Dell EMC PowerMax Storage for Mission-Critical SQL Server Databases

October 2019

H17234.1

Abstract

This white paper describes how mission-critical SQL Server databases benefit from the Dell EMC PowerMax storage system, which uses compression to deliver high performance and storage efficiency. PowerMax storage provides ease of use, reliability, high availability, security, and versatility.

VMAX and PowerMax Engineering White Paper

DELL EMC

Copyright

Contents

Executive summary

Overview

In 2018, Dell EMC introduced the PowerMax storage family with a Non-Volatile Memory Express (NVMe) back end for PCI Express (PCIe)-based access to Non-Volatile Memory (NVM) media, which includes modern NAND-based flash along with high-performance storage class memory (SCM) media. Dual-ported NVMe drives with flash-optimized protocol access provide very low latencies and extremely high I/O densities for mission-critical applications.

In Q3 2019, Dell EMC enhanced PowerMax systems to support end-to-end NVMe, SCM drives, and 32 Gb Fibre Channel (FC) NVMe to offer an unprecedented level of performance for host applications.

The Dell EMC PowerMax family includes PowerMax 2000 and PowerMax 8000 systems for scale-up and scale-out multicontroller reliability, availability, large write-cache buffering, back-end write aggregation, and security.

Benefits of PowerMax systems include:

- **Easy provisioning and storage management**—PowerMax systems use virtual provisioning to create new storage devices in seconds. All devices are thin provisioned, consuming only the storage capacity that is written to, which increases storage efficiency without compromising performance. You can put the devices into storage groups (SGs) and manage them as a unit to improve performance, quality of service (QoS), data protection, and data storage efficiency. In addition, you can manage these systems using Dell EMC Unisphere for PowerMax, the Dell EMC Solutions Enabler CLI, or REST APIs with integration with Microsoft Windows PowerShell.

  Integration with Kubernetes—The Container Storage Interface (CSI) driver for PowerMax systems enables integration with Kubernetes open-source container orchestration infrastructure and delivers scalable persistent storage provisioning operations for PowerMax all-flash arrays. See Appendix C: SQL Server on Kubernetes with PowerMax persistent volumes for more details about how customers can use the CSI driver for SQL Server deployment on Kubernetes containers.

- **High performance**—Each PowerMax brick features a multicontroller architecture, front-end and back-end connectivity, InfiniBand internal fabric, and a large mirrored and persistent cache. The brick architecture offers predictable high performance for any kind of workload including transaction processing, log writes, checkpoints, and batch processing. PowerMax systems also excel in servicing high-bandwidth sequential workloads by using prefetch algorithms, optimized writes, and fast front-end and back-end interfaces.

- **Superior data services**—PowerMax systems excel at providing data services. The systems natively protect all data with T10-DIF from the moment that data enters the array until it leaves (including replications). With Dell EMC SnapVX and Dell EMC SRDF technologies, PowerMax systems offer many topologies for consistent local and remote replications and integrations with Dell EMC Data Domain, such as Dell EMC ProtectPoint for data backups using storage snapshots. Other PowerMax

system data services include host I/O limits and other QoS features, data reduction, “call-home” support, nondisruptive upgrades, and nondisruptive migrations.

- **Built-in, real time machine learning**—PowerMax systems use automated I/O recognition and data placement across flash and SCM media to maximize performance with no additional overhead.

PowerMax systems extend the benefits of VMAX All Flash systems with high performance and low latencies for Tier 0 applications using an NVMe back end. PowerMaxOS advanced data reduction that includes inline compression and deduplication (dedupe) further improving storage efficiency without any overhead on the applications.

PowerMax systems are ideal for running SQL Server mission-critical databases, where high performance, resiliency, protection, and security are required. The following figure shows the multidimensional scalability, performance, and data services that are offered by PowerMax systems:

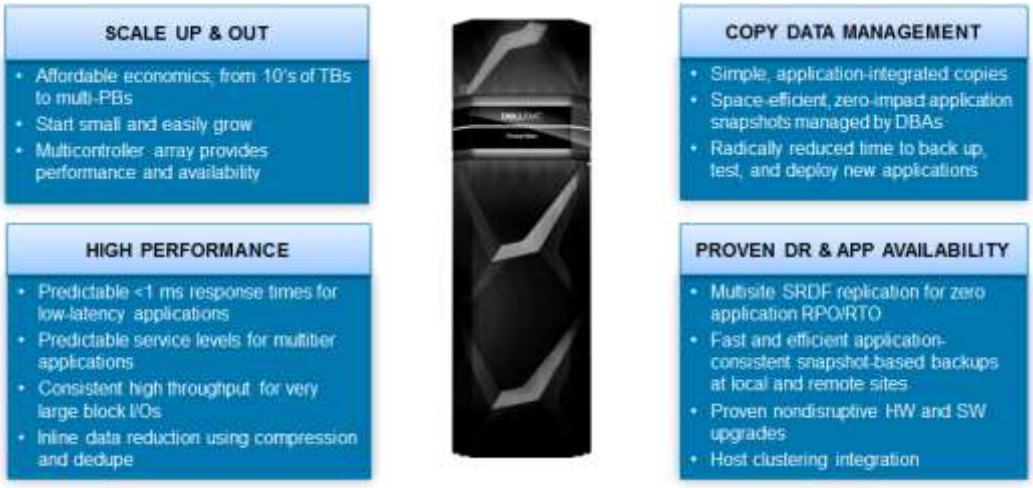

Figure 1. Multidimensional performance and scalability of PowerMax systems

This white paper describes the benefits of using Dell EMC PowerMax storage for Microsoft SQL Server databases. It includes test cases that demonstrate how to deploy PowerMax systems in a SQL Server environment for optimal performance to support mission-critical databases.

