PURESTORAGE®
Uncomplicate Data Storage, Forever

TECHNICAL WHITE PAPER

High Availability for MySQL with FlashArray Solutions

Detailed guidance for choosing a high-availability solution for MySQL databases.

Contents

Introduction

Achieving high availability for databases can be a major challenge for database professionals. The height of availability for databases is driven by the interplay of consistency, availability, and partition tolerance for the data that database administrators (DBAs) maintain. However, database professionals can only prioritize two of these factors, as optimizations for any two factors must come at the expense of the third.

Navigating the high-availability and disaster-recovery solution landscape can be a challenge. Administrators must consider data consistency; for example, to what degree do they need to worry about data loss during failover? Latency is another important consideration; do applications need low latency, or can they tolerate slightly higher latency?

Sometimes the relevant questions for designing a system that meets a business's needs are more nuanced. For example, does the business need failover capabilities, or is a distributed system needed? For scalability, does database performance need to scale? However, answering these questions does not need to be overwhelming.

Selecting the Correct High-availability Solution

This guide can help you select the right solution that will work best to meet your high-availability needs for MySQL workloads in conjunction with Pure Storage® FlashArray™ products. While there is some overlap, each workload environment and application architecture described in this guide provides unique high-availability benefits. These differences should be accounted for in selecting a high-availability solution for MySQL. However, regardless of the backup solution you choose, FlashArray can help increase the speed of your high availability failover. Table 1 highlights tools and functions that you can use to implement a high-availability strategy for MySQL with FlashArray.

Table 1. Tools for high availability with MySQL and FlashArray

MySQL Clustering for Increased Availability		FlashArray Storage Availability	
InnoDB Cluster	Galera Cluster	Controller redundancy	ActiveCluster™ (99.9999% availability)
A solution for MySQL that provides high availability through a combination of MySQL Router, MySQL Group Replication, and MySQL Shell	Synchronous multi-primary clustering for MySQL databases	Active/passive storage architecture that provides seamless controller failover	Synchronous storage replication between FlashArray devices
Link to external documentation	Link to external documentation	Link to external documentation	Link to external documentation

Unified Block and File on FlashArray

Pure Storage FlashArray is a software-defined, unified block- and file-storage product. It offers an effortless and consistent experience, and it behaves in an efficient manner by offering data reduction without an impact on performance. All Pure Storage products offer an Evergreen® subscription model to increase capacity and performance without the need to purchase new storage products. FlashArray also enables businesses to reduce direct carbon usage in their data storage systems by up to 80 percent, compared to competitive all-flash systems, with even better results compared to magnetic disks.

Pure Storage Purity software in FlashArray protects against concurrent dual-drive failures and initiates re-builds automatically within minutes. It also treats performance variability as a failure and uses parity to work around bottlenecks to deliver consistent latency.

The FlashArray product line caters to multiple business needs and use cases, with the distinct offerings shown in Figure 1:

- FlashArray//C™: An all-quad-level-cell (QLC) FlashArray product with consistent performance at 2–4ms latency for capacity-oriented workloads.
- FlashArray//X™: FlashArray//X products provide latency as low as 150µs to power critical applications and business operations.
- FlashArray//XL™: These products provide enterprise-grade performance and scalability for demanding workloads.

