WHITE PAPER

Accelerating and Optimizing Web-based Applications

www.crescendonetworks.com

Corporate Headquarters
6 Yoni Netanyahu Street
Or-Yehuda 60376, Israel
Phone: +972-3-538-5100

US Headquarters
633 Menlo Avenue, Suite 230
Menlo Park, CA 94025
Phone: (866)830-0400

Contents

Introduction

Internet performance and economy have become the driving force for network architects and operators. Several techniques have been developed to increase the performance and economy of server farms and the web applications they provide. This document looks at the structure of web traffic and discusses the benefits web applications receive from server offload technology.

Crescendo Networks' unique approach towards server optimization and application delivery, its advantages, and the functionality provided by AppBeat DC will be discussed.

Background

For Internet driven applications, web traffic makes up the majority of what servers deal with. HTTP (Hyper Text Transfer Protocol) is the communication protocol responsible for delivering web-based objects and applications from servers to clients. HTTP relies on TCP as its IP transport protocol. TCP is a session-based protocol that provides error checking and guarantees delivery of its upper layer protocol (HTTP). Although TCP is an ideal delivery mechanism for web traffic, it has drawbacks. TCP possesses overhead that often leads to significant resource utilization on a web server, and poor performance.

In the earliest versions of HTTP there was a one-to-one relationship between a web object request/response and a TCP session. Each TCP session between client and server was used to carry only one object from server to the client via HTTP. The figure below better illustrates this point:

Figure 1: A simple example of an HTTP request/response

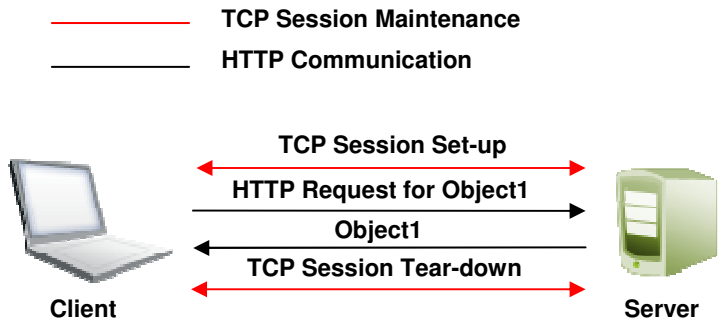


Figure 1 is a simplified version of a web transaction, and only shows the major stages of the process. A TCP session is opened between the client and the server. An HTTP request (e.g. for "Object1") is sent from the client to the server over this newly established session. The server responds with the object and after delivery and closes the connection. The process is repeated for every object requested. The TCP stack at each end is responsible for maintaining its side of the TCP connection. The session maintenance on

the server side is significant since it's handling a large number of connections from many clients. As a consequence most server resources are spent on TCP session handling, rather than object serving. It became obvious that this was not an optimal way for HTTP to interact with TCP leading to HTTP version 1.1 and **persistent connections**[1]. With persistent connections, the client may request multiple objects from a server over a single TCP connection.

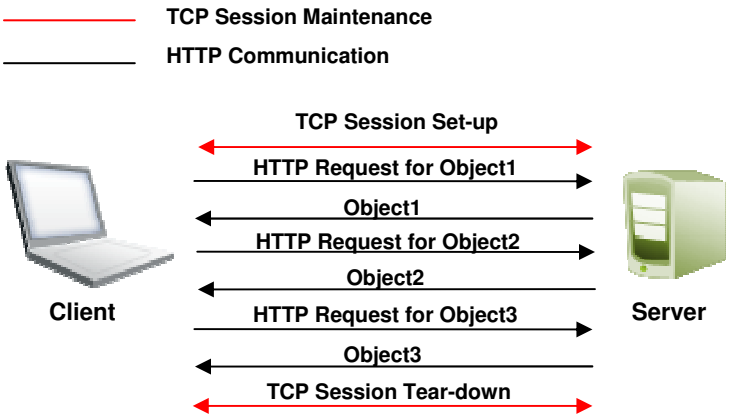


Figure 2: A simple example of a persistent HTTP connection

As before, the client opens the TCP connection to the server. Multiple objects may be handled over that single connection. Either side of the connection then tears down the session. The benefits of persistent connections are immediately clear. Both sides deal with fewer TCP sessions, and the client ends up using less bandwidth for session maintenance and more for object retrieval. At the same time, the server minimizes the amount of resources spent on TCP session maintenance with each client. Support for persistent connections has been one of the most major improvements to the HTTP protocol, and specifically to the way it interacts with TCP.

