WHITE PAPER

# PURE STORAGE ACTIVECLUSTER

## WITH SAP HANA 2.0 MDC

PURESTORAGE®

# TABLE OF CONTENTS

## EXECUTIVE SUMMARY

Protecting mission-critical business applications running on the SAP HANA platform, like S/4HANA and BW/4HANA, means securing the highest possible resiliency to ensure business operations do not stop in the event of a disaster – either localized or site-wide. Using SAP HANA in a tailored data center (TDI) deployment gives customers the flexibility to utilize different strategies to maintain business continuity, and, while SAP HANA provides its own capability for backups and disaster recovery using HANA system replication, there is another route that is significantly more cost efficient, easy to scale, and simple to maintain and implement.

The focus of this paper will be to discuss a new Pure Storage® solution for SAP HANA® Storage Replication, Purity ActiveCluster. All Pure Storage FlashArrays are SAP-certified enterprise storage arrays and support both single-host (scale-up) and multi-host (scale-out) SAP HANA systems in TDI deployments.

SAP HANA offers system replication that provides the highest platform availability at the server level. As far as ensuring the high availability of data used by these applications, data needs to be spread across two arrays, often in more than one geographic location and, importantly, must be available in both sites at the same time. To achieve this, some storage arrays offer synchronous replication, which provides the ability to write to the same block storage volume simultaneously while maintaining write-order. This is traditionally called Active-Active or synchronous replication. The most common use case in SAP HANA for Active-Active replication is when there are isolated sites or availability domains (sometimes referred to as “fire cells”) with independent power, cooling, and resources.

Pure Storage has now introduced its own Active-Active replication, named ActiveCluster, in the Purity 5.0.0 release.

This paper overviews configuration and best practices for using the Purity ActiveCluster feature with SAP HANA to achieve the best possible Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for SAP HANA environments. Pure Storage FlashArray combines scale-out architecture and high performance by utilizing industry-best data reduction and deduplication.

### SAP HANA

Traditional databases store data on disk, except for active data sets, which get cached in memory. SAP HANA stands out by being an in-memory database, in which data is stored in RAM until it is offloaded to persistent storage. The persistent storage is then used in the event of a crash, where the database needs to be reloaded again into memory.

Pure Storage ActiveCluster supports both scale-up and scale-out SAP HANA deployments. Depending on the size and type of the SAP solution, customers prefer different types of architecture. OLTP workloads such as SAP S/4HANA and SAP Business Suite are usually scale-ups, while OLAP applications such as SAP BW and SAP BW/4HANA are scale-outs.

## BACKUP AND RECOVERY

Customers will need local or remote copies of their data for protection against disasters and errors. For backup and recovery of SAP HANA using Pure Storage snapshot technology, please see the whitepaper "Pure Storage SAP HANA Backup and Recovery, April 2018". This paper focuses on HANA Storage Replication; while SAP HANA provides built-in capabilities using SAP HANA system replication, there are alternatives, such as HANA Storage Replication, that take advantage of snapshot technology and integrate the database's business continuity solution.

## AUDIENCE

This paper is a guide to using Pure Storage ActiveCluster in an SAP HANA environment. It is intended for storage, system, network, SAP HANA Basis administrator, or other administrators who plan to implement Pure Storage FlashArray ActiveCluster with SAP HANA. A working knowledge of SAP HANA, Linux, server, storage, and networks is assumed but is not a prerequisite to read this document. The paper guides Pure Storage customers interested in utilizing HANA Storage Replication in their environment. Readers should be knowledgable with SAP HANA and Pure Storage products.

As always, for specific questions or requests for assistance, please reach out to Pure Storage support at any time at support@purestorage.com.

## IMPORTANT NOTES

Before setting up ActiveCluster and SAP HANA, it is important to read the ActiveCluster specific documentation and SAP HANA notes on SLES 12:

- https://support.purestorage.com/FlashArray/PurityFA/Protect/Replication/ActiveCluster_Requirements_and_Best_Practices
- https://support.purestorage.com/FlashArray/PurityFA/Protect/Replication/ActiveCluster_Solution_Overview

> Ensure that the above stated best practices have been followed. This document assumes that the aforementioned documentation has been read prior to reading this document itself.
>
> SUSE Linux 12 SP01 was crashing when half the paths were removed during the active cluster storage failover. It is highly recommended to be on SUSE Linux 12 SP02.

**SOLUTION OVERVIEW & INTRODUCTION**

SAP HANA ActiveCluster (storage replication) is a highly recommended disaster recovery solution based on the replication of data, log, and shared volumes to remote storage. In this method, all the data that is written to the persistence layer (data and log volumes) by the primary HANA system is replicated to a (usually) secondary cross-site location. Because the secondary host is in a remote location, this solution requires a reliable, high bandwidth and low latency connection between the primary site and the secondary site.

## SAP HANA ActiveCluster Scenarios

The SAP HANA platform is tested on the following scenarios in the document.

1. **Local High Availability** – In this scenario, SAP HANA is deployed on a single host with ActiveCluster-enabled storage. In case of storage failure on the primary side, the SAP HANA application will be running uninterruptedly on the secondary storage. This is a zero RTO and zero RPO scenario.

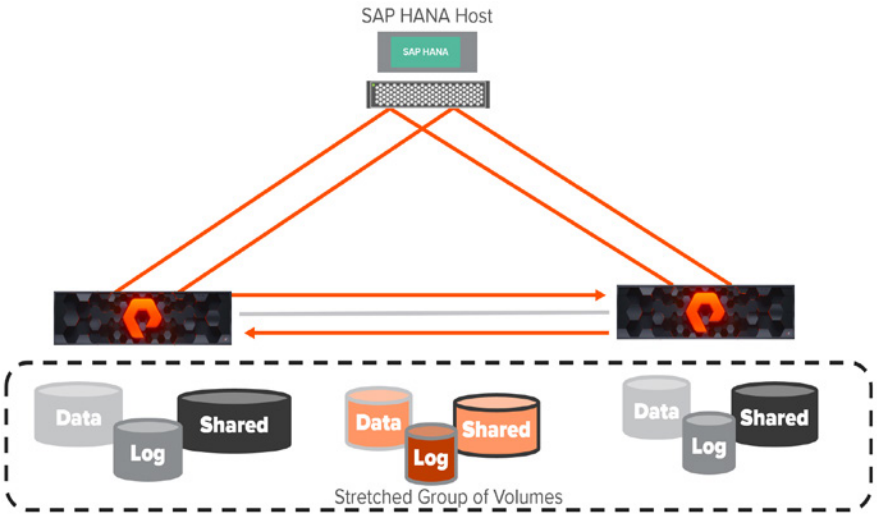



FIGURE 1. Local high availability scenario

2. **Remote Disaster Recovery** – In this scenario, SAP HANA is deployed on two different hosts with ActiveCluster-enabled storage. In the event of a failure on the primary site that includes primary host and storage, the SAP HANA application can be recovered on the secondary site. This is not a zero RTO scenario.

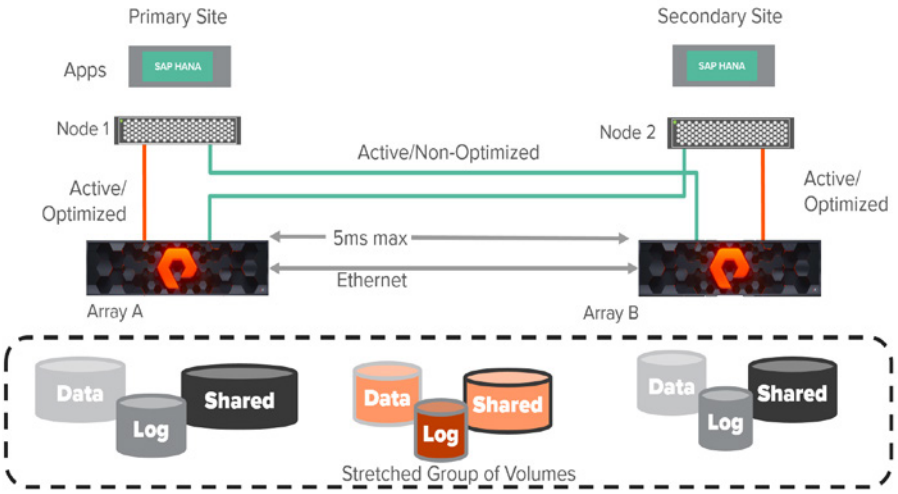



FIGURE 2. Remote disaster recovery scenario

## ActiveCluster

Pure Storage Purity ActiveCluster is a fully symmetric active/active bidirectional replication solution that provides synchronous replication for RPO zero and automatic transparent failover for RTO zero. ActiveCluster spans multiple sites, enabling clustered arrays and clustered hosts to be used to deploy flexible active/active datacenter configurations.

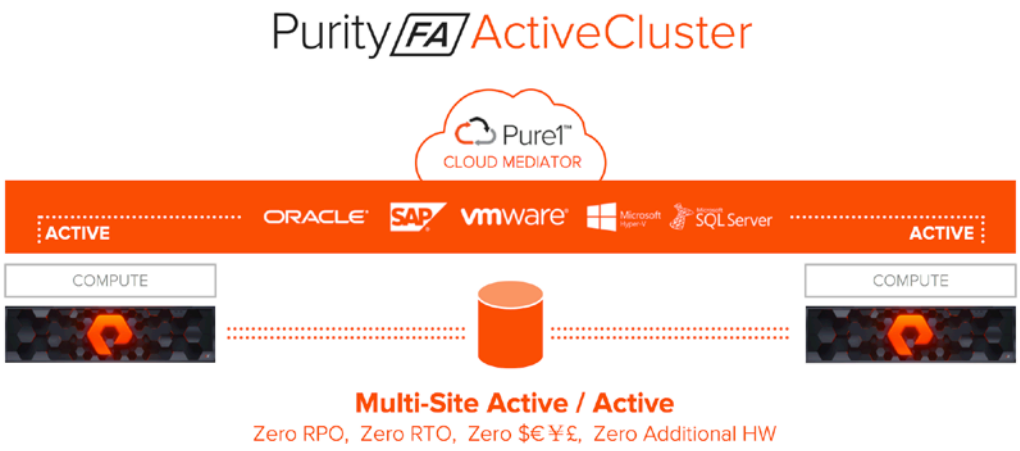



FIGURE 3. ActiveCluster: active/active bidirectional replication solution

- **Synchronous Replication** – Writes are synchronized between arrays and protected in non-volatile RAM (NVRAM) on both arrays before being acknowledged to the host.
- **Symmetric Active/Active** – Read and write to the same volumes at either side of the mirror, with optional host-to-array site awareness.

- **Transparent Failover** – Automatic non-disruptive failover between synchronously replicating arrays and sites with automatic resynchronization and recovery.
- **Async Replication Integration** – Uses async for baseline copies and resynchronizing. Convert async relationships to sync without resending data. Async data to a 3rd site for DR.
- **No Bolt-ons & No Licenses** – No additional hardware required, no costly software licenses required, just upgrade the Purity Operating Environment and go active/active!
- **Simple Management** – Perform data management operations from either side of the mirror, provision storage, connect hosts, create snapshots, create clones.
- **Integrated Pure1® Cloud Mediator** – Automatically configured passive mediator that allows transparent failover and prevents split-brain, without the need to deploy and manage another component.

## COMPONENTS

Purity ActiveCluster is composed of three core components: the Pure1 Mediator, active/active clustered array pairs, and stretched storage containers.

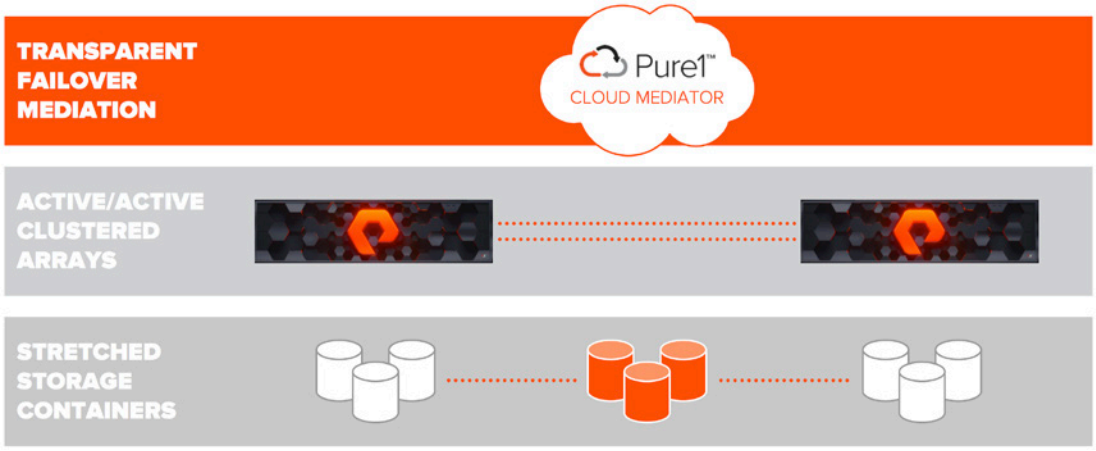



FIGURE 4. ActiveCluster core components

**The Pure1 Cloud Mediator** – A required component of the solution that is used to determine which array will continue data services should an outage occur in the environment.

**Active/Active Clustered FlashArrays** – Utilize synchronous replication to maintain a copy of data on each array and present those as one consistent copy to hosts that are attached to either, or both, arrays.

**Stretched Storage Containers** – Management containers that collect storage objects such as volumes into groups that are stretched between two arrays.

## ADMINISTRATION

ActiveCluster introduces a new management object: Pods. A pod is a stretched storage container that defines both (1) a set of objects that are synchronously replicated together, and (2) the arrays they are replicated between. An array can support multiple pods. Pods can exist on just one array or on two arrays simultaneously with synchronous replication. Pods that are synchronously replicated between two arrays are said to be stretched between arrays.

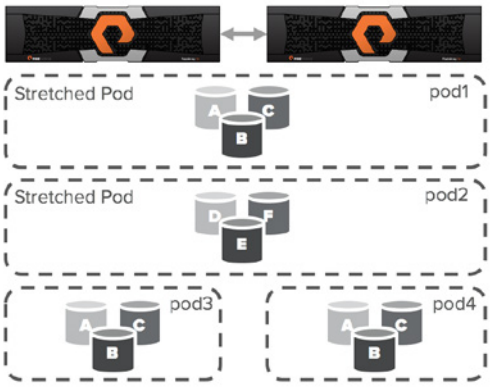



FIGURE 5. Examples of ActiveCluster pods

Pods can contain volumes, protection groups (for snapshot scheduling and asynchronous replication), and other configuration information such as which volumes are connected to which hosts. The pod acts as a consistency group, ensuring that multiple volumes within the same pod remain write-order consistent.

