PURESTORAGE®
Uncomplicate Data Storage, Forever

TECHNICAL WHITE PAPER

Disaster Recovery for MySQL with FlashArray

Detailed guidance for choosing a data-protection and disaster-recovery solution for MySQL databases

Contents

Introduction

Disaster-recovery planning is a major challenge for database professionals. One of the first challenges is the growing volume of data and metadata that needs to be managed. Large amounts of data can be difficult to transfer, store, and manage efficiently. Increasingly, large backups themselves can strain resources, even though they can still provide some coverage in case of disaster, compromise, or corruption. Disaster recovery requires the ability to accommodate data volume growth and changing business needs over time. Moreover, storing, maintaining, and managing large amounts of backup data can be costly.

Another challenge is the need to keep data consistent and secure. Ensuring that backup data can be used to accurately recover a database, in case of an error or disaster, is fundamental to protecting business functions. And it's now critical that sensitive backup and stored data is protected, as now criminals are increasingly targeting backups.

Backups can impact database performance, particularly for large databases with frequent backups. Prioritizing data for backup can help, but it can also present its own challenge. Database administrators (DBAs) also need to keep in mind recovery time objectives (RTOs), backup windows, and service level agreements (SLAs). Time is literally money when databases are down for recovery. And even when downtime is scheduled, it still represents a disruption for organizations.

Selecting the Right Disaster-recovery Solution

This guide will help you select the right solution to meet your disaster-recovery (DR) needs for MySQL workloads in conjunction with Pure Storage® FlashArray™ products. While there is certainly some overlap, each workload environment and application architecture described in this guide will have unique backup requirements. These differences should be accounted for in selecting a backup solution for MySQL. Finally, regardless of the backup solution you choose, FlashArray can help increase the deduplication, speed, and consistency of your backups. The following table highlights tools and functions that can be used to implement a DR strategy for MySQL with FlashArray.

Tool	Description	Documentation
Backup Utilities		
Mysqldump	Client-side MySQL tool for creating logical backups	External documentation
mysqlpump	Client-side MySQL tool for creating logical backups	External documentation
MySQL Enterprise Backup	Oracle licensed, enterprise-grade solution for creating physical backups	External documentation
Percona XtraBackup	Storage-side tool for taking point-in-time snapshots of entire storage volumes	External documentation
FlashArray Data Service		
Pure Storage Volume snapshots	Storage-side tool for taking point-in-time snapshots of entire storage volumes	External documentation
Purity ActiveDR™	Advanced data-protection solution designed for efficiency and reliability	External documentation

TABLE 1 Tools and functions that can be used to implement a DR strategy for MySQL with FlashArray

Unified Block and File on FlashArray

FlashArray, a software-defined, unified block- and file-storage solution, offers an effortless and consistent experience, 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. Lastly, FlashArray enables businesses and organizations to drive out direct carbon usage in their data storage systems by up to 80% compared to competitive all-flash systems, and even more against magnetic disks.

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) product with consistent performance at 2–4ms latency for capacity-oriented workloads
- FlashArray//X™: Provides latency as low as 150µs to power critical applications and business operations
- FlashArray//XL™: Provides enterprise-grade performance and scalability for demanding workloads

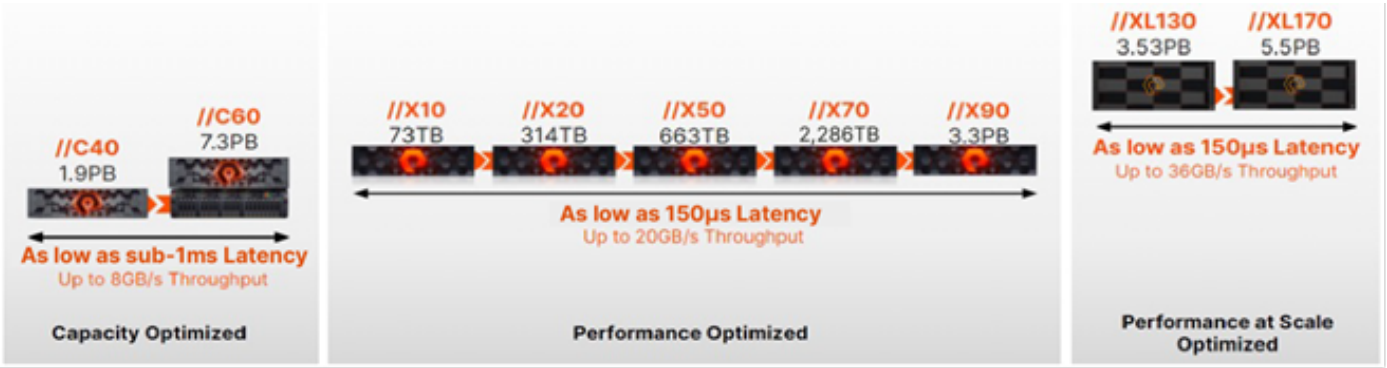

FIGURE 1 Pure Storage FlashArray suite of products

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 SLAs to satisfy your customers—between both physical and virtual machines (VMs).
- **Rich data services**: Data-service functionality, such as ActiveCluster™, ActiveDR, and asynchronous replication, provide increased resilience and availability and enhance applications' workflows.

Protecting MySQL with Backup Utilities

Whether they create logical backups or physical ones, these backup and recovery tools focus specifically on the data in one or more databases on a server rather than all of the data that might be associated with that server.

mysqldump

mysqldump is a client-side utility packaged with MySQL server installation. It reads object definitions and contents from the database and writes them as queries to an output in order to create logical backups. Because of this, migrating or restoring data between different types of MySQL deployments is relatively easy. Moreover, mysqldump can be used in scripts for automated backups.

mysqldump creates a set of SQL statements during a backup that can be executed to reproduce the original database object definitions and table data. It can view and/or edit data in a backup file before restoring and has many filter options to back up only schema structure, data, users, routines, and so forth. The tool can perform backup and restore operations at the instance, schema, and table/object levels. Backup files can then be placed on remote or local hosts.

NOTE: *mysqldump performs backup and restore operations on user-defined databases only.*

Figure 2 provides an architectural overview of the mysqldump process flow when using FlashArray block or file storage as a target.

FIGURE 2 Architectural overview of the mysqldump process flow when using FlashArray block or file storage as a target

How to Use mysqldump

There are several ways to use mysqldump for backup and restore, as detailed in Table 2.