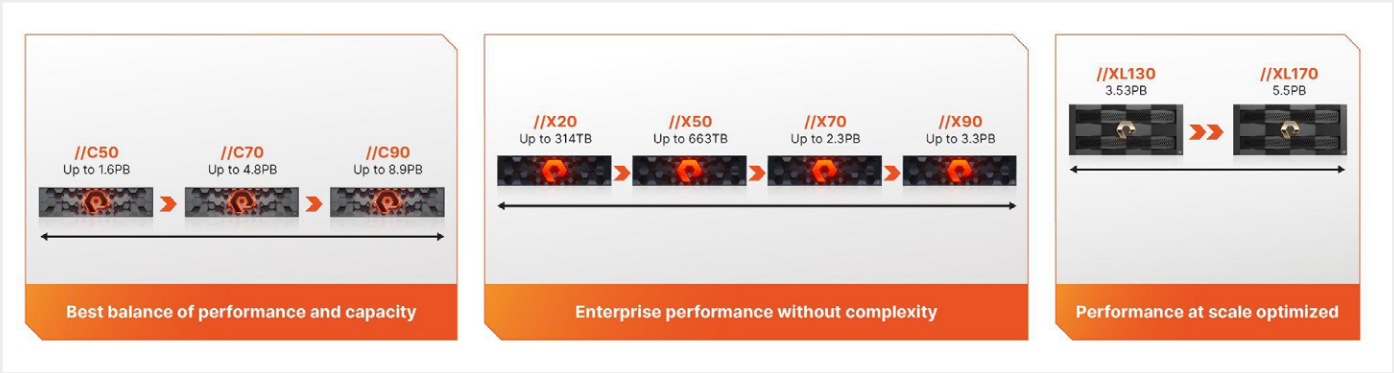

FIGURE 1 . Pure Storage FlashArray suite of products

FlashArray is powered by Purity for FlashArray. Purity delivers rich, enterprise-level data services that help ensure data is stored in the most secure and efficient way while providing additional functionality to extend storage capabilities:

- **Data reduction**: Purity averages an industry-leading 5:1 data reduction, with a total efficiency of 10:1 (including thin provisioning).
- **High availability**: Purity protects against concurrent dual-drive failures and initiates re-builds automatically within minutes.
- **Always-on ransomware remediation**: Cost-efficient, portable SafeMode™ Snapshots prevent cyber-attackers from tampering with or maliciously destroying critical recovery data.
- **On-demand data portability**: Quickly and easily move data where it most cost-effectively meets service-level agreements (SLAs) to satisfy customers.
- **Rich data services**: Data-service functionalities, such as ActiveCluster, ActiveDR™, and asynchronous replication, provide increased resilience and availability and enhance applications' workflows.

High Availability with MySQL Utilities

MySQL utilities provide high availability by making one or more of the nodes in a cluster the primary node and by routing traffic to another node should the primary node (or one of the primary nodes) fail. Failover to another node can be automatic. Examples of utilities that can provide high availability for MySQL are InnoDB Cluster and Galera Cluster.

InnoDB Cluster

InnoDB Cluster is a high-availability solution provided by MySQL. It consists of at least three MySQL Server instances and provides high-availability and scaling features. Each server in InnoDB Cluster replicates data to all members of the cluster while also providing fault tolerance and automated failover.

InnoDB Cluster is a solution for MySQL consisting of three principal components:

- **MySQL Shell:** An advanced client and code editor for MySQL that provides scripting capabilities and includes APIs for working with MySQL, including InnoDB Cluster.
- **MySQL Router:** A lightweight middleware solution that provides transparent routing between applications and InnoDB Cluster.
- **MySQL Group Replication:** A plug-in that provides virtually synchronous replication with built-in conflict detection/handling and consistency guarantees. MySQL Group Replication supports multi-primary, write-anywhere replication topologies. InnoDB Cluster uses MySQL Group Replication as its replication technology. When using a single primary node, it replicates from the primary node to one or more secondary nodes, and, in the event of a primary-node failure, one of the secondary nodes can take over as the new primary node.

Note that FlashArray can be used to provide additional storage-level availability, even when using software-level replication. (For more information, see Using InnoDB with FlashArray below.)

How to Use

The following steps provide an overview of using InnoDB Cluster for achieving high availability for MySQL:

1. **Set up a multi-node cluster** with multiple MySQL servers running InnoDB Cluster, such as MySQL Group Replication; this cluster should have a minimum of three instances and a maximum of nine instances of MySQL running in it.
2. **Set up replication** between multiple MySQL servers running InnoDB.

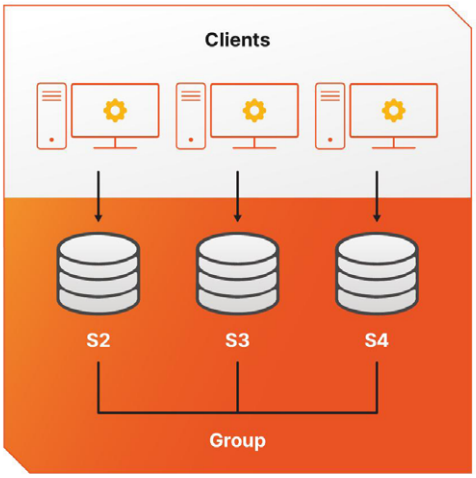

FIGURE 2 Example configuration for a MySQL Group Replication group

Using InnoDB with FlashArray

Copying data to the secondary nodes in an InnoDB Cluster can be time consuming. This challenge can be addressed by combining FlashArray snapshots with group replication to speed up seeding times for replicas. Moreover, because the high availability of the MySQL database is not available until the seeding is complete, FlashArray snapshots speed up this process by enabling the quick seeding of replicas through restoring snapshots of primary nodes to the secondary nodes in order to provide protection to MySQL environments faster.

