Why Layer 7 Load Balancing Doesn’t Suck

Lori MacVittie, 2012-14-03

Alternative title: Didn't We Resolve This One 10 Years Ago?

There's always been a bit of a disconnect between traditional network-focused ops and more modern, application-focused ops when it comes to infrastructure architecture. The arguments for an against layer 7 (application) load balancing first appeared nearly a decade ago when the concept was introduced. These arguments were being tossed around at the same time we were all arguing for or against various QoS (Quality of Service) technologies, namely the now infamous "rate shaping" versus "queuing" debate. If you missed that one, well, imagine an argument similar to that today of "public" versus "private" clouds. Same goals, different focus on how to get there.

Seeing this one crop up again is not really any more surprising than seeing some of the other old debates bubbling to the surface. cloud computing and virtualization have brought network infrastructure and its capabilities, advantages and disadvantages – as well as its role in architecting a dynamic data center – to the fore again. But seriously? You'd think the arguments would have evolved in the past ten years.

> While load balancing hardware marketing execs get very excited about the fact that their product can magically scale your application by using amazing Layer 7 technology in the Load balancer such as cookie inserts and tracking/re-writing. What they fail to mention is that any application that requires the load balancer to keep track of session related information within the communications stream can never ever be scalable or reliable.
>
> -- Why Layer 7 load balancing sucks…

I myself have never shied away from mentioning the session management capabilities of a full-proxy application delivery controller (ADC) and, in fact, I have spent much time advocating on behalf of its benefits to architectural flexibility, scalability, and security. Second, argument by selective observation is a poor basis upon which to make any argument, particularly this one. While persistence-based load balancing is indeed one of the scenarios in which an advanced application delivery controller is often used (and in some environments, such as scaling out VDI deployments, is a requirement) this is a fairly rudimentary task assigned to such devices. The use of layer 7 routing, aka page routing in some circles (such as Facebook), is a highly desirable capability to have at your disposable. It enables more flexible, scalable architectures by creating scalability domains based on application-specific functions.

There are a plethora of architectural patterns that leverage the use of an intelligent, application-aware intermediary (i.e. layer 7 load balancing capable ADC). Here's a few I've written myself, but rest assured there are many more examples of infrastructure scalability patterns out there on the Internets:

- Infrastructure Architecture: Avoiding a Technical Ambush
- Infrastructure Scalability Pattern: Partition by Function or Type
- Applying Scalability Patterns to Infrastructure Architecture
- Infrastructure Scalability Pattern: Sharding Streams

Interestingly, ten years ago this argument may have held some water. This was before full-proxy application delivery controllers were natively able to extract the necessary HTTP headers (where cookies are exchanged). That means in the past, such devices had to laboriously find and extract the cookie and its value from the textual representation of the headers, which obviously took time and processing cycles (i.e. added latency). But that's not been the case since oh, 2004-2005 when such capabilities were recognized as being a requirement for most organizations and were moved into native processing, which reduced the impact of such extraction and evaluation to a negligible, i.e. trivial, amount of delay and effectively removed as an objection this argument.

THE SECURITY FACTOR

Both this technique and trivial tasks like tracking and re-writing do, as pointed out, require session tracking – TCP session tracking, to be exact. Interestingly, it turns out that this is a Very Good Idea™ from a security perspective as it provides a layer of protection against DDoS attacks at lower levels of the stack and enables an application-aware ADC to more effectively thwart application-layer DDoS attacks, such as SlowLoris and HTTP GET floods.

A simple, layer 4 load balancing solution (one that ignores session tracking, as is implied by the author) can neither recognize nor defend against such attacks because they maintain no sense of state. Ultimately this means the application and/or web server is at high risk of being overwhelmed by even modest attacks, because these servers are incapable of scaling sessions at the magnitude required to sustain availability under such conditions.

This is true in general of high volumes of traffic or under overwhelming loads due to processor-intense workloads. A full-proxy device mitigates many of the issues associated with concurrency and throughput simply by virtue of its dual-stacked nature.

SCALABILITY

As far as scalability goes, really? This is such an old and inaccurate argument (and put forth with no data, to boot) that I'm not sure it's worth presenting a counter argument. Many thousands of customers use application delivery controllers to perform layer 7 load balancing specifically for availability assurance. Terminating sessions does require session management (see above), but to claim that this fact is a detriment to availability and business continuity shows a decided lack of understanding of failover mechanisms that provide near stateful-failover (true stateful-failover is impossible at any layer). The claim that such mechanisms require network bandwidth indicates either a purposeful ignorance with respect to modern failover mechanisms or simply failure to investigate. We've come a long way, baby, since then.

In fact, it is nigh unto impossible for a simple layer 4 load balancing mechanism to provide the level of availability and consistency claimed, because such an architecture is incapable of monitoring the entire application (which consists of all instances of the application residing on app/web servers, regardless of form-factor). See axiom #3 (Context-Aware) in "The Three Axioms of Application Delivery".

- The Case (For & Against) Network-Driven Scalability in Cloud Computing Environments
- The Case (For & Against) Management-Driven Scalability in Cloud Computing Environments
- The Case (For & Against) Application-Driven Scalability in Cloud Computing Environments
- Resolution to the Case (For & Against) X-Driven Scalability in Cloud Computing Environments

The author's claim seems to rest almost entirely on the argument "Google does layer 4 not layer 7" to which I say, "So?" If you're going to tell part of the story, tell the entire story. Google also does not use a traditional three-tiered approach to application architecture, nor do its most demanding services (search) require state, nor does it lack the capacity necessary to thwart attacks (which cannot be said for most organizations on the planet). There is a big difference between modern, stateless applications (and many benefits) and traditional three-tiered application architectures.

Now it may in fact be the case that regardless of architecture an application and its supporting infrastructure do not require layer 7 load balancing services. In that case, absolutely – go with layer 4. But to claim layer 7 load balancing is not scalable, or resilient, or high-performance enough when there is plenty of industry-wide evidence to prove otherwise is simply a case of not wanting to recognize it.

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