The Problem

Although persistency enhances the interaction between HTTP and TCP, it doesn't solve all TCP-related resource issues for servers. Persistent connections work well for the server if it has to deal with only a handful of web clients.

In widespread Internet applications, however, a server is faced with a large number of clients approaching the application through various WAN connections. This poses three areas of concern for a server, when it comes to TCP processing:

[1] Actually, persistent connections were introduced as an addition to version 1.0 of the HTTP protocol, after it was originally released. However, HTTP 1.1 is where persistent connections became an official part of the HTTP specifications.

- Dealing with a large number of TCP connection setup/teardown operations.
- Maintaining a large number of simultaneous connections.
- Extra processing necessary in dealing with WAN-based connections (slow start, congestion avoidance, dropped packets, retransmissions, etc). The bottom line is that in a web environment, servers have a significant burden when it comes to dealing with TCP session maintenance. Studies have shown that in a high throughput environment, a server can spend significant amounts of its resources on dealing with TCP connections. This is why offloading these tasks from servers is critical to gaining maximum performance.

The Solution

Although a number of approaches have been deployed for handling application scalability, they often fail to address server resource optimization. Load balancing traffic among multiple servers, for example, is a vital to a web application for scale and high availability, but it doesn't solely address a single server whose resources are not optimized. Likewise, upgrading to bigger/stronger servers is great for providing more available processing cycles, but those cycles are still not being used to their full potential.

The conclusion is that a broader approach is necessary to address the problem. A successful solution is one that significantly reduces a server's TCP-related pain points discussed above, while not requiring any change to the servers or the network around them.

Connection Consolidation is a technology that addresses all these issues, with the purpose of minimizing the number of connections a server handles, and maximizing the amount of resources dedicated to object delivery. The basic concept is relatively simple. Figure 3 and 4 shows a simplified version:

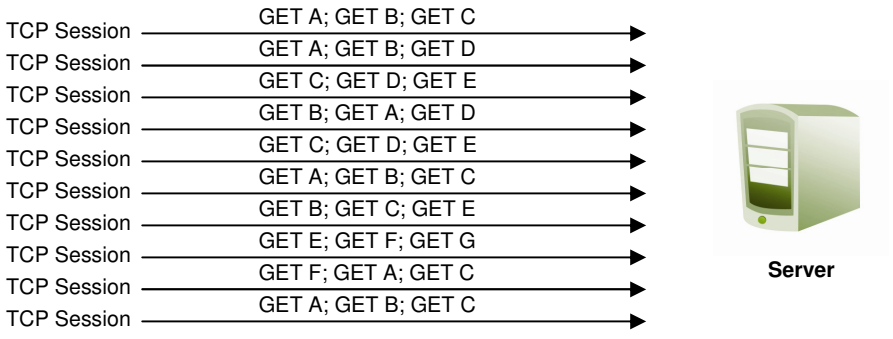
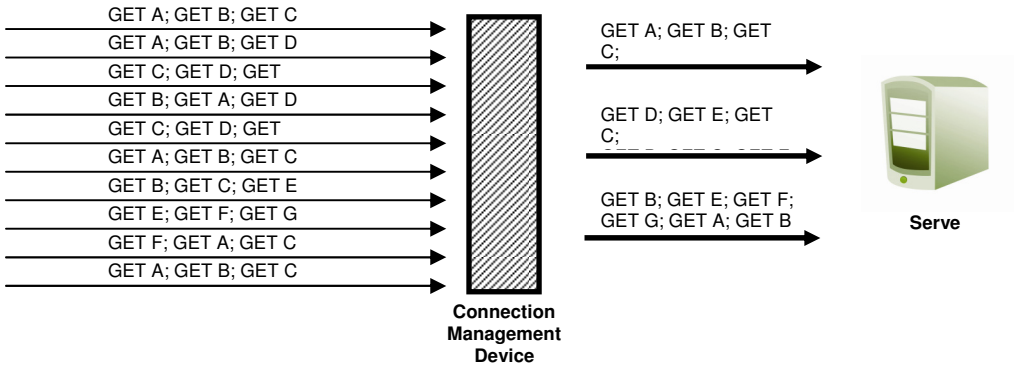


