VMware vSphere 6.0 High Availability

deepdive

Duncan Epping

Table of Contents

Introduction	1.1
Disclaimer	1.2
About the author	1.3
Introduction to HA	1.4
Components of HA	1.5
Fundamental Concepts	1.6
Restarting Virtual Machines	1.7
Virtual SAN and Virtual Volumes specifics	1.8
Adding resiliency to HA	1.9
Admission Control	1.10
VM and Application Monitoring	1.11
vSphere HA and ...	1.12
Stretched Clusters	1.13
Advanced Settings	1.14
Summarizing	1.15
Changelog	1.16

VMware vSphere 6.0 Update 2 HA Deepdive

Like many of you I am constantly trying to explore new ways to share content with the rest of the world. Over the course of the last decade I have done this in many different formats, some of them were easy to do and others not so much. Books always fell in that last category, which is a shame as I have always enjoyed writing them.

I wanted to explore the different options there are to create content and share it in different ways, without the need to re-do formatting and waste a lot of time on things I do not want to waste time on. After an afternoon of reading and researching GitBook popped up. It looked like an interesting platform / solution that would allow me to create content both online and offline, push and pull it to and from a repository and build both a static website from it as well as publish it in a variety of different formats.

Let it be clear that this is a trial, and this may or may not result in a follow up. I am starting with the vSphere High Availability content as that is what I am most familiar with and as such it will be easiest to update.

A special thanks goes out to everyone who has contributed in any shape or form to this project. First of all Frank Denneman, my co-author on the vSphere Clustering Deepdive publications and the designer of all the great diagrams you find throughout this publication. Of course also: Doug Baer for editing the content in the past and my technical conscious: Keith Farkas, Cormac Hogan, Manoj Krishnan.

For offline reading, feel free to download this publication in any of the following formats: PDF - ePub - Mobi. The source of this publication is stored on both Gitbook as well as Github. Feel free to submit/contribute where possible and needed. Note that it is also possible to leave feedback on the content by simply clicking on the "+" on the right side of the paragraph you want to comment on (hover over it with your mouse).

vSphere 6.0 U2 HA Deepdive, book version: 1.1.3.
Book built with GitBook version: 3.1.1.

Thanks for reading, and enjoy!

Duncan Epping
Author / Owner @ Yellow-Bricks.com
Chief Technologist @ VMware

Disclaimer

Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

The author of this publication works for VMware. The opinions expressed here is the author's personal opinion. Content published was not approved in advance by VMware and does not necessarily reflect the views and opinion of VMware. This is the author's book, not a VMware.

Copyrights / Licensing



vSphere HA Deepdive by Duncan Epping is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. What does this mean? This means you can fork, contribute, and (re)distribute the material. As long as appropriate credits are given and it is not used for commercial purposes! More information about the licensing can be found here.

About the Author

Duncan Epping is a Chief Technologist working in the Office of CTO of VMware's Storage and Availability business unit. In that role, he serves as a partner and trusted adviser to VMware’s customers primarily in EMEA. Main responsibilities are ensuring VMware’s future innovations align with essential customer needs and translating customer problems to opportunities. Duncan specializes in Software Defined Storage, hyper-converged infrastructures and business continuity / disaster recovery solutions. He has 1 patent granted and 4 patents pending on the topic of availability, storage and resource management. Duncan is a VMware Certified Design Expert (VCDX007) and the main author and owner of VMware/Virtualization blog Yellow-Bricks.com.

He can be followed on twitter @DuncanYB.

Introduction to vSphere High Availability

Availability has traditionally been one of the most important aspects when providing services. When providing services on a shared platform like VMware vSphere, the impact of downtime exponentially grows as many services run on a single physical machine. As such VMware engineered a feature called VMware vSphere High Availability. VMware vSphere High Availability, hereafter simply referred to as HA, provides a simple and cost effective solution to increase availability for any application running in a virtual machine regardless of its operating system. It is configured using a couple of simple steps through vCenter Server (vCenter) and as such provides a uniform and simple interface. HA enables you to create a cluster out of multiple ESXi hosts. This will allow you to protect virtual machines and their workloads. In the event of a failure of one of the hosts in the cluster, impacted virtual machines are automatically restarted on other ESXi hosts within that same VMware vSphere Cluster (cluster).

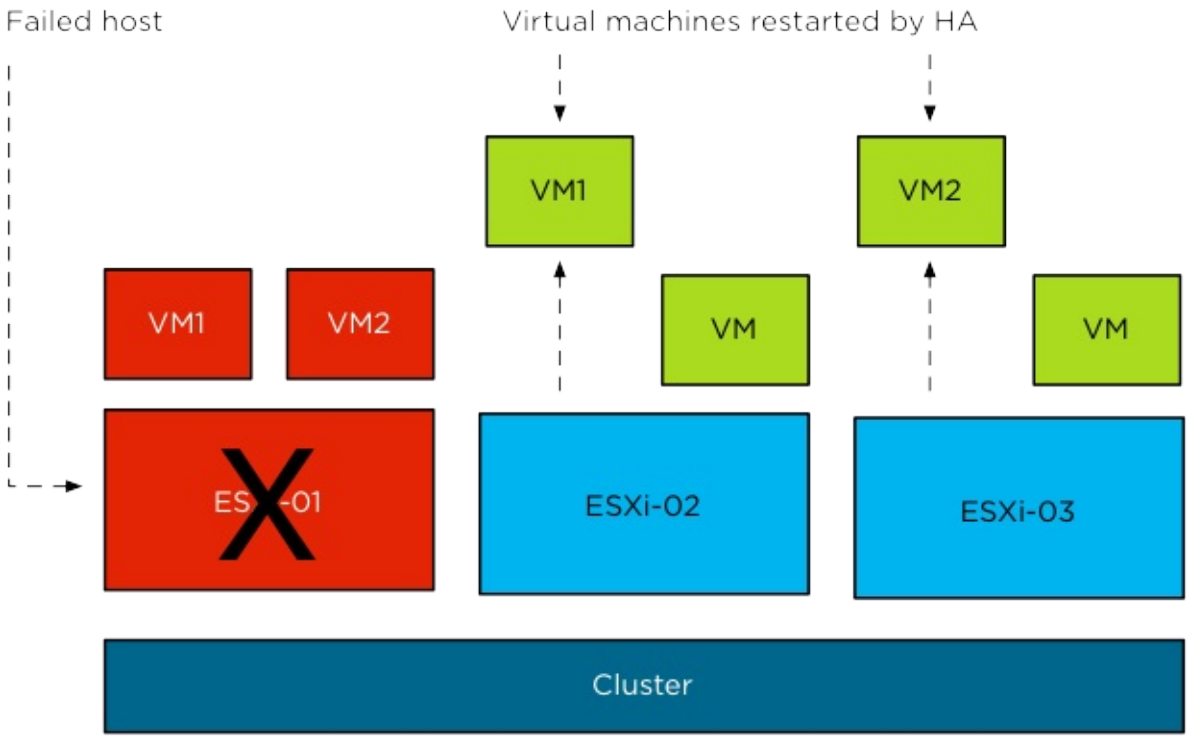


Figure 2 - High Availability in action

On top of that, in the case of a Guest OS level failure, HA can restart the failed Guest OS. This feature is called VM Monitoring, but is sometimes also referred to as VM-HA. This might sound fairly complex but again can be implemented with a single click.

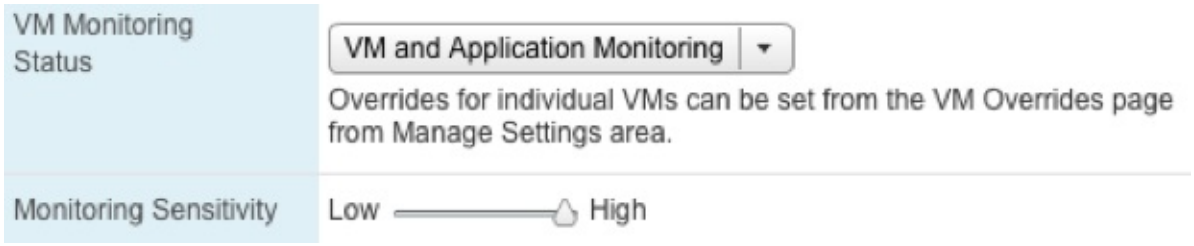


Figure 3 - OS Level HA just a single click away

Unlike many other clustering solutions, HA is a simple solution to implement and literally enabled within 5 clicks. On top of that, HA is widely adopted and used in all situations. However, HA is not a 1:1 replacement for solutions like Microsoft Clustering Services / Windows Server Failover Clustering (WSFC). The main difference between WSFC and HA being that WSFC was designed to protect stateful cluster-aware applications while HA was designed to protect any virtual machine regardless of the type of workload within, but also can be extended to the application layer through the use of VM and Application Monitoring.

In the case of HA, a fail-over incurs downtime as the virtual machine is literally restarted on one of the remaining hosts in the cluster. Whereas MSCS transitions the service to one of the remaining nodes in the cluster when a failure occurs. In contrary to what many believe, WSFC does not guarantee that there is no downtime during a transition. On top of that, your application needs to be cluster-aware and stateful in order to get the most out of this mechanism, which limits the number of workloads that could really benefit from this type of clustering.

One might ask why would you want to use HA when a virtual machine is restarted and service is temporarily lost. The answer is simple; not all virtual machines (or services) need 99.999% uptime. For many services the type of availability HA provides is more than sufficient. On top of that, many applications were never designed to run on top of an WSFC cluster. This means that there is no guarantee of availability or data consistency if an application is clustered with WSFC but is not cluster-aware.

In addition, WSFC clustering can be complex and requires special skills and training. One example is managing patches and updates/upgrades in a WSFC environment; this could even lead to more downtime if not operated correctly and definitely complicates operational procedures. HA however reduces complexity, costs (associated with downtime and MSCS), resource overhead and unplanned downtime for minimal additional costs. It is important to note that HA, contrary to WSFC, does not require any changes to the guest as HA is provided on the hypervisor level. Also, VM Monitoring does not require any additional software or OS modifications except for VMware Tools, which should be installed anyway as a best practice. In case even higher availability is required, VMware also provides a level of

application awareness through Application Monitoring, which has been leveraged by partners like Symantec to enable application level resiliency and could be used by in-house development teams to increase resiliency for their application.

HA has proven itself over and over again and is widely adopted within the industry; if you are not using it today, hopefully you will be convinced after reading this section of the book.

vSphere 6.0

Before we dive into the main constructs of HA and describe all the choices one has to make when configuring HA, we will first briefly touch on what's new in vSphere 6.0 and describe the basic requirements and steps needed to enable HA. This book covers all the released versions of what is known within VMware as "Fault Domain Manager" (FDM) which was introduced with vSphere 5.0. We will call out the differences in behavior in the different versions where applicable, our baseline however is vSphere 6.0.

What's New in 6.0?

Compared to vSphere 5.0 the changes introduced with vSphere 6.0 for HA appear to be minor. However, some of the new functionality will make the life of many of you much easier. Although the list is relatively short, from an engineering point of view many of these things have been an enormous effort as they required change to the deep fundaments of the HA architecture.

- Support for Virtual Volumes – With Virtual Volumes a new type of storage entity is introduced in vSphere 6.0. This has also resulted in some changes in the HA architecture to accommodate for this new way of storing virtual machines
- Support for Virtual SAN – This was actually introduced with vSphere 5.5, but as it is new to many of you and led to changes in the architecture we decided to include it in this update
- VM Component Protection – This allows HA to respond to a scenario where the connection to the virtual machine's datastore is impacted temporarily or permanently
  - HA "Response for Datastore with All Paths Down"
  - HA "Response for Datastore with Permanent Device Loss"
- Increased host scale – Cluster limit has grown from 32 to 64 hosts
- Increased VM scale – Cluster limit has grown from 4000 VMs to 8000 VMs per cluster
- Secure RPC – Secures the VM/App monitoring channel
- Full IPv6 support
- Registration of "HA Disabled" VMs on hosts after failure

What is required for HA to Work?

Each feature or product has very specific requirements and HA is no different. Knowing the requirements of HA is part of the basics we have to cover before diving into some of the more complex concepts. For those who are completely new to HA, we will also show you how to configure it.

Prerequisites

Before enabling HA it is highly recommend validating that the environment meets all the prerequisites. We have also included recommendations from an infrastructure perspective that will enhance resiliency.

Requirements:

- Minimum of two ESXi hosts
- Minimum of 5GB memory per host to install ESXi and enable HA
- VMware vCenter Server
- Shared Storage for virtual machines
- Pingable gateway or other reliable address

Recommendation:

- Redundant Management Network (not a requirement, but highly recommended)
- 8GB of memory or more per host
- Multiple shared datastores

Firewall Requirements

The following table contains the ports that are used by HA for communication. If your environment contains firewalls external to the host, ensure these ports are opened for HA to function correctly. HA will open the required ports on the ESX or ESXi firewall.

Port	Protocol	Direction
8182	UDP	Inbound
8182	TCP	Inbound
8182	UDP	Outbound
8182	TCP	Outbound

Configuring vSphere High Availability

HA can be configured with the default settings within a couple of clicks. The following steps will show you how to create a cluster and enable HA, including VM Monitoring, using the vSphere Web Client. Each of the settings and the design decisions associated with these steps will be described in more depth in the following chapters.

1. Click “Hosts & Clusters” under Inventories on the Home tab.
2. Right-click the Datacenter in the Inventory tree and click New Cluster.
3. Give the new cluster an appropriate name. We recommend at a minimum including the location of the cluster and a sequence number ie. ams-hadrs-001.
4. Select Turn On vSphere HA.
5. Ensure “Enable host monitoring” and “Enable admission control” is selected.
6. Select “Percentage of cluster resources…” under Policy and specify a percentage.
7. Enable VM Monitoring Status by selecting “VM and Application Monitoring”.
8. Click “OK” to complete the creation of the cluster.

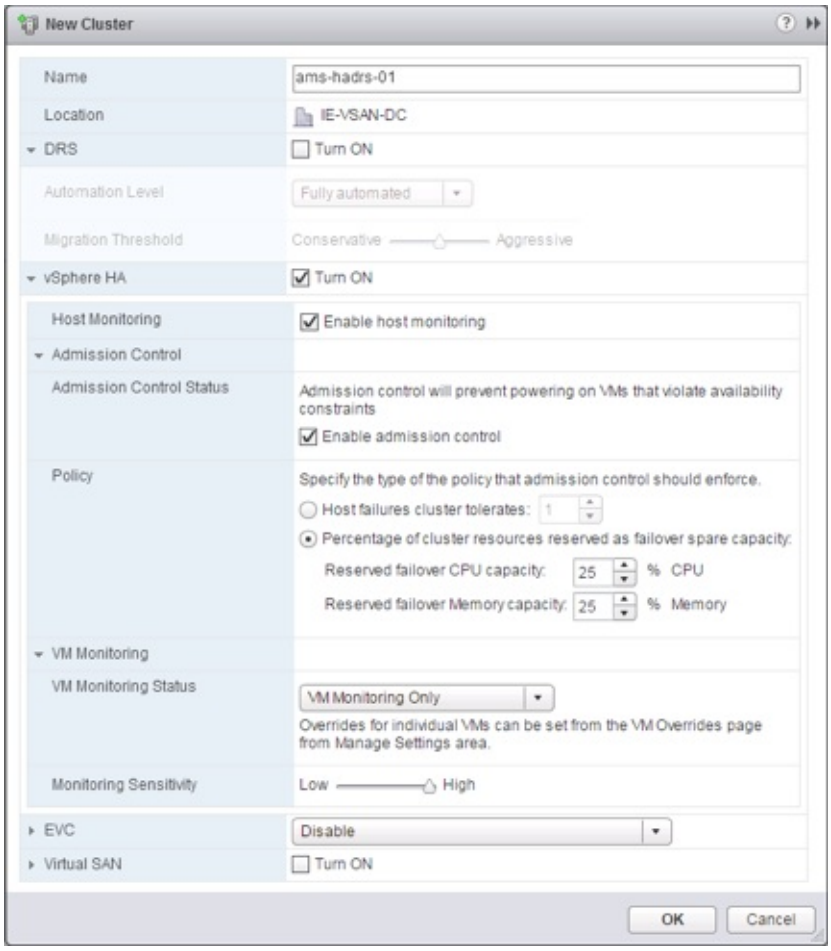


Figure 4 - Ready to complete the New Cluster Wizard

When the HA cluster has been created, the ESXi hosts can be added to the cluster simply by right clicking the host and selecting “Move To”, if they were already added to vCenter, or by right clicking the cluster and selecting “Add Host”.

When an ESXi host is added to the newly-created cluster, the HA agent will be loaded and configured. Once this has completed, HA will enable protection of the workloads running on this ESXi host.

As we have clearly demonstrated, HA is a simple clustering solution that will allow you to protect virtual machines against host failure and operating system failure in literally minutes. Understanding the architecture of HA will enable you to reach that extra 9 when it comes to availability. The following chapters will discuss the architecture and fundamental concepts of HA. We will also discuss all decision-making moments to ensure you will configure HA in such a way that it meets the requirements of your or your customer’s environment.

Components of High Availability

Now that we know what the pre-requisites are and how to configure HA the next steps will be describing which components form HA. Keep in mind that this is still a “high level” overview. There is more under the cover that we will explain in following chapters. The following diagram depicts a two-host cluster and shows the key HA components.

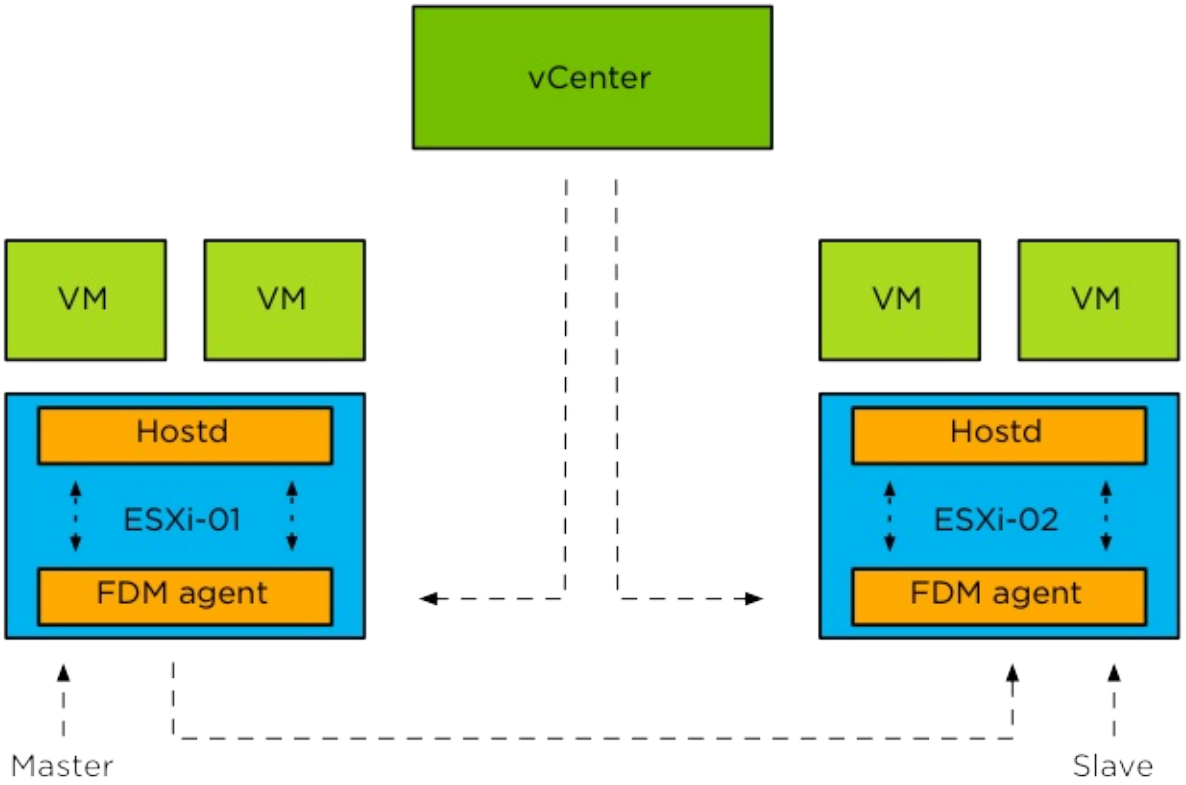


Figure 5 - Components of High Availability

As you can clearly see, there are three major components that form the foundation for HA as of vSphere 6.0:

- FDM
- HOSTD
- vCenter

The first and probably the most important component that forms HA is FDM (Fault Domain Manager). This is the HA agent.

The FDM Agent is responsible for many tasks such as communicating host resource information, virtual machine states and HA properties to other hosts in the cluster. FDM also handles heartbeat mechanisms, virtual machine placement, virtual machine restarts, logging and much more. We are not going to discuss all of this in-depth separately as we feel that this will complicate things too much.

FDM, in our opinion, is one of the most important agents on an ESXi host, when HA is enabled, of course, and we are assuming this is the case. The engineers recognized this importance and added an extra level of resiliency to HA. FDM uses a single-process agent. However, FDM spawns a watchdog process. In the unlikely event of an agent failure, the watchdog functionality will pick up on this and restart the agent to ensure HA functionality remains without anyone ever noticing it failed. The agent is also resilient to network interruptions and “all paths down” (APD) conditions. Inter-host communication automatically uses another communication path (if the host is configured with redundant management networks) in the case of a network failure.

HA has no dependency on DNS as it works with IP addresses only. This is one of the major improvements that FDM brought. This does not mean that ESXi hosts need to be registered with their IP addresses in vCenter; it is still a best practice to register ESXi hosts by its fully qualified domain name

(FQDN) in vCenter. Although HA does not depend on DNS, remember that other services may depend on it. On top of that, monitoring and troubleshooting will be much easier when hosts are correctly registered within vCenter and have a valid FQDN.

> **Basic design principle**: Although HA is not dependent on DNS, it is still recommended to register the hosts with their FQDN for ease of operations/management.

vSphere HA also has a standardized logging mechanism, where a single log file has been created for all operational log messages; it is called fdm.log. This log file is stored under /var/log/ as depicted in Figure 5.

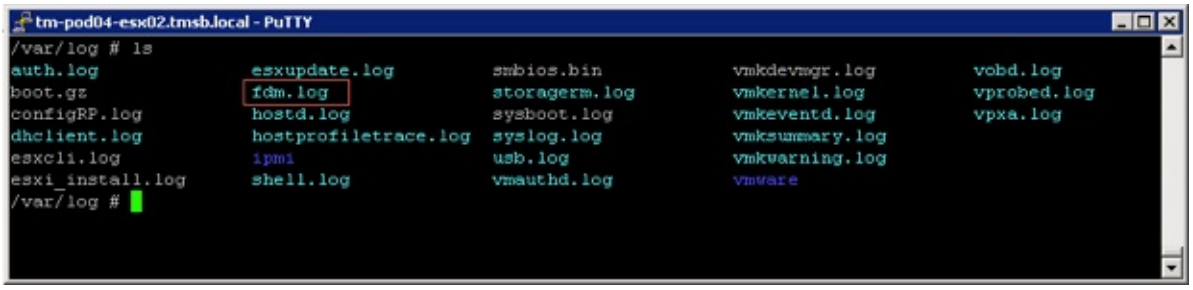


Figure 6 - HA log file

> **Basic design principle**: Ensure syslog is correctly configured and log files are offloaded to a safe location to offer the possibility of performing a root cause analysis in case disaster strikes.

HOSTD Agent

One of the most crucial agents on a host is HOSTD. This agent is responsible for many of the tasks we take for granted like powering on virtual machines. FDM talks directly to HOSTD and vCenter, so it is not dependent on VPXA, like in previous releases. This is, of course, to avoid any unnecessary overhead and dependencies, making HA more reliable than ever before and enabling HA to respond faster to power-on requests. That ultimately results in higher VM uptime.

When, for whatever reason, HOSTD is unavailable or not yet running after a restart, the host will not participate in any FDM-related processes. FDM relies on HOSTD for information about the virtual machines that are registered to the host, and manages the virtual machines using HOSTD APIs. In short, FDM is dependent on HOSTD and if HOSTD is not operational, FDM halts all functions and waits for HOSTD to become operational.

vCenter

That brings us to our final component, the vCenter Server. vCenter is the core of every vSphere Cluster and is responsible for many tasks these days. For our purposes, the following are the most important and the ones we will discuss in more detail:

- Deploying and configuring HA Agents
- Communication of cluster configuration changes
- Protection of virtual machines

vCenter is responsible for pushing out the FDM agent to the ESXi hosts when applicable. The push of these agents is done in parallel to allow for faster deployment and configuration of multiple hosts in a cluster. vCenter is also responsible for communicating configuration changes in the cluster to the host which is elected as the master. We will discuss this concept of master and slaves in the following chapter. Examples of configuration changes are modification or addition of an advanced setting or the introduction of a new host into the cluster.

HA leverages vCenter to retrieve information about the status of virtual machines and, of course, vCenter is used to display the protection status (Figure 6) of virtual machines. (What “virtual machine protection” actually means will be discussed in chapter 3.) On top of that, vCenter is responsible for the protection and unprotection of virtual machines. This not only

applies to user initiated power-offs or power-ons of virtual machines, but also in the case where an ESXi host is disconnected from vCenter at which point vCenter will request the master HA agent to unprotect the affected virtual machines.



Figure 7 - Virtual machine protection state

Although HA is configured by vCenter and exchanges virtual machine state information with HA, vCenter is not involved when HA responds to failure. It is comforting to know that in case of a host failure containing the virtualized vCenter Server, HA takes care of the failure and restarts the vCenter Server on another host, including all other configured virtual machines from that failed host.

There is a corner case scenario with regards to vCenter failure: if the ESXi hosts are so called “stateless hosts” and Distributed vSwitches are used for the management network, virtual machine restarts will not be attempted until vCenter is restarted. For stateless environments, vCenter and Auto Deploy availability is key as the ESXi hosts literally depend on them.

If vCenter is unavailable, it will not be possible to make changes to the configuration of the cluster. vCenter is the source of truth for the set of virtual machines that are protected, the cluster configuration, the virtual machine-to-host compatibility information, and the host membership. So, while HA, by design, will respond to failures without vCenter, HA relies on vCenter to be available to configure or monitor the cluster.

When a virtual vCenter Server, or the vCenter Server Appliance, has been implemented, we recommend setting the correct HA restart priorities for it. Although vCenter Server is not required to restart virtual machines, there are multiple components that rely on vCenter and, as such, a speedy recovery is desired. When configuring your vCenter virtual machine with a

high priority for restarts, remember to include all services on which your vCenter server depends for a successful restart: DNS, MS AD and MS SQL (or any other database server you are using).

> **Basic design principles:**
>
> 1. In stateless environments, ensure vCenter and Auto Deploy are highly available as recovery time of your virtual machines might be dependent on them.
> 2. Understand the impact of virtualizing vCenter. Ensure it has high priority for restarts and ensure that services which vCenter Server depends on are available: DNS, AD and database.

Fundamental Concepts

Now that you know about the components of HA, it is time to start talking about some of the fundamental concepts of HA clusters:

- Master / Slave agents
- Heartbeating
- Isolated vs Network partitioned
- Virtual Machine Protection
- Component Protection

Everyone who has implemented vSphere knows that multiple hosts can be configured into a cluster. A cluster can best be seen as a collection of resources. These resources can be carved up with the use of vSphere Distributed Resource Scheduler (DRS) into separate pools of resources or used to increase availability by enabling HA.

The HA architecture introduces the concept of master and slave HA agents. Except during network partitions, which are discussed later, there is only one master HA agent in a cluster. Any agent can serve as a master, and all others are considered its slaves. A master agent is in charge of monitoring the health of virtual machines for which it is responsible and restarting any that fail. The slaves are responsible for forwarding information to the master agent and restarting any virtual machines at the direction of the master. The HA agent, regardless of its role as master or slave, also implements the VM/App monitoring feature which allows it to restart virtual machines in the case of an Operating System or restart services in the case of an application failure.

Master Agent

As stated, one of the primary tasks of the master is to keep track of the state of the virtual machines it is responsible for and to take action when appropriate. In a normal situation there is only a single master in a cluster. We will discuss the scenario where multiple masters can exist in a single cluster in one of the following sections, but for now let’s talk about a cluster with a single master. A master will claim responsibility for a virtual machine by taking “ownership” of the datastore on which the virtual machine’s configuration file is stored.

> **Basic design principle:** To maximize the chance of restarting virtual machines after a failure we recommend masking datastores on a cluster basis. Although sharing of datastores across clusters will work, it will increase complexity from an administrative perspective.

That is not all, of course. The HA master is also responsible for exchanging state information with vCenter. This means that it will not only receive but also send information to vCenter when required. The HA master is also the host that initiates the restart of virtual machines when a host has failed. You may immediately want to ask what happens when the master is the one that fails, or, more generically, which of the hosts can become the master and when is it elected?

Election

A master is elected by a set of HA agents whenever the agents are not in network contact with a master. A master election thus occurs when HA is first enabled on a cluster and when the host on which the master is running:

- fails,
- becomes network partitioned or isolated,
- is disconnected from vCenter Server,
- is put into maintenance or standby mode,
- or when HA is reconfigured on the host.

The HA master election takes approximately 15 seconds and is conducted using UDP. While HA won't react to failures during the election, once a master is elected, failures detected before and during the election will be handled. The election process is simple but robust. The host that is participating in the election with the greatest number of connected datastores will be elected master. If two or more hosts have the same number of datastores connected, the one with the highest Managed Object Id will be chosen. This however is done lexically; meaning that 99 beats 100 as 9 is larger than 1. For each host, the HA State of the host will be shown on the Summary tab. This includes the role as depicted in screenshot below where the host is a master host.

After a master is elected, each slave that has management network connectivity with it will setup a single secure, encrypted, TCP connection to the master. This secure connection is SSL-based. One thing to stress here though is that slaves do not communicate with each other after the master has been elected unless a re-election of the master needs to take place.

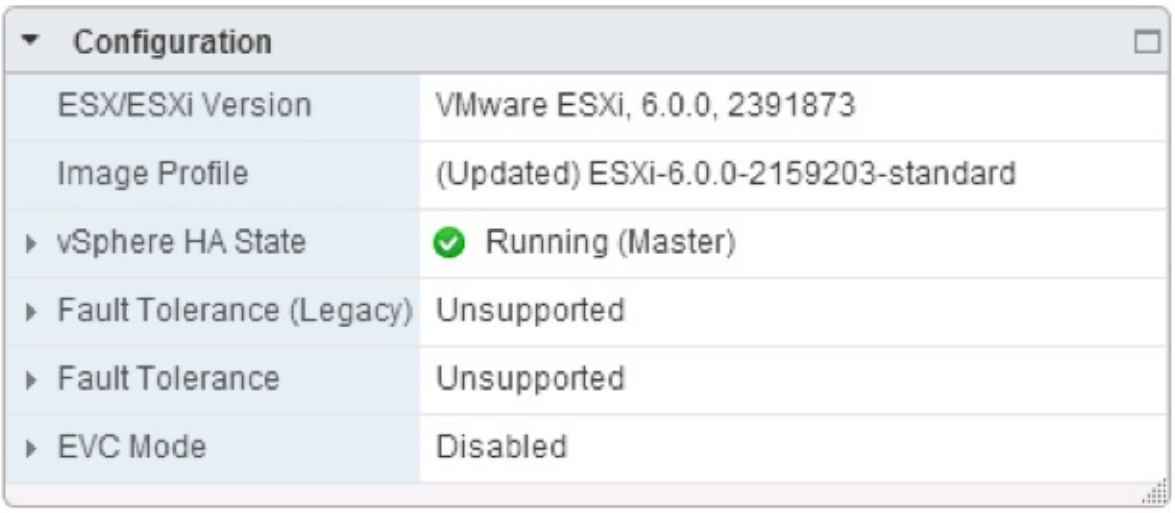


Figure 8 - Master Agent

As stated earlier, when a master is elected it will try to acquire ownership of all of the datastores it can directly access or access by proxying requests to one of the slaves connected to it using the management network. For regular storage architectures it does this by locking a file called “protectedlist” that is stored on the datastores in an existing cluster. The master will also attempt to take ownership of any datastores it discovers along the way, and it will periodically retry any it could not take ownership of previously.

The naming format and location of this file is as follows:

```
/<root of datastore>/.vSphere-HA/<cluster-specific-directory>/protectedlist
```

For those wondering how “cluster-specific-directory” is constructed:

```
<uuid of vCenter Server>-<number part of the MoID of the cluster>-<random 8 char string>-
<name of the host running vCenter Server>
```

The master uses this protectedlist file to store the inventory. It keeps track of which virtual machines are protected by HA. Calling it an inventory might be slightly overstating: it is a list of protected virtual machines and it includes information around virtual machine CPU reservation and memory overhead. The master distributes this inventory across all datastores in use by the virtual machines in the cluster. The next screenshot shows an example of this file on one of the datastores.

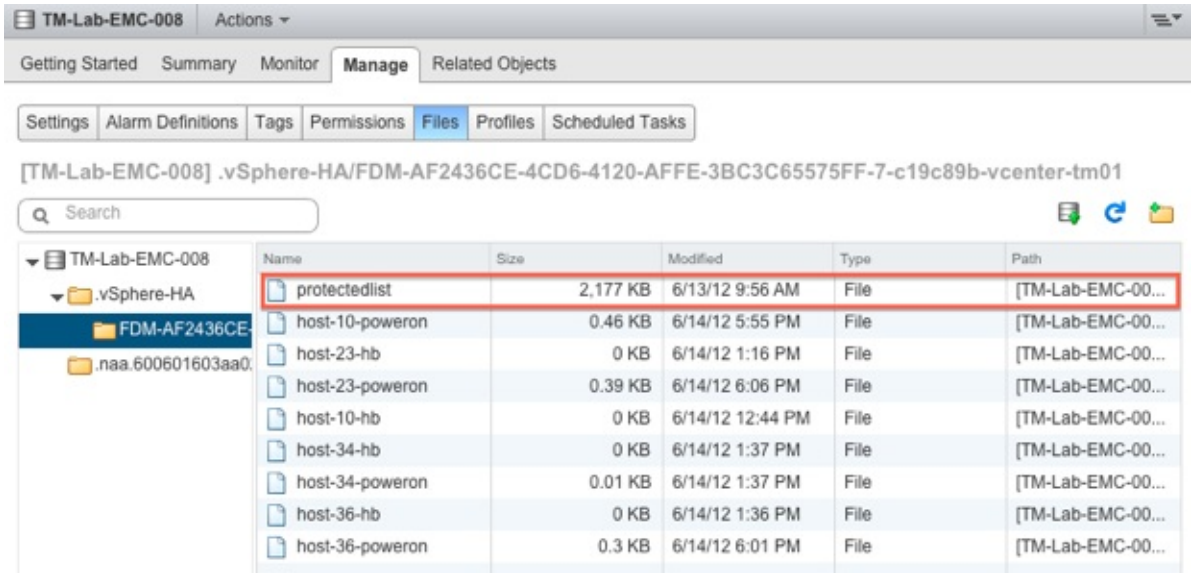


Figure 9 - Protectedlist file

Now that we know the master locks a file on the datastore and that this file stores inventory details, what happens when the master is isolated or fails? If the master fails, the answer is simple: the lock will expire and the new master will relock the file if the datastore is accessible to it.

In the case of isolation, this scenario is slightly different, although the result is similar. The master will release the lock it has on the file on the datastore to ensure that when a new master is elected it can determine the set of virtual machines that are protected by HA by reading the file. If, by any chance, a master should fail right at the moment that it became isolated, the restart of the virtual machines will be delayed until a new master has been elected. In a scenario like this, accuracy and the fact that virtual machines are restarted is more important than a short delay.

Let’s assume for a second that your master has just failed. What will happen and how do the slaves know that the master has failed? HA uses a point-to-point network heartbeat mechanism. If the slaves have received no network heartbeats from the master, the slaves will try to elect a new master. This new master will read the required information and will initiate the restart of the virtual machines within roughly 10 seconds.

Restarting virtual machines is not the only responsibility of the master. It is also responsible for monitoring the state of the slave hosts and reporting this state to vCenter Server. If a slave fails or becomes isolated from the management network, the master will determine which virtual machines must be restarted. When virtual machines need to be restarted, the master is also responsible for determining the placement of those virtual machines. It uses a placement engine that will try to distribute the virtual machines to be restarted evenly across all available hosts.

All of these responsibilities are really important, but without a mechanism to detect a slave has failed, the master would be useless. Just like the slaves receive heartbeats from the master, the master receives heartbeats from the slaves so it knows they are alive.

Slaves

A slave has substantially fewer responsibilities than a master: a slave monitors the state of the virtual machines it is running and informs the master about any changes to this state.

The slave also monitors the health of the master by monitoring heartbeats. If the master becomes unavailable, the slaves initiate and participate in the election process. Last but not least, the slaves send heartbeats to the master so that the master can detect outages. Like the master to slave communication, all slave to master communication is point to point. HA does not use multicast.

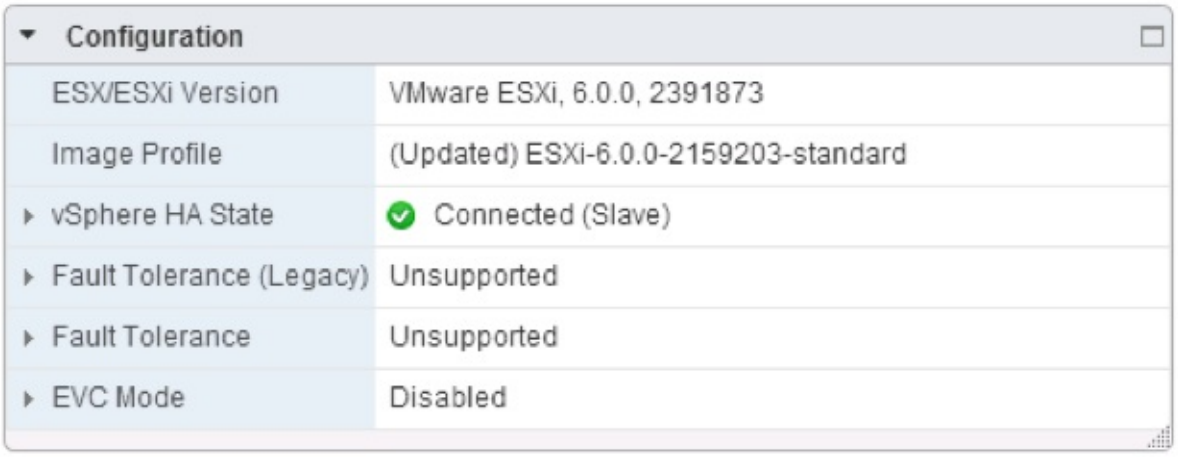


Figure 10 - Slave Agent

Files for both Slave and Master

Before explaining the details it is important to understand that both Virtual SAN and Virtual Volumes have introduced changes to the location and the usage of files. For specifics on these two different storage architectures we refer you to those respective sections in the book.

Both the master and slave use files not only to store state, but also as a communication mechanism. We've already seen the protectedlist file (Figure 8) used by the master to store the list of protected virtual machines. We will now discuss the files that are created by both

the master and the slaves. Remote files are files stored on a shared datastore and local files are files that are stored in a location only directly accessible to that host.

Remote Files

The set of powered on virtual machines is stored in a per-host “poweron” file. It should be noted that, because a master also hosts virtual machines, it also creates a “poweron” file.

The naming scheme for this file is as follows: `host-number-poweron`

Tracking virtual machine power-on state is not the only thing the “poweron” file is used for. This file is also used by the slaves to inform the master that it is isolated from the management network: the top line of the file will either contain a 0 or a 1. A 0 (zero) means not-isolated and a 1 (one) means isolated. The master will inform vCenter about the isolation of the host.

Local Files

As mentioned before, when HA is configured on a host, the host will store specific information about its cluster locally.

```
~ # cd /etc/opt/vmware/fdm/
/etc/opt/vmware/fdm # ls
clusterconfig  fdm.cfg        hostlist       vmmetadata
/etc/opt/vmware/fdm # ls -lah
drwxr-xr-x    1 root     root         512 Jun 14 16:00 .
-r--------T   1 root     root           0 May 13 17:00 .#clusterconfig
-r--------T   1 root     root           0 May 13 17:00 .#hostlist
-r--------T   1 root     root           0 May 13 17:00 .#vmmetadata
drwxr-xr-x    1 root     root         512 Jun 14 11:28 ..
-rw------T    1 root     root         653 Jun 14 11:36 clusterconfig
-rw------T    1 root     root        2.2K May 13 17:00 fdm.cfg
-rw------T    1 root     root        2.6K Jun 14 11:36 hostlist
-rw------T    1 root     root         589 Jun 14 16:00 vmmetadata
/etc/opt/vmware/fdm #
```

Figure 11 - Locally stored files

Each host, including the master, will store data locally. The data that is locally stored is important state information. Namely, the VM-to-host compatibility matrix, cluster configuration, and host membership list. This information is persisted locally on each host. Updates to this information is sent to the master by vCenter and propagated by the master to the slaves. Although we expect that most of you will never touch these files – and we highly recommend against modifying them – we do want to explain how they are used:

- **clusterconfig** This file is not human-readable. It contains the configuration details of the cluster.
- **vmmetadata** This file is not human-readable. It contains the actual compatibility info matrix for every HA protected virtual machine and lists all the hosts with which it is compatible plus a vm/host dictionary
- **fdm.cfg** This file contains the configuration settings around logging. For instance, the level of logging and syslog details are stored in here.
- **hostlist** A list of hosts participating in the cluster, including hostname, IP addresses, MAC addresses and heartbeat datastores.

Heartbeating

We mentioned it a couple of times already in this chapter, and it is an important mechanism that deserves its own section: heartbeating. Heartbeating is the mechanism used by HA to validate whether a host is alive. HA has two different heartbeating mechanisms. These heartbeat mechanisms allows it to determine what has happened to a host when it is no longer responding. Let's discuss traditional network heartbeating first.

Network Heartbeating

Network Heartbeating is used by HA to determine if an ESXi host is alive. Each slave will send a heartbeat to its master and the master sends a heartbeat to each of the slaves, this is a point-to-point communication. These heartbeats are sent by default every second.

When a slave isn't receiving any heartbeats from the master, it will try to determine whether it is Isolated– we will discuss "states" in more detail later on in this chapter.

> **Basic design principle:** Network heartbeating is key for determining the state of a host. Ensure the management network is highly resilient to enable proper state determination.

Datastore Heartbeating

Datastore heartbeating adds an extra level of resiliency and prevents unnecessary restart attempts from occurring as it allows vSphere HA to determine whether a host is isolated from the network or is completely unavailable. How does this work?

Datastore heartbeating enables a master to more determine the state of a host that is not reachable via the management network. The new datastore heartbeat mechanism is used in case the master has lost network connectivity with the slaves. The datastore heartbeat mechanism is then used to validate whether a host has failed or is merely isolated/network

partitioned. Isolation will be validated through the “poweron” file which, as mentioned earlier, will be updated by the host when it is isolated. Without the “poweron” file, there is no way for the master to validate isolation. Let that be clear! Based on the results of checks of both files, the master will determine the appropriate action to take. If the master determines that a host has failed (no datastore heartbeats), the master will restart the failed host’s virtual machines. If the master determines that the slave is Isolated or Partitioned, it will only take action when it is appropriate to take action. With that meaning that the master will only initiate restarts when virtual machines are down or powered down / shut down by a triggered isolation response.

By default, HA selects 2 heartbeat datastores – it will select datastores that are available on all hosts, or as many as possible. Although it is possible to configure an advanced setting (*das.heartbeatDsPerHost*) to allow for more datastores for datastore heartbeating we do not recommend configuring this option as the default should be sufficient for most scenarios, except for stretched cluster environments where it is recommended to have two in each site manually selected.

The selection process gives preference to VMFS over NFS datastores, and seeks to choose datastores that are backed by different LUNs or NFS servers when possible. If desired, you can also select the heartbeat datastores yourself. We, however, recommend letting vCenter deal with this operational “burden” as vCenter uses a selection algorithm to select heartbeat datastores that are presented to all hosts. This however is not a guarantee that vCenter can select datastores which are connected to all hosts. It should be noted that vCenter is not site-aware. In scenarios where hosts are geographically dispersed it is recommend to manually select heartbeat datastores to ensure each site has one site-local heartbeat datastore at minimum.

> **Basic design principle:** In a metro-cluster / geographically dispersed cluster we recommend setting the minimum number of heartbeat datastores to four. It is recommended to manually select site local datastores, two for each site.

