Scaling AJAX Applications is More About Architecture than Apache

Lori MacVittie, 2010-08-02

Scaling applications that include AJAX and non-AJAX components may require more than just tuning your web server

A common problem after deploying a Web 2.0 AJAX-based application shows itself through poor performance or lower capacity on the server, often both. Web serving tuning is almost always the first step in improving performance and capacity, but the inherently competing behavior of AJAX-requests and “normal” HTTP requests quickly becomes problematic as well. Tune for the AJAX requests and performance of regular old HTTP requests suffers. Tune for regular old HTTP requests, and performance of AJAX-requests suffer.

This is primarily because of the way in which the client-side application, the browser, interacts with the server. “Regular old HTTP requests” are typically those that GET a piece of content, static or dynamic, and that’s it. There may be many of these requests whenever a page (URI) is requested – all the images, client-side scripting files, style sheets, etc… – but they are not interactive. The browser requests them, receives them, and that’s it. AJAX-based requests, however, are inherently interactive. They are often automatically refreshed on an ongoing basis, on a prescheduled interval, or invoked by the user as they interact with the application. These requests are not “load and forget” like their traditional staticesque counterparts, but rather they are expected to be made often.

The overhead associated with opening and closing connections is well understood, and it is often the case that the web server configuration will be adjusted to meet the more demanding nature of the AJAX-based requests in an application. This is often accomplished by ensuring the KeepAlive setting (in Apache) is “on” and that the KeepAliveTimeout (in Apache) is high enough that AJAX-based requests occur ***before*** the timeout closes the connection. This allows the continued reuse of an existing connection between the browser and the server and improves performance. But it also ties up resources on the server keeping that connection open, which reduces the overall capacity of the server in terms of its ability to serve users. Optimally a short KeepAliveTimeout, if any, is best for non-interactive requests and often disabling KeepAlive actually improves performance for non-interactive applications.

Obviously these two behaviors are completely at odds with one another.

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

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com