REFERENCE ARCHITECTURE

Optimizing SQL Server Operations and Scale with Pure Storage

A Reference Architecture for Microsoft SQL Server with Pure Storage

Contents

Executive Summary

This reference architecture covers the use of Pure Storage® products in Microsoft SQL Server environments. It offers solution overviews, technical configurations, and best practice recommendations for use cases such as database copy optimization, data protection, high availability, and near-synchronous replication. These solutions utilize Pure Storage FlashArray™ for primary block storage in the SQL Server landscape, Pure Storage Cloud Dedicated block storage for storage in public cloud environments, and FlashBlade® products for T-SQL backup solutions with data virtualization using PolyBase.

SQL Server environments must be properly architected to ensure the optimal database performance, availability, and security needed for business continuity. This means all layers, from system hardware through the software stack, must be correctly architected for a variety of environments. A well-architected database environment with the correct storage configuration can help ensure that business operations run smoothly; it can also facilitate rapid troubleshooting in the event of any issues.

The solution overview provided in this document outlines the interconnected components of a typical SQL Server environment, including SQL Server instances, FlashArray, Pure Storage Cloud Dedicated, and FlashBlade storage, and it summarizes the benefits of Pure Storage solutions, including high performance, seamless scalability, nondisruptive upgrades, data integrity, and continuous availability.

The technical solution design section provides technical guidance and best practice recommendations for compute and network resources, storage systems, and SQL Server configurations. The architectural use cases provided in this document feature common scenarios that highlight using FlashArray snapshots to create database copies for development and testing, using volume snapshots on FlashArray or Pure Storage Cloud Dedicated for data protection, ActiveCluster™ for high availability databases, and ActiveDR™ for near-synchronous replication and transparent failover.

Introduction

SQL Server is a robust relational database management system developed by Microsoft and widely used for structured data storage, processing, and business intelligence applications. When paired with Pure Storage, an all-flash storage solution provider, SQL Server enables high performance, low latency, and efficient management of database workloads at any scale.

This reference architecture focuses on the design and implementation of Pure Storage solutions in SQL Server environments. It offers an overview of technologies and solution designs for use cases such as disaster recovery, high availability, data protection, and database copy.

These solution designs leverage Pure Storage technologies, including FlashArray, which serves both as primary storage and as a backup repository, and Pure Storage Cloud Dedicated, which provides storage capabilities in public cloud environments. Additionally, FlashBlade supports T-SQL native backups and can be used with PolyBase for data virtualization with object storage.

Solution Overview

This reference architecture and the associated solutions within this document cover a common SQL Server environment of multiple interconnected components, including:

- SQL Server instances running on Windows Server or a supported Linux platform
- FlashArray used for primary storage via block storage protocols such as Fibre Channel Protocol, iSCSI, or NVM Express over Fabrics with Linux
- FlashArray used as a file- or block-based backup repository for SQL Server T-SQL backups
- Pure Storage Cloud Dedicated for primary storage of SQL Server instances in Microsoft Azure or Amazon Web Services
- FlashBlade used as a file- or object-backup repository for SQL Server T-SQL backups and storage for virtualized data from parquet files in an object store access through PolyBase

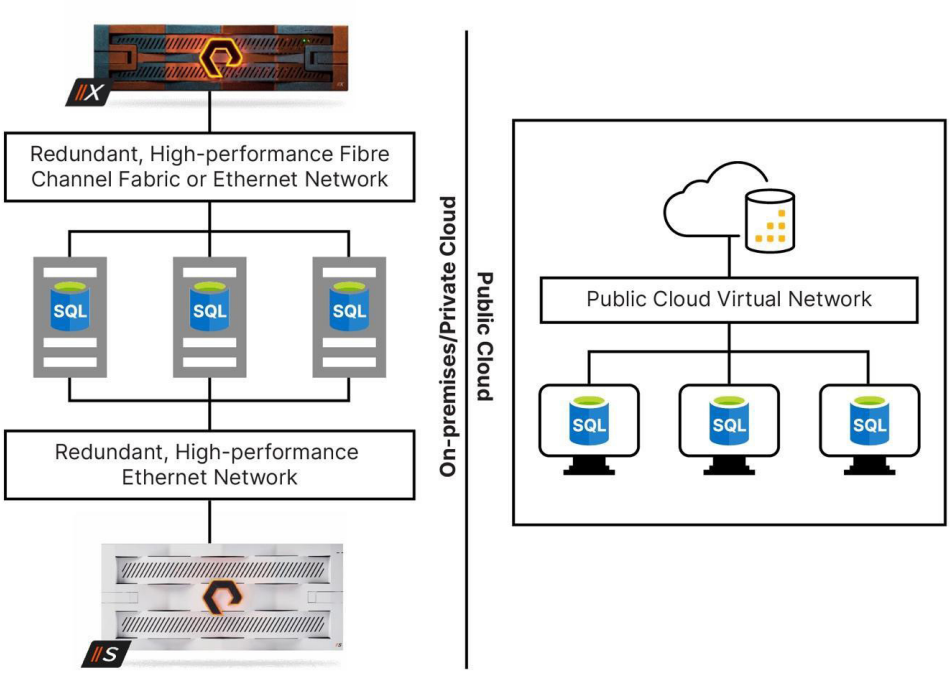

FIGURE 1 High-level overview of the interconnected components and environments showing FlashArray used as primary storage and FlashBlade used as a target for T-SQL backups

Solution Benefits

Pure Storage solutions deliver multiple benefits to SQL Server environments, including:

- **High performance**: Pure Storage all-flash arrays deliver low latency and high throughput, significantly improving SQL Server performance. Faster query execution and higher input/output operations per second enable better handling of transactional and analytical workloads, boosting overall efficiency.
- **Scalability**: The Pure Storage architecture allows SQL Server environments to scale seamlessly as demand grows, with nondisruptive capacity expansion ensuring that growth can happen without impacting performance or requiring downtime.
- **Nondisruptive everything**: Pure Storage enables zero-downtime maintenance, upgrades, and migrations for storage. Its nondisruptive software and hardware upgrades allow for continuous operations, minimizing service interruptions in SQL Server environments.
- **High availability**: With technologies like ActiveCluster, SQL Server deployments benefit from multisite synchronous replication, 99.9999% availability, and integration with Always On Availability Groups. This ensures near-zero downtime and reliable failover.

- **Disaster recovery:** Pure Storage provides robust disaster recovery solutions through snapshot-based recovery, asynchronous and synchronous replication, and SafeMode™ Snapshots that protect against ransomware. These features ensure rapid recovery and minimal data loss.
- **Consolidation:** Pure Storage allows for the consolidation of multiple SQL Server instances and databases on a single storage platform. This reduces hardware sprawl, simplifies management, and lowers operational costs while maintaining high performance through data reduction technologies like compression and deduplication.
- **Security and compliance:** Pure Storage helps ensure data security with in-flight and at-rest encryption, immutable snapshots for protection against ransomware, and enhanced compliance through detailed auditing and logging capabilities, which are critical in regulated industries.
- **Simplified management and automation:** The Pure1® management platform offers a centralized view of storage with predictive analytics and AI-driven insights. Pure Storage API integrations enable automation, simplifying SQL Server provisioning and management, while offering full-stack visibility.
- **Cost efficiency:** Pure Storage delivers lower total cost of ownership through data reduction, efficient use of storage, and reduced power consumption. The ability to consolidate workloads and reduce backup and restore times contributes to greater overall savings and improved return on investment.

Technology Overview

This reference architecture consists of a common environment with a set of interconnected technology components that communicate with one another over redundant high-speed Ethernet connections or a Fibre Channel fabric.

User databases are created within SQL Server instances with block storage provisioned from FlashArray as the primary storage for user database data files. File or object storage from FlashBlade serves as a target for T-SQL backups and storage for data virtualization via PolyBase. FlashArray can also be used as a target for T-SQL backups. No distinction is made between the various editions of SQL Server in this reference architecture.

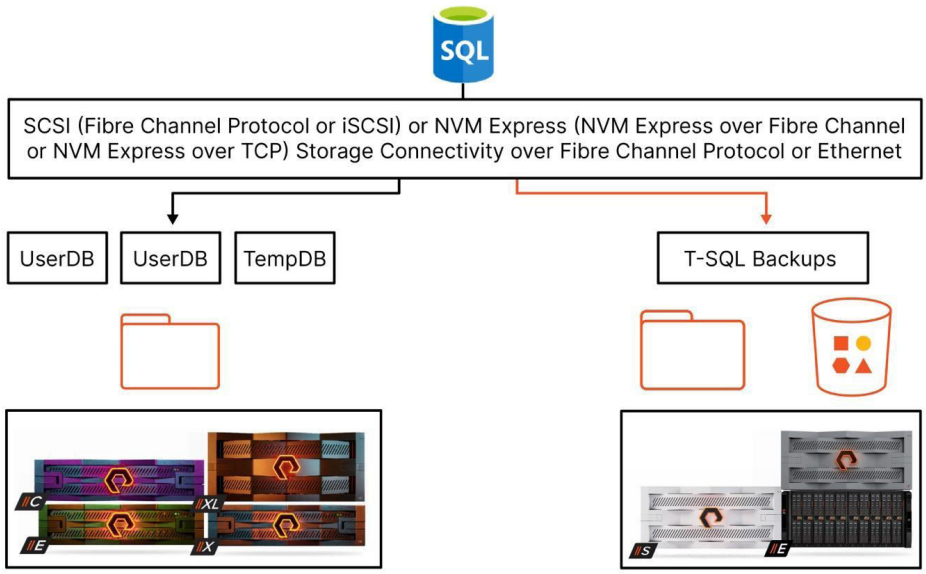

FIGURE 2 Technology components and network protocols for FlashArray in a SQL Server landscape

Compute Resources

The solutions within this document apply to physical or virtual multicore servers with Windows or supported Linux operating systems. These are referred to in this document as either physical or virtual hosts.

SQL Server requires x64 processor hardware with high processor speeds. High memory capacity is necessary to avoid disk input/output traffic bottlenecks that can delay query response times and impact memory capacities.

Processor:

- x64 processors supported include AMD Opteron processors, AMD Athlon 64 processors, Intel Xeon processors with Intel Extended Memory 64 Technology (Intel EM64T) support, and Intel Pentium 4 processors with Intel EM64T support.
- Minimum processor speed is 1.4GHz; the recommended speed is 2.0GHz or faster.

Memory:

- The minimum memory capacity is 1GB; the recommended capacity is 4GB or higher.

For detailed installation requirements, see Microsoft's SQL Server 2022: Hardware and software requirements page.

Network Resources

The network considerations for these solutions are the same as for any enterprise IT infrastructure solution: availability, performance, and extensibility. The compute resources in the solution can attach to any compatible TCP/IPv4 or TCP/IPv6 network infrastructure with the general recommendation that minimum network speeds are capable of 25 gigabit Ethernet. For this solution, the network should be configured using dual switches to eliminate a single point of failure.

Where possible, client-side (6-balance-alb) or switch-side link aggregation (802.3ad) should be used to achieve the highest performance and availability.