Method	Syntax	Examples
Backup		
Back up all databases	mysqldump -u<username> -p<password> --all-databases > <backup_file_name.extension>	mysqldump -uroot -ppassword --all-databases > allschemas.sql
Back up a single database	mysqldump -u<username> -p<password> --databases <dbname> > <backup_file_name.extension>	mysqldump -uroot -ppassword --databases database1 > database1.sql

Method	Syntax	Examples
Back up objects or tables from a single database	mysqldump -u<username> -p<password> --databases <dbname> --tables <table name> > <backup_file_ name.extension>	mysqldump -uroot -ppassword --databases database1 --tables table1 > db1_tb1.sql
Online backup for a database that has only InnoDB tables	mysqldump --source-data -u<username> -p<password> --databases <dbname> --single-transaction > <backup_ file_name.extension>	mysqldump --source-data -uroot -ppassword --databases database1 --single-transaction > database1.sql
Restore		
Restore all databases	mysql -u<username> -p<password> < <backup_file_ name.extension>	mysql -uroot -ppassword < allschemas.sql
Restore a single database	mysql -u<username> -p<password> --databases <dbname> < <backup_file_name.extension>	mysql -uroot -ppassword --databases database1 < database1.sql
Restore an object or table from single database	mysql -u<username> -p<password> --databases <dbname> < <backup_file_name.extension>	mysql -uroot -ppassword --databases database1 < db1_tb1.sql
Alternate way to restore a database from backup (from system prompt)	mysql -u<username> -p<password> <database name> -e "source/path-to-backup/ backup_file.extension"	mysql -uroot -ppassword dbname -e "source /backup/database1.sql"

TABLE 2 Syntax and examples for backup and restore methods in mysqldump

mysqldump provides many options for customizing the backup process—such as to back up all objects/routines/views, to back up the structure of the database and/or objects, and to back up data only. For a complete list of options and their usage, refer to the mysqldump user guide.

NOTE: *mysqldump requires the database to be temporarily locked during backup, which can impact database performance.*

Performance

To evaluate the performance of mysqldump, a ten (10) gigabyte database was created alongside a target network file system (NFSv3) file share on a FlashArray//C. The contents of the database were output to a single file (database1.sql). The performance achieved for backup was 0.16TB/hr and recovery was 0.01Tb/hr. The database's ability to process the SQL commands for both ingress and egress was found to be the bottleneck when using this utility.

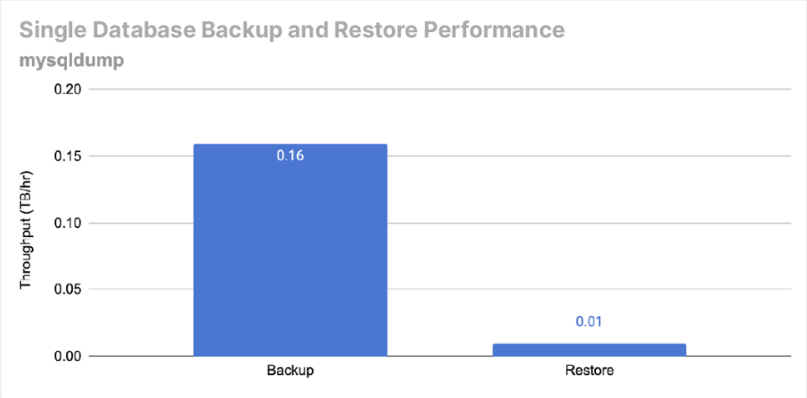

FIGURE 3 mysqldump backup and recovery performance

Best Usage Scenarios

mysqldump is a good fit for environments with a requirement to migrate and recover schemas between multiple systems or servers. It is most appropriately used when backing up and restoring smaller databases (up to 10GB in size).

Also, mysqldump is useful to migrate/export database and object-structure definitions only. It is also useful for populating databases by copying data from one MySQL server to another to export/migrate user accounts between systems.

mysqlpump

mysqlpump is a client-side utility packaged with the typical MySQL server installation (available from version 5.7 and above). It is a utility that reads object definitions and contents from the database and writes them to an output in order to create logical backups, making migrating or restoring data between different MySQL deployments or to other database management systems (DBMSs) relatively easy. Moreover, mysqlpump can be used in scripts for automated backups.

mysqlpump creates a set of SQL statements during backup with which it can restore the original database with all object definitions and data. It has the flexibility to view and/or edit data in a backup file before restoring, and it has many filter options to back up only schema structure, data, users, routines, and so forth. The tool has the ability to perform backup and restore operations at the instance, schema, and table/object levels. Backup files can then be placed on remote or local hosts.

mysqlpump is similar to the mysqldump client with enhanced features such as parallel processing, excluding and including schemas and objects, compression, and a progress indicator for backup process. mysqlpump has the ability to back up system and user databases. Figure 4 provides an architectural overview of the mysqlpump process flow when using FlashArray block or file storage as a target.

FIGURE 4 mysqlpump process flow

How to Use mysqlpump

There are several ways to use mysqlpump for backup and restore, as detailed in Table 3.

Method	Syntax	Examples
Backup		
Back up all databases	mysqlpump -u<username> -p<password> --all-databases > <backup_file_name.extension>	mysqlpump -uroot -ppassword --all-databases > allschemas.sql
Back up a single database	mysqlpump -u<username> -p<password> --databases <dbname> > <backup_file_name.extension>	mysqlpump -uroot -ppassword --databases database1 > database1.sql
Back up objects or tables from a single database	mysqlpump -u<username> -p<password> --databases <dbname> --tables <table name> > <backup_file_name.extension>	mysqlpump -uroot -ppassword --databases database1 --tables table1 > db1_tb1.sql
Online backup for a database that has only InnoDB tables	mysqlpump --source-data -u<username> -p<password> --databases <dbname> --single-transaction > <backup_file_name.extension>	mysqlpump --source-data -uroot -ppassword --databases database1 --single-transaction > database1.sql
Exclusion or inclusion of database objects for backup	mysqlpump -u<username> -p<password> --include-databases=<db1>,<db2> --excludetables=<db1.t1>, <db2.t2> > <backupfilename.sql>	mysqlpump -uroot -ppassword --include-databases=test,test2 --exclude-tables=test.table1,test2.table1 > dbbackup.sql
Parallel processing to improve backup time by performing concurrent schema backups (default parallelism/thread count is two)	mysqlpump -u<username> -p<password> --parallel-schemas=<N:db1,db2> --parallel-schemas=<N:db3> > <backupfile.sql> **Note:** N indicates thread count/ queue size.	mysqlpump -uroot -ppassword --parallel-schemas=4:db1,db2 --parallel-schemas=4:db3 > databasedump.sql **Note:** This example command enables mysqlpump to back up db1 and db2 with four threads in parallel, and another four threads to back up db3 schemas.