For details on cloning a volume from a snapshot in MySQL Group Replication, see the documentation (login required).

FlashArray increases database availability beyond speeding up node seeding. FlashArray can be used to add additional layers of database availability through its highly available controller architecture or the implementation of ActiveCluster. This adds database availability at the storage layer, in addition to availability solutions in place at the software layer, such as InnoDB. (Note that automatic failover options can also be added to the application layer if availability at yet another layer of the database is desired.)

Best Usage Scenarios

InnoDB Cluster is best for scenarios such as:

- **Low-latency applications:** InnoDB Cluster provides almost synchronous replication, which can result in lower latency than synchronous replication solutions and can also result in less additional work that could be needed to resynchronize replicas.
- **Applications with complex SQL queries:** InnoDB Cluster supports row-level locking and multiple isolation levels, which makes it suitable for applications with complex SQL queries.
- **Applications with specialized needs:** InnoDB Cluster can be more easily tailored to meet specific application needs than can traditional synchronous-replication solutions. This is due to the solution's support for flexible replication modes, dynamic node addition and removal, and automatic failover, all of which combine to make InnoDB Cluster customizable to meet specific application needs.

Galera Cluster

Galera Cluster is a replication solution that can be used to create high-availability MySQL data through the use of synchronous multi-primary replication. Because it provides synchronous replication, all nodes in the cluster are equal and can write to the database simultaneously, providing true multi-primary capabilities. Galera Cluster provides scalability by simply adding more nodes. Galera Cluster is open source and is also part of some other high-availability solutions such as Percona XtraDB Cluster, which uses Galera Cluster for its replication capabilities.

Note that FlashArray can be used to provide additional storage-level availability, even when using software-level replication. (For more information, see Using Galera Cluster with FlashArray below.)

How to Use

Galera Cluster is powered by the MySQL-wsrep patch for MySQL and MariaDB. The patch only supports Linux/Unix operating systems, and thus Galera Cluster will only work on those operating systems.

The following steps provide an overview for using Galera Cluster for achieving high availability for MySQL:

1. Install Galera Cluster on each node in the cluster. Configure Galera Cluster replication settings, including:
    a. Node addresses
    b. Replication user credentials
2. Configure load balancing so that client requests are distributed evenly among all nodes in the cluster. Note that it is possible to use a hardware load balancer or a software-based solution, such as HAProxy.
3. Enable synchronous replication by setting the wsrep_sync_wait parameter to “1” in the Galera Cluster configuration.

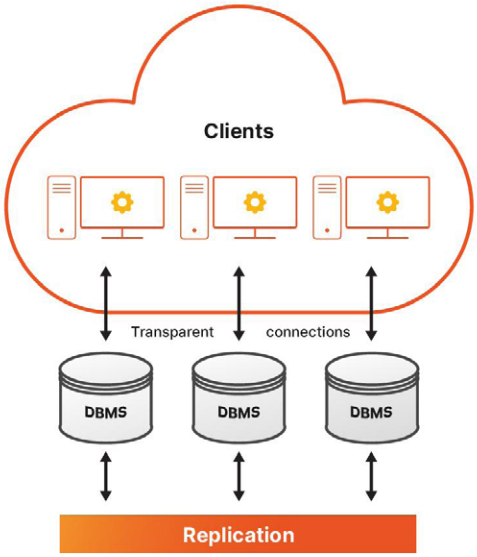

FIGURE 3 Example configuration of Galera Cluster

Using Galera Cluster with FlashArray

FlashArray snapshots can be combined with Galera Cluster to speed up node seeding times for replicas. Replicating data between the other nodes in Galera Cluster can take a long time using native copying in Galera Cluster. Copying FlashArray snapshots of one node to the other nodes in the cluster can significantly speed up cluster setup and more quickly achieve high-availability protection for MySQL databases.

For details on cloning a volume from a snapshot in Galera Cluster, see the documentation (login required).

FlashArray increases database availability beyond speeding up node seeding. FlashArray can be used to add additional layers of database availability through its highly available controller architecture or the implementation of ActiveCluster. This adds database availability at the storage layer, in addition to availability solutions in place at the software layer, such as InnoDB. (Note that automatic failover options can also be added to the application layer if availability at yet another layer of the database is desired.)

Best Usage Scenarios

Galera Cluster is best for scenarios such as:

- **Large-scale applications:** Galera Cluster easily scales by adding more nodes.
- **Real-time applications:** Galera Cluster provides synchronous replication, which helps ensure that data is consistent across all nodes in the cluster in real time.
- **Global deployments:** Multiple primary nodes distributed across the globe can provide database access wherever applications might be running.

