Long Live(d) AJAX

Lori MacVittie, 2009-07-10

The problem of AJAX, interstitial request patterns, and the effect on the performance and availability of your applications.

There are several reasons why applications need to be scaled out but they all come down to essentially addressing the same core problem: resource consumption. In the case of networked applications this often means specifically TCP

connection resources. Now most people don't think of TCP connections as a resource, per se, but every web and application server has an upper limit to the number of TCP connections it can hold open at any given time. In some cases this limit can be changed; for example Apache can be recompiled with a higher limit to increase capacity but depending on the underlying resources (RAM, CPU) available this can result in a degradation of performance. And even if you do recompile Apache with a higher limit the operative part of that statement is higher *limit.* No matter what you do at some point you will run out resources.

In many cases administrators have learned that by tweaking the idle and timeout values in the web/application server they can eek out a higher capacity. That's because the TCP connections are being recovered more quickly and thus more requests can be served by essentially making sure there are no "hangers on" wasting resources that other users could be using.

But again, even if you tweak the configuration you're still only eeking out a slightly higher capacity and potentially degrading performance. In some cases *that* counts against availability metrics just as a complete failure to reply because if an application is so slow that a user can't efficiently use it, it's really not available for the purposes of serving the business.

Load balancing (and clustering) solves this problem by doubling, tripling, etc… the available resources by adding more web/application servers that can answer requests. This process is transparent to the user – they probably don't even know there's a load balancer between them and the server.

Along comes Web 2.0 and AJAX and suddenly one of two things happens: web/application servers that had a capacity of X in the past suddenly seem to have a capacity of X – 20% or users begin complaining that the application is failing – a lot. The reason for this phenomenon is a combination of several behaviors associated applications that take advantage of AJAX: interstitial request patterns. These request patterns interrupt the normal idle timeout that causes TCP connections to be closed (and thus frees up the resources for other users) and keeps them open as long as the AJAX-based requests are being made. Stock quotes refreshing on a page is probably the most common example, but "users online" is another good example of an application behavior that exhibits an interstitial request pattern.

Interstitial request patterns happen in between the "natural" timeout that would occur in a less-interactive interface and thus result in what we can call "long-lived TCP connections." Idle/timeout settings in web servers are often in the 15-30 second range. This means if no requests are received on the connection within the specified time, it can be closed and the resources freed to be used by another user. But AJAX-based applications often make requests in specified intervals, often as low as every 5 seconds. Other applications may not initiate an automatic request on an interval, but may contain inline-editable fields which update individually by making a request. Depending on the speed with which the user interacts, this can have the same effect as automatic update requests displaying near-time information.

The result is that TCP connections remain open, i.e. long-lived, and decrease the overall capacity of the web/application server by maintaining a technological death-grip on the resources.

SOLUTIONS

There are a couple of ways to address the consumption of resources by interstitial request patterns. The first involves architecting application such that server-side scripts and applications that will respond to these interstitial requests are on different servers than static or even traditional dynamic content. This allows you to better tune the web/application server according to the request patterns for each type of content and more efficiently allocate virtual machines to scale out the web/application servers responding to interstitial requests. It also ensures that the consumption of processing resources by the interstitial requests do not overwhelm the system and cause requests for static and traditional dynamic content to fail or become unacceptably slow.

Another solution is to leverage the TCP multiplexing capabilities of an application delivery controller. This solution separates the client from the server and reuses TCP connections on the server-side, making it possible to support millions of client-side connections with only a few hundred server-side connections.

Both solutions require an investment; the first requires additional hardware/software and possibly virtualization solutions if you go that route. The second requires an investment in a TCP multiplexing capable application delivery controller. The first solution may require some modification to your application (if moving interstitial requests to a separate host and you weren't too careful about loosely coupling host names to requests in your code) and the second requires no modifications at all.

Regardless of which way you go, the good news is that there *is* a solution to the performance and availability challenges associated with AJAX-based interfaces and the resource hogging nature of interstitial request patterns.

- Have a can of Duh! It's on me
- Impact of Load Balancing on SOAPy and RESTful Applications
- The Impact of AJAX on the Network
- The AJAX Application Delivery Challenge
- What is server offload and why do I need it?
- 3 Really good reasons you should use TCP multiplexing
- SOA + WOA = NOA
- SOA & Web 2.0: The Connection Management Challenge
- The Impact of the Network on AJAX
- When Reuse Becomes a Four Letter Word

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com