Five questions you need to ask about load balancing and the cloud

Lori MacVittie, 2009-25-06

Whether you are aware of it or not, if you're deploying applications in the cloud or building out your own "enterprise class" cloud, you're going to be using load balancing. Horizontal scaling of applications is a fairly well understood process that involves (old skool) server virtualization of the network kind: making many servers (instances) look like one to the outside world. When you start adding instances to increase capacity for your application, load balancing necessarily gets involved as it's the way in which horizontal scalability is implemented today.

The fact that you may have already deployed an application in the cloud and scaled it up without recognizing this basic fact may lead you to believe you don't ***need*** to care about load balancing options. But nothing could be further from the truth. If you haven't asked already, you should. But more than that you need to understand the importance of load balancing and its implications to the application. That's even more true if you're considering an enterprise cloud, because it will most assuredly be your problem in the long run. Do not be fooled; the options available for load balancing and assuring availability of your application in the cloud will affect your application – if not right now, then later.

So let's start with the five most important things you need to ask about load balancing and cloud environments regardless of where they may reside.

#5 DIRECT SERVER RETURN

If you're going to be serving up video or audio – real-time streaming media – you should definitely be interested in whether or not the load balancing solution is capable of supporting ***direct server return (DSR)***. While there are pros and cons to using DSR, for video and audio content it's nearly an untouchable axiom of application delivery that you should enable this capability. DSR basically allows the server to return content ***directly*** to the client without being processed by any intermediary (other than routers/switches/etc… which of course ***need*** to process individual packets). In most load balancing situations the responses from the server are returned via the same path they took to get to the server, notably through the load balancer. DSR allows responses to return outside the path of the load balancer or, if still returning through it, to do so unmolested. In the latter scenario the load balancer basically acts as a simple packet forwarder and does no additional processing on the packets.

The advantage to DSR is that it removes any additional latency imposed by additional processing by intermediaries. Because real-time streaming media is very sensitive to the effects of latency (jitter), DSR is often suggested as a best practice when load balancing servers responsible for serving such content.

Question: Is it supported?

#4 HEALTH CHECKING

One of the ways in which load balancers/application delivery controllers make decisions regarding which server should handle which request is to understand the current status of the application. It's part of being context-aware, and it provides information about the application that is invaluable not just to the load balancing decision but to the overall availability of the application. Health checking allows a load balancing solution to determine whether a server/instance is "available" based on a variety of factors. At the simplest level an ICMP ping can be used to determine whether the ***server*** is available. But that tells it nothing of the ***state*** of the application.

A three-way TCP handshake is the next "step" up the ladder, and this will tell the load balancing solution whether an application is capable of accepting connections, but still tells it nothing of the state of the application. A simple HTTP GET takes it one step further, but what's really necessary is the ability of the load balancing solution to retrieve actual data and ensure it is valid in order to consider an application "available".

As the availability of an application may be (and should be if it is not) one way to determine whether new instances are necessary or not, the ability to determine whether the actual application is available ***and responding appropriately*** are important in keeping costs down in a cloud environment lest instances be launched for no reason or – more dangerously – instances are ***not*** launched when necessary due to an outage or failure.

In an external cloud environment it is important to understand how the infrastructure determines when an application is "available" or "not", based on such monitoring, as the subtle differences in what is actually being monitored/tested can impact application availability.

Question: What determines when an application (instance) is available and responding as expected?

#3 PERSISTENCE

Persistence is one of the most important facets of load balancing that every application developer, architect, and network professional needs to understand. Nearly every application today makes heavy use of application sessions to maintain state, but not every application utilizes a shared database model for its session management. If you're using standard application or web server session features to manage state in your application, you will need to understand whether the load balancing solutions available supports persistence and how that persistence is implemented.

Persistence basically ensures that once a user has been "assigned" a server/instance that all subsequent requests go to that same server/instance in order to preserve access to the application session. Persistence can be based on just about anything depending on the load balancing solution available, but most commonly takes the form of either ***source ip address*** or ***cookie-based.*** In the case of the former there's very little for you to do, though you should be somewhat concerned over the use of such a rudimentary method of enabling persistence as it is quite possible – probable, in fact – that many users will be sharing the same source IP address based on NAT and masquerading at the edge of corporate and shared networks.