Increased Storage Availability with ActiveCluster

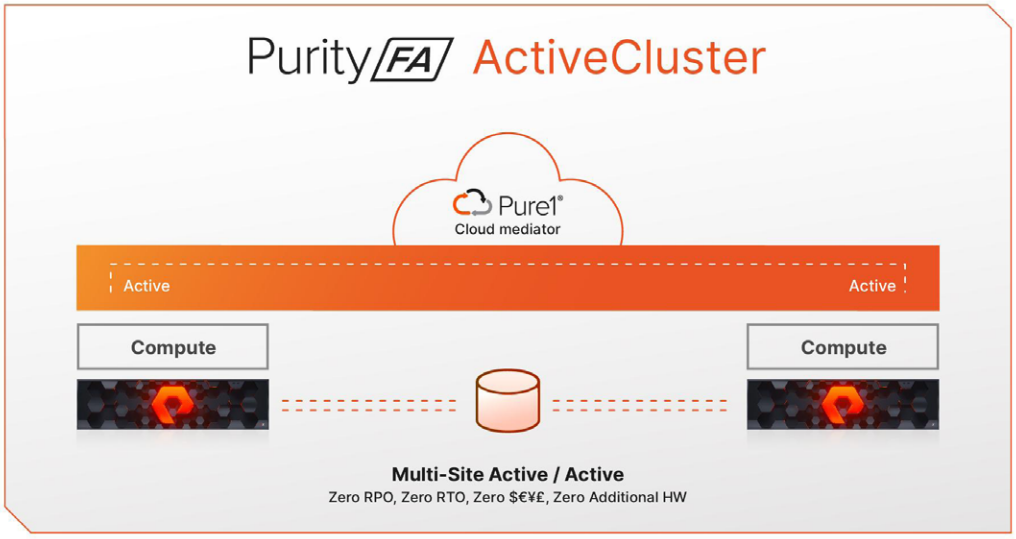

FIGURE 4 ActiveCluster synchronous replication

ActiveCluster provides several features for providing high availability for MySQL workloads:

- **Synchronous replication:** Writes are synchronized between arrays and protected in non-volatile RAM (NVRAM) on both arrays before being acknowledged by 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.
- **Asynchronous replication integration:** Uses asynchronous replication for baseline copies and resynchronizing. Can convert asynchronous relationships to synchronous ones without resending data. Asynchronous data can also be replicated to a third site for disaster recovery.
- **No bolt-ons and no licenses:** No additional hardware required, and 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, such as provisioning storage, connecting hosts, creating snapshots, and creating clones.
- **Integrated Pure1® Cloud Mediator:** This automatically-configured passive mediator allows transparent failover and prevents split brain, without the need to deploy and manage another component.
- **Pure1 Cloud Mediator:** This required solution component determines which array will continue data services should an outage occur in the environment.
- **Active/active clustered FlashArray solutions:** Utilize synchronous replication to maintain a copy of data on each array and present those copies as one consistent copy to hosts that are attached to either array 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 is built on the concept of a pod, a stretched storage container that defines a set of objects (hosts, volumes, or protection groups) that are replicated together, and the arrays they are replicated between. Any FlashArray or Pure Cloud Block Store™ instance can support multiple pods, and any pod that is replicated between them is considered to be "stretched." A pod is also considered a consistency group in which multiple volumes in the same pod are write-order consistent. Pods also provide volume namespaces in which different volumes can have the same name if they are in a different pod, allowing the migration of workloads between arrays or consolidating multiple workloads from many arrays into one without volume-name conflicts.