Pure Storage FlashArray

Pure Storage FlashArray is a unified block and file storage solution designed to deliver a seamless and reliable user experience in SQL Server environments. FlashArray supports DirectFlash® software, NVM Express over Fabrics, Fibre Channel, and iSCSI network protocols, in addition to Server Message Block and Network File System file-sharing protocols. Driven by software-defined technology, FlashArray incorporates high data reduction that does not compromise performance. FlashArray is ideal for organizations seeking to improve storage system sustainability. For example, FlashArray//E™ generates 80% fewer direct carbon emissions than competing all-flash systems. This difference is even greater when compared to magnetic disks.

FlashArray is ideal for the following SQL Server use cases:

- As primary block storage for user databases
- As a storage target for T-SQL backups

The FlashArray family consists of the following:

- **FlashArray//C™:** low-latency storage for capacity-oriented workloads
- **FlashArray//X™:** high-performance, high-capacity storage that is ideal for performance-oriented workloads
- **FlashArray//XL™:** high-performance storage at scale that helps reduce the number of arrays needed to run large applications
- **FlashArray//E:** economical-at-scale storage for workloads that aren't time-sensitive

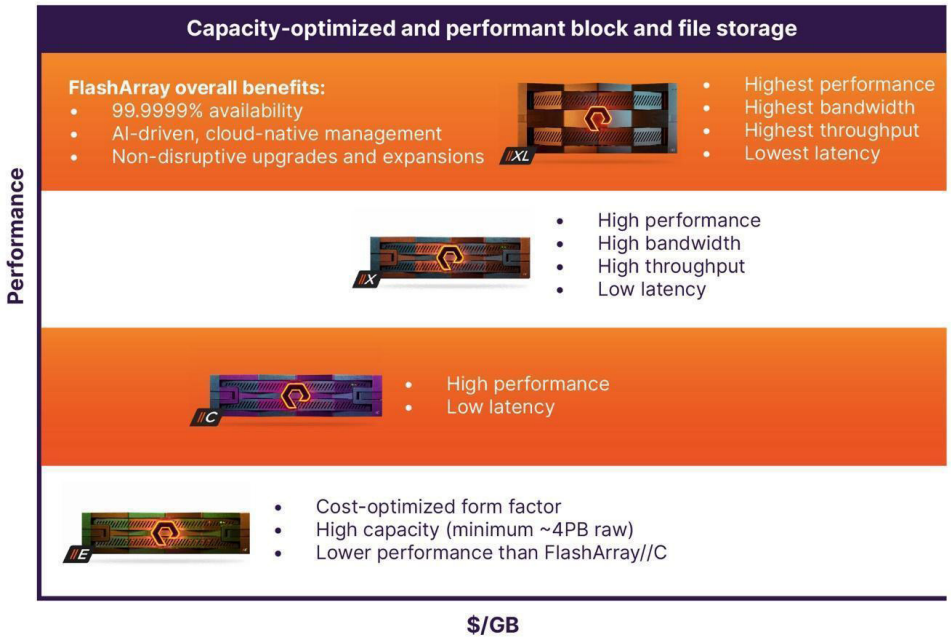

FIGURE 3 Technical specifications of the FlashArray product family

Pure Storage Cloud Dedicated

Pure Storage Cloud Dedicated provides seamless data mobility across on-premises and cloud environments with a consistent experience regardless of where data lives. It provides enterprise-grade storage features in the cloud, and its industry-leading data efficiency means you can buy less capacity in the cloud without sacrificing agility and flexibility. Pure Storage Cloud Dedicated is available in the Amazon Web Services Marketplace and the Microsoft Azure Marketplace.

Pure Storage Cloud Dedicated is available in two versions, //V10 and //V20. Each version provides different capacity and performance capabilities.

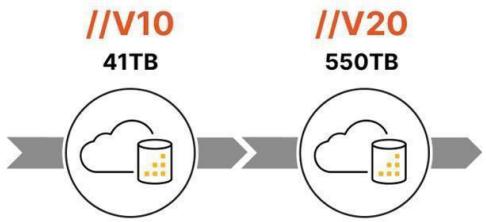

FIGURE 4 The //V10 and //V20 Pure Storage Cloud Dedicated versions; the capacity listed for each model is effective capacity with a 4:1 data reduction rate

FlashBlade

FlashBlade empowers organizations with a simple, adaptable, and scalable storage infrastructure that effortlessly meets the demands of modern data analytics. The innovative blade architecture prioritizes speed, efficiency, and scalability, making FlashBlade ideal for the most demanding and intensive workloads such as rapid recovery, analytics, AI, and machine learning (ML).

FlashBlade is an optimal storage solution for the following SQL Server use cases:

- As a rapid restore target and as storage for a data lake with SQL Server or a hybrid data platform
- For object virtualization of data in either delimited text or parquet format; PolyBase can be used to query relational and nonrelational databases using either external tables or the OPENROWSET function within a T-SQL statement

PolyBase uses the T-SQL language to query data directly from SQL Server, Oracle Database, Teradata, MongoDB, Apache Hadoop clusters, Microsoft Azure Cosmos DB, and Amazon Simple Storage Service (S3)-compatible object storage without having to install additional client-connection software. PolyBase uses data virtualization to combine different data types into a single result set, without moving or copying the data. PolyBase also enables setting FlashBlade as a direct target for T-SQL backups.

The FlashBlade product line includes the following models:

- **FlashBlade//S™:** The latest evolution in enterprise scale-out storage, offering a blend of high density, capacity, performance, and scalability to meet the demands of modern applications.
- **FlashBlade//E:** A cost-efficient storage platform that offers effective performance, uncompromising reliability, and all-flash capabilities to a broader audience. It delivers 40% lower total cost of ownership for an acquisition cost comparable to hard disk drives.

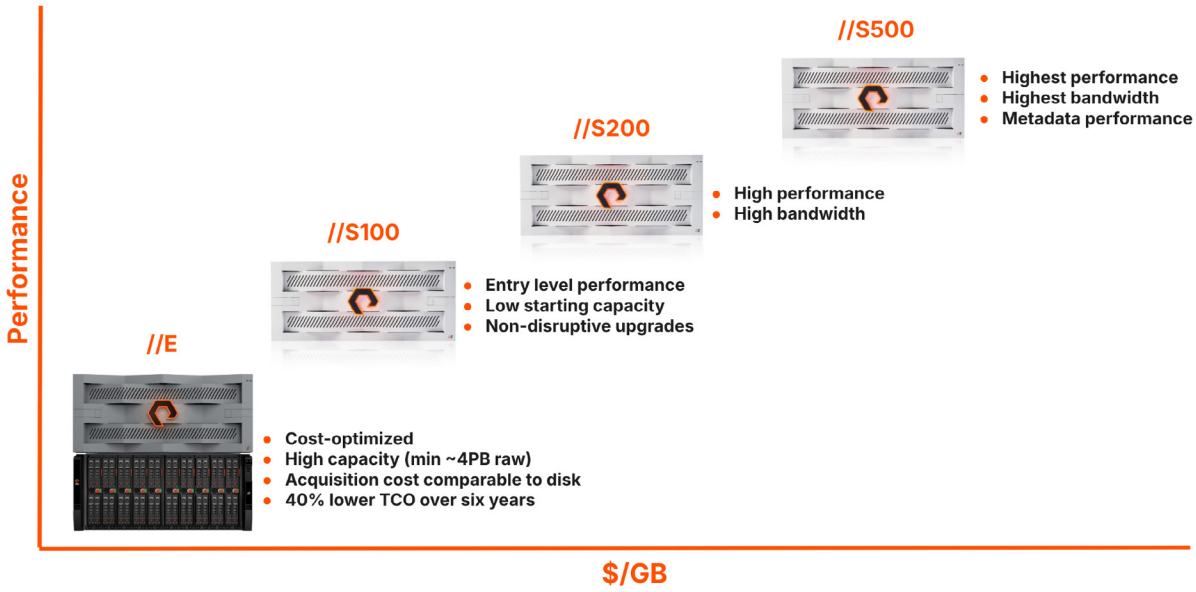

FIGURE 5 Technical specifications of the FlashBlade product family

Microsoft SQL Server

SQL Server is a relational database developed by Microsoft and designed to store, retrieve, and manage large volumes of data for enterprise applications. It supports a wide range of workloads, from online transaction processing to complex online analytical processing. SQL Server ensures data integrity, provides robust security features, and supports high availability through features like Always On Availability Groups. It is widely used for its integration with the Microsoft ecosystem, making it a powerful database solution for both on-premises and cloud environments.

In this reference architecture, SQL Server serves as the core database platform, supporting key business applications and services. It handles structured data with ACID compliance, ensuring reliable transaction processing. SQL Server provides scalability, performance tuning, and data protection features that make it ideal for mission-critical workloads in industries such as finance, healthcare, and manufacturing. Through integration with various tools, SQL Server can seamlessly support real-time data analysis, reporting, and complex workflows within an organization's infrastructure.

Technical Solution Design

The following sections describe multiple technical solution designs within a unified framework for SQL Server databases using Pure Storage solutions. They address distinct business needs through specialized architectures in the following areas:

- Provisioning storage for SQL Server
- Provisioning storage for user databases in public cloud virtual machine infrastructure using Pure Storage Cloud Dedicated
- SQL Server native T-SQL backup and recovery workflows with FlashBlade file or object storage
- SQL Server native T-SQL backup and recovery workflows with FlashArray file storage
- SQL Server PolyBase data virtualization with FlashBlade object storage

Provisioning Storage for SQL Server

This solution is a high-level overview of how Pure Storage provisions storage for SQL Server databases. It is divided into separate technical layers that include compute, networking, storage, and instance optimization.

Compute Layer

This solution uses at least one host in its design of the compute layer. The required servers are commodity x86-64 architectures with a recommended minimum configuration for this reference architecture, as follows:

- Four cores
- 4GB of RAM
- Network cards/Ethernet adapters with at least 2 × 10Gb Ethernet ports for application traffic
- Network cards/Ethernet adapters with at least 2 × 25Gb Ethernet ports for storage traffic (if using iSCSI, NVM Express over TCP, or Server Message Block/Network File System file storage)
- Fibre Channel adapter with at least 2 × 16Gb Ethernet ports for storage traffic using Fibre Channel Protocol

Note: These recommended configurations will vary based on customer need and are only intended to provide a starting template for the implementation journey.

Network and Fabric Layer

Network ports on the hosts should be bonded in any one of the following configurations:

- Highest availability/lowest performance—active/passive (Mode 1)
- Highest performance with no switch-side configuration—active load balancing (Mode 6)
- Highest performance and highest availability—802.3ad/LACP (Mode 4)

It is strongly recommended that dual switches are used and configured with an interconnect between them for availability and performance. For high-performance use cases, the servers and storage should be connected to the same switches or have dedicated storage networking with no routing or hops between them.

FlashArray Fibre Channel Protocol recommendations include:

- If host limits allow it, use all available Fibre Channel Protocol ports on FlashArray.
- Use single-initiator, multi-target zoning, where one SQL Server instance acts as the initiator and targets multiple FlashArray devices. This will create multiple paths to a single volume.
- Use consistent port speeds across the fabric for the best experience.
- Verify that all network paths are clean; address any cyclic redundancy checks or other errors.

FlashArray iSCSI/NVM Express over TCP recommendations include:

- Ensure that the maximum transmission unit is set consistently end to end.
- Disable network adapter power management.
- Verify all network paths are clean; address any cyclic redundancy checks or other errors.

FlashArray file services recommendations include:

- FlashArray file services are built using physical Ethernet interfaces mapped to a virtual interface. Use a virtual interface to combine one or more physical ports on both controllers into a group for managing the storage network.
- The FlashArray file services support page contains comprehensive details on how to configure and manage virtual interfaces.