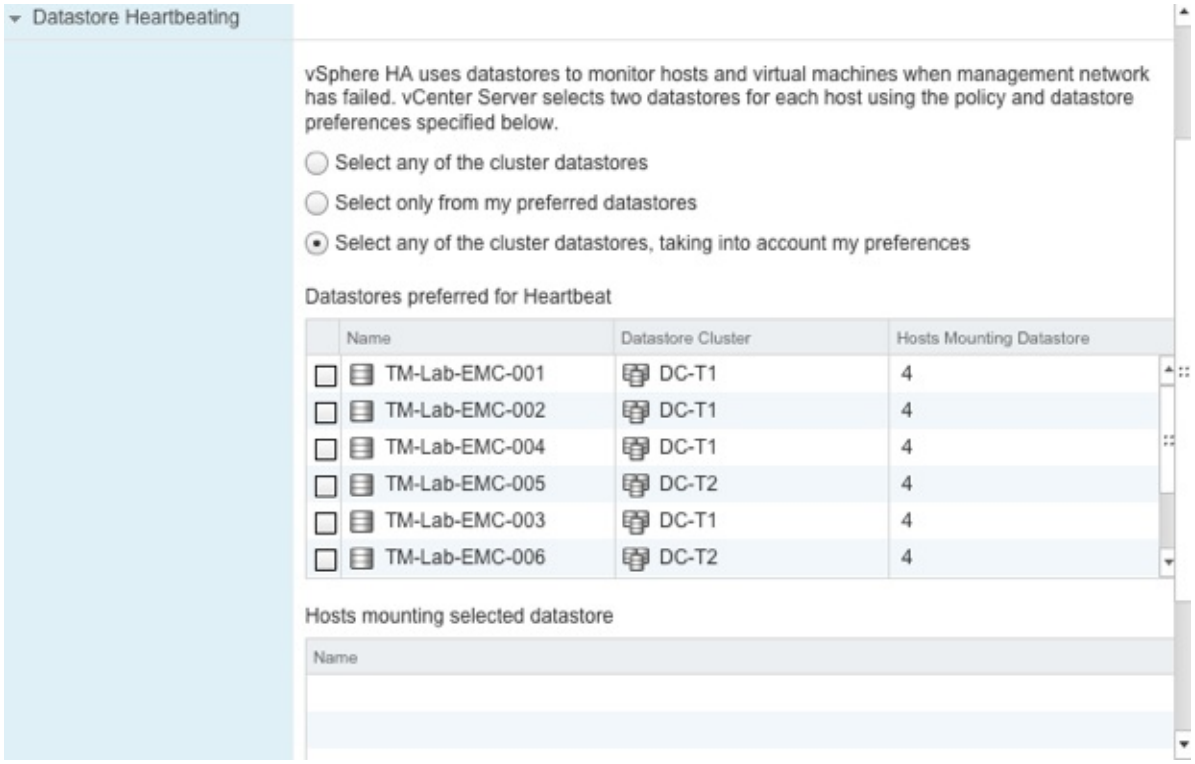
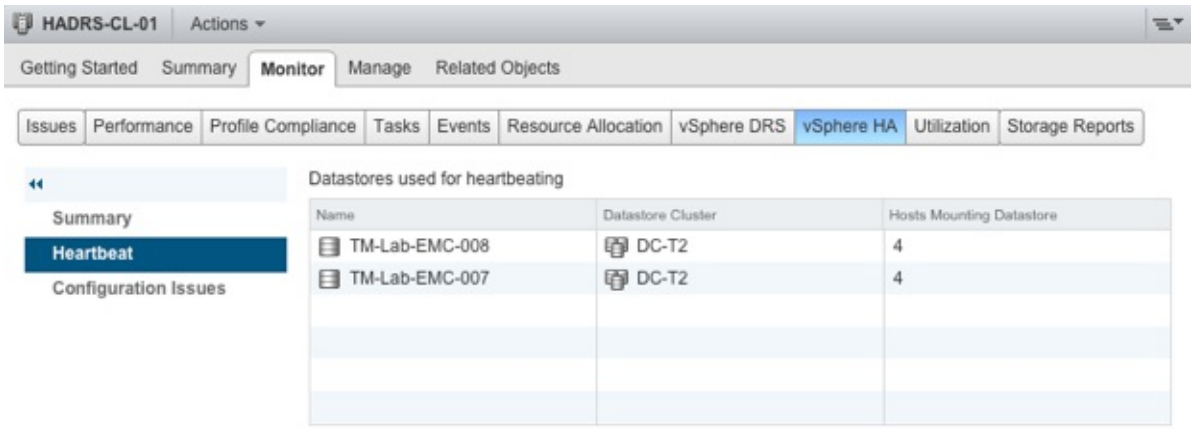


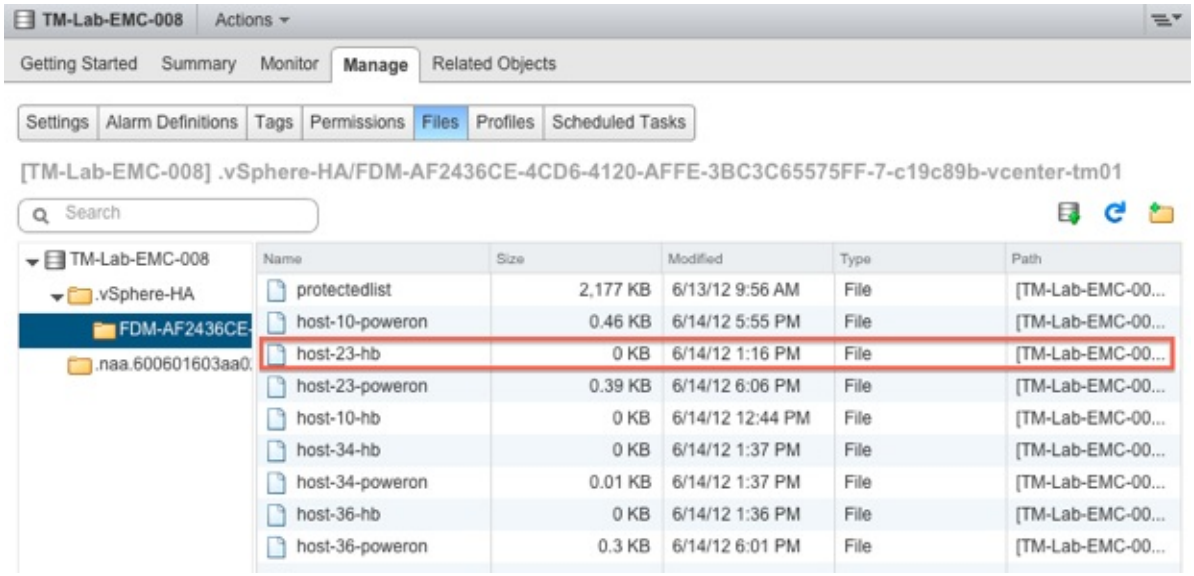
Figure 12 - Selecting the heartbeat datastores

The question now arises: what, exactly, is this datastore heartbeating and which datastore is used for this heartbeating? Let's answer which datastore is used for datastore heartbeating first as we can simply show that with a screenshot, see below. vSphere displays extensive details around the "Cluster Status" on the Cluster's Monitor tab. This for instance shows you which datastores are being used for heartbeating and which hosts are using which specific datastore(s). In addition, it displays how many virtual machines are protected and how many hosts are connected to the master.



In block based storage environments HA leverages an existing VMFS file system mechanism. The datastore heartbeat mechanism uses a so called "heartbeat region" which is updated as long as the file is open. On VMFS datastores, HA will simply check whether the heartbeat region has been updated. In order to update a datastore heartbeat region, a

host needs to have at least one open file on the volume. HA ensures there is at least one file open on this volume by creating a file specifically for datastore heartbeating. In other words, a per-host file is created on the designated heartbeating datastores, as shown below. The naming scheme for this file is as follows: `host-number-hb`.



On NFS datastores, each host will write to its heartbeat file once every 5 seconds, ensuring that the master will be able to check host state. The master will simply validate this by checking that the time-stamp of the file changed.

Realize that in the case of a converged network environment, the effectiveness of datastore heartbeating will vary depending on the type of failure. For instance, a NIC failure could impact both network and datastore heartbeating. If, for whatever reason, the datastore or NFS share becomes unavailable or is removed from the cluster, HA will detect this and select a new datastore or NFS share to use for the heartbeating mechanism.

```
Basic design principle
Datastore heartbeating adds a new level of resiliency but is not the be-all end-all. I
n converged networking environments, the use of datastore heartbeating adds little val
ue due to the fact that a NIC failure may result in both the network and storage becom
ing unavailable.
```

Isolated versus Partitioned

We've already briefly touched on it and it is time to have a closer look. When it comes to network failures there are two different states that exist. What are these exactly and when is a host Partitioned rather than Isolated? Before we will explain this we want to point out that there is the state as reported by the master and the state as observed by an administrator and the characteristics these have.

First, consider the administrator's perspective. Two hosts are considered partitioned if they are operational but cannot reach each other over the management network. Further, a host is isolated if it does not observe any HA management traffic on the management network and it can't ping the configured isolation addresses. It is possible for multiple hosts to be isolated at the same time. We call a set of hosts that are partitioned but can communicate with each other a "management network partition". Network partitions involving more than two partitions are possible but not likely.

Now, consider the HA perspective. When any HA agent is not in network contact with a master, they will elect a new master. So, when a network partition exists, a master election will occur so that a host failure or network isolation within this partition will result in appropriate action on the impacted virtual machine(s). The screenshot below shows possible ways in which an Isolation or a Partition can occur.

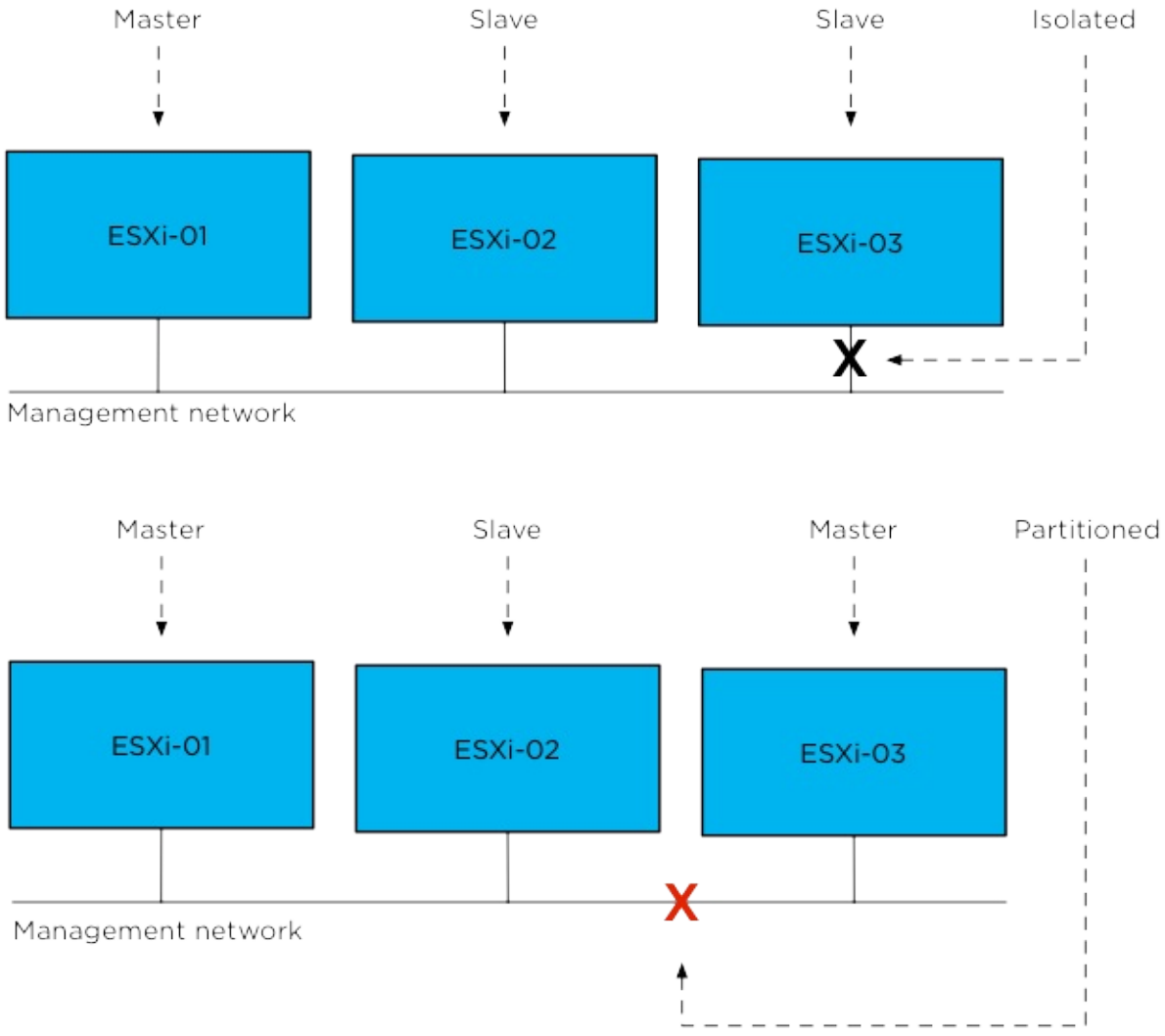


Figure 13 - Validating

If a cluster is partitioned in multiple segments, each partition will elect its own master, meaning that if you have 4 partitions your cluster will have 4 masters. When the network partition is corrected, any of the four masters will take over the role and be responsible for the cluster again. It should be noted that a master could claim responsibility for a virtual machine that lives in a different partition. If this occurs and the virtual machine happens to fail, the master will be notified through the datastore communication mechanism.

In the HA architecture, whether a host is partitioned is determined by the master reporting the condition. So, in the above example, the master on host ESXi-01 will report ESXi-03 and 04 partitioned while the master on host 04 will report 01 and 02 partitioned. When a partition occurs, vCenter reports the perspective of one master.

A master reports a host as partitioned or isolated when it can't communicate with the host over the management network, it can observe the host's datastore heartbeats via the heartbeat datastores. The master cannot alone differentiate between these two states – a host is reported as isolated only if the host informs the master via the datastores that is isolated.

This still leaves the question open how the master differentiates between a Failed, Partitioned, or Isolated host.

When the master stops receiving network heartbeats from a slave, it will check for host "liveness" for the next 15 seconds. Before the host is declared failed, the master will validate if it has actually failed or not by doing additional liveness checks. First, the master will validate if the host is still heartbeating to the datastore. Second, the master will ping the management IP address of the host. If both are negative, the host will be declared Failed. This doesn't necessarily mean the host has PSOD'ed; it could be the network is unavailable, including the storage network, which would make this host Isolated from an administrator's perspective but Failed from an HA perspective. As you can imagine, however, there are a various combinations possible. The following table depicts these combinations including the "state".

State	NetworkHeartbeat	StorageHeartbeat	Host Live-nessPing	Isolation Criteria Met
Running	Yes	N/A	N/A	N/A
Isolated	No	Yes	No	Yes
Partitioned	No	Yes	No	No
Failed	No	No	No	N/A
FDM Agent Down	N/A	N/A	Yes	N/A

HA will trigger an action based on the state of the host. When the host is marked as Failed, a restart of the virtual machines will be initiated. When the host is marked as Isolated, the master might initiate the restarts.

The one thing to keep in mind when it comes to isolation response is that a virtual machine will only be shut down or powered off when the isolated host knows there is a master out there that has taken ownership for the virtual machine or when the isolated host loses access to the home datastore of the virtual machine.

For example, if a host is isolated and runs two virtual machines, stored on separate datastores, the host will validate if it can access each of the home datastores of those virtual machines. If it can, the host will validate whether a master owns these datastores. If no master owns the datastores, the isolation response will not be triggered and restarts will not be initiated. If the host does not have access to the datastore, for instance, during an “All Paths Down” condition, HA will trigger the isolation response to ensure the “original” virtual machine is powered down and will be safely restarted. This to avoid so-called “split-brain” scenarios.

To reiterate, as this is a very important aspect of HA and how it handles network isolations, the remaining hosts in the cluster will only be requested to restart virtual machines when the master has detected that either the host has failed or has become isolated and the isolation response was triggered.

Virtual Machine Protection

Virtual machine protection happens on several layers but is ultimately the responsibility of vCenter. We have explained this briefly but want to expand on it a bit more to make sure everyone understands the dependency on vCenter when it comes to protecting virtual machines. We do want to stress that this only applies to protecting virtual machines; virtual machine restarts in no way require vCenter to be available at the time.

When the state of a virtual machine changes, vCenter will direct the master to enable or disable HA protection for that virtual machine. Protection, however, is only guaranteed when the master has committed the change of state to disk. The reason for this, of course, is that a failure of the master would result in the loss of any state changes that exist only in memory. As pointed out earlier, this state is distributed across the datastores and stored in the “*protectedlist*” file.

When the power state change of a virtual machine has been committed to disk, the master will inform vCenter Server so that the change in status is visible both for the user in vCenter and for other processes like monitoring tools.

To clarify the process, we have created a workflow diagram of the protection of a virtual machine from the point it is powered on through vCenter:

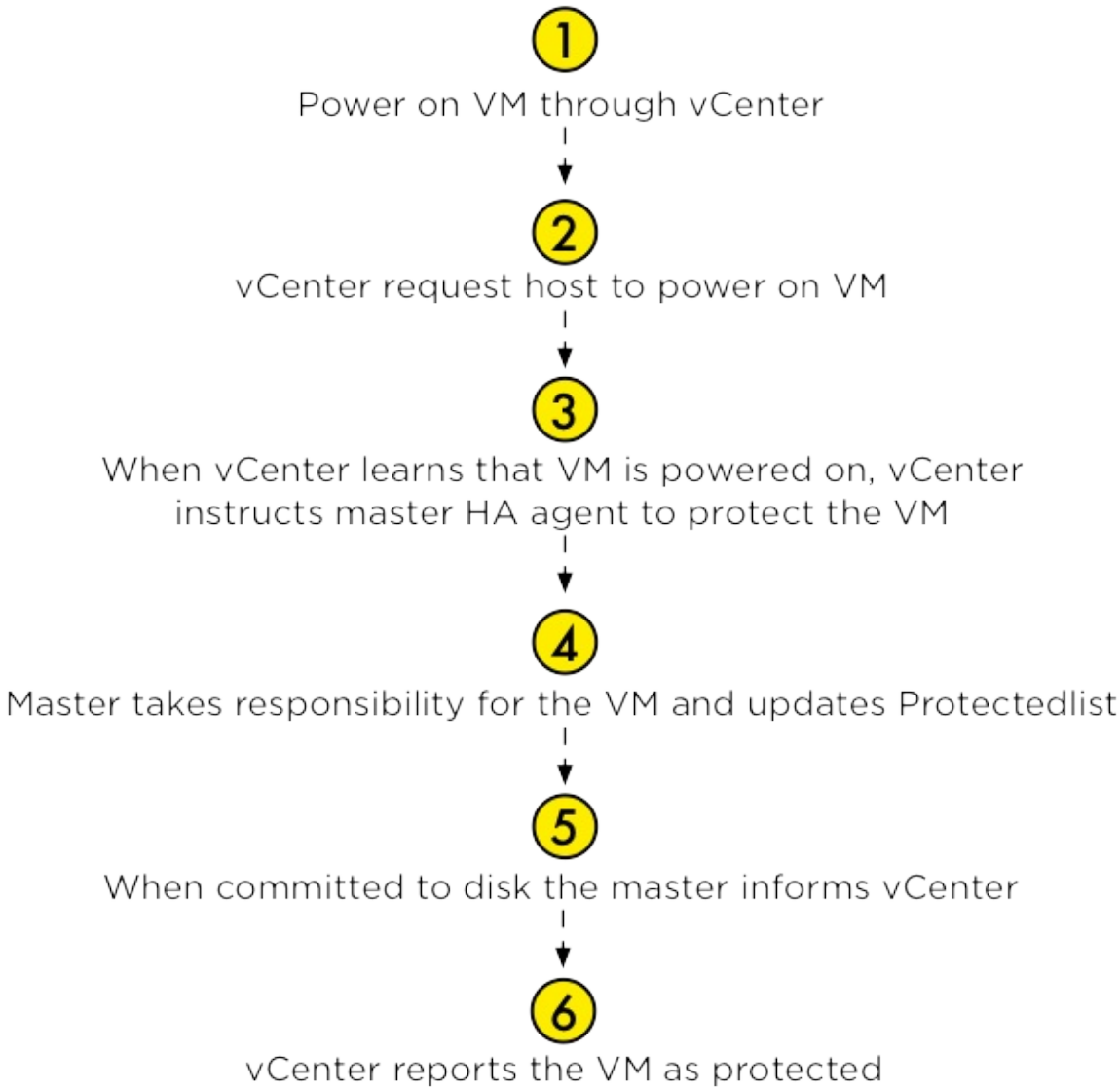


Figure 14 - Datastores used

But what about “unprotection?” When a virtual machine is powered off, it must be removed from the protectedlist. We have documented this workflow in the following diagram for the situation where the power off is invoked from vCenter.

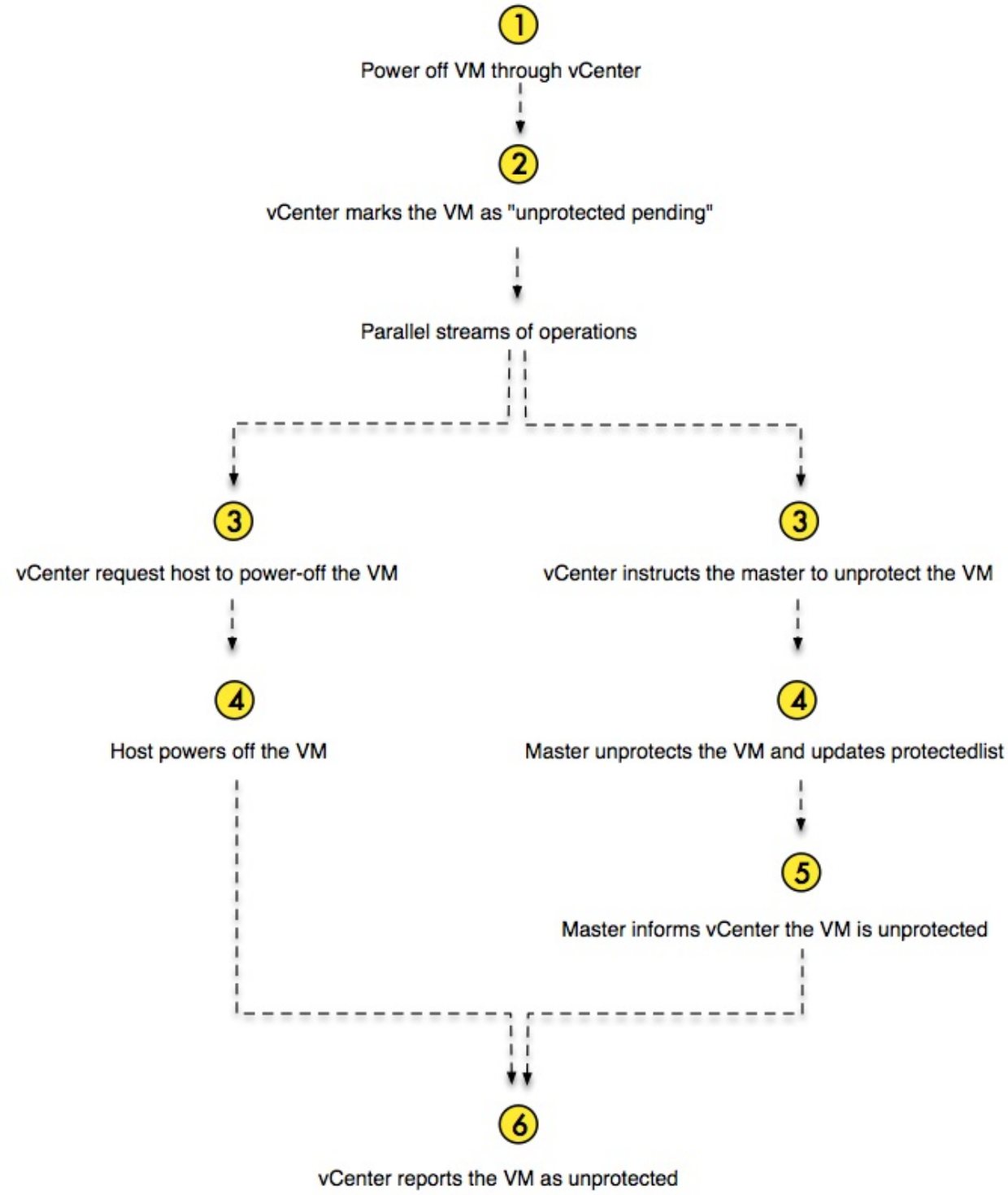


Figure 15 - Isolated versus Partitioned

Restarting Virtual Machines

In the previous chapter, we have described most of the lower level fundamental concepts of HA. We have shown you that multiple mechanisms increase resiliency and reliability of HA. Reliability of HA in this case mostly refers to restarting (or resetting) virtual machines, as that remains HA's primary task.

HA will respond when the state of a host has changed, or, better said, when the state of one or more virtual machines has changed. There are multiple scenarios in which HA will respond to a virtual machine failure, the most common of which are listed below:

- Failed host
- Isolated host
- Failed guest operating system

Depending on the type of failure, but also depending on the role of the host, the process will differ slightly. Changing the process results in slightly different recovery timelines. There are many different scenarios and there is no point in covering all of them, so we will try to describe the most common scenario and include timelines where possible.

Before we dive into the different failure scenarios, we want to explain how restart priority and retries work.

Restart Priority and Order

HA can take the configured priority of the virtual machine into account when restarting VMs. However, it is good to know that Agent VMs take precedence during the restart procedure as the "regular" virtual machines may rely on them. A good example of an agent virtual machine is a virtual storage appliance.

Prioritization is done by each host and not globally. Each host that has been requested to initiate restart attempts will attempt to restart all top priority virtual machines before attempting to start any other virtual machines. If the restart of a top priority virtual machine fails, it will be retried after a delay. In the meantime, however, HA will continue powering on the remaining virtual machines. Keep in mind that some virtual machines might be dependent on the agent virtual machines. You should document which virtual machines are dependent on which agent virtual machines and document the process to start up these services in the right order in the case the automatic restart of an agent virtual machine fails.

> **Basic design principle**: Virtual machines can be dependent on the availability of agent virtual machines or other virtual machines. Although HA will do its best to ensure all virtual machines are started in the correct order, this is not guaranteed. Document the proper recovery process.

Besides agent virtual machines, HA also prioritizes FT secondary machines. We have listed the full order in which virtual machines will be restarted below:

- Agent virtual machines
- FT secondary virtual machines
- Virtual Machines configured with a restart priority of high
- Virtual Machines configured with a medium restart priority
- Virtual Machines configured with a low restart priority

It should be noted that HA will not place any virtual machines on a host if the required number of agent virtual machines are not running on the host at the time placement is done.

Now that we have briefly touched on it, we would also like to address “restart retries” and parallelization of restarts as that more or less dictates how long it could take before all virtual machines of a failed or isolated host are restarted.

Restart Retries

The number of retries is configurable as of vCenter 2.5 U4 with the advanced option “*das.maxvmrestartcount*”. The default value is 5. Note that the initial restart is included.

HA will try to start the virtual machine on one of your hosts in the affected cluster; if this is unsuccessful on that host, the restart count will be increased by 1. Before we go into the exact timeline, let it be clear that T0 is the point at which the master initiates the first restart attempt. This by itself could be 30 seconds after the virtual machine has failed. The elapsed time between the failure of the virtual machine and the restart, though, will depend on the scenario of the failure, which we will discuss in this chapter.

As said, the default number of restarts is 5. There are specific times associated with each of these attempts. The following bullet list will clarify this concept. The ‘m’ stands for “minutes” in this list.

- T0 – Initial Restart
- T2m – Restart retry 1
- T6m – Restart retry 2
- T14m – Restart retry 3
- T30m – Restart retry 4

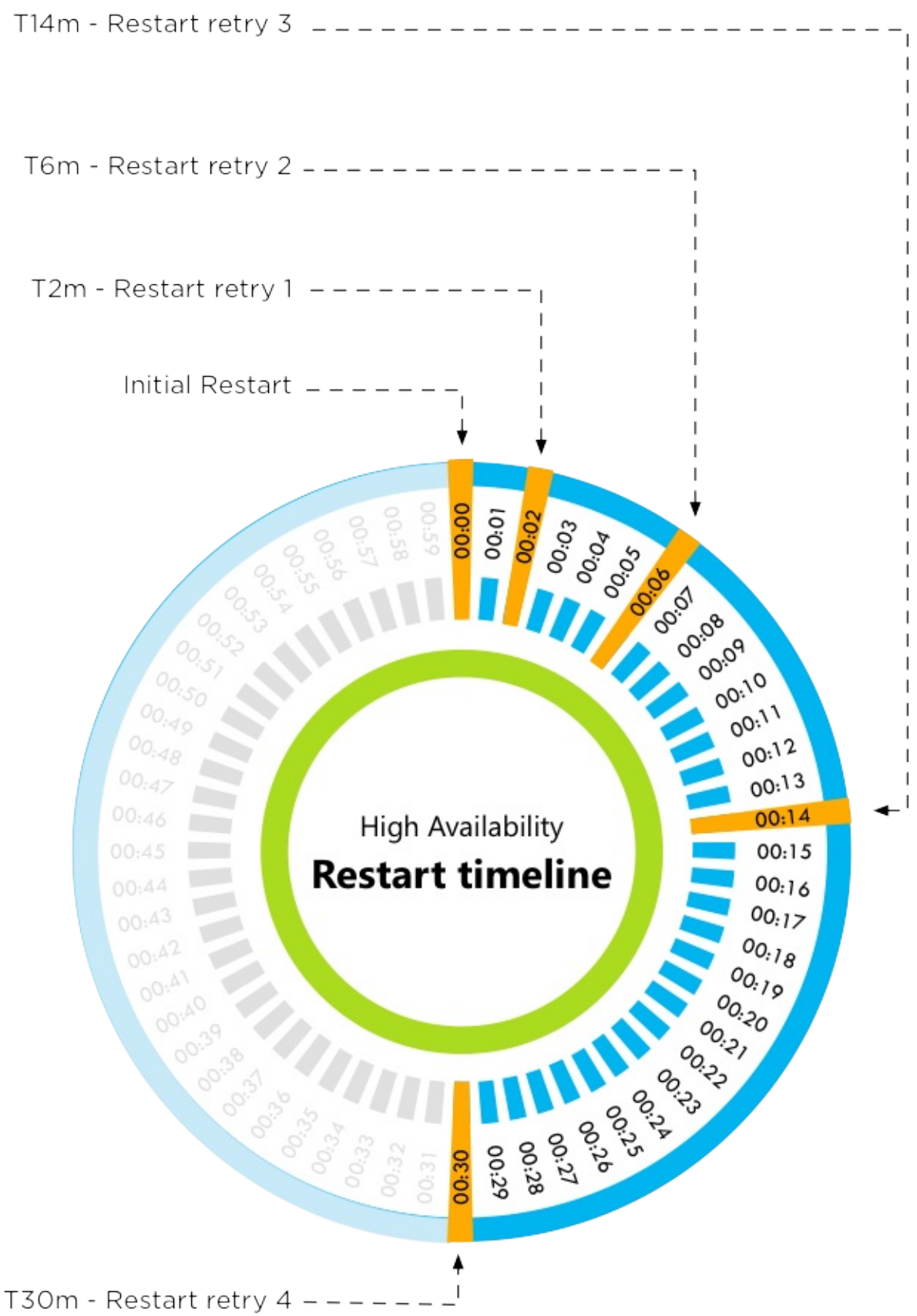


Figure 18 - High Availability restart timeline

As clearly depicted in the diagram above, a successful power-on attempt could take up to ~30 minutes in the case where multiple power-on attempts are unsuccessful. This is, however, not exact science. For instance, there is a 2-minute waiting period between the initial restart and the first restart retry. HA will start the 2-minute wait as soon as it has detected that the initial attempt has failed. So, in reality, T2 could be T2 plus 8 seconds. Another important fact that we want emphasize is that there is no coordination between masters, and so if multiple ones are involved in trying to restart the virtual machine, each will retain their own sequence. Multiple masters could attempt to restart a virtual machine. Although only one will succeed, it might change some of the timelines.

What about VMs which are "disabled" for HA? What will happen with those VMs? Before vSphere 6.0 those VMs would be left alone, as of vSphere 6.0 these VMs will be registered on another host after a failure. This will allow you to easily power-on that VM when needed without needed to manually re-register it yourself. Note, HA will not do a power-on of the VM, it will just register it for you!

Let’s give an example to clarify the scenario in which a master fails during a restart sequence:

```
Cluster: 4 Host (esxi01, esxi02, esxi03, esxi04)
Master: esxi01
```

The host “esxi02” is running a single virtual machine called “vm01” and it fails. The master, esxi01, will try to restart it but the attempt fails. It will try restarting “vm01” up to 5 times but, unfortunately, on the 4th try, the master also fails. An election occurs and “esxi03” becomes the new master. It will now initiate the restart of “vm01”, and if that restart would fail it will retry it up to 4 times again for a total including the initial restart of 5.

Be aware, though, that a successful restart might never occur if the restart count is reached and all five restart attempts (the default value) were unsuccessful.

When it comes to restarts, one thing that is very important to realize is that HA will not issue more than 32 concurrent power-on tasks on a given host. To make that more clear, let’s use the example of a two host cluster: if a host fails which contained 33 virtual machines and all of these had the same restart priority, 32 power on attempts would be initiated. The 33rd power on attempt will only be initiated when one of those 32 attempts has completed regardless of success or failure of one of those attempts.

Now, here comes the gotcha. If there are 32 low-priority virtual machines to be powered on and a single high-priority virtual machine, the power on attempt for the low-priority virtual machines will not be issued until the power on attempt for the high priority virtual machine has completed. Let it be absolutely clear that HA does not wait to restart the low-priority virtual machines until the high-priority virtual machines are started, it waits for the issued

power on attempt to be reported as “completed”. In theory, this means that if the power on attempt fails, the low-priority virtual machines could be powered on before the high priority virtual machine.

The restart priority however does guarantee that when a placement is done, the higher priority virtual machines get first right to any available resources.

> **Basic design principle**: Configuring restart priority of a virtual machine is not a guarantee that virtual machines will actually be restarted in this order. Ensure proper operational procedures are in place for restarting services or virtual machines in the appropriate order in the event of a failure.

Now that we know how virtual machine restart priority and restart retries are handled, it is time to look at the different scenarios.

- Failed host
  - Failure of a master
  - Failure of a slave
- Isolated host and response

Failed Host

When discussing a failed host scenario it is needed to make a distinction between the failure of a master versus the failure of a slave. We want to emphasize this because the time it takes before a restart attempt is initiated differs between these two scenarios. Although the majority of you probably won’t notice the time difference, it is important to call out. Let’s start with the most common failure, that of a host failing, but note that failures generally occur infrequently. In most environments, hardware failures are very uncommon to begin with. Just in case it happens, it doesn’t hurt to understand the process and its associated timelines.

The Failure of a Slave

The failure of a slave host is a fairly complex scenario. Part of this complexity comes from the introduction of a new heartbeat mechanism. Actually, there are two different scenarios: one where heartbeat datastores are configured and one where heartbeat datastores are not configured. Keeping in mind that this is an actual failure of the host, the timeline is as follows:

- T0 – Slave failure.
- T3s – Master begins monitoring datastore heartbeats for 15 seconds.
- T10s – The host is declared unreachable and the master will ping the management network of the failed host. This is a continuous ping for 5 seconds.

- T15s – If **no** heartbeat datastores are configured, the host will be declared dead.
- T18s – If heartbeat datastores are configured, the host will be declared dead.

The master monitors the network heartbeats of a slave. When the slave fails, these heartbeats will no longer be received by the master. We have defined this as T0. After 3 seconds (T3s), the master will start monitoring for datastore heartbeats and it will do this for 15 seconds. On the 10th second (T10s), when no network or datastore heartbeats have been detected, the host will be declared as “unreachable”. The master will also start pinging the management network of the failed host at the 10th second and it will do so for 5 seconds. If no heartbeat datastores were configured, the host will be declared “dead” at the 15th second (T15s) and virtual machine restarts will be initiated by the master. If heartbeat datastores have been configured, the host will be declared dead at the 18th second (T18s) and restarts will be initiated. We realize that this can be confusing and hope the timeline depicted in the diagram below makes it easier to digest.

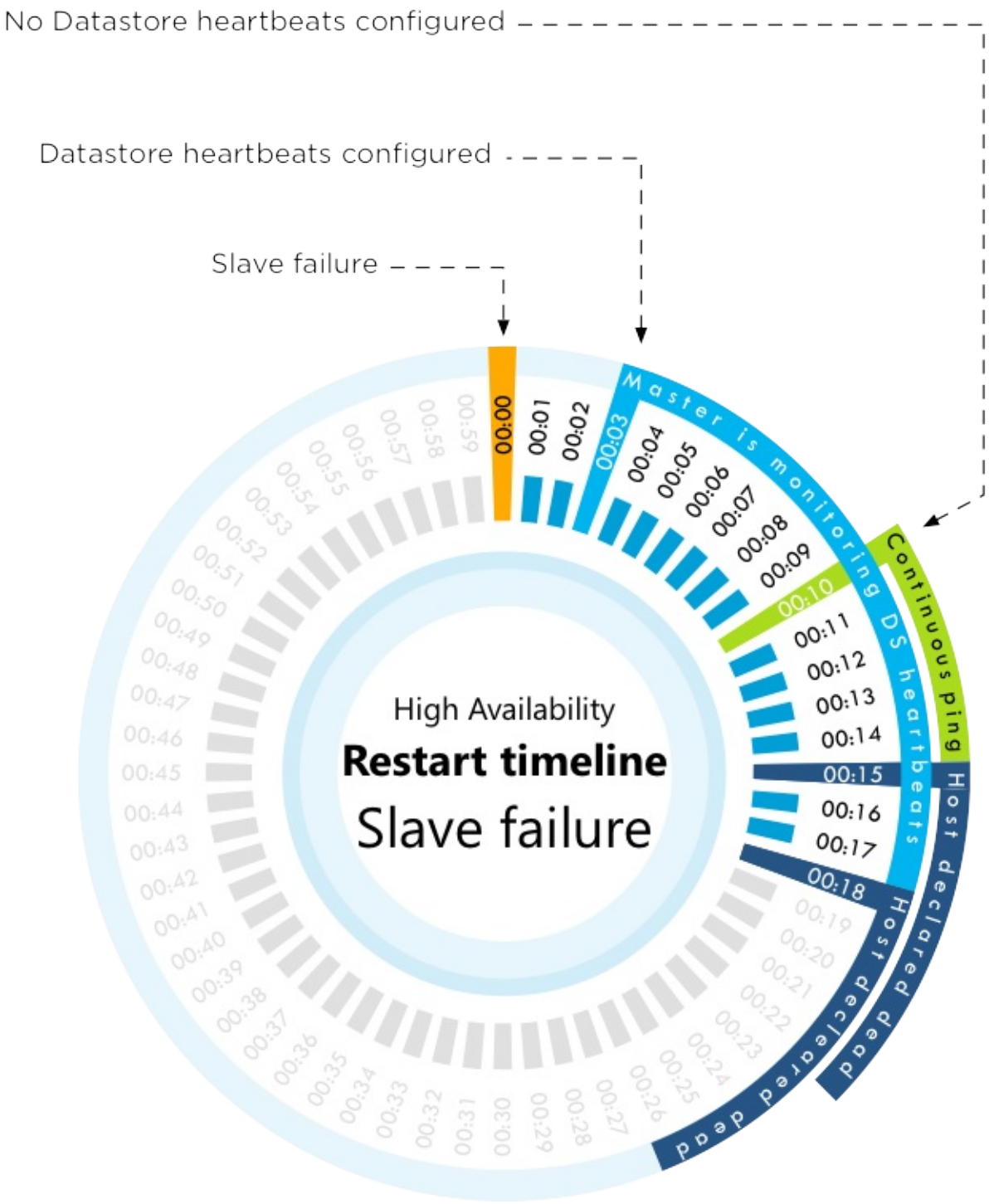


Figure 19 - Restart timeline slave failure

The master filters the virtual machines it thinks failed before initiating restarts. The master uses the protectedlist for this, on-disk state could be obtained only by one master at a time since it required opening the protectedlist file in exclusive mode. If there is a network partition multiple masters could try to restart the same virtual machine as vCenter Server also provided the necessary details for a restart. As an example, it could happen that a master has locked a virtual machine's home datastore and has access to the protectedlist

while the other master is in contact with vCenter Server and as such is aware of the current desired protected state. In this scenario it could happen that the master which does not own the home datastore of the virtual machine will restart the virtual machine based on the information provided by vCenter Server.

This change in behavior was introduced to avoid the scenario where a restart of a virtual machine would fail due to insufficient resources in the partition which was responsible for the virtual machine. With this change, there is less chance of such a situation occurring as the master in the other partition would be using the information provided by vCenter Server to initiate the restart.

That leaves us with the question of what happens in the case of the failure of a master.

The Failure of a Master

In the case of a master failure, the process and the associated timeline are slightly different. The reason being that there needs to be a master before any restart can be initiated. This means that an election will need to take place amongst the slaves. The timeline is as follows:

- T0 – Master failure.
- T10s – Master election process initiated.
- T25s – New master elected and reads the protectedlist.
- T35s – New master initiates restarts for all virtual machines on the protectedlist which are not running.

Slaves receive network heartbeats from their master. If the master fails, let’s define this as T0 (T zero), the slaves detect this when the network heartbeats cease to be received. As every cluster needs a master, the slaves will initiate an election at T10s. The election process takes 15s to complete, which brings us to T25s. At T25s, the new master reads the protectedlist. This list contains all the virtual machines, which are protected by HA. At T35s, the master initiates the restart of all virtual machines that are protected but not currently running. The timeline depicted in the diagram below hopefully clarifies the process.



Figure 20 - Restart timeline master failure

Besides the failure of a host, there is another reason for restarting virtual machines: an isolation event.

Isolation Response and Detection

Before we will discuss the timeline and the process around the restart of virtual machines after an isolation event, we will discuss Isolation Response and Isolation Detection. One of the first decisions that will need to be made when configuring HA is the “Isolation Response”.

Isolation Response

The Isolation Response (or Host Isolation as it is called in vSphere 6.0) refers to the action that HA takes for its virtual machines when the host has lost its connection with the network and the remaining nodes in the cluster. This does not necessarily mean that the whole network is down; it could just be the management network ports of this specific host. Today there are two isolation responses: “Power off”, and “Shut down”. In previous versions (pre vSphere 6.0) there was also an isolation response called "leave powered on", this has been renamed to "disabled" as "leave powered on" means that there is no response to an isolation event.

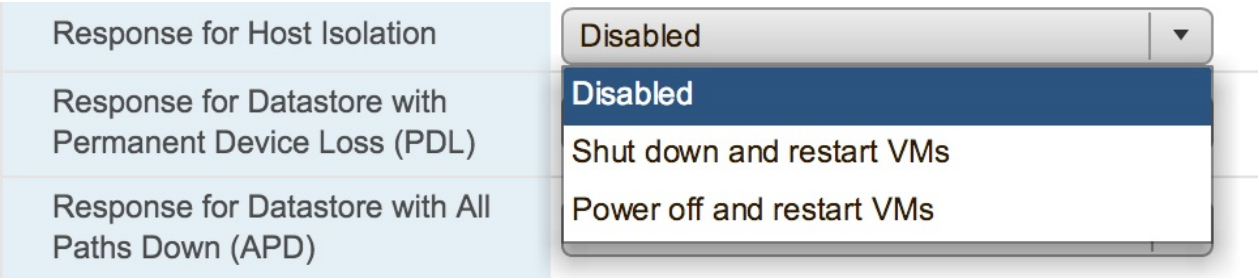


Figure 21 - vSphere 6.0 Isolation Response settings

The isolation response features answers the question, “what should a host do with the virtual machines it manages when it detects that it is isolated from the network?” Let’s discuss these three options more in-depth:

- Disabled (default) – When isolation occurs on the host, the state of the virtual machines remains unchanged.
- Power off – When isolation occurs, all virtual machines are powered off. It is a hard stop, or to put it bluntly, the “virtual” power cable of the virtual machine will be pulled out!
- Shut down – When isolation occurs, all virtual machines running on the host will be shut down using a guest-initiated shutdown through VMware Tools. If this is not successful within 5 minutes, a “power off” will be executed. This time out value can be adjusted by setting the advanced option *das.isolationShutdownTimeout*. If VMware Tools is not installed, a “power off” will be initiated immediately.

This setting can be changed on the cluster settings under virtual machine options.

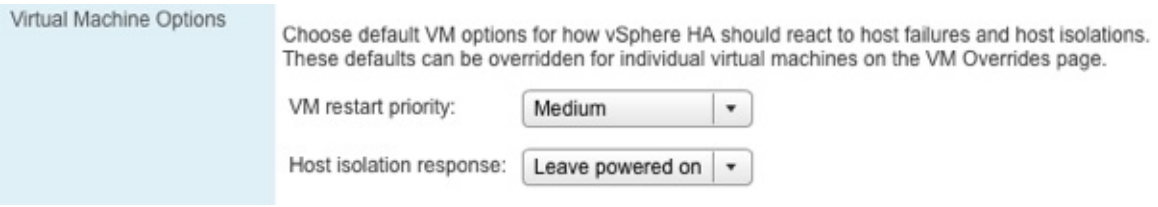


Figure 22 - Cluster default settings

The default setting for the isolation response has changed multiple times over the last couple of years and this has caused some confusion.

- Up to ESXi3.5 U2 / vCenter 2.5 U2 the default isolation response was “Power off”
- With ESXi3.5 U3 / vCenter 2.5 U3 this was changed to “Leave powered on”
- With vSphere 4.0 it was changed to “Shut down”.
- With vSphere 5.0 it has been changed to “Leave powered on”.
- With vSphere 6.0 the "leave powered on" setting is now renamed to "Disabled".

Keep in mind that these changes are only applicable to newly created clusters. When creating a new cluster, it may be required to change the default isolation response based on the configuration of existing clusters and/or your customer’s requirements, constraints and expectations. When upgrading an existing cluster, it might be wise to apply the latest default values. You might wonder why the default has changed once again. There was a lot of feedback from customers that “Disabled” was the desired default value.

> **Basic design principle**: Before upgrading an environment to later versions, ensure you validate the best practices and default settings. Document them, including justification, to ensure all people involved understand your reasons.

The question remains, which setting should be used? The obvious answer applies here; it depends. We prefer “Disabled” because it eliminates the chances of having a false positive and its associated down time. One of the problems that people have experienced in the past is that HA triggered its isolation response when the full management network went down. Basically resulting in the power off (or shutdown) of every single virtual machine and none being restarted. This problem has been mitigated. HA will validate if virtual machines restarts can be attempted – there is no reason to incur any down time unless absolutely necessary. It does this by validating that a master owns the datastore the virtual machine is stored on. Of course, the isolated host can only validate this if it has access to the datastores. In a converged network environment with iSCSI storage, for instance, it would be impossible to validate this during a full isolation as the validation would fail due to the inaccessible datastore from the perspective of the isolated host.

We feel that changing the isolation response is most useful in environments where a failure of the management network is likely correlated with a failure of the virtual machine network(s). If the failure of the management network won’t likely correspond with the failure of the virtual machine networks, isolation response would cause unnecessary downtime as the virtual machines can continue to run without management network connectivity to the host.

A second use for power off/shutdown is in scenarios where the virtual machine retains access to the virtual machine network but loses access to its storage, leaving the virtual machine powered-on could result in two virtual machines on the network with the same IP

address.

It is still difficult to decide which isolation response should be used. The following table was created to provide some more guidelines.

Likelihood that host will retain access to VM datastore	Likelihood VMs will retain access to VM network	Recommended Isolation Policy	Rationale
Likely	Likely	Disabled	Virtual machine is running fine, no reason to power it off
Likely	Unlikely	Either Disabled On or Shutdown.	Choose shutdown to allow HA to restart virtual machines on hosts that are not isolated and hence are likely to have access to storage
Unlikely	Likely	Power Off	Use Power Off to avoid having two instances of the same virtual machine on the virtual machine network
Unlikely	Unlikely	Disabled or Power Off	Disabled if the virtual machine can recover from the network/datastore outage if it is not restarted because of the isolation, and Power Off if it likely can’t.

The question that we haven’t answered yet is how HA knows which virtual machines have been powered-off due to the triggered isolation response and why the isolation response is more reliable than with previous versions of HA. Previously, HA did not care and would always try to restart the virtual machines according to the last known state of the host. That is no longer the case. Before the isolation response is triggered, the isolated host will verify whether a master is responsible for the virtual machine.

As mentioned earlier, it does this by validating if a master owns the home datastore of the virtual machine. When isolation response is triggered, the isolated host removes the virtual machines which are powered off or shutdown from the “poweron” file. The master will recognize that the virtual machines have disappeared and initiate a restart. On top of that, when the isolation response is triggered, it will create a per-virtual machine file under a “poweredoff” directory which indicates for the master that this virtual machine was powered down as a result of a triggered isolation response. This information will be read by the master node when it initiates the restart attempt in order to guarantee that only virtual machines that were powered off / shut down by HA will be restarted by HA.

This is, however, only one part of the increased reliability of HA. Reliability has also been improved with respect to “isolation detection,” which will be described in the following section.

