Cloud: Impact of DNS on Performance

Lori MacVittie, 2013-13-03

#webperf #devops Developers and operations must work together to mitigate the impact of hybrid architectures on application performance

One of the ramifications of relying on off-premise cloud infrastructure is that you're necessarily stuck with some of the idiosyncrasies that come with it. For example, it's not your network, and thus topologically-related identifiers such as host names and IP address are not within your purview. But you certainly aren't going to ask your customers to visit "host111-east-virginia-zone3-subnet5.cloudproivder.com". At least not if you want them visit, you won't.

Luckily, you control your own DNS destiny, so you'll just CNAME that crazy long host name provided by the provider to be something more catchy and inline with your branding, say, "coolappz.com".

While certainly more appealing to everyone (easy to remember, fits better on a bumper sticker and on branded swag) it does have a downside: double the latency.

You see, CNAME lookups require two distinct DNS queries to resolve - the first retrieves the ultra-ugly-long host name, the second resolves the ultra-ugly-long host name into an IP address that can actually be used by the browser to connect.

So that's double the lookup, double the roundtrips, double the latency.

Of course, no web page comprises just one host. That would be so 90s and this, this is the 21st century! This is Web 2.0, the age of integration and interconnection and inter-everything. And if the services upon which you rely to build that web app are using CNAMEs, too, well... I hope you like math cause you're going to be added up some roundtrips and latency for a while.

The point here is not to scare you off of hybrid architectures due to the potential impact on performance, but rather to remind you to keep the impact in the fore. It is important to remember the impact of topology, proximity, and the technology in general on the overall performance of your web applications.

A Google Developers article nails down where DNS latency comes from quite well:

> There are two components to DNS latency:
>
> - - Latency between the client (user) and DNS resolving server. In most cases this is largely due to the usual round-trip time (RTT) constraints in networked systems: geographical distance between client and server machines; network congestion; packet loss and long retransmit delays (one second on average); overloaded servers, denial-of-service attacks and so on.
> -
> - - Latency between resolving servers and other nameservers. This source of latency is caused primarily by the following factors:
>   - - Cache misses. If a response cannot be served from a resolver's cache, but requires recursively querying other nameservers, the added network latency is considerable, especially if the authoritative servers are geographically remote.
>   - - Underprovisioning. If DNS resolvers are overloaded, they must queue DNS resolution requests and responses, and may begin dropping and retransmitting packets.
>   - - Malicious traffic. Even if a DNS service is overprovisioned, DoS traffic can place undue load on the servers. Similarly, Kaminsky-style attacks can involve flooding resolvers with queries that are guaranteed to bypass the cache and require outgoing requests for resolution.
>   -
>
> -- Introduction: causes and mitigations of DNS latency

Interestingly, Google is arguing *for* public DNS services, even though this may in fact contribute to location-induced DNS latency, particularly for custom domains for which the authoritative zone is served by relatively few number of DNS servers, most of which are geographically located far from the majority of users. Intercontinental latency is still very much problematic.

Catchpoint, a web performance monitoring service, mentions this in its exhaustive list of the ways in which DNS impacts performance:

> **Exotic Domains**: be careful with the exotic domain names, .ly, .tv… these domains have authoritative servers that are often far away from you end user ISPs. The records will have almost always 2 day TTL, however you never know when someone will be impacted because the query has to go to the authoritative servers and they fail. Example ".ly", 2 authoritative servers are in Libya, 2 in the US, and 1 in the Netherlands.

So when we go connecting clouds and data centers, we need to be concerned with where and how domains are being disseminated, sharded, and resolved. We need to more carefully consider how we are referencing content and whether or not the performance boosts we get from some techniques (such as domain sharding) are being offset by the impact of double the latency from the need to resolve those extra hosts.

We need to examine *that* in the context of other contributing factors, such as TTL (time to live). If the time to live is long enough, then perhaps the initial hit from the extra lookup required to resolve a CNAME isn't going to matter over the life of the session. If we're looking at supporting a stateless API in which sessions don't really exist, then the second lookup may indeed be problematic, but only if the calls are generally spread out over a time interval that is greater than the TTL.

It's a balancing act, where understanding how application network services contribute to the performance of applications is critical to pushing the right buttons and twisting the right knobs will alleviate performance issues that can damage adoption and growth of the web applications that are key to business.

You're Not Off The Hook, Developers

So often it's the case that applications are written with a specific behavior in mind and it is left to devops to figure out how to mitigate these kinds of potential performance issues. But it is just as important for developers to understand how the application network services contribute to performance because sometimes, all it takes is for the application to be "tweaked' with respect to an update interval or use of a different host name to generate a significant improvement in performance. It is increasingly difficult for - and sometimes even impossible - for operations to make adjustments in the infrastructure, particularly in hybrid environments where infrastructure services are black-box and off-limits.

Thus, it is of growing importance that developers and operations work together to map the interaction of applications with application network services such that each group can make appropriate modifications and configuration changes that serve to improve the overall performance of the application, no matter where it might be deployed.

As more and more organizations adopt hybrid, distributed applications that span geographies in addition to environments, this level of cooperation and collaboration will be key to managing web application performance issues.

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