Method	Syntax	Examples
Compressed backups	By default, mysqlpump doesn't compress backups. Compression algorithms need to be enabled manually. LZ4 and ZLIB are supported algorithms.	
LZ4 compression	mysqlpump -u<username> -p<password> <dbname> --compress-output=LZ4 ><backupfilename.lz4>	
ZLIB compression	mysqlpump -u<username> -p<password> <dbname> --compress-output=ZLIB ><backupfilename.zlib>	
Restore		
Restore all databases	mysql -u<username> -p<password> < <backup_file_ name.extension>	mysql -uroot -ppassword < allschemas.sql
Restore a single database	mysql -u<username> -p<password> --databases <dbname> < <backup_file_name.extension>	mysql -uroot -ppassword –databases database1 < database1.sql
Restore an object or table from single database	`mysql -u<username> -p<password> --databases <dbname> < <backup_file_name.extension>`	`mysql -uroot -ppassword --databases database1 < db1_tb1.sql`
Restore compressed backups	`To restore compressed backup files, first decompress the backup file to the required format and restore it.`	
Restore from a LZ4 backup file	`lz4_decompress <backupfilename.lz4> <backup.sql>, and then run the restore command.`	
Restore from a ZLIB backup file	`zlib_decompress <backupfilename.zlib> <backup.sql>, and then run the restore command.`	

TABLE 3 Syntax and examples for backup and restore methods in mysqlpump

mysqlpump provides many options for customizing the backup process—like specifying a list of tables to backup, compression, and more. For a complete list of options and their usage, refer to the mysqlpump user guide.

Performance

To evaluate the performance of mysqlpump, four 10GB databases were created alongside a target network file system (NFSv3) file share on a FlashArray//C. The contents of the database were output to a single file (database1.sql). The backup performance was 3.29TB/hr and recovery was 0.02Tb/hr. When using this utility, we determined the database's ability to process the SQL command replay for ingress was the bottleneck.

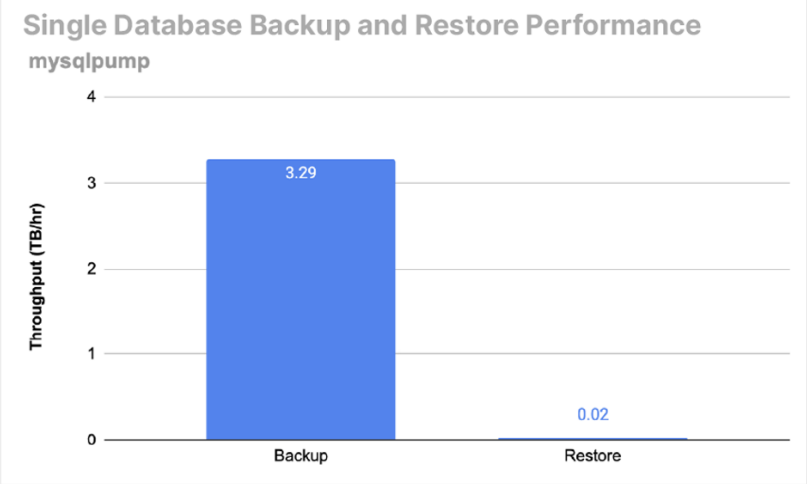

FIGURE 5 mysqlpump backup and recovery performance

Best Usage Scenarios

mysqlpump is a good fit for environments that need to migrate and recover schemas/databases between multiple systems or servers. It is most appropriately used when backing up multiple smaller databases.

mysqlpump can be used as an alternative to mysqldump for environments with a shorter backup window that have datasets larger than 10GB spread across multiple databases/schemas that would benefit from the reduced backup time provided by mysqlpump's parallelism and compression features. However, the restore process is similar to mysqldump in terms of performance.

MySQL Enterprise Backup

MySQL Enterprise Backup is a licensed backup utility provided by Oracle with select commercial editions. It enables DBAs to create hot, online, and non-blocking physical backups of MySQL databases. It includes features like compression and incremental backups to reduce the size of the physical backups and minimize storage requirements. Data backed up with MySQL Enterprise Backup can also be encrypted.

MySQL Enterprise Backup functions are executed through the mysqlbackup client. This client performs different types of backup and restore operations, in addition to other tasks such as backup compression, validation, and parallel processing.

Figure 6 provides an architectural overview of the MySQL Enterprise Backup process flow when using FlashArray block or file storage as a target.