Isolation Detection

We have explained what the options are to respond to an isolation event and what happens when the selected response is triggered. However, we have not extensively discussed how isolation is detected. The mechanism is fairly straightforward and works with heartbeats, as earlier explained. There are, however, two scenarios again, and the process and associated timelines differ for each of them:

- Isolation of a slave
- Isolation of a master

Before we explain the differences in process between both scenarios, we want to make sure it is clear that a change in state will result in the isolation response not being triggered in either scenario. Meaning that if a single ping is successful or the host observes election traffic and is elected a master or slave, the isolation response will not be triggered, which is exactly what you want as avoiding down time is at least as important as recovering from down time. When a host has declared itself isolated and observes election traffic it will declare itself no longer isolated.

Isolation of a Slave

HA triggers a master election process before it will declare a host is isolated. In the below timeline, “s” refers to seconds.

- T0 – Isolation of the host (slave)
- T10s – Slave enters “election state”
- T25s – Slave elects itself as master
- T25s – Slave pings “isolation addresses”
- T30s – Slave declares itself isolated
- T60s – Slave “triggers” isolation response

When the isolation response is triggered HA creates a “power-off” file for any virtual machine HA powers off whose home datastore is accessible. Next it powers off the virtual machine (or shuts down) and updates the host’s poweron file. The power-off file is used to record that HA powered off the virtual machine and so HA should restart it. These power-off files are deleted when a virtual machine is powered back on or HA is disabled, the below screenshot shows such a power-off file, which in this case is stored in a VVol.

```
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.f21565c0-6e5c-4fc5-999d-a64c0d477b1c] ls -lah
total 488464
drwxr-xr-t    1 root     root        3.0K Apr 15 09:58 .
drwxr-xr-x    1 root     root         512 Apr 15 10:56 ..
drwxr-xr-x    1 root     root         420 Apr 15 10:35 .dvsData
-r--------    1 root     root      128.0K Apr 15 09:52 .fbb.sf
-r--------    1 root     root       79.8M Apr 15 09:52 .fdc.sf
-r--------    1 root     root        1.1M Apr 15 09:52 .pb2.sf
-r--------    1 root     root      256.0M Apr 15 09:52 .pbc.sf
-rw-------    1 root     root           0 Apr 15 09:52 .rfc4122.c492c080-6a96-483c-8559-ca45333faecf.lck
-rw-------    1 root     root         423 Apr 15 09:52 .rfc4122.c492c080-6a96-483c-8559-ca45333faecf.meta
-r--------    1 root     root      128.3M Apr 15 09:52 .sbc.sf
drwx------    1 root     root         280 Apr 15 09:52 .sdd.sf
-rw-------    1 root     root          55 Apr 15 10:56 .vSphereHA-poweroff
-r--------    1 root     root        4.0M Apr 15 09:52 .vh.sf
-rw-r--r--    1 root     root      165.6K Apr 15 10:35 vmware-1.log
-rw-r--r--    1 root     root      166.4K Apr 15 10:56 vmware.log
-rw-r--r--    1 root     root          90 Apr 15 10:35 win-file-srv-01-59643fe5.hlog
-rw-------    1 root     root        8.5K Apr 15 10:56 win-file-srv-01.nvram
-rw-------    1 root     root         563 Apr 15 09:52 win-file-srv-01.vmdk
-rw-r--r--    1 root     root           0 Apr 15 09:52 win-file-srv-01.vmsd
-rwxr-xr-x    1 root     root        2.9K Apr 15 10:56 win-file-srv-01.vmx
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.f21565c0-6e5c-4fc5-999d-a64c0d477b1c]
```

Figure 23 - VVol based poweroff file

After the completion of this sequence, the master will learn the slave was isolated through the “poweron” file as mentioned earlier, and will restart virtual machines based on the information provided by the slave.

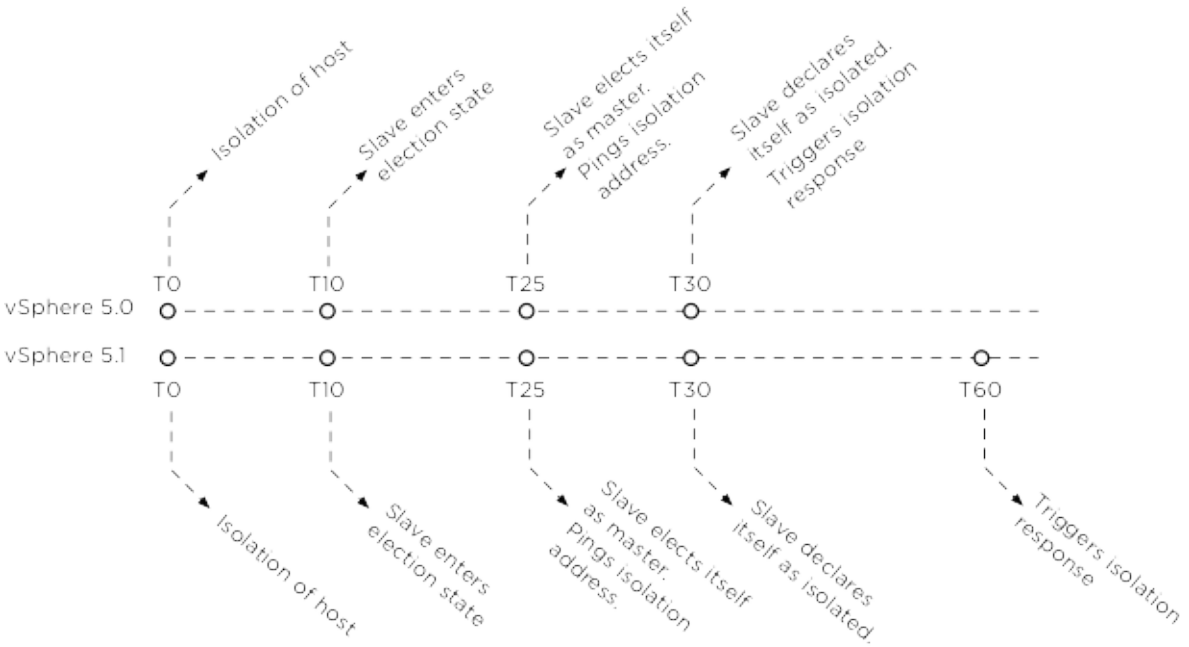


Figure 24 - Isolation of a slave timeline

Isolation of a Master

In the case of the isolation of a master, this timeline is a bit less complicated because there is no need to go through an election process. In this timeline, “s” refers to seconds.

- T0 – Isolation of the host (master)

- T0 – Master pings “isolation addresses”
- T5s – Master declares itself isolated
- T35s – Master “triggers” isolation response

Additional Checks

Before a host declares itself isolated, it will ping the default isolation address which is the gateway specified for the management network, and will continue to ping the address until it becomes unisolated. HA gives you the option to define one or multiple additional isolation addresses using an advanced setting. This advanced setting is called *das.isolationaddress* and could be used to reduce the chances of having a false positive. We recommend setting an additional isolation address. If a secondary management network is configured, this additional address should be part of the same network as the secondary management network. If required, you can configure up to 10 additional isolation addresses. A secondary management network will more than likely be on a different subnet and it is recommended to specify an additional isolation address which is part of the subnet.

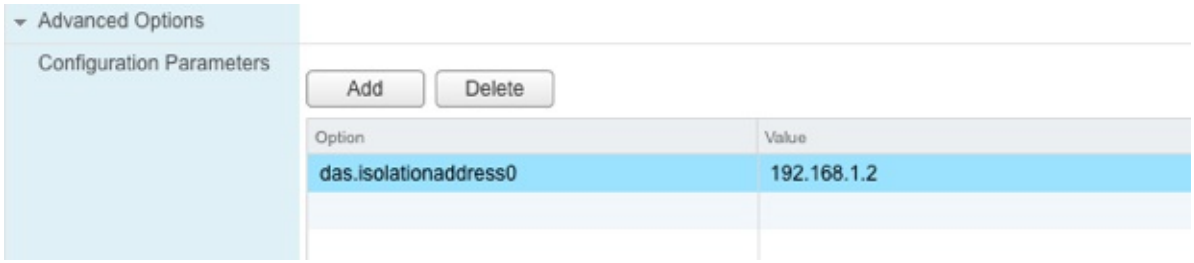


Figure 25 - Isolation Address

Selecting an Additional Isolation Address

A question asked by many people is which address should be specified for this additional isolation verification. We generally recommend an isolation address close to the hosts to avoid too many network hops and an address that would correlate with the liveness of the virtual machine network. In many cases, the most logical choice is the physical switch to which the host is directly connected. Basically, use the gateway for whatever subnet your management network is on. Another usual suspect would be a router or any other reliable and pingable device on the same subnet. However, when you are using IP-based shared storage like NFS or iSCSI, the IP-address of the storage device can also be a good choice.

> **Basic design principle:** Select a reliable secondary isolation address. Try to minimize the number of “hops” between the host and this address.

Isolation Policy Delay

For those who want to increase the time it takes before HA executes the isolation response an advanced setting is available. Thus setting is called "*das.config.fdm.isolationPolicyDelaySec*" and allows changing the number of seconds to wait before the isolation policy is executed is. The minimum value is 30. If set to a value less than 30, the delay will be 30 seconds. We do not recommend changing this advanced setting unless there is a specific requirement to do so. In almost all scenarios 30 seconds should suffice.

Restarting Virtual Machines

The most important procedure has not yet been explained: restarting virtual machines. We have dedicated a full section to this concept.

We have explained the difference in behavior from a timing perspective for restarting virtual machines in the case of a both master node and slave node failures. For now, let's assume that a slave node has failed. When the master node declares the slave node as Partitioned or Isolated, it determines which virtual machines were running on using the information it previously read from the host's "poweron" file. These files are asynchronously read approximately every 30s. If the host was not Partitioned or Isolated before the failure, the master uses cached data to determine the virtual machines that were last running on the host before the failure occurred.

Before it will initiate the restart attempts, though, the master will first validate that the virtual machine should be restarted. This validation uses the protection information vCenter Server provides to each master, or if the master is not in contact with vCenter Server, the information saved in the protectedlist files. If the master is not in contact with vCenter Server or has not locked the file, the virtual machine is filtered out. At this point, all virtual machines having a restart priority of "disabled" are also filtered out.

Now that HA knows which virtual machines it should restart, it is time to decide where the virtual machines are placed. HA will take multiple things in to account:

- CPU and memory reservation, including the memory overhead of the virtual machine
- Unreserved capacity of the hosts in the cluster
- Restart priority of the virtual machine relative to the other virtual machines that need to be restarted
- Virtual-machine-to-host compatibility set
- The number of dvPorts required by a virtual machine and the number available on the candidate hosts

- The maximum number of vCPUs and virtual machines that can be run on a given host
- Restart latency
- Whether the active hosts are running the required number of agent virtual machines.

Restart latency refers to the amount of time it takes to initiate virtual machine restarts. This means that virtual machine restarts will be distributed by the master across multiple hosts to avoid a boot storm, and thus a delay, on a single host.

If a placement is found, the master will send each target host the set of virtual machines it needs to restart. If this list exceeds 32 virtual machines, HA will limit the number of concurrent power on attempts to 32. If a virtual machine successfully powers on, the node on which the virtual machine was powered on will inform the master of the change in power state. The master will then remove the virtual machine from the restart list.

If a placement cannot be found, the master will place the virtual machine on a “pending placement list” and will retry placement of the virtual machine when one of the following conditions changes:

- A new virtual-machine-to-host compatibility list is provided by vCenter.
- A host reports that its unreserved capacity has increased.
- A host (re)joins the cluster (For instance, when a host is taken out of maintenance mode, a host is added to a cluster, etc.)
- A new failure is detected and virtual machines have to be failed over.
- A failure occurred when failing over a virtual machine.

But what about DRS? Wouldn’t DRS be able to help during the placement of virtual machines when all else fails? It does. The master node will report to vCenter the set of virtual machines that were not placed due to insufficient resources, as is the case today. If DRS is enabled, this information will be used in an attempt to have DRS make capacity available.

Component Protection

In vSphere 6.0 a new feature as part of vSphere HA is introduced called VM Component Protection. VM Component Protection (VMCP) in vSphere 6.0 allows you to protect virtual machines against the failure of your storage system. There are two types of failures VMCP will respond to and those are Permanent Device Loss (PDL) and All Paths Down (APD). Before we look at some of the details, we want to point out that enabling VMCP is extremely easy. It can be enabled by a single tick box as shown in the screenshot below.

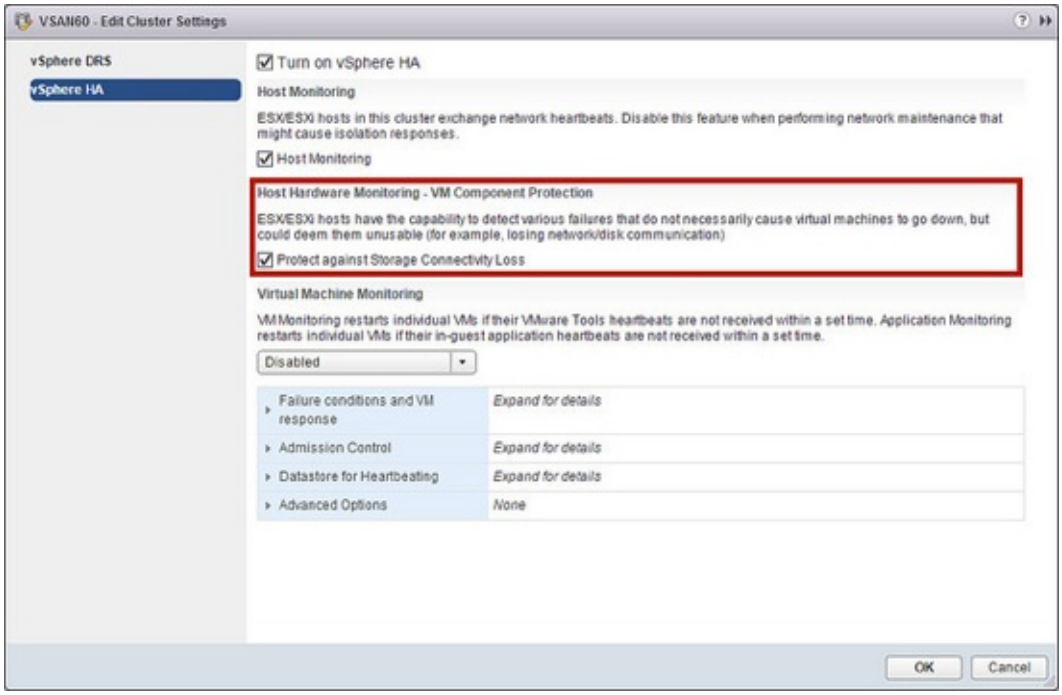


Figure 26 - Virtual Machine Component Protection

As stated there are two scenarios HA can respond to, PDL and APD. Lets look at those two scenarios a bit closer. With vSphere 5.0 a feature was introduced as an advanced option that would allow vSphere HA to restart VMs impacted by a PDL condition.

A PDL condition, is a condition that is communicated by the array controller to ESXi via a SCSI sense code. This condition indicates that a device (LUN) has become unavailable and is likely permanently unavailable. An example scenario in which this condition would be communicated by the array would be when a LUN is set offline. This condition is used during a failure scenario to ensure ESXi takes appropriate action when access to a LUN is revoked. It should be noted that when a full storage failure occurs it is impossible to generate the PDL condition as there is no communication possible between the array and the ESXi host. This state will be identified by the ESXi host as an APD condition.

Although the functionality itself worked as advertised, enabling and managing it was cumbersome and error prone. It was required to set the option “disk.terminateVMOnPDLDefault” manually. With vSphere 6.0 a simple option in the Web Client is introduced which allows you to specify what the response should be to a PDL sense code.

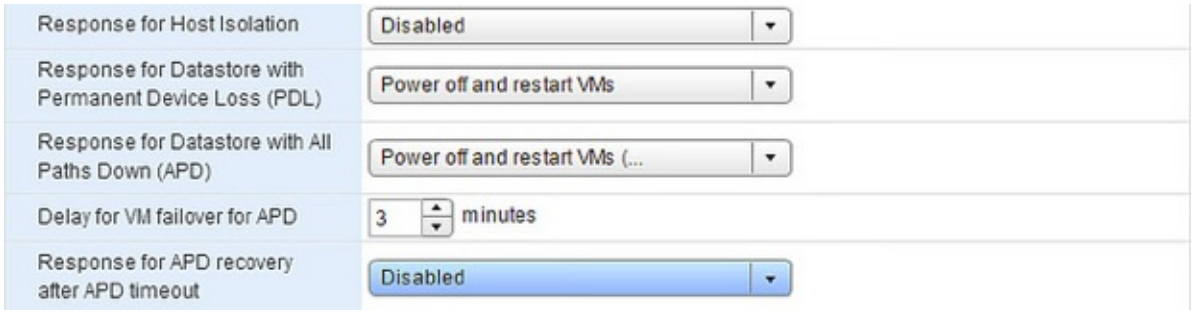


Figure 27 - Enabling Virtual Machine Component Protection

The two options provided are “Issue Events” and “Power off and restart VMs”. Note that “Power off and restart VMs” does exactly that, your VM process is killed and the VM is restarted on a host which still has access to the storage device.

Until now it was not possible for vSphere to respond to an APD scenario. APD is the situation where the storage device is inaccessible but for unknown reasons. In most cases where this occurs it is typically related to a storage network problem. With vSphere 5.1 changes were introduced to the way APD scenarios were handled by the hypervisor. This mechanism is leveraged by HA to allow for a response.

When an APD occurs a timer starts. After 140 seconds the APD is declared and the device is marked as APD time out. When the 140 seconds has passed HA will start counting. The HA time out is 3 minutes by default at shown in Figure 24. When the 3 minutes has passed HA will take the action defined. There are again two options “Issue Events” and “Power off and restart VMs”.

You can also specify how aggressively HA needs to try to restart VMs that are impacted by an APD. Note that aggressive / conservative refers to the likelihood of HA being able to restart VMs. When set to “conservative” HA will only restart the VM that is impacted by the APD if it knows another host can restart it. In the case of “aggressive” HA will try to restart the VM even if it doesn’t know the state of the other hosts, which could lead to a situation where your VM is not restarted as there is no host that has access to the datastore the VM is located on.

It is also good to know that if the APD is lifted and access to the storage is restored during the total of the approximate 5 minutes and 20 seconds it would take before the VM restart is initiated, that HA will not do anything unless you explicitly configure it do so. This is where the “Response for APD recovery after APD timeout” comes in to play. If there is a desire to do so you can restart the VM even when the host has recovered from the APD scenario, during the 3 minute (default value) grace period.

> **Basic design principle:** Without access to shared storage a virtual machine becomes useless. It is highly recommended to configure VMCP to act on a PDL and APD scenario. We recommend to set both to “power off and restarts VMs” but leave the “response for APD recovery after APD timeout” disabled so that VMs are not rebooted unnecessarrily.

vSphere HA nuggets

Prior to vSphere 5.5, HA did nothing with VM to VM Affinity or Anti Affinity rules. Typically for people using “affinity” rules this was not an issue, but those using “anti-affinity” rules did see this as an issue. They created these rules to ensure specific virtual machines would never be running on the same host, but vSphere HA would simply ignore the rule when a failure had occurred and just place the VMs “randomly”. With vSphere 5.5 this has changed! vSphere HA is now “anti affinity” aware. In order to ensure anti-affinity rules are respected you can set an advanced setting or configure in the vSphere Web Client as of vSphere 6.0.

```
das.respectVmVmAntiAffinityRules - Values: "false" (default) and "true"
```

Now note that this also means that when you configure anti-affinity rules and have this advanced setting configured to “true” and somehow there aren’t sufficient hosts available to respect these rules… then rules will be respected and it could result in HA not restarting a VM. Make sure to understand this potential impact when configuring this setting and configuring these rules.

With vSphere 6.0 support for respecting VM to Host affinity rules has been included. This is enabled through the use of an advanced setting called “das.respectVmHostSoftAffinityRules”. When the advanced setting “das.respectVmHostSoftAffinityRules” is configured vSphere HA will try to respect the rules when it can. If there are any hosts in the cluster which belong to the same VM-Host group then HA will restart the respective VM on that host. As this is a “should rule” HA has the ability to ignore the rule when needed. If there is a scenario where none of the hosts in the VM-Host should rule is available HA will restart the VM on any other host in the cluster.

```
das. respectVmHostSoftAffinityRules - Values: "false" (default) and "true"
```

Virtual SAN and Virtual Volumes specifics

In the last couple of sections we have discussed the ins and out of HA. All of it based on VMFS based or NFS based storage. With the introduction of Virtual SAN and Virtual Volumes also comes changes to some of the discussed concepts.

HA and Virtual SAN

Virtual SAN is VMware's approach to Software Defined Storage. We are not going to explain the ins and outs of Virtual SAN, but do want to provide a basic understanding for those who have never done anything with it. Virtual SAN leverages host local storage and creates a shared data store out of it.

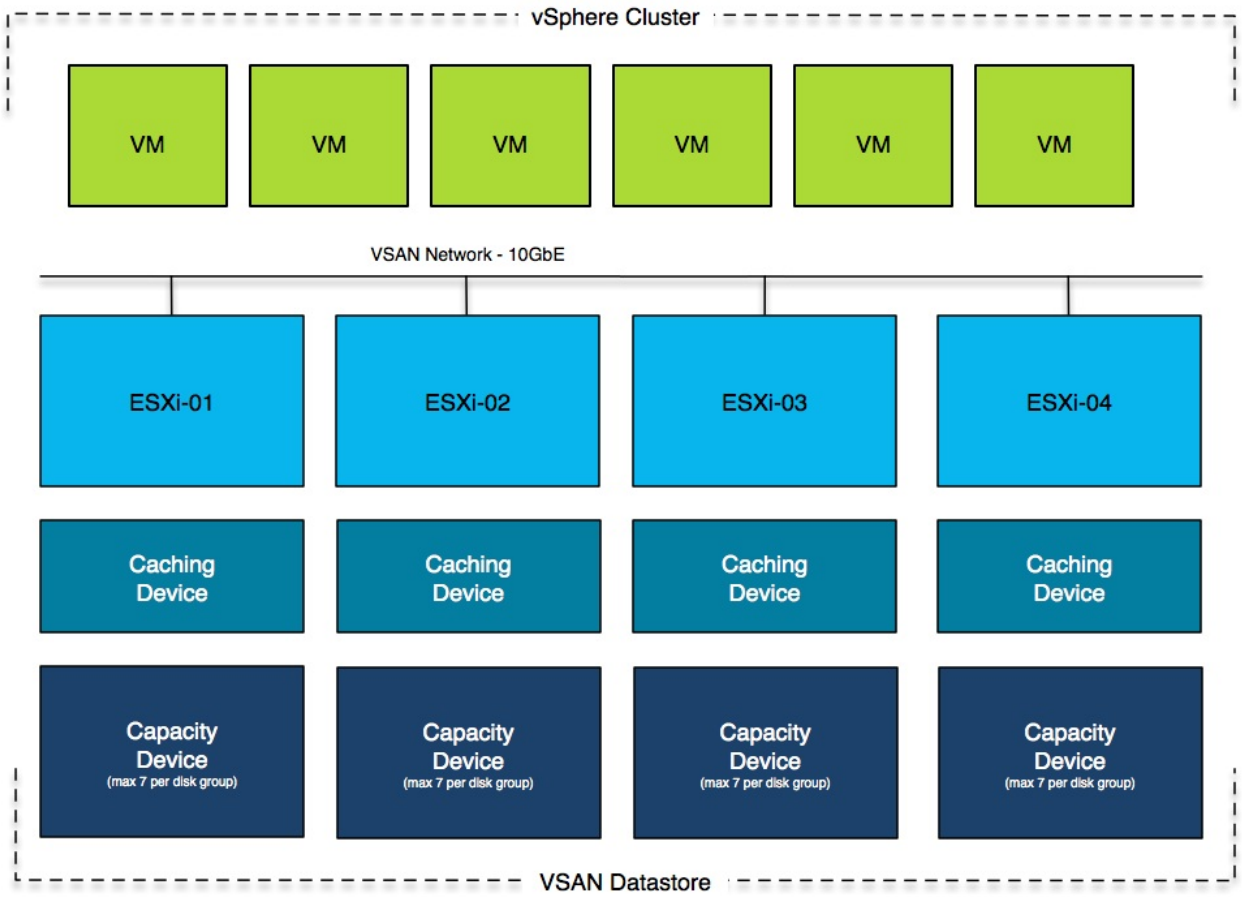


Figure 28 - Virtual SAN Cluster

Virtual SAN requires a minimum of 3 hosts and each of those 3 hosts will need to have 1 SSD for caching and 1 capacity device (can be SSD or HDD). Only the capacity devices will contribute to the available capacity of the datastore. If you have 1TB worth of capacity devices per host then with three hosts the total size of your datastore will be 3TB.

Having that said, with Virtual SAN 6.1 VMware introduced a "2-node" option. This 2-node option is actually 2 regular VSAN nodes with a third "witness" node.

The big differentiator between most storage systems and Virtual SAN is that availability of the virtual machine's is defined on a per virtual disk or per virtual machine basis. This is called "Failures To Tolerate" and can be configured to any value between 0 (zero) and 3. When configured to 0 then the virtual machine will have only 1 copy of its virtual disks which means that if a host fails where the virtual disks are stored the virtual machine is lost. As such all virtual machines are deployed by default with Failures To Tolerate(FTT) set to 1. A virtual disk is what VSAN refers to as an object. An object, when FTT is configured as 1 or higher, has multiple components. In the diagram below we demonstrate the FTT=1 scenario, and the virtual disk in this case has 2 "data components" and a "witness components". The witness is used as a "quorom" mechnanism.

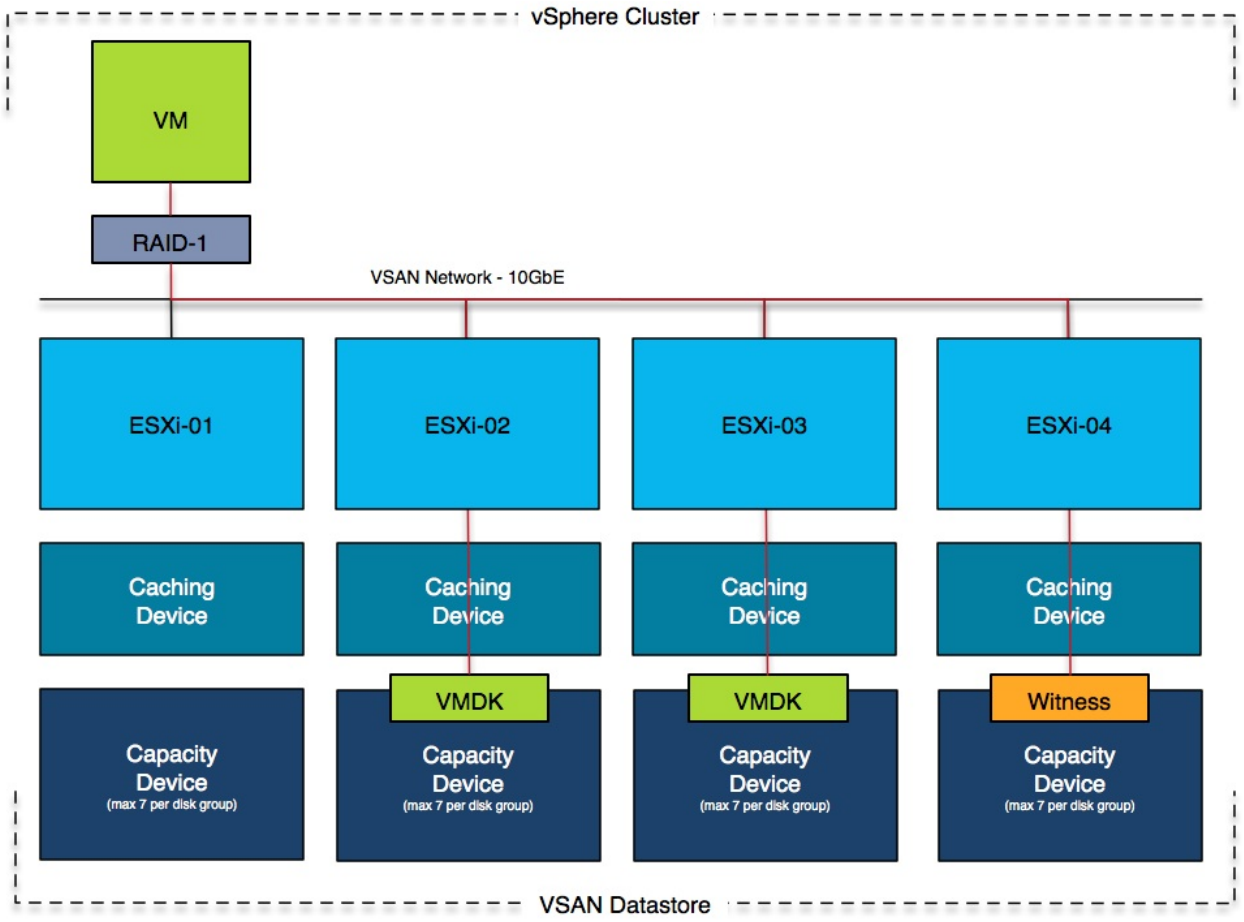


Figure 29 - Virtual SAN Object model

As the diagram above depicts, a virtual machine can be running on the first host while its storage components are on the remaining hosts in the cluster. As you can imagine from an HA point of view this changes things as access to the network is not only critical for HA to function correctly but also for Virtual SAN. When it comes to networking note that when Virtual SAN is configured in a cluster HA will use the same network for its communications (heartbeating etc). On top of that, it is good to know that VMware highly recommends 10GbE to be used for Virtual SAN.

> **Basic design principle:** 10GbE is highly recommend for Virtual SAN, as vSphere HA also leverages the Virtual SAN network and availability of VMs is dependent on network connectivity ensure that at a minimum two 10GbE ports are used and two physical switches for resiliency.

The reason that HA uses the same network as Virtual SAN is simple, it is too avoid network partition scenarios where HA communications is separated from Virtual SAN and the state of the cluster is unclear. Note that you will need to ensure that there is a pingable isolation address on the Virtual SAN network and this isolation address will need to be configured as such through the use of the advanced setting “das.isolationAddress0”. We also recommend to disable the use of the default isolation address through the advanced setting “das.useDefaultIsolationAddress” (set to false).

When an isolation does occur the isolation response is triggered as explained in earlier chapters. For Virtual SAN the recommendation is simple, configure the isolation response to “Power Off, then fail over”. This is the safest option. Virtual SAN can be compared to the “converged network with IP based storage” example we provided. It is very easy to reach a situation where a host is isolated all virtual machines remain running but are restarted on another host because the connection to the Virtual SAN datastore is lost.

> **Basic design principle:** Configure your Isolation Address and your Isolation Policy accordingly. We recommend selecting “power off” as the Isolation Policy and a reliable pingable device as the isolation address. It is recommended to configure the Isolation Policy to “power off”.

What about things like heartbeat datastores and the folder structure that exists on a VMFS datastore, has any of that changed with Virtual SAN. Yes it has. First of all, in a “Virtual SAN” only environment the concept of Hearbeat Datastores is not used at all. The reason for this is straight forward, as HA and Virtual SAN share the same network it is safe to assume that when the HA heartbeat is lost because of a network failure so is access to the Virtual SAN datastore. Only in an environment where there is also traditional storage the heartbeat datastores will be configured, leveraging those traditional datastores as a heartbeat datastore. Note that we do not feel there is a reason to introduce traditional storage just to provide HA this functionality, HA and Virtual SAN work perfectly fine without heartbeat datastores.

Normally HA metadata is stored in the root of the datastore, for Virtual SAN this is different as the metadata is stored in the VMs namespace object. The protectedlist is held in memory and updated automatically when VMs are powered on or off.

Now you may wonder, what happens when there is an isolation? How does HA know where to start the VM that is impacted? Lets take a look at a partition scenario.

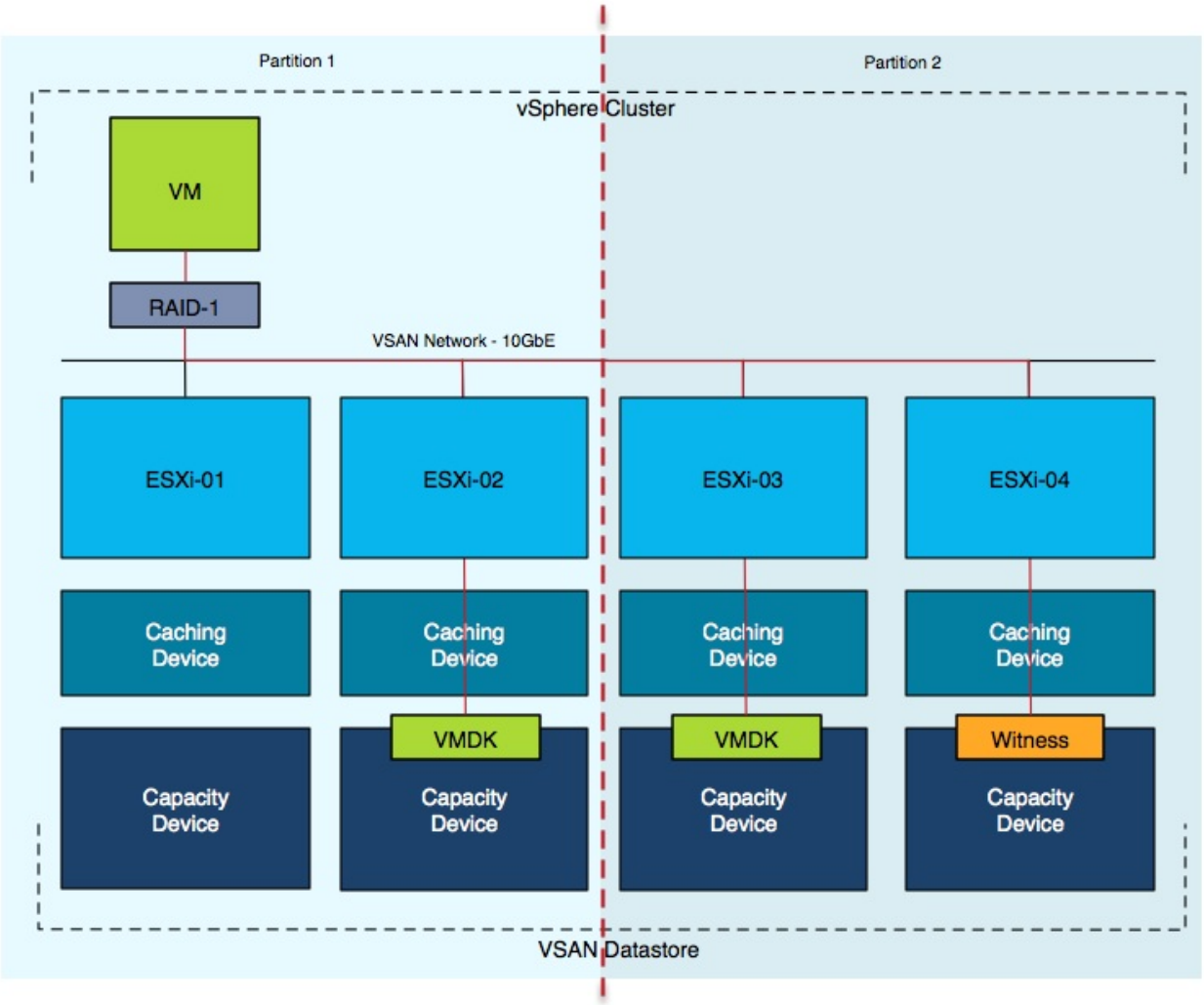


Figure 30 - VSAN Partition scenario

In this scenario there a network problem has caused a cluster partition. Where a VM is restarted is determined by which partition owns the virtual machine files. Within a VSAN cluster this is fairly straight forward. There are two partitions, one of which is running the VM with its VMDK and the other partition has a VMDK replica and a witness. Guess what happens? Right, VSAN uses the witness to see which partition has quorum and based on that result, one of the two partitions will win. In this case, Partition 2 has more than 50% of the components of this object and as such is the winner. This means that the VM will be

restarted on either “esxi-03″ or “esxi-04″ by vSphere HA. Note that the VM in Partition 1 will be powered off only if you have configured the isolation response to do so. We would like to stress that this is highly recommended! (Isolation response –> power off)

One final thing which is different for Virtual SAN is how a partition is handled in a stretched cluster configuration. In a regular stretched cluster configuration using VMFS/NFS based storage VMs impacted by APD or PDL will be killed by HA. With VSAN this is slightly different. HA VMCP in 6.0 is not supported with VSAN unfortunately. VSAN has its own mechanism. VSAN recognizes when a VM running on a group of hosts has no access to **any** of the components. When this is the case VSAN will simply kill the impacted VM. You can disable this behavior, although we do not recommend doing this, by setting the advanced host setting called `VSAN.AutoTerminateGhostVm` to 0.

HA and Virtual Volumes

Let us start with first describing what Virtual Volumes is and what value it brings for an administrator. Virtual Volumes was developed to make your life (vSphere admin) and that of the storage administrator easier. This is done by providing a framework that enables the vSphere administrator to assign policies to virtual machines or virtual disks. In these policies capabilities of the storage array can be defined. These capabilities can be things like snapshotting, deduplication, raid-level, thin / thick provisioning etc. What is offered to the vSphere administrator is up to the Storage administrator, and of course up to what the storage system can offer to begin with. In the below screenshot we show an example for instance of some of the capabilities Nimble exposes through policy.

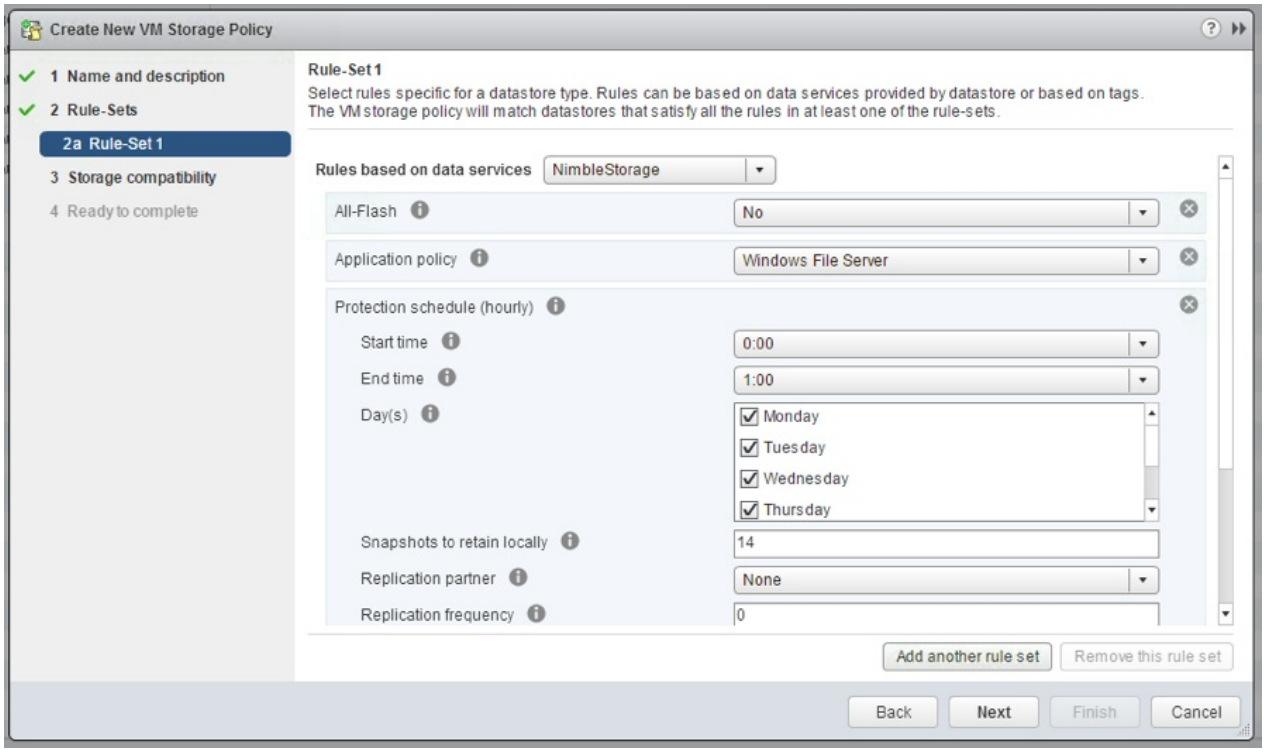


Figure 31 - Capabilities exposed by Nimble array

When a virtual machine is deployed and a policy is assigned then the storage system will enable certain functionality of the array based on what was specified in the policy. So no longer a need to assign capabilities to a LUN which holds many VMs, but rather a per VM or even per VMDK level control. So how does this work? Well lets take a look at an architectural diagram first.

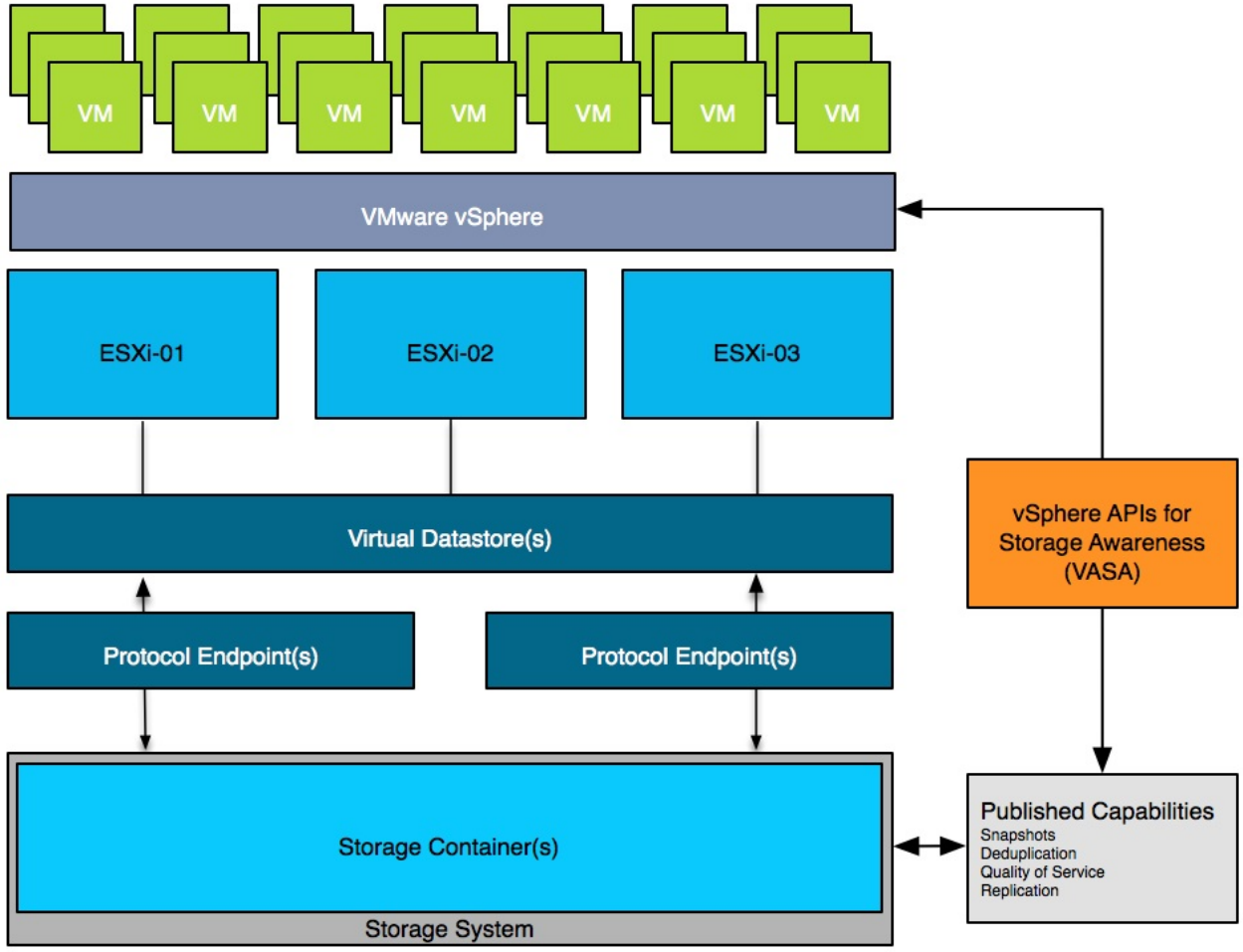


Figure 32 - Virtual Volumes Architecture

The diagram shows a couple of components which are important in the VVol architecture. Lets list them out:

- Protocol Endpoints aka PE
- Virtual Datastore and a Storage Container
- Vendor Provider / VASA
- Policies
- Virtual Volumes

Lets take a look at all of these three in the above order. Protocol Endpoints, what are they?

Protocol Endpoints are literally the access point to your storage system. All IO to virtual volumes is proxied through a Protocol Endpoint and you can have 1 or more of these per storage system, if your storage system supports having multiple of course. (Implementations of different vendors will vary.) PEs are compatible with different protocols (FC, FCoE, iSCSI, NFS) and if you ask me that whole discussion with Virtual Volumes will come to an end. You

could see a Protocol Endpoint as a “mount point” or a device, and yes they will count towards your maximum number of devices per host (256). (Virtual Volumes it self won’t count towards that!)

Next up is the **Storage Container**. This is the place where you store your virtual machines, or better said where your virtual volumes end up. The Storage Container is a storage system logical construct and is represented within vSphere as a “virtual datastore”. You need 1 per storage system, but you can have many when desired. To this Storage Container you can apply capabilities. So if you like your virtual volumes to be able to use array based snapshots then the storage administrator will need to assign that capability to the storage container. Note that a storage administrator can grow a storage container without even informing you. A storage container isn’t formatted with VMFS or anything like that, so you don’t need to increase the volume in order to use the space.

But how does vSphere know which container is capable of doing what? In order to discover a storage container and its capabilities we need to be able to talk to the storage system first. This is done through the **vSphere APIs for Storage Awareness**. You simply point vSphere to the **Vendor Provider** and the vendor provider will report to vSphere what’s available, this includes both the storage containers as well as the capabilities they possess. Note that a single Vendor Provider can be managing multiple storage systems which in its turn can have multiple storage containers with many capabilities. These vendor providers can also come in different flavours, for some storage systems it is part of their software but for others it will come as a virtual appliance that sits on top of vSphere.

Now that vSphere knows which systems there are, what containers are available with which capabilities you can start creating **policies**. These policies can be a combination of capabilities and will ultimately be assigned to virtual machines or virtual disks even. You can imagine that in some cases you would like Quality of Service enabled to ensure performance for a VM while in other cases it isn’t as relevant but you need to have a snapshot every hour. All of this is enabled through these policies. No longer will you be maintaining that spreadsheet with all your LUNs and which data service were enabled and what not, no you simply assign a policy. (Yes, a proper naming scheme will be helpful when defining policies.) When requirements change for a VM you don’t move the VM around, no you change the policy and the storage system will do what is required in order to make the VM (and its disks) compliant again with the policy. Not the VM really, but the Virtual Volumes.