Pods also provide volume namespaces: that is, different volumes may have the same volume name if they are in different pods. In the image above, the volumes in pod3 and **pod4** are different volumes than those in **pod1**, a stretched active/active pod. This allows migration of workloads between arrays or consolidation of workloads from multiple arrays to one, without volume name conflicts.

## MEDIATOR

Transparent failover between arrays in ActiveCluster is automatic and requires no intervention from the storage administrator. Failover occurs within standard host I/O timeouts, similar to the way failover occurs between two controllers in one array during non-disruptive hardware or software upgrades.

ActiveCluster is designed to provide maximum availability across symmetric active/active storage arrays while preventing a split-brain condition from occurring – split brain being the case where two arrays might serve I/O to the same volume without keeping the data in sync between the two arrays.

Any active/active synchronous replication solution designed to provide continuous availability across two different sites requires a component referred to as a witness, or voter, to mediate failovers while preventing split brain. ActiveCluster includes a simple to use, lightweight, and automatic way for applications to failover transparently, or simply move, between sites in the event of a failure without user intervention: the Pure1 Cloud Mediator.

The Pure1 Cloud Mediator is responsible for ensuring that only one array is allowed to stay active for each pod when there is a loss of communication between the arrays.

In the event that the arrays can no longer communicate with each other over the replication interconnect, both arrays will pause I/O and reach out to the mediator to determine which array can stay active for each sync-replicated pod. The first array to reach the mediator is allowed to keep its synchronously replicated pods online. The second array to reach the mediator must stop servicing I/O to its synchronously replicated volumes, in order to prevent split brain. The entire operation occurs within standard host I/O timeouts to ensure that applications experience no more than a pause and resume of I/O.

### THE PURE1 CLOUD MEDIATOR

A failover mediator must be located in a 3rd site that is in a separate failure domain from either site where the arrays are located. Each array site must have independent network connectivity to the mediator such that a single network outage does not prevent both arrays from accessing the mediator. A mediator should also provide a very lightweight and easy to administer component of the solution. The Pure Storage solution provides this automatically by utilizing an integrated cloud-based mediator. The Pure1 Cloud Mediator provides two main functions:

1. **Prevent a split brain condition from occurring**, where both arrays are independently allowing access to data without synchronization between arrays.
2. **Determine which array will continue to service IO** to synchronously replicated volumes in the event of an array failure, replication link outage, or site outage.

The Pure1 Cloud Mediator has the following advantages over a typical non-Pure, heavy-handed voter or witness component:

- **SaaS Operational Benefits** – As with any SaaS solution the operational maintenance complexity is removed: nothing to install onsite, no hardware or software to maintain, nothing to configure and support for HA, no security patch updates, etc.
- **Automatically a 3rd Site** – The Pure1 Cloud Mediator is inherently in a separate failure domain from either of the two arrays.
- **Automatic Configuration** – Arrays configured for synchronous replication will automatically connect to and use the Pure1 Cloud Mediator.
- **No Misconfiguration** – With automatic and default configuration there is no risk that the mediator might be incorrectly configured.
- **No Human Intervention** – A significant number of issues in non-Pure active/active synchronous replication solutions, particularly those related to accidental split brain, are related to human error. Pure's automated non-human mediator eliminates operator error from the equation.
- **Passive Mediation** – Continuous access to the mediator is not required for normal operations: the arrays will maintain a heartbeat with the mediator. However, if the arrays lose connection to the mediator, they will continue to synchronously replicate and serve data as long as the replication link is active.

### ON-PREMISES FAILOVER MEDIATOR

Failover mediation for ActiveCluster can also be provided using an on-premises mediator distributed as an OVF file and deployed as a VM. Failover behaviors are exactly the same as described above. The on-premises mediator simply replaces the role of the Pure1 Cloud Mediator during failover events.

The on-premises mediator has the following basic requirements:

- The on-premises mediator can only be deployed as a VM on virtualized hardware. It is not installable as a stand-alone application.
- High Availability for the mediator must be provided by the hosts on which the mediator is deployed. For example, using VMware HA, or Microsoft Hyper-V HA Clustering.
- Storage for the mediator must not allow the configuration of the mediator to be rolled back to previous versions. This applies to situations such as storage snapshot restores, or cases where the mediator might be stored on mirrored storage.
- The arrays must be configured to use the on-premises mediator rather than the Pure1 Cloud Mediator.
- The mediator must be deployed in a third site, in a separate failure domain that will not be affected by any failures in either of the sites where the arrays are installed.
- Both array sites must have independent network connections to the mediator such that a failure of one network connection does not prevent both arrays from accessing the mediator.

## Uniform Host Access

Hosts can be configured to see just the array that is local to it, or to both arrays. The former option is called non-uniform host access; the latter is referred to as uniform host access.

### UNIFORM ACCESS

A uniform host access model can be used in environments where there is host-to-array connectivity of either FC or Ethernet (for iSCSI), and array-to-array Ethernet connectivity, between the two sites. When deployed in this way, a host has access to the same volume through both the local array and the remote array. The solution supports connecting arrays with up to 5ms of round trip time (RTT) latency between the arrays.

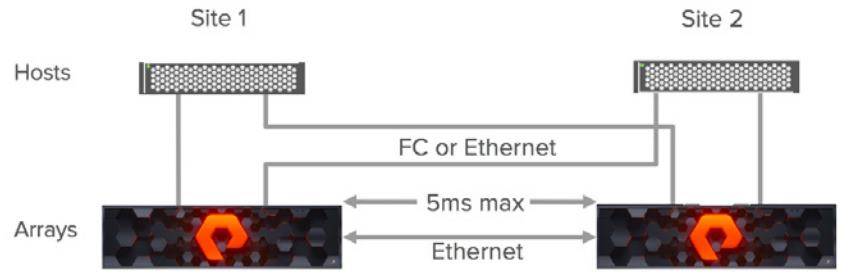



FIGURE 6. Logical paths between hosts and arrays

The image above represents the logical paths that exist between the hosts and arrays, and the replication connection between the two arrays, in a uniform access model. Because a uniform host access model allows all hosts, regardless of site location, to access both arrays, there will be paths with different latency characteristics. Paths from hosts to the local array will have lower latency; paths from each local host to the remote array will have higher latency.

For best performance in active/active synchronous replication environments, hosts should be prevented from using paths that access the remote array unless necessary.

For example, in the image below, if the application SVC-B were to perform a write to volume A over the host side connection to array A, that write would incur 2x the latency of the inter-site link, 1x for each traverse of the network. The write would experience 5ms of latency for the trip from host B to array A and experience another 5ms of latency while array A synchronously sends the write back to array B.

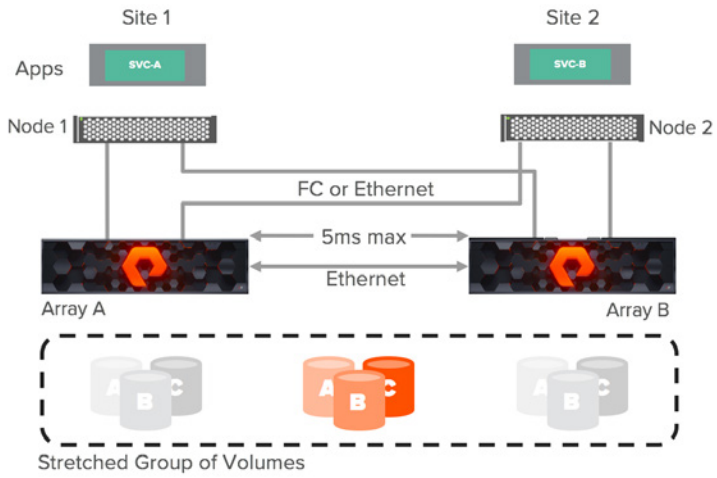



FIGURE 7. Non-optimized network

With ActiveCluster there are no such management headaches. ActiveCluster does make use of ALUA to expose paths to local hosts as active/optimized paths and to expose paths to remote hosts as active/non-optimized. However, there are two advantages in the ActiveCluster implementation.

1. In ActiveCluster, volumes in stretched pods are read/write on both arrays. There is no such thing as a passive volume that cannot service both reads and writes.
2. The optimized path is defined on a per host-to-volume connection basis using a preferred-array option; this ensures that, regardless of what host a VM or application is running on, it will have a local-optimized path to that volume.

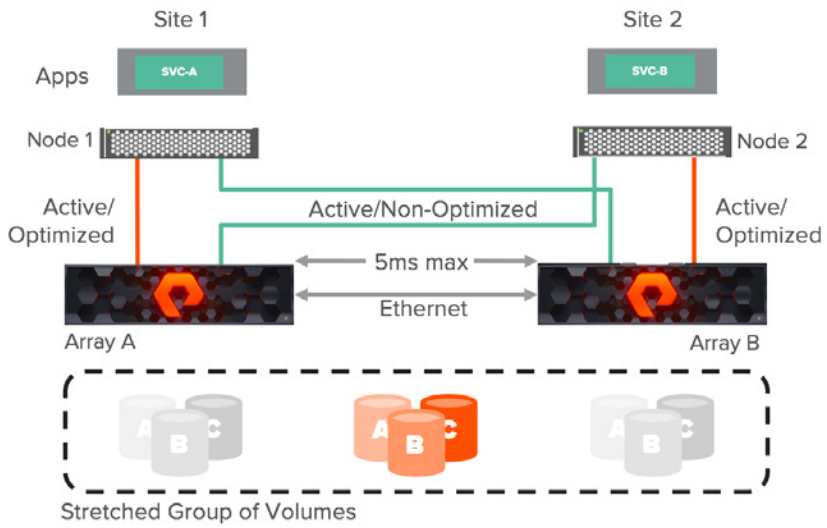



FIGURE 8. Active/Active optimized network

In cases of a datacenter or campus HA setup in which the hosts and arrays are close to each other, hosts performing I/O across all paths will not result in performance degradation – rather, they can improve performance.

**CONFIGURING ACTIVECLUSTER**

A major benefit of using an ActiveCluster stretched storage solution is how simple it is to setup.

Before moving forward, ensure environment configuration requirements are followed as dictated in this KB article: https://support.purestorage.com/FlashArray/PurityFA/Protect/Replication/ActiveCluster_Requirements_and_Best_Practices

## ActiveCluster Glossary of Terms

The following terms have been introduced for ActiveCluster and will be used repeatedly in this document:

- **Pod** – A pod is a namespace and a consistency group. Synchronous replication can be activated on a pod, which makes all volumes in that pod present on both FlashArrays in the pod.
- **Stretching** – Stretching a pod is the act of adding a second FlashArray to a pod. When stretched to another array, the volume data will begin to synchronize, and when complete all volumes in the pod will be available on both FlashArrays.
- **Unstretching** – Unstretching a pod is the act of removing a FlashArray from a pod. This can be done from either FlashArray. When removed, the volumes and the pod itself are no longer available on the FlashArray from which they were removed.

- **Restretching** – When a pod is unstretched, the other array (the array unstretched from) will keep a copy of the pod in the trash can for 24 hours. This allows the pod to be quickly re-stretched without having to resend all data, if restretched within 24 hours.

## Creating a Synchronous Connection

The first step to enable ActiveCluster is to create a synchronous connection with another FlashArray. It does not matter which FlashArray is used to create the connection – either one is fine.

Login to the FlashArray Web Interface and click on the **Storage** section. Click either on the **plus sign** or on the **vertical ellipsis**, and choose **Connect Array**.

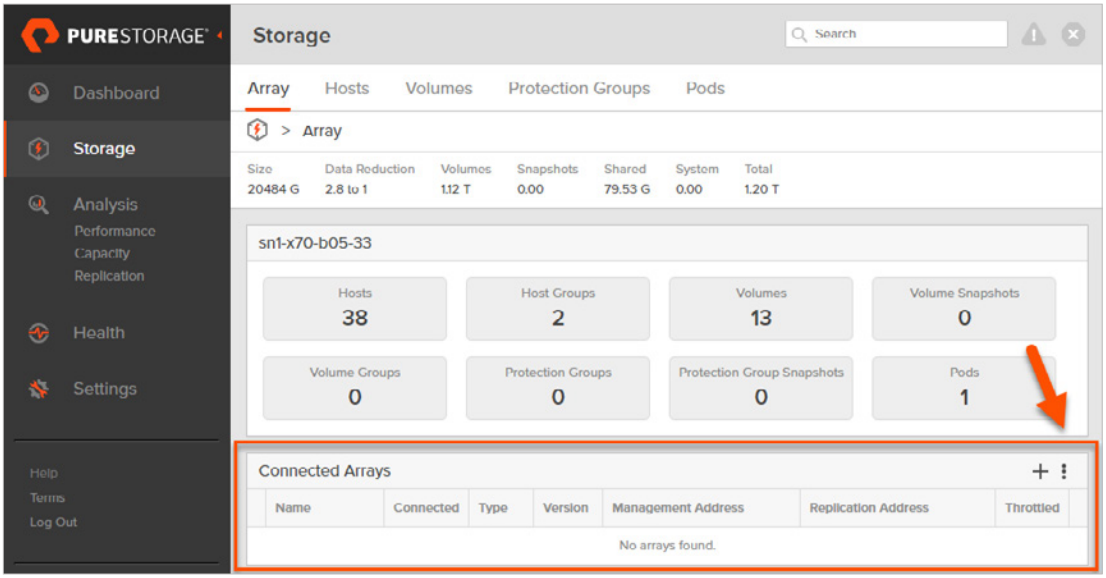



FIGURE 9. FlashArray GUI, Storage panel

The window that pops-up requires three pieces of information:

1. **Management Address** – This is the **virtual IP address** or **FQDN** of the remote FlashArray.
2. **Connection Type** – Choose **Sync Replication** for ActiveCluster.
3. **Connection Key** – This is an **API token** that can be retrieved from the remote FlashArray.

To obtain the connection key, login to the remote FlashArray Web Interface, click on the **Storage** section, and then click on the **vertical ellipsis** and choose **Get Connection Key**.