FlashBlade networking considerations include:

- FlashBlade network fabrics are built using link aggregation groups, subnets, and virtual network interfaces.
- Link aggregation groups are aggregated physical uplinks that are used to optimize availability and load balancing.
- A subnet is a virtual representation of a network range and its associated configurations, and it is bonded to a link aggregation group. Subnet settings include options for VLANs and maximum transmission units.
- A virtual network interface is an interface that connects clients to specific storage protocols and is bonded to a subnet. A virtual network interface usually uses a dedicated static IP address, and it includes options for VLANs and maximum transmission units.
- To avoid network bottlenecks, the best practice recommendation is to implement a link aggregation group capable of the same speed as the number of hosts times the speed of one network port on each. For example, if the servers have 2 × 25Gb Ethernet ports, then the FlashBlade link aggregation group needs to be capable of 100Gb/s minimum. All subnets and virtual interfaces should have the same maximum transmission unit configuration.

Storage Layer

SQL Server supports shared storage using block- or file-based storage. File-based storage on Windows Server uses the Server Message Block file protocol. Block-based storage can use Fibre Channel Protocol or iSCSI Ethernet protocols. Supported deployments of SQL Server on Linux operating systems can use NVM Express over Fabrics storage protocols and Network File System for shared storage and backup targets.

FlashArray

FlashArray block storage is ideal for SQL Server workloads that demand a unified storage solution along with high performance and low latency, such as transactional databases, data warehousing, and analytics or mixed workloads.

The following best practice recommendations help ensure FlashArray is configured for optimal performance and reliability in SQL Server environments:

- Place SQL Server data files (.mdf and .ndf) and log files (.ldf) on separate volumes to manage input/output more effectively.
- Distribute data files across multiple volumes to balance the input/output load.
- Align the file system and block size to avoid misaligned input/output, which can degrade performance.
- Use a 64KB allocation unit size for NTFS volumes hosting SQL Server data and log files.

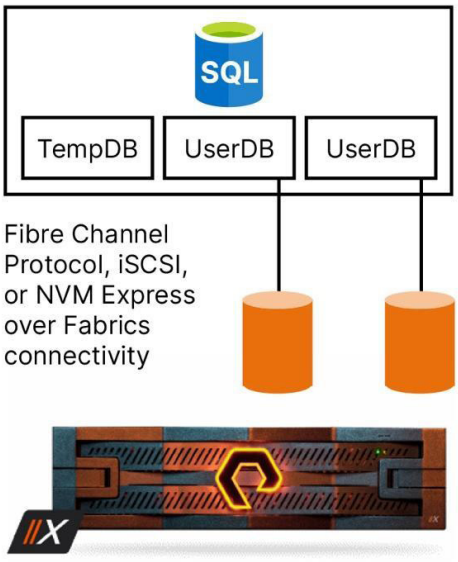

FIGURE 6 SQL Server using FlashArray for primary storage

Pure Storage Cloud Dedicated

When used for primary storage in SQL Server environments, Pure Storage Cloud Dedicated provides consistent performance and seamless data mobility across on-premises, hybrid, and cloud environments.

The following best practice recommendations help ensure Pure Storage Cloud Dedicated is configured for optimal performance
and reliability in SQL Server environments:

- Place SQL Server data files (.mdf and .ndf) and log files (.ldf) on separate volumes to manage input/output more effectively.
- Distribute data files across multiple volumes to balance the input/output load.
- As TempDB is heavily utilized by SQL Server, make sure to provision low-latency volumes for these files.
- Utilize space-efficient, easily captured volume snapshots for data protection, backups, and restores.
- Use ActiveDR for asynchronous replication to a secondary site for disaster recovery that will not affect the performance of the primary site.
- Use ActiveCluster to provide high availability for SQL Server, using zero and near-zero recovery point objective capabilities.
- Use data encryption and access controls to secure data at rest. Regularly review permissions to ensure data security.

Figure 7 illustrates a high-level SQL Server architecture showing how Pure Storage Cloud Dedicated supports high-performance storage across diverse environments.

FIGURE 7 SQL Server using Pure Storage Cloud Dedicated storage across multiple environments

FlashBlade

FlashBlade overcomes many of the challenges posed by traditional data backup and recovery in SQL Server. FlashBlade offers high-frequency backups and rapid data recovery, making it ideal for scenarios involving high transaction volumes, critical data changes, and iterative dev/test environments.

The following best practice recommendations help ensure FlashBlade is configured for optimal performance and reliability in SQL Server environments:

- Configure parallel backup and restore to enable high-frequency backups and minimize downtime during data recovery.
- Use snapshots to create point-in-time database copies that can be used for fast data recovery without impacting performance.
- Use high-speed network connectivity between SQL Server and FlashBlade and configure multiple network paths to facilitate storage redundancy and load balancing.
- Format and organize data to enhance query performance. Use parquet or optimized row columnar file formats and configure volume partitioning to optimize data access.
- Use the T-SQL query language to directly access FlashBlade, along with PolyBase to query external tables or the OPENROWSET function, enabling database integration and efficient processing.
- Use PolyBase for fast access to large data sets stored as objects in S3 buckets.

Figure 8 illustrates a high-level SQL Server architecture using FlashBlade as a backup target.

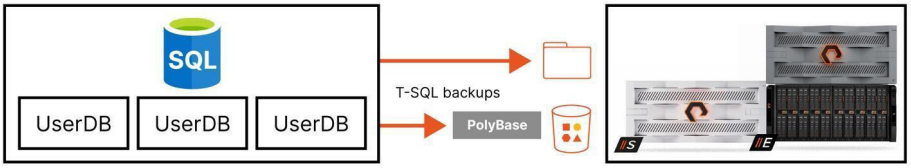

FIGURE 8 SQL Server with FlashBlade as a backup target for file or object storage

Operating System Layer

Windows Server

The following recommendations should be followed for SQL Server using Windows Server:

Folder structure: Both virtual and physical disks can use either drive letters or mount points.
Make sure to provision specific locations for each database file function in alignment with the following examples:

- **SQL Server binaries:** Install SQL Server on the operating system drive.
- **Data files:** Do not place database and backup files on the same disk where the operating system and binaries exist. Do not put SQL Server data and log files on the root of a drive or a mount point. Do not create data files on the root of a file system in Windows Server. Use naming conventions for drive letters or mount points that make sense for that environment. The following are two examples of naming conventions:
  - Use a folder on the D drive, such as **D:\DATA**, or a mount point under the operating system drive, such as **C:\SQL\DATA**. All the data files can be in a single virtual disk, or each database can have a virtual disk provisioned for it (recommended) and then mounted under this structure.
  - The data files in "UserDB" would be placed in a virtual disk mounted at **C:\SQL\DATA\UserDB** or **D:\DATA\UserDB**.
- **Transaction log files:** All transaction log files can be located on a single virtual disk, or each database can have a virtual disk provisioned for it (recommended) and then mounted under this structure. The following is an example of a naming convention:
  - Create a folder named LOGS on the L drive, **L:\LOGS**, or a mount point under the operating system drive, such as **C:\SQL\LOGS**.
  - The log files in "UserDB" would be placed in a virtual disk mounted at **C:\SQL\LOGS\UserDB** or **L:\LOGS\UserB**.

- **TempDB files:** Provision a single virtual disk to contain the data and log files for TempDB. Either **T:\TempDB** or a mount point under the operating system drive, such as **C:\SQL\TempDB**.
- **Spread database files across multiple volumes:** If leveraging the FlashArray SQL Server Management Studio extension, the best practice structure for multiple databases uses a consistent deployment of data and transaction log files. Either place all the data and log files for multiple databases into a single volume/folder or use a dedicated volume for storing the data and log files of a single database. Avoid a multiple database structure that dedicates data files for each database to a separate volume/folder while placing transaction log files for multiple databases on a single volume/folder.

File system and disk formatting: Format all SQL Server volumes with the NTFS file system and a minimum of 64KB allocation units *before* placing any data on the disks.

Block alignment: Check if the block alignment is configured properly. See the "Windows Server File Systems FAQ for FlashArray" knowledge article for more information.

Antivirus and threat detection systems: The minimum recommendation is to configure antivirus and threat detection so they exclude files with .mdf, .ndf, .ldf, and .bak extensions. Other folders and files might need to be excluded as well. For more information, see "Configure antivirus software to work with SQL Server."

Power plan: Set the power plan to high performance.

Instant file initialization: Enable instant file initialization as per "Database instant file initialization."

Page file size: Configure SQL Server with sufficient memory to avoid saving page files to disk. For more information, see "How to determine the appropriate page file size for 64-bit versions of Windows."

Linux

Linux is only supported as a platform from SQL Server 2017 onward. The following are best practice recommendations for SQL Server on Linux:

Folder structure: The default folder for all SQL Server data files is **/var/opt/mssql**. Default folders for data, transaction log, and TempDB files are created during installation:

- **Data**: **/var/opt/mssql/data**
- **Log**: **/var/opt/mssql/log**
- **TempDB**: **/var/opt/mssql/tempdb**

TempDB files should be migrated to a separate virtual or physical disk to achieve the best performance. The virtual or physical disk can be mounted at the default TempDB location after the files have been migrated to it. Create separate folders for data and transaction log files for any additional user databases.

File system, volume formatting, and mount options: Format virtual disks with the XFS or EXT4 file systems and mount them with the "**rw,attr2,noatime**" attributes.

Open file limitations: Set a soft limit of **16,000** and a hard limit of **32,727**.

Forced unit access input/output subsystem capability: For Linux operating systems that support this capability, configure forced unit access using the commands **control.writethrough = 1** and **control.alternewritethrough = 0**.

Disk settings:

- **Input/output scheduler:** Use the None scheduler for all virtual disks that contain SQL Server data files. For more information, see the "How to use the Noop or None IO Schedulers" solution page.
- **Disk readahead**: Set the readahead setting to **4,096**.
- **vm.swappiness:** Set **vm.swappiness** to **1** for high-performance systems.
- **vm.dirty_***: Set **vm.dirty_ratio** to **80** and **vm.dirty_background_ratio** to **3**.
- **kernel.sched_***: Set **kernel.sched_min_granularity_ns** to **15,000,000** and **kernel.sched_wakeup_granularity_ns** to **2,000,000**.

More recommendations for Linux operating systems can be found on the "Linux Recommended Settings" page.

SQL Server Instance

A SQL Server instance is an installation of the database engine, which along with its associated services, is running in an operating system. A SQL Server instance is responsible for managing one or more user databases and several system databases. System databases are not shared between multiple SQL Server instances.

SQL Server databases consist of at least two file types:

- Data files contain data and objects such as tables, indexes, stored procedures, and views. Data files can be either primary (with the **.mdf** extension), containing database startup information, or secondary (with the **.ndf** extension), containing optional user-defined data. Data files can be grouped together into filegroups for allocation and administration purposes.
- Transaction log files (with the **.ldf** extension) contain the information required to recover all transactions in a database.

SQL Server parameters and configurations can be configured using:

- SQL Server Management Studio: Use the graphical user interface to configure server and database settings.
- PowerShell: Use command-line interface, scripting language, and cmdlets to configure server and database settings.
- mssql extension: Use this command-line interface for T-SQL commands to set various parameters.