Okay, those are the basics, now what about Virtual Volumes and vSphere HA. What changes when you are running Virtual Volumes, what do you need to keep in mind when running Virtual Volumes when it comes to HA?

First of all, let me mention this, in some cases storage vendors have designed a solution where the "vendor provider" isn't designed in an HA fashion (VMware allows for Active/Active, Active/Standby or just "Active" as in a single instance). Make sure to validate

what kind of implementation your storage vendor has, as the Vendor Provider needs to be available when powering on VMs. The following quote explains why:

> *When a Virtual Volume is created, it is not immediately accessible for IO. To Access Virtual Volumes, vSphere needs to issue a "Bind" operation to a VASA Provider (VP), which creates IO access point for a Virtual Volume on a Protocol Endpoint (PE) chosen by a VP. A single PE can be the IO access point for multiple Virtual Volumes. "Unbind" Operation will remove this IO access point for a given Virtual Volume.*

That is the "Virtual Volumes" implementation aspect, but of course things have also changed from a vSphere HA point of view. No longer do we have VMFS or NFS datastores to store files on or use for heartbeating. What changes from that perspective. First of all a VM is carved up in different Virtual Volumes:

- VM Configuration
- Virtual Machine Disk's
- Swap File
- Snapshot (if there are any)

Besides these different types of objects, when vSphere HA is enabled there also is a volume used by vSphere HA and this volume will contain all the metadata which is normally stored under " `/<root of datastore>/.vSphere-HA/<cluster-specific-directory>/` " on regular VMFS. For each Fault Domain a seperate folder will be created in this VVol as shown in the screenshot below.

```
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.1afaad68-d4fb-438f-80cb-8a61250a9cb6] ls -lah
total 485384
drwxr-xr-t    1 root     root        1.4K Apr 15 09:36 .
drwxr-xr-x    1 root     root         512 Apr 15 10:04 ..
-r--------    1 root     root      128.0K Apr 15 09:34 .fbb.sf
-r--------    1 root     root       79.8M Apr 15 09:34 .fdc.sf
-r--------    1 root     root        1.1M Apr 15 09:34 .pb2.sf
-r--------    1 root     root      256.0M Apr 15 09:34 .pbc.sf
-r--------    1 root     root      128.3M Apr 15 09:34 .sbc.sf
drwx------    1 root     root         280 Apr 15 09:34 .sdd.sf
-r--------    1 root     root        4.0M Apr 15 09:34 .vh.sf
drwx------    1 root     root        1.5K Apr 15 09:58 FDM-97FCC0A1-D94C-4023-9135-A7ED9914D4B0-26-5913ec2-vcsa-05
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.1afaad68-d4fb-438f-80cb-8a61250a9cb6]
```

Figure 33 - Single FDM cluster on VVol array

All VM related HA files which normally would be under the VM folder, like for instance the power-on file, heartbeat files and the protectedlist, are now stored in the VM Configuration VVol object. Conceptually speaking similar to regular VMFS, implementation wise however completely different.

```
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.1afaad68-d4fb-438f-80cb-8a61250a9cb6/FDM-97FCC0A1-D94C-4023-9135-A7ED9914D4B0-26-5913ec2-vcsa-05] ls -lah
total 1048
drwx------    1 root     root        1.5K Apr 15 09:58 .
drwxr-xr-t    1 root     root        1.4K Apr 15 09:36 ..
-rw-r--r--    1 root     root           8 Apr 15 10:04 host-63-hb
-rw-r--r--    1 root     root        1.3K Apr 15 09:58 host-63-poweron
-rw-r--r--    1 root     root           8 Apr 15 10:04 host-70-hb
-rw-r--r--    1 root     root         921 Apr 15 09:36 host-70-poweron
-rw-r--r--    1 root     root           8 Apr 15 10:04 host-74-hb
-rw-r--r--    1 root     root         323 Apr 15 09:36 host-74-poweron
-rw-r--r--    1 root     root           8 Apr 15 10:04 host-78-hb
-rw-r--r--    1 root     root         502 Apr 15 09:36 host-78-poweron
-rw-r--r--    1 root     root        2.0K Apr 15 09:59 protectedlist
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.1afaad68-d4fb-438f-80cb-8a61250a9cb6/FDM-97FCC0A1-D94C-4023-9135-A7ED9914D4B0-26-5913ec2-vcsa-05]
```

Figure 34 - VVol FDM files

The power-off file however, which is used to indicate that a VM has been powered-off due to an isolation event, is not stored under the .vSphere-HA folder any longer, but is stored in the VM config VVol (in the UI exposed as the VVol VM folder) as shown in the screenshot below. The same applies for VSAN, where it is now stored in the VM Namespace object, and for traditional storage (NFS or VMFS) it is stored in the VM folder. This change was made when Virtual Volumes was introduced and done to keep the experience consistent across storage platforms.

```
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.f21565c0-6e5c-4fc5-999d-a64c0d477b1c] ls -lah
total 488464
drwxr-xr-t    1 root     root         3.0K Apr 15 09:58 .
drwxr-xr-x    1 root     root          512 Apr 15 10:56 ..
drwxr-xr-x    1 root     root          420 Apr 15 10:35 .dvsData
-r--------    1 root     root       128.0K Apr 15 09:52 .fbb.sf
-r--------    1 root     root        79.8M Apr 15 09:52 .fdc.sf
-r--------    1 root     root         1.1M Apr 15 09:52 .pb2.sf
-r--------    1 root     root       256.0M Apr 15 09:52 .pbc.sf
-rw-------    1 root     root            0 Apr 15 09:52 .rfc4122.c492c080-6a96-483c-8559-ca45333faecf.lck
-rw-------    1 root     root          423 Apr 15 09:52 .rfc4122.c492c080-6a96-483c-8559-ca45333faecf.meta
-r--------    1 root     root       128.3M Apr 15 09:52 .sbc.sf
drwx------    1 root     root
-rw-------    1 root     root           55 Apr 15 10:56 .vSphereHA-poweroff
-r--------    1 root     root
-rw-r--r--    1 root     root       165.6K Apr 15 10:35 vmware-1.log
-rw-r--r--    1 root     root       166.4K Apr 15 10:56 vmware.log
-rw-r--r--    1 root     root           90 Apr 15 10:35 win-file-srv-01-59643fe5.hlog
-rw-------    1 root     root         8.5K Apr 15 10:56 win-file-srv-01.nvram
-rw-------    1 root     root          563 Apr 15 09:52 win-file-srv-01.vmdk
-rw-r--r--    1 root     root            0 Apr 15 09:52 win-file-srv-01.vmsd
-rwxr-xr-x    1 root     root         2.9K Apr 15 10:56 win-file-srv-01.vmx
[root@esxi-hp-06:/vmfs/volumes/vvol:0000000100004001-84590195c951be61/rfc4122.f21565c0-6e5c-4fc5-999d-a64c0d477b1c]
```

Figure 35 - VVol based poweroff file

And that explains the differences between traditional storage systems using VMFS / NFS and new storage systems leveraging Virtual Volumes or even a full VSAN based solution.

Adding Resiliency to HA (Network Redundancy)

In the previous chapter we extensively covered both Isolation Detection, which triggers the selected Isolation Response and the impact of a false positive. The Isolation Response enables HA to restart virtual machines when “Power off” or “Shut down” has been selected and the host becomes isolated from the network. However, this also means that it is possible that, without proper redundancy, the Isolation Response may be unnecessarily triggered. This leads to downtime and should be prevented.

To increase resiliency for networking, VMware implemented the concept of NIC teaming in the hypervisor for both VMkernel and virtual machine networking. When discussing HA, this is especially important for the Management Network.

> *NIC teaming is the process of grouping together several physical NICs into one single logical NIC, which can be used for network fault tolerance and load balancing.*

Using this mechanism, it is possible to add redundancy to the Management Network to decrease the chances of an isolation event. This is, of course, also possible for other “Portgroups” but that is not the topic of this chapter or book. Another option is configuring an additional Management Network by enabling the “management network” tick box on another VMkernel port. A little understood fact is that if there are multiple VMkernel networks on the same subnet, HA will use all of them for management traffic, even if only one is specified for management traffic!

Although there are many configurations possible and supported, we recommend a simple but highly resilient configuration. We have included the vMotion (VMkernel) network in our example as combining the Management Network and the vMotion network on a single vSwitch is the most commonly used configuration and an industry accepted best practice.

Requirements:

- 2 physical NICs
- VLAN trunking

Recommended:

- 2 physical switches
- If available, enable “link state tracking” to ensure link failures are reported

The vSwitch should be configured as follows:

- vSwitch0: 2 Physical NICs (vmnic0 and vmnic1).
- 2 Portgroups (Management Network and vMotion VMkernel).
- Management Network active on vmnic0 and standby on vmnic1.
- vMotion VMkernel active on vmnic1 and standby on vmnic0.
- Failback set to No.

Each portgroup has a VLAN ID assigned and runs dedicated on its own physical NIC; only in the case of a failure it is switched over to the standby NIC. We highly recommend setting failback to “No” to avoid chances of an unwanted isolation event, which can occur when a physical switch routes no traffic during boot but the ports are reported as “up”. (NIC Teaming Tab)

Pros: Only 2 NICs in total are needed for the Management Network and vMotion VMkernel, especially useful in blade server environments. Easy to configure.

Cons: Just a single active path for heartbeats.

The following diagram depicts this active/standby scenario:

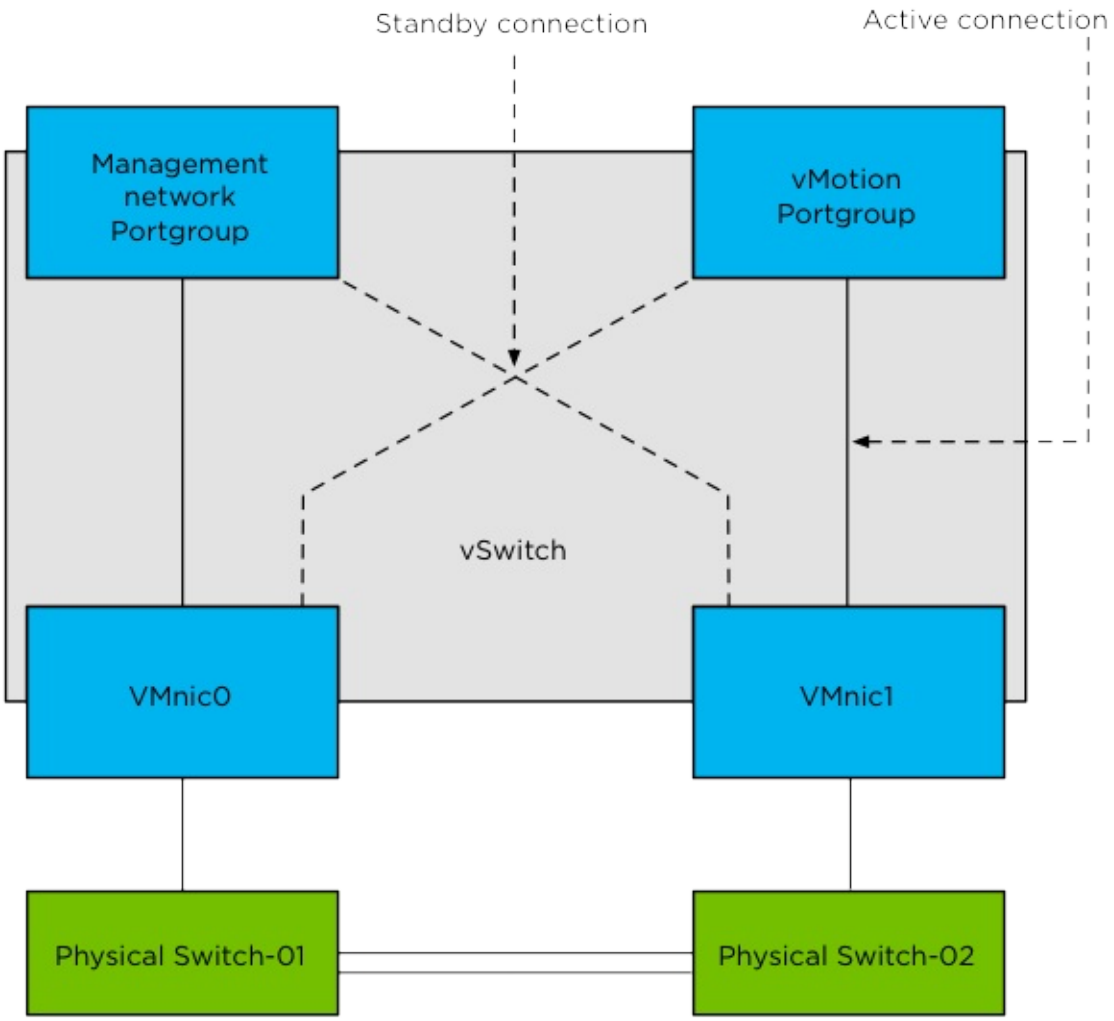


Figure 36 - Active-Standby Management Network design

To increase resiliency, we also recommend implementing the following advanced settings and using NIC ports on different PCI busses – preferably NICs of a different make and model. When using a different make and model, even a driver failure could be mitigated.

Advanced Settings: `das.isolationaddressX = <ip-address>`

The isolation address setting is discussed in more detail in the section titled "Fundamental Concepts". In short; it is the IP address that the HA agent pings to identify if the host is completely isolated from the network or just not receiving any heartbeats. If multiple VMkernel networks on different subnets are used, it is recommended to set an isolation address per network to ensure that each of these will be able to validate isolation of the host.

> **Basic design principle:** Take advantage of some of the basic features vSphere has to offer like NIC teaming. Combining different physical NICs will increase overall resiliency of your solution.

Corner Case Scenario: Split-Brain

A split brain scenario is a scenario where a single virtual machine is powered up multiple times, typically on two different hosts. This is possible in the scenario where the isolation response is set to “Disabled” and network based storage, like NFS / iSCSI and even Virtual SAN, is used. This situation can occur during a full network isolation, which may result in the lock on the virtual machine’s VMDK being lost, enabling HA to actually power up the virtual machine. As the virtual machine was not powered off on its original host (isolation response set to “Disabled”), it will exist in memory on the isolated host and in memory with a disk lock on the host that was requested to restart the virtual machine.

Keep in mind that this truly is a corner case scenario which is very unlikely to occur in most environments. In case it does happen, HA relies on the “lost lock detection” mechanism to mitigate this scenario. In short ESXi detects that the lock on the VMDK has been lost and, when the datastore becomes accessible again and the lock cannot be reacquired, issues a question whether the virtual machine should be powered off; HA automatically answers the question with Yes. However, you will only see this question if you directly connect to the ESXi host during the failure. HA will generate an event for this auto-answered question though.

As stated above the question will be auto-answered and the virtual machine will be powered off to recover from the split brain scenario. The question still remains: in the case of an isolation with iSCSI or NFS, should you power off virtual machines or leave them powered on?

As just explained, HA will automatically power off your original virtual machine when it detects a split-brain scenario. This process however is not instantaneous and as such it is recommended to use the isolation response of “Power Off” or “Disabled". We also recommend increasing heartbeat network resiliency to avoid getting in to this situation. We will discuss the options you have for enhancing Management Network resiliency in the next chapter.

Link State Tracking

This was already briefly mentioned in the list of recommendations, but this feature is something we would like to emphasize. We have noticed that people often forget about this even though many switches offer this capability, especially in blade server environments.

Link state tracking will mirror the state of an upstream link to a downstream link. Let’s clarify that with a diagram.

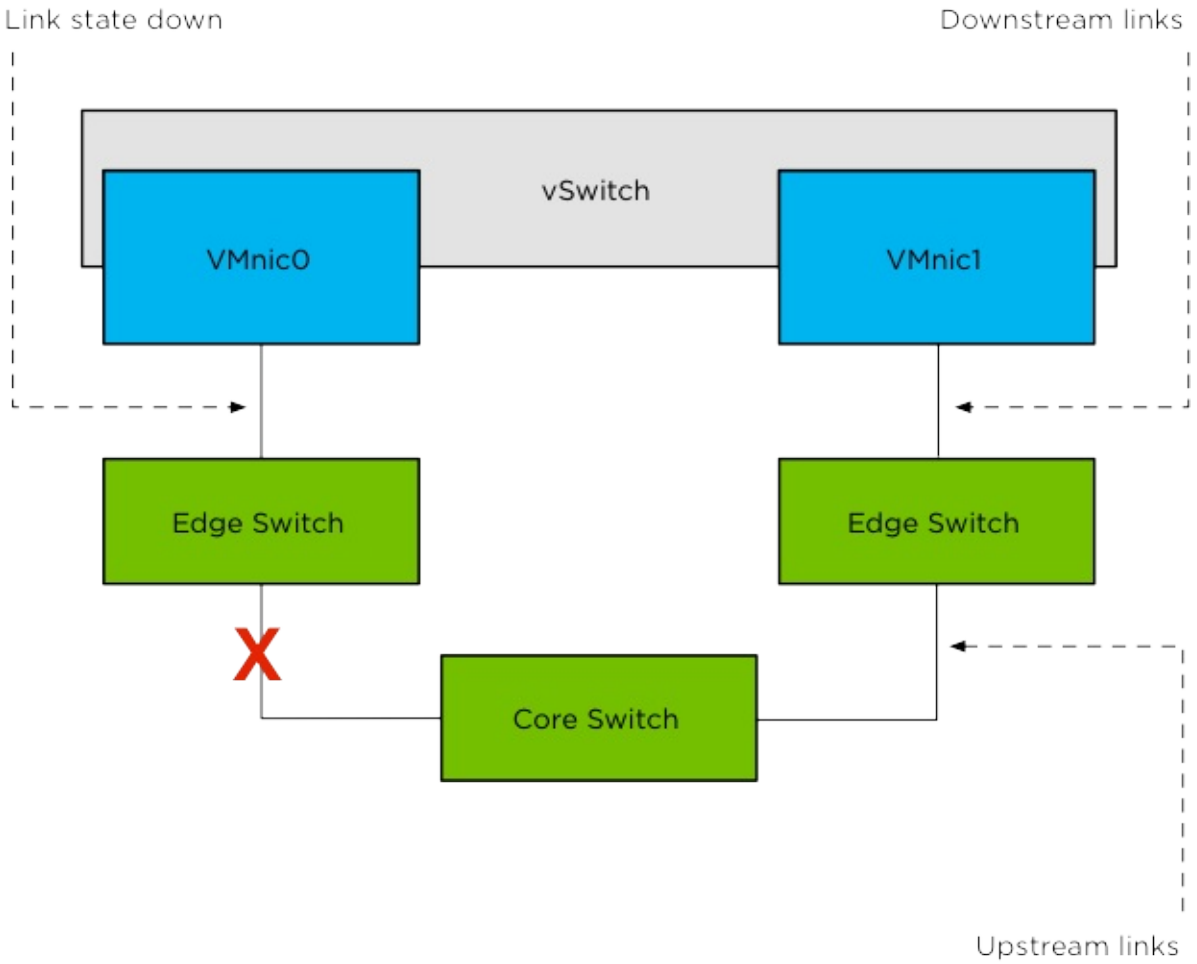


Figure 37 - Link State tracking mechanism

The diagram above depicts a scenario where an uplink of a “Core Switch” has failed. Without Link State Tracking, the connection from the “Edge Switch” to vmnic0 will be reported as up. With Link State Tracking enabled, the state of the link on the “Edge Switch” will reflect the state of the link of the “Core Switch” and as such be marked as “down”. You might wonder why this is important but think about it for a second. Many features that vSphere offer rely on networking and so do your virtual machines. In the case where the state is not reflected, some functionality might just fail, for instance network heartbeating could fail if it needs to flow through the core switch. We call this a ‘black hole’ scenario: the host sends traffic down a path that it believes is up, but the traffic never reaches its destination due to the failed upstream link.

> **Basic design principle:** Know your network environment, talk to the network administrators and ensure advanced features like Link State Tracking are used when possible to increase resiliency.

Admission Control

Admission Control is more than likely the most misunderstood concept vSphere holds today and because of this it is often disabled. However, Admission Control is a must when availability needs to be guaranteed and isn't that the reason for enabling HA in the first place?

What is HA Admission Control about? Why does HA contain this concept called Admission Control? The "Availability Guide" a.k.a HA bible states the following:

> *vCenter Server uses admission control to ensure that sufficient resources are available in a cluster to provide failover protection and to ensure that virtual machine resource reservations are respected.*

Please read that quote again and especially the first two words. Indeed it is vCenter that is responsible for Admission Control, contrary to what many believe. Although this might seem like a trivial fact it is important to understand that this implies that Admission Control will not disallow HA initiated restarts. HA initiated restarts are done on a host level and not through vCenter.

As said, Admission Control guarantees that capacity is available for an HA initiated failover by reserving resources within a cluster. It calculates the capacity required for a failover based on available resources. In other words, if a host is placed into maintenance mode or disconnected, it is taken out of the equation. This also implies that if a host has failed or is not responding but has not been removed from the cluster, it is still included in the equation. "Available Resources" indicates that the virtualization overhead has already been subtracted from the total amount.

To give an example; VMkernel memory is subtracted from the total amount of memory to obtain the memory available memory for virtual machines. There is one gotcha with Admission Control that we want to bring to your attention before drilling into the different policies. When Admission Control is enabled, HA will in no way violate availability constraints. This means that it will always ensure multiple hosts are up and running and this applies for manual maintenance mode actions and, for instance, to VMware Distributed Power Management. So, if a host is stuck trying to enter Maintenance Mode, remember that it might be HA which is not allowing Maintenance Mode to proceed as it would violate the Admission Control Policy. In this situation, users can manually vMotion virtual machines off the host or temporarily disable admission control to allow the operation to proceed.

But what if you use something like Distributed Power Management (DPM), would that place all hosts in standby mode to reduce power consumption? No, DPM is smart enough to take hosts out of standby mode to ensure enough resources are available to provide for HA initiated failovers. If by any chance the resources are not available, HA will wait for these resources to be made available by DPM and then attempt the restart of the virtual machines. In other words, the retry count (5 retries by default) is not wasted in scenarios like these.

Admission Control Policy

The Admission Control Policy dictates the mechanism that HA uses to guarantee enough resources are available for an HA initiated failover. This section gives a general overview of the available Admission Control Policies. The impact of each policy is described in the following section, including our recommendation. HA has three mechanisms to guarantee enough capacity is available to respect virtual machine resource reservations.

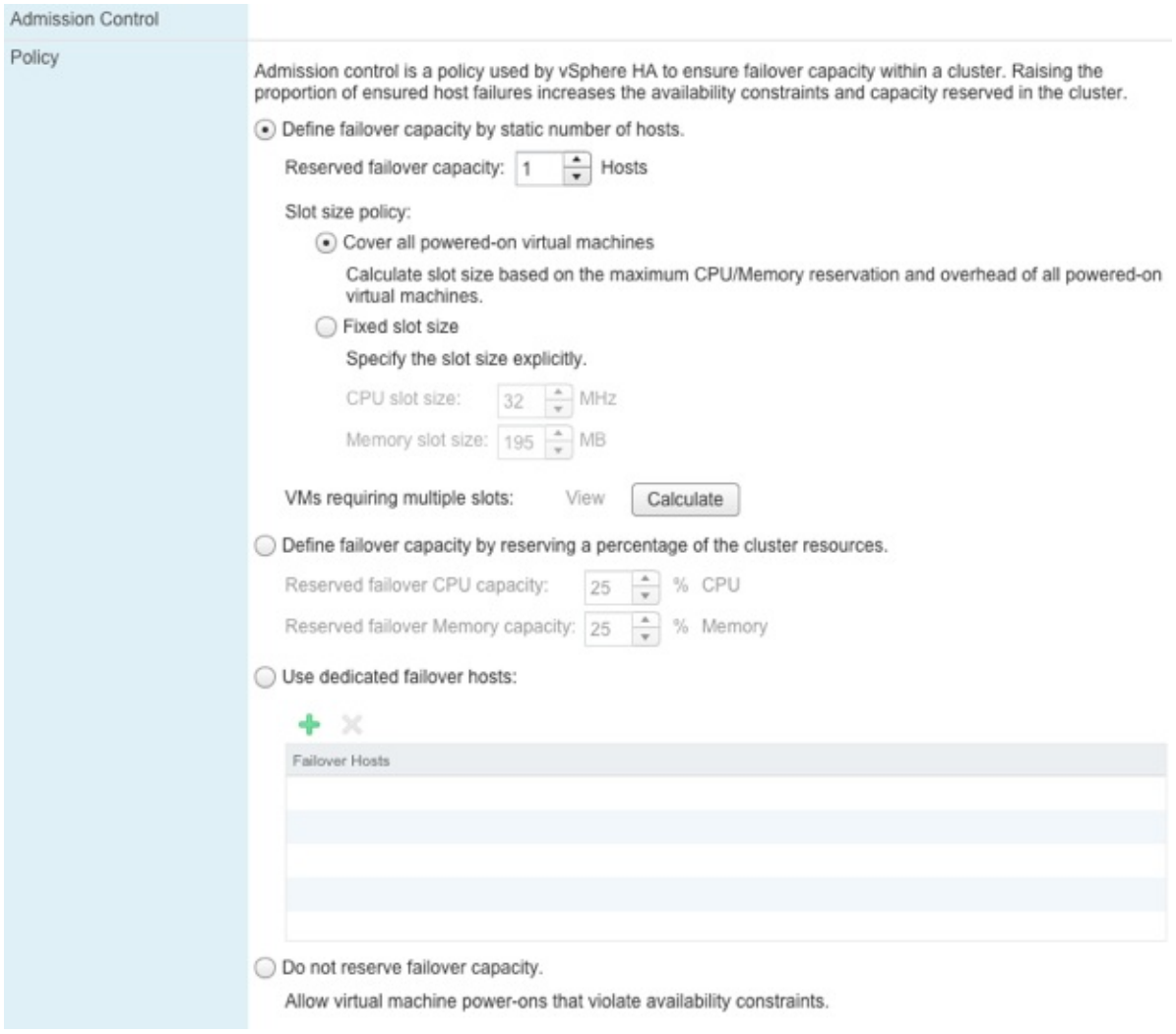


Figure 38 - Admission control policy

Below we have listed all three options currently available as the Admission Control Policy. Each option has a different mechanism to ensure resources are available for a failover and each option has its caveats.

Admission Control Mechanisms

Each Admission Control Policy has its own Admission Control mechanism. Understanding each of these Admission Control mechanisms is important to appreciate the impact each one has on your cluster design. For instance, setting a reservation on a specific virtual machine can have an impact on the achieved consolidation ratio. This section will take you on a journey through the trenches of Admission Control Policies and their respective mechanisms and algorithms.

Host Failures Cluster Tolerates

The Admission Control Policy that has been around the longest is the “Host Failures Cluster Tolerates” policy. It is also historically the least understood Admission Control Policy due to its complex admission control mechanism.

This admission control policy can be configured in an N-1 fashion. This means that the number of host failures you can specify in a 32 host cluster is 31.

Within the vSphere Web Client it is possible to manually specify the slot size as can be seen in the below screenshot. The vSphere Web Client also allows you to view which virtual machines span multiple slots. This can be very useful in scenarios where the slot size has been explicitly specified, we will explain why in just a second.

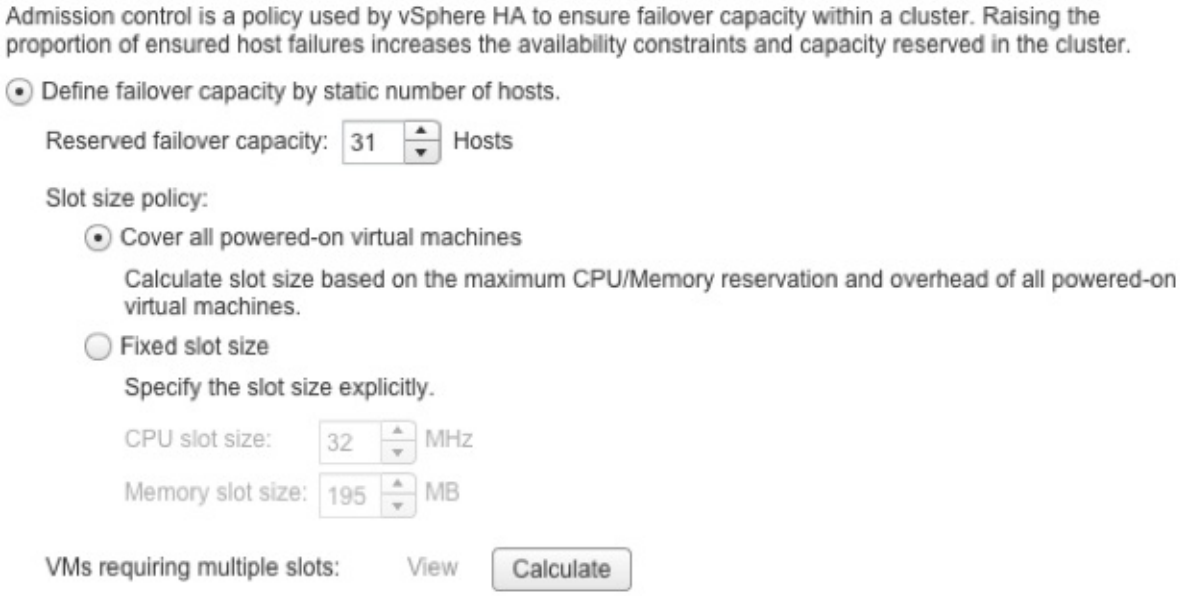


Figure 39 - Host Failures

The so-called “slots” mechanism is used when the “Host failures cluster tolerates” has been selected as the Admission Control Policy. The details of this mechanism have changed several times in the past and it is one of the most restrictive policies; more than likely, it is also the least understood.

Slots dictate how many virtual machines can be powered on before vCenter starts yelling “Out Of Resources!” Normally, a slot represents one virtual machine. Admission Control does not limit HA in restarting virtual machines, it ensures enough unfragmented resources are available to power on all virtual machines in the cluster by preventing “over-commitment”. Technically speaking “over-commitment” is not the correct terminology as Admission Control ensures virtual machine reservations can be satisfied and that all virtual machines’ initial memory overhead requirements are met. Although we have already touched on this, it doesn’t hurt repeating it as it is one of those myths that keeps coming back; **HA initiated failovers are not prone to the Admission Control Policy**. Admission Control is done by vCenter. HA initiated restarts, in a normal scenario, are executed directly on the ESXi host without the use of vCenter. The corner-case is where HA requests DRS (DRS is a vCenter task!) to defragment resources but that is beside the point. Even if resources are low and vCenter would complain, it couldn’t stop the restart from happening.

Let’s dig in to this concept we have just introduced, slots.

> *A slot is defined as a logical representation of the memory and CPU resources that satisfy the reservation requirements for any powered-on virtual machine in the cluster.*

In other words a slot is the worst case CPU and memory **reservation** scenario in a cluster. This directly leads to the first “gotcha.”

HA uses the highest CPU reservation of any given powered-on virtual machine and the highest memory reservation of any given powered-on virtual machine in the cluster. If no reservation of higher than 32 MHz is set, HA will use a default of 32 MHz for CPU. If no memory reservation is set, HA will use a default of 0 MB+memory overhead for memory. (See the VMware vSphere Resource Management Guide for more details on memory overhead per virtual machine configuration.) The following example will clarify what “worst-case” actually means.

Example: If virtual machine “VM1” has 2 GHz of CPU reserved and 1024 MB of memory reserved and virtual machine “VM2” has 1 GHz of CPU reserved and 2048 MB of memory reserved the slot size for memory will be 2048 MB (+ its memory overhead) and the slot size for CPU will be 2 GHz. It is a combination of the highest reservation of both virtual machines that leads to the total slot size. Reservations defined at the Resource Pool level however, will not affect HA slot size calculations.

> **Basic design principle:** Be really careful with reservations, if there’s no need to have them on a per virtual machine basis; don’t configure them, especially when using host failures cluster tolerates. If reservations are needed, resort to resource pool based reservations.

Now that we know the worst-case scenario is always taken into account when it comes to slot size calculations, we will describe what dictates the amount of available slots per cluster as that ultimately dictates how many virtual machines can be powered on in your cluster.

First, we will need to know the slot size for memory and CPU, next we will divide the total available CPU resources of a host by the CPU slot size and the total available memory resources of a host by the memory slot size. This leaves us with a total number of slots for both memory and CPU for a host. The most restrictive number (worst-case scenario) is the number of slots for this host. In other words, when you have 25 CPU slots but only 5 memory slots, the amount of available slots for this host will be 5 as HA always takes the worst case scenario into account to “guarantee” all virtual machines can be powered on in case of a failure or isolation.

The question we receive a lot is how do I know what my slot size is? The details around slot sizes can be monitored on the HA section of the Cluster’s Monitor tab by checking the the “Advanced Runtime Info” section when the “Host Failures” Admission Control Policy is configured.

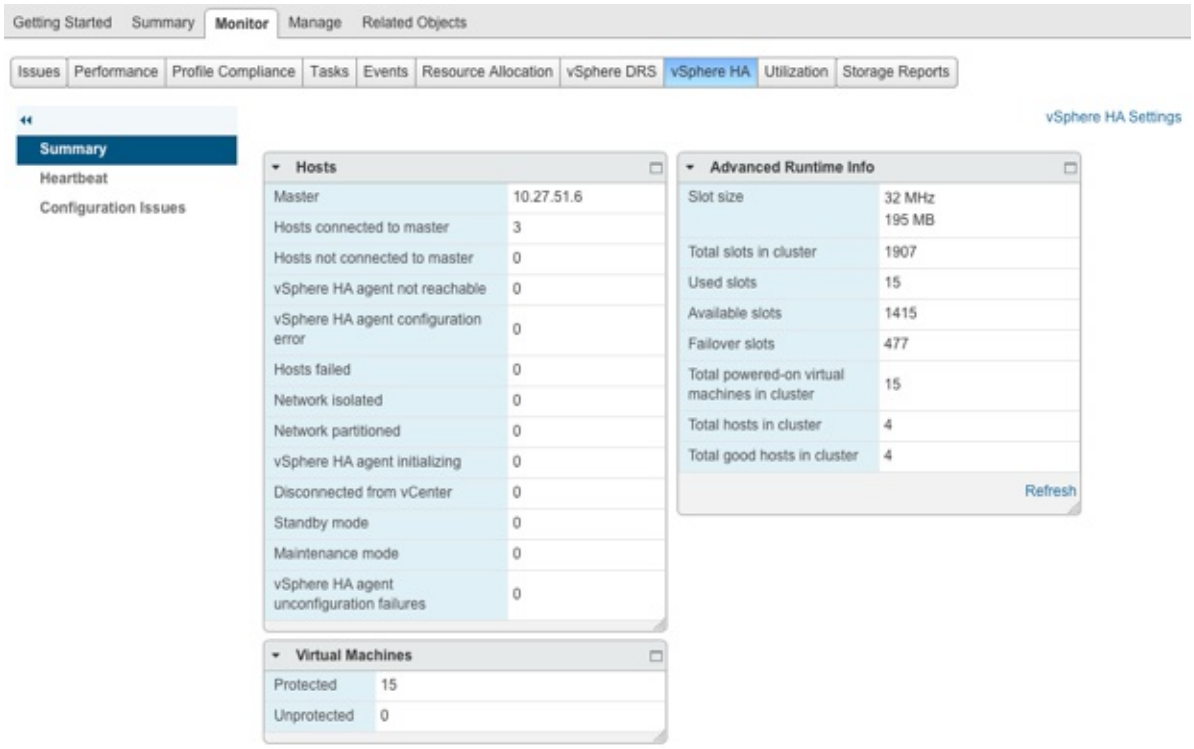


Figure 40 - High Availability cluster monitor section

Advanced Runtime Info will show the specifics the slot size and more useful details such as the number of slots available as depicted in Figure 30.

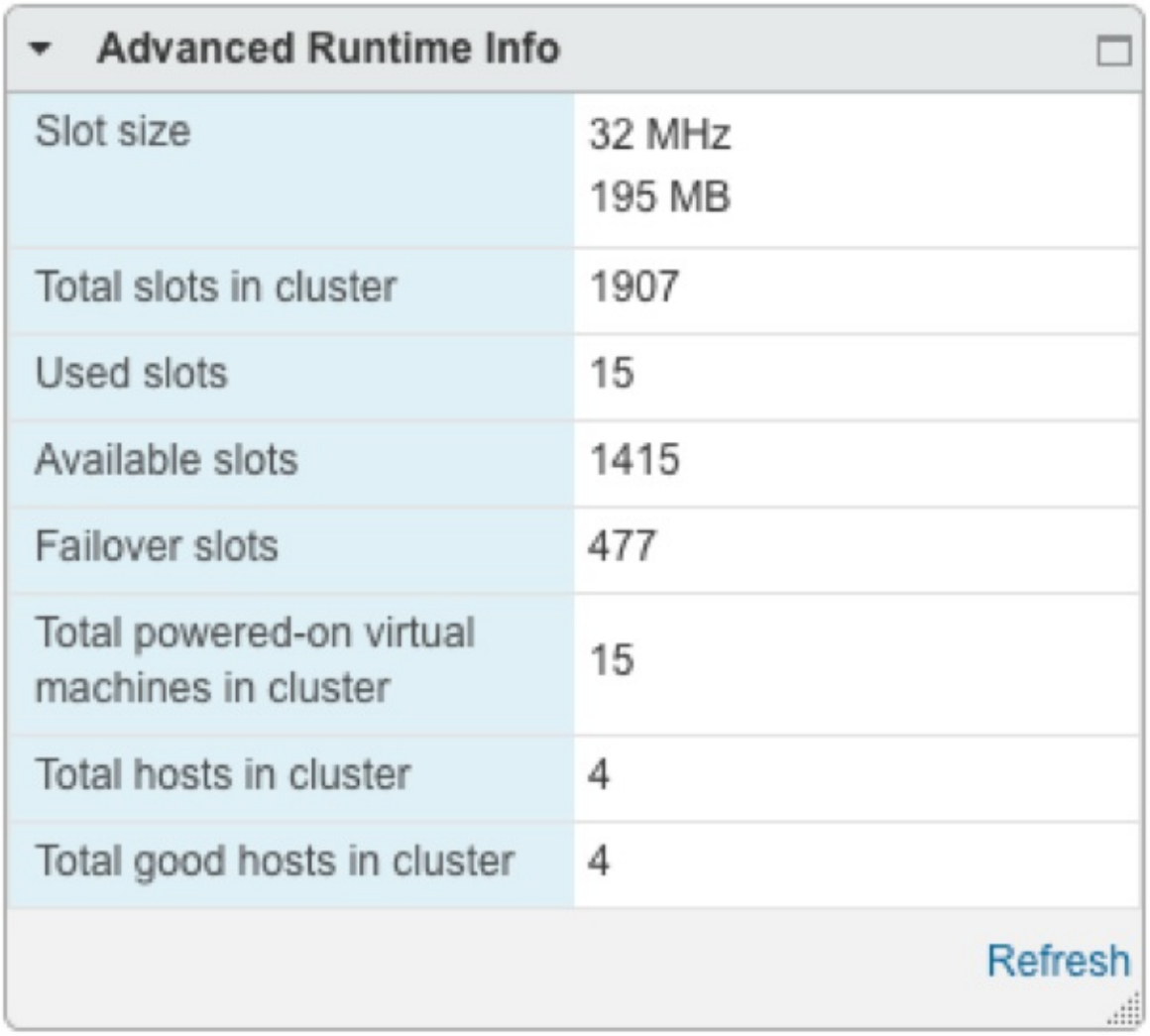


Figure 41 - High Availability advanced runtime info

As you can imagine, using reservations on a per virtual machine basis can lead to very conservative consolidation ratios. However, this is something that is configurable through the Web Client. If you have just one virtual machine with a really high reservation, you can set an explicit slot size by going to “Edit Cluster Services” and specifying them under the Admission Control Policy section as shown in Figure 29.

If one of these advanced settings is used, HA will ensure that the virtual machine that skewed the numbers can be restarted by “assigning” multiple slots to it. However, when you are low on resources, this could mean that you are not able to power on the virtual machine with this reservation because resources may be fragmented throughout the cluster instead of available on a single host. HA will notify DRS that a power-on attempt was unsuccessful and a request will be made to defragment the resources to accommodate the remaining virtual machines that need to be powered on. In order for this to be successful DRS will need to be enabled and configured to fully automated. When not configured to fully automated user action is required to execute DRS recommendations.

The following diagram depicts a scenario where a virtual machine spans multiple slots:

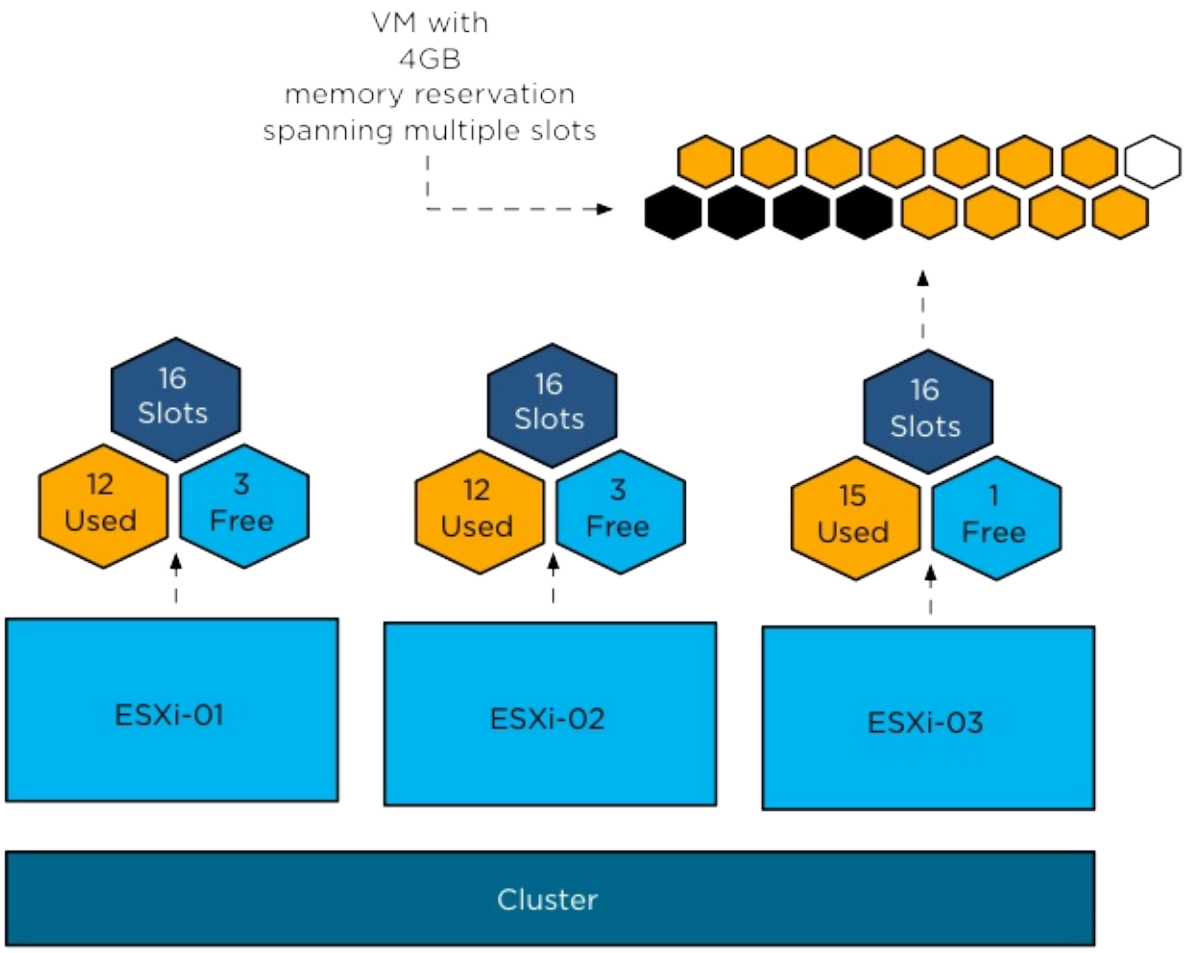


Figure 42 - Virtual machine spanning multiple HA slots

Notice that because the memory slot size has been manually set to 1024 MB, one of the virtual machines (grouped with dotted lines) spans multiple slots due to a 4 GB memory reservation. As you might have noticed, none of the hosts has enough resources available to satisfy the reservation of the virtual machine that needs to failover. Although in total there are enough resources available, they are fragmented and HA will not be able to power-on this particular virtual machine directly but will request DRS to defragment the resources to accommodate this virtual machine's resource requirements.

Admission Control does not take fragmentation of slots into account when slot sizes are manually defined with advanced settings. It will take the number of slots this virtual machine will consume into account by subtracting them from the total number of available slots, but it will not verify the amount of available slots per host to ensure failover. As stated earlier, though, HA will request DRS to defragment the resources. This is by no means a guarantee of a successful power-on attempt.

> **Basic design principle:** Avoid using advanced settings to decrease the slot size as it could lead to more down time and adds an extra layer of complexity. If there is a large discrepancy in size and reservations we recommend using the percentage based admission control policy.

Within the vSphere Web Client there is functionality which enables you to identify virtual machines which span multiple slots, as shown in Figure 29. We highly recommend monitoring this section on a regular basis to get a better understand of your environment and to identify those virtual machines that might be problematic to restart in case of a host failure.

Unbalanced Configurations and Impact on Slot Calculation

It is an industry best practice to create clusters with similar hardware configurations. However, many companies started out with a small VMware cluster when virtualization was first introduced. When the time has come to expand, chances are fairly large the same hardware configuration is no longer available. The question is will you add the newly bought hosts to the same cluster or create a new cluster?

From a DRS perspective, large clusters are preferred as it increases the load balancing opportunities. However there is a caveat for DRS as well, which is described in the DRS section of this book. For HA, there is a big caveat. When you think about it and understand the internal workings of HA, more specifically the slot algorithm, you probably already know what is coming up.

Let's first define the term "unbalanced cluster."

An unbalanced cluster would, for instance, be a cluster with 3 hosts of which one contains substantially more memory than the other hosts in the cluster.

Let's try to clarify that with an example.

Example: What would happen to the total number of slots in a cluster of the following specifications?

- Three host cluster
- Two hosts have 16 GB of available memory
- One host has 32 GB of available memory

The third host is a brand new host that has just been bought and as prices of memory dropped immensely the decision was made to buy 32 GB instead of 16 GB.

