Using F5 BIG-IP to Eliminate Tiers When Scaling Apps in Containers

Lori MacVittie, 2018-02-08

Or is that 'tears' of frustration? ¯_(ツ)_/¯ Perhaps it's both.

There is a relationship between network and application architectures. Usually we like to talk about how changes and shifts in application architectures impact the network – both in terms of solutions and its architecture. But the converse is also true; the architecture and behavior of the network can have a rather dramatic impact on applications and their architecture.

To wit, one of the reasons we're seeing decomposition of monoliths into many microservices *now* instead of during the SOA years is because of the network. Or to be more precise, because of the speed of the network. Back in the days of SOA there were limitations imposed by the nature of networking (and the laws of physics) that made composing a single response from more than three or four services impossible. Okay, not impossible, but undesirable due to the latency incurred by each call.

Today, we've got faster, fatter networks and an order of magnitude quicker compute that makes that decomposition feasible. Making it even easier is the nature of the containers that are so often paired with microservices like coffee with donuts. Because "the network" between two services that need to communicate is often a virtual construct (packets never actually leave the host server and thus do not suffer the latency incurred to be 'put on the wire') the number of services that can be invoked to respond to a single request is much higher than we could reasonably achieve in the days of SOA.

That does not, however, mean that we should ignore how many connections and hops we have to traverse to respond to a request or the impact architecture has on application performance. Because even though compute is faster and the pipes are fatter, operational overhead is still a thing we have to deal with. A thing that still impedes performance. One of the easiest ways to deal with operational overhead (and improve performance) is to reduce the complexity by eliminating (unnecessary) tiers in an architecture.

Reducing Complexity of Container Environments

Most container deployments are going to use some type of load balancer inside the cluster to manage the scale of microservices/apps inside the container environment. That's because they're often tasked with doing L7 routing of APIs (that's ingress control) and native load balancing constructs are based on **iptables**, which of course doesn't speak HTTP – the language of L7 routing . So there's a bunch of L7 load balancers inside the cluster container.

Now, most deployments are also going to want load balancing *outside* the cluster to get external traffic to the right load balancer *inside* the cluster. To do that, they use plain-old load balancing to distribute requests to the right load balancer inside.

We'll call the load balancer outside BIG-IP and the one inside the cluster the *internal lb*. Because it's my blog, that's why.

The problem is that the number of *internal lb* instances fluctuates (sometimes dramatically). Each time there's a change, the BIG-IP needs to know. Traditionally, this has been a manual operation, requiring the BIG-IP owner to go out and modify the pool by hand. That's frustrating for dev and DevOps, and tedious for NetOps. In other words, no one wants to do it this way.

That's why solutions like the F5 Container Connector exist. F5 Container Connector is a container-native service that integrate with the container orchestrator and observes the environment. When there is a change that impacts an ***internal lb,*** it triggers a process to update BIG-IP. This means that as demand ebbs and flows, the BIG-IP is automatically kept up to date and able to appropriately distribute requests to an active, healthy ***internal lb.*** No manual modification necessary.

This two-tiered scaling architecture has the advantage of providing a convenient inbound location (the BIG-IP) at which SSL/TLS can be terminated, providing measurable performance improvements. Nice.

But why stop there? BIG-IP is capable of providing L7 routing. If you're employing the services of F5 Container Connector, you can realize further performance gains (and lower operational overhead) by eliminating the ***internal lb*** completely. Really. BIG-IP can act as an ingress controller for Kubernetes and Red Hat OpenShift.

By moving the ingress responsibility to BIG-IP, you eliminate an entire tier of scale (the ***internal lb),*** which immediately improves performance. Because the ***external lb*** is an F5 BIG-IP, you can further deploy security-focused application services like an advanced WAF with bot defense at the point of contact rather than inside the container cluster (or not at all).

Clash of Ops Clans

As containers push more frequently into production (and they will) there is going to be a need for increased collaboration between DevOps and NetOps in order to implement these kinds of improvements and ensure the scale, speed, and security of apps. It's not just about pushing buttons and self-service pipelines, after all. Architecture is a critical component that will need to be designed with the input of both DevOps and NetOps, lest we ignore opportunities to improve things like application performance.

Because application performance is a team sport. It is impacted by code (AppDev), by the platform the code is deployed on (DevOps), by the network architecture and by the application services used to secure and scale the app (NetOps). Employing architectural optimization by eliminating tiers when possible makes good operational and business sense. But it requires the participation of all the players on the team.

So order some pizza and beer, and bring DevOps and NetOps together and start talking. Find out if you, too, can improve the performance of apps by shedding unnecessary tiers in your container environment.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | wwww.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