The following recommendations should be followed for configuring the SQL Server instance.

Service Accounts

Because most environments using SQL Server use Active Directory Domain Services, the SQL Server service accounts should use a domain-based service account. A Group Managed Service Account is recommended because the password is managed by Active Directory Domain Services. This means it is more secure. Ensure that this Group Managed Service Account user is the same one who has the correct permissions to the mount points or drive letters being used for the various SQL Server data, transaction log, and backup folders and files.

TempDB

For an initial configuration, create one TempDB data file per vCPU, up to eight cores. Ensure the files are evenly sized and grow capacity at the same rate. Know your usage of SQL Server to create the right TempDB configuration.

Optimizations

The following are best practice recommendations for optimizing SQL Server performance:

- Set **Max Memory** to recommend 75% of server memory. There is no need to set the minimum value. Microsoft recently released recommendations on memory settings, and you can find more information about it in this guidance.

Note: There are always exceptions based on workload and features in use.

- Set **Max Degree of Parallelism (MAXDOP)** to less than or equal to the number of cores in a NUMA node of less than or equal to eight. This is an advanced setting. It is recommended that you read the Microsoft documentation before changing this setting.
- The **Cost Threshold** for parallelism should be optimized while a workload is running, but start at a value of fifty.
- **Lock pages in memory**, when set, can prevent external pressure to deallocate SQL Server memory processes.
- Turn on accelerated database recovery (SQL Server 2019 and newer) to improve database availability, especially in the presence of long-running transactions.
- Enable query store (SQL Server 2016 and newer) to collect and store information about query processing and runtimes. While query store should be enabled for most cases, test with your workloads to confirm optimal performance. Query store does not work in some scenarios, such as Always On Availability Group readable replicas.
- Turn on ad hoc workload optimization, which limits an execution plan from being cached until it is executed a second time. The first execution plan is saved only as a stub, which prevents memory bloat from too many execution plans being cached. Additional information can be found at the "optimize for ad hoc workloads (server configuration option)" page.
- Turn on auto-update statistics asynchronously, which allows queries to continue executing during the update of index statistics.

T-SQL Backups Using FlashBlade for File or Object Storage

This section explains how to design T-SQL backups using FlashBlade as the target for file- or object-based storage.

Overview

FlashBlade can be used as a target for T-SQL backups in SQL Server 2022 or later. By providing high-throughput and low-latency data storage, it delivers efficient backup and restore operations for large data volumes. FlashBlade seamlessly integrates with the native T-SQL backup capabilities of SQL Server, supporting Server Message Block file shares, S3 object storage, and multiple security mechanisms. Its streamlined data management helps maintain high availability and performance for SQL Server during backup operations.

Benefits

FlashBlade provides flexible, scalable, and secure data storage that enhances the reliability of backups and ensures data integrity and availability. Integration with native SQL Server tooling helps ensure efficient backup and recovery processes, minimizing data recovery times that can bottleneck operations and disrupt business continuity. A parallel architecture and all-flash technologies help significantly accelerate data transfer, ensuring the most recent data is always protected, even with massive data volumes. FlashBlade supports file- and object-based storage via Server Message Block file shares and S3 buckets for object storage. This allows backups to be designed for a variety of environments, including on-premises, hybrid, private cloud, and public cloud. The scalable architecture enables easy expansion of storage capacity as businesses and their data volumes grow, without compromising SQL Server performance.

By using FlashBlade as the target for T-SQL backups, organizations can address several challenges presented by traditional backup systems. Its comprehensive data services, such as instant recovery, multisite replication, cloud integration, and granular application recovery, are typically lacking from traditional storage solutions. FlashBlade uses data reduction technologies to optimize resource usage, improve storage scalability, and lower the total cost of ownership.

Technical Solution

FlashBlade integrates compute, networking, and storage into a single platform, which simplifies storage management and enhances performance. FlashBlade provides file or object storage via Server Message Block file shares and S3 object storage over Ethernet or Fibre Channel to any hosts. Server Message Block is a versatile and widely supported protocol for T-SQL backups that enables seamless integration and user-friendly configuration with various operating systems. S3-compatible object storage ensures a straightforward transition to hybrid and cloud storage for users familiar with the S3 protocol, enabling efficient backup and recovery processes. S3-based object storage inherits the durability and reliability associated with the S3 standard.

High-speed Ethernet or Fibre Channel Protocol is recommended for optimal backup and recovery performance. FlashBlade can handle multiple concurrent streams, enabling backups and restores of multiple databases simultaneously. The best practice recommendation is to have SQL Server allocate multiple write threads to run in parallel, splitting the backup into multiple files on the target FlashBlade.

Code examples for creating T-SQL-based snapshots for point-in-time database recovery can be found on the Pure Storage SQL Server scripts GitHub page, which can be implemented using the Pure Storage PowerShell SDK2 module.

Figure 9 shows FlashBlade configured to store T-SQL backups on Server Message Block and S3.

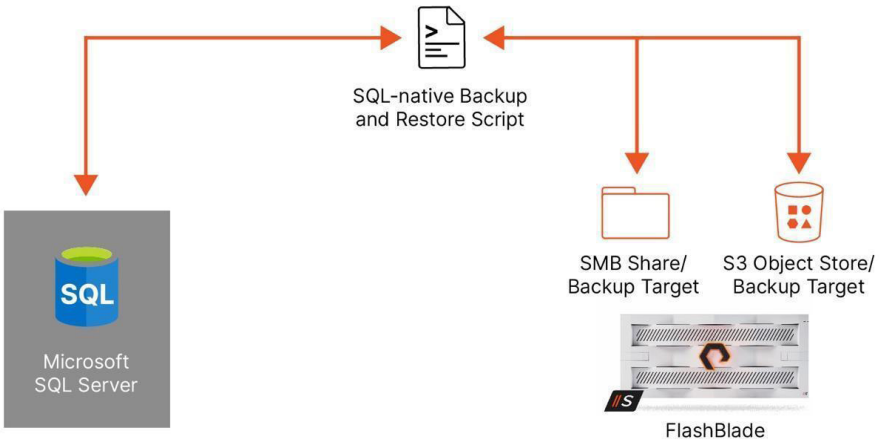

FIGURE 9 SQL Server with FlashBlade as the target for T-SQL backups

Database Copies for Development, Testing, or Analysis Using FlashArray Snapshots

This section explains how to use FlashArray snapshots to generate database copies for development, testing, or analysis.

Overview

FlashArray snapshots can be used for copying databases to other devices. These cloned databases are useful for conducting dev/test outside of the production environment. For example, a dev/test database can be used to test whether SQL Server backups meet data-restore requirements, such as length of downtime and percentage of data loss, without impacting the performance and integrity of production systems.

Solution Benefits

Databases copied using FlashArray snapshots offer multiple advantages for SQL Server development, testing, and analysis. They can be used in dev/test environments to speed up dev/test refresh cycles, streamline in-place upgrades, and offload dev/test operations from production systems.

Developers shorten dev/test refresh cycles by enabling access to multiple refresh variations on copied volumes rather than accessing full databases in production. Database copies can also deliver nearly instantaneous reversion to a pre-upgrade state, which can significantly reduce recovery time and risk in case of upgrade failures. CHECKDB database maintenance tasks can also be offloaded to avoid impacting production performance. Extract, transform, and load processes can be offloaded to database copies in dev/test environments, reducing resource-intensive production loads such as buffer pool, CPU, input/output subsystem, and networking operations.

Technical Solution

FlashArray can generate two kinds of snapshots for database copies: crash-consistent and application-consistent. Crash-consistent snapshots preserve SQL Server data and log files but not uncommitted transactional data. When a crash-consistent snapshot is taken, the FlashArray works with SQL Server's write-ahead logging protocol to ensure the database is recoverable. Application-consistent snapshots capture transactional data, enabling point-in-time recovery, provided the database is paused (input/output quiescing) before taking the snapshots. Crash-consistent volume snapshots are appropriate for use when point-in-time recovery is not required.

FlashArray volume snapshots can be created, configured, and deleted by using:

- FlashArray web or command-line interface (manual configuration)
- Pure Storage FlashArray PowerShell SDK2 or Python REST client (automated configuration)
- T-SQL snapshot backup procedure (SQL Server 2022 and newer)
- Pure Storage Volume Shadow Copy Service provider (SQL Server 2019 and older)

Code examples for snapshots can be found on the Pure Storage SQL Server scripts GitHub page, which can be implemented using the Pure Storage PowerShell SDK2 module.

Figure 10 illustrates a SQL Server database copy used for development or testing.

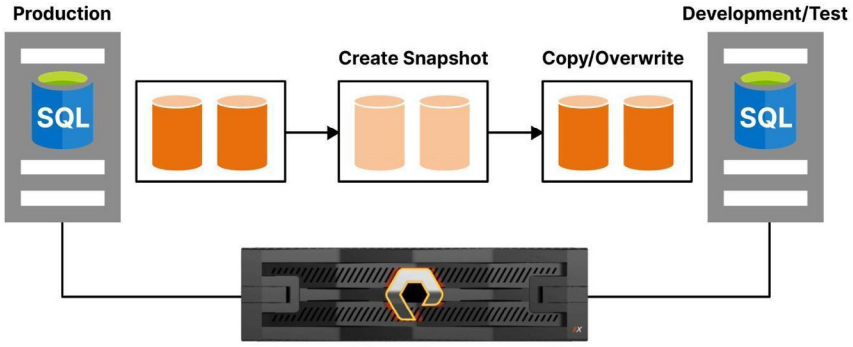

FIGURE 10 Cloned SQL Server database for dev/test

Data Protection Using Storage Snapshots on FlashArray or Pure Storage Cloud Dedicated

This section explains how to implement data protection using storage snapshots on FlashArray or Pure Storage Cloud Dedicated.

Overview

Volume snapshots can be used for data protection in SQL Server environments by creating immutable, point-in-time images of the contents of one or more volumes and saving them to FlashArray or the cloud using Pure Storage Cloud Dedicated. Volume snapshots are easily created using various interfaces, such as the Purity web interface, a command-line interface, the REST API, PowerShell, or a Python REST client. Unlike server backups, snapshots are extremely compact files using metadata to capture system settings and references. Each snapshot is an independent record of the data-retention structure that cannot be altered or encrypted. A retention policy ensures a snapshot cannot be deleted before the designated term.

Benefits

FlashArray and Pure Storage Cloud Dedicated incorporate the same fundamental engineering for their volume snapshot technologies, making this approach to data protection suitable for any business environment. These volume snapshots deliver several unique advantages, beginning with their ability to be created or recover instantly. The snapshot uses a redirect-on-write technique that redirects any writes to an original data block that is referenced by a snapshot to a new data block, thereby preserving the state of the original data block. This feature maintains the integrity of a snapshot's data while requiring minimal storage space.

Snapshots can be made for individual volumes or groups of volumes using protection groups, ensuring point-in-time consistency. They are storage-space-efficient because they are always thin-provisioned, deduplicated, and compressed, and they require no capacity reservations. Snapshot transfers use space-efficient copies along with volume metadata, ensuring fast and complete data recovery for any FlashArray or Pure Storage Cloud Dedicated instance. Recovery options include restoring or overwriting the original volume or cloning it to a new volume.

Note: If creating snapshots for protection groups on multiple databases, all volumes associated with the protection group must be included.