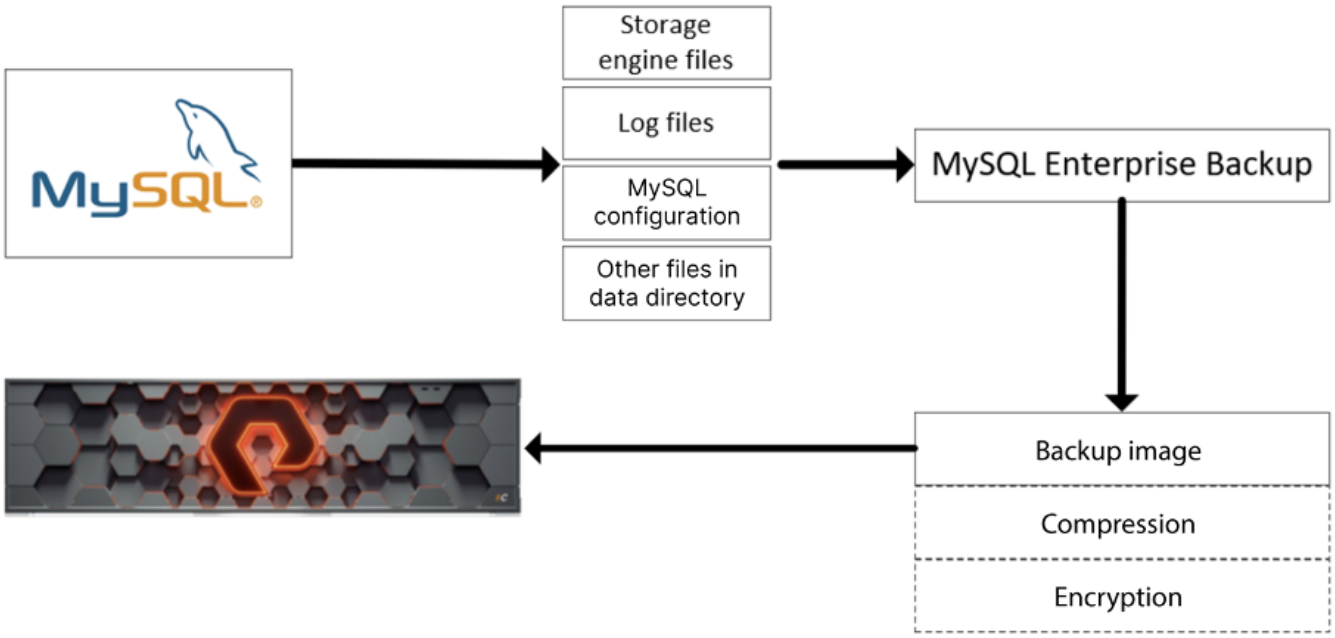

FIGURE 6 MySQL Enterprise Backup process flow

How to Use MySQL Enterprise Backup

There are multiple ways to use the MySQL Enterprise Backup tool for database backup, as detailed in Table 4.

Method	Syntax	Examples
Backup		
Back up all databases	mysqlbackup --user=<username> --password=<password> --backup-dir=<path_to_backup_dir> --backup-image=<backup-image-name> --with-timestamp backup-to-image	mysqlbackup --user=root --password=password --backup-dir=/backup/dbbackup --backup-image=/backup/dbbackup/fulldb_backup.mbi –-with-timestamp backup-to-image

Method	Syntax	Examples
Back up an entire instance by increasing the thread count (parallelism) in order to reduce backup time	mysqlbackup --read-threads=N --process-threads=N --write-threads=N --user=<username> --password=<password> --backup-dir=<path_to_backup_dir> --backup-image=<backup-image-name> --with-timestamp backup-to-image (where N indicates number of parallel threads)	mysqlbackup --read-threads=3 --process-threads=6 --write-threads=3 --user=root --password=password --backup-dir=/backup/dbbackup --backup-image=/backup/dbbackup/fulldb_backup.mbi --with-timestamp backup-to-image
Compression		
By default, mysqlbackup doesn't compress backups; you must enable the compression algorithm manually. mysqlbackup supports LZ4, LZMA, and ZLIB algorithms.	mysqlbackup --compress --compress-method=<algorithm> --user=<username> --password=<password> --backup-dir=<path_to_backup_dir> --backup-image=<backup_image_name> --with-timestamp backup-to-image	mysqlbackup --read-threads=3 --process-threads=6 --write-threads=3 --compress --compress-method=zlib --user=root --password=password --backup-dir=/backup/compress_backup/ --backup-image=/backup/compress_backup/fulldbcompress.mbi --with-timestamp backup-to-image
Incremental Backups		

Prior to incremental backup, ensure a full backup exists to avoid data loss during the restore process. mysqlbackup supports three methods for incremental backups:

- page-track: This is the default method used for incremental backups. This method looks for changed pages in the InnoDB data files that have been modified since the last backup using the page tracking functionality on the server, and it then copies them.
- full-scan: This method scans InnoDB data files in the server's data directory to find pages that have been changed since the last backup, and it then copies them.
- optimistic: Optimistic incremental backups are faster, as they scan for changed pages in the InnoDB data files that have been modified since the last backup and then copy them.

Method	Syntax	Examples
	mysqlbackup --user=<username> --password=<password> --incremental=optimistic --incremental-base=history:[last_backup|last_full_backup] --backup-dir=<path_to_backup_dir> --backup-image=<image_name> backup-to-image	mysqlbackup --user=root --password=password --incremental=optimistic --incremental-base=history:last_backup --backup-dir=/backup/dbbackup --backup-image=incremental_image1.bi backup-to-image
Restore methods	Before restoring the backup file, the integrity of the backup data must be ensured. Execute the "validate" command to verify whether the backup file is corrupted, truncated, or damaged. This operation validates the checksum value for each data page in a backup.	
Validate command	mysqlbackup -u<username> -p<password> --backup-image=<image_name> validate	mysqlbackup -uroot -ppassword --backup-image=/backup/dbbackup/fulldb_backup.mbi validate
Pre-recovery/restore steps (for Linux systems)	Stop mysql services: systemctl stop mysqld Ensure data directories are empty on the system where data restore is planned; if not, delete all files in the data directory rm -rf <data_dir_path>/*	
Restore instance from full backup Note: In general, using parallelism improves restore time.	mysqlbackup -u<username> -p<password> --datadir=<data_dir_path> --backup-image=<backup_image_path> --backup-dir=<temp_dir_path> copy-back-and-apply-log	Without parallelism: mysqlbackup -uroot -ppassword --datadir=/var/lib/mysql --backup-image=/backup/dbbackup//fulldb8.mbi --backup-dir=/backup/temp copyback-and-apply-log
		With parallelism: mysqlbackup -uroot -ppassword --read-threads=3 --process-threads=6 --write-threads=3 --datadir=/var/lib/mysql --backup-image=/backup/dbbackup//fulldb8.mbi --backup-dir=/backup/temp copy-back-and-apply-log

Method	Syntax	Examples
Restore an incremental backup	To restore an incremental backup, ensure that the last full backup is restored and then make it up-to-date to the time you use the restore process described above. Finally, restore the incremental backup image on top of the full backup that you have just restored.	
	mysqlbackup --user=<username> --password=<password> --backup-image=<inc_image_ name> --backup-dir=<incBackupTmpDir> --datadir=<restoreDir> --incremental copy-back-and- apply-log	mysqlbackup --user=root --password=password --backup-image=incremental_image1.bi --backup- dir=/backup/dbbackup/inc_temp --datadir=/var/lib/mysql --incremental copy-back-and-apply-log
Post-recovery/restore steps (for Linux systems)	After restore completion, perform the following tasks as a root user to bring the database instance online: Change ownership for all files/directories in the data directory to the mysql user. chown -R mysql:mysql <data_dir_path>/*. (Ex: chown -R mysql:mysql /var/lib/mysql/*) Start the MySQL service: systemctl start mysqld	

TABLE 4 Syntax and examples for backup and restore methods in MySQL Enterprise Backup

MySQL Enterprise Backup provides options for customizing the backup process, including options for data compression, handling binary log files, and more. For specifics, refer to the MySQL Enterprise Backup documentation for a complete list of options and usage.