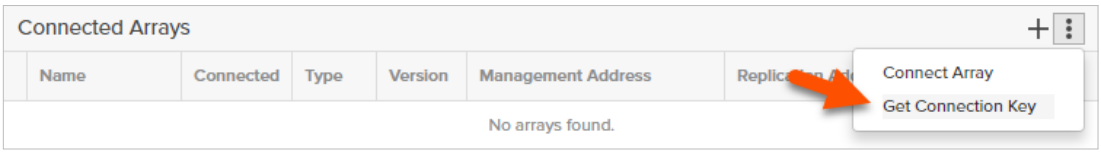



FIGURE 10. Getting a Connection Key

Copy the key to the clipboard using the **Copy** button.

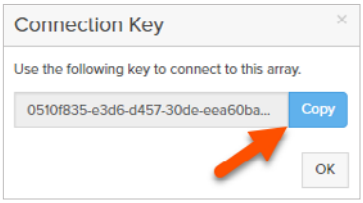



FIGURE 11. Connection key Copy button

Go back to the original FlashArray Web Interface and paste in the key.

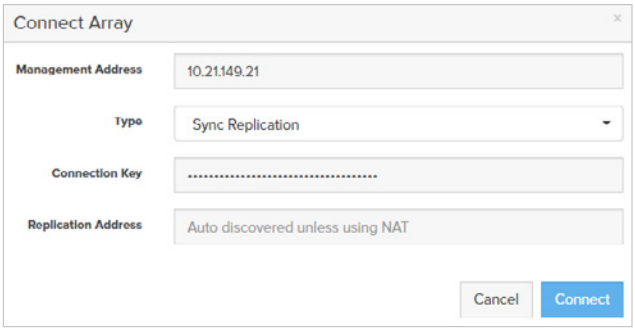



FIGURE 12. Connect Array panel

The replication address field may be left blank and Purity will automatically discover all the replication port addresses. If the target addresses are via Network Address Translation (NAT) then it is necessary to enter the replication port NAT addresses.

When complete, click **Connect**.

If everything is valid, the connection will appear in the **Connected Arrays** panel.

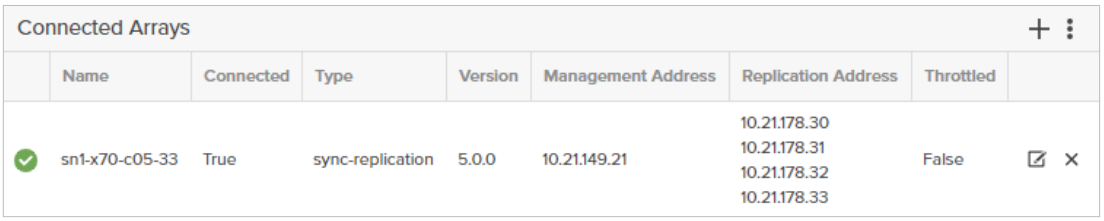



FIGURE 13. Connected Arrays panel

If the connection fails, verify network connectivity and IP information and/or contact Pure Storage Support.

## Creating a Pod

The next step to enable ActiveCluster is to create a consistency group. With ActiveCluster, this is a called a “pod”.

A pod is both a consistency group and a namespace – in effect creating a grouping for related objects involved in ActiveCluster replication. One or more pods can be created. Pods are stretched, unstretched, and failed over together.

Therefore, the basic idea for one or more pods is simply to put related volumes in the same pod. If the host includes related applications that should remain in the same data center or have consistency with one another, put them in the same pod. Otherwise, put them in the same pod for simplicity, or into different pods if the applications have different requirements.

To create a pod, login to the FlashArray GUI and click on **Storage**, then **Pods**, and then click on the **plus sign**.

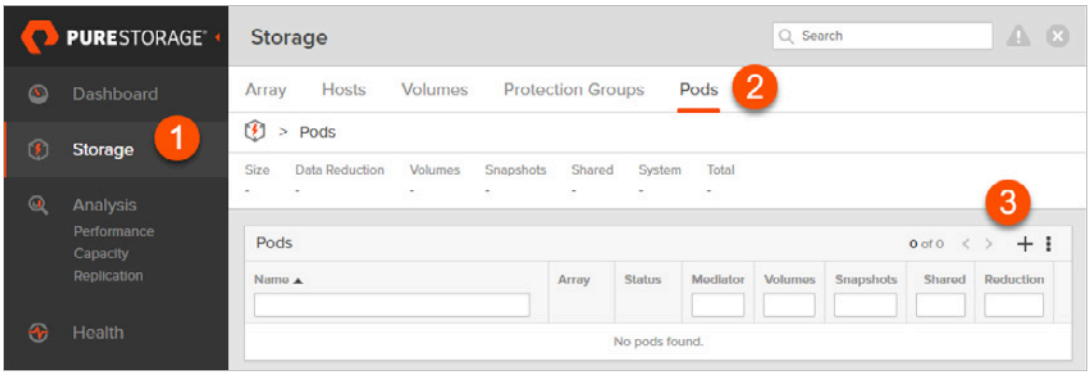



FIGURE 14. FlashArray GUI, Pods panel

Next, enter a name for the pod. This can be letters, numbers, or dashes (must start with a letter or number). Then click **Create**.


FIGURE 15. Create Pod panel

The pod will then appear in the **Pods** panel.

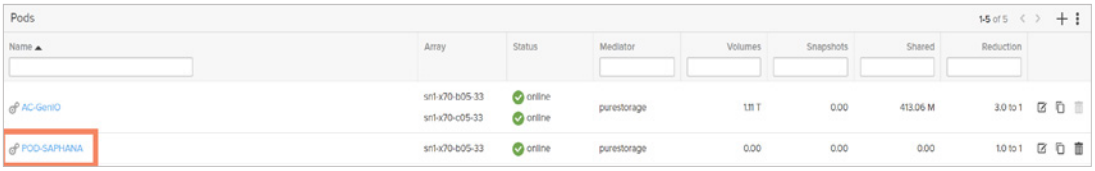



FIGURE 16. Pods panel

To further configure the pod, click on the pod name.

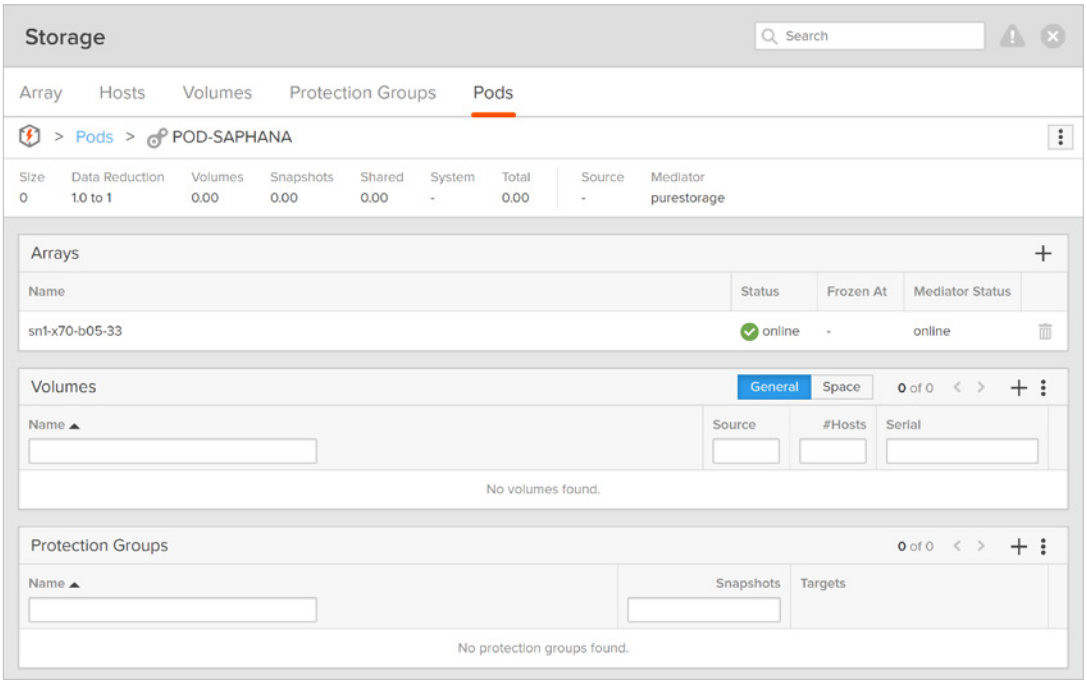



FIGURE 17. Configuring a pod

The default configuration for ActiveCluster is to use the Cloud Mediator – no configuration is required other than to ensure the management network from the FlashArray is redundant (i.e., uses two NICs per controller) and to have IP access to the mediator. Refer to the networking section in the below KB for more details: https://support.purestorage.com/FlashArray/PurityFA/Protect/Replication/ActiveCluster_Requirements_and_Best_Practices

The mediator in use can be seen in the overview panel under the **Mediator** heading. If the mediator is listed as "purestorage", the Cloud Mediator is in use. (For configuring an on-premises mediator in the case of a dark site configuration, refer to ActiveCluster documentation.)

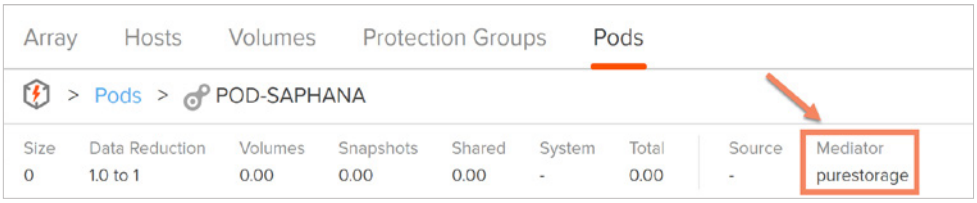



FIGURE 18. Cloud Mediator in use

Pod configuration is complete.

## Adding Volumes to a Pod

The next step is to add to the pod any pre-existing volumes on which you would like to enable ActiveCluster. Once a pod has been enabled for replication, pre-existing volumes cannot be moved into the pod; only new volumes can be created in the pod.

To add a volume to a pod, go to the **Storage** screen in the FlashArray Web Interface, click on **Volumes**, and then click on the **name** of the volume you would like to add to the pod. To find the volume quickly you can search by name.

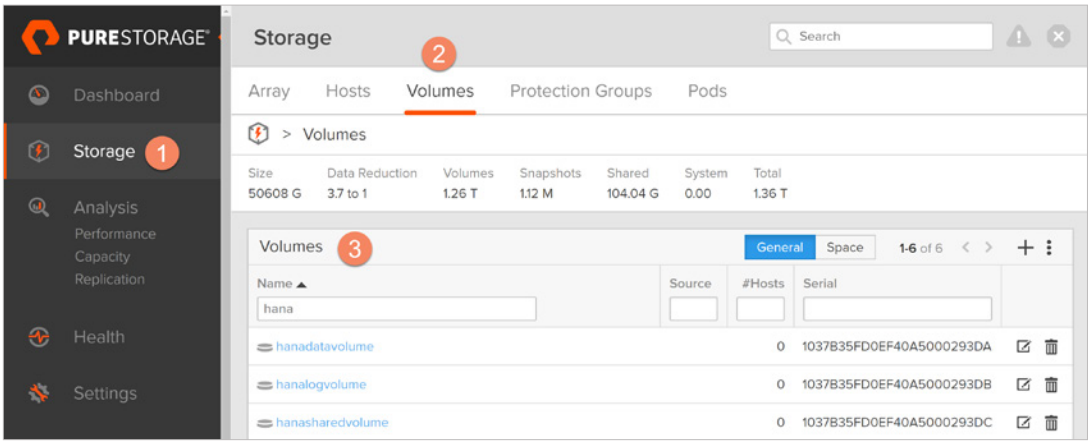



FIGURE 19. Adding a voluime to a Pod

When the volume screen loads, click on the **vertical ellipsis** in the upper right-hand corner and choose **Move**.

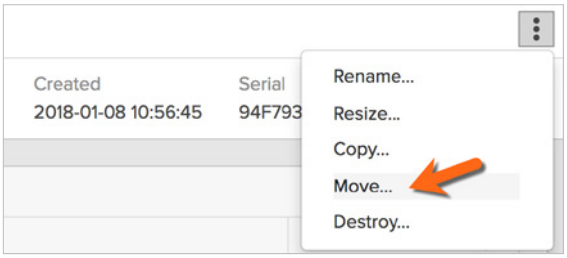



FIGURE 20. Moving a volume to a Pod

To choose the pod, click on the **Container** box and choose the pod name.

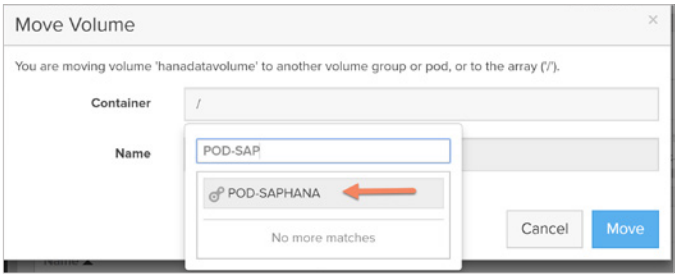



FIGURE 21. Choosing a Pod name

Note that as of Purity 5.0.0, the following limitations apply to moving a volume into a pod:

- A volume in a volume group cannot be added to a pod. It must be removed from the volume group first.
- A volume in a stretched pod already cannot be added to a pod. In this case, you must first unstretch the pod.
- If the target pod is stretched, you must first unstretch the pod to add an existing volume into it.

Some of these restrictions may relax in future Purity versions.

Choose a valid target pod and click **Move**.

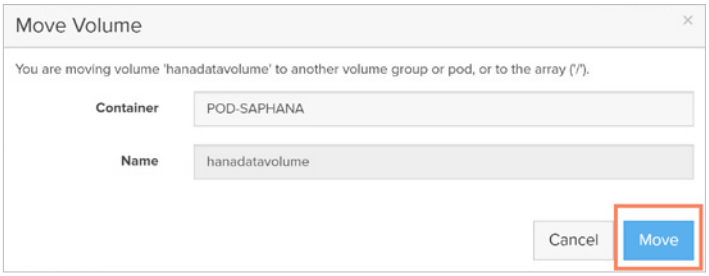



FIGURE 22. Choosing a valid target

This will move the volume into the pod and rename the volume to have a prefix consisting of the pod name and two colons. The volume name will then be in the format of **<podname>::<volumename>**.