The cluster contains a virtual machine that has 1 vCPU and 4 GB of memory. A 1024 MB memory reservation has been defined on this virtual machine. As explained earlier, a reservation will dictate the slot size, which in this case leads to a memory slot size of 1024

MB + memory overhead. For the sake of simplicity, we will calculate with 1024 MB. The following diagram depicts this scenario:

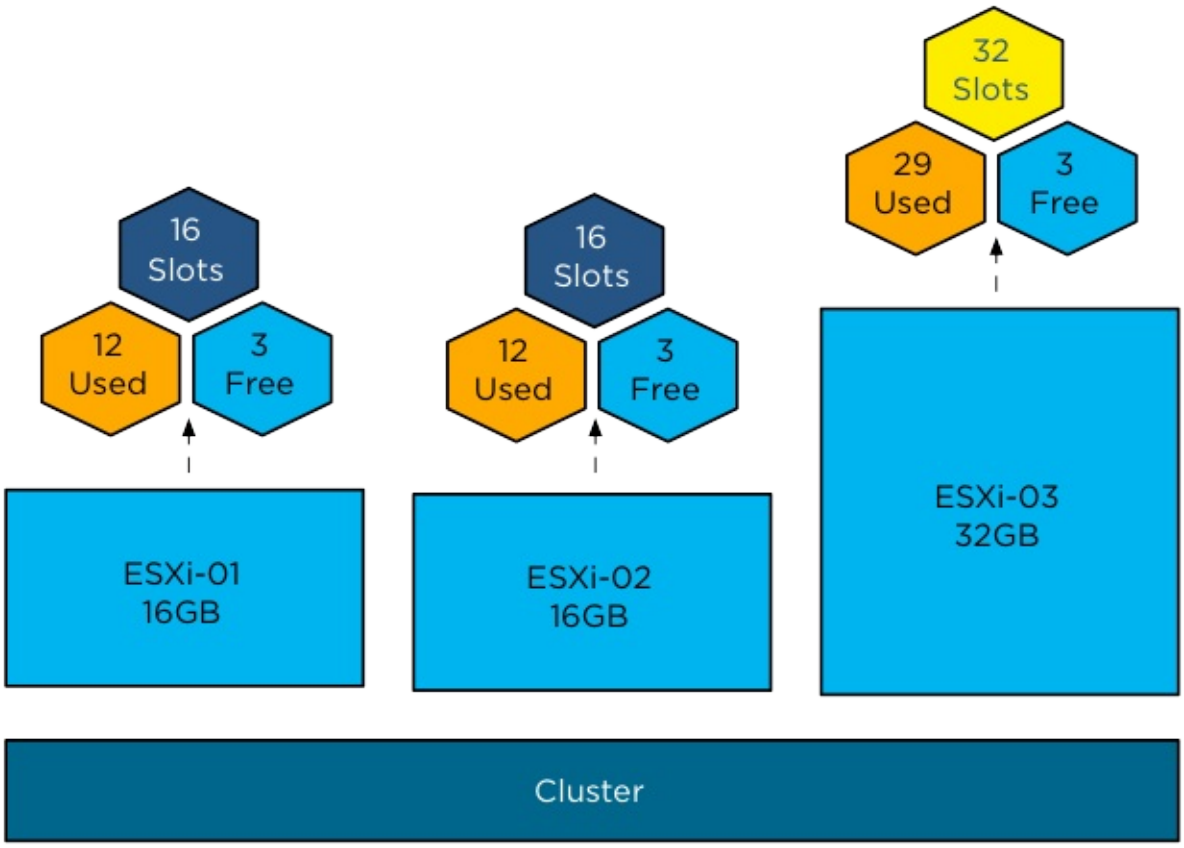


Figure 43 - High Availability memory slot size

When Admission Control is enabled and the number of host failures has been selected as the Admission Control Policy, the number of slots will be calculated per host and the cluster in total. This will result in:

Host	Number of slots
ESXi-01	16 Slots
ESXi-02	16 Slots
ESXi-03	32 Slots

As Admission Control is enabled, **a worst-case scenario is taken into account**. When a single host failure has been specified, this means that the host with the largest number of slots will be taken out of the equation. In other words, for our cluster, this would result in:

```
ESXi-01 + ESXi-02 = 32 slots available
```

Although you have doubled the amount of memory in one of your hosts, you are still stuck with only 32 slots in total. As clearly demonstrated, there is absolutely no point in buying additional memory for a single host when your cluster is designed with Admission Control enabled and the number of host failures has been selected as the Admission Control Policy.

In our example, the memory slot size happened to be the most restrictive; however, the same principle applies when CPU slot size is most restrictive.

> **Basic design principle:** When using admission control, balance your clusters and be conservative with reservations as it leads to decreased consolidation ratios.

Now, what would happen in the scenario above when the number of allowed host failures is to 2? In this case ESXi-03 is taken out of the equation and one of any of the remaining hosts in the cluster is also taken out, resulting in 16 slots. This makes sense, doesn't it?

Can you avoid large HA slot sizes due to reservations without resorting to advanced settings? That's the question we get almost daily and the answer is the "Percentage of Cluster Resources Reserved" admission control mechanism.

Percentage of Cluster Resources Reserved

The Percentage of Cluster Resources Reserved admission control policy is one of the most used admission control policies. The simple reason for this is that it is the least restrictive and most flexible. It is also very easy to configure as shown in the screenshot below.

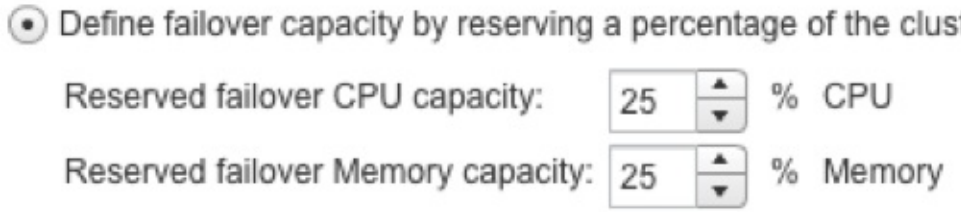


Figure 44 - Setting a different percentage for CPU/Memory

The main advantage of the percentage based Admission Control Policy is that it avoids the commonly experienced slot size issue where values are skewed due to a large reservation. But if it doesn't use the slot algorithm, what does it use?

When you specify a percentage, and let's assume for now that the percentage for CPU and memory will be configured equally, that percentage of the total amount of available resources will stay reserved for HA purposes. First of all, HA will add up all available

resources to see how much it has available (virtualization overhead will be subtracted) in total. Then, HA will calculate how much resources are currently reserved by adding up all reservations for memory and for CPU for all powered on virtual machines.

For those virtual machines that do not have a reservation, a default of 32 MHz will be used for CPU and a default of 0 MB+memory overhead will be used for Memory. (Amount of overhead per configuration type can be found in the “Understanding Memory Overhead” section of the Resource Management guide.)

In other words:

((total amount of available resources – total reserved virtual machine resources)/total amount of available resources) <= (percentage HA should reserve as spare capacity)

Total reserved virtual machine resources includes the default reservation of 32 MHz and the memory overhead of the virtual machine.

Let’s use a diagram to make it a bit clearer:

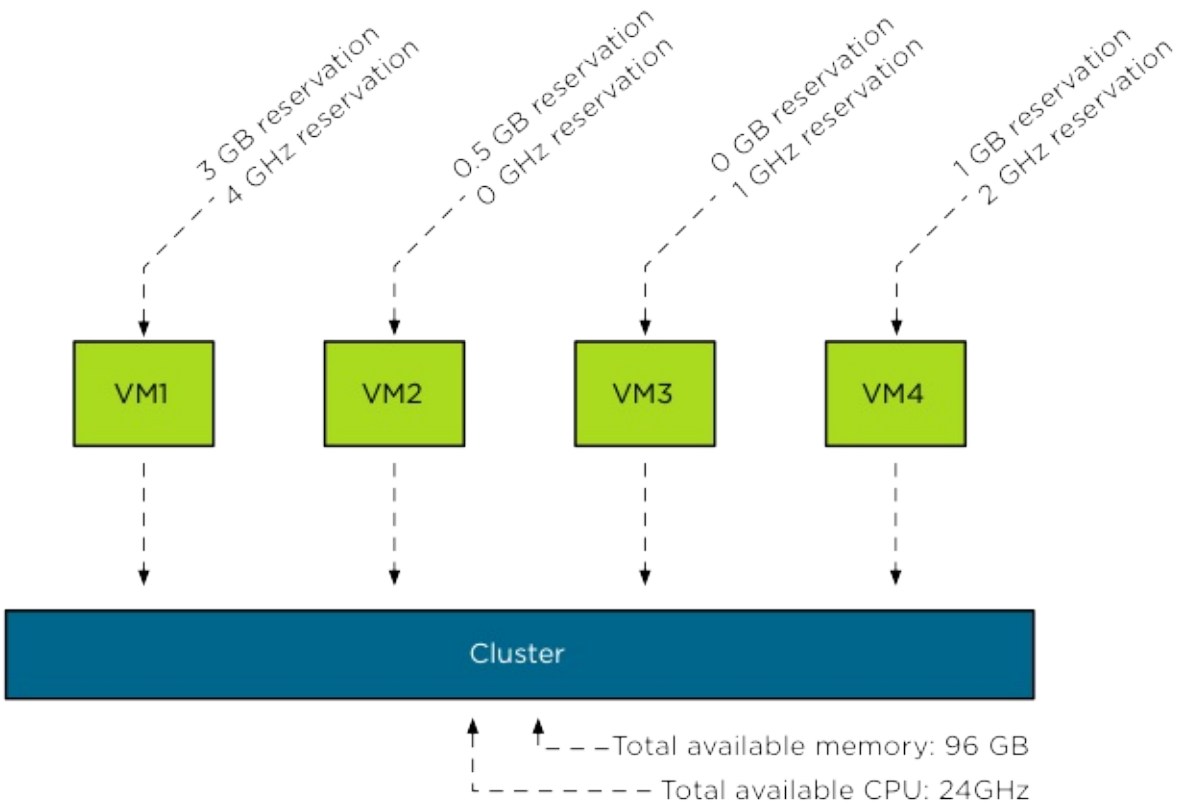


Figure 45 - Percentage of cluster resources reserved

Total cluster resources are 24GHz (CPU) and 96GB (MEM). This would lead to the following calculations:

```
((24 GHz - (2 GHz + 1 GHz + 32 MHz + 4 GHz)) / 24 GHz) = 69 % available
```

```
((96 GB - (1,1 GB + 114 MB + 626 MB + 3,2 GB)/96 GB= 85 % available
```

As you can see, the amount of memory differs from the diagram. Even if a reservation has been set, the amount of memory overhead is added to the reservation. This example also demonstrates how keeping CPU and memory percentage equal could create an imbalance. Ideally, of course, the hosts are provisioned in such a way that there is no CPU/memory imbalance. Experience over the years has proven, unfortunately, that most environments run out of memory resources first and this might need to be factored in when calculating the correct value for the percentage. However, this trend might be changing as memory is getting cheaper every day.

In order to ensure virtual machines can always be restarted, Admission Control will constantly monitor if the policy has been violated or not. Please note that this Admission Control process is part of vCenter and not of the ESXi host! When one of the thresholds is reached, memory or CPU, Admission Control will disallow powering on any additional virtual machines as that could potentially impact availability. These thresholds can be monitored on the HA section of the Cluster's summary tab.



Figure 46 - High Availability summary

If you have an unbalanced cluster (hosts with different sizes of CPU or memory resources), your percentage should be equal or preferably larger than the percentage of resources provided by the largest host. This way you ensure that all virtual machines residing on this host can be restarted in case of a host failure.

As earlier explained, this Admission Control Policy does not use slots. As such, resources might be fragmented throughout the cluster. Although DRS is notified to rebalance the cluster, if needed, to accommodate these virtual machines resource requirements, a

guarantee cannot be given. We recommend selecting the highest restart priority for this virtual machine (of course, depending on the SLA) to ensure it will be able to boot.

The following example and diagram (Figure 37) will make it more obvious: You have 3 hosts, each with roughly 80% memory usage, and you have configured HA to reserve 20% of resources for both CPU and memory. A host fails and all virtual machines will need to failover. One of those virtual machines has a 4 GB memory reservation. As you can imagine, HA will not be able to initiate a power-on attempt, as there are not enough memory resources available to guarantee the reserved capacity. Instead an event will get generated indicating "not enough resources for failover" for this virtual machine.



Figure 47 - Available resources

> **Basic design principle:** Although HA will utilize DRS to try to accommodate for the resource requirements of this virtual machine a guarantee cannot be given. Do the math; verify that any single host has enough resources to power-on your largest virtual machine. Also take restart priority into account for this/these virtual machine(s).

Failover Hosts

The third option one could choose is to select one or multiple designated Failover hosts. This is commonly referred to as a hot standby.

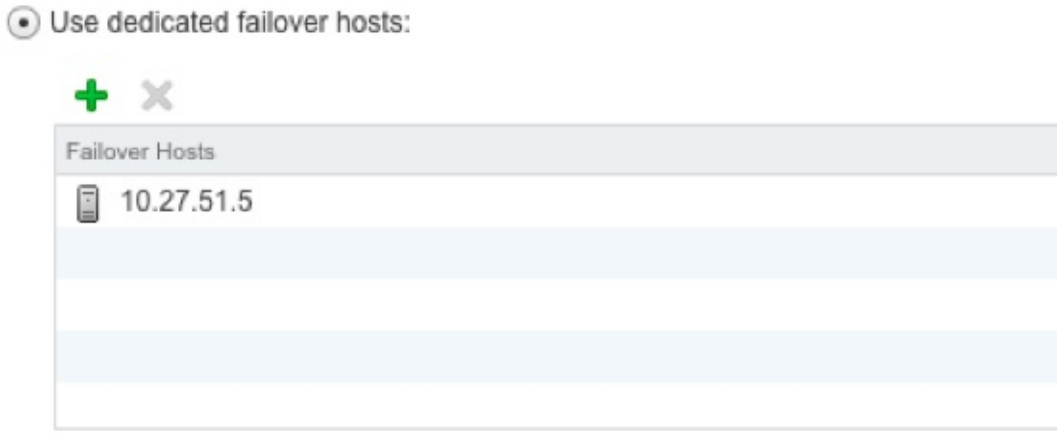


Figure 48 - Select failover hosts Admission Control Policy

It is “what you see is what you get”. When you designate hosts as failover hosts, they will not participate in DRS and you will not be able to run virtual machines on these hosts! Not even in a two host cluster when placing one of the two in maintenance. These hosts are literally reserved for failover situations. HA will attempt to use these hosts first to failover the virtual machines. If, for whatever reason, this is unsuccessful, it will attempt a failover on any of the other hosts in the cluster. For example, when three hosts would fail, including the hosts designated as failover hosts, HA will still try to restart the impacted virtual machines on the host that is left. Although this host was not a designated failover host, HA will use it to limit downtime.

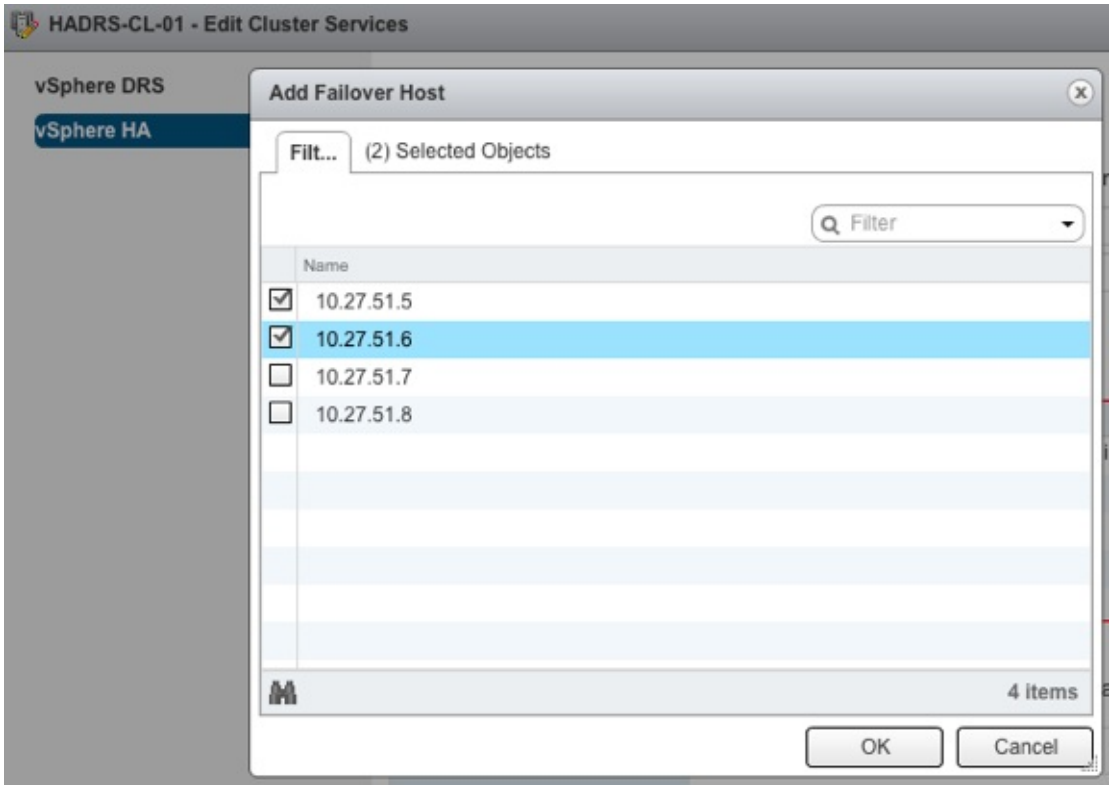


Figure 49 - Select multiple failover hosts

Decision Making Time

As with any decision you make, there is an impact to your environment. This impact could be positive but also, for instance, unexpected. This especially goes for HA Admission Control. Selecting the right Admission Control Policy can lead to a quicker Return On Investment and a lower Total Cost of Ownership. In the previous section, we described all the algorithms and mechanisms that form Admission Control and in this section we will focus more on the design considerations around selecting the appropriate Admission Control Policy for your or your customer's environment.

The first decision that will need to be made is whether Admission Control will be enabled. We generally recommend enabling Admission Control as it is the only way of guaranteeing your virtual machines will be allowed to restart after a failure. It is important, though, that the policy is carefully selected and fits your or your customer's requirements.

```
Basic design principle
Admission control guarantees enough capacity is available for virtual machine failover
. As such we recommend enabling it.
```

Although we already have explained all the mechanisms that are being used by each of the policies in the previous section, we will give a high level overview and list all the pros and cons in this section. On top of that, we will expand on what we feel is the most flexible Admission Control Policy and how it should be configured and calculated.

Host Failures Cluster Tolerates

This option is historically speaking the most used for Admission Control. Most environments are designed with an N+1 redundancy and N+2 is also not uncommon. This Admission Control Policy uses “slots” to ensure enough capacity is reserved for failover, which is a fairly complex mechanism. Slots are based on VM-level reservations and if reservations are not used a default slot size for CPU of 32 MHz is defined and for memory the largest memory overhead of any given virtual machine is used.

Pros:

- Fully automated (When a host is added to a cluster, HA re-calculates how many slots are available.)
- Guarantees failover by calculating slot sizes.

Cons:

- Can be very conservative and inflexible when reservations are used as the largest reservation dictates slot sizes.
- Unbalanced clusters lead to wastage of resources.
- Complexity for administrator from calculation perspective.

Percentage as Cluster Resources Reserved

The percentage based Admission Control is based on per-reservation calculation instead of the slots mechanism. The percentage based Admission Control Policy is less conservative than “Host Failures” and more flexible than “Failover Hosts”.

Pros:

- Accurate as it considers actual reservation per virtual machine to calculate available failover resources.
- Cluster dynamically adjusts when resources are added.

Cons:

- Manual calculations needed when adding additional hosts in a cluster and number of host failures needs to remain unchanged.
- Unbalanced clusters can be a problem when chosen percentage is too low and

resources are fragmented, which means failover of a virtual machine can't be guaranteed as the reservation of this virtual machine might not be available as a block of resources on a single host.

Please note that, although a failover cannot be guaranteed, there are few scenarios where a virtual machine will not be able to restart due to the integration HA offers with DRS and the fact that most clusters have spare capacity available to account for virtual machine demand variance. Although this is a corner-case scenario, it needs to be considered in environments where absolute guarantees must be provided.

Specify Failover Hosts

With the "Specify Failover Hosts" Admission Control Policy, when one or multiple hosts fail, HA will attempt to restart all virtual machines on the designated failover hosts. The designated failover hosts are essentially "hot standby" hosts. In other words, DRS will not migrate virtual machines to these hosts when resources are scarce or the cluster is imbalanced.

Pros:

- What you see is what you get.
- No fragmented resources.

Cons:

- What you see is what you get.
- Dedicated failover hosts not utilized during normal operations.

Recommendations

We have been asked many times for our recommendation on Admission Control and it is difficult to answer as each policy has its pros and cons. However, we generally recommend a Percentage based Admission Control Policy. It is the most flexible policy as it uses the actual reservation per virtual machine instead of taking a "worst case" scenario approach like the number of host failures does. However, the number of host failures policy guarantees the failover level under all circumstances. Percentage based is less restrictive, but offers lower guarantees that in all scenarios HA will be able to restart all virtual machines. With the added level of integration between HA and DRS we believe a Percentage based Admission Control Policy will fit most environments.

> **Basic design principle**: Do the math, and take customer requirements into account. We recommend using a “percentage” based admission control policy, as it is the most flexible.

Now that we have recommended which Admission Control Policy to use, the next step is to provide guidance around selecting the correct percentage. We cannot tell you what the ideal percentage is as that totally depends on the size of your cluster and, of course, on your resiliency model (N+1 vs. N+2). We can, however, provide guidelines around calculating how much of your resources should be set aside and how to prevent wasting resources.

Selecting the Right Percentage

It is a common strategy to select a single host as a percentage of resources reserved for failover. We geneårally recommend selecting a percentage which is the equivalent of a single or multiple hosts, Let’s explain why and what the impact is of not using the equivalent of a single or multiple hosts.

Let’s start with an example: a cluster exists of 8 ESXi hosts, each containing 70 GB of available RAM. This might sound like an awkward memory configuration but to simplify things we have already subtracted 2 GB as virtualization overhead. Although virtualization overhead is probably less than 2 GB, we have used this number to make calculations easier. This example zooms in on memory but this concept also applies to CPU, of course.

For this cluster we will define the percentage of resources to reserve for both Memory and CPU to 20%. For memory, this leads to a total cluster memory capacity of 448 GB:

```
(70 GB + 70 GB + 70 GB + 70 GB + 70 GB + 70 GB + 70 GB + 70 GB) * (1 – 20%)
```

A total of 112 GB of memory is reserved as failover capacity.

Once a percentage is specified, that percentage of resources will be unavailable for virtual machines, therefore it makes sense to set the percentage as close to the value that equals the resources a single (or multiple) host represents. We will demonstrate why this is important in subsequent examples.

In the example above, 20% was used to be reserved for resources in an 8-host cluster. This configuration reserves more resources than a single host contributes to the cluster. HA’s main objective is to provide automatic recovery for virtual machines after a physical server failure. For this reason, it is recommended to reserve resources equal to a single or multiple hosts. When using the per-host level granularity in an 8-host cluster (homogeneous configured hosts), the resource contribution per host to the cluster is 12.5%. However, the

percentage used must be an integer (whole number). It is recommended to round up to the value guaranteeing that the full capacity of one host is protected, in this example (Figure 40), the conservative approach would lead to a percentage of 13%.

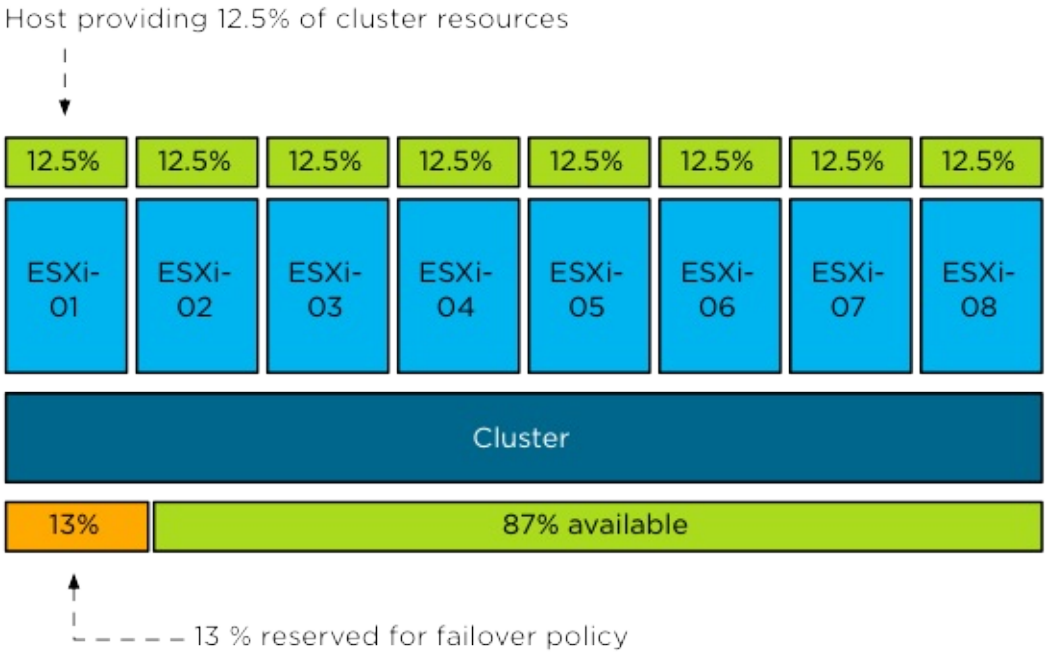


Figure 50 - Setting the correct value

Aggressive Approach

We have seen many environments where the percentage was set to a value that was less than the contribution of a single host to the cluster. Although this approach reduces the amount of resources reserved for accommodating host failures and results in higher consolidation ratios, it also offers a lower guarantee that HA will be able to restart all virtual machines after a failure. One might argue that this approach will more than likely work as most environments will not be fully utilized; however it also does eliminate the guarantee that after a failure all virtual machines will be recovered. Wasn't that the reason for enabling HA in the first place?

Adding Hosts to Your Cluster

Although the percentage is dynamic and calculates capacity at a cluster-level, changes to your selected percentage might be required when expanding the cluster. The reason being that the amount of reserved resources for a fail-over might not correspond with the contribution per host and as a result lead to resource wastage. For example, adding 4 hosts to an 8-host cluster and continuing to use the previously configured admission control policy value of 13% will result in a failover capacity that is equivalent to 1.5 hosts. Figure 41 depicts

a scenario where an 8-host cluster is expanded to 12 hosts. Each host holds 8 2 GHz cores and 70 GB of memory. The cluster was originally configured with admission control set to 13%, which equals to 109.2 GB and 24.96 GHz. If the requirement is to allow a single host failure 7.68 Ghz and 33.6 GB is “wasted” as clearly demonstrated in the diagram below.

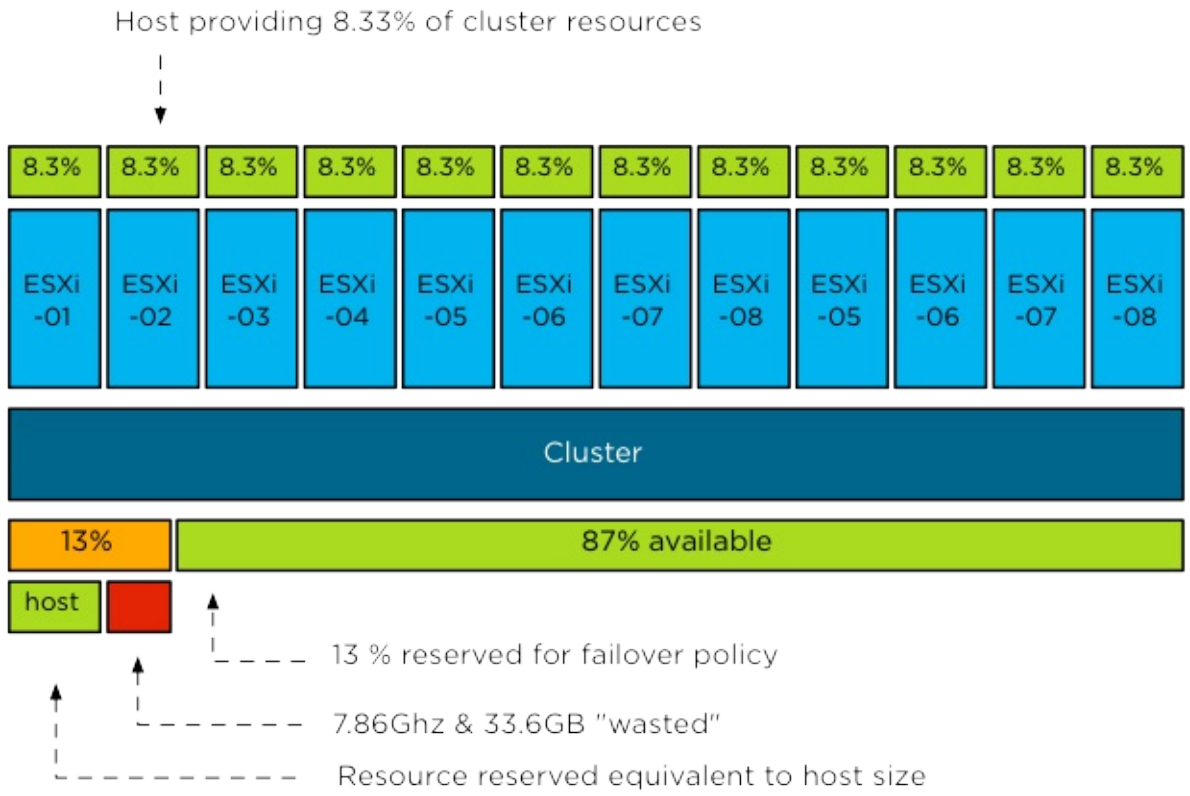


Figure 51 - Avoid wasting resources

How to Define Your Percentage?

As explained earlier it will fully depend on the N+X model that has been chosen. Based on this model, we recommend selecting a percentage that equals the amount of resources a single host represents. So, in the case of an 8 host cluster and N+2 resiliency, the percentage should be set as follows: `2 / 8 (*100) = 25%`

> **Basic design principle**: In order to avoid wasting resources we recommend carefully selecting your N+X resiliency architecture. Calculate the required percentage based on this architecture.

VM and Application Monitoring

VM and Application Monitoring is an often overlooked but really powerful feature of HA. The reason for this is most likely that it is disabled by default and relatively new compared to HA. We have tried to gather all the information we could around VM and Application Monitoring, but it is a pretty straightforward product that actually does what you expect it would do.

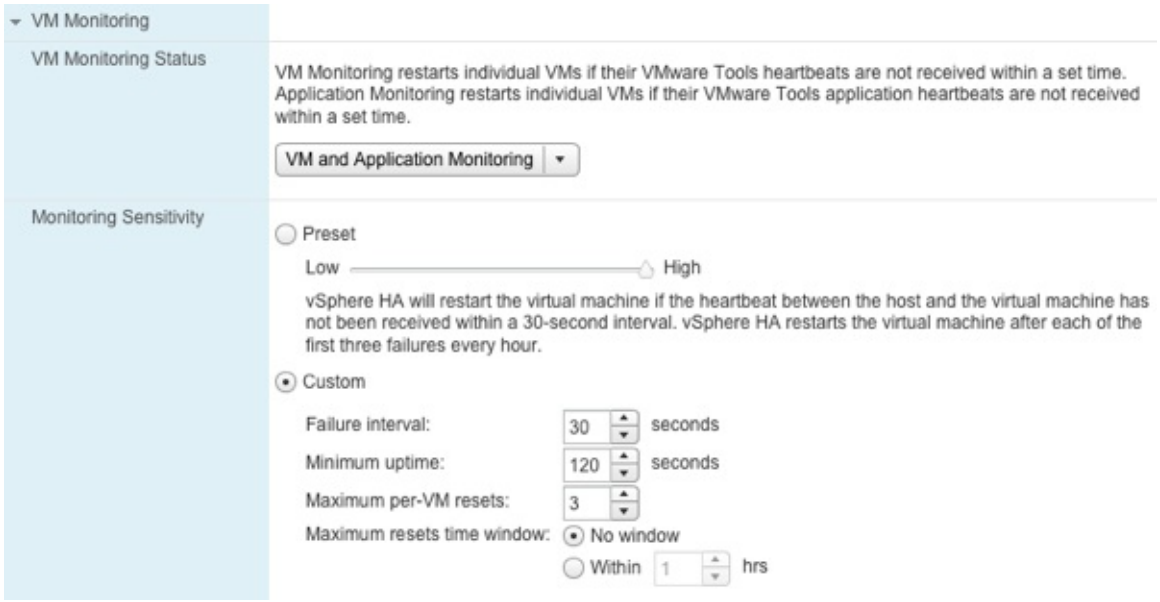


Figure 52 - VM and Application Monitoring

Why Do You Need VM/Application Monitoring?

VM and Application Monitoring acts on a different level from HA. VM/App Monitoring responds to a single virtual machine or application failure as opposed to HA which responds to a host failure. An example of a single virtual machine failure would, for instance, be the infamous “blue screen of death”. In the case of App Monitoring the type of failure that triggers a response is defined by the application developer or administrator.

How Does VM/App Monitoring Work?

VM Monitoring resets individual virtual machines when needed. VM/App monitoring uses a heartbeat similar to HA. If heartbeats, and, in this case, VMware Tools heartbeats, are not received for a specific (and configurable) amount of time, the virtual machine will be

restarted. These heartbeats are monitored by the HA agent and are not sent over a network, but stay local to the host.

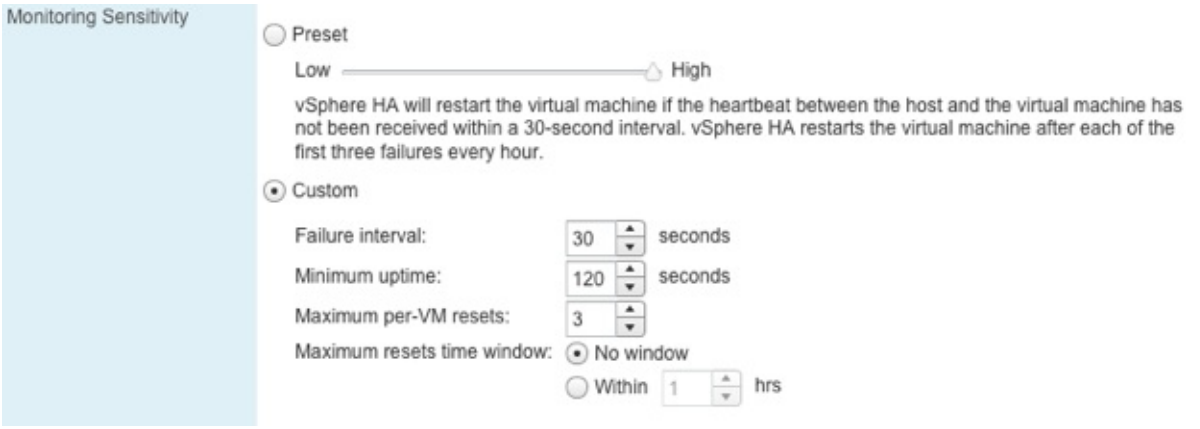


Figure 53 - VM Monitoring sensitivity

When enabling VM/App Monitoring, the level of sensitivity (Figure 43) can be configured. The default setting should fit most situations. Low sensitivity basically means that the number of allowed “missed” heartbeats is higher and the chances of running into a false positive are lower. However, if a failure occurs and the sensitivity level is set to Low, the experienced downtime will be higher. When quick action is required in the event of a failure, “high sensitivity” can be selected. As expected, this is the opposite of “low sensitivity”. Note that the advanced settings mentioned in the following table are deprecated and listed for educational purposes.

Sensitivity	Failure interval	Max failures	Maxim resets time window
Low	120 Seconds	3	7 Days
Medium	60 Seconds	3	24 Hours
High	30 Seconds	3	1 hour

It is important to remember that VM Monitoring does not infinitely reboot virtual machines unless you specify a custom policy with this requirement. This is to avoid a problem from repeating. By default, when a virtual machine has been rebooted three times within an hour, no further attempts will be taken. Unless the specified time has elapsed. The following advanced settings can be set to change this default behavior or “custom” can be selected as shown in Figure 43.

Although the heartbeat produced by VMware Tools is reliable, VMware added a further verification mechanism. To avoid false positives, VM Monitoring also monitors I/O activity of the virtual machine. When heartbeats are not received AND no disk or network activity has

occurred over the last 120 seconds, per default, the virtual machine will be reset. Changing the advanced setting "*das.iostatsInterval*" can modify this 120-second interval.

It is recommended to align the *das.iostatsInterval* with the *failure interval* selected in the VM Monitoring section of vSphere HA within the Web Client or the vSphere Client.

> **Basic design principle**: Align *das.iostatsInterval* with the failure interval.

Screenshots

One of the most useful features as part of VM Monitoring is the fact that it takes screenshots of the virtual machine's console. The screenshots are taken right before VM Monitoring resets a virtual machine. It is a very useful feature when a virtual machine "freezes" every once in a while for no apparent reason. This screenshot can be used to debug the virtual machine operating system when needed, and is stored in the virtual machine's working directory as logged in the Events view on the Monitor tab of the virtual machine.

> **Basic design principle:** VM and Application monitoring can substantially increase availability. It is part of the HA stack and we strongly recommend using it!

VM Monitoring Implementation Details

VM/App Monitoring is implemented as part of the HA agent itself. The agent uses the "Performance Manager" to monitor disk and network I/O; VM/App Monitoring uses the "usage" counters for both disk and network and it requests these counters once enough heartbeats have been missed that the configured policy is triggered.

As stated before, VM/App Monitoring uses heartbeats just like host-level HA. The heartbeats are monitored by the HA agent, which is responsible for the restarts. Of course, this information is also being rolled up into vCenter, but that is done via the Management Network, not using the virtual machine network. This is crucial to know as this means that when a virtual machine network error occurs, the virtual machine heartbeat will still be received. When an error occurs, HA will trigger a restart of the virtual machine when all three conditions are met:

1. No VMware Tools heartbeat received
2. No network I/O over the last 120 seconds
3. No storage I/O over the last 120 seconds

Just like with host-level HA, the HA agent works independently of vCenter when it comes to virtual machine restarts.

Timing

The VM/App monitoring feature monitors the heartbeat(s) issued by a guest and resets the virtual machine if there is a heartbeat failure that satisfies the configured policy for the virtual machine. HA can monitor just the heartbeats issued by the VMware tools process or can monitor these heartbeats plus those issued by an optional in-guest agent.

If the VM monitoring heartbeats stop at time T-0, the minimum time before HA will declare a heartbeat failure is in the range of 81 seconds to 119 seconds, whereas for heartbeats issued by an in-guest application agent, HA will declare a failure in the range of 61 seconds to 89 seconds. Once a heartbeat failure is declared for application heartbeats, HA will attempt to reset the virtual machine. However, for VMware tools heartbeats, HA will first check whether any IO has been issued by the virtual machine for the last 2 minutes (by default) and only if there has been no IO will it issue a reset. Due to how HOSTD publishes the I/O statistics, this check could delay the reset by approximately 20 seconds for virtual machines that were issuing I/O within approximately 1 minute of T-0.

Timing details: the range depends on when the heartbeats stop relative to the HOSTD thread that monitors them. For the lower bound of the VMware tools heartbeats, the heartbeats stop a second before the HOSTD thread runs, which means, at T+31, the FDM agent on the host will be notified of a tools yellow state, and then at T+61 of the red state, which HA reacts to. HA then monitors the heartbeat failure for a minimum of 30 seconds, leading to the min of T+91. The 30 seconds monitoring period done by HA can be increased using the das.failureInterval policy setting. For the upper bound, the FDM is not notified until T+59s (T=0 the failure occurs, T+29 HOSTD notices it and starts the heartbeat failure timer, and at T+59 HOSTD reports a yellow state, and at T+89 reports a red state).

For the heartbeats issued by an in-guest agent, no yellow state is sent, so the there is no additional 30 seconds period.

Application Monitoring

Application Monitoring is a part of VM Monitoring. Application Monitoring is a feature that partners and / or customers can leverage to increase resiliency, as shown in the screenshot below but from an application point of view rather than from a VM point of view. There is an SDK available to the general public and it is part of the guest SDK.

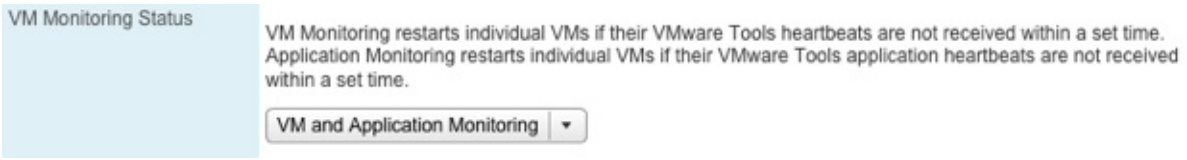


Figure 54 -

The Guest SDK is currently primarily used by application developers from partners like Symantec to develop solutions that increase resilience on a different level than VM Monitoring and HA. In the case of Symantec, a simplified version of Veritas Cluster Server (VCS) is used to enable application availability monitoring, including responding to issues. Note that this is not a multi-node clustering solution like VCS itself, but a single node solution.

Symantec ApplicationHA, as it is called, is triggered to get the application up and running again by restarting it. Symantec's ApplicationHA is aware of dependencies and knows in which order services should be started or stopped. If, however, this fails for a certain number (configurable option within ApplicationHA) of times, VMware HA will be requested to take action. This action will be a restart of the virtual machine.

Although Application Monitoring is relatively new and there are only a few partners currently exploring the capabilities, in our opinion, it does add a whole new level of resiliency. Your in-house development team could leverage functionality offered through the API, or you could use a solution developed by one of VMware’s partners. We have tested ApplicationHA by Symantec and personally feel it is the missing link. It enables you as System Admin to integrate your virtualization layer with your application layer. It ensures you as a System Admin that services which are protected are restarted in the correct order and it avoids the common pitfalls associated with restarts and maintenance. Note that VMware also introduced an "Application Monitoring" solution which was based on Hyperic technology, this product however has been deprecated and as such will not be discussed in this publication.

Application Awareness API

The Application Awareness API is open for everyone. We feel that this is not the place to do a full deepdive on how to use it, but we do want to discuss it briefly.

The Application Awareness API allows for anyone to talk to it, including scripts, which makes the possibilities endless. Currently there are 6 functions defined:

- `_VMGuestAppMonitor_Enable_()`
  - Enables Monitoring
- `_VMGuestAppMonitor_MarkActive_()`

    - Call every 30 seconds to mark application as active
- `_VMGuestAppMonitor_Disable_()`
    - Disable Monitoring
- `_VMGuestAppMonitor_IsEnabled_()`
    - Returns status of Monitoring
- `_VMGuestAppMonitor_GetAppStatus_()`
    - Returns the current application status recorded for the application
- `_VMGuestAppMonitor_Free(_)`
    - Frees the result of the VMGuestAppMonitor_GetAppStatus() call

These functions can be used by your development team, however App Monitoring also offers a new executable. This allows you to use the functionality App Monitoring offers without the need to compile a full binary. This new command, vmware-appmonitoring.exe, takes the following arguments, which are not coincidentally similar to the functions:

- Enable
- Disable
- markActive
- isEnabled
- getAppStatus

When running the command vmware-appmonitor.exe, which can be found under "VMware-GuestAppMonitorSDK\bin\win32\" the following output is presented:

```
Usage: vmware-appmonitor.exe {enable | disable | markActive | isEnabled | getApp Status}
```

As shown there are multiple ways of leveraging Application Monitoring and to enhance resiliency on an application level.

vSphere HA and ...

Now that you know how HA works inside out, we want to explain the different integration points between HA, DRS and SDRS.

HA and Storage DRS

vSphere HA informs Storage DRS when a failure has occurred. This to prevent the relocation of any HA protected virtual machine, meaning, a virtual machine that was powered on, but which failed, and has not been restarted yet due to their being insufficient capacity available. Further, Storage DRS is not allowed to Storage vMotion a virtual machine that is owned by a master other than the one vCenter Server is talking to. This is because in such a situation, HA would not be able to reprotect the virtual machine until the master to which vCenter Server is talking is able to lock the datastore again.

Storage vMotion and HA

If a virtual machine needs to be restarted by HA and the virtual machine is in the process of being Storage vMotioned and the virtual machine fails, the restart process is not started until vCenter informs the master that the Storage vMotion task has completed or has been rolled back. If the source host fails, however, virtual machine will restart the virtual machine as part of the normal workflow. During a Storage vMotion, the HA agent on the host on which the Storage vMotion was initiated masks the failure state of the virtual machine. If, for whatever reason, vCenter is unavailable, the masking will timeout after 15 minutes to ensure that the virtual machine will be restarted.

Also note that when a Storage vMotion completes, vCenter will report the virtual machine as unprotected until the master reports it protected again under the new path.

HA and DRS

HA integrates on multiple levels with DRS. It is a huge improvement and it is something that we wanted to stress as it has changed both the behavior and the reliability of HA.

HA and Resource Fragmentation

When a failover is initiated, HA will first check whether there are resources available on the destination hosts for the failover. If, for instance, a particular virtual machine has a very large reservation and the Admission Control Policy is based on a percentage, for example, it could happen that resources are fragmented across multiple hosts. (For more details on this scenario, see Chapter 7.) HA will ask DRS to defragment the resources to accommodate for this virtual machine's resource requirements. Although HA will request a defragmentation of resources, a guarantee cannot be given. As such, even with this additional integration, you should still be cautious when it comes to resource fragmentation.

Flattened Shares

When shares have been set custom on a virtual machine an issue can arise when that VM needs to be restarted. When HA fails over a virtual machine, it will power-on the virtual machine in the Root Resource Pool. However, the virtual machine's shares were those configured by a user for it, and not scaled for it being parented under the Root Resource Pool. This could cause the virtual machine to receive either too many or too few resources relative to its entitlement.

