;login:
THE MAGAZINE OF USENIX & SAGE
April 2003 • volume 28 • number 2

inside:

SYSADMIN
Pierzchala: Compressing Web Output Using mod_gzip and Apache

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

compressing Web output using mod_gzip and Apache

by Stephen Pierzchala

Stephen Pierzchala is senior diagnostic analyst for Keynote Systems in San Mateo, CA. He spends his time reminding Web developers about the need for Web performance.

stephen@pierzchala.com

Web-page compression is not a new technology but has only recently gained higher recognition in the minds of IT administrators and managers because of the rapid return on investment it generates. Compression extensions exist for most of the major Web server platforms, but in this article I will focus on the Apache and mod_gzip solution.

The idea behind GZIP-encoding documents is very straightforward. Take a file that is to be transmitted to a Web client and send a compressed version of the data rather than the raw file as it exists on the file system. Depending on the size of the file, the compressed version can run anywhere from 20% to 50% of the original file size.

In Apache, this can be achieved using a couple of different methods. Content negotiation, which requires that two separate sets of HTML files be generated – one for clients that can handle GZIP-encoding and one for those that can't – is one method. The problem with this solution should be readily apparent: There is no provision in this methodology for GZIP-encoding dynamically generated pages.

The more graceful solution for administrators who want to add GZIP-encoding to Apache is the use of mod_gzip. I consider it one of the overlooked gems for designing a high-performance Web server. Using this module, configured file types – based on file extension or MIME type – will be compressed using GZIP-encoding after they have been processed by all of Apache's other modules, and before they are sent to the client. The compressed data that is generated reduces the number of bytes transferred to the client, without any loss in the structure or content of the original, uncompressed document.

mod_gzip can be compiled into Apache as either a static or dynamic module; I have chosen to compile it as a dynamic module in my own server.[1] The advantage of using mod_gzip is that this method does not require anything to be done on the client side to make it work. All current browsers – e.g., Mozilla, Opera, Internet Explorer – understand and process GZIP-encoded text content.

1. The servers used for this article were from the Apache 1.3.x family.

On the server side, all the server or site administrator has to do is compile the module, edit the appropriate configuration directives that were added to the httpd.conf file, enable the module in the httpd.conf file, and restart the server. In less than 10 minutes, you can be serving static and dynamic content using GZIP-encoding without the need to maintain multiple code bases for clients that can or cannot accept GZIP-encoded documents.

When a request is received from a client, Apache determines if mod_gzip should be invoked by noting if the Accept-Encoding: gzip HTTP request header has been sent by the client. If the client sends the header, mod_gzip will automatically compress the output of all configured file types when sending them to the client.

This client header announces to Apache that the client will understand files that have been GZIP-encoded. mod_gzip then processes the outgoing content and includes the following server response headers:

```
Content-Type: text/html
Content-Encoding: gzip
```

These server response headers announce that the content returned from the server is GZIP-encoded, but that when the content is expanded by the client application, it should be treated as a standard HTML file. Not only is this successful for static HTML files, but it can be applied to pages that contain dynamic elements, such as those produced by Server Side Includes (SSI), PHP,[2] and other dynamic page-generation methods. You can also use it to compress your Cascading Style Sheets (CSS) and plaintext files. My httpd.conf file sets the following configuration for the file types handled by mod_gzip:

```
mod_gzip_item_exclude     file     \.js$
mod_gzip_item_exclude     mime     ^application/.*$
mod_gzip_item_exclude     mime     ^image/.*$
mod_gzip_item_include     file     \.html$
mod_gzip_item_include     file     \.shtml$
mod_gzip_item_include     file     \.php$
mod_gzip_item_include     file     \.txt$
mod_gzip_item_include     mime     ^text/.*$
```

I have had limited success compressing other file formats, mainly because Microsoft's Internet Explorer appears to examine the "Content-Type" header message before it examines the "Content-Encoding" header message. So, say you configure your server to GZIP-encode PDF files using the following mod_gzip directives:

```
mod_gzip_item_include     file     \.pdf$
mod_gzip_item_include     mime     ^application/pdf$
```

When downloaded by Mozilla and Opera, the PDF files are immediately decoded and passed to the appropriate helper application. These browsers know to decode all GZIP-encoded content before passing it along to the appropriate helper application.

However, Internet Explorer simply passes the GZIP-encoded content directly to the PDF reader without first decoding it. A quick rummage through newsgroup archives turned up evidence that this "feature" has been in Internet Explorer since at least 1997. I chalk it up to the lingering integration of browser and operating system through the Component Object Model (COM). This has a potentially detrimental impact on the Web community as a whole range of additional file types could be compressed if this bug was fixed.

How beneficial is sending GZIP-encoded content? In some simple tests I ran on my Web server using WGET, GZIP-encoded documents showed that even on a small Web server there is the potential to produce a substantial savings in bandwidth usage. For *http://www.pierzchala.com/bio.html*, uncompressed file size was 3122 bytes, compressed was 1578 bytes. And for *http://www.pierzchala.com/compress/homepage2.html*, uncompressed file size was 56279 bytes, compressed was 16286 bytes.

Server administrators may be concerned that mod_gzip will place a heavy burden on their systems as files are compressed on the fly. I argue against that, pointing out that this does not seem to concern the administrators of Slashdot (*http://slashdot.org/*), one of the busiest Web servers on the Internet, who use mod_gzip in their very high-traffic environment.

The mod_gzip project page is located at SourceForge: *http://sourceforge.net/projects/mod-gzip/*.

2. PHP can also be compressed using the integration with the native ZLIB compression libraries. This integration can be built in at compile time and activated through the php.ini file.