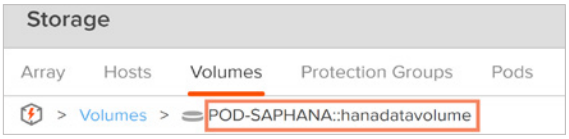



FIGURE 23. Displaying the Pod

The pod will list the volume under its **Volumes** panel.

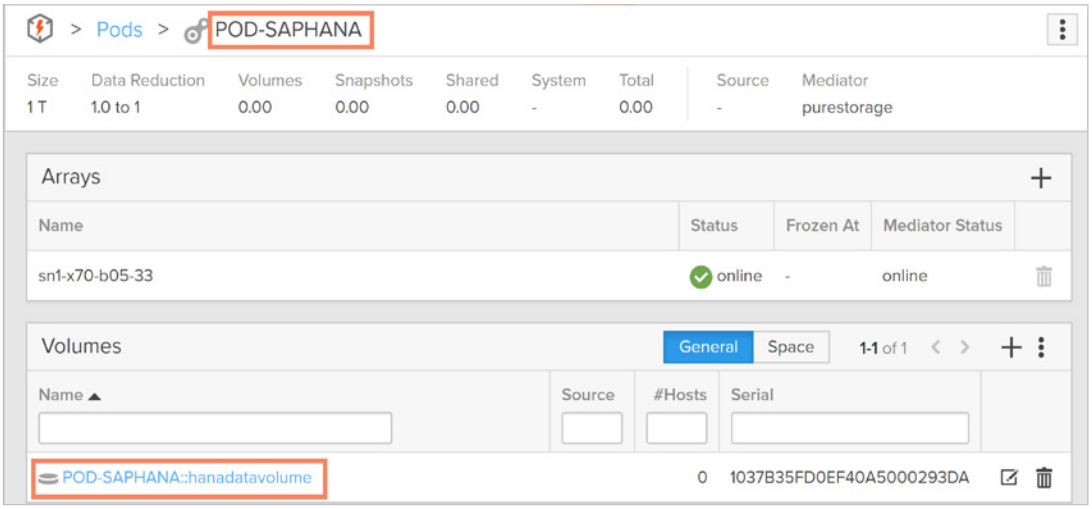



FIGURE 24. Pod Volumes panel

Alternatively, you can click the **Pods** from the **Storage** screen and select the pod to which the volumes have to be moved.

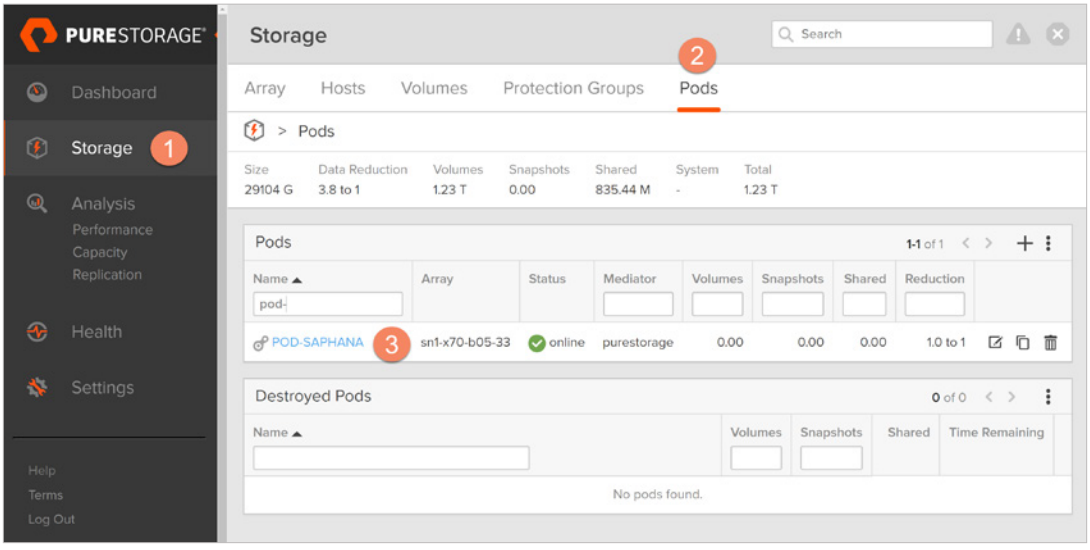



FIGURE 25. Pods listed in the Storage panel

When the Pods screen is loaded for the selected pod, click the **vertical ellipsis** on the right-hand side of the **Volumes** section and choose **Move In** option.

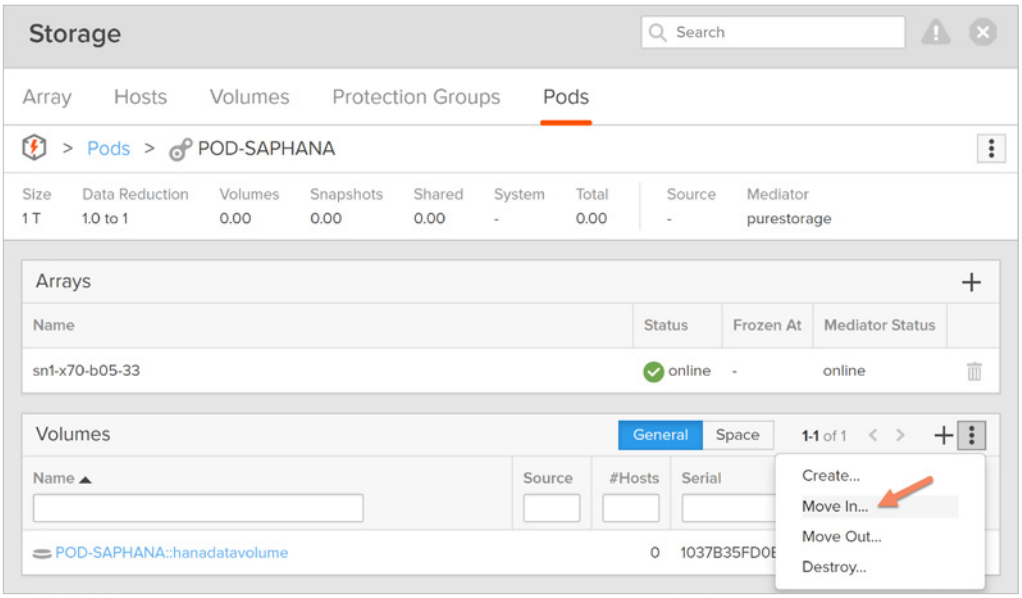



FIGURE 26. Choosing the Move In option

The **Move In** option will bring up a window with a list of volumes. Select the volume(s) that you want to move into the pod and click **Move**.

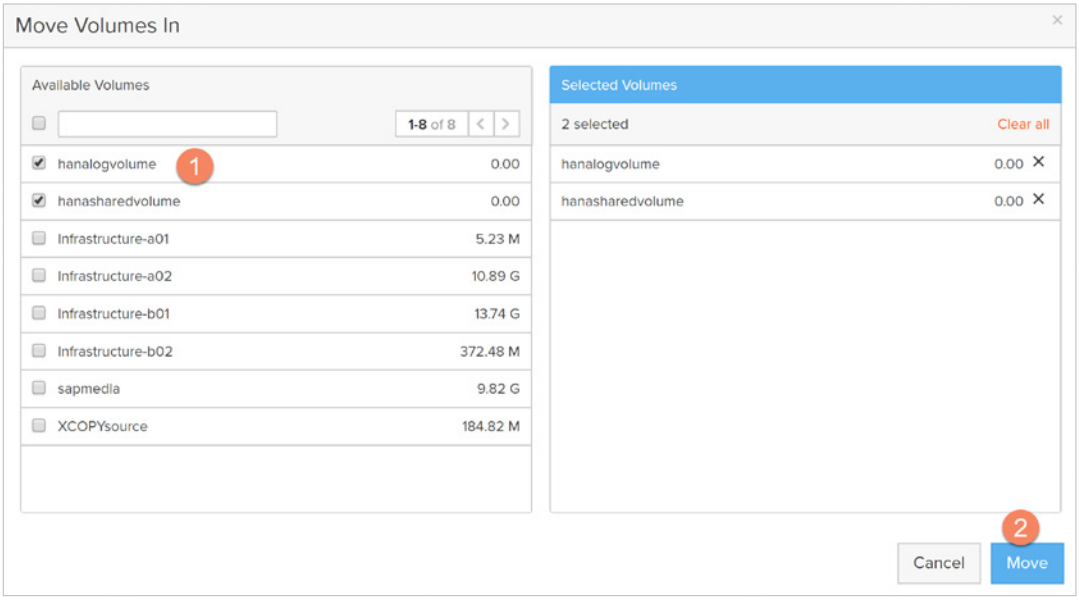



FIGURE 27. Volumes to be moved

## Creating a New Volume in a Pod

Users can also create new volumes directly in a pod. Click on **Storage**, then **Pods**, then choose your pod. Under the **Volumes** panel, click the **plus sign** or click the vertical ellipsis and choose **Create** to create a new volume.


FIGURE 28. Creating a new volume

In the creation window, enter a valid name and a size and click **Create**. This can be done whether or not the pod is actively replicating.

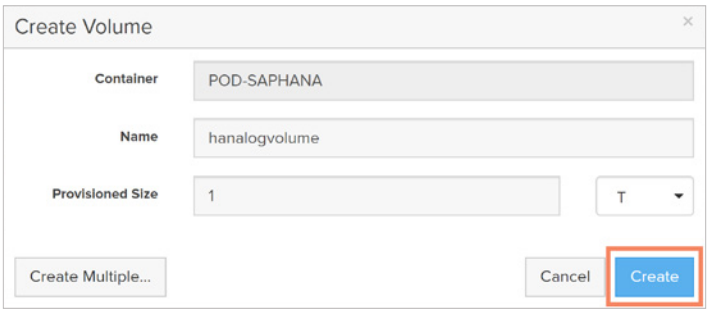



FIGURE 29. Entering a name in the Creation window

## Stretching a Pod

The next step is adding a FlashArray target to the pod. This is called “stretching” the pod, because it automatically makes the pod and its content available on the second array. Only a stretched pod can offer synchronous, or active-active, replication.

> Please note that once a pod has been “stretched”, pre-existing volumes cannot be added to it until it is “un-stretched”. Alternatively, once a pod has been stretched, only new volumes can be created in the pod. Therefore, if you would like to add existing volumes to the pod, follow the instructions in the section **Adding Volumes to a Pod** on page 17, then stretch the pod.

To stretch a pod, add a second array to the pod. Do this by clicking on the **plus sign** in the **Arrays** panel.

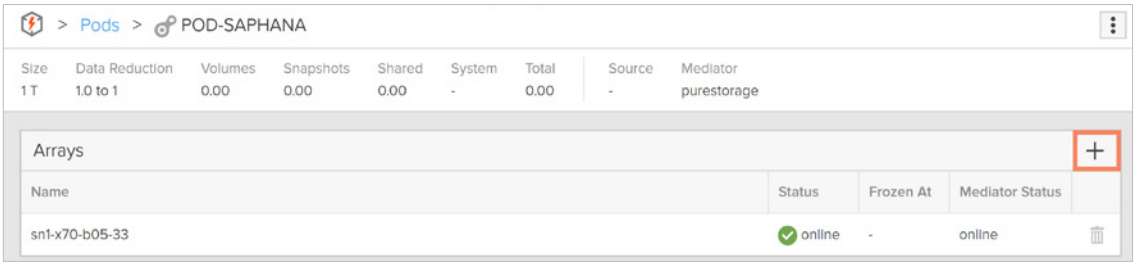



FIGURE 30. Stretching a Pod

Choose a target FlashArray and click **Add**. This will only show the arrays that are connected in synchronous replication mode.

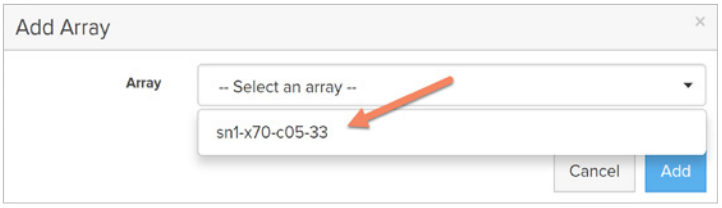



FIGURE 31. Adding a target FlashArray

The two arrays will immediately start synchronizing data between them.

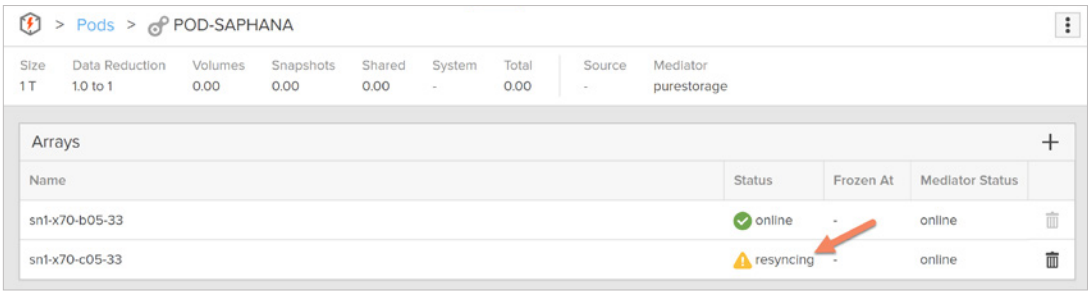



FIGURE 32. FlashArray synchronization

Active/Active storage is not available until the synchronization completes, which will be shown when the resyncing status ends and both arrays are online.


FIGURE 33. FlashArrays synchronized

On the remote FlashArray, the pod and volumes will now exist in identical fashion and will be available for provisioning to a host or hosts on either FlashArray. Initially it will show the number of host connections from the source array.

As seen below, the data volume, log volume, and shared volume will be present on the other array.