A scenario where and when this can occur would be the following:

VM1 has a 1000 shares and Resource Pool A has 2000 shares. However Resource Pool A has 2 virtual machines and both virtual machines will have 50% of those "2000" shares. The following diagram depicts this scenario:

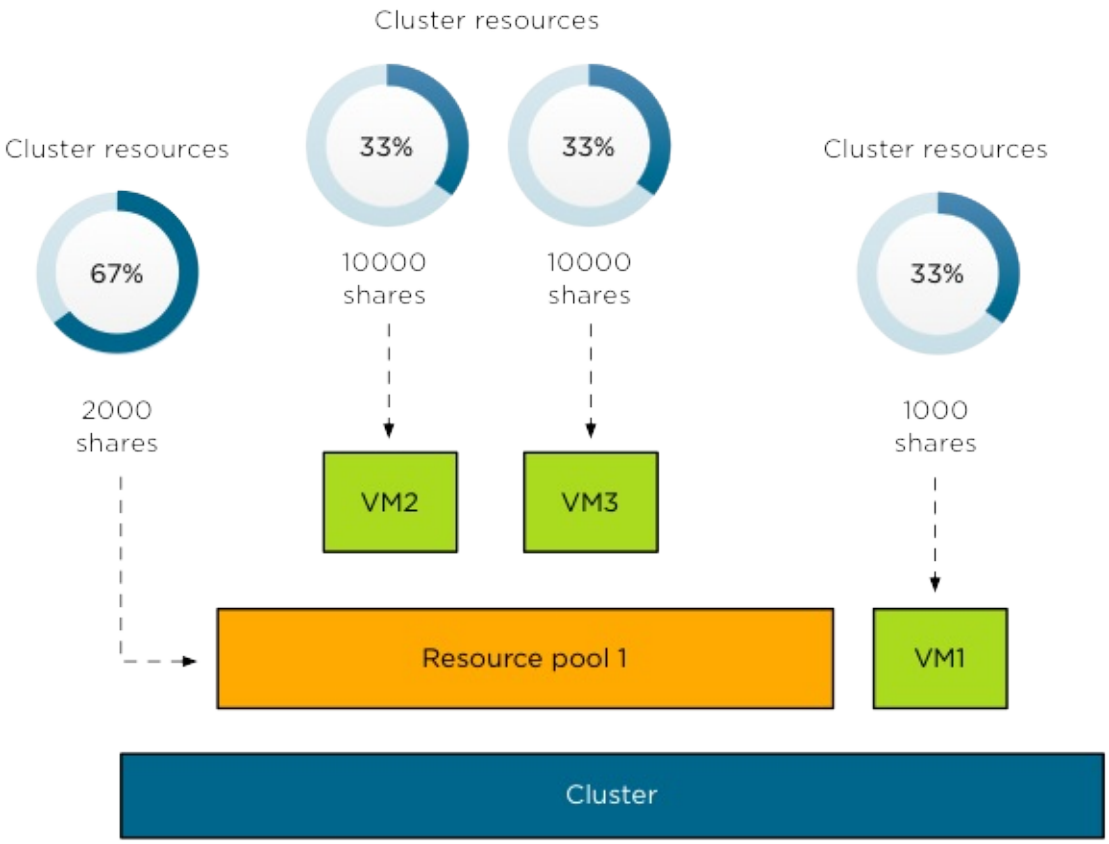


Figure 55 - Flatten shares starting point

When the host fails, both VM2 and VM3 will end up on the same level as VM1, the Root Resource Pool. However, as a custom shares value of 10,000 was specified on both VM2 and VM3, they will completely blow away VM1 in times of contention. This is depicted in the following diagram:

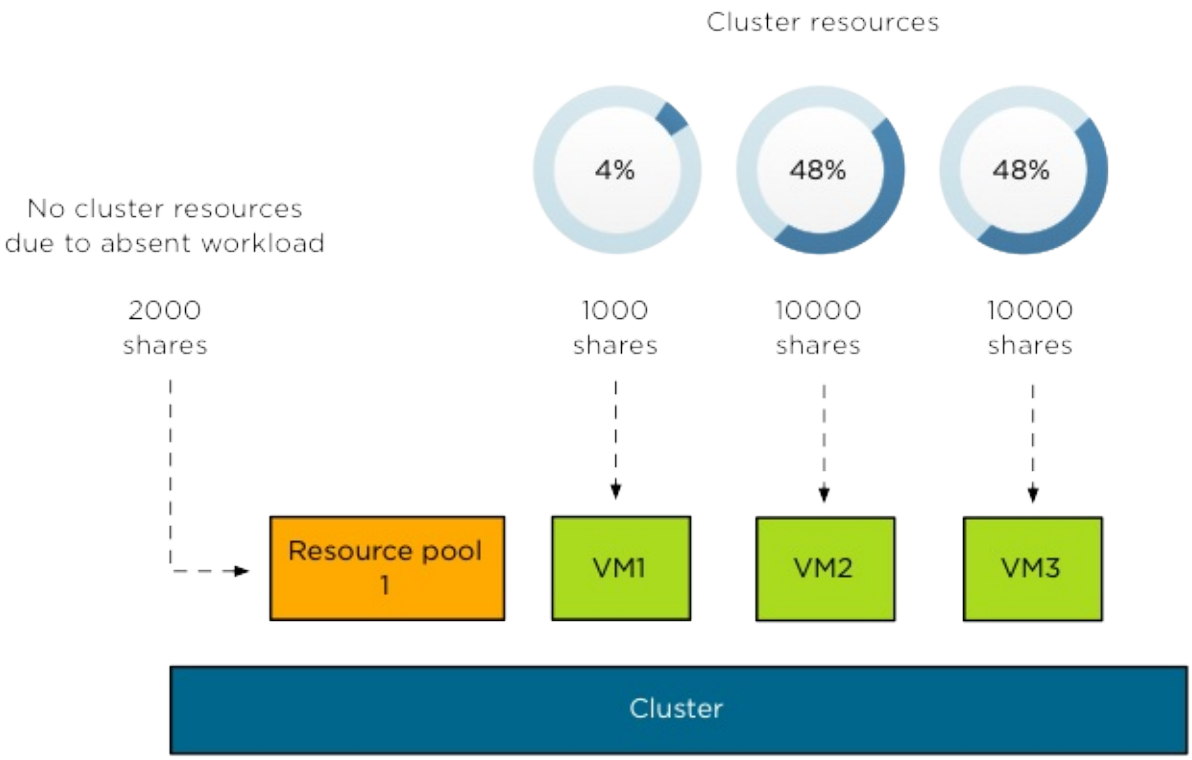


Figure 56 - Flatten shares host failure

This situation would persist until the next invocation of DRS would re-parent the virtual machines VM2 and VM3 to their original Resource Pool. To address this issue HA calculates a flattened share value before the virtual machine's is failed-over. This flattening process ensures that the virtual machine will get the resources it would have received if it had failed over to the correct Resource Pool. This scenario is depicted in the following diagram. Note that both VM2 and VM3 are placed under the Root Resource Pool with a shares value of 1000.

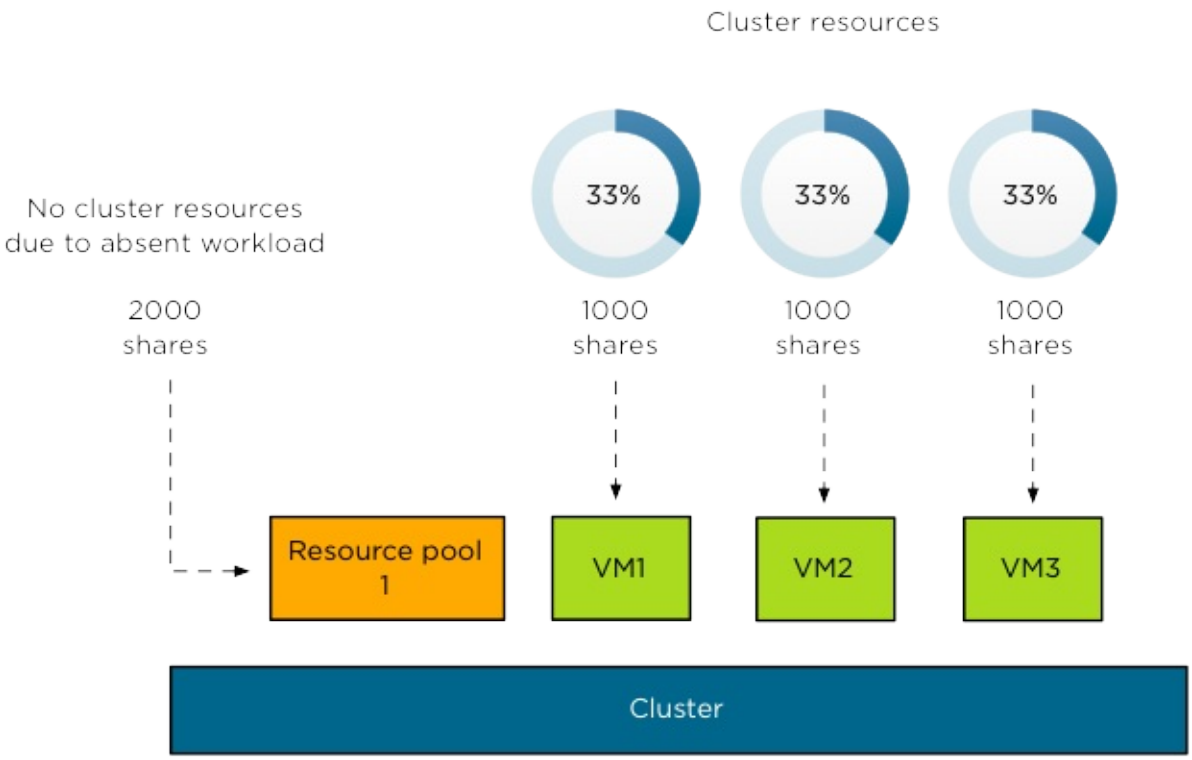


Figure 57 - Flatten shares after host failure before DRS invocation

Of course, when DRS is invoked, both VM2 and VM3 will be re-parented under Resource Pool 1 and will again receive the number of shares they had been originally assigned.

DPM and HA

If DPM is enabled and resources are scarce during an HA failover, HA will use DRS to try to adjust the cluster (for example, by bringing hosts out of standby mode or migrating virtual machines to defragment the cluster resources) so that HA can perform the failovers.

If HA strict Admission Control is enabled (default), DPM will maintain the necessary level of powered-on capacity to meet the configured HA failover capacity. HA places a constraint to prevent DPM from powering down too many ESXi hosts if it would violate the Admission Control Policy.

When HA admission control is disabled, HA will prevent DPM from powering off all but one host in the cluster. A minimum of two hosts are kept up regardless of the resource consumption. The reason this behavior has changed is that it is impossible to restart virtual machines when the only host left in the cluster has just failed.

In a failure scenario, if HA cannot restart some virtual machines, it asks DRS/DPM to try to defragment resources or bring hosts out of standby to allow HA another opportunity to restart the virtual machines. Another change is that DRS/DPM will power-on or keep on

hosts needed to address cluster constraints, even if those host are lightly utilized. Once again, in order for this to be successful DRS will need to be enabled and configured to fully automated. When not configured to fully automated user action is required to execute DRS recommendations and allow the restart of virtual machines to occur.

Stretched Clusters

In this part we will be discussing a specific infrastructure architecture and how HA, DRS and Storage DRS can be leveraged and should be deployed to increase availability. Be it availability of your workload or the resources provided to your workload, we will guide you through some of the design considerations and decision points along the way. Of course, a full understanding of your environment will be required in order to make appropriate decisions regarding specific implementation details. Nevertheless, we hope that this section will provide a proper understanding of how certain features play together and how these can be used to meet the requirements of your environment and build the desired architecture.

Scenario

The scenario we have chosen is a stretched cluster also referred to as a VMware vSphere Metro Storage Cluster solution. We have chosen this specific scenario as it allows us to explain a multitude of design and architectural considerations. Although this scenario has been tested and validated in our lab, every environment is unique and our recommendations are based on our experience and your mileage may vary.

A VMware vSphere Metro Storage Cluster (vMSC) configuration is a VMware vSphere certified solution that combines synchronous replication with storage array based clustering. These solutions are typically deployed in environments where the distance between datacenters is limited, often metropolitan or campus environments.

The primary benefit of a stretched cluster model is to enable fully active and workload-balanced datacenters to be used to their full potential. Many customers find this architecture attractive due to the capability of migrating virtual machines with vMotion and Storage vMotion between sites. This enables on-demand and non-intrusive cross-site mobility of workloads. The capability of a stretched cluster to provide this active balancing of resources should always be the primary design and implementation goal.

Stretched cluster solutions offer the benefit of:

- Workload mobility
- Cross-site automated load balancing
- Enhanced downtime avoidance
- Disaster avoidance

Technical requirements and constraints

Due to the technical constraints of an online migration of VMs, the following specific requirements, which are listed in the VMware Compatibility Guide, must be met prior to consideration of a stretched cluster implementation:

- Storage connectivity using Fibre Channel, iSCSI, NFS, and FCoE is supported.
- The maximum supported network latency between sites for the VMware ESXi management networks is 10ms round-trip time (RTT).
- vMotion, and Storage vMotion, supports a maximum of 150ms latency as of vSphere 6.0, but this is not intended for stretched clustering usage.
- The maximum supported latency for synchronous storage replication links is 10ms RTT. Refer to documentation from the storage vendor because the maximum tolerated latency is lower in most cases. The most commonly supported maximum RTT is 5ms.
- The ESXi vSphere vMotion network has a redundant network link minimum of 250Mbps.

The storage requirements are slightly more complex. A vSphere Metro Storage Cluster requires what is in effect a **single storage subsystem** that spans both sites. In this design, a given datastore must be accessible—that is, be able to be read and be written to—simultaneously from both sites. Further, when problems occur, the ESXi hosts must be able to continue to access datastores from either array transparently and with no impact to ongoing storage operations.

This precludes traditional synchronous replication solutions because they create a primary–secondary relationship between the active (primary) LUN where data is being accessed and the secondary LUN that is receiving replication. To access the secondary LUN, replication is stopped, or reversed, and the LUN is made visible to hosts. This “promoted” secondary LUN has a completely different LUN ID and is essentially a newly available copy of a former primary LUN. This type of solution works for traditional disaster recovery–type configurations because it is expected that VMs must be started up on the secondary site. The vMSC configuration requires simultaneous, uninterrupted access to enable live migration of running VMs between sites.

The storage subsystem for a vMSC must be able to be read from and write to both locations simultaneously. All disk writes are committed synchronously at both locations to ensure that data is always consistent regardless of the location from which it is being read. This storage architecture requires significant bandwidth and very low latency between the sites in the cluster. Increased distances or latencies cause delays in writing to disk and a dramatic decline in performance. They also preclude successful vMotion migration between cluster nodes that reside in different locations.

Uniform versus Non-Uniform

vMSC solutions are classified into two distinct categories. These categories are based on a fundamental difference in how hosts access storage. It is important to understand the different types of stretched storage solutions because this influences design considerations. The following two main categories are as described on the VMware Hardware Compatibility List:

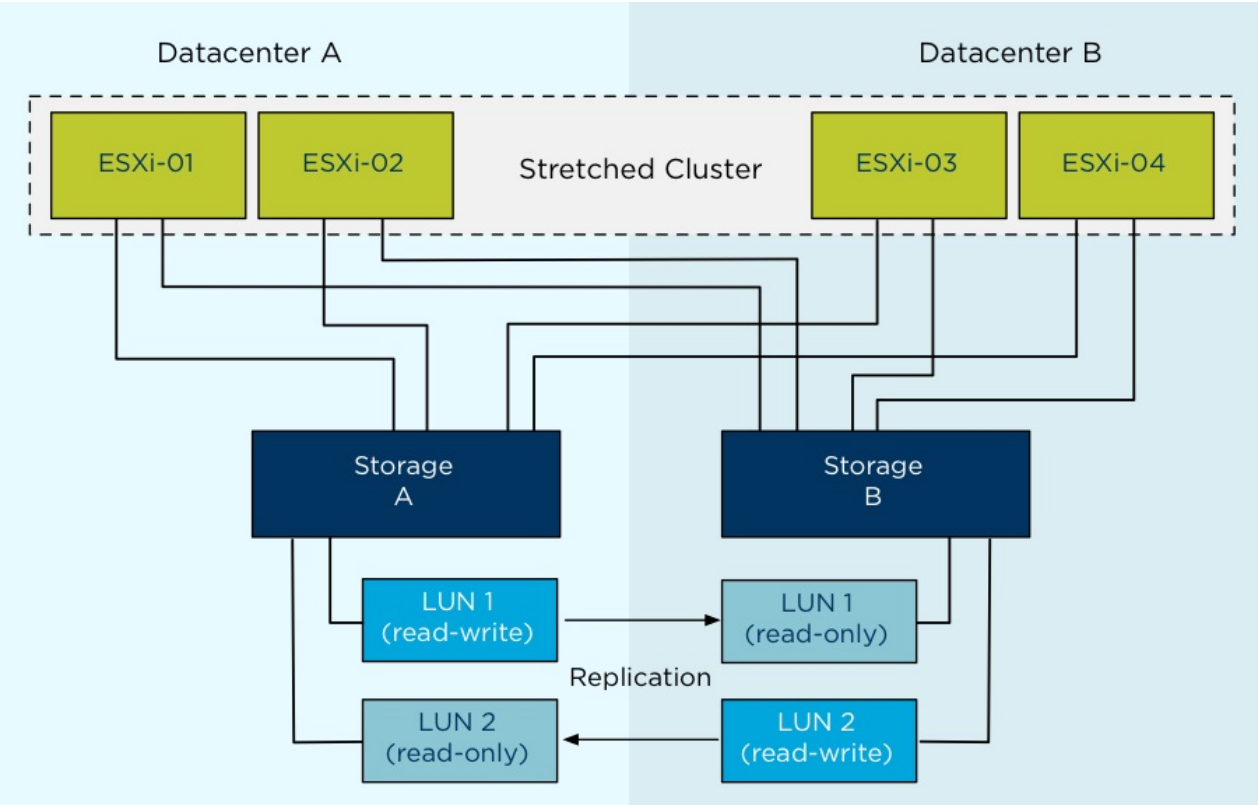
- Uniform host access configuration – ESXi hosts from both sites are all connected to a storage node in the storage cluster across all sites. Paths presented to ESXi hosts are stretched across a distance.
- Nonuniform host access configuration – ESXi hosts at each site are connected only to storage node(s) at the same site. Paths presented to ESXi hosts from storage nodes are limited to the local site.

The following in-depth descriptions of both categories clearly define them from architectural and implementation perspectives.

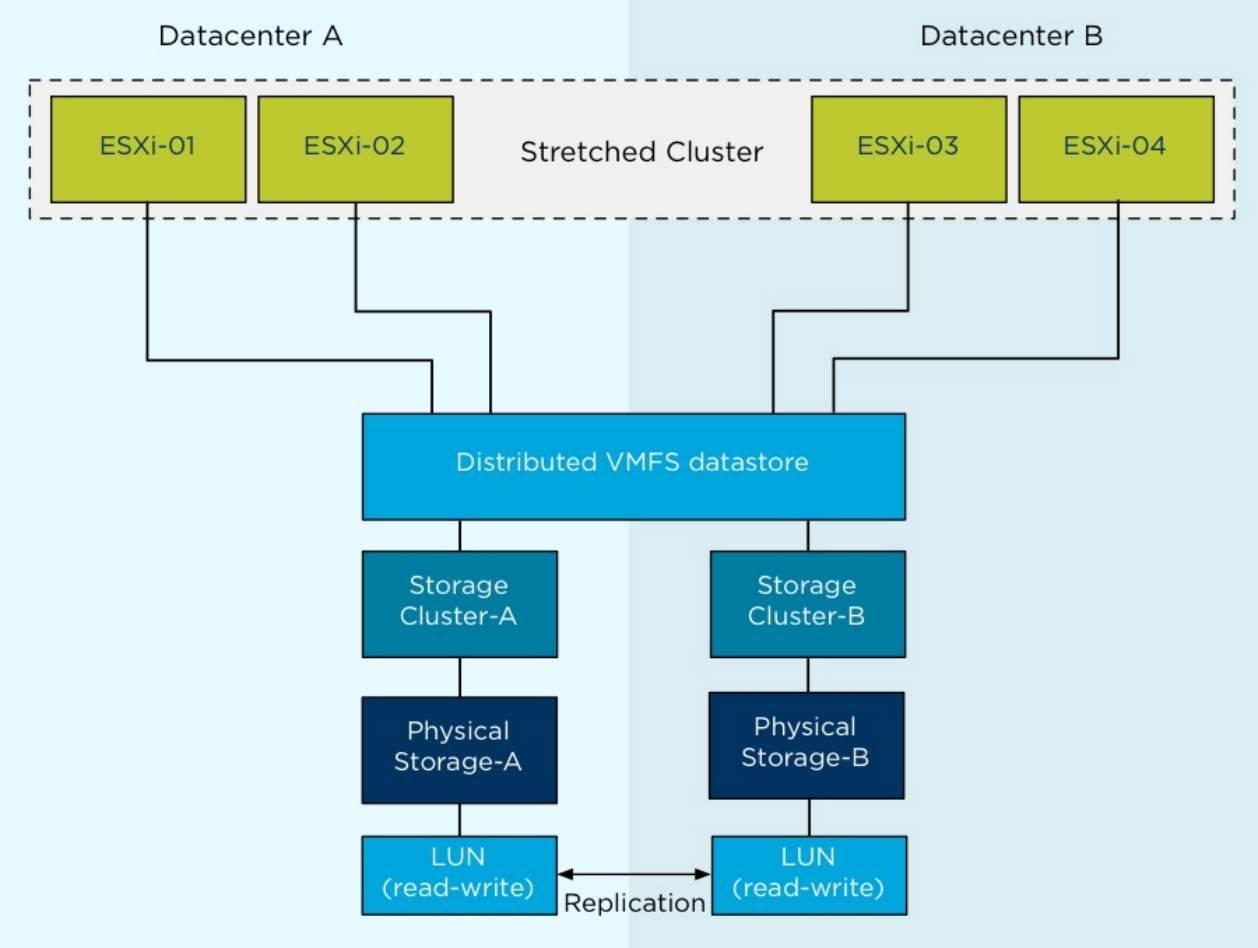
With **uniform** host access configuration, hosts in data center A and data center B have access to the storage systems in both data centers. In effect, the storage area network is stretched between the sites, and all hosts can access all LUNs. NetApp MetroCluster software is an example of uniform storage. In this configuration, read\/write access to a LUN takes place on one of the two arrays, and a synchronous mirror is maintained in a hidden, read-only state on the second array. For example, if a LUN containing a datastore is read\/write on the array in data center A, all ESXi hosts access that datastore via the array in data center A. For ESXi hosts in data center A, this is local access. ESXi hosts in data center B that are running VMs hosted on this datastore send read\/write traffic across the network between data centers. In case of an outage or an operator-controlled shift of control of the LUN to data center B, all ESXi hosts continue to detect the identical LUN being presented, but it is now being accessed via the array in data center B.

The ideal situation is one in which VMs access a datastore that is controlled (read\/write) by the array in the same data center. This minimizes traffic between data centers to avoid the performance impact of reads' traversing the interconnect.

The notion of "site affinity" for a VM is dictated by the read\/write copy of the datastore. "Site affinity" is also sometimes referred to as "site bias" or "LUN locality." This means that when a VM has site affinity with data center A, its read\/write copy of the datastore is located in data center A. This is explained in more detail in the "vSphere DRS" subsection of this section.



With **nonuniform** host access configuration, hosts in data center A have access only to the array within the local data center; the array, as well as its peer array in the opposite data center, is responsible for providing access to datastores in one data center to ESXi hosts in the opposite data center. EMC VPLEX is an example of a storage system that can be deployed as a nonuniform storage cluster, although it can also be configured in a uniform manner. VPLEX provides the concept of a “virtual LUN,” which enables ESXi hosts in each data center to read and write to the same datastore or LUN. VPLEX technology maintains the cache state on each array so ESXi hosts in either data center detect the LUN as local. EMC calls this solution “write anywhere.” Even when two VMs reside on the same datastore but are located in different data centers, they write locally without any performance impact on either VM. A key point with this configuration is that each LUN or datastore has “site affinity,” also sometimes referred to as “site bias” or “LUN locality.” In other words, if anything happens to the link between the sites, the storage system on the preferred site for a given datastore will be the only one remaining with read\/write access to it. This prevents any data corruption in case of a failure scenario.



Our examples use uniform storage because these configurations are currently the most commonly deployed. Many of the design considerations, however, also apply to nonuniform configurations. We point out exceptions when this is not the case.

Scenario Architecture

In this section we will describe the architecture deployed for this scenario. We will also discuss some of the basic configuration and behavior of the various vSphere features. For an in-depth explanation of each respective feature, refer to the HA and the DRS section of this book. We will make specific recommendations based on VMware best practices and provide operational guidance where applicable. In our failure scenarios it will be explained how these practices prevent or limit downtime.

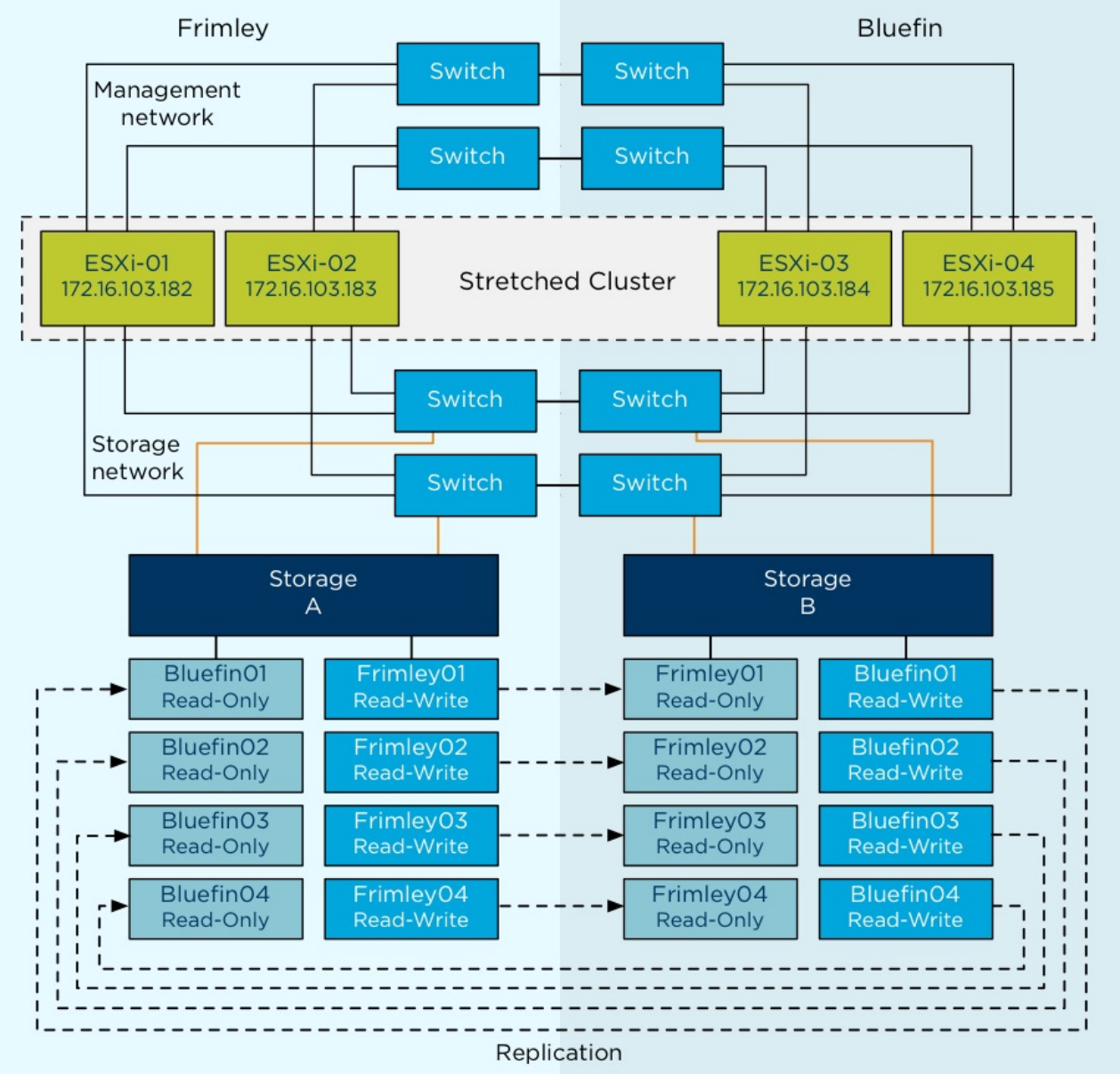
Infrastructure

The described infrastructure consists of a single vSphere 6.0 cluster with four ESXi 6.0 hosts. These hosts are managed by a single vCenter Server 6.0 instance. The first site is called Frimley; the second site is called Bluefin. The network between Frimley data center

and Bluefin data center is a stretched layer 2 network. There is a minimal distance between the sites, as is typical in campus cluster scenarios.

Each site has two ESXi hosts, and the vCenter Server instance is configured with vSphere DRS affinity to the hosts in Bluefin data center. In a stretched cluster environment, only a single vCenter Server instance is used. This is different from a traditional VMware Site Recovery Manager™ configuration in which a dual vCenter Server configuration is required. The configuration of VM-to-host affinity rules is discussed in more detail in the “vSphere DRS” subsection of this document.

Eight LUNs are depicted the diagram below. Four of these are accessed through the virtual IP address active on the iSCSI storage system in the Frimley data center; four are accessed through the virtual IP address active on the iSCSI storage system in the Bluefin data center.



Location	Hosts	Datastores	Local Isolation Address
Bluefin	172.16.103.184	Bluefin01	172.16.103.10
	172.16.103.185	Bluefin02	n\/a
		Bluefin03	n\/a
		Bluefin04	n\/a
Frimley	172.16.103.182	Frimley01	172.16.103.11
	172.16.103.183	Frimley02	n\/a
		Frimley03	n\/a
		Frimley04	n\/a

The vSphere cluster is connected to a stretched storage system in a fabric configuration with a uniform device access model. This means that every host in the cluster is connected to both storage heads. Each of the heads is connected to two switches, which are connected to two similar switches in the secondary location. For any given LUN, one of the two storage heads presents the LUN as read\/write via iSCSI. The other storage head maintains the replicated, read-only copy that is effectively hidden from the ESXi hosts.

vSphere Configuration

Our focus in this section is on vSphere HA, vSphere DRS, and vSphere Storage DRS in relation to stretched cluster environments. Design and operational considerations regarding vSphere are commonly overlooked and underestimated. Much emphasis has traditionally been placed on the storage layer, but little attention has been applied to how workloads are provisioned and managed.

One of the key drivers for using a stretched cluster is workload balance and disaster avoidance. How do we ensure that our environment is properly balanced without impacting availability or severely increasing the operational expenditure? How do we build the requirements into our provisioning process and validate periodically that we still meet them? Ignoring the requirements makes the environment confusing to administrate and less predictable during the various failure scenarios for which it should be of help.

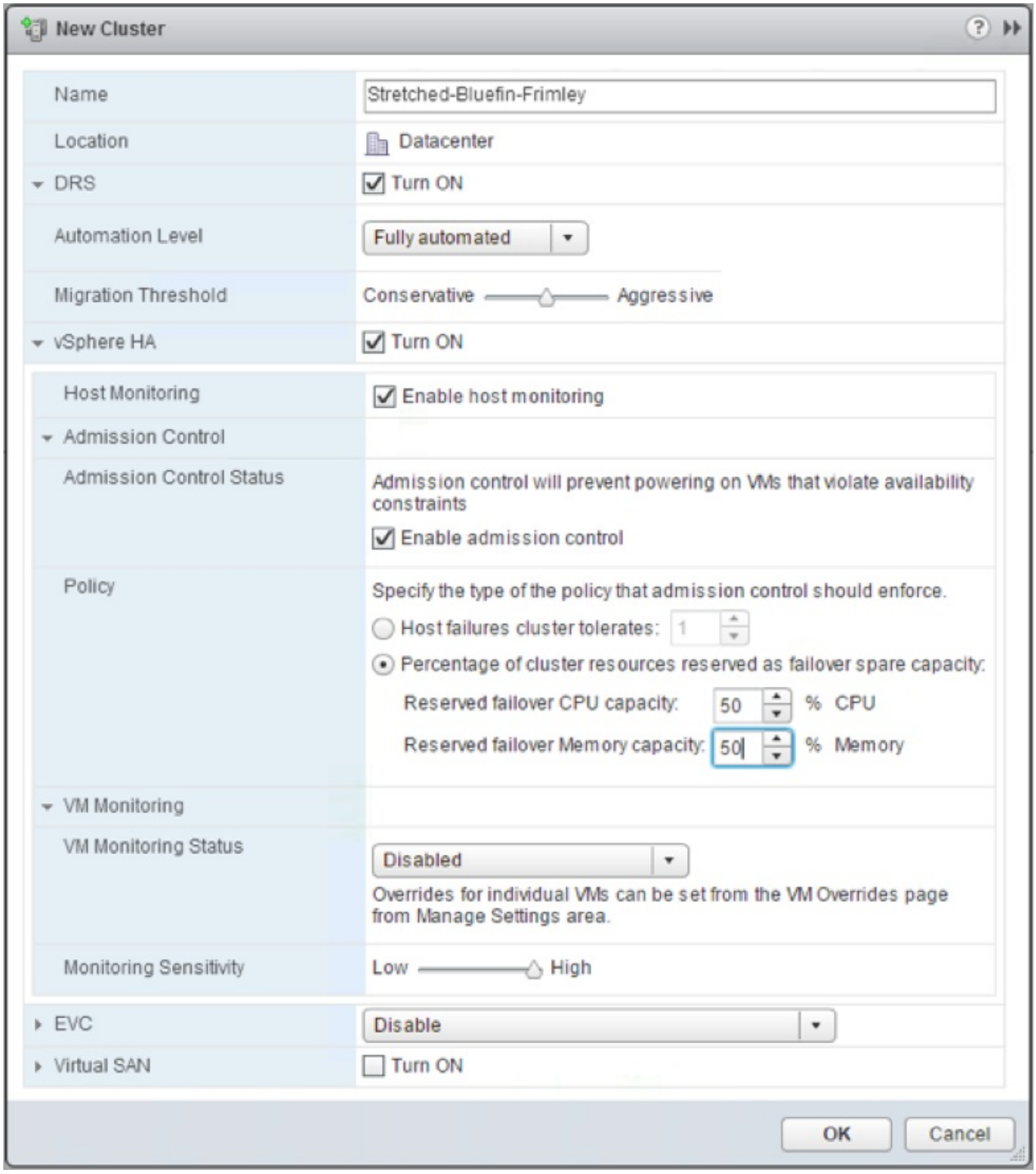
Each of these three vSphere features has very specific configuration requirements and can enhance environment resiliency and workload availability. Architectural recommendations based on our findings during the testing of the various failure scenarios are given throughout this section.

vSphere HA

The environment has four hosts and a uniform stretched storage solution. A full site failure is one scenario that must be taken into account in a resilient architecture. VMware recommends enabling vSphere HA admission control. Workload availability is the primary driver for most stretched cluster environments, so providing sufficient capacity for a full site failure is recommended. Such hosts are equally divided across both sites. To ensure that all workloads can be restarted by vSphere HA on just one site, configuring the admission control policy to 50 percent for both memory and CPU is recommended.

VMware recommends using a percentage-based policy because it offers the most flexibility and reduces operational overhead. Even when new hosts are introduced to the environment, there is no need to change the percentage and no risk of a skewed consolidation ratio due to possible use of VM-level reservations.

The screenshot below shows a vSphere HA cluster configured with admission control enabled and with the percentage-based policy set to 50 percent.



vSphere HA uses heartbeat mechanisms to validate the state of a host. There are two such mechanisms: network heartbeating and datastore heartbeating. Network heartbeating is the primary mechanism for vSphere HA to validate availability of the hosts. Datastore heartbeating is the secondary mechanism used by vSphere HA; it determines the exact state of the host after network heartbeating has failed.

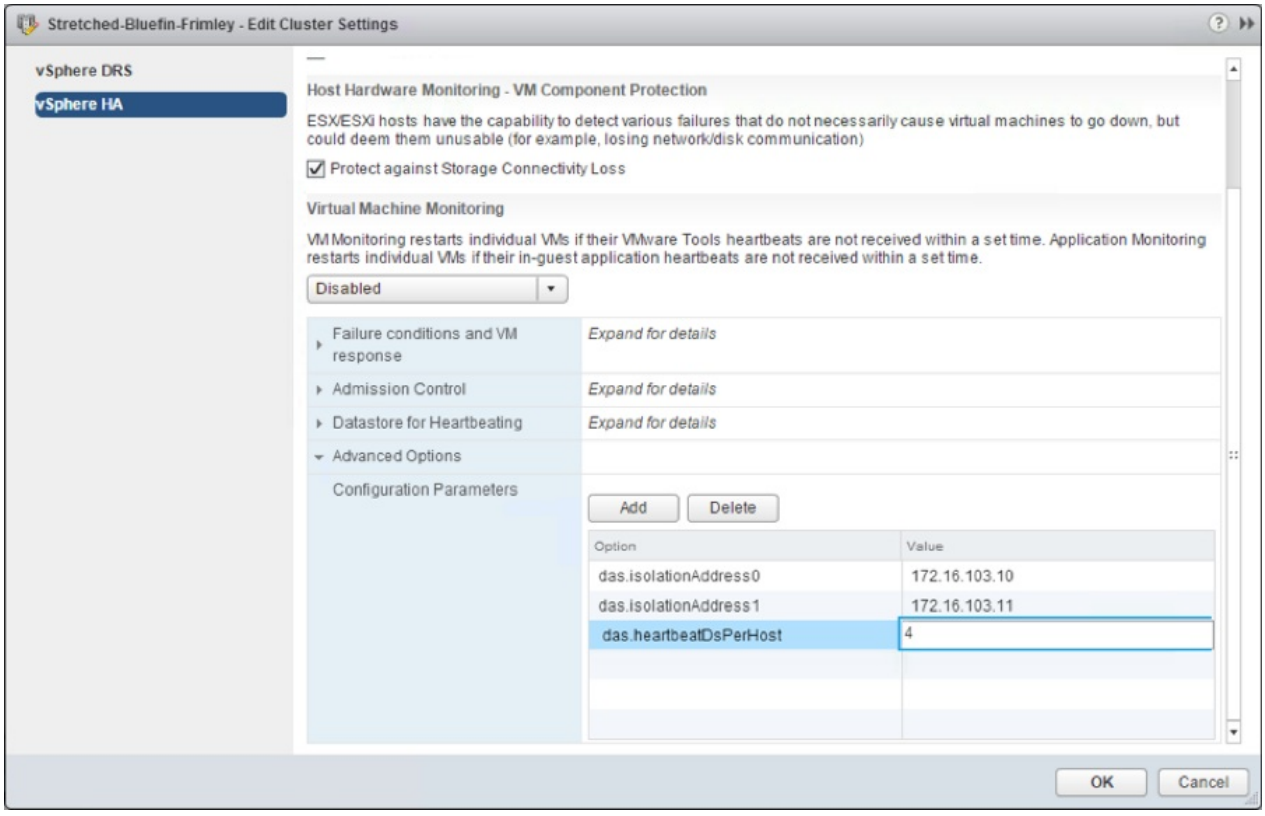
If a host is not receiving any heartbeats, it uses a fail-safe mechanism to detect if it is merely isolated from its master node or completely isolated from the network. It does this by pinging the default gateway. In addition to this mechanism, one or more isolation addresses can be

specified manually to enhance reliability of isolation validation. VMware recommends specifying a minimum of two additional isolation addresses, with each address site local.

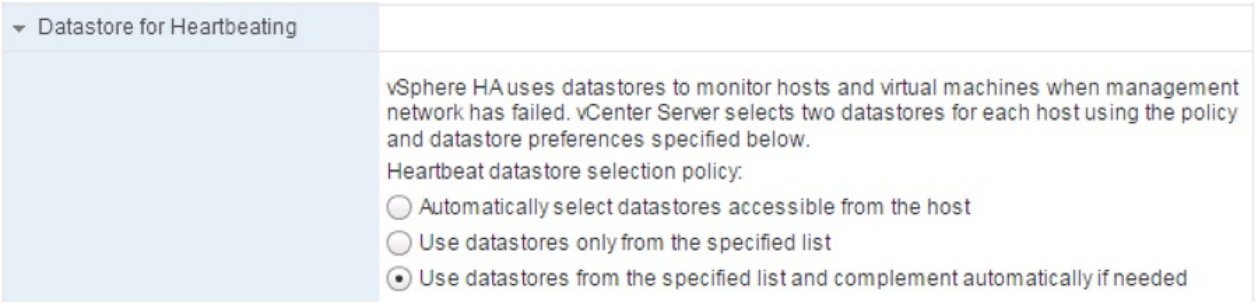
In our scenario, one of these addresses physically resides in the Frimley data center; the other physically resides in the Bluefin data center. This enables vSphere HA validation for complete network isolation, even in case of a connection failure between sites. The next screenshot shows an example of how to configure multiple isolation addresses. The vSphere HA advanced setting used is *das.isolationaddress*. More details on how to configure this can be found in VMware Knowledge Base article 1002117.

The minimum number of heartbeat datastores is two and the maximum is five. For vSphere HA datastore heartbeating to function correctly in any type of failure scenario, VMware recommends increasing the number of heartbeat datastores from two to four in a stretched cluster environment. This provides full redundancy for both data center locations. Defining four specific datastores as preferred heartbeat datastores is also recommended, selecting two from one site and two from the other. This enables vSphere HA to heartbeat to a datastore even in the case of a connection failure between sites and the loss of a datastore. Subsequently, it enables vSphere HA to determine the state of a host in almost any scenario.

Adding an advanced setting called *das.heartbeatDsPerHost* can increase the number of heartbeat datastores. This is shown in the screenshot below.



To designate specific datastores as heartbeat devices,VMware recommends using **Select any of the cluster datastores taking into account my preferences**. This enables vSphere HA to select any other datastore if the four designated datastores that have been manually selected become unavailable. VMware recommends selecting two datastores in each location to ensure that datastores are available at each site in the case of a site partition.

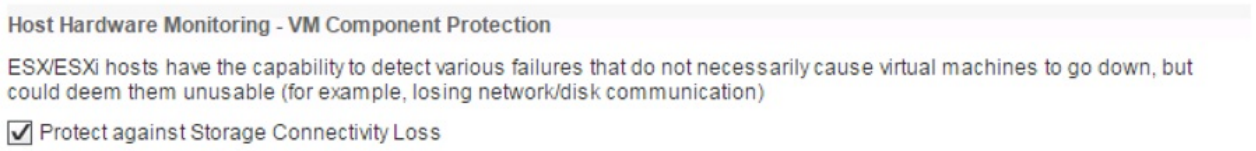


Permanent Device Loss and All Paths Down Scenarios

As of vSphere 6.0, enhancements have been introduced to enable an automated failover of VMs residing on a datastore that has either an all paths down (APD) or a permanent device loss (PDL) condition. PDL is applicable only to block storage devices.

A PDL condition, as is discussed in one of our failure scenarios, is a condition that is communicated by the array controller to the ESXi host via a SCSI sense code. This condition indicates that a device (LUN) has become unavailable and is likely permanently unavailable. An example scenario in which this condition is communicated by the array is when a LUN is set offline. This condition is used in nonuniform models during a failure scenario to ensure that the ESXi host takes appropriate action when access to a LUN is revoked. When a full storage failure occurs, it is impossible to generate the PDL condition because there is no communication possible between the array and the ESXi host. This state is identified by the ESXi host as an APD condition. Another example of an APD condition is where the storage network has failed completely. In this scenario, the ESXi host also does not detect what has happened with the storage and declares an APD.

To enable vSphere HA to respond to both an APD and a PDL condition, vSphere HA must be configured in a specific way. VMware recommends enabling VM Component Protection (VMCP). After the creation of the cluster, VMCP must be enabled, as is shown below.



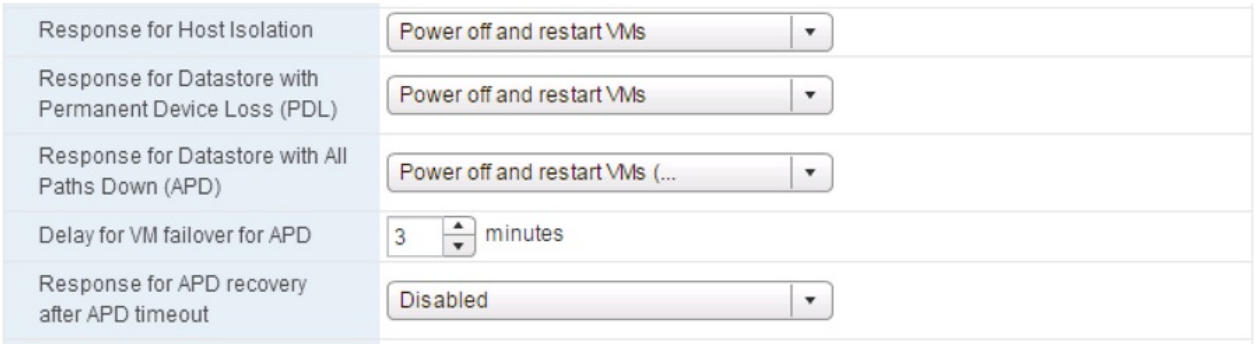
The configuration screen can be found as follows:

- Log in to VMware vSphere Web Client.

- Click **Hosts and Clusters**.
- Click the cluster object.
- Click the **Manage** tab.
- Click **vSphere HA** and then **Edit**.
- Select **Protect against Storage Connectivity Loss**.
- Select individual functionality, as described in the following, by opening **Failure conditions and VM response**.

The configuration for PDL is basic. In the **Failure conditions and VM response** section, the response following detection of a PDL condition can be configured. VMware recommends setting this to **Power off and restart VMs**. When this condition is detected, a VM is restarted instantly on a healthy host within the vSphere HA cluster.

For an APD scenario, configuration must occur in the same section, as is shown in the screenshot below. Besides defining the response to an APD condition, it is also possible to alter the timing and to configure the behavior when the failure is restored before the APD timeout has passed.



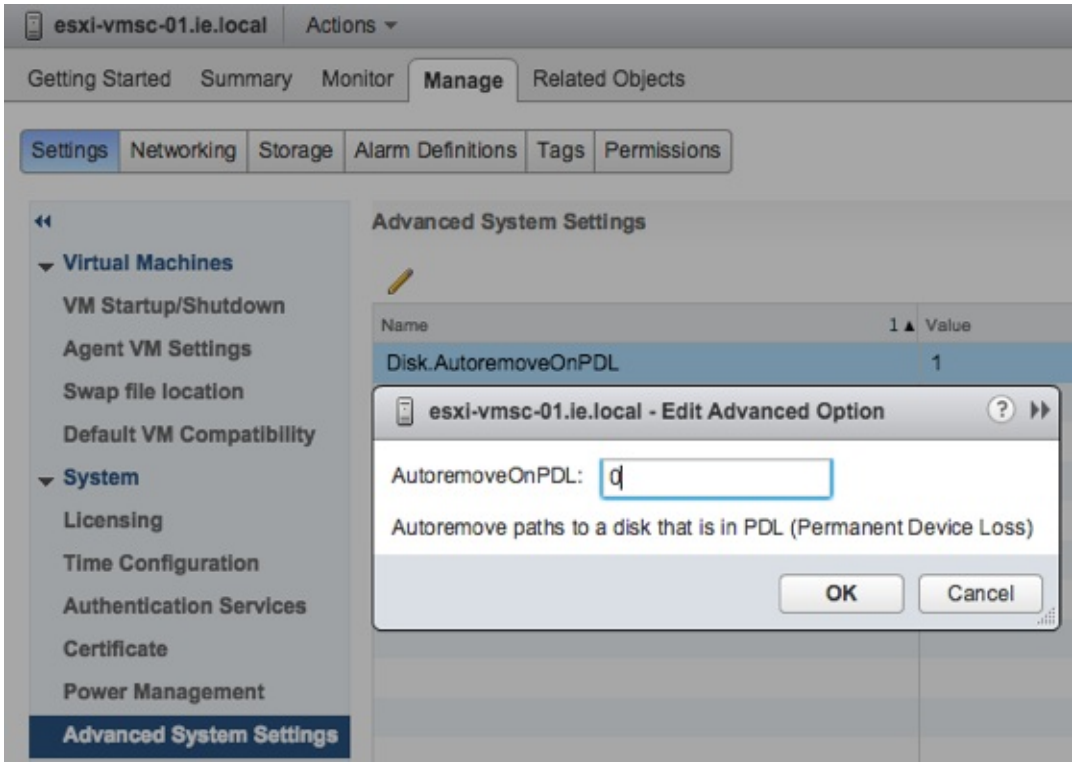
When an APD condition is detected, a timer is started. After 140 seconds, the APD condition is officially declared and the device is marked as APD timeout. When 140 seconds have passed, vSphere HA starts counting. The default vSphere HA timeout is 3 minutes. When the 3 minutes have passed, vSphere HA restarts the impacted VMs, but VMCP can be configured to respond differently if preferred. VMware recommends configuring it to **Power off and restart VMs (conservative)**.

