# Client-side vs Server-side Load Balancing

**Lori MacVittie, 2007-08-10**

Lei Zhu @ Digital Web Magazine has an interesting article on Client Side Load Balancing for Web 2.0 Applications. It is interesting in that it presents an alternative mechanism for implementing high-availability *without* the use of an intermediate load balancing solution. His solution relies solely on the client and takes advantage of the dynamic nature of Web 2.0.

The problem with Lei's article is that there are a few assumptions made that are simply inaccurate.

Lei contends that the negatives to using an intermediate load balancing solution are:

> 1. There is a limit to the number of requests the load balancer itself can handle. However, this problem can be resolved with the combination of round-robin DNS and dedicated load balancers.
> *The upper bounds of a hardware load balancing solution are typically higher than any given server can handle. We are talking about hundreds of thousands of requests per second. Round-robin DNS, while an industry standard, is one of the most rudimentary implementations of "load balancing" that exists. Load balancers, a.k.a. application delivery controllers, today are capable of making routing decisions based on much more intelligent factors than a simple list of servers. There is no server that can handle more requests than a load balancing solution today. Even SMB load balancers can process requests faster than any server and maintain a session table much larger than existing web and application servers.*
> 2. There is an extra cost related to operating a dedicated load balancer, which can run into tens of thousands of dollars. The backup load balancer generally does nothing other than wait for the primary to fail.
> *It is absolutely true that an intermediate load balancer is going to require an investment. That the backup "does nothing" is entirely dependent upon the implementation. Load balancers today are capable of both active-standby (this is Lei's assumption here) as well as active-active configurations. In an active-active configuration both load balancers are working all the time. It is completely up to the implementor to determine which configuration should be used.*

Lei also claims:

> *It is easier to make the client code and resources highly available and scalable than to do so for the servers—serving non-dynamic content requires fewer server resources.*

There are two statements here, one regarding the ease with which you can deploy a highly available site and that serving non-dynamic content requires fewer server resources. The latter is absolutely correct, it does indeed take less resources on the server to serve non-dynamic content. The former assertion, however, is not entirely true. Lei's article describes a methodology that requires modifications to not only the application (the client code) but also to the infrastructure (additional servers *and* entries in DNS to accomodate the new servers). A *good* application delivery controller will **not** require changes to the application and will be less disruptive to the infrastructure because it effectively intercepts requests for the domain and handles them without changes to the server-side of the equation.

For example, if your domain is *www.example.com* and you implement a load balancing solution, the load balancer *becomes* www.example.com. Servers can be added to the pool (a.k.a. cluster, farm) of servers and it will then distribute requests amongst those servers without any further changes to the client or the servers. A good application delivery controller can distribute requests based on more than DNS round-robin algorithms, and can base its decisions *in real time* on the health and status of any given server. This is one the problems with Lei's arguments - he bases his decision upon a single, outdated mechanism for load balancing and uses that as the basis for recommending a client-side solution. And as far as scalability - let's look at that for a moment.

In the intermediate load balancer scenario if we need more resources we (1) add another server, and (2) add it to the pool on the load balancer. In Lei's scenario we (1) add another server, (2) reconfigure the client-application, and (3) add another entry to DNS. Lei's solution is much more invasive and disruptive to every piece of the equation than the intermediate load balancing solution scenario, and Lei's solution is not guaranteed to work as he cannot ensure that the client browser's caching scheme will definitively update and include the new server.

And lest we forget, Lei does not discuss the problem of persistence inherent in his client-side solution. A purely round-robin based solution that ignores client-session is limited in use and value. The solution ***must*** take into consideration the possibility that a client needs to be tied to a particular server for the duration of a session. This is particularly true of e-commerce sites. A client-side solution that does not take this into consideration (and Lei's solution does not) will not be useful in many situations. A good application delivery controller ***by default*** is capable of handling persistence and "stickiness" to a server - and again does so without requiring modification to the client or server side code.

Lei's client-side load balancing solution is novel, and while it certainly might be interesting for individual developers whose livelihoods do not rely on the availability of their site this solution is ***not*** something that should be recommended lightly and without a great deal of forethought. The assumptions upon which his article is based are, with the exception the additional cost of a load balancing solution, outdated and inaccurate. Client-side load balancing requires more maintenance, more initial work to deploy, and introduces security risks by "opening the organization's kimono" and showing clients what lies beneath, namely the infrastructure architecture.

*Imbibing: Coffee*

Technorati tags: MacVittie, F5, load balancing, application delivery, scalability, high availability, Web 2.0

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