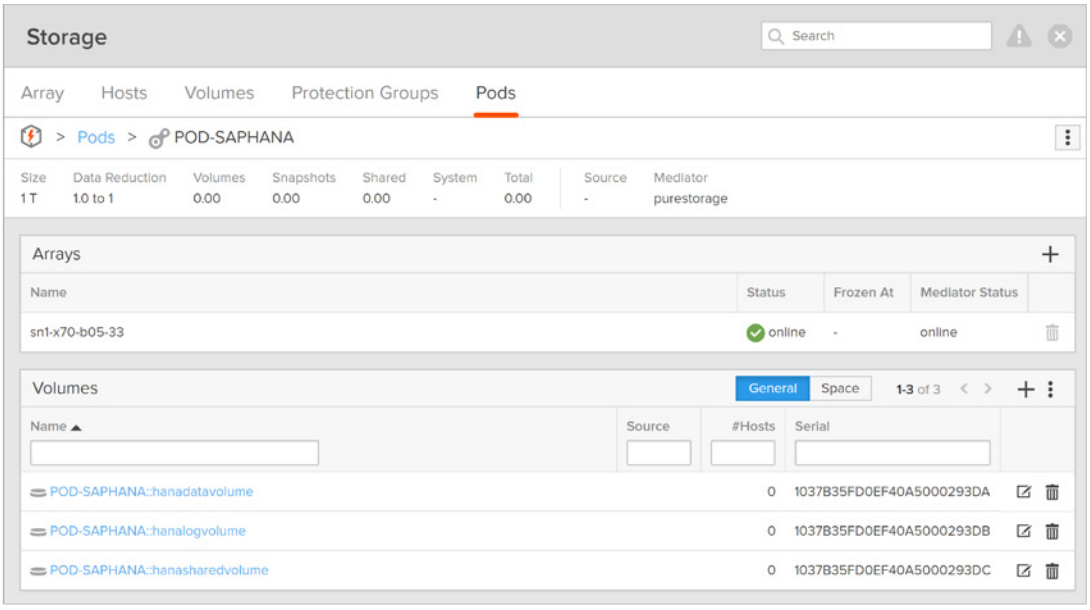



FIGURE 34. Data volume, log volume, and shared volume present on other array

## Un-Stretching a Pod

Once ActiveCluster has been enabled, you might want to terminate the replication, either to change the pod volume membership, or perhaps the replication was temporarily enabled to migrate the volumes from one array to another.

The act of terminating replication is called un-stretching. To un-stretch a pod, remove the array which you no longer need the volumes on. In other words, consider the following pod:

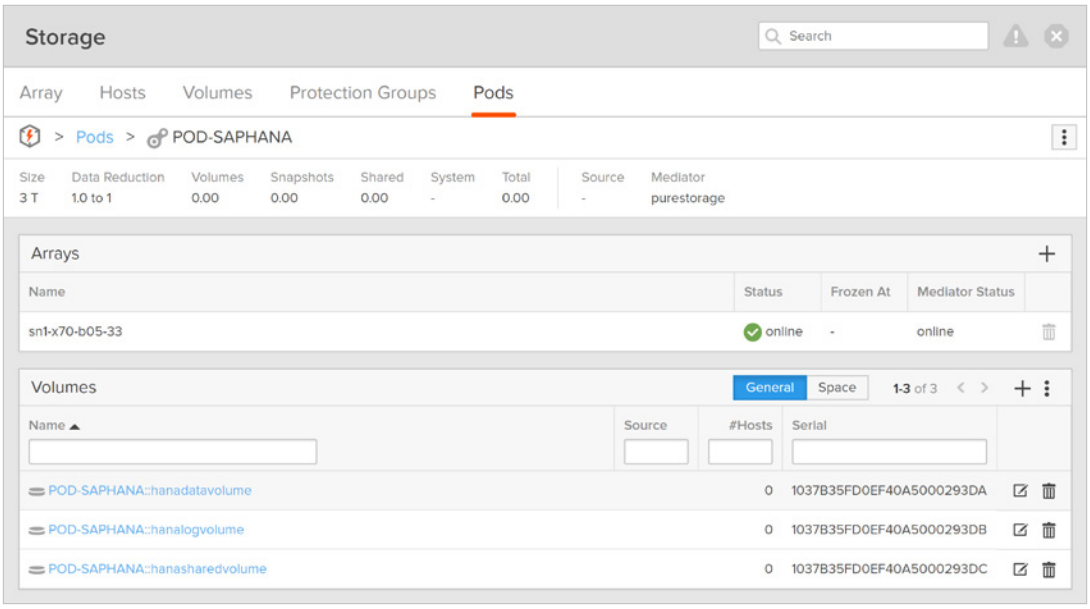



FIGURE 35. Example of active/active Pod

The pod has two arrays, s**n1-x70-b05-33** and **sn1-x70-c05-33**. Since this pod is online, both arrays offer up the volumes in the pod. If we want the volumes only to remain on sn1-x70-b05-33, we would remove the other FlashArray, sn1-x70-c05-33. This action can be performed on either array.

Before removing a FlashArray from a pod:

- Disconnect the pod's volumes from the host that is connected to the FlashArray that is being removed from the pod. Purity will not allow un-stretching a pod if even one of its volumes is connected to a host or host groups.
- Ensure that any hosts using those volumes have access to the FlashArray that you intend to keep in the pod. If hosts only have access to the pod you plan to remove through the FlashArray that is being removed, they will lose access to their volumes once the FlashArray is removed from the pod, which may result in an application error based on the setup.
- Ensure you are removing the FlashArray you intend to – double check the array before removing it.

> **Ensure you do not remove the wrong FlashArray** from a pod, or hosts may lose access to their storage.

To remove a FlashArray, choose the appropriate pod and, inside of the **Arrays** panel, click on the **trash icon** next to the array you would like to remove.


FIGURE 36. Choosing a Pod to eradicate

When you are sure it is the proper array to remove, click the **Remove** button to confirm the removal.

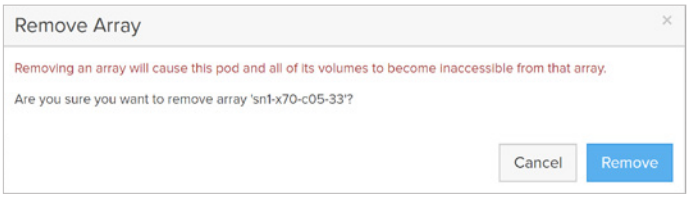



FIGURE 37. Confirm eradication

As a safety precaution, as well as a replication optimization, if you decide to restretch the pod within 24 hours, Purity places the pod that was removed under the **Destroyed Pods** panel on the FlashArray from which the pod was removed.

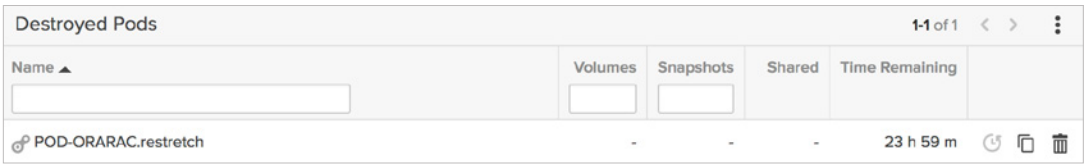



FIGURE 38. Destroyed Pods panel

If the un-stretch was done in error, go back to the pod on the GUI of the FlashArray that was removed and add the other FlashArray back. This will move the pod from **Destroyed Pods** status back to active status.

The pod can be instantly re-stretched for 24 hours. At 24 hours, the removed FlashArray will permanently remove its references to the pod. You can force this permanent removal early by selecting the destroyed pod and clicking the **trash icon** next to it.


FIGURE 39. Forcing permanent removal of a destroyed Pod

Click **Eradicate** to confirm the removal.

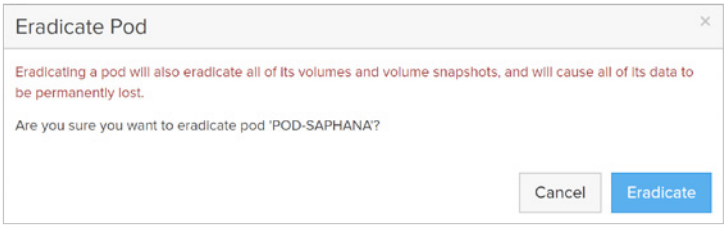



FIGURE 40. Confirm permanent removal

Once the pod is eradicated, either manually or via a time lapse beyond 24-hours, re-stretching the pod involves re-synchronizing all of the data from the other FlashArray.

### CONFIGURING SAP HANA HOSTS WITH ACTIVECLUSTER

Configuration of the SAP HANA hosts is no different than configuration for non-ActiveCluster FlashArrays, so all best practices described in the following document still apply: http://www.purestorage.com/content/dam/purestorage/pdf/whitepapers/pure-storage-for-sap-hana-setup-best-practices-guide.pdf.

With that in mind, there are still a few things worth mentioning in this document.

## Host Connectivity

FlashArray supports connections through iSCSI and Fibre Channel, and both are supported with ActiveCluster. Prior to being able to provision storage, a "host" object must be created on FlashArray for each SAP HANA node, respectively. Hosts can then be further organized into Host Groups. Host groups collect host objects together so that you can provision to them all at once. Host groups are recommended for clustered hosts, like an SAP HANA scale-out system, so that storage can be provisioned uniformly to all hosts.

### HOSTS

A FlashArray host object is a collection of a host's initiators that you can then proceed to "connect" a volume to. This allows those specified initiators (and therefore that host) to access that volume or volumes.

Create a host object on FlashArray by going to the **Storage** section and then the **Hosts** tab.

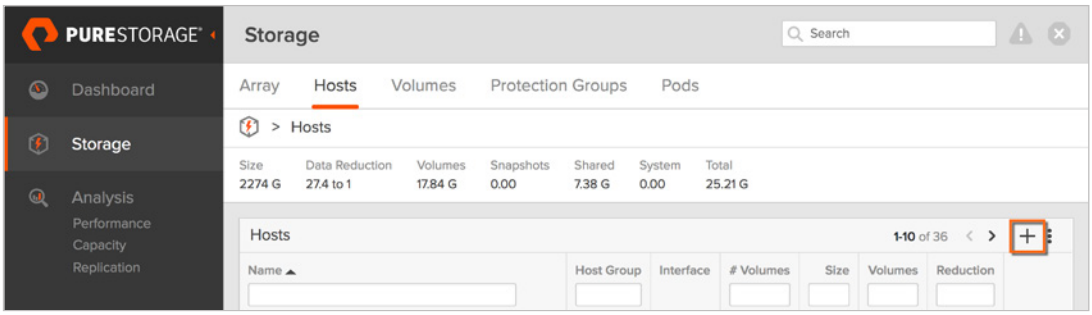



FIGURE 41. FlashArray GUI, Hosts panel

Click on the **plus sign** in the **Hosts** panel to create a new host. Assign a host a name that makes sense and click **Create**.


FIGURE 42. Create Host panel

Click on the newly created host, and then, in the **Host Ports** panel, click the **vertical ellipsis** and choose either **Configure WWNs** (for Fibre Channel) or **Configure IQNs** (for iSCSI).

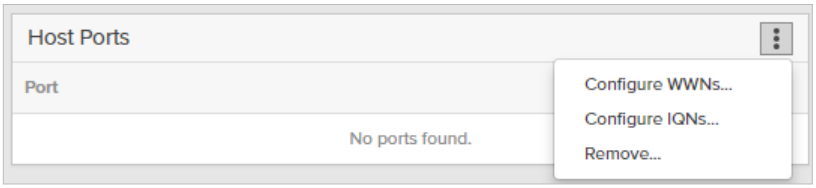



FIGURE 43. Host Ports panel

For WWNs, if the initiator is presented on the fabric to FlashArray (meaning zoning is complete), click in the correct WWN in the left pane to add it to the host, or alternatively click the **plus sign** and type it in manually. iSCSI IQNs must always be typed in manually.

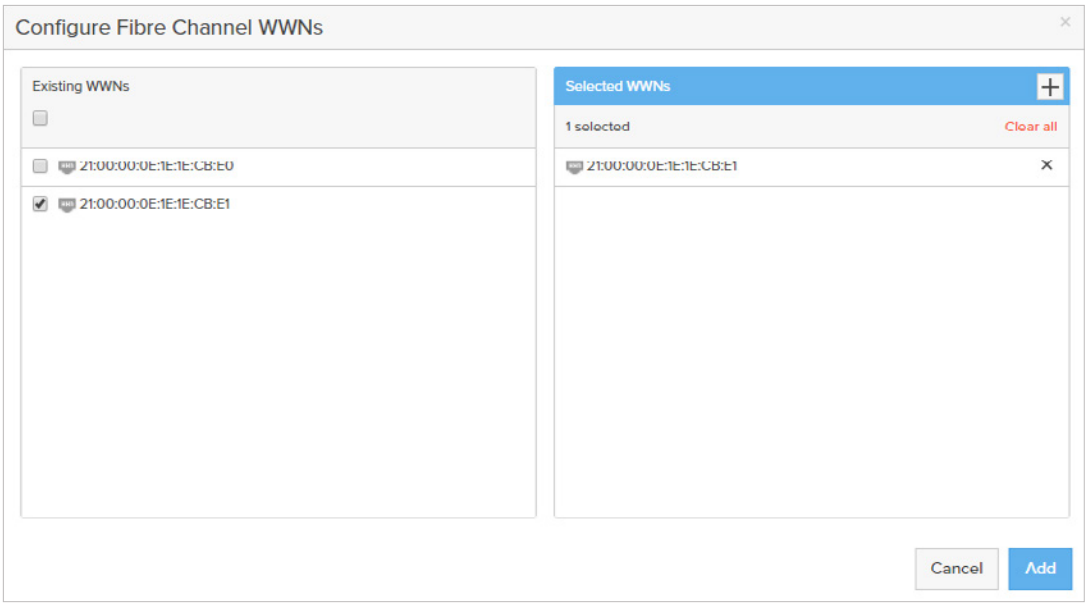



FIGURE 44. Selecting or adding the correct WWN

When all the initiators are added/selected, click **Add**.

> **Best Practice** – All Fibre Channel hosts should have at least two initiators for redundancy.

Verify connectivity by navigating to the **Health** section and then the **Connections** tab. Find the newly created host and look at the **Paths** column. If it lists anything besides **Redundant**, investigate the reported status.