Conservative refers to the likelihood that vSphere HA will be able to restart VMs. When set to conservative, vSphere HA restarts only the VM that is impacted by the APD if it detects that a host in the cluster can access the datastore on which the VM resides. In the case of *aggressive*, vSphere HA attempts to restart the VM even if it doesn't detect the state of the other hosts. This can lead to a situation in which a VM is not restarted because there is no host that has access to the datastore on which the VM is located.

If the APD is lifted and access to the storage is restored before the timeout has passed, vSphere HA does not unnecessarily restart the VM unless explicitly configured to do so. If a response is chosen even when the environment has recovered from the APD condition,

Response for APD recovery after APD timeout can be configured to **Reset VMs**. VMware recommends leaving this setting disabled.

With the release of vSphere 5.5, an advanced setting called **Disk.AutoremoveOnPDL** was introduced. It is implemented by default. This functionality enables vSphere to remove devices that are marked as PDL and helps prevent reaching, for example, the 256-device limit for an ESXi host. However, if the PDL scenario is solved and the device returns, the ESXi host's storage system must be rescanned before this device appears. VMware recommends disabling **Disk.AutoremoveOnPDL** in the host advanced settings by setting it to **0** for vSphere environment using 5.5. For vSphere environments using 6.0 or later this setting should be set to 1 (default value) due to a change in PDL handling behaviour with vSphere 6.0 and higher.



vSphere DRS

vSphere DRS is used in many environments to distribute load within a cluster. It offers many other features that can be very helpful in stretched cluster environments. VMware recommends enabling vSphere DRS to facilitate load balancing across hosts in the cluster. The vSphere DRS load-balancing calculation is based on CPU and memory use. Care should be taken with regard to both storage and networking resources as well as to traffic flow. To avoid storage and network traffic overhead in a stretched cluster environment, VMware recommends implementing vSphere DRS affinity rules to enable a logical

separation of VMs. This subsequently helps improve availability. For VMs that are responsible for infrastructure services, such as Microsoft Active Directory and DNS, it assists by ensuring separation of these services across sites.

vSphere DRS affinity rules also help prevent unnecessary downtime, and storage and network traffic flow overhead, by enforcing preferred site affinity. VMware recommends aligning vSphere VM-to-host affinity rules with the storage configuration—that is, setting VM-to-host affinity rules with a preference that a VM run on a host at the same site as the array that is configured as the primary read\/write node for a given datastore. For example, in our test configuration, VMs stored on the Frimley01 datastore are set with VM-to-host affinity with a preference for hosts in the Frimley data center. This ensures that in the case of a network connection failure between sites, VMs do not lose connection with the storage system that is primary for their datastore. VM-to-host affinity rules aim to ensure that VMs stay local to the storage primary for that datastore. This coincidentally also results in all read I\/O's staying local.

NOTE: *Different storage vendors use different terminology to describe the relationship of a LUN to a particular array or controller. For the purposes of this document, we use the generic term "storage site affinity," which refers to the preferred location for access to a given LUN.*

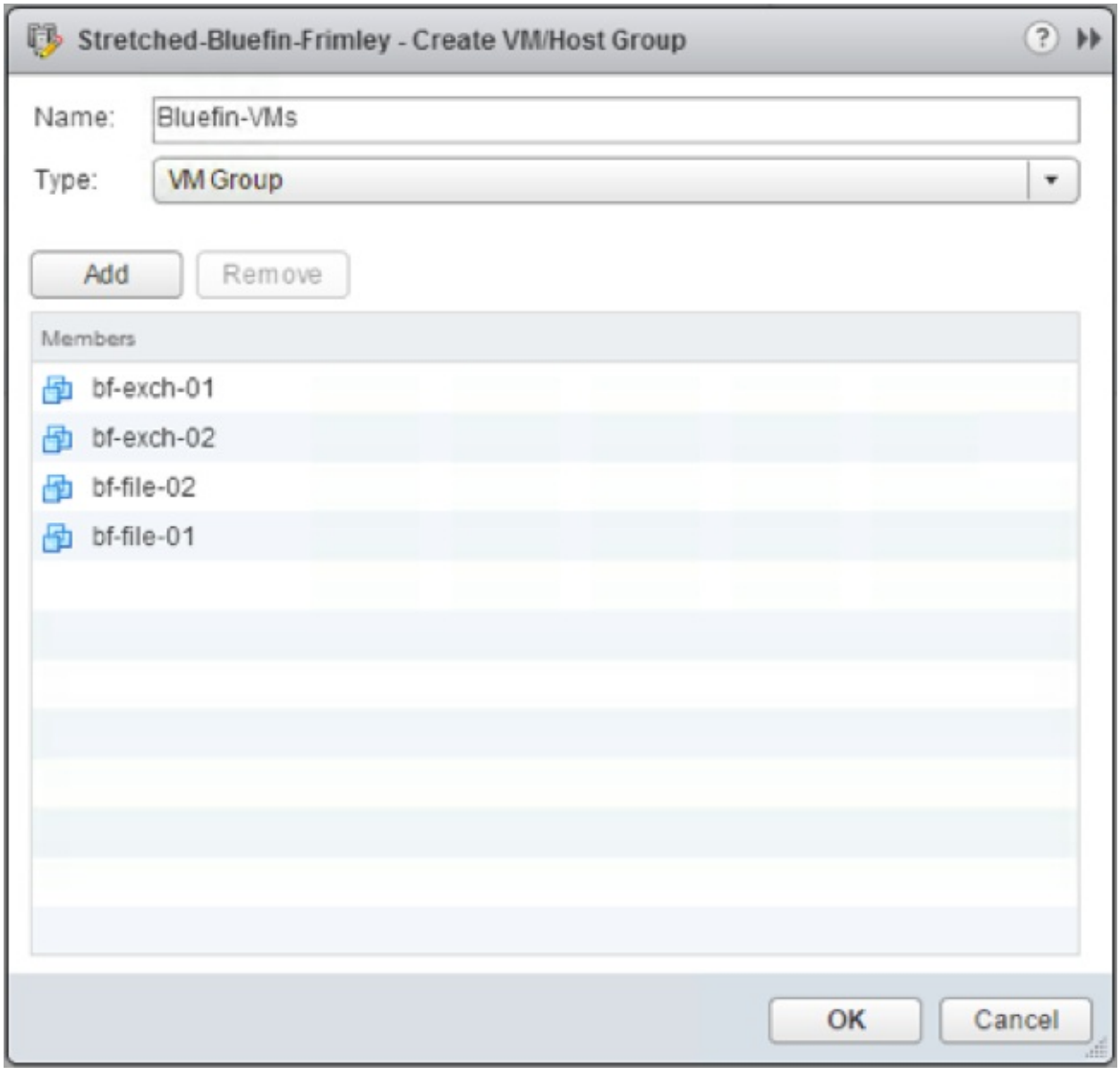
VMware recommends implementing "should rules" because these are violated by vSphere HA in the case of a full site failure. Availability of services should always prevail. In the case of "must rules," vSphere HA does not violate the rule set, and this can potentially lead to service outages. In the scenario where a full data center fails, "must rules" do not allow vSphere HA to restart the VMs, because they do not have the required affinity to start on the hosts in the other data center. This necessitates the recommendation to implement "should rules." vSphere DRS communicates these rules to vSphere HA, and these are stored in a "compatibility list" governing allowed start-up. If a single host fails, VM-to-host "should rules" are ignored by default. VMware recommends configuring vSphere HA rule settings to respect VM-to-host affinity rules where possible. With a full site failure, vSphere HA can restart the VMs on hosts that violate the rules. Availability takes preference in this scenario.



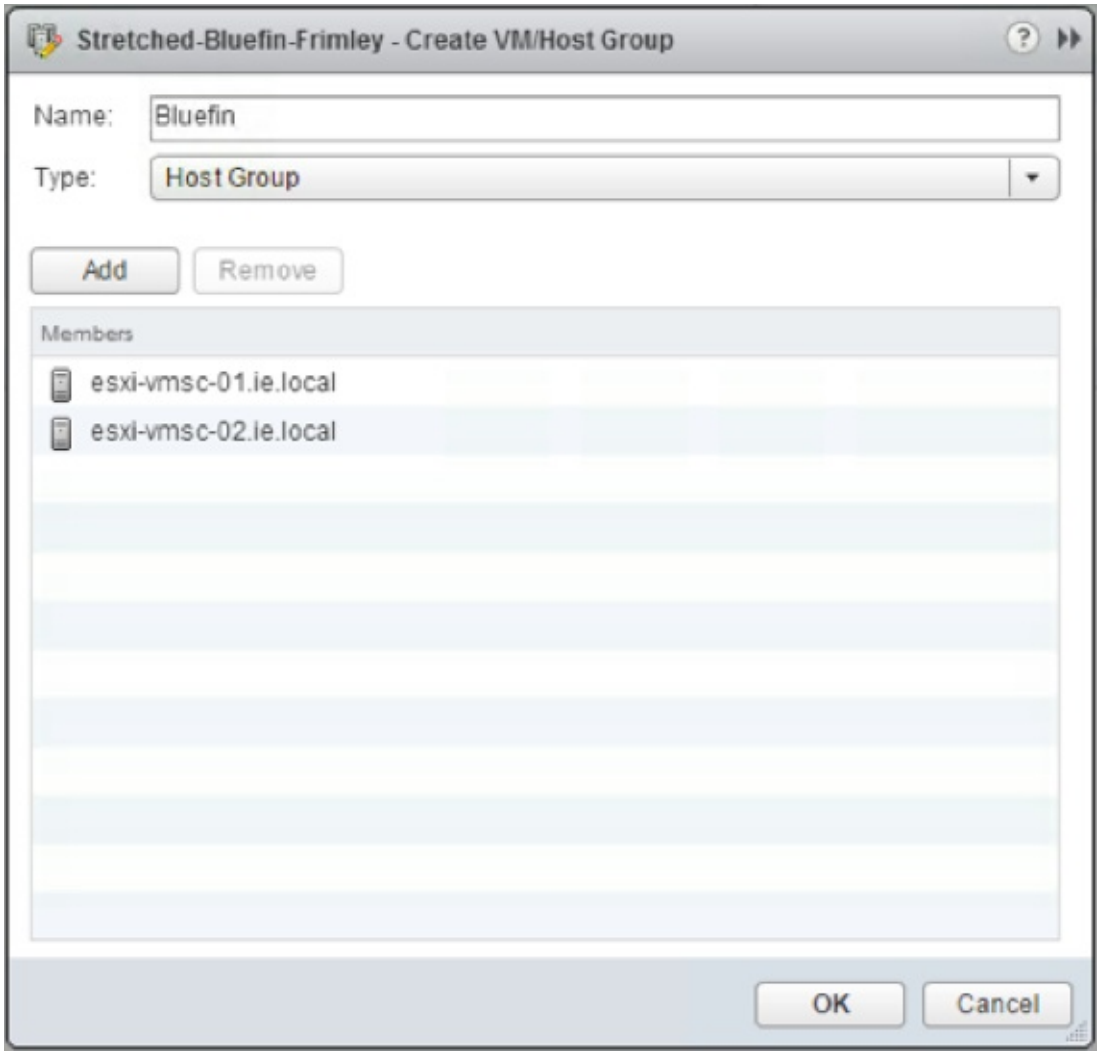
Under certain circumstances, such as massive host saturation coupled with aggressive recommendation settings, vSphere DRS can also violate "should rules." Although this is very rare, we recommend monitoring for violation of these rules because a violation might impact availability and workload performance.

VMware recommends manually defining “sites” by creating a group of hosts that belong to a site and then adding VMs to these sites based on the affinity of the datastore on which they are provisioned. In our scenario, only a limited number of VMs were provisioned. VMware recommends automating the process of defining site affinity by using tools such as VMware vCenter Orchestrator™ or VMware vSphere PowerCLI™. If automating the process is not an option, use of a generic naming convention is recommended to simplify the creation of these groups. VMware recommends that these groups be validated on a regular basis to ensure that all VMs belong to the group with the correct site affinity.

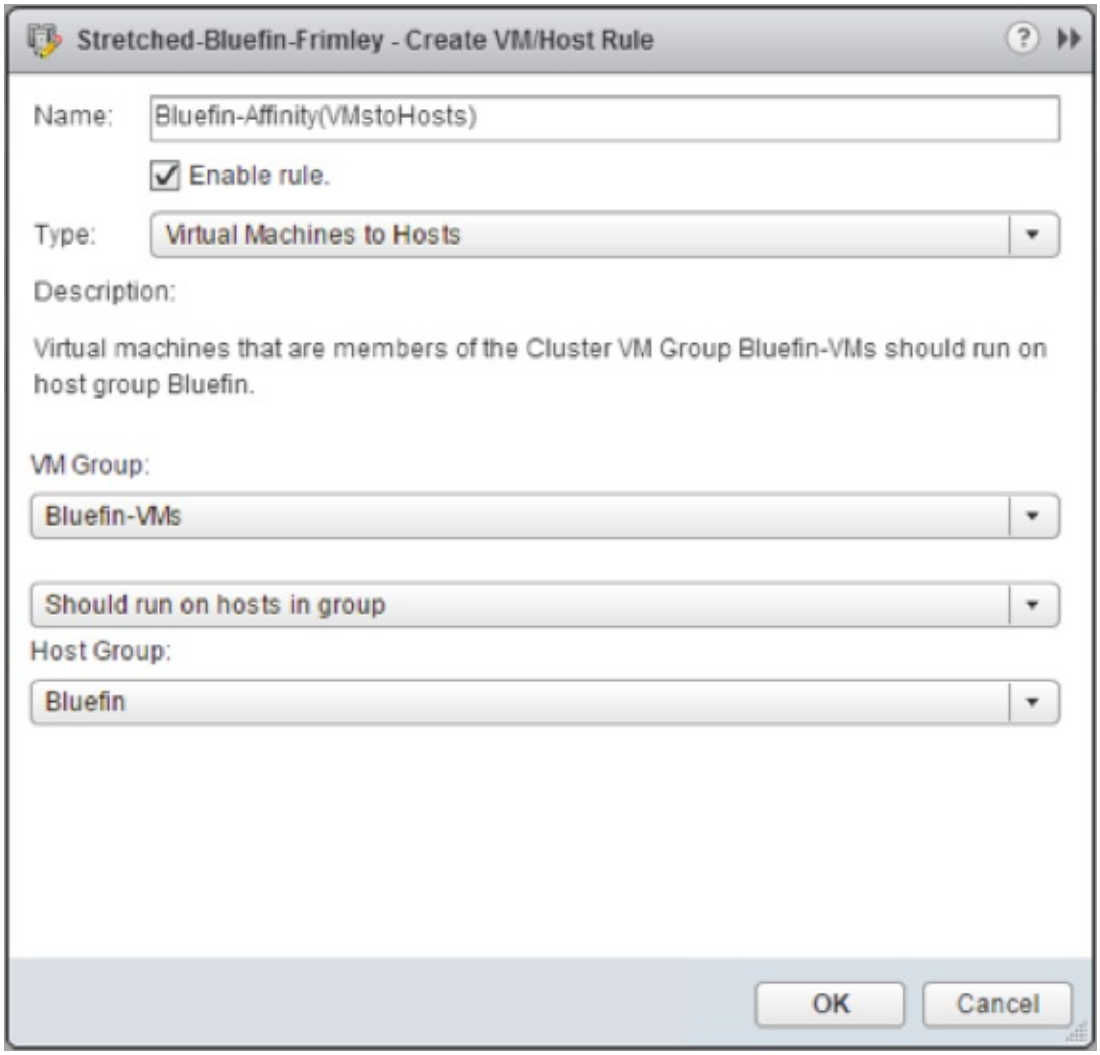
The following screenshots depict the configuration used for our scenario. In the first screenshot, all VMs that should remain local to the Bluefin data center are added to the Bluefin VM group.



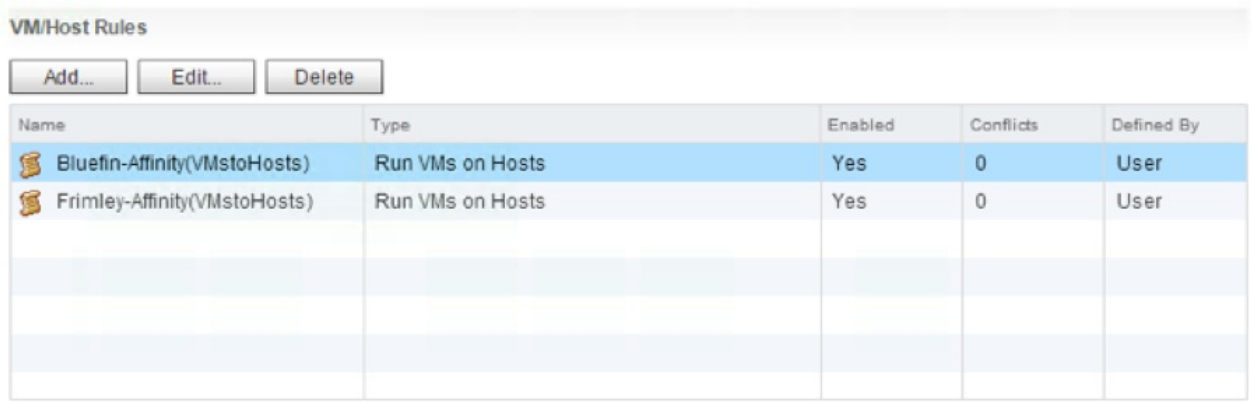
Next, a Bluefin host group is created that contains all hosts residing in this location.



Next, a new rule is created that is defined as a “should run on rule.” It links the host group and the VM group for the Bluefin location.



This should be done for both locations, which should result in two rules.



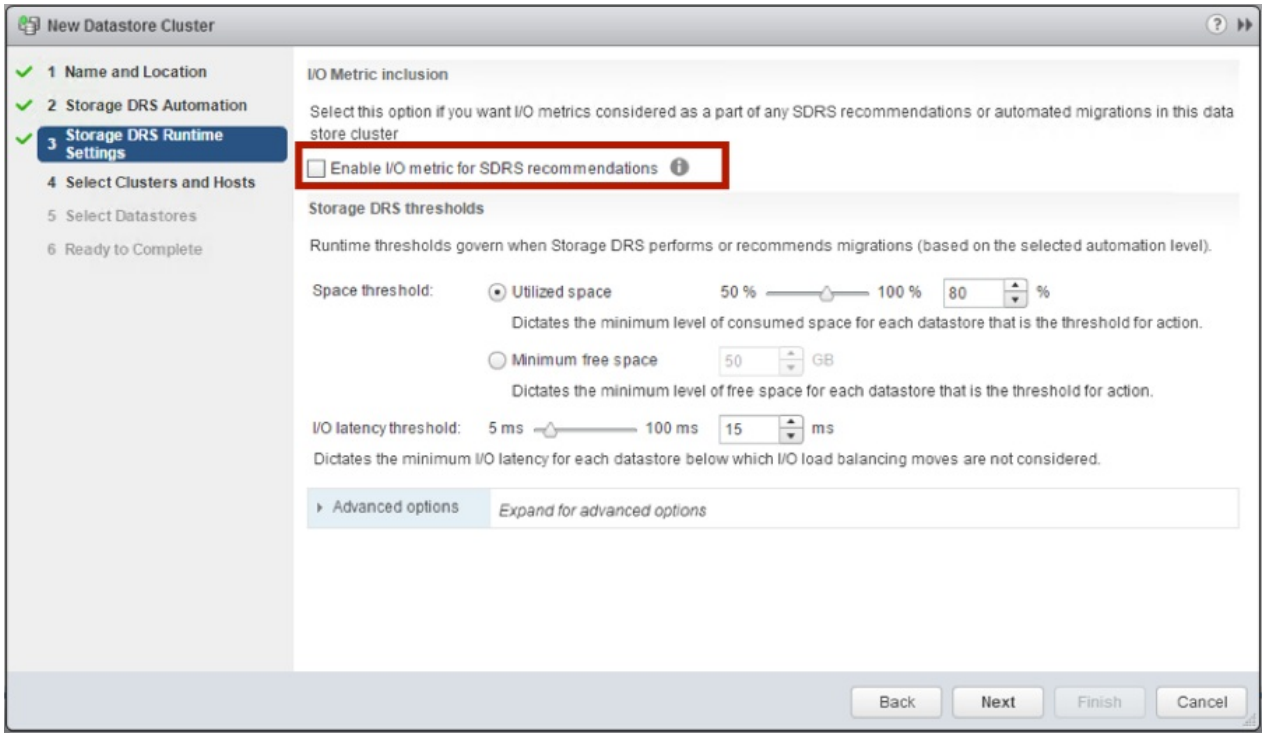
Correcting Affinity Rule Violation

vSphere DRS assigns a high priority to correcting affinity rule violations. During invocation, the primary goal of vSphere DRS is to correct any violations and generate recommendations to migrate VMs to the hosts listed in the host group. These migrations have a higher priority than load-balancing moves and are started before them.

vSphere DRS is invoked every 5 minutes by default, but it is also triggered if the cluster detects changes. For instance, when a host reconnects to the cluster, vSphere DRS is invoked and generates recommendations to correct the violation. Our testing has shown that vSphere DRS generates recommendations to correct affinity rules violations within 30 seconds after a host reconnects to the cluster. vSphere DRS is limited by the overall capacity of the vSphere vMotion network, so it might take multiple invocations before all affinity rule violations are corrected.

vSphere Storage DRS

vSphere Storage DRS enables aggregation of datastores to a single unit of consumption from an administrative perspective, and it balances VM disks when defined thresholds are exceeded. It ensures that sufficient disk resources are available to a workload. VMware recommends enabling vSphere Storage DRS with I\/O Metric disabled. The use of I\/O Metric or VMware vSphere Storage I\/O Control is not supported in a vMSC configuration, as is described in VMware Knowledge Base article 2042596.

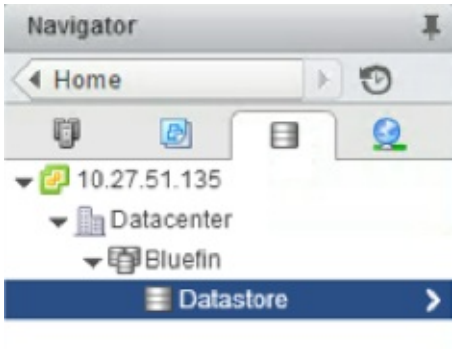


vSphere Storage DRS uses vSphere Storage vMotion to migrate VM disks between datastores within a datastore cluster. Because the underlying stretched storage systems use synchronous replication, a migration or series of migrations have an impact on replication traffic and might cause the VMs to become temporarily unavailable due to contention for network resources during the movement of disks. Migration to random datastores can also potentially lead to additional I\/O latency in uniform host access configurations if VMs are not migrated along with their virtual disks. For example, if a VM residing on a host at site A has its disk migrated to a datastore at site B, it continues operating but with potentially degraded

performance. The VM's disk reads now are subject to the increased latency associated with reading from the virtual iSCSI IP at site B. Reads are subject to intersite latency rather than being satisfied by a local target.

To control if and when migrations occur, VMware recommends configuring vSphere Storage DRS in manual mode. This enables human validation per recommendation as well as recommendations to be applied during off-peak hours, while gaining the operational benefit and efficiency of the initial placement functionality.

VMware recommends creating datastore clusters based on the storage configuration with respect to storage site affinity. Datastores with a site affinity for site A should not be mixed in datastore clusters with datastores with a site affinity for site B. This enables operational consistency and eases the creation and ongoing management of vSphere DRS VM-to-host affinity rules. Ensure that all vSphere DRS VM-to-host affinity rules are updated accordingly when VMs are migrated via vSphere Storage vMotion between datastore clusters and when crossing defined storage site affinity boundaries. To simplify the provisioning process, VMware recommends aligning naming conventions for datastore clusters and VM-to-host affinity rules.



The naming convention used in our testing gives both datastores and datastore clusters a site-specific name to provide ease of alignment of vSphere DRS host affinity with VM deployment in the correlate site.

Failure Scenarios

There are many failures that can be introduced in clustered systems. But in a properly architected environment, vSphere HA, vSphere DRS, and the storage subsystem do not detect many of these. We do not address the zero-impact failures, such as the failure of a single network cable, because they are explained in depth in the documentation provided by the storage vendor of the various solutions. We discuss the following "common" failure scenarios:

- Single-host failure in Frimley data center

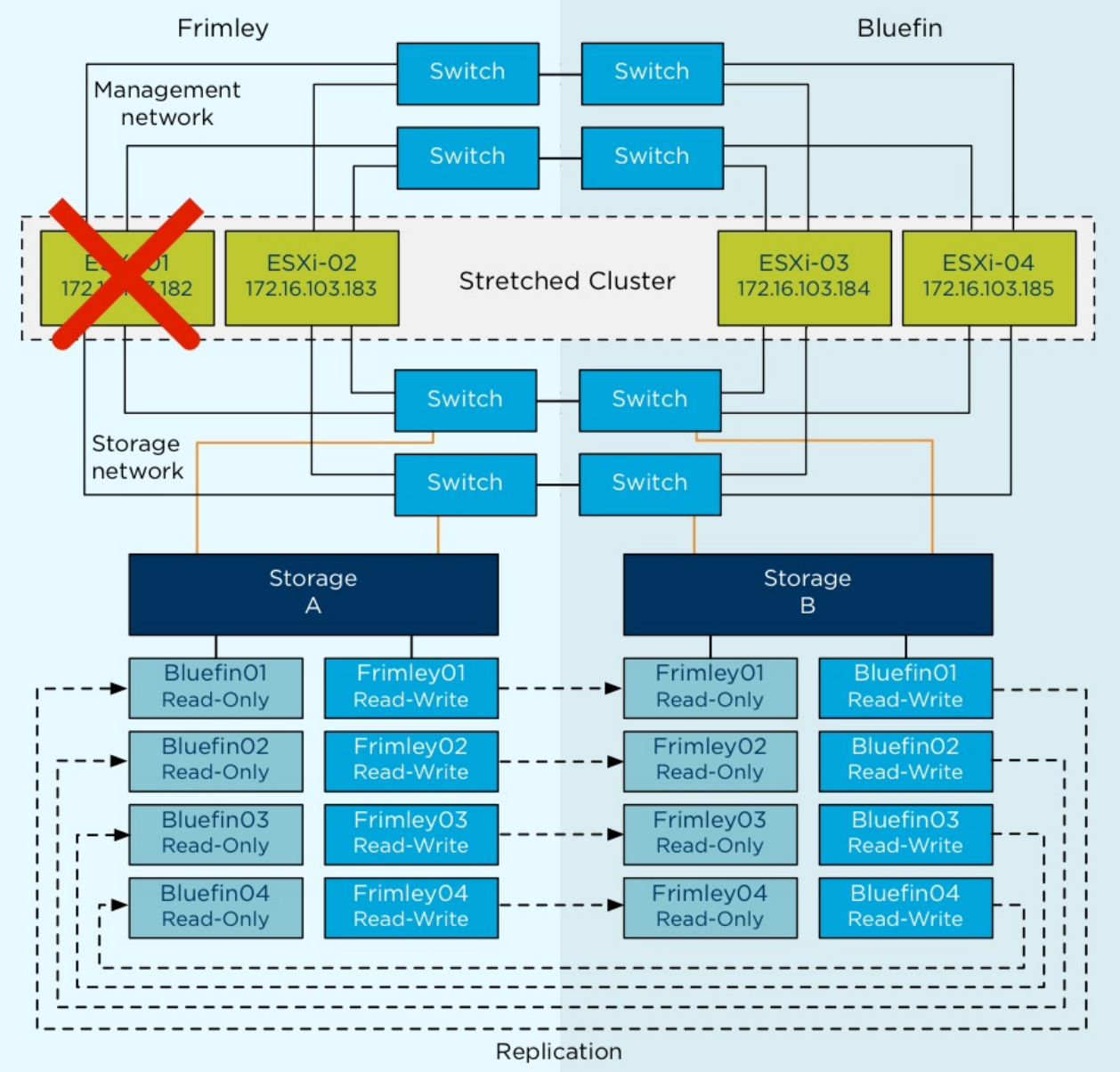
- Single-host isolation in Frimley data center
- Storage partition
- Data center partition
- Disk shelf failure in Frimley data center
- Full storage failure in Frimley data center
- Full compute failure in Frimley data center
- Full compute failure in Frimley data center and full storage failure in Bluefin data center
- Loss of complete Frimley data center

We also examine scenarios in which specific settings are incorrectly configured. These settings determine the availability and recoverability of VMs in a failure scenario. It is important to understand the impact of misconfigurations such as the following:

- Incorrectly configured VM-to-host affinity rules
- Incorrectly configured heartbeat datastores
- Incorrectly configured isolation address
- Incorrectly configured PDL handling
- vCenter Server split-brain scenario

Single-Host Failure in Frimley Data Center

In this scenario, we describe the complete failure of a host in Frimley data center. This scenario is depicted below.



Result: vSphere HA successfully restarted all VMs in accordance with VM-to-host affinity rules.

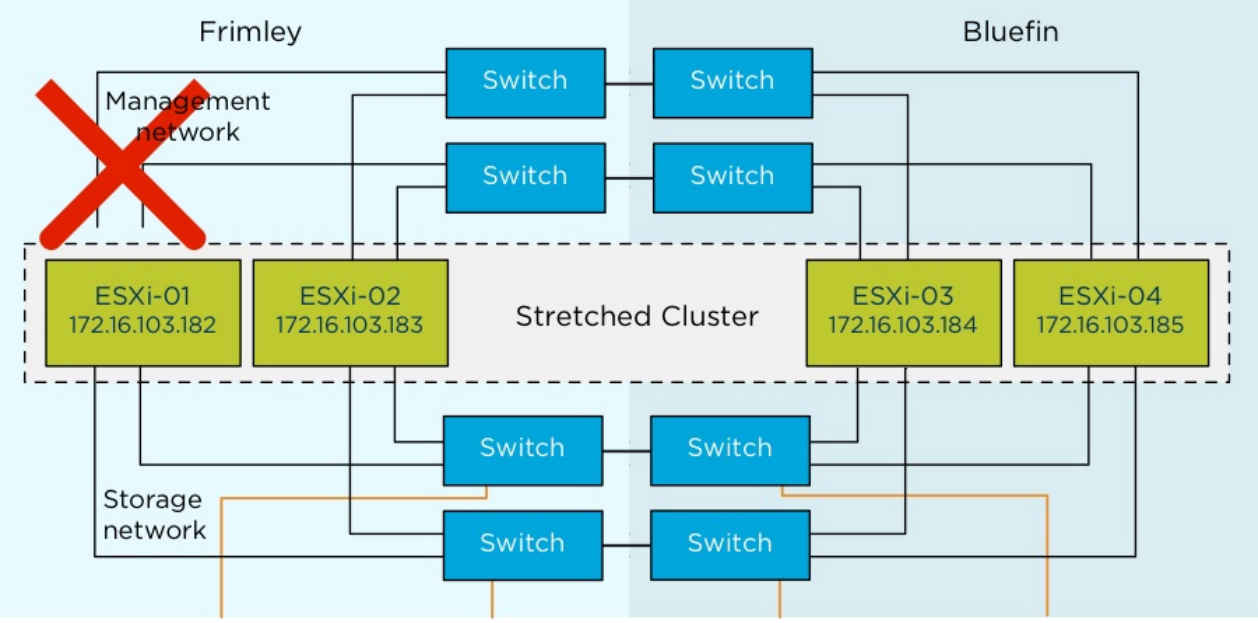
Explanation: If a host fails, the cluster's vSphere HA master node detects the failure because it no longer is receiving network heartbeats from the host. Then the master starts monitoring for datastore heartbeats. Because the host has failed completely, it cannot generate datastore heartbeats; these too are detected as missing by the vSphere HA master node. During this time, a third availability check—pinging the management addresses of the failed hosts—is conducted. If all of these checks return as unsuccessful, the master declares the missing host as dead and attempts to restart all the protected VMs that had been running on the host before the master lost contact with the host.

The vSphere VM-to-host affinity rules defined on a cluster level are "should rules." vSphere HA VM-to-host affinity rules should be respected so all VMs are restarted within the correct site.

However, if the host elements of the VM-to-host group are temporarily without resources, or if they are unavailable for restarts for any other reason, vSphere HA can disregard the rules and restart the remaining VMs on any of the remaining hosts in the cluster, regardless of location and rules. If this occurs, vSphere DRS attempts to correct any violated affinity rules at the first invocation and automatically migrates VMs in accordance with their affinity rules to bring VM placement in alignment. VMware recommends manually invoking vSphere DRS after the cause for the failure has been identified and resolved. This ensures that all VMs are placed on hosts in the correct location to avoid possible performance degradation due to misplacement.

Single-Host Isolation in Frimley Data Center

In this scenario, we describe the response to isolation of a single host in Frimley data center from the rest of the network.



Result: VMs remain running because isolation response is configured to **Disabled**.

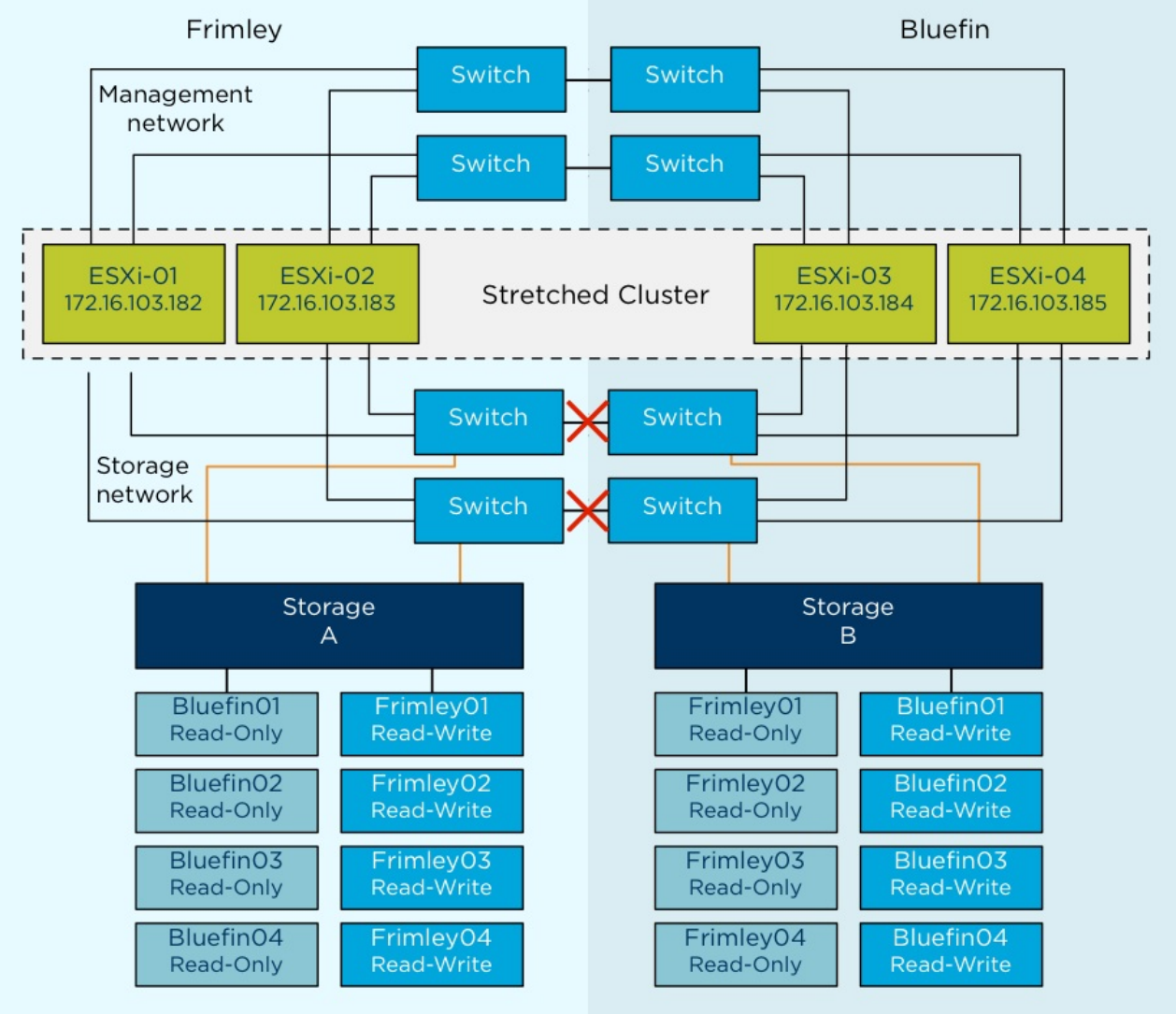
Explanation: When a host is isolated, the vSphere HA master node detects the isolation because it no longer is receiving network heartbeats from the host. Then the master starts monitoring for datastore heartbeats. Because the host is isolated, it generates datastore heartbeats for the secondary vSphere HA detection mechanism. Detection of valid host heartbeats enables the vSphere HA master node to determine that the host is running but is isolated from the network. Depending on the isolation response configured, the impacted host can power off or shut down VMs or can leave them powered on. The isolation response is triggered 30 seconds after the host has detected that it is isolated.

VMware recommends aligning the isolation response to business requirements and physical constraints. From a best practices perspective, **Disabled** is the recommended isolation response setting for the majority of environments. Isolated hosts are rare in a properly architected environment, given the built-in redundancy of most modern designs. In environments that use network-based storage protocols, such as iSCSI and NFS, and where networks are converged, the recommended isolation response is **power off**. In these environments, it is more likely that a network outage that causes a host to become isolated also affects the host's ability to communicate to the datastores.

If an isolation response different from the recommended **Disabled** is selected and a **power off** or **shut down** response is triggered, the vSphere HA master restarts VMs on the remaining nodes in the cluster. The vSphere VM-to-host affinity rules defined on a cluster level are "should rules." However, because the vSphere HA rule settings specify that the vSphere HA VM-to-host affinity rules should be respected, all VMs are restarted within the correct site under "normal" circumstances.

Storage Partition

In this scenario, a failure has occurred on the storage network between data centers, as is depicted below.



Result: VMs remain running with no impact.

Explanation: Storage site affinity is defined for each LUN, and vSphere DRS rules align with this affinity. Therefore, because storage remains available within the site, no VM is impacted.

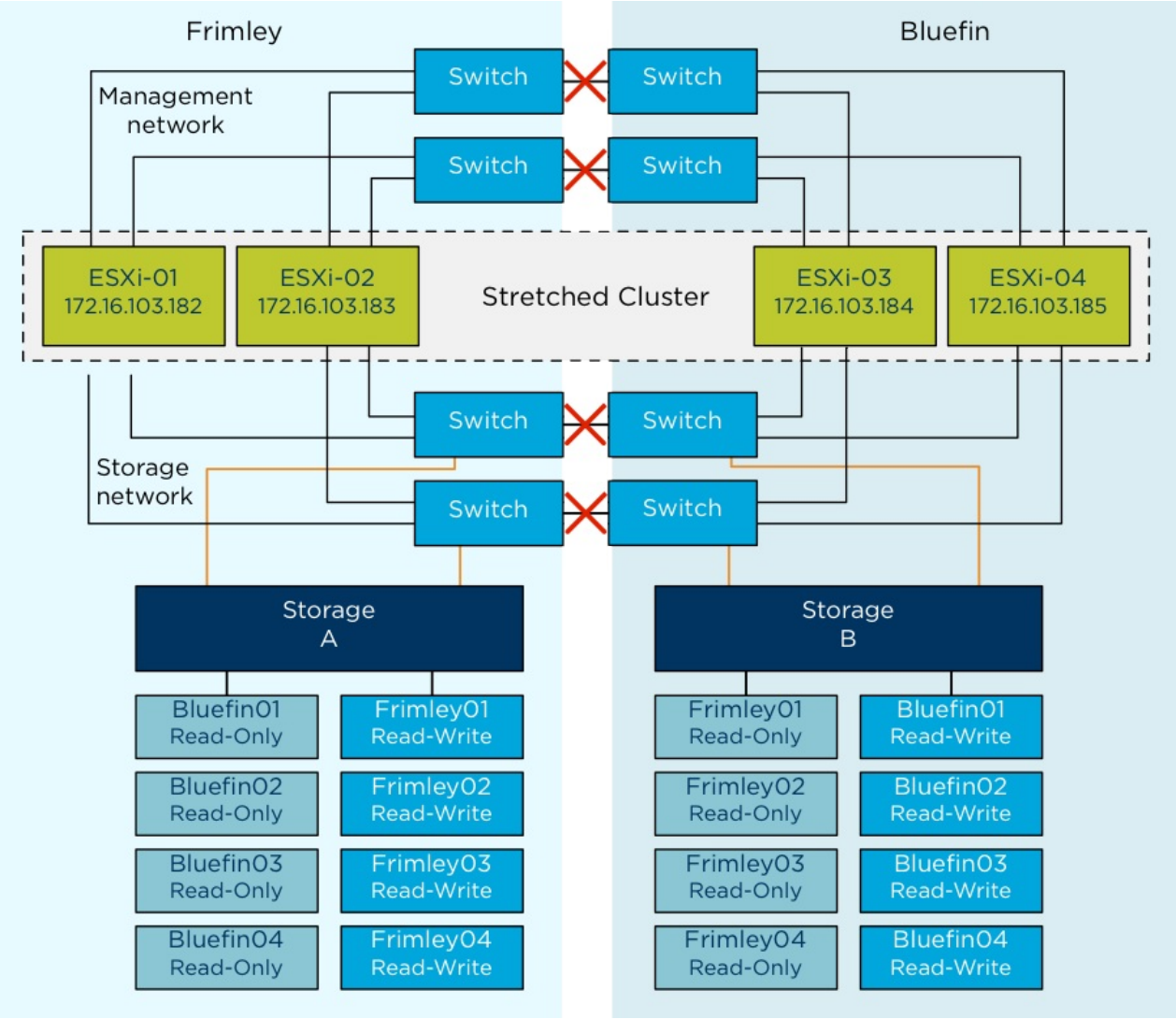
If for any reason the affinity rule for a VM has been violated and the VM is running on a host in Frimley data center while its disk resides on a datastore that has affinity with Bluefin data center, it cannot successfully issue I\/O following an intersite storage partition. This is because the datastore is in an APD condition. In this scenario, the VM can be restarted because vSphere HA is configured to respond to APD conditions. The response occurs after the 3-minute grace period has passed. This 3-minute period starts after the APD timeout of 140 seconds has passed and the APD condition has been declared.

To avoid unnecessary downtime in an APD scenario, VMware recommends monitoring compliance of vSphere DRS rules. Although vSphere DRS is invoked every 5 minutes, this does not guarantee resolution of all affinity rule violations. Therefore, to prevent

unnecessary downtime, rigid monitoring is recommended that enables quick identification of anomalies such as a VM's compute's residing in one site while its storage resides in the other site.

Data Center Partition

In this scenario, the Frimley data center is isolated from the Bluefin data center, as is depicted below.



Result: VMs remain running with no impact.

Explanation: In this scenario, the two data centers are fully isolated from each other. This scenario is similar to both the storage partition and the host isolation scenario. VMs are not impacted by this failure because vSphere DRS rules were correctly implemented and no rules were violated.

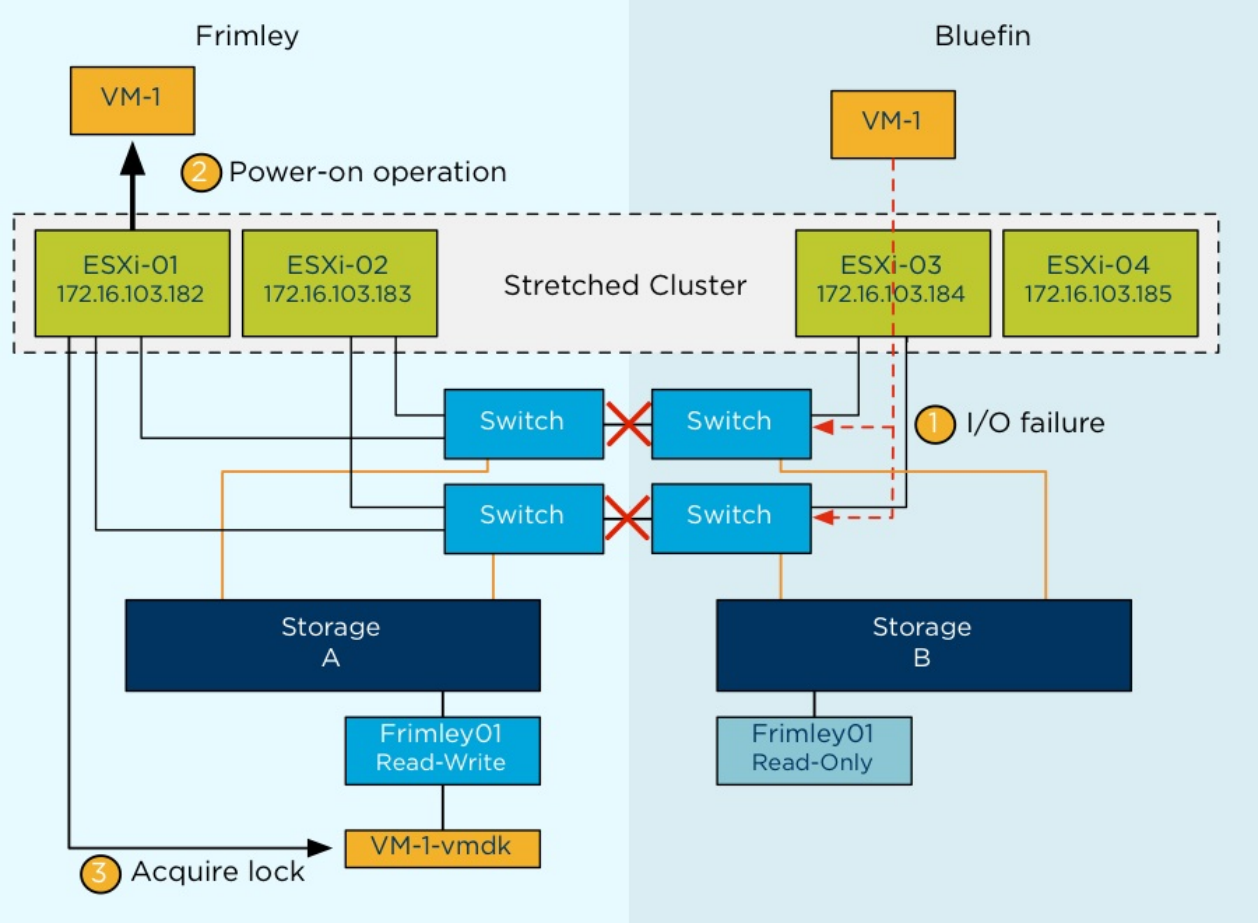
vSphere HA follows this logical process to determine which VMs require restarting during a cluster partition:

The vSphere HA master node running in Frimley data center detects that all hosts in Bluefin data center are unreachable. It first detects that no network heartbeats are being received. It then determines whether any storage heartbeats are being generated. This check does not detect storage heartbeats because the storage connection between sites also has failed, and the heartbeat datastores are updated only “locally.” Because the VMs with affinity to the remaining hosts are still running, no action is needed for them. Next, vSphere HA determines whether a restart can be attempted. However, the read\/write version of the datastores located in Bluefin data center are not accessible by the hosts in Frimley data center. Therefore, no attempt is made to start the missing VMs.