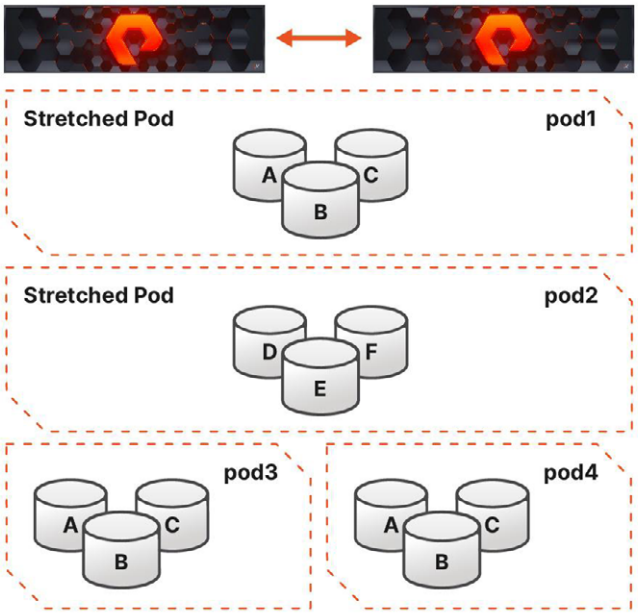

FIGURE 5 Diagram of ActiveCluster pods

Transparent Failover

Transparent failover between arrays when using ActiveCluster is automatic, requiring no intervention from a 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 either a witness or voter to mediate failovers while preventing split brain. ActiveCluster includes Pure1 Cloud Mediator, a simple-to-use, lightweight, and automatic way for applications to failover transparently, or simply move between sites, in the event of a failure and without user intervention.

Pure1 Cloud Mediator

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.

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 virtual machine. Failover behaviors are exactly the same as described above. The on-premises mediator simply replaces the role of 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 virtual machine 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 high availability or Microsoft Hyper-V high-availability 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 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 and Non-uniform Host Access

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

Uniform Host Access

A uniform storage access model can be used in environments where there is host-to-array connectivity of either Fibre Channel (FC) or Ethernet (for iSCSI), and array-to-array Ethernet connectivity, between the two sites (see Figure 6). 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 11ms of round trip time latency between the arrays.

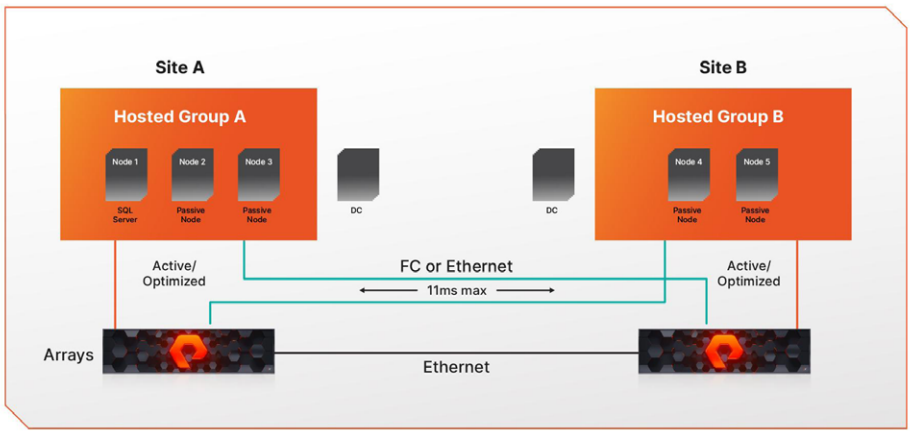

FIGURE 6 . Diagram of uniform host access in ActiveCluster

Figure 6 shows 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 storage 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 the best performance in an active/active synchronous replication environment, hosts should be prevented from using paths that access the remote array unless it is necessary. For example, in the image below, if the host MySQL database were to perform a write to volume A over the host-side connection to array A, that write would incur twice the latency of the inter-site link (one time for each traverse of the network). The write would experience 11ms of latency for the trip from host B to array A, and it would experience another 11ms of latency while array A synchronously sends the write back to array B.

With Purity ActiveCluster, there are no such management headaches. In Figure 7, ActiveCluster does make use of asymmetric logical unit access (ALUA) to expose paths to local hosts as active/optimized paths and expose paths to remote hosts as active/non-optimized. However, there are two advantages in the ActiveCluster implementation:

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

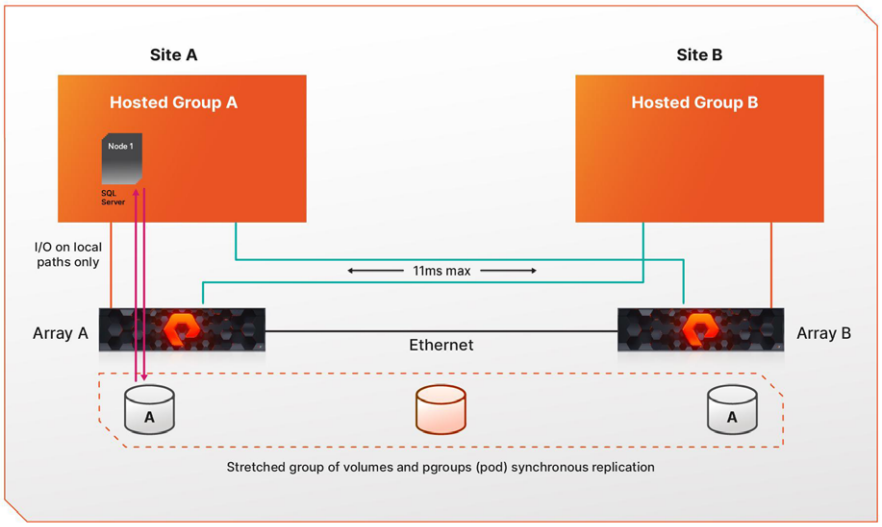

FIGURE 7 . Uniform host access in ActiveCluster between one host and one storage array only