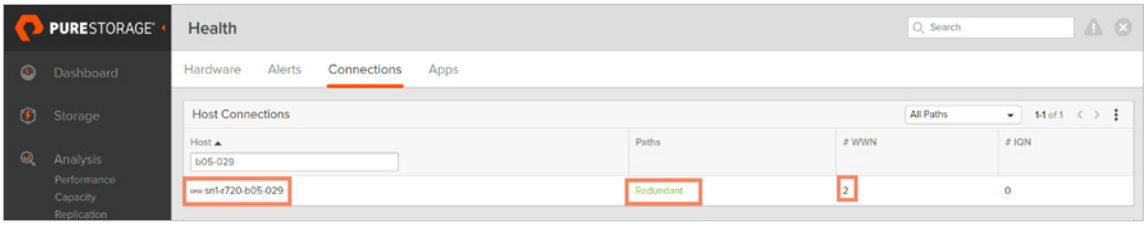



FIGURE 45. FlashArray GUI, Health panel

## Multipathing

Multipathing is an important configuration when it comes to setting up SAP HANA systems on ActiveCluster. In this section we will see in detail different methods and optimizations that need to be done based on read throughput and write latency. SAP HANA write latency for logs is very important, but again it varies customer to customer as to how to setup an SAP HANA system based on read throughput or write latency.

Let us explore all the different combinations and run through the advantages and disadvantages of the various multipathing configurations that are used for setting up optimized paths or non-optimized paths.

This is termed a **Uniform Access Configuration**, in which all hosts have (preferably redundant) access to both FlashArrays. Inter-site network performance and/or reliability are lower than those of the intra-site storage network, hence it is optimal for hosts at each site to route pod volume I/O to the array local to them and to access the remote array only when the local one is unresponsive. FlashArray provides an option named **preferred array** to specify whether a host connection to a volume in a stretched pod is optimized or not.

Most multipathing I/O stacks support Asymmetric Logical Unit Assignment (ALUA), a SCSI feature that allows a host to query an array to determine which of its paths to a logical unit (LUN or volume) are optimized, meaning a direct path to an array owning the LUN, and which are not. Hosts then route I/O commands solely to optimized paths and use non-optimized paths when the optimized paths are unavailable.

Standard multipathing recommendations apply, and are described in more detail in the Linux Recommend Settings found here: https://support.purestorage.com/Solutions/Linux/Reference/Linux_Recommended_Settings.

Recommendations at a high level include the following:

- Use the least queue depth path selection policy for FlashArray storage.
- Use multiple HBAs per host for Fibre Channel or multiple NICs per host for iSCSI.
- Connect each host to both controllers.
- In the storage or network fabric, use redundant switches.

### UNIFORM CONFIGURATION

In a uniform configuration, all hosts have access to both FlashArrays and can therefore see paths for an ActiveCluster-enabled volume to each FlashArray.

In this environment, we have one or two SAP HANA nodes with two hosts on Site A and two hosts on Site B and all hosts are zoned to both FlashArrays. This document covers two important scenarios, one with a single host and the other with two hosts, as seen below.

1. **Local High Availability** – In this scenario, SAP HANA is deployed on a single host with ActiveCluster-enabled storage.

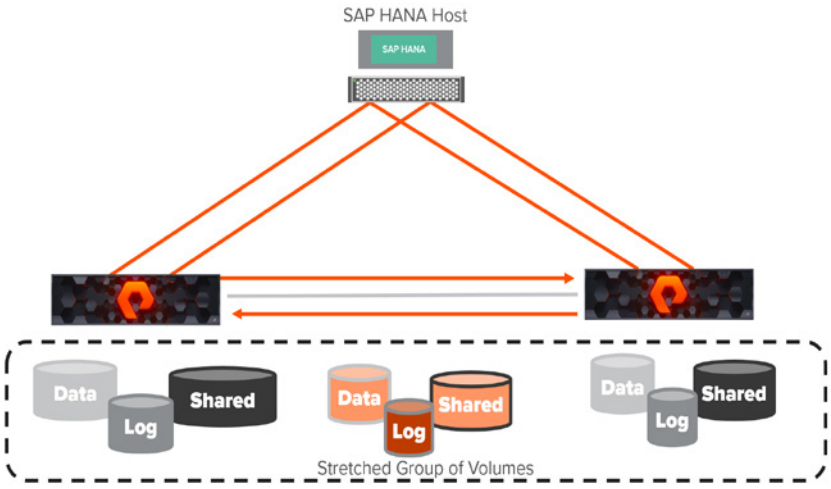



FIGURE 46. Local High Availability diagram

Here we have created one pod which can has three volumes added – data volume, log volume, and shared volume. The pod is stretched between two FlashArrays, so the host will see these volumes in two different paths, as seen below.

At the host level, we will see all these volumes:

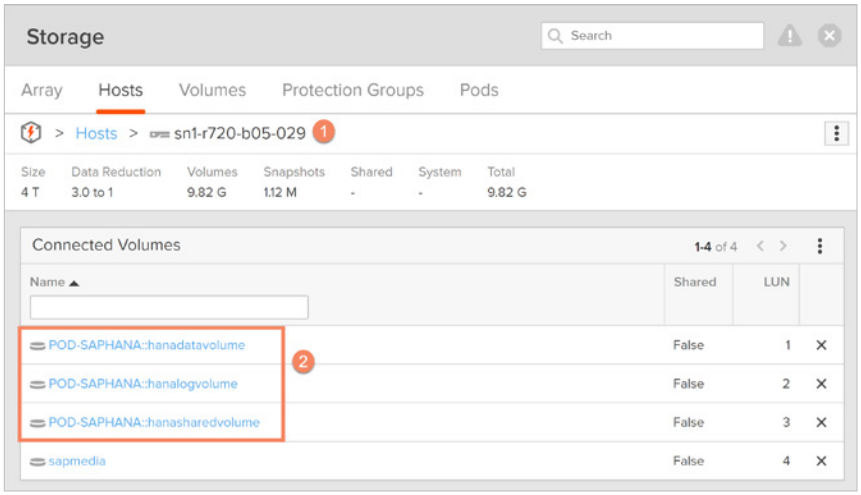



FIGURE 47. Volumes diplayed ont the Hosts panel

As seen here, the pod (**POD-SAPHANA**) is connected to both FlashArrays.

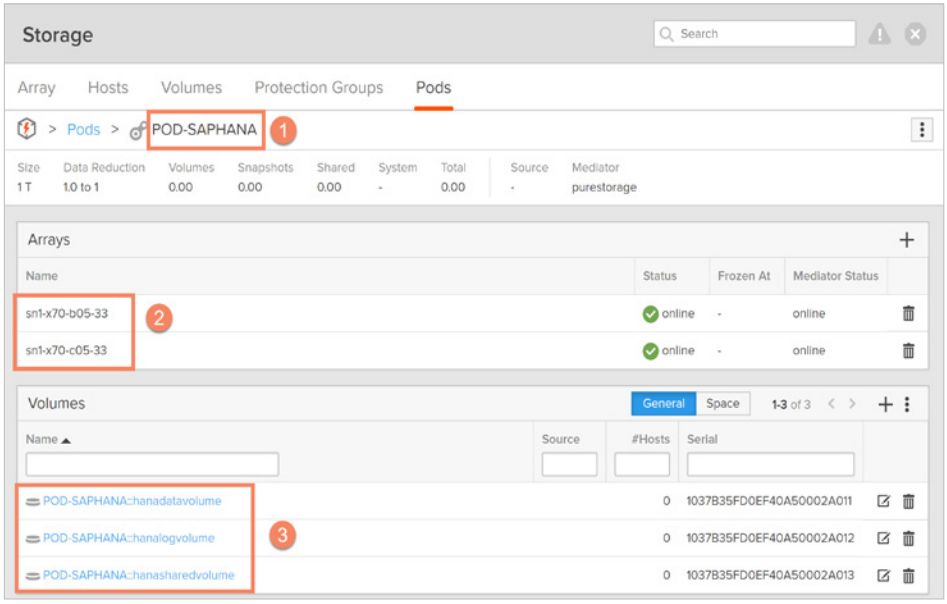



FIGURE 48. Pod connected to both FlashArrays

Since the pod is connected to both arrays, the volumes will be visible to the host from the other FlashArray as well.

2. **Remote Disaster Recovery** – In this scenario, SAP HANA is deployed on two different hosts with ActiveCluster-enabled storage.

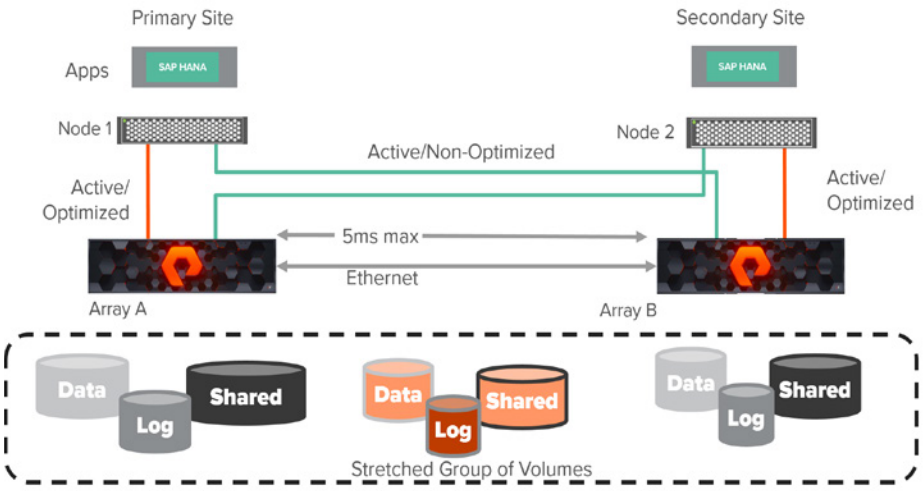



FIGURE 49. Remote Disaster Recovery diagram

In this case, there are two hosts and they will see the same set of volumes, which are added to the stretched pod.

The next step is to discover the volumes on the SAP HANA nodes. On the Linux hosts, this is accomplished by the command **rescan-scsi-bus.sh**.

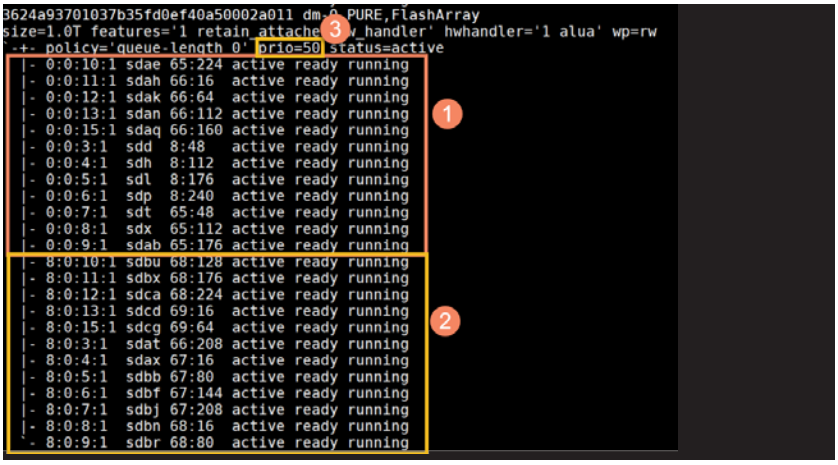
```
saphanaacc05:/dev/mapper # rescan-scsi-bus.sh
Scanning SCSI subsystem for new devices
Scanning host 0 for  all SCSI target IDs, all LUNs
 Scanning for device 0 0 3 1 ...
```

Now you should see the following new volumes in the /dev/mapper.


FIGURE 50. Discovered Volumes

```
saphanaacc05:/dev/mapper # ll
total 0
lrwxrwxrwx 1 root root       7 Feb 28 15:42 3624a93701037b35fd0ef40a5000124ef -> ../dm-3
lrwxrwxrwx 1 root root       7 Feb 28 15:42 3624a93701037b35fd0ef40a50002a011 -> ../dm-0
lrwxrwxrwx 1 root root       7 Feb 28 15:42 3624a93701037b35fd0ef40a50002a012 -> ../dm-1
lrwxrwxrwx 1 root root       7 Feb 28 15:42 3624a93701037b35fd0ef40a50002a013 -> ../dm-2
```

As these volumes are attached to a stretched pod between FlashArrays, the paths at the host level refer to both arrays. The multipath command in Linux will show device details and multipath configurations.




The following is an explanation of the above **multipath –ll** output:

- No. 1 shows the optimized paths from the first array.
- No.2 shows the optimized paths from the second array.
- No.3 shows a priority of 50 (optimized) with a status of “active”.

The number of ports at the initiator, two in this case, times the number of ports at the target, two, yields a total path count of four when the **multipath -ll** command was run. As these paths are from the first FlashArray, they are local and optimized paths. The hardware handler shows the module that should be used to perform hardware-specific actions

when switching path groups or handling IO errors. The recommended setting is **1 alua**, which is the hardware handler for a SCSI-3 ALUA-enabled array.

You will note from the above screenshot that all paths are optimized – even ones from the secondary FlashArray. This is generally not ideal. In situations where the sites are far apart, two performance-impacting things occur:

- Half of the writes (assuming both FlashArrays offer an equal number of paths for each device) sent from a host in site A will be sent to the FlashArray in site B. Since writes must be acknowledged in both sites, this means the data traverses the WAN twice. First the host issues a write across the WAN to the far FlashArray, and then the far FlashArray forwards it back across the WAN to the other FlashArray. This adds unnecessary latency. The optimal path is for the host to send writes to the local FlashArray and then the FlashArray forwards it to the remote FlashArray. In the optimal situation, the write must only traverse the WAN once.
- Half of the reads (assuming both FlashArrays offer an equal number of paths for each device) sent from a host in site A will be sent to the FlashArray in site B. Reads can be serviced by either side, and for reads there is no need for one FlashArray to talk to the other. So, a read need not ever traverse the WAN under normal circumstances. Servicing all reads from the local array to a given host is the best option for performance.

FlashArray offers an option to intelligently tell SAP HANA nodes which FlashArray should optimally service I/O in the event a host can see paths to both FlashArrays for a given device. This is a FlashArray host object setting called **Preferred Arrays**.

For a given host, login to FlashArray's Web Interface. Click on the **Storage** section, then the **Hosts** tab, then choose the host you want to configure. Then, in the **Details** panel, click on the **Add Preferred Arrays** option.