Similarly, the ESXi hosts in Bluefin data center detect that there is no master available, and they initiate a master election process. After the master has been elected, it tries to determine which VMs had been running before the failure and it attempts to restart them. Because all VMs with affinity to Bluefin data center are still running there, there is no need for a restart. Only the VMs with affinity to Frimley data center are unavailable, and vSphere HA cannot restart them because the datastores on which they are stored have affinity with Frimley data center and are unavailable in Bluefin data center.

If VM-to-host affinity rules have been violated—that is, VMs have been running at a location where their storage is not defined as read\/write by default—the behavior changes. The following sequence describes what would happen in that case:

1. The VM with affinity to Frimley data center but residing in Bluefin data center is unable to reach its datastore. This results in the VM’s being unable to write to or read from disk.
2. In Frimley data center, this VM is restarted by vSphere HA because the hosts in Frimley data center do not detect the instance’s running in Bluefin data center.
3. Because the datastore is available only to Frimley data center, one of the hosts in Frimley data center acquires a lock on the VMDK and is able to power on this VM.
4. This can result in a scenario in which the same VM is powered on and running in both data centers.



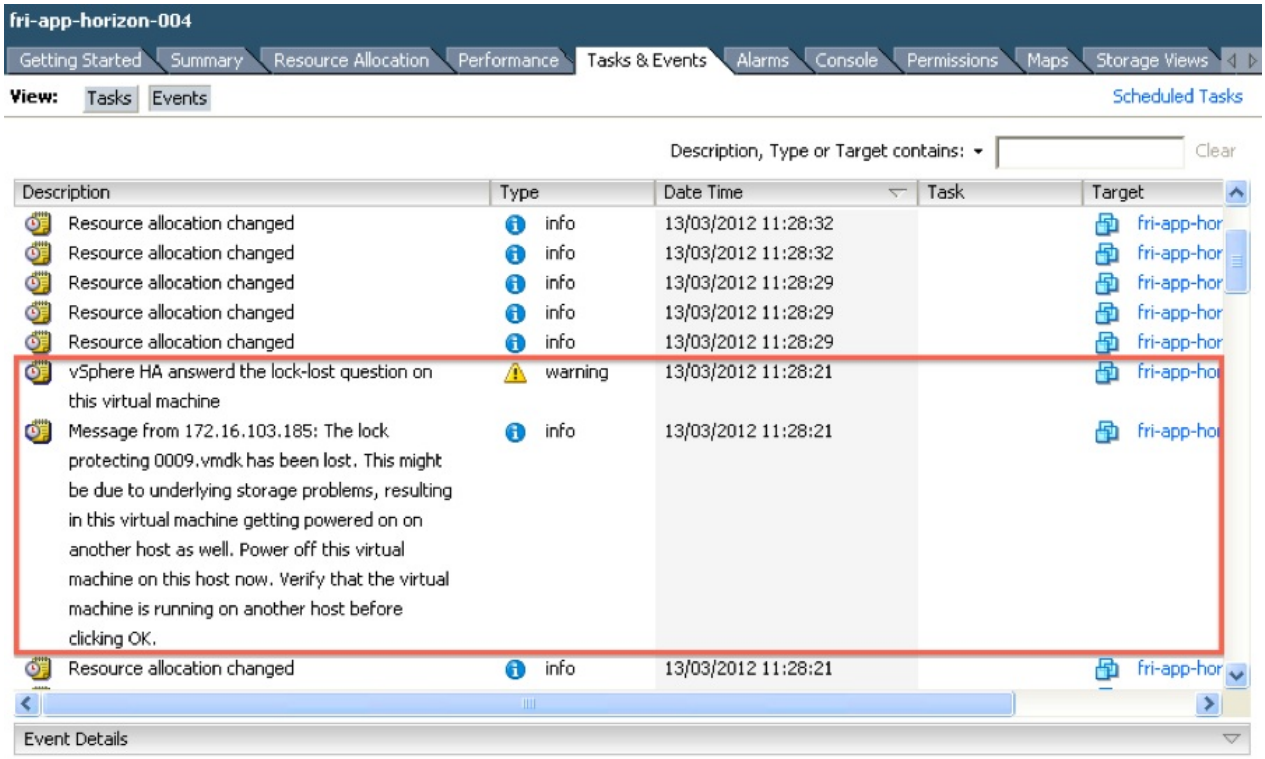
If the APD response is configured to **Power off and restart VMs (aggressive)**, as is recommended in the VM Component Protection section of this white paper, the VM is powered off after the APD timeout and the grace period have passed. This behavior is new in vSphere 6.0.

If the APD response is not correctly configured, two VMs will be running, for the following possible reasons:

- The network heartbeat from the host that is running this VM is missing because there is no connection to that site.
- The datastore heartbeat is missing because there is no connection to that site.
- A ping to the management address of the host that is running the VM fails because there is no connection to that site.
- The master located in Frimley data center detects that the VM had been powered on before the failure. Because it is unable to communicate with the VM's host in Bluefin data center after the failure, it attempts to restart the VM because it cannot detect the actual state.

If the connection between sites is restored, a classic "VM split-brain scenario" will exist. For a short period of time, two copies of the VM will be active on the network, with both having the same MAC address. Only one copy, however, will have access to the VM files, and

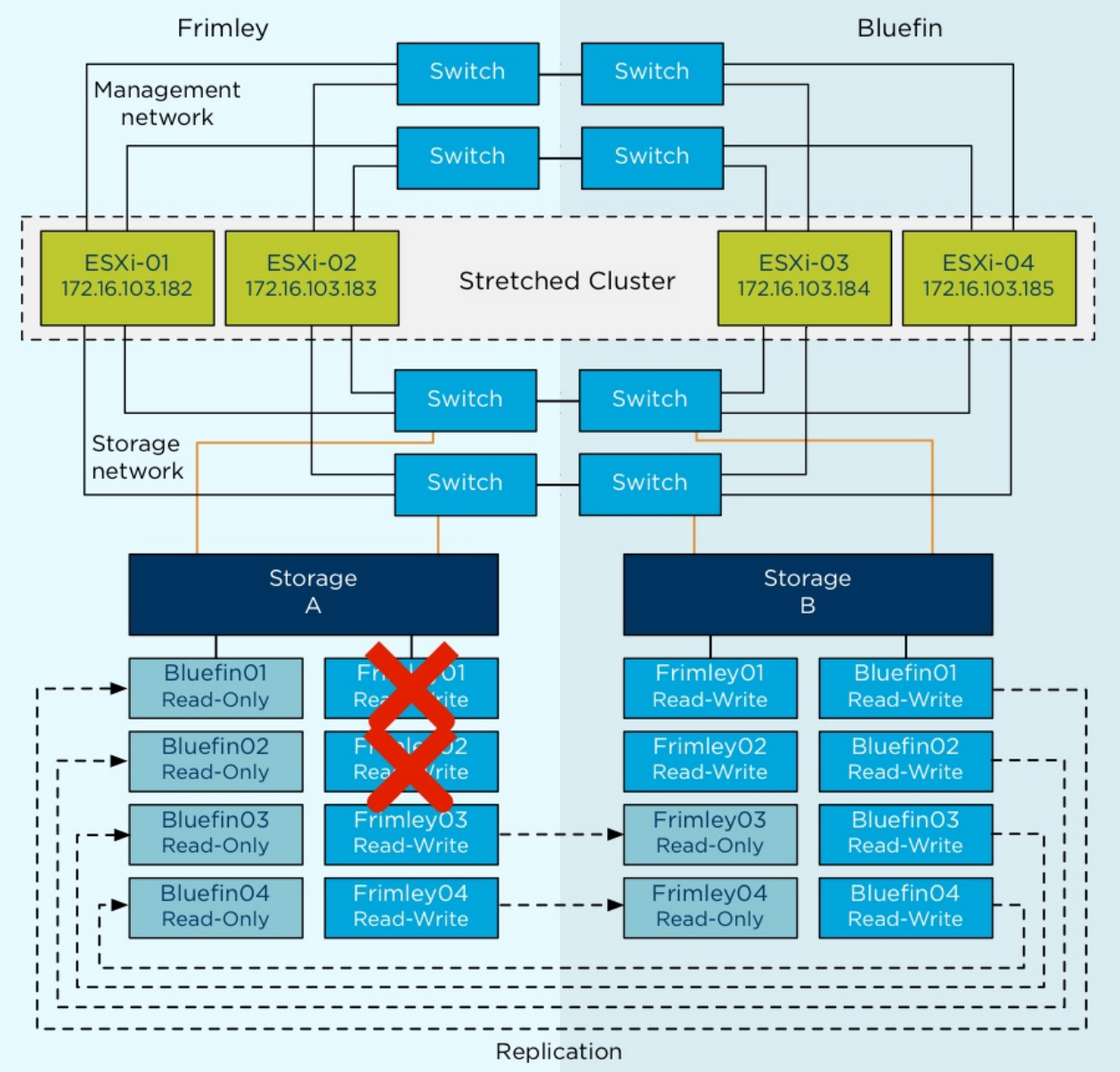
vSphere HA will detect this. As soon as this is detected, all processes belonging to the VM copy that has no access to the VM files will be killed, as is depicted below.



In this example, the downtime equates to a VM's having to be restarted. Proper maintenance of site affinity can prevent this. To avoid unnecessary downtime, VMware recommends close monitoring to ensure that vSphere DRS rules align with datastore site affinity.

Disk Shelf Failure in Frimley Data Center

In this scenario, one of the disk shelves in Frimley data center has failed. Both Frimley01 and Frimley02 on storage A are impacted.

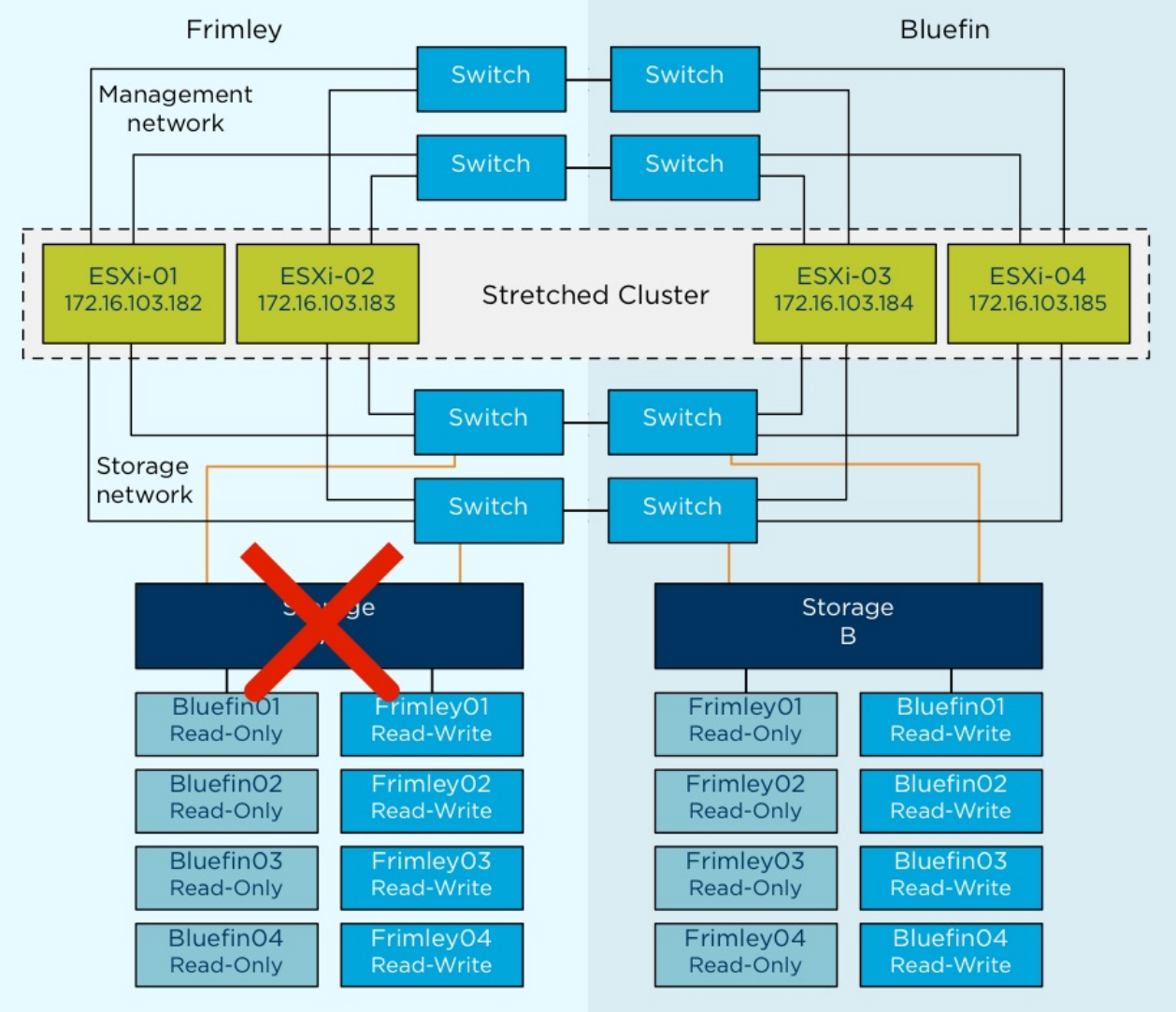


Result: VMs remain running with no impact.

Explanation: In this scenario, only a disk shelf in Frimley data center has failed. The storage processor has detected the failure and has instantly switched from the primary disk shelf in Frimley data center to the mirror copy in Bluefin data center. There is no noticeable impact to any of the VMs except for a typical short spike in I/O response time. The storage solution fully detects and handles this scenario. There is no need for a rescan of the datastores or the HBAs because the switchover is seamless and the LUNs are identical from the ESXi perspective.

Full Storage Failure in Frimley Data Center

In this scenario, a full storage system failure has occurred in Frimley data center.



Result: VMs remain running with no impact**.**

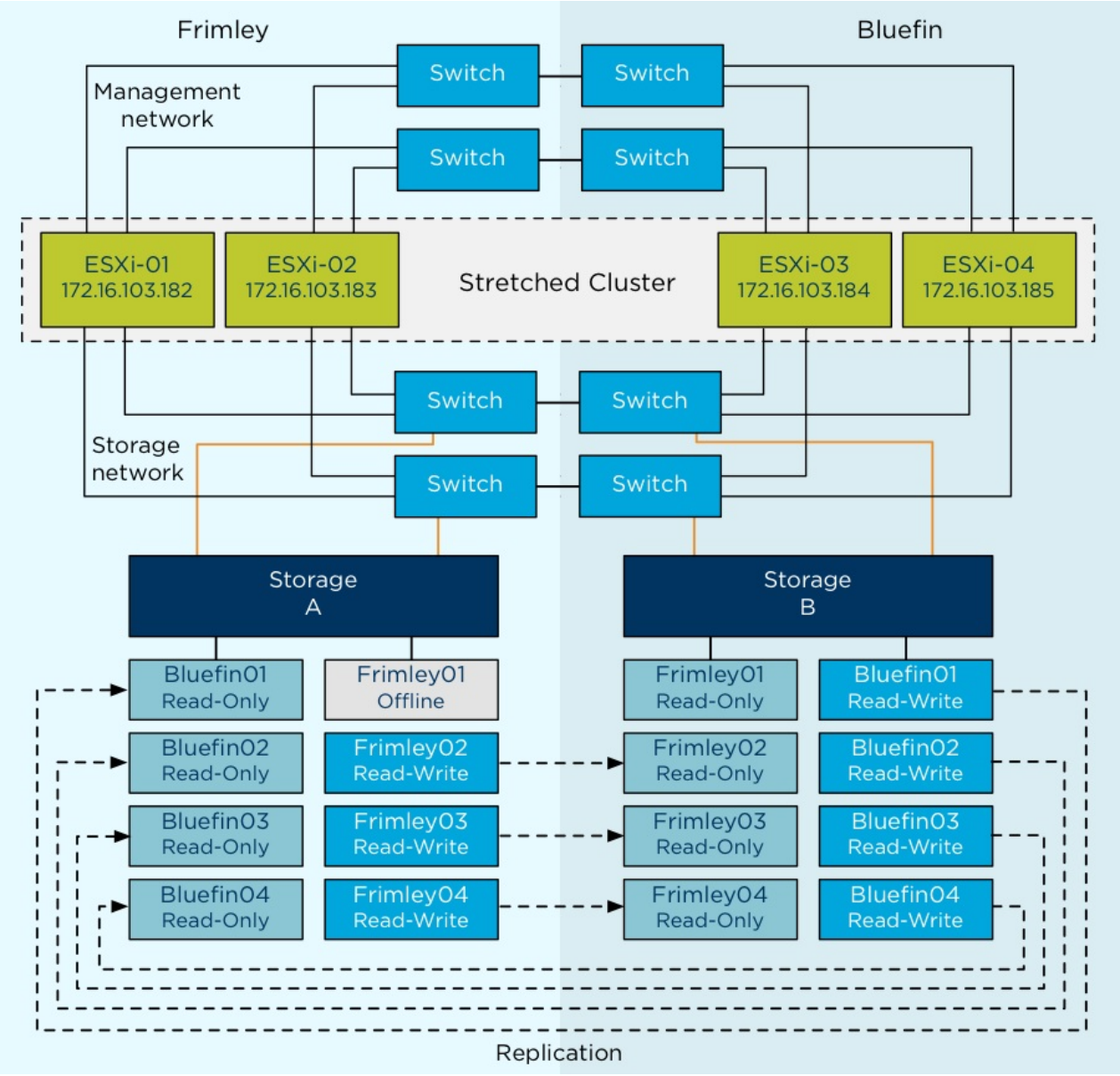
Explanation: When the full storage system fails in Frimley data center, a **take over** command must be initiated manually. As described previously, we used a NetApp MetroCluster configuration to describe this behavior. This **take over** command is particular to NetApp environments; depending on the implemented storage system, the required procedure can differ. After the command has been initiated, the mirrored, read-only copy of each of the failed datastores is set to read\/write and is instantly accessible. We have described this process on an extremely high level. For more details, refer to the storage vendor’s documentation.

From the VM perspective, this failover is seamless: The storage controllers handle this, and no action is required from either the vSphere or storage administrator. All I\/O now passes across the intrasite connection to the other data center because VMs remain running in Frimley data center while their datastores are accessible only in Bluefin data center.

vSphere HA does not detect this type of failure. Although the datastore heartbeat might be lost briefly, vSphere HA does not take action because the vSphere HA master agent checks for the datastore heartbeat only when the network heartbeat is not received for 3 seconds. Because the network heartbeat remains available throughout the storage failure, vSphere HA is not required to initiate any restarts.

Permanent Device Loss

In the scenario shown the diagram below, a permanent device loss (PDL) condition occurs because datastore Frimley01 has been taken offline for ESXi-01 and ESXi-02. PDL scenarios are uncommon in uniform configurations and are more likely to occur in a nonuniform vMSC configuration. However, a PDL scenario can, for instance, occur when the configuration of a storage group changes as in the case of this described scenario.

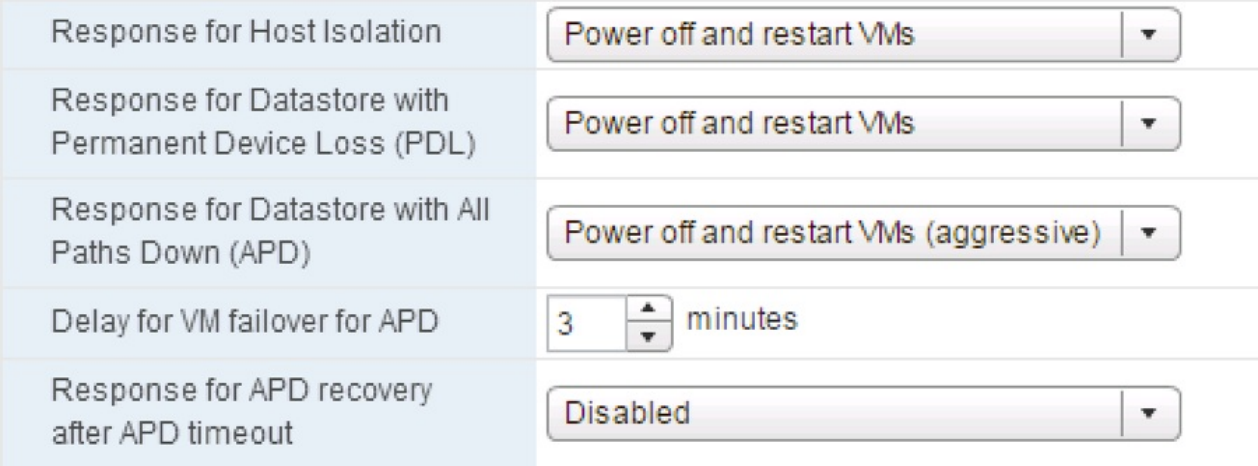


Result: VMs are restarted by vSphere HA on ESXi-03 and ESXi-04.

Explanation: When the PDL condition occurs, VMs running on datastore Frimley01 on hosts ESXi-01 and ESXi-02 are killed instantly. They then are restarted by vSphere HA on hosts within the cluster that have access to the datastore, ESXi-03 and ESXi-04 in this scenario. The PDL and killing of the VM world group can be witnessed by following the entries in the vmkernel.log file located in \/var\/log\/ on the ESXi hosts. The following is an outtake of the vmkernel.log file where a PDL is recognized and appropriate action is taken.

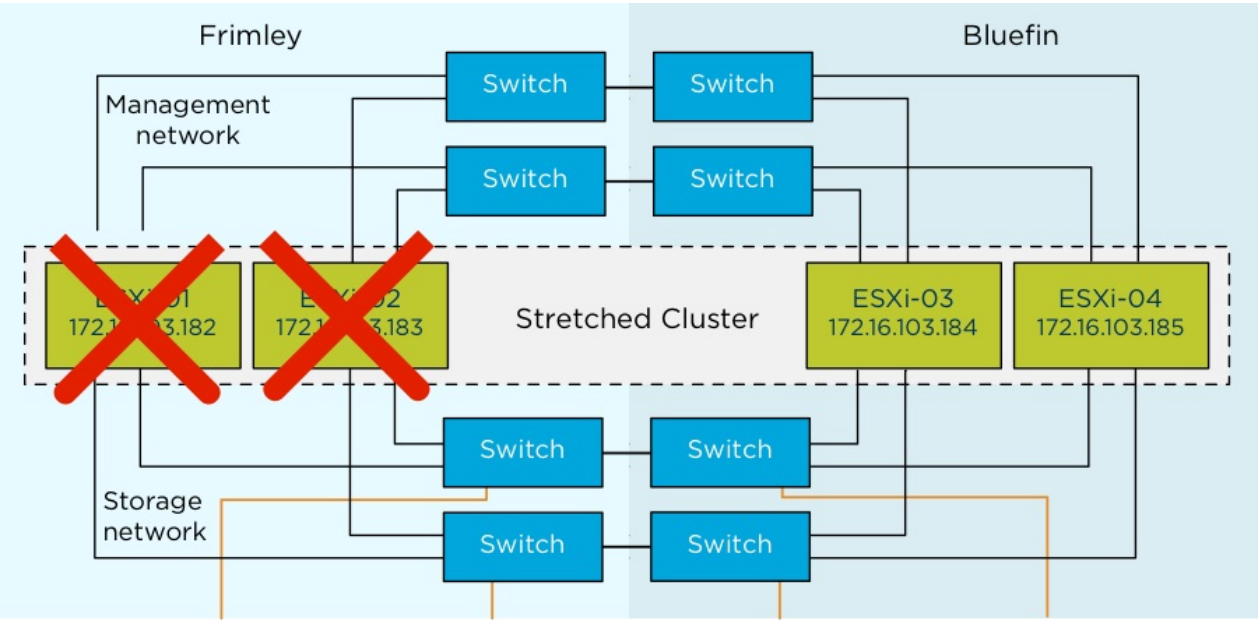
```
2012-03-14T13:39:25.085Z cpu7:4499)WARNING: VSCSI: 4055: handle 8198(vscsi4:0):opened by wid 4499 (vmm0:fri-iscsi-02) has Permanent Device Loss. Killing world group leader 4491
```

VMware recommends configuring **Response for Datastore with Permanent Device Loss (PDL)** to **Power off and restart VMs**. This setting ensures that appropriate action is taken when a PDL condition exists. The correct configuration is shown below.



Full Compute Failure in Frimley Data Center

In this scenario, a full compute failure has occurred in Frimley data center.



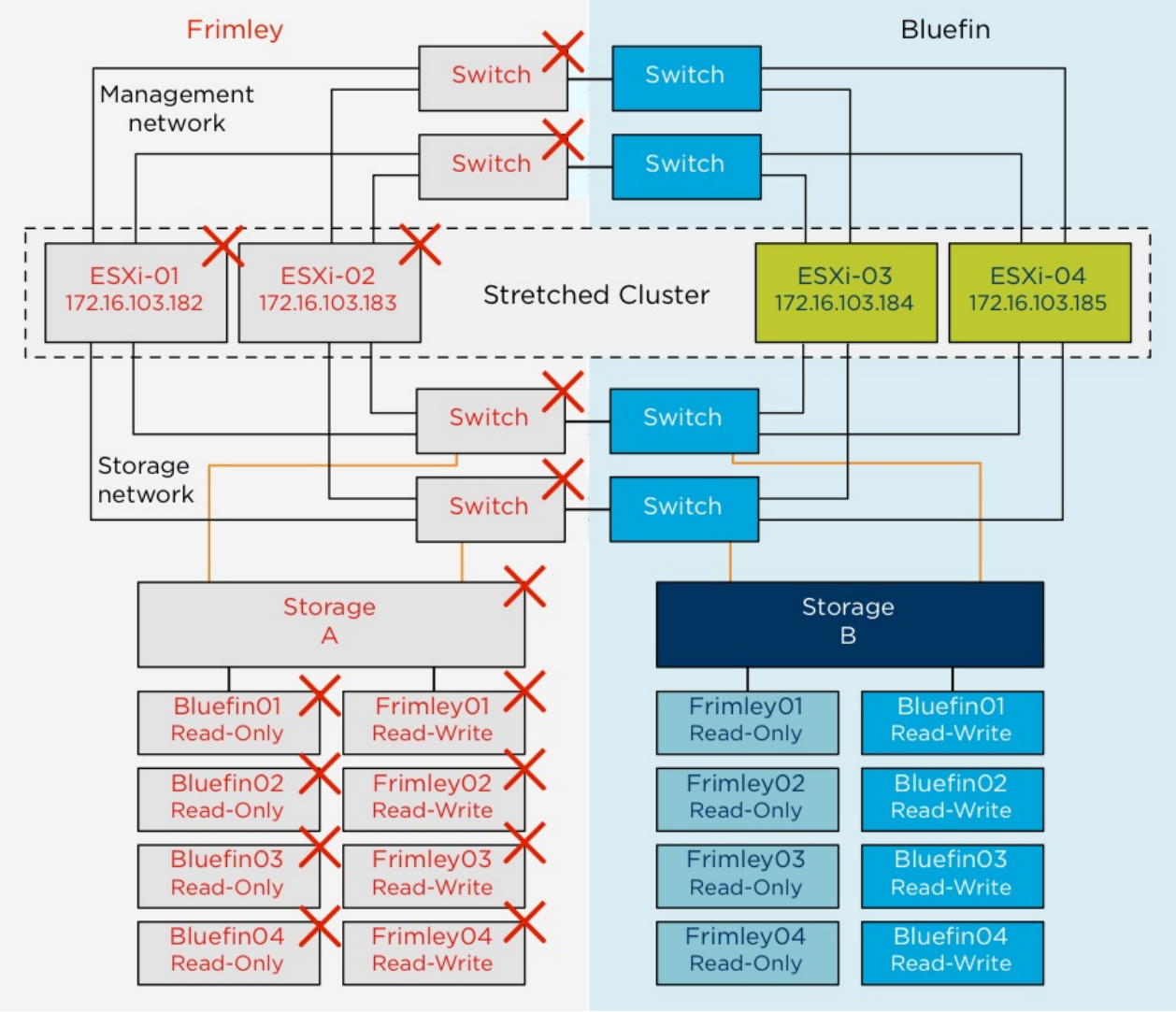
Result: All VMs are successfully restarted in Bluefin data center**.**

Explanation: The vSphere HA master was located in Frimley data center at the time of the full compute failure at that location. After the hosts in Bluefin data center detected that no network heartbeats had been received, an election process was started. Within approximately 20 seconds, a new vSphere HA master was elected from the remaining hosts. Then the new master determined which hosts had failed and which VMs had been impacted by this failure. Because all hosts at the other site had failed and all VMs residing on them had been impacted, vSphere HA initiated the restart of all of these VMs. vSphere HA can initiate 32 concurrent restarts on a single host, providing a low restart latency for most environments. The only sequencing of start order comes from the broad *high*, *medium*, and *low* categories for vSphere HA. This policy must be set on a per-VM basis. These policies were determined to have been adhered to; high-priority VMs started first, followed by medium-priority and low-priority VMs.

As part of the test, the hosts at the Frimley data center were again powered on. As soon as vSphere DRS detected that these hosts were available, a vSphere DRS run was invoked. Because the initial vSphere DRS run corrects only the vSphere DRS affinity rule violations, resource imbalance was not correct until the next full invocation of vSphere DRS. vSphere DRS is invoked by default every 5 minutes or when VMs are powered off or on through the use of the vCenter Web Client.

Loss of Frimley Data Center

In this scenario, a full failure of Frimley data center is simulated.



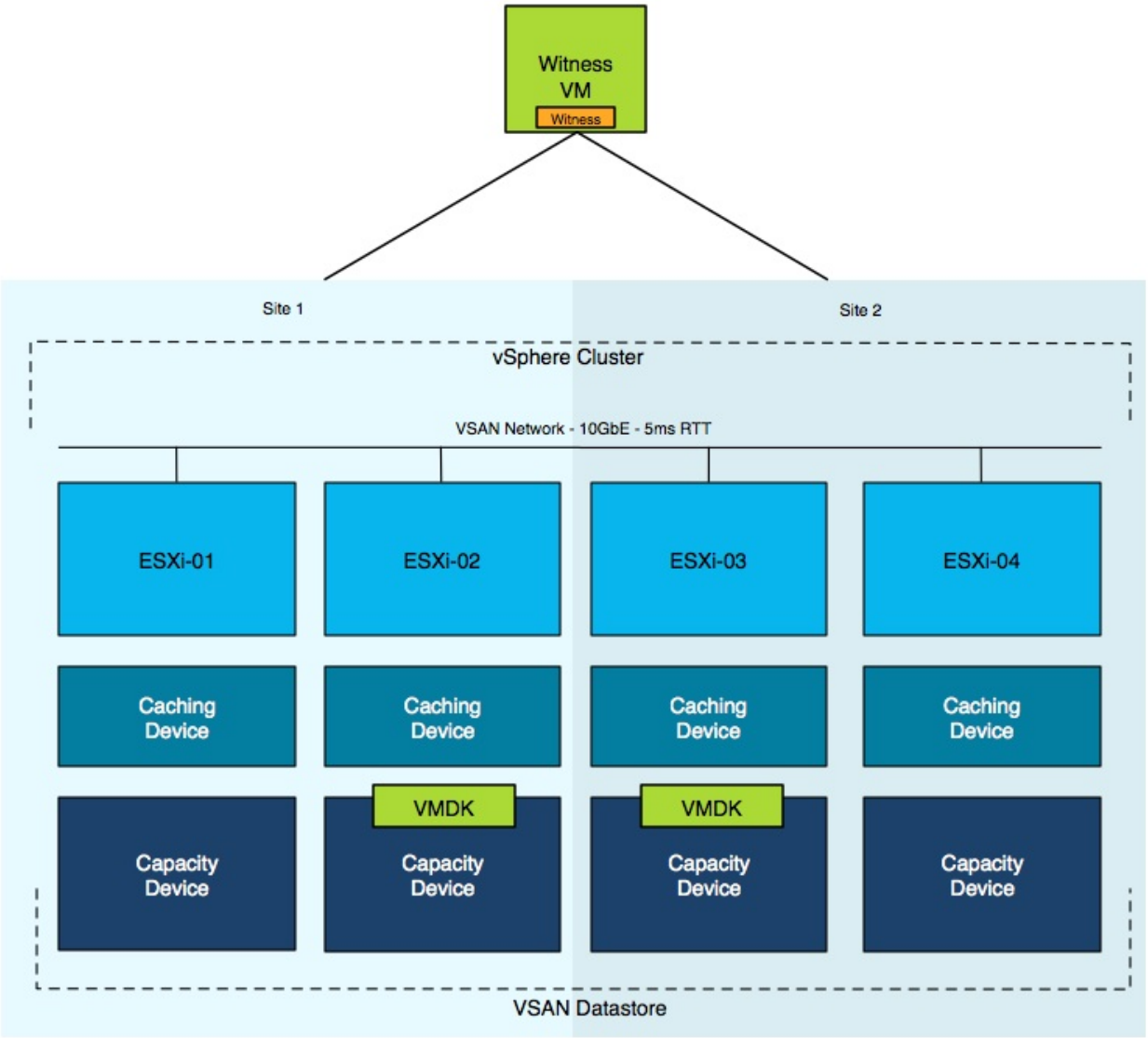
Result: All VMs were successfully restarted in Bluefin data center**.**

Explanation: In this scenario, the hosts in Bluefin data center lost contact with the vSphere HA master and elected a new vSphere HA master. Because the storage system had failed, a **take over** command had to be initiated on the surviving site, again due to the NetApp-specific process. After the **take over** command had been initiated, the new vSphere HA master accessed the per-datastore files that vSphere HA uses to record the set of protected VMs. The vSphere HA master then attempted to restart the VMs that were not running on the surviving hosts in Bluefin data center. In our scenario, all VMs were restarted within 2 minutes after failure and were fully accessible and functional again.

NOTE: *By default, vSphere HA stops attempting to start a VM after 30 minutes. If the storage team does not issue a takeover command within that time frame, the vSphere administrator must manually start up VMs after the storage becomes available.*

Stretched Cluster using VSAN

This question keeps on coming up over and over again lately, Stretched Cluster using Virtual SAN, can I do it? When Virtual SAN was first released the answer to this question was a clear **no**, Virtual SAN did not allow a "traditional" stretched deployment using 2 "data" sites and a third "witness" site. A regular Virtual SAN cluster stretched across 3 sites within campus distance however was possible. With Virtual SAN 6.1 however introduced the "traditional" stretched cluster deployment support.



Everything learned in this publication also applies to a stretched Virtual SAN cluster, with that meaning all HA and DRS best practices. There are a couple of differences though at the time of writing between a vSphere Metro Storage Cluster and a VSAN Stretched Cluster and in this section we will call out these difference. Please note that there is an extensive Virtual SAN Stretched Clustering Guide available written by Cormac Hogan and there is a full Virtual SAN book available written by Cormac Hogan an myself (Duncan Epping). If you want to know more details about Virtual SAN we would like to refer to these two publications.

First thing that needs to be looked at is the network. From a Virtual SAN perspective there are clear requirements:

- 5ms RTT latency max between data sites
- 200ms RTT latency max between data and witness site
- Both L3 and L2 are supported between the data sites
    - 10Gbps bandwidth is recommended, dependent on the number of VMs this could be lower or higher, more guidance will be provided soon around this!
    - Multicast required, which means that if L3 is used, some form of multicast routing is needed.
- L3 is expected between data and the witness sites
    - 100Mbps bandwidth is recommended, dependent on the number of VMs this could be lower or higher, more guidance will be provided soon around this!
    - No multicast required to the witness site.

When it comes to HA and DRS the configuration is pretty straight forward. A couple of things we want to point out as they are configuration details which are easy to forget about. Some are discussed in-depth above, some are settings you actually do not use with VSAN. We will point this out in the list below:

- Make sure to specify additional isolation addresses, one in each site (das.isolationAddress0 – 1).
    - Note that you can specify multiple of these, 2 per site or more if you want
- Disable the default isolation address if it can't be used to validate the state of the environment during a partition (if the gateway isn't available in both sides).
- Disable Datastore heartbeating, without traditional external storage there is no reason to have this.
- Enable HA Admission Control and make sure it is set to 50% for CPU and Memory.
- Keep VMs local by creating "VM\/Host" should rules.

That covers most of it, summarized relatively briefly compared to the excellent document Cormac developed with all details you can wish for. Make sure to read that if you want to know every aspect and angle of a stretched Virtual SAN cluster configuration.

Advanced Settings

There are various types of KB articles and this KB article explains it, but let me summarize it and simplify it a bit to make it easier to digest.

There are various sorts of advanced settings, but for HA three in particular:

- das.* –> Cluster level advanced setting.
- fdm.* –> FDM host level advanced setting
- vpxd.* –> vCenter level advanced setting.

How do you configure these?

Configuring these is typically straight forward, and most of you hopefully know this already, if not, let us go over the steps to help configuring your environment as desired.

Cluster Level
In the Web Client:

- Click “Hosts and Clusters”
- click your cluster object
- click the “Manage” tab
- click “Settings” and “vSphere HA”
- hit the “Edit” button

FDM Host Level

- Open up an SSH session to your host and edit “/etc/opt/vmware/fdm/fdm.cfg”

vCenter Level
In the Web Client:

- Click “vCenter”
- click “vCenter Servers”
- select the appropriate vCenter Server and click the “Manage” tab
- click “Settings” and “Advanced Settings”

In this section we will primarily focus on the ones most commonly used, a full detailed list can be found in KB 2033250. Please note that each bullet details the version which supports this advanced setting.

- das.maskCleanShutdownEnabled - 5.0, 5.1, 5.5

    - Whether the clean shutdown flag will default to false for an inaccessible and poweredOff VM. Enabling this option will trigger VM failover if the VM's home datastore isn't accessible when it dies or is intentionally powered off.
- das.ignoreInsufficientHbDatastore - 5.0, 5.1, 5.5, 6.0
    - Suppress the host config issue that the number of heartbeat datastores is less than das.heartbeatDsPerHost. Default value is “false”. Can be configured as “true” or “false”.
- das.heartbeatDsPerHost - 5.0, 5.1, 5.5, 6.0
    - The number of required heartbeat datastores per host. The default value is 2; value should be between 2 and 5.
- das.failuredetectiontime - 4.1 and prior
    - Number of milliseconds, timeout time, for isolation response action (with a default of 15000 milliseconds). Pre-vSphere 4.0 it was a general best practice to increase the value to 60000 when an active/standby Service Console setup was used. This is no longer needed. For a host with two Service Consoles or a secondary isolation address a failuredetection time of 15000 is recommended.
- das.isolationaddress[x] - 5.0, 5.1, 5.5, 6.0
    - IP address the ESX hosts uses to check on isolation when no heartbeats are received, where [x] = 0 - 9. (see screenshot below for an example) VMware HA will use the default gateway as an isolation address and the provided value as an additional checkpoint. I recommend to add an isolation address when a secondary service console is being used for redundancy purposes.
- das.usedefaultisolationaddress - 5.0, 5.1, 5.5, 6.0
    - Value can be “true” or “false” and needs to be set to false in case the default gateway, which is the default isolation address, should not or cannot be used for this purpose. In other words, if the default gateway is a non-pingable address, set the “das.isolationaddress0” to a pingable address and disable the usage of the default gateway by setting this to “false”.
- das.isolationShutdownTimeout - 5.0, 5.1, 5.5, 6.0
    - Time in seconds to wait for a VM to become powered off after initiating a guest shutdown, before forcing a power off.
- das.allowNetwork[x] - 5.0, 5.1, 5.5
    - Enables the use of port group names to control the networks used for VMware HA, where [x] = 0 – ?. You can set the value to be "Service Console 2" or "Management Network" to use (only) the networks associated with those port group names in the networking configuration. In 5.5 this option is ignored when VSAN is enabled by the way!
- das.bypassNetCompatCheck - 4.1 and prior
    - Disable the “compatible network” check for HA that was introduced with ESX 3.5 Update 2. Disabling this check will enable HA to be configured in a cluster which

contains hosts in different subnets, so-called incompatible networks. Default value is “false”; setting it to “true” disables the check.

- das.ignoreRedundantNetWarning - 5.0, 5.1, 5.5
  - Remove the error icon/message from your vCenter when you don’t have a redundant Service Console connection. Default value is “false”, setting it to “true” will disable the warning. HA must be reconfigured after setting the option.
- das.vmMemoryMinMB - 5.0, 5.1, 5.5
  - The minimum default slot size used for calculating failover capacity. Higher values will reserve more space for failovers. Do not confuse with “das.slotMemInMB”.
- das.slotMemInMB - 5.0, 5.1, 5.5
  - Sets the slot size for memory to the specified value. This advanced setting can be used when a virtual machine with a large memory reservation skews the slot size, as this will typically result in an artificially conservative number of available slots.
- das.vmCpuMinMHz - 5.0, 5.1, 5.5
  - The minimum default slot size used for calculating failover capacity. Higher values will reserve more space for failovers. Do not confuse with “das.slotCpuInMHz”.
- das.slotCpuInMHz - 5.0, 5.1, 5.5
  - Sets the slot size for CPU to the specified value. This advanced setting can be used when a virtual machine with a large CPU reservation skews the slot size, as this will typically result in an artificially conservative number of available slots.
- das.perHostConcurrentFailoversLimit - 5.0, 5.1, 5.5
  - By default, HA will issue up to 32 concurrent VM power-ons per host. This setting controls the maximum number of concurrent restarts on a single host. Setting a larger value will allow more VMs to be restarted concurrently but will also increase the average latency to recover as it adds more stress on the hosts and storage.
- das.config.log.maxFileNum - 5.0, 5.1, 5.5
  - Desired number of log rotations.
- das.config.log.maxFileSize - 5.0, 5.1, 5.5
  - Maximum file size in bytes of the log file.
- das.config.log.directory - 5.0, 5.1, 5.5
  - Full directory path used to store log files.
- das.maxFtVmsPerHost - 5.0, 5.1, 5.5
  - The maximum number of primary and secondary FT virtual machines that can be placed on a single host. The default value is 4.
- das.includeFTcomplianceChecks - 5.0, 5.1, 5.5
  - Controls whether vSphere Fault Tolerance compliance checks should be run as part of the cluster compliance checks. Set this option to false to avoid cluster compliance failures when Fault Tolerance is not being used in a cluster.
- das.iostatsinterval (VM Monitoring) - 5.0, 5.1, 5.5, 6.0
  - The I/O stats interval determines if any disk or network activity has occurred for the

virtual machine. The default value is 120 seconds.

- das.config.fdm.deadIcmpPingInterval - 5.0, 5.1, 5.5
  - Default value is 10. ICPM pings are used to determine whether a slave host is network accessible when the FDM on that host is not connected to the master. This parameter controls the interval (expressed in seconds) between pings.
- das.config.fdm.icmpPingTimeout - 5.0, 5.1, 5.5
  - Default value is 5. Defines the time to wait in seconds for an ICMP ping reply before assuming the host being pinged is not network accessible.
- das.config.fdm.hostTimeout - 5.0, 5.1, 5.5
  - Default is 10. Controls how long a master FDM waits in seconds for a slave FDM to respond to a heartbeat before declaring the slave host not connected and initiating the workflow to determine whether the host is dead, isolated, or partitioned.
- das.config.fdm.stateLogInterval - 5.0, 5.1, 5.5
  - Default is 600. Frequency in seconds to log cluster state.
- das.config.fdm.ft.cleanupTimeout - 5.0, 5.1, 5.5
  - Default is 900. When a vSphere Fault Tolerance VM is powered on by vCenter Server, vCenter Server informs the HA master agent that it is doing so. This option controls how many seconds the HA master agent waits for the power on of the secondary VM to succeed. If the power on takes longer than this time (most likely because vCenter Server has lost contact with the host or has failed), the master agent will attempt to power on the secondary VM.
- das.config.fdm.storageVmotionCleanupTimeout - 5.0, 5.1, 5.
  - Default is 900. When a Storage vMotion is done in a HA enabled cluster using pre 5.0 hosts and the home datastore of the VM is being moved, HA may interpret the completion of the storage vmotion as a failure, and may attempt to restart the source VM. To avoid this issue, the HA master agent waits the specified number of seconds for a storage vmotion to complete. When the storage vmotion completes or the timer expires, the master will assess whether a failure occurred.
- das.config.fdm.policy.unknownStateMonitorPeriod - 5.0, 5.1, 5.5, 6.0
  - Defines the number of seconds the HA master agent waits after it detects that a VM has failed before it attempts to restart the VM.
- das.config.fdm.event.maxMasterEvents - 5.0, 5.1, 5.5
  - Default is 1000. Defines the maximum number of events cached by the master
- das.config.fdm.event.maxSlaveEvents - 5.0, 5.1, 5.5
  - Default is 600. Defines the maximum number of events cached by a slave.

That is a long list of advanced settings indeed, and hopefully no one is planning to try them all out on a single cluster, or even on multiple clusters. Avoid using advanced settings as much as possible as it definitely leads to increased complexity, and often to more down time rather than less.

Summarizing

Hopefully I have succeeded in giving you a better understanding of the internal workings of HA. I hope that this publication has handed you the tools needed to update your vSphere design and ultimately to increase the resiliency and up-time of your environment.

I have tried to simplify some of the concepts to make it easier to understand, still we acknowledge that some concepts are difficult to grasp and the amount of architectural changes that vSphere 5 and new functionality that vSphere 6 have brought can be confusing at times. I hope though that after reading this everyone is confident enough to make the required or recommended changes.

If there are any questions please do not hesitate to reach out me via twitter or my blog, or leave a comment on the online version of this publication. I will do my best to answer your questions.

Changelog

1.0.1 - Minor edits
1.0.2 - Start with VSAN Stretched Cluster in Usecase section
1.0.3 - Start with VVol section in VSAN and VVol specifics section
1.0.4 - Update to VVol section and replaced diagram (figure 15)
1.0.5 - Minor edits
1.0.6 - Removed a duplicate section in "Restarting Virtual Machines"
1.0.7 - Added new best practice around Disk.autoRemoveOnPDL
1.0.8 - Added some details on VVols
1.0.9 - Added new screenshot for Isolation Response and changed "Leave powered on" to "Disabled" as per 6.0 UI
1.1.0 - Added a short section for VSAN Stretched Clusters and partition handling
1.1.1 - Minor edits and rendering using Gitbook 3.0
1.1.2 - Included cover for all versions of the book
1.1.3 - Minor spelling/grammar edits