Audience

This paper is intended for database and system administrators, storage administrators, and system architects who are responsible for implementing, managing, and maintaining SQL Server databases. Readers should be familiar with SQL Server and have an interest in achieving higher database availability, better performance, and simplified storage management.

We value your feedback

Dell EMC and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell EMC Solutions team with your comments.

Author: Udgith Mankad

PowerMax features and benefits for SQL Server

Overview

SQL Server databases running on PowerMax storage systems can use all-flash storage utilizing both NAND flash and SCM media. This capability provides unparalleled levels of performance and low latency with faster transactions to ensure better business agility and improved client satisfaction. Flash storage provides better total cost of ownership (TCO) because it uses less floor space for a few high-capacity solid-state drives (SSDs) and uses less power than hard drives. Flash storage also provides consistent performance regardless of whether the I/O profile is random, sequential, intermittent, or continuous.

SQL Server databases can greatly benefit from both server-side cache and flash storage. As large as the server-side cache is, often the database capacity is even larger. While frequently accessed data fits in the database cache, there are always queries that access less-frequently needed data. Database consolidation also often means a smaller portion of the cache is used for each tenant. Finally, in a cluster, server-side cache is not cumulative, and each cluster node caches its own data regardless of others. When the requested data is not in cache, flash storage enables quick completion of I/O operations.

PowerMax systems are even faster and more reliable than other third-party flash-only systems, due to the high capacity, persistent cache, end-to-end NVMe and 32 Gb FC and FC-NVMe support. The following figure shows a high-level hardware overview of the PowerMax 2000 and 8000 systems:

Figure 2. PowerMax family

This platform complements the use of database cache with faster I/O requests for blocks that are not already in cache. PowerMax systems also provide a storage system that enables high performance, consolidation, and easy data replications for backup, high availability (HA), and disaster recovery (DR).

As adoption of flash storage grows, organizations are moving away from traditional hybrid arrays to all-flash storage arrays such as PowerMax systems. In consolidated environments, organizations must still provide prioritized data access to mission-critical

applications while minimizing the impact caused by noisy neighbors. In addition, as the application performance profile and importance changes, service levels must be set to ensure predictable and consistent performance for the applications. PowerMaxOS provides service levels at the SG level for PowerMax storage systems.

PowerMax systems provide automatic, scheduled, and application-consistent snapshots for Microsoft SQL Server and other applications for creating point-in-time copies for backup, reporting, and test/dev by using Dell EMC SnapVX local replication. Dell EMC AppSync data protection enables you to manage application snapshots with tighter integration between SnapVX and Microsoft Volume Shadow Copy Service (VSS) and SQL Server Virtual Device Interface (VDI).

PowerMax systems offer active/active high availability of storage devices at synchronous distances for Microsoft SQL Server failover clusters using SRDF/Metro. With storage devices always read/write-enabled, in the event of failover, SQL Server cluster resources can be restarted quickly, thus improving RTO and providing ease of management for SQL Server databases on a Windows Server Failover Cluster.

PowerMax systems can perform data compression to significantly increase the effective capacity of the array. With the system's fine-grained data packing and activity-based hardware-accelerated compression capabilities, all application environments can achieve storage efficiency with optimum performance.

PowerMaxOS enhances the data compression capability even further. PowerMaxOS uses data reduction hardware that is available exclusively on PowerMax systems, further reducing the processing overhead for data reduction. Also, on PowerMax systems exclusively, PowerMaxOS provides inline data dedupe that is enabled whenever compression is enabled on an SG.

PowerMax NVMe back end

The PowerMax architecture incorporates an NVMe back end that reduces I/O latency and increases data throughput while maintaining full redundancy. NVMe is an interface that enables host software to communicate with a nonvolatile memory subsystem. The interface is optimized for SSDs and is typically attached as a register-level interface to the PCIe interface.

The NVMe back-end subsystem provides redundant paths to the data that is stored on SSDs. This redundancy provides seamless access to information even if a component fails or is being replaced.

Each PowerMax disk array enclosure (DAE) can hold twenty-four 2.5-inch NVMe SSDs. The DAE also houses redundant canister modules (link control cards, or LCCs) and redundant AC/DC power supplies with integrated cooling fans.

