Infrastructure Matters: Challenges of Cloud-based Testing

Lori MacVittie, 2009-10-06

An interesting thing happened on the way to testing that application from the cloud. We broke the innertubes!

Pros and Cons of Application Testing in the Cloud

> A firm wanted to test their application and need 100 browser instances. In the old days it would have required 100 machines -- that would be a massive undertaking. Even with hardware virtualization, you would need 5 to 10 machines, and there would be some complex configuration issues. However, by putting it all in the cloud, they were able to sync up 100 virtual instances of the browsers and take them down over a weekend at a dramatic cost reduction.

I won't argue the point about needing one machine per browser instance for application testing (even though it's not entirely accurate as plenty of tools support such requirements without requiring multiple machines – Ixia and Spirent to name just a couple of my favorites) because there's so much more that needs to be said in terms of infrastructure and this kind of test environment.

The truth is that when you start spinning up multiple instances of any client for the purposes of testing an application you can inadvertently doom your test to fail due to underlying infrastructure attributes that aren't always all that obvious to the uninitiated regardless of whether those clients are in the cloud or not.

TRIGGERS and POLICIES and PROBLEMS, OH MY

Back in the days of AOL and Compuserve we had what the industry termed the "megaproxy". The megaproxy caused all sorts of issues for web sites and web infrastructure because large blocks of visitors coming from one of the large providers all appeared to be accessing sites from the same IP address. The specific problem of load balancing, though largely a non-issue today, was solved primarily by moving from layer 4 source IP-based persistence to cookie-based layer 7 persistence.

It turns out that cookie-based persistence, however, is something often overlooked by neophyte load balancing administrators when scaling out an application for the first time. Combine this with a cloud-based client scenario, which essentially reintroduces the mega-proxy problem, and we're right back in the early days of applications not working as expected due to configuration and infrastructure issues. Unless the cloud provider has an unlimited source of public IP addresses, they're going to have to NAT (masquerade) all of the clients in the cloud as just a few IP addresses. That means the tested application – and the underlying infrastructure – is going to suddenly see a lot of requests coming from the *same* IP addresses.

This is where things get wonky.

There are a couple issues that can arise from this scenario, the most obvious being the problem with load balancing and persistence. If you're only using Layer 4 persistence based on IP address, the load is going to be distributed very unevenly across the servers (virtual or physical) responsible for scaling your site. That means your testing is going to be useless for determining capacity and even average end-user performance, as more heavily loaded servers inherently degrade application performance.

A second, more well-hidden surprise for network administrators can arise if internal network infrastructure (that's switches and routers, remember) is also doing any load balancing via port trunking. This is because the load balancing across a set of trunked ports is almost always based on a hash algorithm that relies on MAC or IP address, which in the aforementioned scenario will lead to traffic being routed over only a few – or one – port. If that port is unable to handle all the traffic it will quickly become oversubscribed and degrade performance due to the congestion, retransmissions, and potentially QoS policies that kick in.

Using cloud-hosted agents for application testing necessitates that traffic is arriving from ***outside*** the organization. In addition to application network and network infrastructure potential pitfalls, awareness of what security policies are in place that may be triggered by repeated requests sent from the same "client' is a necessity.

There often exist security policies that dictate how many requests per second/minute/hour can be made by any single client in an attempt to thwart DoS (Denial of Service) attacks as well as to ensure the quality of the service for all users by not allowing a single client to "hog" the application. This obviously can have a very negative affect on the ability to stress/load test an application as the number of requests that can come from the client will be limited by said policies. Firewalls, too, may put artificial limits on connections and bandwidth based on any number of variables: time of day, location, ports, etc… and policies in its infrastructure may be triggered during a test.

INFRASTRUCTURE MATTERS

Basically the most important thing to remember is that if you're using a cloud-based testing service or a cloud-based testing environment that you're probably testing a ***production*** environment and its underlying infrastructure.

Let me repeat that one because it's the most important thing I've said in this post: you are not just testing the application, you are testing its ***infrastructure***, too. And remember that when that traffic is generated externally that you're also poking at the infrastructure of the Internet, and may well trigger policies "out there" that your ISP – or others – have set in place to protect your applications and network.

It's therefore important to understand what policies may be triggered by sudden bursts of traffic as well as what security policies may be triggered by the constant hammering of a single client at an application. It's also important to understand the flow of requests and responses through the infrastructure and recognize the various touch points along that path. Doing so may ease the troubleshooting process greatly.

Testing applications in a lab or even a QA environment has its own pitfalls as you are testing only the application and as we all know, no application is an island. But testing them in a production environment, while ideal for gathering a real view of how the application will perform when faced with users and customers, has its own challenges because you're testing against a real, live environment with all the mechanisms in place to ensure quality, performance, and security of those applications.

Gather up the right folks before you start testing and understand the potential pitfalls and how to properly address them – or avoid them – before you spend money firing up hundreds of instances of clients in the cloud or you may find yourself wasting more money than you might have just testing them on your own.

Related articles & blogs:

- Tony Bourke: Mega Proxy Not So Mega, Akshually
- Cloud Computing: Is Your Cloud Sticky? It Should Be
- This network ain't big enough for the both of us
- OVF: A Few Layers Short of a Full Stack
- Five Key Questions Developers Need to Ask Before Starting the Troubleshooting Process
- Load Testing as a Service: A Look at Load Impact (beta)

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