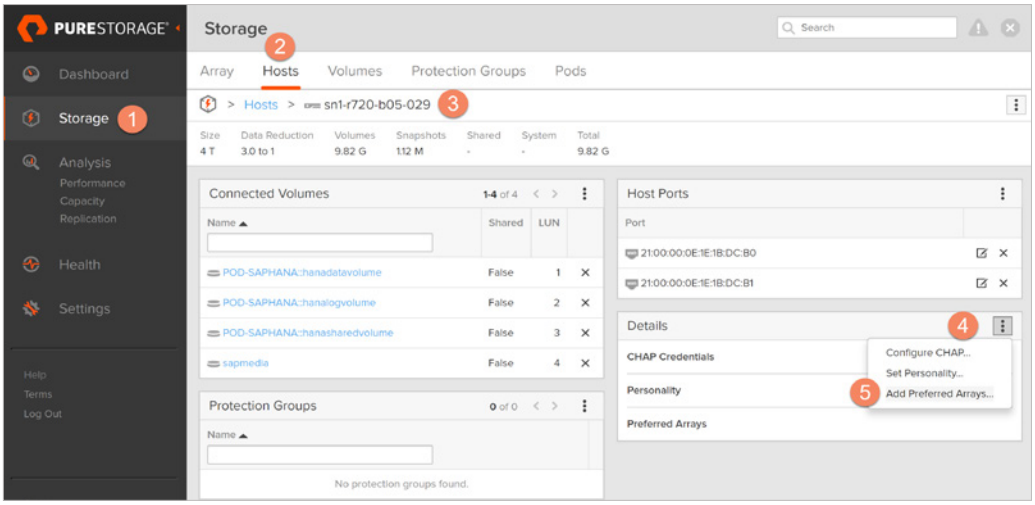



FIGURE 51. Steps to add Preferred Arrays

Choose the FlashArray that you would like to be local for that SAP HANA node and click **Add**. Repeat the steps on the other FlashArray as well, but *make sure the host points to the same preferred FlashArray on both arrays*.

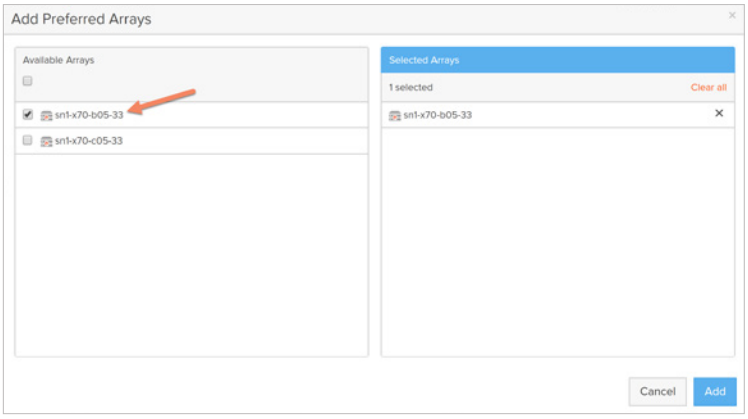
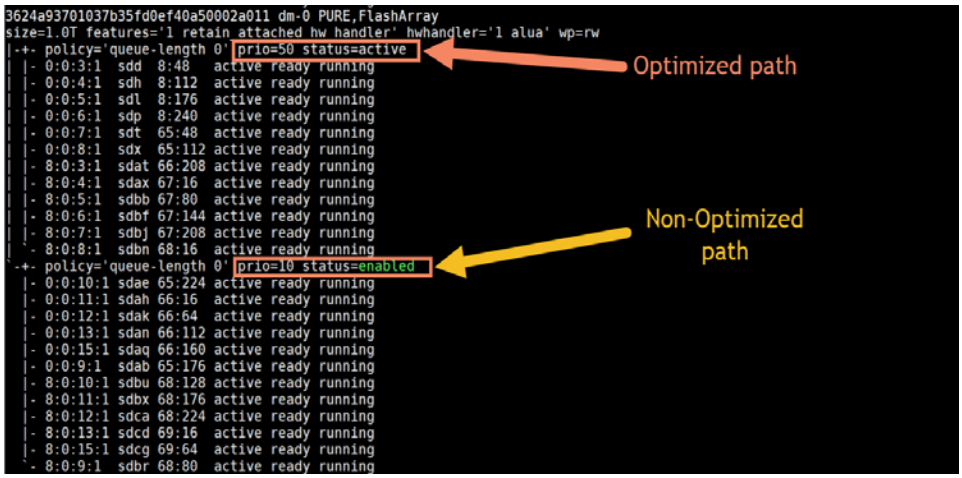



FIGURE 52. Add Preferred Arrays panel

You will notice that half of the paths will be marked with a priority of 50 (optimized), with a status of “active” and the other half will be marked with a priority of 10 (non-optimized), with a status of “enabled”.




## MULTIPATH.CONF

This is the recommended configuration for multipath.conf

```
defaults {
       polling_interval      10
}

devices {
      device {
            vendor                  "PURE"
            path_selector           "queue-length 0"
            hardware_handler        "1 alua"
            path_grouping_policy    group_by_prio
            prio                    alua
            failback                immediate
            path_checker            tur
            fast_io_fail_tmo        10
            user_friendly_names     no
            no_path_retry           0
            features                0
            dev_loss_tmo            60
      }
}
```

## DEPLOYMENT OVERVIEW

The SAP HANA deployment in our environment uses two FlashArray //x70 arrays to host the database storage and two Cisco UCS B200 M4 blades for the SAP HANA nodes which are connected to the storage arrays over FC protocol. FlashArrays also support iSCSI protocol for customers who prefer Ethernet switching over Fabric.

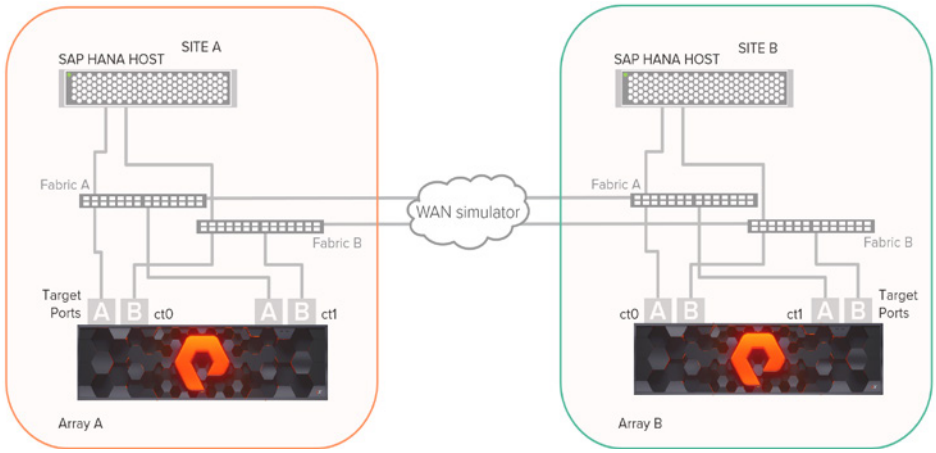



FIGURE 53. Deployment overview

The following describes each of the components used for active cluster testing.

| COMPONENT | DESCRIPTION |
|---|---|
| STORAGE | 2 X FLASHARRAY //M70 WITH 16GB FC CONNECTIVITY |
| SERVERS | 4 CISCO UCS B200 M4 BLADES EACH WITH<br>2 X INTEL XEON E5-2670 @ 2.6 GHZ (16 CORES TOTAL)<br>256 GB RAM<br>SUSE LINUX 12 SP2 FOR SAP APPLICATIONS |
| FABRIC SWITCH | 4 X BROCADE 7840 |
| WAN SIMULATOR | NETEM (NETWORK EMULATOR) |
| SAP HANA | SAP HANA 2.0 SP02 |

## SAP HANA 2.0 TESTS WITH ACTIVECLUSTER

In this section we will compare and test the local high availability and Remote Disaster Recovery scenario for the SAP HANA platform.

### LOCAL HIGH AVAILABILITY SETUP FOR SAP HANA

SAP HANA is deployed on a single host with ActiveCluster-enabled arrays. In the event of storage failure on the primary side, the SAP HANA application will run uninterruptedly on the secondary storage. This is a zero RTO and zero RPO scenario. We should make sure that the pod has data volume, log volume, and shared volume.

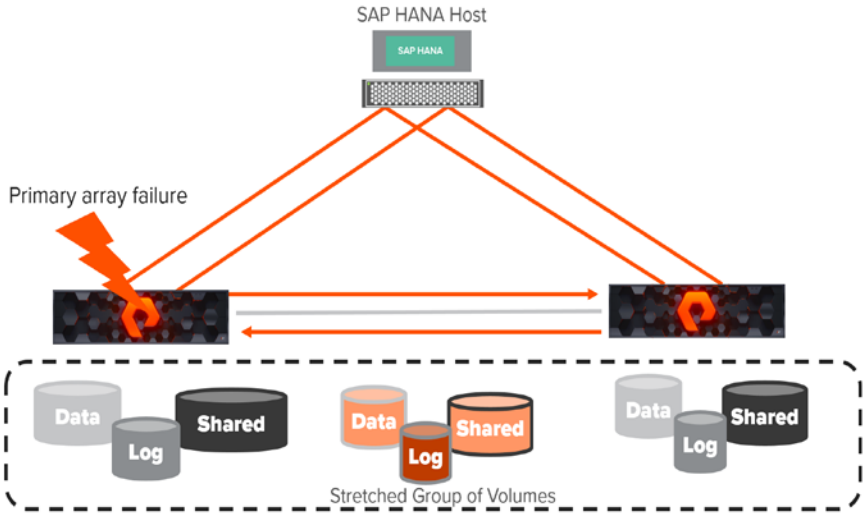



FIGURE 54. Local High Availability diagram

## Failure Scenarios

### ARRAY FAILURE

In this test, an array failure was simulated by powering off one of the arrays while running standard queries on SAP HANA. The figure illustrates the high level set up of the test environment. The hosts are connected in the Uniform access configuration with ALUA settings, in which the host has access to both arrays. Paths connected to the local array appear to the host as “Active/Optimized”, while paths connected to the remote array appear to the host as either “Active/Optimized or “Active/Non-Optimized”.

Once the primary array fails, SAP HANA continues to function as if nothing has happened, as long as data volume, log volume, and shared volume are present in the pod. The host continues to see half the paths still alive, and that is the reason SAP HANA continues to function normally.

## Performance Results

To check the write latency for SAP HANA, the **SAP HWCCT** tool was used. In data center mode, even when the arrays are very close to each other (which is a round trip time of 0ms), it will still cause write latency to go up. The latency of

writes, whether one of them is a preferred path or not, will cause all the writes to increase in latency, as there will be a round trip between the arrays before sending the acknowledgement back to the SAP HANA node.

The diagram below shows the write latency difference between SAP HANA deployed on an active cluster to an SAP HANA system deployed without active cluster. It shows that latency for log writes for an average 4K – 16K block size without active cluster is around 350 microseconds. With active cluster, the log writes need to be written to the other array, so there will be an IO round trip between the arrays which will cause the latency to go around ~650 microseconds. This is still within the limits of SAP HANA certification requirements for latency.

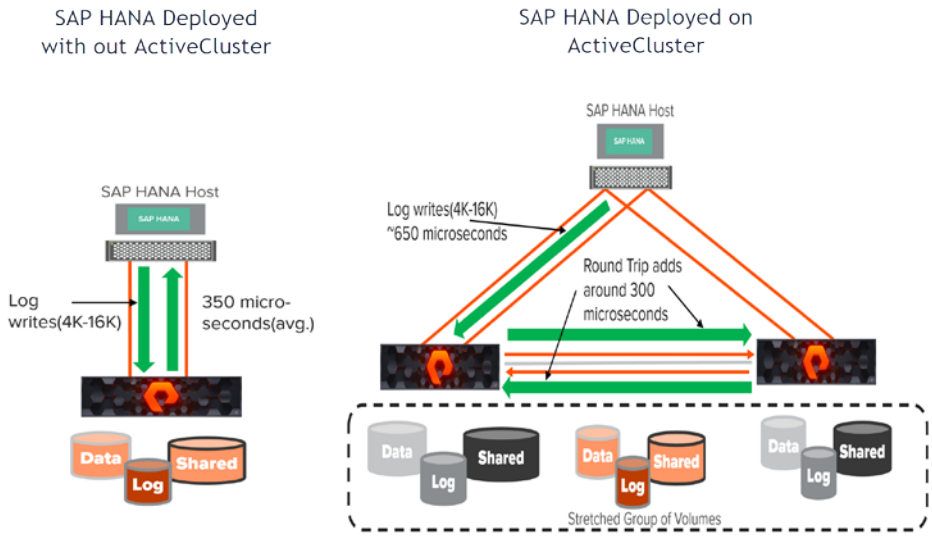



FIGURE 55. Write latency difference with and without an active cluster

The average latency was calculated based on the results of five runs of SAP HWCCT.

## PERFORMANCE RESULTS (DETAILED ANALYSIS)

In this section, we will drill down and see how optimizing paths to an array or both the arrays will cause a performance difference in read and write IO. We will explore the different scenarios and test the performance of reads and writes for a local high availability setup for SAP HANA on ActiveCluster using the SAP HANA HW Configuration Check Tool (HWCCT). In addition, we will simulate distance between the arrays by increasing the round trip time from 0ms to 4ms using the WAN simulator between the arrays.

## OPTIMIZED PATHS TO BOTH ARRAYS

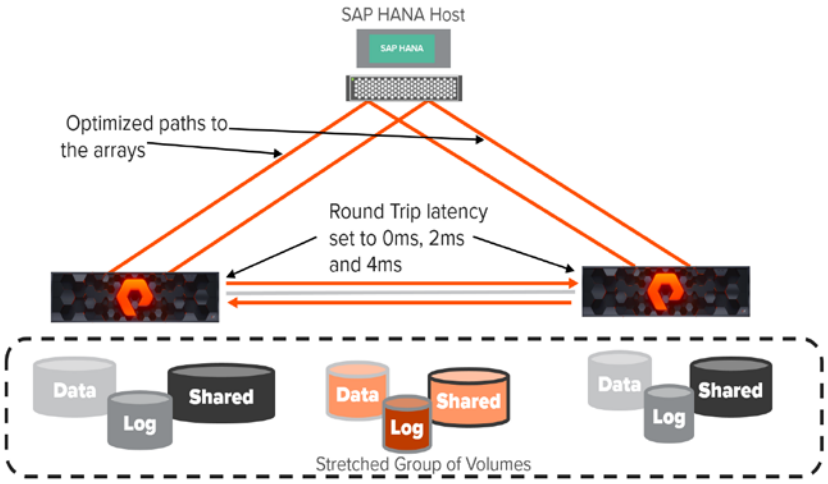



FIGURE 56. Optimized paths to both arrays

We will test read and write performance when the host has equal priorities to both the active clustered arrays. The WAN latency simulator that is simulating distance was set to 0ms, 2ms, and 4ms. With optimized paths to both arrays, the throughput increases for reads and writes from 1.6x to 2x as IO is distributed and parallelized from the host. However, this also increases the average latency for the logs (4k – 16k block sizes) as compared to Optimized/Non-optimized paths to the arrays.