Performance

To evaluate the performance of MySQL Enterprise Backup, a one (1) terabyte database was created alongside a target network file system (NFSv3) file share on FlashArray//C. Different performance for backup and recovery was achieved when varying the number of threads were utilized. When using the default thread configuration (with eight threads) backup performance was 1.51TB/hr and recovery performance was 0.95TB/hr. Increasing the number of threads to 12 more than doubled the performance, with backup achieving 2.88TB/hr and recovery achieving 2.53TB/hr.

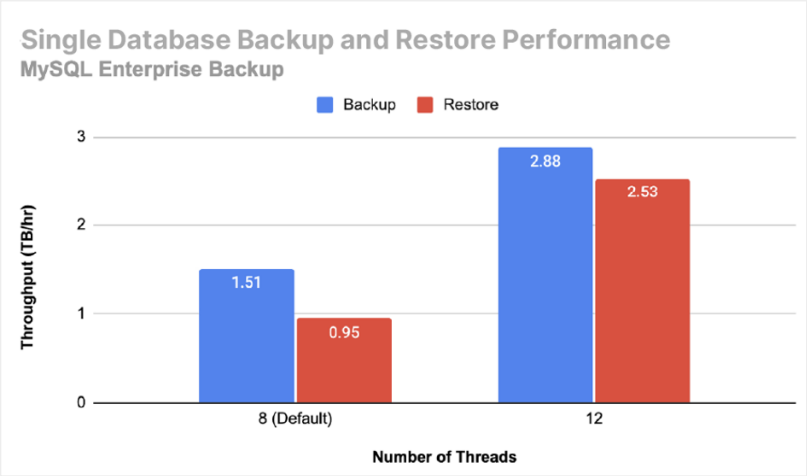

FIGURE 7 MySQL Enterprise backup and recovery performance at different thread counts

Best Usage Scenarios

MySQL Enterprise Backup is appropriate for any environment as it includes several features to provide better performance, security, and resilience.

Because MySQL Enterprise Backup features multiple platform support, it can be used in MySQL deployments that run on different operating systems.

Percona XtraBackup

Percona XtraBackup enables DBAs to create hot, online, non-blocking, tightly compressed, and highly secured backups of MySQL databases that facilitate faster DR processes. Incremental backups reduce the size of the physical backups and minimize storage requirements.

XtraBackup is an open-source utility for MySQL available from Percona. Percona XtraBackup provides compressed and incremental backups that reduce the time and size of physical backups and that minimize storage requirements. Data backed up with Percona XtraBackup can be compressed or encrypted. This tool supports multiple storage engines including InnoDB, Percona XtraDB, MyISAM, and MyRocks.

Backup and restore functions in Percona XtraBackup are executed with the XtraBackup client. By default, backup and restore operations are a single-thread process.

Figure 8 provides an architectural overview of the XtraBackup process flow when using FlashArray block or file storage as a target.

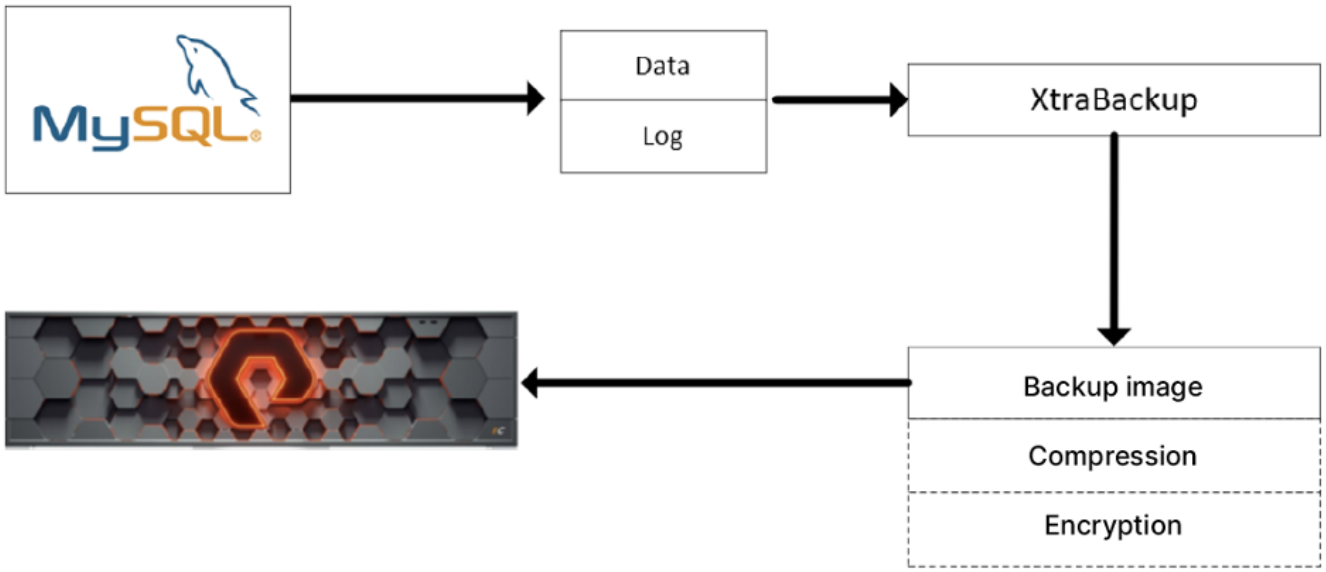

FIGURE 8 Percona XtraBackup process flow

How to Use Percona XtraBackup

Percona XtraBackup is a third-party backup tool for MySQL and can be installed either from a package manager or by downloading the software directly from the Percona website. It is a set of distinct utilities with the XtraBackup utility being the one used for backup and recovery.

Method	Syntax	Examples
Backup		
Back up an entire instance	xtrabackup -u<username> -p<password> --backup --target-dir=<backup_dir_path>	xtrabackup -uroot -ppassword --backup --target-dir=/backup/xbp/
Back up an entire instance using parallelism (as increasing the number of parallel threads reduces backup time)	xtrabackup --parallel=N -u<username> -p<password> --backup --target-dir=<backup_dir_path> (where N indicates the number of parallel threads)	xtrabackup --parallel=8 -uroot -ppassword --backup --target-dir=/backup/xbp/
Compressed backups	By default, XtraBackup doesn't compress backups; you must enable the compression algorithm manually. Percona XtraBackup supports quicklz (default), lz4, and zstd algorithms.	

