Not all application requests are created equal

Lori MacVittie, 2009-17-03

ArsTechnica has an interesting little article on what Windows Azure is and is not. During the course of discussion with Steven Martin, Microsoft's senior director of Developer Platform Product Management, a fascinating – or disturbing in my opinion – statement was made:

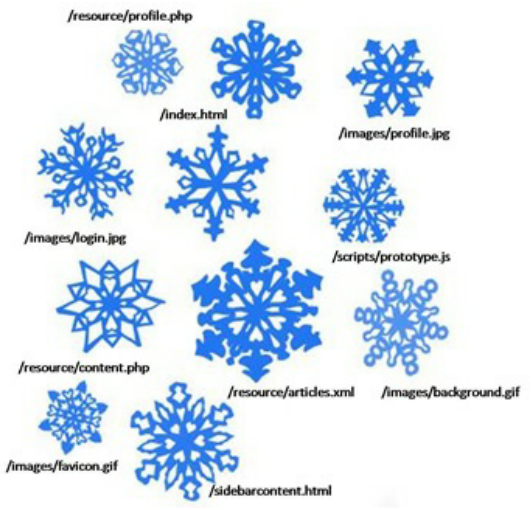
> *There is a distinction between the hosting world and the cloud world that Martin wanted to underline. Whereas hosting means simply the purchase of space under certain conditions (as opposed to buying the actual hardware), the cloud completely hides all issues of clustering and/or load balancing, and it offers an entirely virtualized instance that takes care of all your application's needs.* [emphasis added]

The reason this is disturbing is because not all application requests are created equal and therefore should not necessarily be handled in the same way by a "clustering and/or load balancing solution". But that's exactly what hiding clustering and/or load balancing ends up doing. While it's nice that the nitty-gritty details are obscured in the cloud from developers and, in most cases today, the administrators as well, the lack of control over *how* application requests are distributed actually makes the cloud and its automatic scalability (elasticity) *less* effective.

To understand why you need a bit of background regarding industry standard load balancing algorithms. In the beginning there was Round Robin, an algorithm that is completely application agnostic and simply distributes request based on a list of servers, one after the other. If there are five servers in a pool/farm/cluster, then each one gets a turn. It's an egalitarian algorithm that treats all servers and all requests the same. Round Robin achieves availability, but often at the cost of application performance.

When application performance became an issue we got new algorithms like Least Connections and Fastest Response Time. These algorithms tried to take into account the load on the servers in the pool/farm/cluster before making a decision, and could therefore better improve utilization such that application performance started getting better. But these algorithms only consider the *server* and its load, and don't take into consideration the actual request itself.

And therein lies the problem, for not all requests are created equal. A request for an image requires X processing on a server and Y memory and is usually consistent across time and users. But a request that actually invokes application logic and perhaps executes a database query is variable in its processing time and memory utilization. Some may take longer than others, and require more memory than others. Each request is a unique snowflake whose characteristics are determined by user, by resource, and by the conditions that exist at the time it was made.



It turns out the problem is that in order to effectively determine how to load balance requests in a way that optimizes utilization on servers *and* offers the best application performance you actually have to understand the request. That epiphany gave rise to layer 7 load balancing and the ability to exact finer-grained control over load balancing. Between understanding the request and digging deeper into the server – understanding CPU utilization, memory, network capacity – load balancers were suddenly very effective at distributing load in a way that made sense on a *per request* basis. The result was better architectures, better performing applications, and better overall utilization of the resources available.

Now comes the cloud and its “we hide all the dirty infrastructure details from you” mantra. The problem with this approach is simple: a generic load balancing algorithm is not the most effective method of distributing load across servers, but a cloud provider is not prescient and therefore has no idea what algorithm might be best for your application. Therefore the provider has very little choice in which algorithm is used for load balancing and therefore any choice made will certainly provide availability, but will likely not be the most effective for your specific application.

So while it may sound nice that all the dirty details of load balancing and clustering is “taken care of for you” in the cloud, it’s actually doing you and your application a disservice. Hiding the load balancing and/or clustering capabilities of the cloud, in this case Azure, from the developer is not necessarily the bonus Martin portrays it to be. The ability to control how requests are distributed is just as important in the cloud as it is in your own data center. As Gartner analyst Daryl Plummer points out, underutilizing resources in the cloud, as may happen when using simplistic load balancing algorithms, can be as expensive as running your own data center and may negatively impact application performance. Without some input into the configuration of load balancers and other relevant infrastructure, there isn’t much you can do about that, either, but start up another instance and hope that horizontal scalability will improve performance – at the expense of your budget.

Remember that when someone else makes decisions for you that you are necessarily giving up control. That’s not always a bad thing. But it’s important for you to understand what you are giving up before you hand over the reins. So do your research. You may not have direct control, but you can ask about the “clustering and/or load balancing” provided and understand what affect that may – or may not – have on the performance of your application and the effectiveness of the utilization of the resources for which you are paying.

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