## OPTIMIZED/NON-OPTIMIZED PATHS TO BOTH ARRAYS

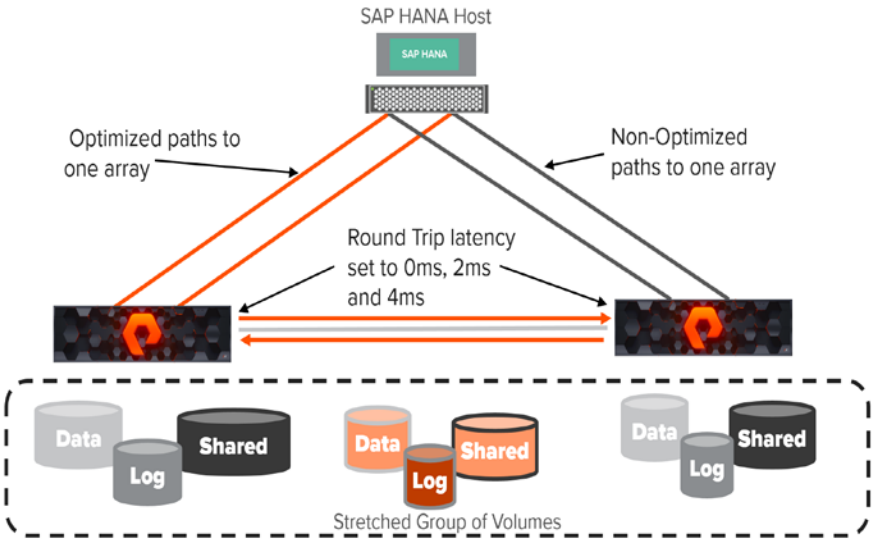



FIGURE 57. Optimized/non-optimized paths to both arrays

Here we test read and write performance when the host has unequal priorities, meaning the host has a preferred array for IO among the active clustered arrays. The WAN latency simulator that is simulating distance was set to 0ms, 2ms, and 4ms. With optimized paths in which each host has a preferred array, the latency is much less for the logs (4k – 16k block sizes) as compared to configuration of optimized paths to both arrays.

| | IO Type/WAN latency | Round trip latency(0ms) | Round trip latency(1ms) | Round trip latency(2ms) | Round trip latency(3ms) | Round trip latency(4ms) |
|---|---|---|---|---|---|---|
| Optimized paths to both arrays | Write throughput (1MB block size) | ~1920 MB/sec | 1873 MB/sec | ~1830 MB/sec | ~1794 MB/sec | ~1720 MB/sec |
| | Avg. Write Latency logs (4K block size) | ~1.1 ms | ~2.68 ms | ~4 ms | ~5.62 ms | ~7 ms |
| | Avg. Write Latency logs (16K block size) | ~1.6 ms | ~3.15 ms | ~4.6 ms | ~6.27 ms | ~8 ms |
| Optimized/Non-optimized paths to arrays | Write throughput (1MB block size) | ~810 MB/sec | ~863 MB/sec | ~890 MB/sec | ~913 MB/sec | ~978 MB/sec |
| | Avg. Write Latency logs (4K block size) | ~0.650 ms | ~1.77 ms | ~2.63 ms | ~3.81 ms | ~4.93 ms |
| | Avg. Write Latency logs (16K block size) | ~0.7 ms | ~1.83 ms | ~ 2.85 ms | ~3.76 ms | ~4.83 ms |

Let us start with a throughput comparison between these two different configurations: clearly, the throughput is much better when both arrays are configured to have optimized paths. This increases the number of paths and IO is distributed across multiple paths to both the arrays. The throughput comparison chart shows that, for a 1 MB block size, writes to data volumes when both arrays have optimized paths are much higher – almost 2x more than the throughput achieved by the configuration when there are optimized paths to a single array or if there is a preferred array to the host. The charts below show throughput data points for different distances, which are simulated using the WAN simulator. However, this also comes with the drawback that the average latency is also much higher than having only one preferred array.

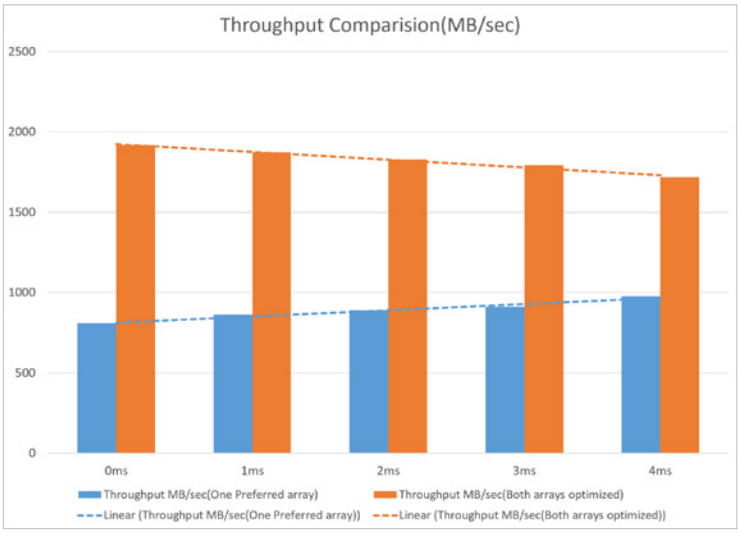



TABLE 1. Throughput comparison for 1MB block size

Below is the latency comparison between two different configurations for 4K and 16K block size log writes; clearly, the latency is much better when there are optimized paths only to a single array. As IO always goes to the preferred array, the latency will be much less, especially if the preferred array is a closer array to the host.

This also comes with the drawback that average throughput is much lower than having optimized paths to both arrays.

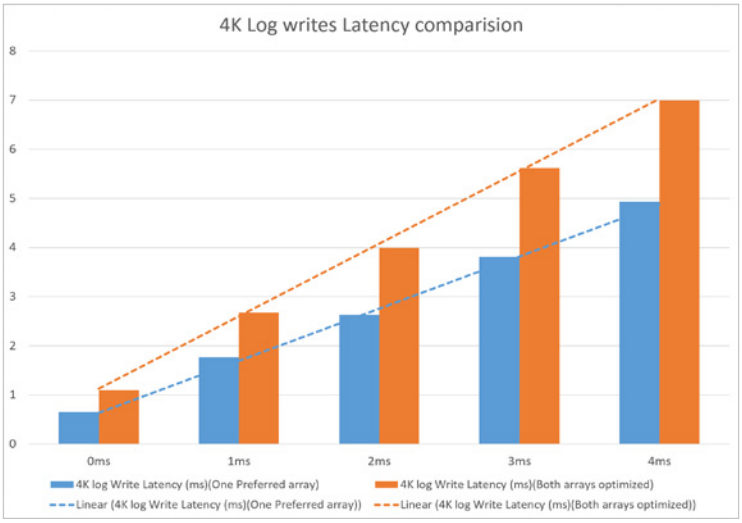



TABLE 2. Latency comparison for 4K block log writes

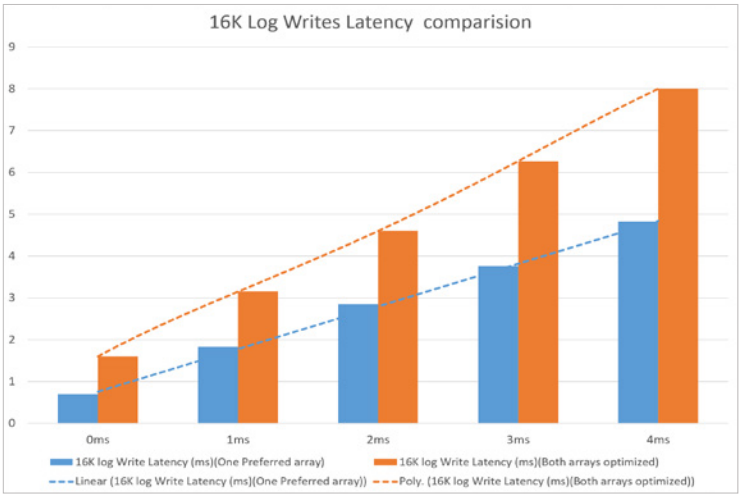



TABLE 3. Latency comparison for 16K block log writes

We highly recommend using the preferred array configuration, as latency for log volumes is lower and transactions in SAP HANA are committed faster. The advantage of having a higher throughput option, in which paths to both the arrays are optimized, is that operations like delta merges will be much faster as throughput is distributed evenly across the arrays.

### REMOTE DISASTER RECOVERY

In this scenario, SAP HANA is deployed on two different hosts with ActiveCluster-enabled storage. In the event of a failure on the primary site which includes the primary host and storage for the SAP HANA application, the secondary site SAP HANA application can be brought up. This is not a zero RTO scenario.

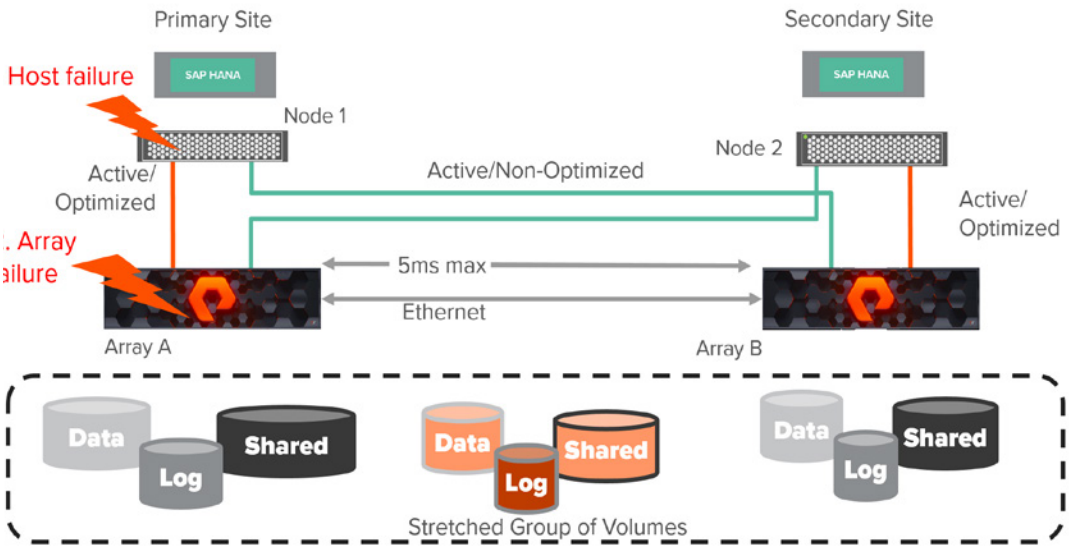



FIGURE 58. Remote Disaster Recovery diagram

## Failure Scenarios: Primary Site Failure (Host and Array) Failure

In this test, an array failure and host failure was simulated by powering off one of the arrays and shooting the node down. The figure above illustrates the high level set up of the test environment. The hosts are connected in the Uniform access configuration (where all hosts have access to both arrays). With ALUA settings, paths connected to the local array appear to the host as “Active/Optimized” while paths connected to the remote array appear to the host as “Active/Non-Optimized”.

### FAILURE SCENARIO 1: PRIMARY SITE ARRAY FAILURE

For this scenario, the primary array failure is simulated and then the primary SAP HANA host loses half of the paths but keeps half the paths alive due to the secondary array. This is similar to the local high availability example discussed in the previous section. SAP HANA will continue to function normally – with degraded performance if the secondary site is far away.

### FAILURE SCENARIO 2: PRIMARY SITE HOST AND ARRAY FAILURE

For this scenario, a primary host and array failure is simulated. The SAP HANA node on the secondary host can be recovered without any issues as long as SAP HANA data volume, log volume, and shared volume are added to the pod.

On the secondary site, the following activities are required to be performed:

1. If there is another instance of SAP HANA running on the secondary site, it needs to be shut down.
2. Unmount the other instances of SAP HANA volumes.
3. Mount the primary instance SAP HANA volumes, which are present in the pod.
4. Start the SAP HANA system by switching to **<sid>adm** user.

> An SAP HANA shared volume based on NFS was also added to the pod, and we built the primary SAP HANA system based on it. It came up successfully on the secondary host, which shows that scale out systems can also be recovered on the secondary site.
>
> It is also highly recommended to run the **hdbnsutil –convertTopology** user using the **sidadm** user when bringing up the SAP HANA instance on the secondary site.

## SUMMARY

SAP HANA was tested in two main scenarios. Depending on RPO and RTO requirements and the criticality of the system, either of the following scenarios will work.

- **Local HA** – This is a scenario in which RTO and RPO should stay as close as possible to zero. As the name implies, this is a configuration where High Availability in the same datacenter is required.
- **Remote Disaster Recovery** – This is a scenario in which HANA Storage Replication is used for disaster recovery needs in a different data center. In this scenario, RTO will no longer be at zero, as there are other variables in play.

We believe simplicity helps reduce errors, which is a significant requirement in SAP environments, and also provides more options to customers looking to deliver on their SAP investment. Synchronous replication and metro clustering remain advantageous to storage technology, but unfortunately complexity and additional cost can hinder such deployments. With Pure's simple ActiveCluster solution, a true active/active solution is introduced. ActiveCluster is fully integrated into Purity, and available at no additional cost as part of a Pure customer's Evergreen® subscription.

## REFERENCES


1. https://support.purestorage.com/FlashArray/PurityFA/Protect/Replication/ActiveCluster_Requirements_and_Best_Practices

2. https://support.purestorage.com/FlashArray/PurityFA/Protect/Replication/ActiveCluster_Solution_Overview

3. https://support.purestorage.com/Solutions/Linux/Reference/Linux_Recommended_Settings





ps_wp43p_sap-hana-with-activecluster_ltr_01