Moreover, FlashArray volume snapshots are portable and transport-efficient, allowing them to be transferred to other FlashArray or Pure Storage Cloud Dedicated instances or offloaded to Network File System storage, Amazon S3 buckets, or Azure Blob Storage. When enabled, the SafeMode Snapshots feature offers robust ransomware protection for backup data and metadata.

Technical Solution

FlashArray can create either application-consistent or crash-consistent volume snapshots. The main difference between the two types lies in the level of granularity and recovery point objective times. An application-consistent snapshot provides a point-in-time snapshot that combines a volume snapshot with transaction log restoration that can bring the database recovery point up to any point within the transaction log. This allows the database to achieve transaction-level point-in-time data restores from native SQL Server backups. Using SQL transaction log backups is recommended when a snapshot must be rolled forward to a recovery point in time in the transaction log.

A crash-consistent snapshot can achieve a recovery point at the time the snapshot was taken. Crash-consistent snapshots work independently from an application and will always present a valid database from the point in time of the snapshot.

Note: Crash-consistent snapshots do not support point-in-time recovery. Application-consistent snapshots are recommended for transaction-level point-in-time recovery.

Code examples for volume snapshots can be found on the Pure Storage SQL Server scripts GitHub page, which can be implemented using the Pure Storage PowerShell SDK2 module.

Figure 11 illustrates how FlashArray volume snapshots are used for SQL Server data protection.

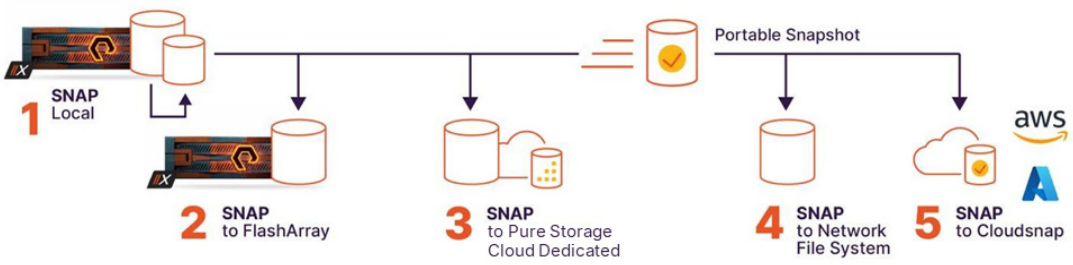

FIGURE 11 SQL Server data protection using volume snapshots in FlashArray or Pure Storage Cloud Dedicated

High Availability for SQL Server Using ActiveCluster

This section explains how to use ActiveCluster to deploy a high availability SQL Server database.

Overview

ActiveCluster is a feature of FlashArray designed to provide high availability and business continuity for SQL Server. ActiveCluster uses synchronous replication and automatic transparent failover to deliver both zero recovery point objectives and zero recovery time objectives. ActiveCluster allows multiple SQL Server instances to serve input/output data traffic on the same volume simultaneously, simplifying management and enhancing performance.

Benefits

ActiveCluster can be configured to work within the same data center or across multiple locations. If using ActiveCluster across multiple locations, network latency and bandwidth must be carefully considered. When ActiveCluster is implemented across multiple data centers, this enables the deployment of clustered arrays and hosts in a flexible and active-active data center configuration. This means that data can be accessed and utilized from different sites, providing high availability and disaster recovery capabilities. The active-active configuration allows both the primary and backup systems to actively serve input/output traffic on the same volume for enhanced application performance and resource utilization. ActiveCluster delivers zero recovery point objectives and zero recovery time objectives, ensuring no data loss occurs and that systems remain immediately available during failovers. Failovers are automatic and transparent to applications, ensuring continuous availability without manual intervention.

Additionally, ActiveCluster optimizes latency by serving reads locally, ensuring high performance. It supports both uniform and non-uniform configurations, allowing the setup to be tailored to specific environments. The solution is scalable, enabling the expansion of storage infrastructure without disrupting operations, and it provides enhanced data protection through synchronous replication, ensuring data consistency across sites. These benefits collectively create a highly available, resilient, and efficient SQL Server environment with minimal downtime and optimal performance. ActiveCluster management and setup is as straightforward as managing a single array, thereby reducing administrative overhead.

Technical Solution

To use ActiveCluster for SQL Server high availability, the SQL Server instance should be connected to local (Site A) and remote (Site B) FlashArray storage using Ethernet or Fibre Channel networking, as shown in Figure 12. This uniform storage configuration is recommended to simplify the deployment of failover cluster instances, and it ensures that failovers are transparent and automatic. Should failover occur, local and remote storage are synchronized automatically, ensuring low latency and continuous availability of applications and systems.

Transparent failover with zero recovery point objectives and zero recovery time objectives is achieved using active-active synchronous replication. Unlike active-active solutions that are active-passive at the volume level, ActiveCluster serves reads and writes on a given volume from both sites simultaneously. It also reduces application latency by having reads served locally. ActiveCluster maintains storage efficiency with inline compression on the wire and by performing XCOPY operations, ZeroSnap snapshots, and clones efficiently across arrays.

Code examples for ActiveCluster can be found on the Pure Storage SQL Server scripts GitHub page, which can be implemented using the Pure Storage PowerShell SDK2 module.

Figure 12 illustrates how FlashArray and ActiveCluster enable high availability for SQL Server.

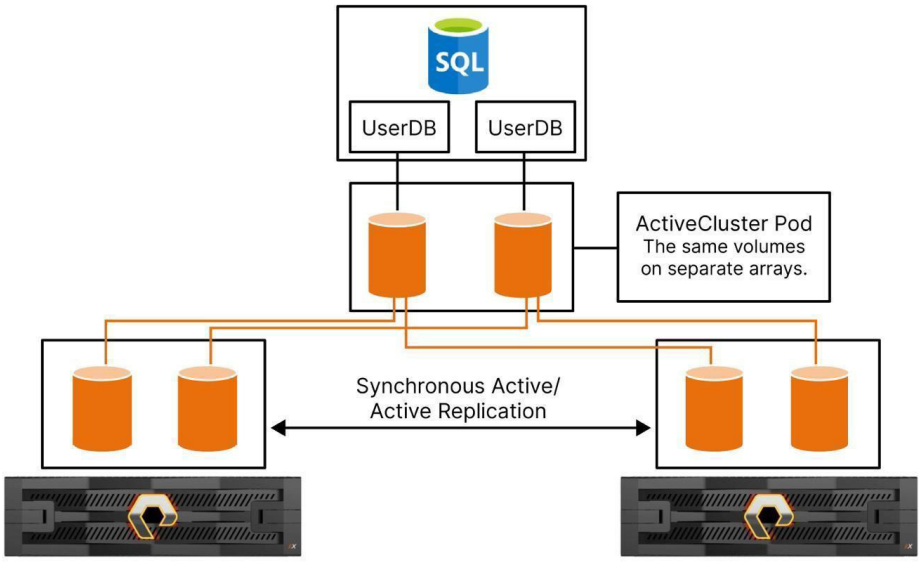

FIGURE 12 High availability SQL Server with ActiveCluster architecture

Near-synchronous Replication for SQL Server with ActiveDR

This section explains how to implement near-synchronous replication for SQL Server using the ActiveDR feature in FlashArray.

Overview

Pure Storage ActiveDR is implemented to provide continuous replication for SQL Server databases using block storage volumes on FlashArray or Pure Storage Cloud Dedicated. ActiveDR continuously transfers data changes from the primary storage array to a secondary array, maintaining an up-to-date copy of the data that can be quickly accessed in the event of a failure. The data is continuously replicated, enabling near-synchronous replication and near-zero recovery point objectives that significantly reduce the risk of data loss during a disaster. This replication occurs with no impact on front-end application performance, ensuring data consistency and availability in SQL Server environments.

Benefits

ActiveDR can be used for nondisruptive failover to a second site, where the instance can be used for test, development, and quality assurance scenarios without interruption to the primary active site. ActiveDR delivers near-zero recovery point objectives with continuous replication for fast data recovery and failover time across almost any distance, while minimizing recovery points and recovery times. It also provides single-command failover, intelligent failback, and nondisruptive data recovery testing to help accelerate response times to outages. With ActiveDR, recovery for SQL Server databases can be conducted without stopping replication, ensuring business continuity during testing and failover. ActiveDR helps improve SQL Server resilience without impacting application or system latency.

Technical Solution

ActiveDR uses storage management containers called **pods** to group FlashArray storage objects and configuration settings, enabling coordinated failover and failback. **Replica links** facilitate directional and intelligent auto-reversing replication between these pods. The replication process starts with initial synchronization and baselining, followed by continuous replication operations to maintain data consistency.

Figure 13 depicts a disaster recovery process between two sites, each having a SQL Server instance and FlashArray storage. The wide-area network should have sufficient bandwidth to handle high volumes of replication traffic.

At the production site, pod 1 (source) on the primary FlashArray contains the volumes to be replicated. At the disaster recovery site, pod 2 (target) holds the replicated volumes. Data and applications are continuously replicated from the production site to the disaster recovery site via the replica link. This setup ensures that if the production site fails, the disaster recovery site can quickly take over with minimal data loss.

ActiveDR begins synchronization by transferring baseline data from the source to the target pod, ensuring both sites have a consistent starting point. After the initial synchronization, ActiveDR continuously transfers data changes from the primary to the secondary site in near real time. If the primary site fails, pod 2 is automatically or manually promoted to the new primary source pod. Once pod 1 has been recovered and is back online, ActiveDR can apply re-protection, reversing the replication direction and returning pod 1 to its original status as the source pod.

Code examples for ActiveDR can be found on the Pure Storage SQL Server scripts GitHub page, which can be implemented using the Pure Storage PowerShell SDK2 module.

Figure 13 illustrates how FlashArray ActiveDR provides near-synchronous replication for SQL Server.

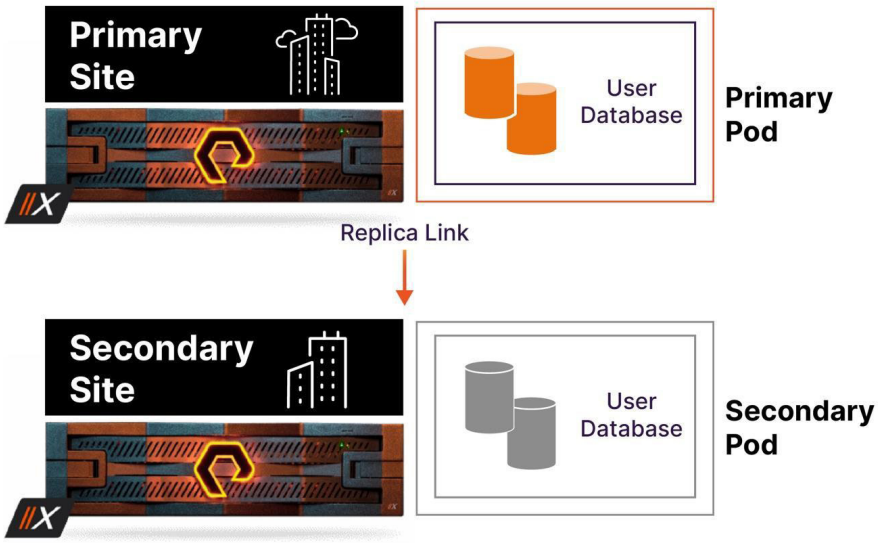

FIGURE 13 Near-synchronous replication with ActiveDR architecture

Design Validation and Benchmarking

Design validation of any SQL Server solutions with Pure Storage is necessary to ensure applications and systems will perform optimally in the production environment. Validations can be performed to assess SQL Server performance.

