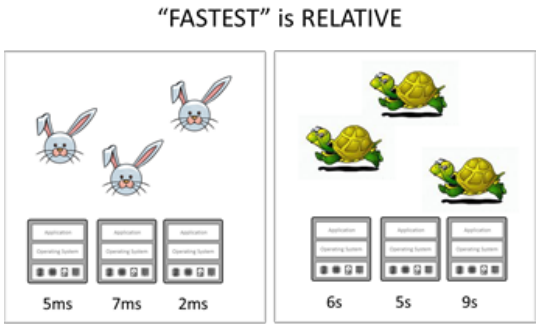
Back to Basics: The Theory of (Performance) Relativity

Lori MacVittie, 2012-14-11

#webperf #ado Choice of load balancing algorithms is critical to ensuring consistent and acceptable performance



One of the primary reasons folks use a Load balancer is scalability with a secondary driver of maintaining performance. We all know the data exists to prove that "seconds matter" and current users of the web have itchy fingers, ready to head for the competition the microsecond they experience any kind of delay.

Similarly, we know that **productivity is inherently tied to performance**. With more and more critical business functions "webified", the longer it takes to load a page the longer the delay a customer or help desk service representative experiences, reducing the number of calls or customers that can be serviced in any given measurable period.

So performance is paramount, I see no reason to persuade you further to come to that conclusion.

Ensuring performance then is a vital operational directive. One of the ways operations tries to meet that objective is through load balancing. Distributing load ensures available and can be used to offset any latency introduced by increasing capacity (again, I don't think there's anyone who'll argue against the premise that load and performance degradation are inherently tied together).

But just adding a load balancing service isn't enough. The algorithm used to distribute load will invariably impact performance – for better or for worse.

Consider the industry standard "fastest response time" algorithm. This algorithm distributes load based on the historical performance of each instance in the pool (farm). On the surface, this seems like a good choice. After all, what you want is the fastest response time, so why not base load balancing decisions on the metric against which you are going to be measured?

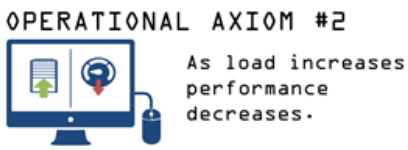
The answer is simple: "fastest" is relative. With very light load on a pool of, say, three servers, "fastest" might mean sub-second responses. But as load increases and performance decreases, "fastest" might start creeping up into the seconds – if not more. Sure, you're still automagically choosing the fastest of the three servers, but "fastest" is absolutely relative to the measurements of all three servers.

Thus, "fastest response time" is probably a poor choice if one of your goals is measured in response time to the ultimate customer – unless you combine it with an upper connection limit.

HOW TO USE "FASTEST RESPONSE TIME" ALGORITHMS CORRECTLY

One of the negatives of adopting a cloud computing paradigm with a nearly religious-like zeal is that you buy into the notion that utilization is the most important metric in the data center. You simply do not want to be wasting CPU cycles, because that means you're inefficient and not leveraging cloud to its fullest potential.

Well, horse-puckey. The reality is that 100% utilization and consistently well-performing applications do not go hand in hand. Period. You can have one, but not the other. You're going to have to choose which is more important a measurement – fast applications or full utilization.



In the six years I spent load testing everything from web applications to web application firewalls to load balancers to XML gateways one axiom always, *always*, remained true:

As load increases performance decreases.

You're welcome to test and retest and retest again to prove that wrong, but good luck. I've never seen performance increase or even stay the same as utilization approaches 100%.

Now, once you accept that reality you can use it to your advantage. You ***know*** that performance is going to decrease as load increases, you just don't know at what point the degradation will become unacceptable to your users. So you need to test to find that breaking point. You want to stress the application and measure the degradation, noting the number of concurrent connections at which performance starts to degrade into unacceptable territory. **That is your connection limit.**

Keep track of that limit (for the application, specifically, because not all applications will have the same limits). When you configure your load balancing service you can now select **fastest response time** but you also need to input hard connection limits on a per-instance basis. This prevents each instance from passing through the load-performance confluence that causes end-users to start calling up the help desk or sighing "the computer is slow" while on the phone with ***their*** customers.

This means testing. Not once, not twice, but at least three runs. Make sure you've found the right load-performance confluence point and write it down. On your hand, in permanent marker.

While cloud computing and virtualization have certainly simplified load balancing services in terms of deployment, it's still up to you to figure out the right settings and configuration options to ensure that your applications are performing with the appropriate levels of "fast".

- Back to Basics: Load balancing Virtualized Applications
- Performance in the Cloud: Business Jitter is Bad
- The “All of the Above” Approach to Improving Application Performance
- The Three Axioms of Application Delivery
- Face the facts: Cloud performance isn't always stable
- Data Center Feng Shui: Architecting for Predictable Performance
- A Formula for Quantifying Productivity of Web Applications
- Enterprise Apps are Not Written for Speed

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