October 28th, 2009 at 9:52 am

**[How to Ruin Your Weekend and Other Hazards of Mis-Configured HA](http://www.mooselogic.com/blog/how-to-ruin-your-weekend-and-other-hazards-of-mis-configured-ha%22%20%5Co%20%22Permanent%20Link%20to%20How%20to%20Ruin%20Your%20Weekend%20and%20Other%20Hazards%20of%20Mis-Configured%20HA)**

» by [Steve Parlee](http://www.mooselogic.com/blog/author/sparlee/%22%20%5Co%20%22Posts%20by%20Steve%20Parlee) *[(View Profile](http://www.mooselogic.com/blog/?page_id=7&user=Steve%20Parlee))* in: [Citrix](http://www.mooselogic.com/blog/category/citrix%22%20%5Co%20%22View%20all%20posts%20in%20Citrix),[High Availability](http://www.mooselogic.com/blog/category/high-availability%22%20%5Co%20%22View%20all%20posts%20in%20High%20Availability),[Virtualization](http://www.mooselogic.com/blog/category/virtualization-resources%22%20%5Co%20%22View%20all%20posts%20in%20Virtualization),[XenServer](http://www.mooselogic.com/blog/category/xenserver%22%20%5Co%20%22View%20all%20posts%20in%20XenServer)

The Level 1 HA (High Availability) feature that comes with Citrix Essentials for XenServer may be one of the best ways to crash your whole virtual infrastructure if you don’t understand how it works and don’t design in an appropriate level of redundancy. This of course will lead to hours of down time, unhappy management, possible data loss, and lots of extra work for you (most likely on a weekend).

**The basics -**
HA is designed to monitor the XenServer virtualization environment. When HA is enabled, the administrator can specify which virtual machines (VMs) need to be automatically restarted if the host server they’re running on should fail. If there is a failure of a host server, HA should then automatically restart its designated guest VMs on another host in the XenServer “resource pool.” Note that the HA function does not “live migrate” the guest VMs, because when a host fails the VMs on that host also fail. Rather, it selects another host server and restarts the VMs on that host. For all of this to happen correctly, Citrix’s HA requires two things to be true at all times:

1. Each XenServer must be able to communicate with its peers in the pool.
2. Each XenServer in the pool requires access at all times to the HA heartbeat disk, which is shared by all the XenServers in the pool.

If either of these two items is not true for any given XenServer in the pool, that server will “fence.” The short definition of “fencing” is that the XenServer *suspects* – although it’s not absolutely sure – that it is experiencing some kind of failure, so to protect against possible data corruption it shuts itself down – essentially sacrificing itself to protect the data – until a human comes along and sorts things out. If the fenced server is in a correctly configured HA pool, guest VMs that were configured for HA restart will be restarted on a surviving XenServer.

**Considerations -**
So… you have two XenServers all set up and all your VMs configured just the way you like them, and you decide to turn on HA. Everything appears to be working until one of the hosts suffers a failure and goes off line. (Murphy’s Law says this will happen on a Saturday evening right before your BBQ party is starting.) With HA enabled, you would expect, based on the whole “High Availability” concept, that everything would be OK. Critical VMs should get restarted on the other host and you should be able to deal with the failed host on Monday.

Oh, but wait, remember HA rule #1? The XenServer host that is still running suddenly does not have any peers to talk to. It no longer knows whether or not it’s healthy so, in the interest of protecting your data from corruption, it does what it’s designed to do – it fences, and now both of your XenServers are down. They may try to reboot, but you are now in an endless loop of fencing, and to get it resolved, you’re going to have to know how to use the “xe host-emergency-ha-disable force=true” command to resolve your problems. (And if you don’t understand that last sentence, you’re in for a long weekend.)

This results in a situation that we in IT refer to as “not good,” with a chance of “career altering,” and you’re going to miss your BBQ party.

Here’s another scenario that will spoil your party: What if both XenServers are actually healthy, and all the virtual servers are up and functioning, but the network link for the management communications between the XenServers fails? Again, each XenServer would think it was stranded from the pool and fence itself in an attempt to correct the issue. With both servers fencing, this would again create an endless loop of server fencing. In essence, one server would start to come back online and would still not see the other XenServer and would fence again, and so on, and so on.