SQL Server performance can be impacted by these additional factors:

- Host-specific bottlenecks, such as CPU and memory overhead
- Storage bottlenecks, such as array configuration and resource limits
- Network bottlenecks, such as number of hops and bandwidth

The use of a database benchmarking tool is recommended for identifying and addressing potential performance bottlenecks that could impact SQL Server latency, availability, and scalability. For example, benchmarking results can be analyzed to verify that a SQL Server design with Pure Storage meets the required efficiency and scalability performance standards.

HammerDB is a widely used benchmarking tool that can be used to validate the performance of a SQL Server architecture using FlashArray or FlashBlade for data storage. To conduct a validation benchmark, HammerDB should be configured to simulate a transactional SQL Server workload that mirrors database operations in the production environment. For example, the test schema should generate a substantial volume of transactions to enable an accurate evaluation of database and storage throughput and latency. Parameters such as the number of virtual users and the complexity of transactions should be set at levels that enable stress-testing the SQL Server design with FlashArray and FlashBlade.

Deployment

This section provides key details for deploying SQL Server on Pure Storage FlashArray, with a focus on provisioning block storage and setting up SQL Server databases on it.

See the following links for more information:

- Pure Storage documentation: Microsoft Platform Guide
- Pure Storage documentation: "FlashArray Admin and CLI Reference Guides
- Microsoft documentation: Installing PowerShell on Windows
- Pure Storage documentation: Best Practices for Microsoft SQL Server on FlashArray

FlashArray

This deployment guidance covers the core steps to provision block storage for SQL Server user databases. It assumes the array has been deployed but that specific configuration of the storage component for the user database has not been performed.

Follow the steps below to provision block storage for a SQL Server user database with FlashArray.

1. Create the storage host by navigating to the **Storage** view and then clicking the **Hosts** tab. To create a new host, click the **+** in the top right of the **Hosts** list.

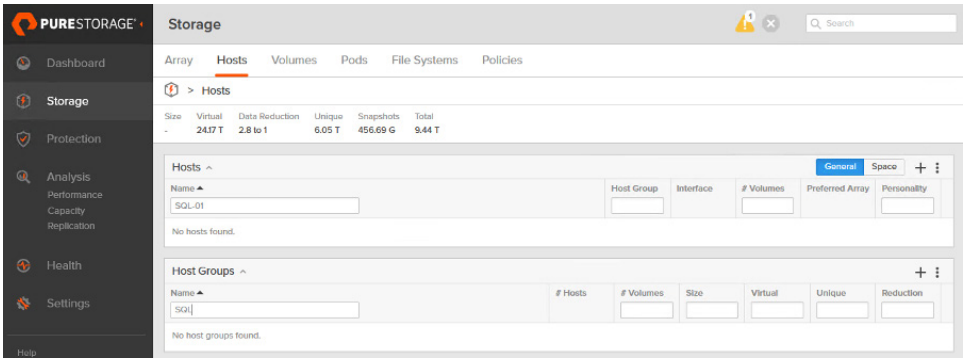

FIGURE 14 The Hosts list in the Storage view in the FlashArray graphical user interface

2. In the **Create Hosts** dialog, provide a name and optionally set the host to be added to a protection group, and then click **Create**.

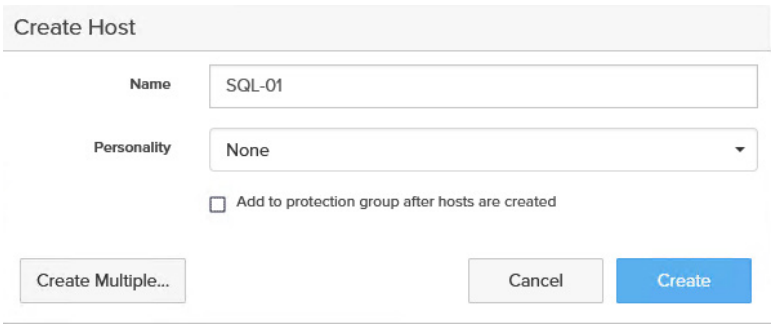

FIGURE 15 The Create Host dialog

3. In the newly created host's detail view, identify the section for **Host Ports**, click the ellipsis, and then, depending on the required storage protocol, select the relevant entry to provide (WWN for Fibre Channel Protocol, IQN for iSCSI, and NQN for NVM Express over Fabrics).

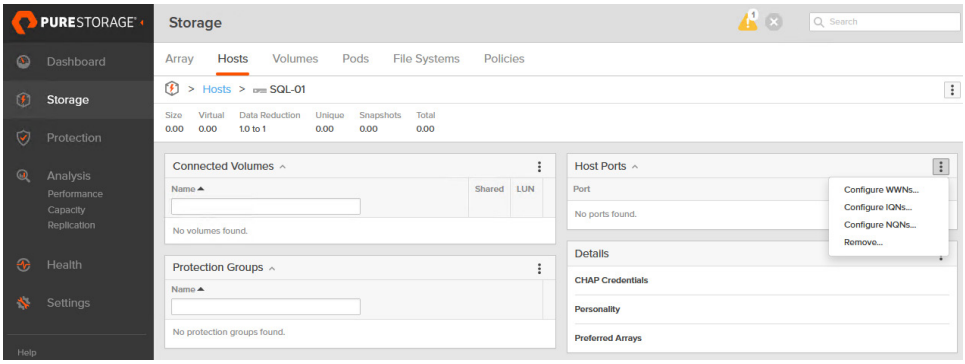

FIGURE 16 The Host Ports configuration options

a. If using Fibre Channel Protocol, the WWNs might be selectable from the existing WWN list. If so, select them, and then click **Add**.

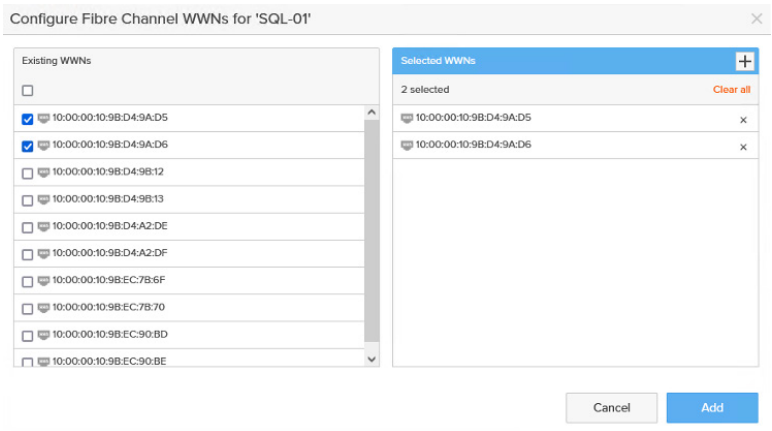

FIGURE 17 The Configure Fibre Channel WWNs for host dialog

b. If using iSCSI, the IQN needs to be obtained from the host and entered in the dialog box. After doing so, click **Add** to complete the process.

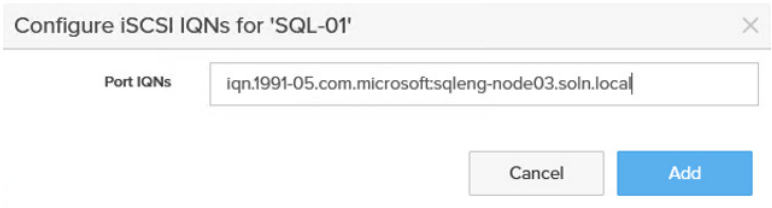

FIGURE 18 The Configure iSCSI IQNs for host dialog

4. To provision storage, navigate to the **Storage** view, and then select the **Volumes** tab. Identify the **Volumes** list section and then click the **+** on the right side to create a new volume.

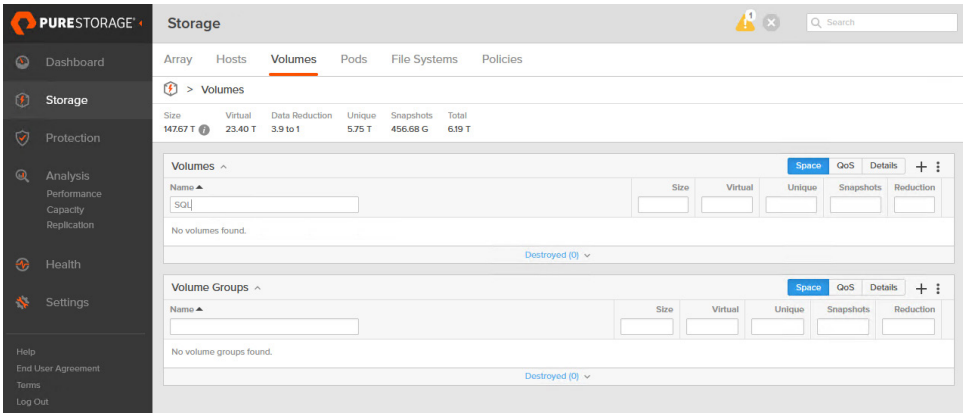

FIGURE 19 The Volumes list in the Storage view in the FlashArray graphical user interface

5. In the **Create Volume** dialog, provide a **Name**, set a **Provisioned Size** for the new volume, and then click **Create**.

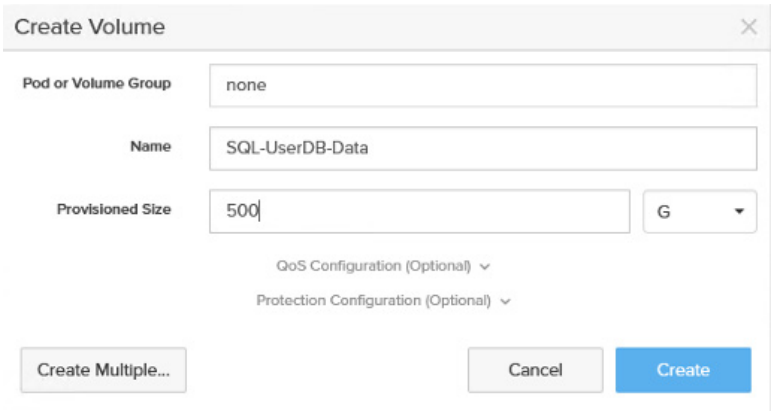

FIGURE 20 The Create Volume dialog

6. Repeat steps 4 and 5 for each new volume that needs to be created. The example shown in Figure 21 creates two volumes, one for the database data files (.mdf and .ndf) and another for the transaction log files.

