C R E S C E N D O N E T W O R K S

Content Compression With The Maestro Application Delivery Platform

February 2005

CN_WPT_230_0205

Telephone Contact:

International
Tel. +972.3.634.6120

In the US
Tel. +1.866.830.0400

Web Address:

www.crescendonetworks.com

Email:
CNInfo@crescendonetworks.com

Crescendo Network's Maestro Application Delivery Platform

Crescendo Networks provides high-performance application delivery and acceleration for enterprises and web sites. The unique design of Crescendo's Maestro Application Delivery Platform brings instant performance relief to overburdened data centers today - and Maestro is the only solution built to be an integral component in emerging data center architectures.

Crescendo's Maestro Application Delivery Platform is a multi-gigabit, hardware-based platform that is capable of offloading task-intensive functions from servers in a web application, optimizing that application and allowing the servers that host it to scale significantly. Maestro is an appliance that front-ends the servers, intercepting and processing all user requests destined for them. By performing this functionality, Maestro can provide various optimization services for the servers in the application, massively improving server performance while reducing user response time and consumed bandwidth.

Content Types in Web Applications

Today's web applications employ a number of different types of content. When a client (i.e. browser) connects to a web application, it can request various types of objects from the server; everything ranging from simple text-based files to images and even application executables. Depending on the richness of the application and its user interface, a variety of these file types can be used.

The problem with some of these file types is that they're often unnecessarily large. For example, some of the text-based files commonly used in many web applications can grow to a large size depending on the embedded code. HTML, for instance, is a rich language often requiring multiple arguments for displaying a single piece of text.

Obviously, the larger the size of the object, the longer it takes a client to download it. Since web pages often include many different objects, these types of files can lead to an increase in overall page download time and, as a result, a decrease in the quality of the user experience. This is true regardless of the access link available to a user (dialup, broadband, or more). However, the problem is compounded when the access bandwidth for the user is limited (e.g. dialup); depending on the perceived download time, the application may seem unusable to these clients.

Let's also not forget that larger objects consume more bandwidth. The file sizes used by an application directly affect the overall outbound bandwidth used by that application, for all its users. This can become both a technical and financial problem for web sites and web applications that have limited outbound bandwidth resources.

Compressing Content

One effective way to solve this problem is to compress content. By applying various compression algorithms to the objects that make up a web application, the size of the files can be sometimes significantly reduced, improving transit time and consumed bandwidth for these objects. Since all popular browsers support receiving content in compressed form, this is a viable solution for addressing the problems associated with larger files in web applications.

The problem with compression is that the process causes extra burden for the server(s) responsible for this task. In today's highly dynamic web applications, with ever changing content, objects would have to be compressed as they're about to be served. Even if pre-compressing files was an option, the server would still have to

allocate resources for the task; not to mention the extra storage necessary to hold both compressed and non-compressed versions of all objects in order to support clients that cannot handle compressed content.

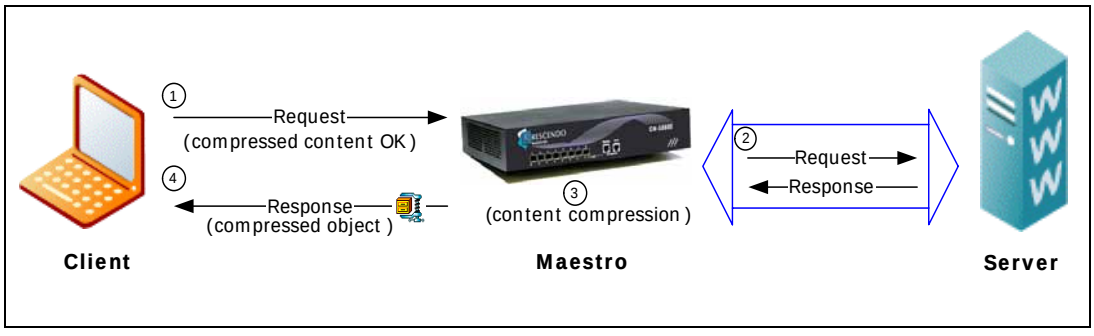
So, although compression is an attractive solution for reducing object sizes in web applications, it isn't always practical to deploy.