So for those reasons a two-XenServer pool ***cannot*** successfully run HA! Just don’t do it – even though you can configure HA on a two-server pool the result can be disastrous and ruin your weekend…not to mention your next performance review.



HA In a Two-Server Pool - Just Don't Do It!

Well, what about HA in a three node XenServer pool? Based upon the previously described scenarios, you now have a valid “pool,” in which HA will function. So you configure and enable HA, and when you test the HA functionality by killing one of the XenServers, everything works like it is supposed to. The guest VMs are restarted on the surviving XenServer hosts and you’re happy that everything is working correctly.

But here is another “gotcha!” If you have only one Ethernet interface per XenServer assigned to management, and they’re all plugged into one switch, what happens if the management link fails because a NIC fails – or even worse, the switch fails? If it’s just a NIC in one server, then that XenServer will fence – not too bad but still not what you want. If you were using a different set of NICs (as you always should) for the guest VMs to communicate with the rest of the world, then the guests on that server were probably up and working just fine until the server fenced. Sure, the critical ones will restart on the remaining servers, but you’ve lost a third of the resources in your pool unnecessarily.

Now let’s consider what would happen if the switch should fail and you had only single management ports on each XenServer all plugged into just that one switch. If this happens, it may be time to dust off the old resume, because you have just lost your entire XenServer pool. Why? Because when the switch went down, *all* the XenServers lost communication with one another, and each assumed that, because it was suddenly isolated from the pool, it must be experiencing some kind of failure. Therefore *the whole pool fenced.*



Non-Redundant Management Links - Don't Do This Either!

**Conclusions -**
Citrix’s HA does not work in a two host pool, period. With a pool of three or more XenServers you’ll be OK ***if*** you design the infrastructure correctly so that there is no single point of failure in your peer communications. How? Simply by bonding together two NICs, dedicating them to the management communication function, and then splitting the bonded pairs between two separate Ethernet switches. That way you’re protected against both a NIC failure and a switch failure.

But you’re not out of the woods yet! Don’t forget HA rule #2 – servers need to see the HA heartbeat disk. This is equally important, and you must consider the topology of that side of the network (iSCSI, Fiber, etc.) and be sure it is also redundant. And if you’re using iSCSI multi-pathing (e.g., with a pair of mirrored DataCore iSCSI SAN nodes), be sure to manually bump up the HA timeout interval so that if one of the SAN nodes should fail, the multi-pathing function has time to fail over to the other node before the XenServers all conclude that the HA heartbeat disk is gone – otherwise, again, they will all fence. Our testing indicates that a two minute timeout appears to have an adequate margin of safety. The default setting of (oops – the default is actually 30 seconds) is definitely too short. Unfortunately, this setting does not appear to be persistent, so if you turn HA off and then back on, you’ll need to manually reset the timeout interval again. (This is probably a job for Workflow Studio, but we just haven’t had time to work through the process yet.)

**NO Single Points of Failure**
HA will do a fine job of protecting you, if you build the network correctly. So make sure you’ve built in enough redundancy that you have no single point of failure, and enjoy your BBQ.



The Right Way to Build an HA Environment

P.S.: If you can’t justify more than two XenServers, but you still have one or more critical guests that need to be highly available, there is a solution: [Marathon Technologies’ everRun VM](http://www.marathontechnologies.com/high_availability_xenserver.html). But that’s another post for another day.

Tags: [Citrix](http://www.mooselogic.com/blog/tag/citrix), [Citrix Essentials](http://www.mooselogic.com/blog/tag/citrix-essentials), [High Availability](http://www.mooselogic.com/blog/tag/high-availability), [Server Virtualization](http://www.mooselogic.com/blog/tag/server-virtualization), [Virtualization](http://www.mooselogic.com/blog/tag/virtualization-resources), [XenServer](http://www.mooselogic.com/blog/tag/xenserver)