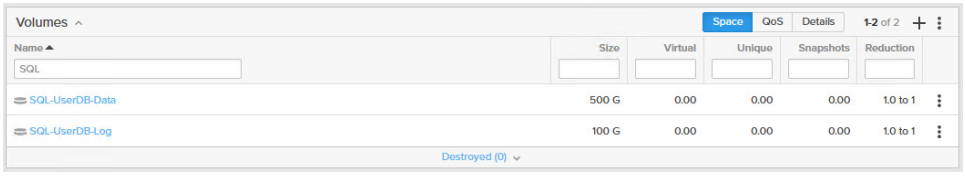

FIGURE 21 The created volumes in the volume list

7. To connect the volumes to the host created in step 2, navigate to the **Storage** view, then to the **Hosts** tab, and then select the host from the list to open its detail view. Identify the section for **Connected Volumes**, click the ellipsis in the top right, and then select **Connect.**

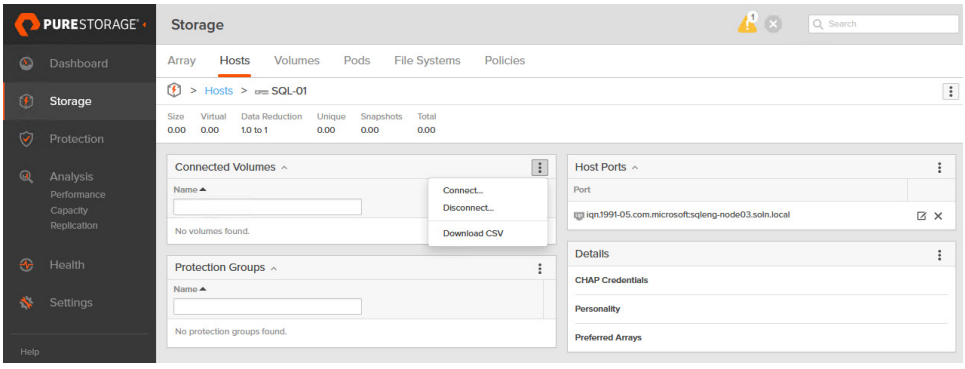

FIGURE 22 The connected volumes configuration options in the host view

8. In the **Connect Volumes to Host** dialog, select the volumes created in step 5, and then select **Connect**.

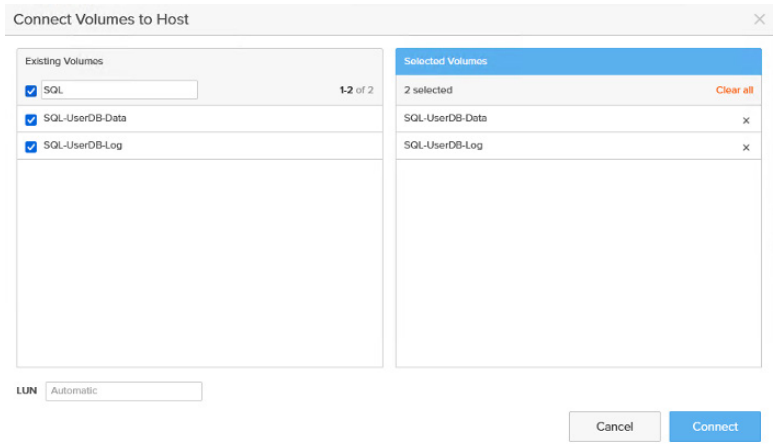

FIGURE 23 The Connect Volumes to Host dialog

9. Once all the required volumes have been connected, the host should show them in the Connected Volumes list.

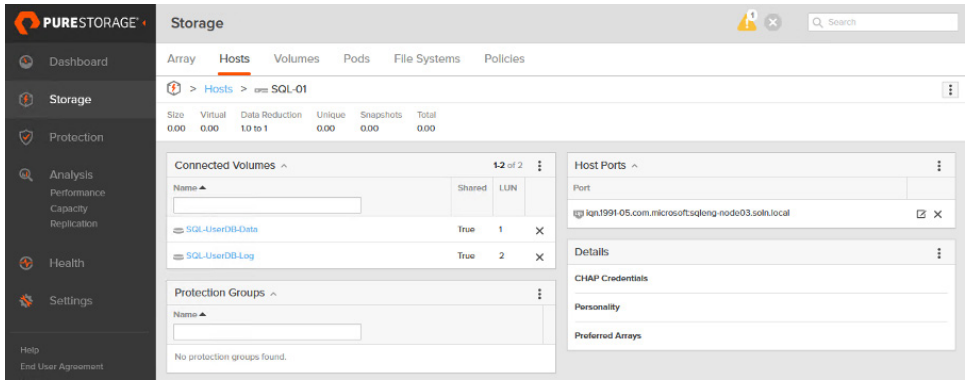

FIGURE 24 A configured host with attached volumes

Operating System

These steps assume that the Fibre Channel or iSCSI device discovery process has already been followed and that the disks have been discovered successfully.

Validate Windows Server with the Test-Pfa2WindowsBestPractices Cmdlet

When deploying FlashArray volumes to a new host, it is important to make sure that Windows Server is configured properly and set to the recommended best practice settings for Multipath I/O (MPIO) and TRIM/UNMAP support and that Offloaded Data Transfer is enabled. In order to make this easy, a Windows PowerShell cmdlet, Test-WindowsBestPractices, was introduced in the Pure Storage PowerShell toolkit.

Note: The cmdlet has been renamed as of the toolkit version 3. Existing scripts and references will need to be updated to reflect the new name—Test-Pfa2WindowsBestPractices.

Disk Management

Follow these steps in Windows Server to provision the FlashArray volumes as disks with a formatted file system for SQL Server data file storage.

1. In **Disk Management**, identify the new volumes in the list of disks.

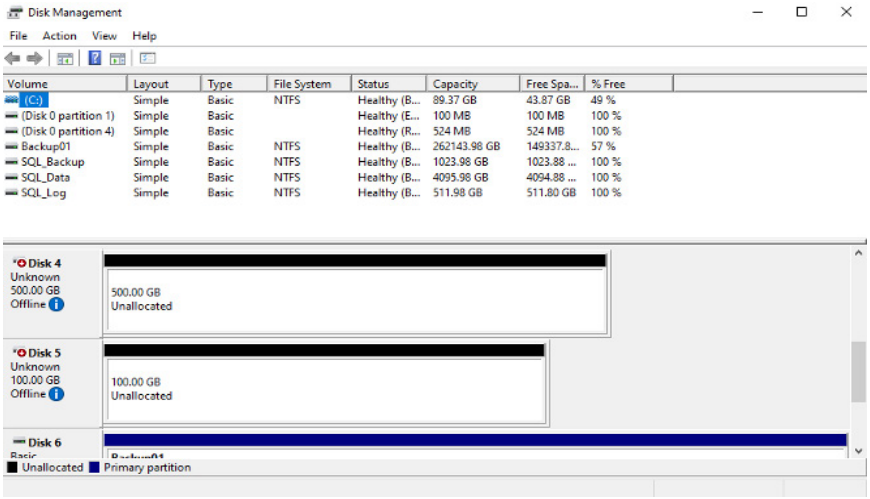

FIGURE 25 Disk Management application in Windows Server

2. Set the disk as online by right-clicking the “Disk” portion of the view, and then selecting **Online**. Repeat this process for each disk being configured.

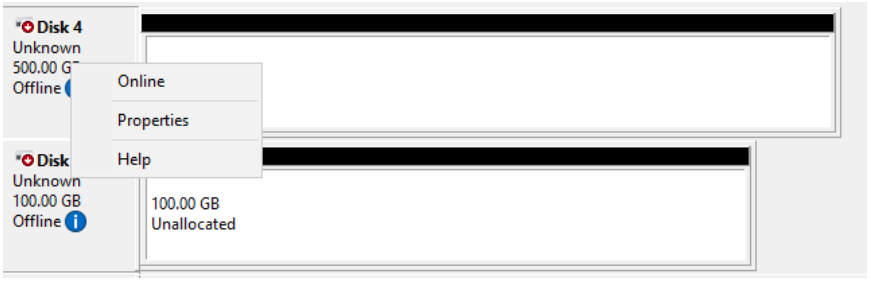

FIGURE 26 The drop-down menu for offline disks

3. When all disks are online, right-click the “Disk” portion of the view, and then select **Initialize Disk**.

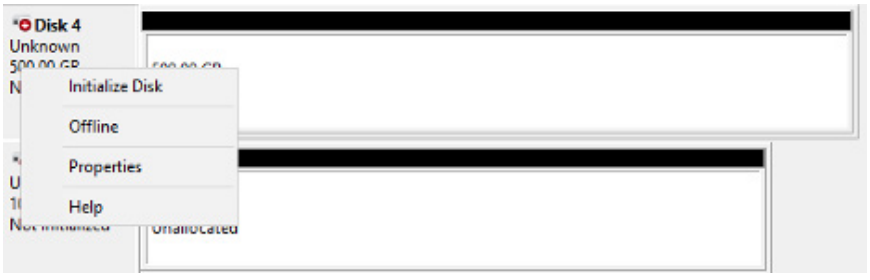

FIGURE 27 The drop-down menu for online disks that require initializing

4. In the **Initialize Disk** dialog, select each disk that needs to be configured, and then ensure that the GPT partition style is selected. Click **OK** to proceed.

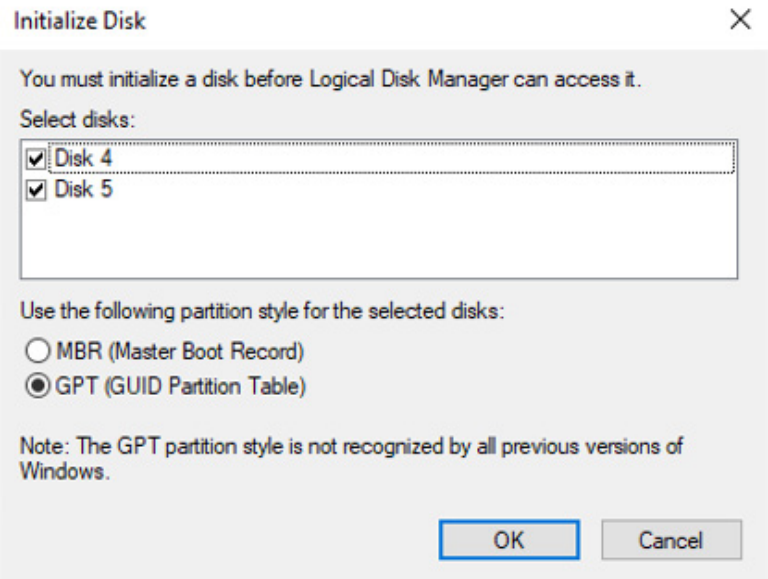

FIGURE 28 The Initialize Disk dialog

5. Select the "Partition" portion of the view from the disk, right-click it to bring up the configuration list, and then select **New Simple Volume**.

