Top 3 Lessons We Can Learn from Amazon's Outage

Lori MacVittie, 2008-10-06

Everyone's now familiar with last week's Amazon outage, it's been all over the web. Amazon is still not detailing what went wrong, but lots of folks are speculating on the reasons for the failure. Alistair Croll over at gigaom had a nice recap of the recent Amazon outage, including a list of facts and what we can deduce them, and the rumor mill is, of course, churning away with fantastic stories of what went wrong.

Rather than add to the speculation on the reasons behind Amazon's outage - because we may never know the truth - it's a good time to look at the outage from the perspective of what lessons we can learn about application delivery and building a resilient web architecture.

> **3. Implement GSLB (Global Server Load Balancing)**
>
> In the fact of disaster - natural or otherwise - a secondary data center physically separated from the main data center can be used to enable business continuity as well as disaster recovery. Using a global server load balancer, if the primary data center is down - regardless of the reasons - requests can be automatically directed to a second data center until the primary returns to service. This results in less disruption of service to customers and remote employees alike.
>
> **2. Configure "Apology" pages**
>
> Apology pages are simply content displayed when no servers can be reached. Apology pages are similar to "maintenance" pages often displayed when a site is unavailable due to maintenance, upgrades, or problems. The ability to display apology or maintenance pages is available in all major load balancers (application delivery controllers) today and are fully configurable. The pages need not even be "real" pages, as most application delivery controllers are capable of returning content directly.
>
> **1. Use intelligent health checks** (a.k.a. Advanced health monitoring)
>
> All load balancers are capable of "health checks" on servers. This is basic feature that uses ICMP or a TCP connection to verify that a server in the pool (farm) is available. The problem with basic health checks is that they only verify network connectivity (ICMP) or transport layer connectivity (TCP) or, if you're lucky, application layer connectivity (HTTP).
>
> What's really necessary is to be able to not only verify that the server is reachable, available, and responding but that it is also responding with the appropriate content. Just because a web servers returns a 200 OK status does not mean the content is correct. It's quite possible that one or more dependent services responded incorrectly, leaving the page with "most of its content" but potentially with errors embedded. This is often the case with data-driven web sites, where SQL errors are not properly caught and end up returning errors *inside* otherwise appropriate content.
>
> Advanced health monitoring in application delivery controllers provides organizations with the means by which both servers and applications can be verified as being "available". By not only verifying IP and TCP availability, organizations can ensure that should an "HTTP 1.1" message be returned by a server that the application delivery controller can (1) detect the problem and (2) remove the affected server from the pool of available servers. If enough of the servers are responding incorrectly, the application delivery controller would discover that no servers are available and then respond with an "apology page".

"Load balancing" has come a long way, baby, since the days of simple round-robin DNS. It has evolved into application delivery, and the products providing application delivery services are no longer dumb endpoints on the network, they're sophisticated, intelligent devices capable of not only providing basic load balancing services, but augmenting the entire application delivery process by ensuring that applications are always available through intelligence and advanced feature sets.

Application delivery controllers aren't your daddy's load balancers, so take advantage of capabilities like advanced health monitoring and dynamic server selection to build a resilient, failsafe architecture that responds to your customers no matter what might be going on behind the data center doors.

Imbibing: Coffee

Resources on BIG-IP and its advanced health checking capabilities:

- HTTP Monitoring with POST
- Monitoring & Management Forum on DevCentral
- Passive Monitoring - Maintaining Performance and Health (white paper)