Method	Syntax	Examples
Compressed backup (with parallelism)	xtrabackup --parallel=N -u<username> -p<password> --backup --compress --compress-threads=N/2 --target-dir=<backup_dir_path>	xtrabackup -uroot -ppassword --parallel=8 --backup --compress --compress-threads=4 --target-dir=/backup/xbp
Uncompressed backup (with parallelism)	Backup files to be uncompressed before executing the prepare command: xtrabackup --parallel=N --decompress --target-dir=<backup_dir_path>	xtrabackup --parallel=8 --decompress --target-dir=/backup/xbp/
Incremental backups	Percona XtraBackup can copy only the data that has changed since the last full backup. Changed data will be stored in files with a .delta extension. Requirements: Prior to incremental backup, ensure a full backup exists to avoid data loss during the recovery/restore process. Create a separate directory/folder/path to store incremental backups. During full backup, XtraBackup records/writes the last LSN (to_lsn) to a file called "xtrabackup_checkpoints." Incremental backup uses the value of "to_lsn" as a starting point.	
	xtrabackup --backup --target-dir=<inc_backup_dir_path> --incremental-basedir=<backup_dir_path> Note: The incremental backup directory and the incremental base directory (that is, the full-backup directory) should be different.	xtrabackup --backup --target-dir=/backup/xbp_inc/ --incremental-basedir=/backup/xbp/

TABLE 5 Syntax and examples for backup and restore methods in Percona XtraBackup

Prepare Incremental Backups	Syntax	Examples
First, prepare the base backup before applying an incremental backup.	`xtrabackup --prepare --apply-log-only --target-dir=<backup_dir_path>`	`xtrabackup --prepare --apply-log-only --target-dir=/backup/xbp/`
Next, apply an incremental backup to the base full backup.	xtrabackup --prepare --apply-log-only --target-dir=<backup_dir_path> --incremental-dir=<inc_backup_dir_path>	xtrabackup --prepare --apply-log-only --target-dir=/backup/xbp --incremental-dir=/backup/xbp_inc/
	Note: The XtraBackup restore process is the same for uncompressed, compressed, and incremental backups. The XtraBackup preparation process differentiates and makes backup files ready to restore using the same approach.	
Restore	Before restoring a backup file, it needs to be prepared; data files are not point-in-time consistent because they were copied at different times when the program ran, and they might have been changed while this was happening.	
Prepare command	xtrabackup –prepare --target-dir=<backup_dir_path>	xtrabackup --prepare --target-dir=/backup/xbp/
Pre-recovery and restore steps (for Linux systems)	Stop MySQL services: systemctl stop mysqld Ensure data directories are empty on the system where the data restore is planned; if not, delete all files in the data directory: rm -rf <data_dir_path>/* To restore an instance from full backup (using parallelism): xtrabackup --parallel=N --copy-back --target-dir=<backup_dir_path> --datadir=<data_dir_path> (where N indicates the number of parallel threads)	xtrabackup --parallel=8 --copy-back --target-dir=/backup/xbp/ --datadir=/var/lib/mysql/

Prepare Incremental Backups	Syntax	Examples
Post-recovery and restore steps (for Linux systems)	After completion of a restore, perform the following tasks as a root user in order to bring the database instance online: Change ownership for all files and directories in the data directory to the MySQL user: chown -R mysql:mysql <data_dir_path>/* Start the MySQL service: systemctl start mysqld	chown -R mysql:mysql /var/lib/mysql/*

TABLE 6 Preparing incremental backups with Percona XtraBackup

Percona XtraBackup provides many options for customizing the backup process including point-in-time recovery, compression, automatic backup verification, and more. For specifics, refer to the Percona XtraBackup documentation for a complete list of options and usage.

Performance

To evaluate the performance of XtraBackup, a 1TB database was created alongside a target network file system (NFSv3) file share on FlashArray//C. Different performance for backup and recovery was achieved when varying the number of threads utilized. When using the default thread configuration (1), backup performance was 2.29TB/hr and recovery performance was 0.74TB/hr. Uppingthe number of threads to eight increased the performance with backup achieving 3.90TB/hr and recovery achieving 4.15TB/hr.

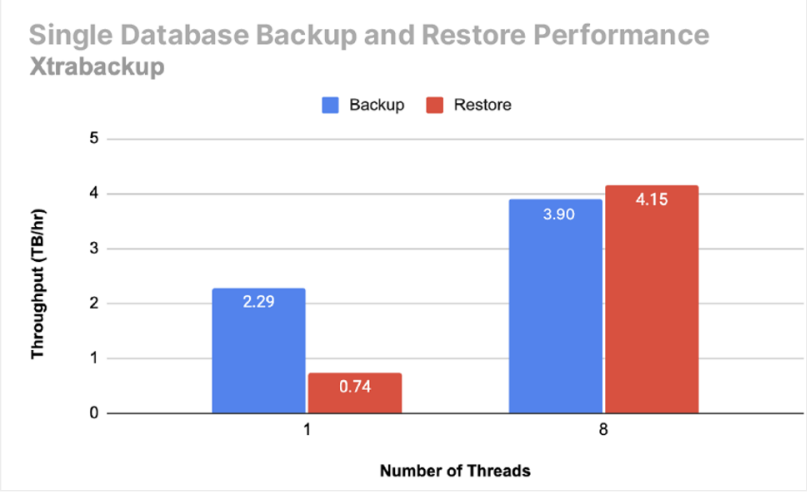

FIGURE 9 Percona XtraBackup backup and recovery performance at different thread counts

When utilizing compression options in XtraBackup, performance increased slightly, achieving 4.43TB/hr for backup and 4.36TB/hr for recovery.

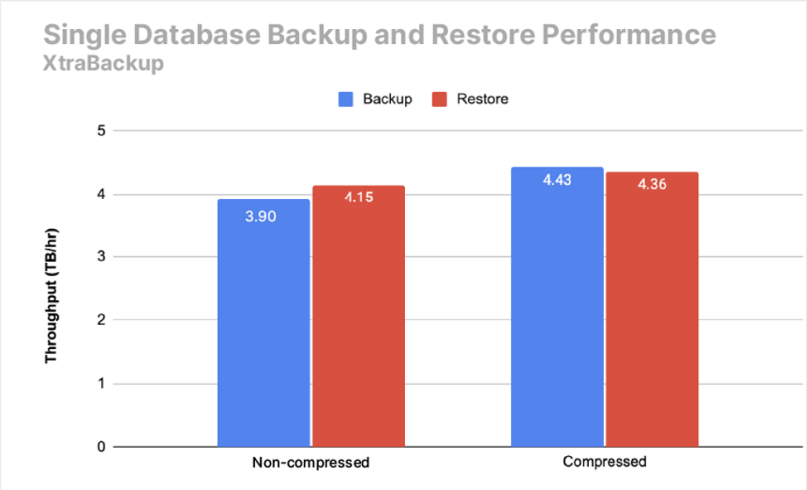

Figure 10. Percona Xtrabackup backup and recovery performance with and without compression

Best Usage Scenarios

Percona XtraBackup is a fast, open-source backup tool for MySQL DR that fits any environment. It gives excellent performance with additional functionalities to satisfy many business rules. It also supports incremental/point-in-time recovery strategies to help ensure no data loss.

