Application Acceleration 2.0 : Breaking Browser Limitations

Lori MacVittie, 2008-17-06

Several years ago it became necessary for browsers to put limitations on the number of simultaneous connections allowed not only to be open, but how many of *those* could be open to a single domain. This helped prevent unintentional (and, in some cases, intentional) denial of service situations where a site's poor web server just couldn't keep up with the demand. After all, managing TCP/IP connections is expensive and if one user hogs all the available connections (as determined by web server configuration and RAM) there may be hundreds of users out there that are denied access to the latest ... well, whatever the latest *thing* might be that is causing users to flock to that site.

Enter Web 2.0, full of its rich, interactive interfaces and veritable cornucopia of gadgets, widgets and other independent nifty-neato-keen components from which community driven sites were suddenly being composed.

And then things s l o w e d down. While it's not always the case that a real-time updating gadget will completely fail if the browser reaches its connection limit, it can certainly slow things down while requests are queued, waiting for an available connection. It's a scenario that we've often seen in the past - but usually on the server side of the world. Now it's happening more and more often on the client side of the world, thanks to Web 2.0 and technologies like AJAX (Asynchronous Javascript and XML).

So how do you solve that problem? It's been suggested that client-side load balancing can alleviate this issue, and while that may be true it's just not A Good Idea to be hard-coding such things on the client. That's because if you take the client-side load balancing route you end up hard-coding additional hosts into the application without a reliable mechanism for handling the potential availability issues that could arise. Basically, instead of making your application *more* reliable, you've actually introduced additional points of failure into the equation.

But the other solution, allowing more connections on the client could certainly adversely affect the performance and capacity of the servers; it would put us right back where we started.

What's necessary is a solution that not only allows more connections on the client, but also protects the servers from becoming overwhelmed by the additional connections and requests. Something like a web application acceleration solution coupled with the capabilities of an application delivery platform.

Combining the features of WebAccelerator with the optimization features in BIG-IP offers a unique solution that solves not only performance and reliability issues on the client, but reduces overhead on the servers resulting in improved performance and capacity - both of which are intrinsically important to the success of a Web 2.0 site/application.

BIG-IP and WebAccelerator combine two core features that result in an excellent solution to this Web 2.0 delivery problem.

> **MultiConnect** – Enables browsers to open more simultaneous connections between the browser and web application for increased parallel data transfers.
>
> **OneConnect** - Aggregates millions of client requests into hundreds of connections on the server, reducing the overhead of managing connections and improving overall response time.

For more information on these awesome technologies, check out this page on WebAccelerator and this User Experience Guide on Server Offloading techniques.

Imbibing: Mountain Dew