Non-uniform Host Access

A non-uniform storage access model is used in environments where there is host-to-array connectivity of either FC or Ethernet (for iSCSI) only locally within the same site. Ethernet connectivity for the array-to-array replication interconnect must still exist between the two sites. When deployed in this way, each host has access to a volume only through the local array and not through the remote array. The solution supports connecting arrays with up to 11ms of round-trip time latency between the arrays.

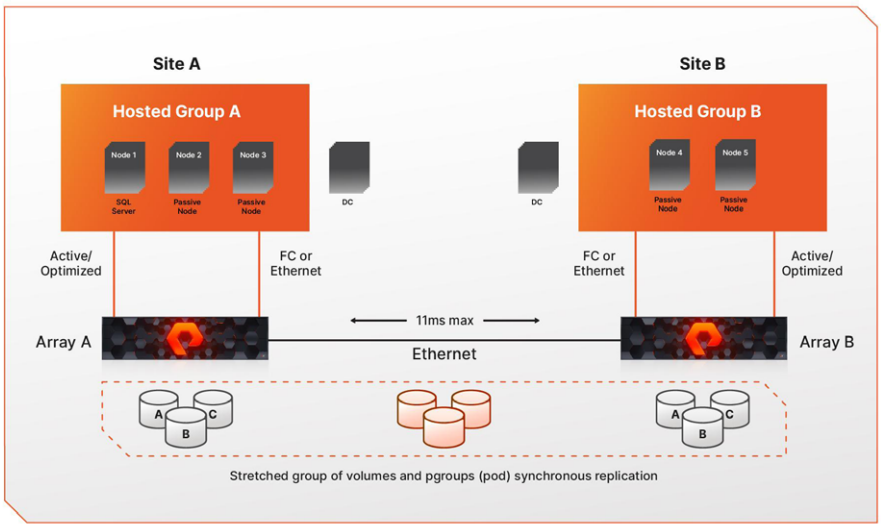

FIGURE 8 . Non-uniform host access in ActiveCluster

Hosts will distribute I/O across all paths to the storage only, because only the local active/optimized paths are available.

Configuring ActiveCluster

A major benefit of using an ActiveCluster stretched storage solution is simplicity. Before moving forward, ensure that the environment configuration requirements match those in this knowledge base article.

ActiveCluster Glossary of Terms

This section uses the following terms:

- **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 FlashArray instances 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. When synchronization completes, all volumes in the pod will be available on both FlashArray instances.
- **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 that was removed.
- **Re-stretching:** 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 in order to allow the pod to be quickly re-stretched without having to resend all the data.

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.

1. Using the command line on one of the arrays, retrieve the connection key:

```
pureuser@Array-A> purearray list --connection-key
Connection Key
d9079431-3bb0-184e-593e-a494059a3684
```

2. On the corresponding array, use the connection key and management IP address to create a sync-replication relationship. The following example uses Ethernet as the replication transport; however, it is also possible to use FC:

```
pureuser@Array-B> purearray connect --management-address Array-A.domain.local --type sync-replication
--connection-key
Enter the connection key of the target array: d9079431-3bb0-184e-593e-a494059a3684
Name           ID                                    Version  Management Address  Replication Address
Status  Throttled  Type                Transport
Array-A  d4eca33c-401c-4c0b-813a-8615590f05a4  6.4.5   10.21.220.69    10.21.220.104           connected
False   sync-replication  ip
```

3. View the connected arrays:

```
pureuser@Array-A> purearray list --connect
Name           ID                                    Version  Management Address  Replication Address
Status  Throttled  Type                Transport
Array-B  f2abf1c1-b1c0-49fd-b6eb-4899c3d1cfc4  6.4.5   -                     10.21.220.65
connected  False       sync-replication  ip
```

Creating a Pod

The second step to perform is to create a pod. If pre-existing database volumes are present, the pod should be created on the array those are local to.