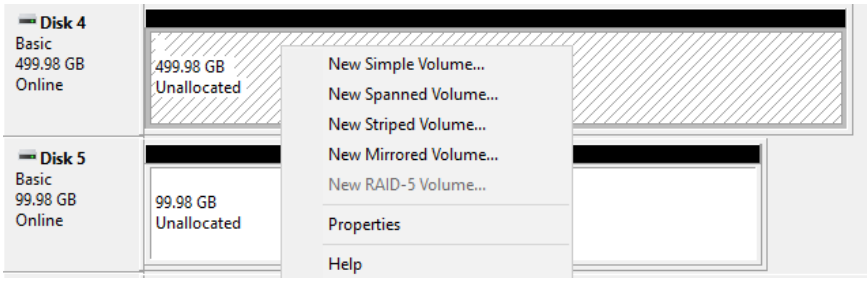

FIGURE 29 The volume management drop-down

6. The **New Simple Volume** wizard will launch. Click **Next** to proceed.

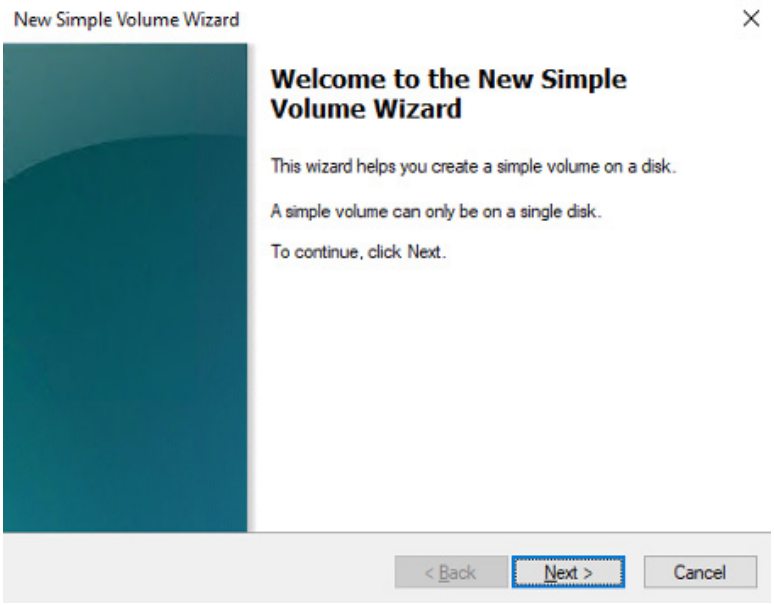

FIGURE 30 The starting state for the New Simple Volume wizard

7. Specify the **Simple Volume Size** in MB up to the maximum disk space, and then click **Next**.

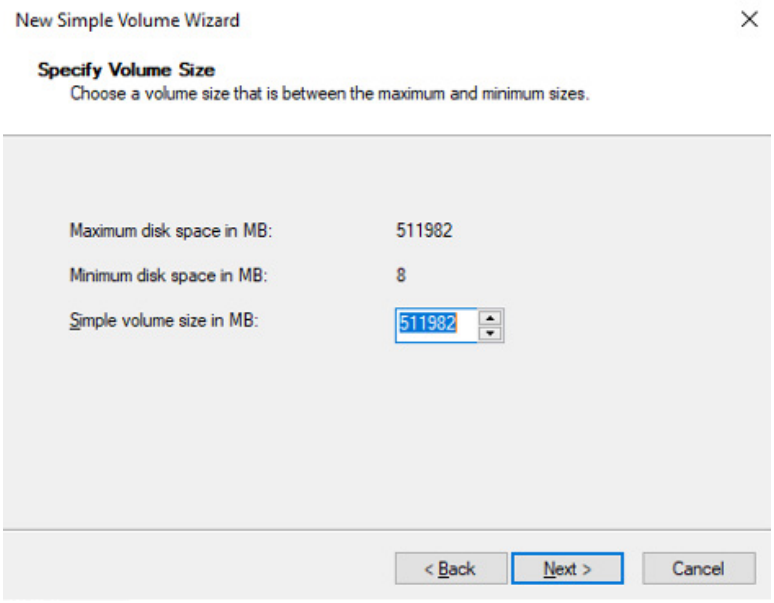

FIGURE 31 Specifying a volume size in the New Simple Volume wizard

8. Specify the drive letter or mount point, and then click **Next**.

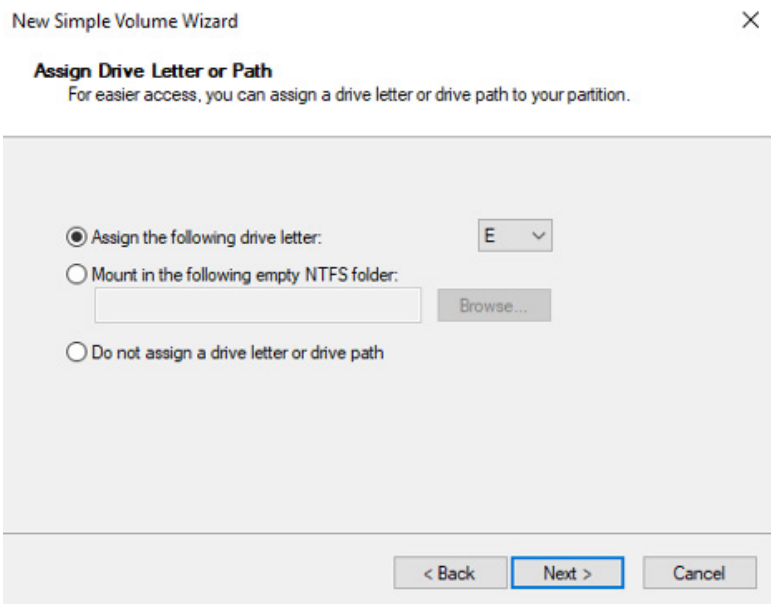

FIGURE 32 Assigning a drive letter or mount point in the New Simple Volume wizard

9. Choose the NTFS file system with an **Allocation unit size** of **64K**, provide a volume label, and then click **Next**.

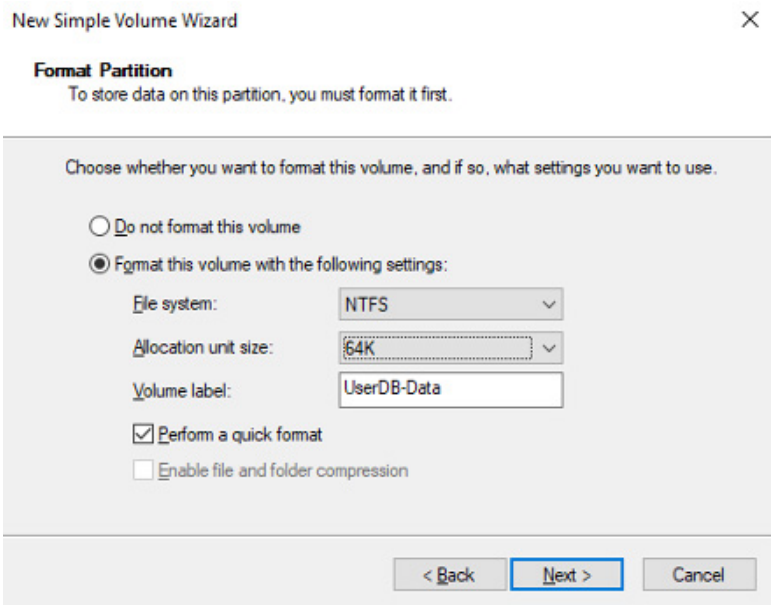

FIGURE 33 Formatting the partition in the New Simple Volume wizard

10. Review the configuration, and then click **Finish**.

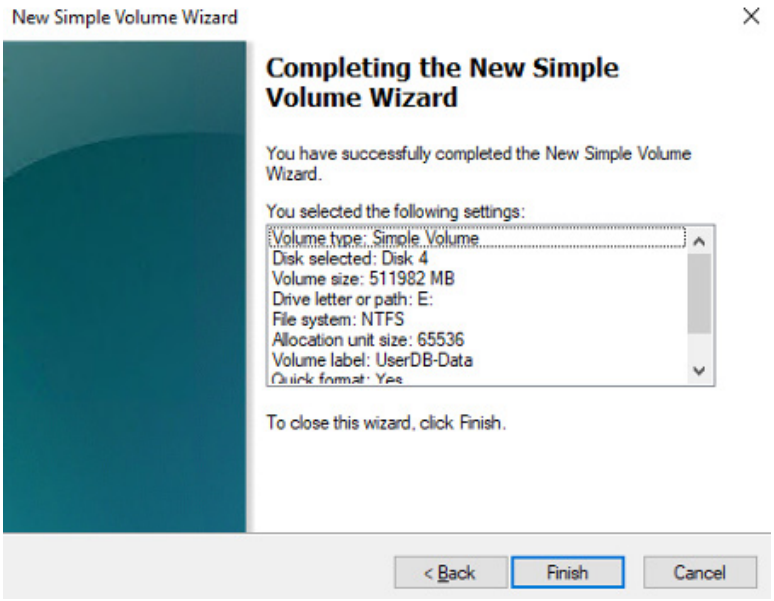

FIGURE 34 The confirmation dialog in the New Simple Volume Wizard

11. Repeat steps 5–10 for each disk to create a file system and assign a mount point or drive letter.

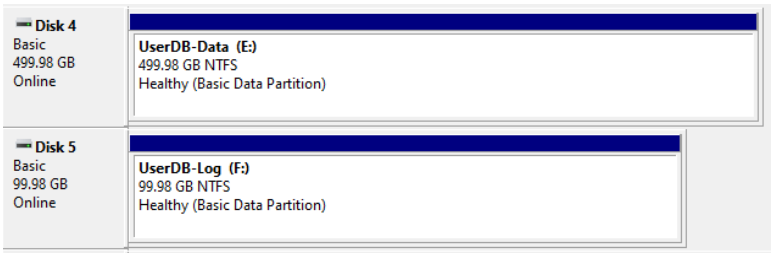

FIGURE 35 The volumes with file systems and drive letters assigned

12. Create a folder on the root of each new file system for the database files to be contained within.

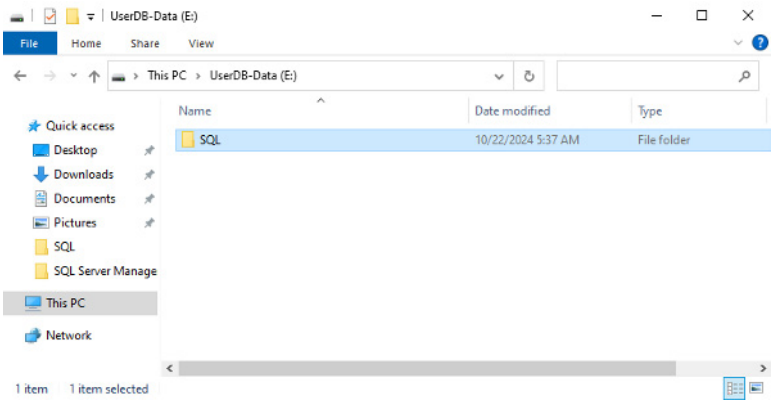

FIGURE 36 A folder is created on the root of the file system to contain the SQL Server data files

Creating the User Database

To create a user database on the newly created volumes and associated file systems, use the **CREATE DATABASE** T-SQL command.

```
CREATE DATABASE UserDB
 ON
( NAME = N'UserDB', FILENAME = N'E:\SQL\UserDB.mdf' , SIZE = 67108864KB , FILEGROWTH = 65536KB
)
 LOG ON
( NAME = N'UserDB_log', FILENAME = N'F:\SQL\UserDB_log.ldf' , SIZE = 16777216KB , FILEGROWTH =
65536KB )
GO
```

Conclusion

This reference architecture provides technical design and best practice guidance for deploying SQL Server on Pure Storage FlashArray, Pure Storage Cloud Dedicated, and FlashBlade products. SQL Server with Pure Storage delivers a powerful database and storage solution that enables organizations to focus on their core objectives, drive innovation, adapt to evolving demands, and support growth. Pure Storage advanced capabilities include versatile management options, robust data protection, high database availability, nondisruptive dev/test and upgrades, near-synchronous replication, and transparent failover. All these Pure Storage capabilities are designed with SQL Server performance, scalability, reliability, and cost savings in mind.

PS27111-02-en 09/25