Cloud Computing Makes Servers Obsolete

Lori MacVittie, 2009-31-07

The concept of a server needs to go the way of the dodo

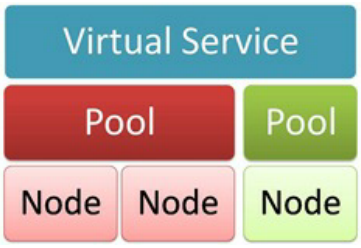
One of the reasons I enjoy Twitter is that quite frequently – if you're following the right people – you'll see a tweet that is absolutely profound despite its simplicity and the constraints placed upon the author.

Recently we were having a mini-discussion on Twitter regarding the definition of availability that elicited just such a golden nugget from botchagalupe: *"Apps designed for a cloud should remove the 'server' concept."*

First, I really like the use of the article "a" in reference to cloud as it speaks to *all* models of cloud: private, public, external, internal, and hybrid. Second, the simplicity of such a statement obscures just how profound the concept is and the long-term effects of cloud and virtualization on how we view architecture and data center design.

THINK RESOURCES, NOT SERVERS

Load balancers, a.k.a. application delivery controllers, have long since discarded traditional terminology such as "servers" and used jargon more germane to the industry like *pool, farm, node,* and *cluster.* Such terminology is, it turns out, perfectly suited to cloud and virtualized environments because it reflects the elimination of tight coupling between an application or service and a physical server.

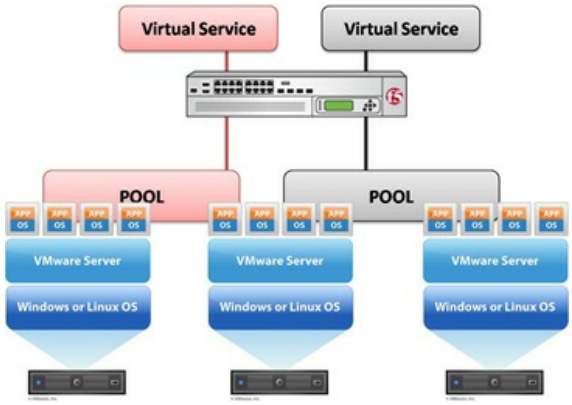


Rather than referring to a *server*, we ought to be referring to a *resource* or a *node.* I'm partial to node, but that's because I'm coming at this from an application delivery perspective. *Resource*, in terms of cloud computing, might be more palatable and accurate to purists but then again we aren't quite at the point where we're actually provisioning compute resources. Regardless, moving away from using the term "server" has the effect of removing the emphasis *off* the physical – or even virtual – server and onto the application or service being delivered. It also better represents the environment as applications really are more akin to *nodes* in a distributed system in a cloud.

In a cloud environment, of course, the server is irrelevant; it's almost assumed in much the same way as organizations have long considered the network to just "be there" when it's needed. That doesn't mean that servers aren't important; they are. After all, without them, where would you deploy applications and what would supply the actual compute resources necessary to execute them – and their virtual containers. It's just that they aren't, in terms of application delivery, really all that much of a factor. That's not a bad thing, as a tightly coupled architecture can be detrimental to designing and implementing a dynamic on-demand infrastructure where applications – not servers – are the focus of attention.

This is a good thing for application and context-aware infrastructure but bad for infrastructure that is still tied to the notion of a server and, as is the wont of network-oriented devices, tied to IP addresses. And that's really the biggest danger in a server-oriented data center: the tendency to tie application to server to IP address. This is dangerous because it inhibits the ability to move an application or take advantage of idle resources when increases in capacity demands must be addressed. It is difficult – and operationally expensive – to decouple an application from a server and an associated IP address in order to move it or clone it on another server and rebind it to a completely different IP address. Server virtualization solves part of this problem by abstracting the application environment from the physical hardware, and network server virtualization solves the rest of the problem by abstracting the notion of pools of resources from the virtual service.

The missing piece in the puzzle is the ability to decouple the application from that associated IP address which, in an IP-based world, is both harder and easier than it sounds. There absolutely must be an IP address, but it can't be static or tied permanently to an application or service.



INFRASTRUCTURE 2.0 to the RESCUE

A dynamic infrastructure is able to remain fluid; its understanding of a pool is that it comprises *nodes*, and those nodes are specifically applications regardless of whether they reside on virtual machines or on physical platforms. A dynamic infrastructure is able to modify its understanding of a *pool* of applications or resources as those nodes change status: from available to unavailable, and vice-versa. The make-up of a "pool" may change from minute to minute or hour to hour or day to day, but the abstraction offered by the application delivery controller assures customers and users that of availability of their applications because they are abstracted from the implementation, in a very SOA-like fashion.

Along with application resources moving from one server to another comes, undoubtedly, a change in IP address as well. This is because there are no guarantees that a new application resource will even be on the same sub-network let alone network – an increasingly possible scenario when you consider cloud bursting, cloud balancing, and other hybrid architectures. What is necessary is the ability to instrument either the application or the hypervisor – or both – with instructions that communicate its current IP address to the infrastructure for which it may be relevant. Application and network security, secure remote access, application acceleration and load balancing infrastructure all need to know the IP address because, well, under the covers every network is still IP-based. The ability to integrate and collaborate with those applications and hypervisors and orchestration systems that make a cloud environment actually work is what makes Infrastructure 2.0 a key component to a successful cloud-based initiative. Without the ability to communicate changes and inevitably for the infrastructure to take decisive action based on those changes none of the "magic" of cloud and dynamic architectures can actually happen.

But notice that there are no *servers* in this equation. There is an application, there is a hypervisor (platform), and an IP address. The server is required, of course, but only in the sense that it's required to provide a physical location in "meat space" on which the application can actually execute. It's a resource provider, and nothing more. The infrastructure – network and application – does not need to even acknowledge its existence because it *must be* irrelevant if emerging data center models are to be as fluid as possible.

In fact, if we're ever to get to the point where we cloud computing truly is about on-demand provisioning of raw compute resources across grids instead of on-demand provisioning of applications across servers we are going to have to start removing the notion of a "server" from our vocabulary.

As you should never put off until tomorrow what you can do today, let's start right now and focus on the application as the central entity to our architectures, not the physical server.

- Cloud Balancing, Cloud Bursting, and Intercloud
- Architects Need to Better Leverage Virtualization
- Dynamic Infrastructure: The Cloud within the Cloud
- The days of IP-based management are numbered
- Managing Virtual Infrastructure Requires an Application Centric Approach
- Server Virtualization versus Server Virtualization

- The Context-Aware Cloud

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