Content Compression

Maestro's server offloading and acceleration functionality includes a purely hardware-based and extremely powerful compression module. When fronting a web application, Maestro can compress content in transit from the application to its clients. Since Maestro has full visibility into the request/response chain between the client and the application, this is a vital service that it can provide the servers in order to optimize the application itself and improve the user experience.

By offering this service to an application, Maestro helps that application in three ways. Firstly, compression itself reduces the size of the web objects (depending on their compressibility) as they traverse the network towards the clients. This significantly reduces the client response time and, as a result, improves the quality of the user experience. Secondly, the overall bandwidth consumption for the application is also reduced; a byproduct that is vital for applications with limited outbound bandwidth. Finally, the servers are offloaded from performing this important task on their own. This way, the servers can focus their processing power on the application itself and not the associated overhead. The end result is significant optimization for both the application and its users.

Maestro's powerful, hardware-based compression functionality is enabled by a dedicated module that performs all compression tasks inline, in real time, and with zero latency, as objects traverse through the system. Rules determining the kinds of files to be compressed, and when to compress them, are configured in Maestro by mime type. The device recognizes those clients that can accept compressed content and performs compression on objects destined for these clients, based on the configured rules. Meanwhile, objects are requested from the servers over the highly optimized server-side connections that Maestro holds with each server. The servers continue to serve regular, non-compressed content at maximum speeds, while Maestro performs compression if and when necessary. The following diagram shows the overall flow of compressing content with Maestro:



Content Compression through Maestro

The process is as follows:

1. The client sends the request to the server, through Maestro, indicating that it can accept compressed content.
2. The request is sent to the server over one of the optimized back-end connections between Maestro and the server. The server responds to Maestro with the object, as it normally would.
3. Maestro examines the response to see if it matches one of the compression rules. If so, it performs inline content compression at real time via its powerful, hardware-based compression module.
4. The compressed object is served to the client, with the appropriate HTTP headers indicating that it's compressed.

For clients that do not support compressed content, Maestro serves the objects as they were received from the server. Likewise, if the server's response did not match one of the compression rules, the object would be served to the client as is.

Maestro's powerful content compression capability significantly improves an application by reducing user response time, minimizing overall bandwidth usage for the application, and offloading any compression functionality from the servers themselves.

Integrated Functionality

Maestro's real-time compression functionality operates with zero additional latency and is fully integrated with all other offloading services provided by the platform. When enabled, processing requests that allow compressed objects, or actually compressing the responses, is done as an integrated part of the system's flow; rather than out of line, as an orthogonal function. Requests that may result in compression are sent to the servers over the optimized back-end connections and enjoy the same benefits as all other requests, such as connection consolidation and buffering. Likewise, content compression can be deployed together with SSL offload functionality, resulting in Maestro serving compressed content over secure connections, if applicable. Because all services are handled in dedicated, task-specific hardware modules, multiple services can be enabled at once with no effect on Maestro's performance or functionality.

As more functions become available to Maestro, this level of integration will continue, allowing all services to be used concurrently in a fully integrated manner. At the same time, the powerful underlying architecture of the device will allow all functions to operate together without any degradation to the performance of the device.

Conclusion

Maestro provides an extremely fast and powerful hardware-based content compression module that can provide significant optimization for an application by reducing user response time, offloading server overhead, and minimizing the bandwidth used by the application. The overall functionality not only improves the quality of the client experience, but also the way in which the application operates. The compression functionality is fully integrated with all other server optimization features provided by Maestro to the application, allowing multiple services and offload functions to be performed on request/response chains, as necessary. At the same time, because of Maestro's unique and powerful task-specific, hardware-based architecture, all services can operate concurrently without any degradation in device performance.