1. Using the command line on the array, create a pod:

```
pureuser@Array-A> purepod create mysql-ha
Name    Source  Array          Status  Frozen At  Promotion Status  Link Count  Quota Limit
mysql-ha  -     Array-A  online  -             promoted       0           -
```

Adding Volumes to a Pod

Pre-existing volumes can be moved into a pod, or new volumes can be created within it.

1. Using the command line on the array, move a volume into the pod:

```
pureuser@Array-A> purevol move DB-01-MySQL mysql-ha
Name                      Size  Source  Created                Serial
mysql-ha::DB-01-MySQL  10752G  -       2023-06-19 05:18:32 PDT  D4ECA33C401C4C0B00016461
```

Stretching a Pod

Stretching a pod performs an initial baseline copy of the volumes from the source to the target array. At the completion of this baseline, the pod will become highly available on both arrays.

1. Using the command line on the array with the pod, add the target array to the pod. The target array will show as offline until the baseline completes.

```
pureuser@Array-A> purepod add --array Array-B mysql-ha
Name    Source  Array          Status   Frozen At  Promotion Status  Link Count  Quota Limit
mysql-ha  -     Array-B  offline  -             promoted       0              -
                Array-A  online   -
```

Viewing a Pod's Status

A pod's status can be seen at any time on either array it is stretched to.

1. Using the command line on any array, view the pod's status:

```
pureuser@Array-A> purepod list mysql-ha
Name    Source  Array          Status  Frozen At  Promotion Status  Link Count  Quota Limit
mysql-ha  -     Array-A  online  -             promoted       0              -
                Array-B  online  -
```

Configuration Information

Once an ActiveCluster configuration has been created and is fully synchronized, the host operating system and various availability preferences can be established using the steps below.

Operating System Configuration

- **Windows:** The configuration considerations for ActiveCluster with Windows can be found in this knowledge base article.
- **Linux:** The configuration considerations for ActiveCluster with Linux can be found in this knowledge base article.
- **IBM AIX:** The configuration considerations for ActiveCluster with AIX can be found in this knowledge base article.
- **Oracle Solaris:** The configuration considerations for ActiveCluster with Solaris can be found in this knowledge base article.

Uniform Configuration

1. Create site A hosts on FlashArray.
2. Present FlashArray storage volumes to each site A host.
3. Create a site A host group.
4. Associate all site A hosts to the host group.
5. Connect site A FlashArray storage volumes.
6. In Windows Server, ensure that storage array volumes are connected at each node located at site A.
7. Online FlashArray storage volumes should be only at the site A primary node where the database will be active.
   Note: Mount volumes only at the site A primary active node.
8. Stretch site A FlashArray storage to site B FlashArray.
   Note: Hosts cannot be connected to the volumes on the secondary array until the stretch is complete and the pod is online at both the primary and secondary arrays.
9. Verify that the pod status is “Online” and the baseline stretch is completed.

To set uniform configuration at primary site B location:

1. Create site B hosts on FlashArray.
2. Present FlashArray storage volumes to each site B host.
3. Create a site B host group.
4. Associate all site B hosts to the host group.
5. Connect site B FlashArray storage volumes.
6. In the operating system, ensure that storage array volumes are connected at each node located at site B.
   Note: Mount volumes only at the site A primary active node.

Preferred Paths

The default behavior is that all paths from a FlashArray to a host will be actively used by the primary active host—even ones from the secondary FlashArray. When replication occurs over extended distances, this is generally not ideal. In situations where the sites are far apart, two performance-impacting scenarios occur:

- Half of the writes (assuming both FlashArray instances 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. Because writes must be acknowledged in both sites, this means that the data traverses the WAN twice. First the host issues a write across the WAN to the remote FlashArray, and then the remote FlashArray forwards it back across the WAN to the local FlashArray. This adds unnecessary latency. The optimal path is for the host to send writes to the local FlashArray, which then forwards it to the remote FlashArray. In the optimal situation, the I/O must only traverse the WAN once.
- Half of the reads (assuming that both FlashArray instances 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 never traverse the WAN in 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 the operating system primary active host which FlashArray should optimally service I/O in the event that an active node can see paths to both FlashArray instances for a given device. This option is a FlashArray host object setting called Preferred Arrays.

In a situation where the FlashArray instances are in geographically different data centers, it is important to set the preferred array for a host on both FlashArray instances.

To set the preferred array to Array A in the command line, use the following command:

```
pureuser@Array-A> purehost setattr –preferred-array Array-A MySQL-Host
Name                    Preferred Array
MySQL-Host  Array-A
```

Note: For every host that has access to both FlashArray instances that host an ActiveCluster volume, set the preferred FlashArray for that host on both instances. Ensure at FlashArray A that it is preferred for host A. Ensure at FlashArray B that FlashArray A is preferred for host A. Doing this on both FlashArray instances allows a host to automatically know which paths are optimized and which ones are not.

