Load Balancing For Developers: Security and TCP Optimizations

Don MacVittie, 2011-07-04

It has been a while since I wrote a Load Balancing for Developers installment, and since they're pretty popular and there's still a lot about Application Delivery Controllers (ADCs) that are taken for granted in the Networking industry but relatively unknown in the development world, I thought I'd throw one out about making your security more resilient with ADCs.

For those who are just joining this series, here's the full list of posts I've tagged as Load Balancing for Developers, though only the ones whose title starts with "Load Balancing for Developers" or "Advance Load Balancing for Developers" were actually written from this perspective, utilizing our fictional web application Zap'N'Go! as an example. This post, like most of them, doesn't require that you read the other entries in the "Load Balancers for Developers" series, but if you're interested in the topic, they are all written from the developer's perspective, and only bring in the networking/ops portions where it makes sense.

So your organization has a truly successful web application called Zap'N'Go! That has taken the Internet by storm. Your hits are in the thousands an hour, and orders are rolling in. All was going well until your server couldn't keep up and you went to a load balanced scenario so that multiple servers could share the load. The problem is that with the money you've generated off of Zap'N'Go, you've bought a competitor and started several new web applications, set up a forum or portal for your customers to communicate with you and each other directly, and are using the old datacenter from the company you purchased as a redundant datacenter in case the worst should happen. And all of that means that you are suffering server (and VM) sprawl. The CPU cycles being eaten up by your applications are truly astounding, and you're looking into ways to drive them down. Virtualization helped you to be more agile in responding to the requests of the business, but also brings a lot of management overhead in making certain servers aren't overloaded with too high a virtual density.

One of the cool bits about an ADC is that they do a lot more than load balance, and much of that can be utilized to improve application performance without re-architecting the entire system. While there are a lot of ways that an ADC can improve application performance, we'll look at a couple of easy ones here, and leave some of the more difficult or involved ones for another time. That keeps me in writing topics, and makes certain that I can give each one the attention it deserves in the space available.

The biggest and most obvious improvement in an ADC is of course load balancing. This blog assumes you already have an ADC in place, and load balancing was your primary reason for purchasing it. While I don't have market numbers in front of me, it is my experience that this is true of the vast majority of ADC customers. If you have overburdened web applications and have not looked into load balancing, before you go rewriting your entire system, take a look at the rest of this series. There really are options out there to help.

After that win, I think the biggest place – in a virtualized environment – that developers can reap benefits from an ADC is one that developers wouldn't normally think of. That's the reason for this series, so I suppose that would be a good thing. Nearly every application out there hits a point where SSL is enabled. That point may be simply the act of accessing it, or it may be when they go to the "shopping cart" section of the web site, but they all use SSL to protect sensitive user data being passed over the Internet. As a developer, you don't have to care too much about this fact. Pay attention to the protocol if you're writing at that level and to the ports if you have reason to, but beyond that you don't have to care. Networking takes care of all of that for you.

But what if you could put a request in to your networking group that would greatly improve performance without changing a thing in your code and from a security perspective wouldn't change much – most companies would see it as not changing anything, while a few will want to talk about it first? What if you could make this change over lunch and users wouldn't know the difference?

Here's the background. SSL Encryption is expensive in terms of CPU cycles. No doubt you know that, most developers have to face this issue head-on at some point. It takes a lot of power to do encryption, and while commodity hardware is now fast enough that it isn't a problem on a stand-alone server, in a VM environment, the number of applications requesting SSL encryption on the same physical hardware is many times what it once was. That creates a burden that, at this time at least, often drags on the hardware. It's not the fault of any one application or a rogue programmer, it is the summation of the burdens placed by each application requiring SSL translation.

One solution to this problem is to try and manage VM deployment such that encryption is only required on a couple of applications per physical server, but this is not a very appealing long-term solution as loads shift and priorities change. From a developers' point of view, do you trust the systems/network teams to guarantee your application is not sharing hardware with a zillion applications that all require SSL encryption? Over time, this is not going to be their number one priority, and when performance troubles crop up, the first place that everyone looks in an in-house developed app is at the development team. We could argue whether that's the right starting point or not, but it certainly is where we start.

Another, more generic solution is to take advantage of a non-development feature of your ADC. This feature is SSL termination. Since the ADC sits between your application and the Internet, you can tell your ADC to handle encryption for your application, and then not worry about it again. If your network team sets this up for all of your applications, then you have no worries that SSL is burning up your CPU cycles behind your back.

Is there a negative? A minor one that most organizations (as noted above) just won't see as an issue. That is that from the ADC to your application, communications will happen in the clear. If your application is internal, this really isn't a big deal at all. If you suspect a bad-guy on your internal network, you have ***much*** more to worry about than whether communications between two boxes are in the clear. If you application is in the cloud, this concern is more realistic, but in that case, SSL termination is limited in usefulness anyway because you ***can't*** know if the other apps on the same hardware are utilizing it.

So you simply flick a switch on your ADC to turn on SSL termination, and then turn it off on your applications, and you have what the ADC industry calls "SSL offload". If your ADC is purpose-built hardware (like our BIG-IP), then there is encryption hardware in the box and you don't have to worry about the impact to the ADC of overloading it with SSL requests, it's built to handle the load. If your ADC is software or a VM (like our BIG-IP LTM VE), then you'll have to do a bit of testing to see what the tolerance level for SSL load is on the hardware you deployed it on – but you can ask the network staff to worry about all of that, once you've started the conversation.

Is this the only security-based performance boost you can get? No, but it is the easy one. Everything on the Internet remains encrypted, but your application is not burdening the server's CPU with encryption requests each time communications in or out occur.

The other easy one is TCP optimizations. This one requires less talk because it is completely out of the realm of the developer. Simply put, TCP is a well designed protocol that sometimes gets bogged down communicating and has a lot of overhead in those situations. Turning on TCP optimizations in your ADC can reduce the overhead – more or less, depending upon what is on the other end of the communications network – and improve ***perceived*** performance, which honestly is one of the most important measures of web application availability. By making it seem to load faster, you've improved your customer experience, and ***nothing*** about your development has to change. TCP optimizations are not new, and thus the ones that are turned on when you activate the option on most ADCs are stable and won't disrupt most applications. Of course you should run a short test cycle with them enabled, just to be certain, but I would be surprised if you saw any issues. They're not unheard of, but they are ***very*** rare.

That's enough for now, I think. I don't want these to get so long that you wander off to develop some more. Keep doing what you do. And strive to keep your users from doing this.

Slow apps anger users

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

©2016 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at f5.com. Any other products, services, or company names referenced herein may be trademarks of their respective owners with no endorsement or affiliation, express or implied, claimed by F5. CS04-00015 0113