If the persistence is cookie-based then you'll need to understand whether you have the ability to determine what data is used to enable that persistence. For example, many applications used PHPSESSIONID or ASPSESSIONID as it is routine for those environments to ensure that these values are inserted into the HTTP header and are available for such use. But if you can't configure the option yourself, you'll need to understand what values are used for persistence and to ensure your application can support that value in order to match up users with their application state.

Question: How is persistence implemented?

#2 QUIESCING (BLEEDING) CONNECTIONS

Part of the allure of a cloud architecture is the ability to provision resources on-demand. With that comes the assumption that you can also de-provision resources when they are no longer needed. One would further hope this process is automated and based on a policy configurable by the user, but we are still in the early days of cloud so that may be just a goal at this point.

Load balancers and clustering solutions can usually be told to begin quiescing (bleeding off) connections. This means that they stop distributing requests to the specified servers/instances but allow existing users to continue using the application until they are finished. It basically takes a server/instance out of the “rotation” but keeps it online until all users have finished and the server/instance is no longer needed. At that point either through a manual or automated process the server/instance can be de-provisioned or taken offline. This is often used in traditional data centers to enable maintenance such as patching/upgrades to occur without interrupting application availability. By taking one server/instance at a time offline the other servers/instances remain in service, serving up requests. In an on-demand environment this is of course used to keep costs controlled by only keeping the instances necessary for current capacity online.

What you need to understand is whether that process is manual, i.e. you need to push a button to begin the process of bleeding off connections, or automated. If the latter, then you’ll need to ask about what variables you can use to create a policy to trigger the process. Variables might be number of total connections, requests, users, or bandwidth. It could also, if the load balancing solution is “smart enough” include application performance (response time) or even time of day variables.

Question: How do connections quiesce (bleed) off – manually or automatically based on thresholds?

#1 FAILOVER

We talk a lot about the cloud as a means to scale applications, but we don’t very often mention *availability.* Availability usually means there needs to be in place some sort of “failover” mechanism, in case an application or server fails. Applications crash, hardware fails, these things happen. What should not happen, however, is that the application becomes unavailable because of these types of inevitable problems. If one instance suddenly becomes unavailable, what happens?

That’s the question you need to ask. If there is more than one instance running at that time, then any load balancing solution worth its salt will direct subsequent requests to the remaining available instances. But if there are no other instances running, what happens? If the provisioning process is manual, you may need to push a button and wait for the new instance to come online. If the provisioning process is not manual, then you need to understand how long it will take for the automated system to bring a new instance online, and perhaps ask about the ability to serve up customized “apology” pages that reassure visitors that the site will return shortly.

Question: What kind of failover options are available (if any)?

THERE ARE NO STUPID QUESTIONS

Folks seem to talk and write as if cloud computing relieves IT staff (customers) of the need to understand the infrastructure and architecture of the environments in which applications will be deployed. Because there is an increasingly symbiotic relationship between applications and its infrastructure – both network and application network – this fallacy needs to be exposed for the falsehood it is. It is *more* important today, with cloud computing, than it ever has been for all of IT – application, network, and security – to understand the infrastructure and how it works together to deliver applications.

That means there are *no* stupid questions when it comes to cloud computing infrastructure. There are certainly other questions you can – and should – ask a potential provider or vendor in order to make the right decision about where to deploy your applications. Because when it comes down to it it’s *your* application and your customers, partners, and users are not going to be calling/e-mailing/tweeting the cloud provider; they’re going to be gunning for *you* if things don’t work as expected.

Getting the answers to these five questions will provide a better understanding of how your application will handle unexpected failures, allow you to plan appropriately for maintenance/upgrades/patches, and formulate the proper policies for dealing with the nuances of a load balanced application environment.

Don't just ask about product/vendor and hope that will answer your questions. Sure, your cloud provider may be using F5 or another advanced application delivery platform, but that doesn't mean that they're utilizing the product in a way that would offer the features you need to ensure your application is always available. So dig deeper and ask questions. It's your application, it's your responsibility, no matter where it ends up running.

- And the Killer App for Private Cloud Computing Is…
- Not All Virtual Servers are Created Equal
- Infrastructure 2.0: The Feedback Loop Must Include Applications
- Cloud Computing: Is your cloud sticky? It should be
- The Disadvantages of DSR (Direct Server Return)
- Cloud Computing: Vertical Scalability is Still Your Problem
- Server Virtualization versus Server Virtualization

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