Figure 3: A web server handling HTTP clients through TCP

Figure 4: Connection consolidation



In Figure 3, the server has many TCP connections to maintain, while serving one or more objects per connection. Figure 4 shows the same scenario after Connection Consolidation has been implemented through an intermediary **Connection Management Device (CMD)**. The CMD Device acts as an intermediary between clients and servers. The task of the CMD is to act as the server towards the clients and appear as a client to the server. The CMD assumes the responsibility of dealing with all client-side connections. The CMD consolidates the front-end object requests from client-side connections to just a few server-side connections. The connections on the server side have long lifetimes and are used repeatedly as new object requests come in. Various algorithms may be used by the CMD to determine when a new session should be opened to the server, or which existing session a request should be carried over. The benefits of Connection Consolidation through the CMD become quickly apparent:

- Since the CMD handles all client side connections, the server is no longer tasked with rapidly setting up and tearing down connections.
- Because of Connection Consolidation, the number of simultaneous TCP sessions a server has to deal with is drastically reduced.
- The CMD shields the server from WAN-imposed client-side connection issues. This means the server resources are not affected by any delay, congestion, or packet loss in the client connection.
- The few connections that the server does use to the CMD allow the server to operate at minimum overhead and maximum performance, as if the clients were on the same LAN. Connection Consolidation is a significant enabler for web servers and their applications. CMDs must have a strong enough architecture to be able to handle the necessary TCP processing. Because of its unique location in a network, it makes sense for a CMD to provide other functionality such as compression, SSL offload, load balancing and more. This opens the door for a large number of auxiliary functionality, which warrants its own separate discussion.

AppBeat DC: Next Generation Web Application Delivery

Crescendo Networks' AppBeat DC is built from the ground up to provide superior server acceleration and resource optimization. AppBeat DC's purely hardware-based platform, Maestro, is based on over 80 micro- engines, explicitly tasked with various application-specific processes. This unique distributed design provides maximum flexibility for deploying new functionality, while maintaining extremely high levels of scalability and performance. AppBeat DC is deployed logically between the servers and the network; typically between the servers and the firewall/router. AppBeat DC is non-intrusive, and requires no service interruptions or network reconfiguration.

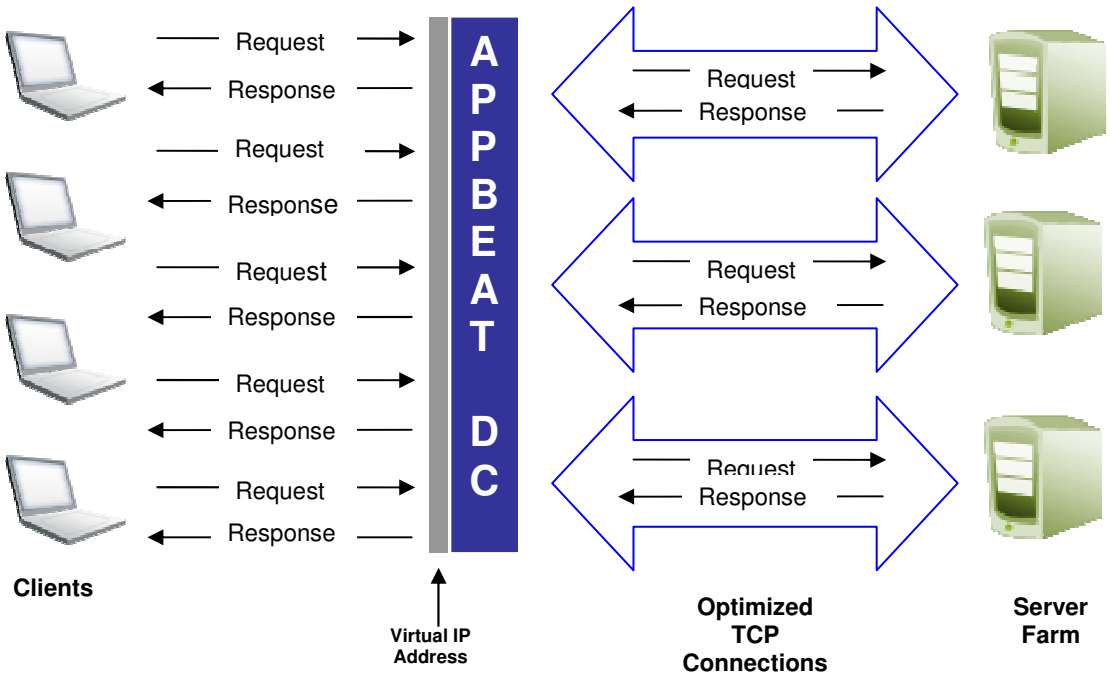


Figure 5:

A common network configuration for an AppBeat DC deployment

As shown in Figure 5, AppBeat DC assumes the final responsibility for delivering client requests to the servers. The core AppBeat DC's TCP connection management functionality is Crescendo's Short- Lived Transaction (SLT™) technology. SLT™ has three main components that work together to provide the relevant services for the network:

- **Connection Management Algorithm:** Server-side connections are managed through a set of advanced algorithms that provide an optimal approach to connection consolidation. These optimized connections operate at maximum LAN speeds and take into account factors such request type to facilitate transaction processing.
- **Request Processing Algorithm:** As a session terminating intermediary, AppBeat DC is responsible for terminating client connections, processing the requests that these connections carry, and then delivering them to the server

over existing, or new, server-side connections. SLT™ optimizes this process by providing the appropriate buffering for both requests and responses.

- **Response Optimization:** By completely shielding the server from network and client issues, AppBeat DC creates a highly optimized environment for servers. Servers deal with fewer connections and can transmit responses to the network at maximum throughput. Objects are served as quickly as possible, allowing the server to quickly move on to the next request to be processed.

AppBeat DC Features

AppBeat DC supports multiple functions, from compression to load balancing, on a single device. And unlike other application acceleration solutions, it can deploy all of these functions at once without any performance penalty.

- **TCP Offload, Multiplexing and Acceleration:** AppBeat DC significantly reduces the processing load on servers by handling TCP termination for clients. AppBeat DC receives all incoming requests and multiplexes and redirects them to servers over a controlled number of persistent server-side connections. This approach relieves the servers of the connection setup, teardown and management processes that normally consume valuable server resources. The result is a dramatic increase in application performance.
- **Compression:** Object compression minimizes the number of bytes that need to travel from web site to client. Most browsers accept compressed data[2]. AppBeat DC improves client response times and significantly reduces bandwidth requirements. With its dedicated, solid-state compression processor, AppBeat DC can compress content by up to 85%, operating at speeds of up to 3 Gbps, with zero latency. It supports multiple compression levels, all with guaranteed zero latency.
- **Comprehensive Load Balancing:** Load balancing shields users from server failures or overloaded, slow servers, while enabling the even distribution of resources across the data center. AppBeat DC provides load balancing on a request-by-request basis, determining the optimal server for each request based upon actual HTTP load. Global server load balancing functionality extends load balancing beyond the single data center, enabling traffic distribution and control across geographically distributed data centers.
- **SSL Acceleration and Offload**: Encryption and decryption tasks involved in handling SSL (Secure Sockets Layer) are a severe burden to any web server. AppBeat DC offloads this CPU-intensive task, freeing server resources and making the site faster and more secure. AppBeat DC handles both SSL session setup and bulk data encryption tasks, employing dedicated hardware designed to accelerate these resource-intensive processes.
- **Application Assurance and Availability:** Severe changes in user patterns, traffic spikes and other traffic anomalies can seriously affect server

[2] Commonly used compression algorithms are gzip and deflate, both supported by common browsers and AppBeat DC.

performance. AppBeat DC maintains a normalized operating environment, shielding servers from erratic client behavior as well as malicious attacks (DDoS) and flash crowd events.

Conclusion

The benefits of accelerating and optimizing Web-based applications are self-evident. AppBeat DC's unique hardware-based capability provides the highest performance and feature concurrency in the industry. AppBeat DC's comprehensive feature set is fully integrated with all of its application optimization and server offload functionality. Load balancing can be used together with SSL offload or content compression, for example, while still being able to use all the TCP optimization functionality of the device. Because of its unique and powerful task-specific, hardware-based architecture, the highly scalable, multi-gigabit AppBeat DC application delivery solution provides all these functions at maximum performance.

About Crescendo Networks

Crescendo Networks is the recognized performance leader for accelerating and optimizing the delivery of business-critical, Web-enabled applications. The company's unique multi-tier application delivery architecture dramatically improves the operation of today's demanding application infrastructure. The world's largest corporations and fastest growing Web properties rely on Crescendo for the application performance and efficiency needed to ensure usability, facilitate rapid business growth, lower IT costs and capture additional revenue. To learn more about Crescendo Networks' application delivery solutions, visit www.crescendonetworks.com.