Because Percona XtraBackup features multiple platform support, it can be used in MySQL deployments that run on different operating systems.

NOTE: *Percona XtraBackup works with FlashArray File only when using Purity 6.4.2 and later.*

Protecting MySQL Using FlashArray Data Services

FlashArray provides a number of data services that can be used to protect or provide an additional layer of resilience for MySQL databases. Volume snapshots and ActiveDR™ are two such data services that can be used to implement a comprehensive DR strategy when utilizing FlashArray as the storage for any MySQL database.

Pure Storage Volume Snapshots and Asynchronous Replication

Pure Storage volume snapshots provide a quick and easy means of capturing the state of a MySQL database. These snapshots are immutable point-in-time images of one or more block-storage volumes. Snapshots of storage volumes provide access to files, applications, and operating systems as they existed at the time the snapshot was taken. These volume snapshots enable quick restoration of data in the event of corruption, failure, or data loss, without having to rely on an offsite backup.

FlashArray snapshots provide the following key features:

- **Efficiency**: FlashArray snapshots are thin-provisioned, deduplicated, compressed, and don't require any snapshot capacity reservation. Snapshots also inherit all data-reduction capabilities to reduce snapshot size.
- **No snapshot hierarchy**: FlashArray snapshots do not rely on other snapshots for data consistency, which lets administrators create and destroy snapshots without affecting other snapshots.
- **No performance degradation**: FlashArray snapshots are full volumes that are created instantaneously without any performance degradation.
- **Portability**: FlashArray snapshots can be transferred to one or more FlashArray or Pure Cloud Block Store™ instances using space-efficient copies accompanied by volume metadata.
- **Rapid recovery**: Administrators can recover data from any snapshot. The data can be recovered directly to the original volume the snapshot was created from, or it can be recovered to a new volume.

FlashArray supports asynchronous replication of volume snapshots between FlashArray devices. With asynchronous transfers, only data that does not exist on the target FlashArray is transferred. This replication works well for applications that require a recovery point objective (RPO) as low as five minutes, or for data-mobility scenarios that require data to be copied to multiple sites.

Volume snapshots can be taken manually or scheduled with automated policies. They can be replicated across physical locations for additional DR in the event of a site-wide disaster. Moreover, they do not take up any extra space when created; snapshots create a reference point for where the data was at a given moment in time. This arrangement significantly reduces storage overhead. Volume snapshots can then be used to create new volumes as duplicates or restored to existing volumes.

Figure 11 provides an architectural overview of volume snapshots and asynchronous replication.

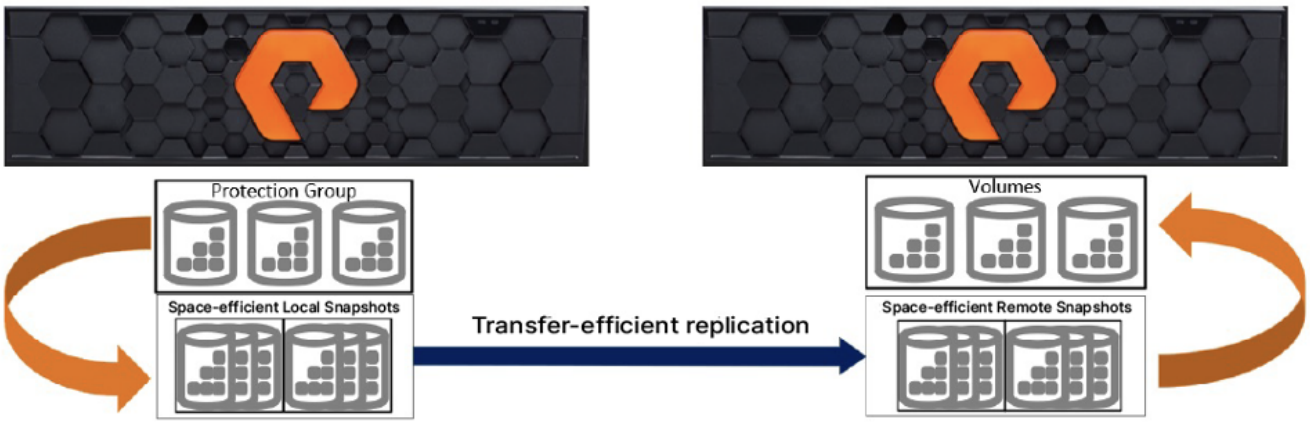

Figure 11. Volume snapshots and asynchronous replication process flow

How to Use

1. **Configuration**: You should ensure that the FlashArray storage is properly configured, including network settings, security settings, and access controls.
1. **Volume creation:** To create a volume on the FlashArray storage, use the Pure Storage management interface, such as the Pure1® management interface or the FlashArray storage's REST API, to define the size, performance, and storage characteristics of the volume.
2. **Volume mapping:** To map the volume to a host, use the Pure Storage management interface to create a host connection to the volume.
3. **Snapshot creation:** Use the Pure Storage management interface to initiate a snapshot. The snapshot can be set to occur at a specific time or frequency, or it can be taken on demand. For details, see "Creating Volume Snapshots for MySQL on FlashArray" on the Pure Storage support site.
4. **Restore from snapshot:** To restore from a snapshot, use the Pure Storage management interface to choose the desired snapshot and initiate a restore. The restore process creates a new volume from the snapshot, which can then be mounted and used as the primary storage volume. For details, see "Recovering MySQL From FlashArray Volume Snapshots" on the Pure Storage support site.
5. **Snapshot scheduling and monitoring:** Monitor the usage of snapshots and manage the snapshot schedule and retention policy to ensure that adequate snapshots are being taken and that older snapshots are being deleted to conserve storage space.