Non-uniform Configuration

To set non-uniform configuration at primary site A location:

1. Create site A hosts on FlashArray.
2. Present FlashArray storage volumes to each site A host.
3. Create a site A host group.
4. Associate all site A hosts to the host group.
5. Connect site A FlashArray storage volumes.
6. In Windows Server, ensure that storage array volumes are connected at each node located at site A.
7. Online FlashArray storage volumes should be only at the site A primary node where MySQL will be active.
   Note: Online volumes should be present only at the site A primary active node.
8. Stretch site A FlashArray storage to the site B FlashArray.
   Note: Hosts cannot be connected to the volumes on the secondary array until the stretch is complete and the pod is online at both the primary and secondary arrays.
9. Verify that pod status is “Online” and the baseline stretch is completed.

To set non-uniform configuration at the primary site B location:

1. Create site B hosts.
2. Present FlashArray storage volumes to each site B host.
3. Create a site B host group.
4. Associate all site B hosts to the host group.
5. Connect site B FlashArray storage volumes.
   Note: Mount volumes only at the site A primary active node.

Failure Scenarios

This section examines the availability of storage and network resources in different failure scenarios.

Storage

Table 2 describes different failure scenarios for ActiveCluster environments and how storage availability is affected.

Table 2. Storage availability under failure scenarios

MySQL CI Solution Component Failure				
One Array	**Other Array**	**Replication Link**	**Mediator**	**Access to Storage**
Up	Up	Up	Up	Available on both arrays
Up	Down	Up	Up	Available on one array
Up	Up	Down	Up	Available on one array
Up	Up	Up	Down	Available on both arrays
Up	Down	Down	Up	Available on one array
Up	Up	Down	Down*	Unavailable
Up	Down	Up	Down*	Unavailable
Down	Down	Not applicable (N/A)	N/A	Unavailable

*These rows refer to simultaneous failures of other components while the mediator is unavailable. If the mediator becomes unavailable after an array failure or a replication link failure has already been sustained, access to the mediator is no longer required and access to storage remains available on one array.

Host and Storage Network Failures

Table 3 describes different failure scenarios for ActiveCluster environments, and how host and storage network availability is affected.

Table 3. Storage availability under failure scenarios

Failure Scenario	Failure Behavior
Single-host or Multiple-host Failure	Applications can automatically failover to other hosts in the same site or to other hosts in the other site connected to the other array. This is driven by Windows Server Failover Cluster (WSFC), assuming that clusters are stretched between sites.
Stretched SAN Fabric Outage (FC or iSCSI) (Failure of SAN Interconnect between Sites)	Host I/O automatically continues on local paths in the local site. Uniform connected hosts: • Experience some storage path failures for paths to the remote array, and continue I/O on paths to the local array. • Each site will maintain access to local volumes with no more than a pause in I/O. Non-uniform connected hosts: • Do not have a SAN interconnect between sites, so this scenario is not applicable.
SAN Fabric Outage in One Site	Applications can automatically failover to hosts at the other site connected to the other array. This is driven by host-cluster software, assuming that clusters are stretched between sites: VMware HA, Oracle Real Application Clusters (RAC), SQL Server Clustering, and so on. Uniform connected hosts: • In the site without the SAN outage, experience some storage path failures for paths to the remote array, and continue I/O on paths to the local array • In the site with the SAN outage, will experience total loss of access to volumes, and applications must failover to the other site as mentioned above Non-uniform connected hosts: • In the site without the SAN outage, will maintain access to local volumes • In the site with the SAN outage, will experience total loss of access to volumes, and applications must failover to the other site as mentioned above

Summary

To build a truly resilient business strategy, it is necessary to ensure availability at multiple levels. However, there is no one high-availability solution to use in all situations. Instead, this guide presents some of the different options that are available, in addition to some of their strengths and limitations. Realizing an availability strategy thus becomes a matter of matching the right capabilities to existing business needs.

The capabilities available for and built into FlashArray storage can complement other high-availability solutions. ActiveCluster in particular can help expedite cluster setup by speeding up node seeding times. FlashArray storage can also help a high-availability solution use less storage space.

For more guidance on selecting the right backup solution for a MySQL environment, and to learn more about how FlashArray storage can help improve high availability for MySQL, speak with a Pure Storage sales representative.

PS2464-01-en 07/23