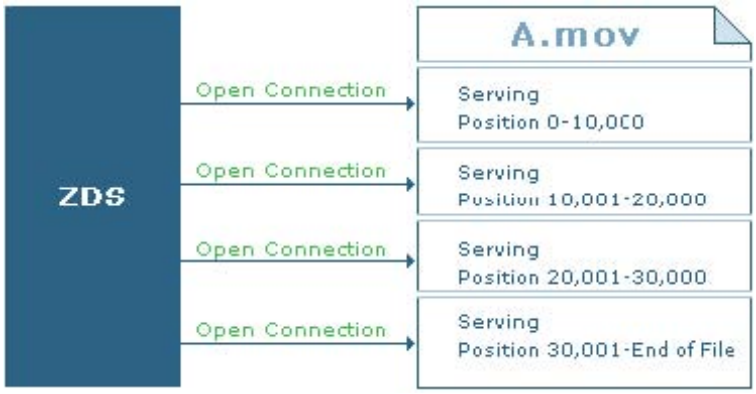
Achieving the same functionality without ZDS using fpassthru() requires much more programming and won't give you the same results, because you will still hit the same scalability issues of the Apache server.

ii The API allows you to run logic such as access restriction checks before the download is started. For example, many sites check the membership status of the user before they authorize a file download. This API call allows the PHP developer to perform these checks prior to authorizing the download.

Byte Range Control

In addition, ZDS supports byte-range download control. Some of the download accelerators can concurrently download multiple parts of the same file in order to accelerate the download process.

Byte Range File Download



This operation takes more bandwidth and horsepower from the server because each user has multiple connections opened to the same file. For some sites pushing the maximum possible data down the lines it is an important feature, as they want to give their users the best response time. On the other hand for sites that have less bandwidth or memory, download accelerators can be problematic. ZDS provides flexibility allowing the Webmaster to enable/ disable them for full control.

Conclusion

- The Zend Download Server scales up your Apache server by maximizing the number of concurrent downloads it can handle
- It optimizes the utilization of your hardware and bandwidth resources
- It protects your valuable content by enabling to store it outside of the server's document root
- It enables you to validate the user and perform your business logic prior to serving the download request