Outcomes for Administrators

Pure Storage volume snapshots enable DBAs to create point-in-time backups of databases, which provide a quick and reliable data-recovery solution. Volume snapshots also reduce downtime and can scale to handle large databases. Volume snapshots consume minimal storage space, which helps reduce costs. And because they are snapshots of data volumes at a specific point in time, they make recovery fast. Volume snapshots can be used in a variety of scenarios, such as DR, testing and development, and replica cloning (MySQL replication, group replication, and Galera Cluster for MySQL).

ActiveDR

ActiveDR provides continuously active replication capabilities for block storage volumes from one FlashArray to another with no impact on front-end application performance.

Synchronous replication technologies like ActiveCluster™ provide higher availability at the cost of requiring network latency between the two targets to be as low as possible. ActiveDR does not require the target system to acknowledge that data has been received before allowing the source to continue, thus eliminating the need for a low-latency network and allowing for continuous replication over larger distances.

Managing ActiveDR is simple, making it suitable for implementation in any scenario requiring a near-zero recovery point objective (RPO). Some management aspects of ActiveDR include:

- **POD replication**: Pods are management containers for volumes using ActiveCluster or ActiveDR. They provide a simple management construct to organize data volumes and associated settings into groups. Once pods are linked together on separate systems via a replica link, data in that pod automatically starts replicating.
- **Continuous change tracking**: Tracking enables you to automatically manage changes without the need to provision or monitor journal devices.
- **Single-command failover**: ActiveDR makes it simple to implement, test, and manage DR. This true DR testing ensures that any runbook or orchestration steps for the entire environment stay the same during a test or in an actual failover event to minimize risk.
- **Multi-direction replication**: Configure multiple pods in different directions between two FlashArray systems.

Figure 12 provides an architectural overview of volume snapshots and asynchronous replication.

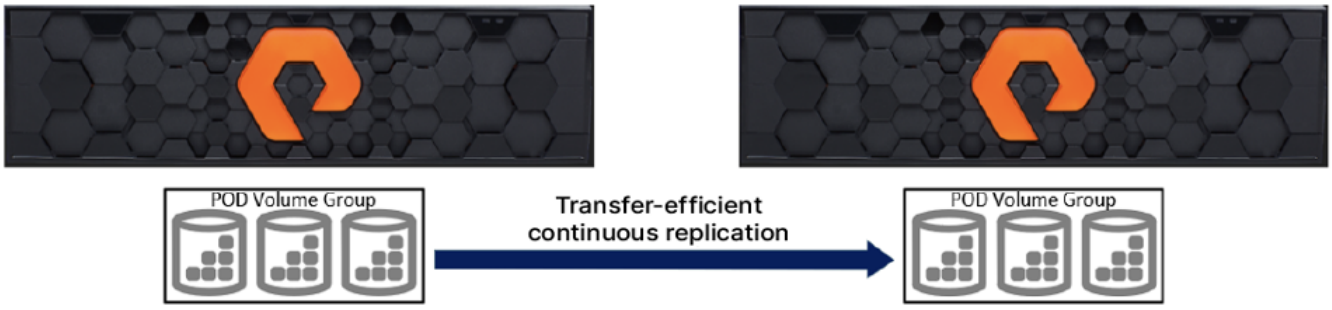

Figure 12. ActiveDR continuous replication process flow

Combining ActiveDR and Asynchronous Replication for Multiple RPOs

ActiveDR and asynchronous replication can be combined to provide the best of both scenarios, where near-zero and five-minute RPOs can be achieved simultaneously. This is achieved by adding a protection group with a snapshot-and-replicate schedule to an existing ActiveDR pod.

How to Use Active DR

1. **Purity ActiveDR configuration**: Determine the configuration of the Purity ActiveDR system, including the number of nodes, network configuration, storage configuration, and access controls.
2. **Purity installation**: Install Purity on the primary and secondary storage nodes according to the manufacturer's instructions.
3. **Primary storage-node configuration**: Configure the primary storage node, including creating volumes and mapping them to hosts as well as configuring the replication settings for Purity ActiveDR.
4. **Secondary storage-node configuration**: Configure the secondary storage node, including creating volumes and mapping them to hosts and configuring the replication settings for Purity ActiveDR.
5. **Replication settings configuration**: Configure the replication settings for Purity ActiveDR, including the frequency of snapshots and the amount of data that will be replicated between the primary and secondary storage nodes.
6. **Configuration testing**: Test the configuration of the Purity ActiveDR system, including testing the replication and failover procedures, to ensure that the system is functioning as expected.
7. **Purity ActiveDR system monitoring**: Monitor the Purity ActiveDR system to ensure that the replication is functioning correctly and that the failover procedures are working as expected. Also monitor the performance of the system and adjust the configuration as necessary to optimize performance.

NOTE: *It is important to note that while Purity ActiveDR can provide real-time replication and failover capabilities, you should carefully plan and test the system before using it in a production environment. You should also be familiar with DR procedures and should plan for the possibility of data loss or corruption in the event of a failure.*

Outcomes for Administrators

Pure Storage Purity ActiveDR helps minimize downtime during failover, which helps ensure that the database remains available for users. It also synchronizes your data between the primary and secondary site in order to provide a seamless failover experience.

Purity ActiveDR supports flexible deployment options, including hybrid, multicloud, and on-premises. It simplifies management by providing a centralized console that streamlines the DR process and reduces DBA workloads.

Purity ActiveDR is cost-effective and can help reduce the expenses associated with traditional DR methods. Moreover, it can be integrated with other backup and recovery tools, providing you with a comprehensive DR solution.

Summary

There is no one database backup solution for every situation. Your individual circumstances will drive specific backup needs for your MySQL environment and your organization. With the right information and guidance, you don't have to guess which solution is right for you.

Backup utilities like mysqldump and mysqlpump provide logical backups that are ideal for DR for smaller databases while utilities such as MySQL Enterprise Backup and Percona XtraBackup supply features, such as compression and incremental backups, to reduce the size of the physical backups and minimize storage requirements for DR for larger databases. And data services like volume snapshots and ActiveDR can be used to implement a comprehensive DR strategy when using FlashArray as the storage for any MySQL database.

The capabilities available for and built into Pure Storage FlashArray storage can complement other backup solutions to help further increase the speed and consistency of your backups while also using less storage. For more guidance on selecting the right backup solution for your MySQL environment, speak with your Pure Storage sales representative.

PS2422-01-en 04/23