The back-end directors are connected to each DAE through a pair of redundant back-end I/O modules. The back-end I/O modules connect to the DAEs at redundant LCCs. Each connection between a back-end I/O module and an LCC uses a completely independent cable assembly. Within the DAE, each NVMe drive has two ports, each of which connects to one of the redundant LCCs. PowerMax systems use an active/active RAID group accessing scheme called Smart RAID. As shown in the following figure, this scheme enables RAID groups to be shared across directors, giving each director active access to all drives on the brick or zBrick.

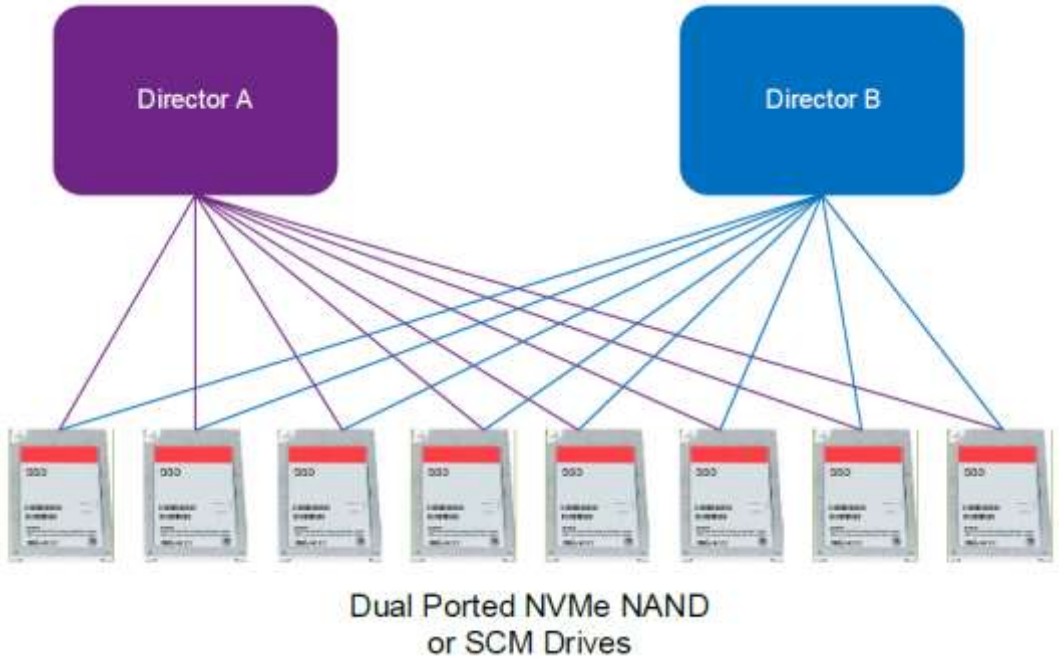

Figure 3. PowerMax Smart RAID for dual-ported drive access

The dual-initiator feature ensures continuous availability of data in the unlikely event of a drive-management hardware failure. Both directors within an engine connect to the same drives through redundant paths. If the sophisticated fencing mechanisms of PowerMaxOS detect a failure of the back-end director, the system can process reads and writes to the drives from the other director within the engine without interruption.

The following figure summarizes the benefits of NVMe on the PowerMax system and the benefits that are realized by the application:

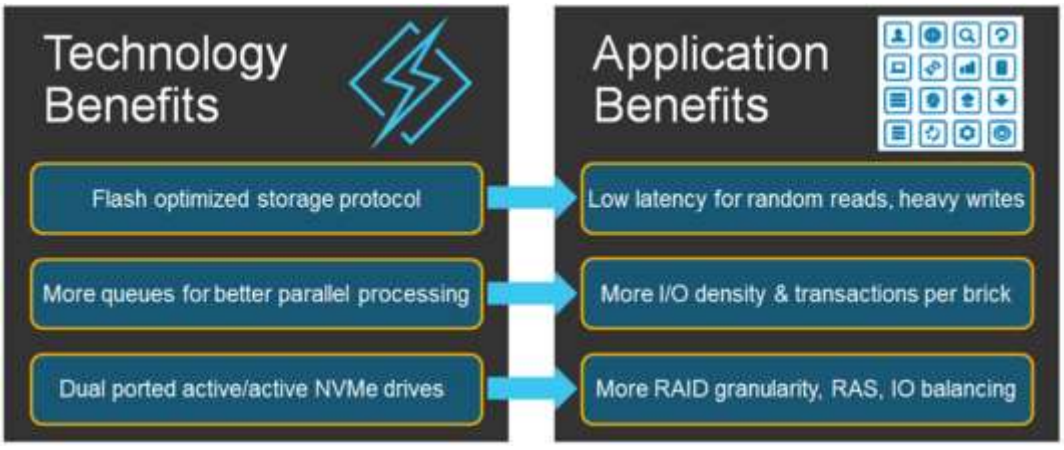

Figure 4. Benefits summary of end-to-end NVMe system

PowerMax NVMe over fabric

PowerMax support for running NVMe over 32 Gb Fibre Channel combined with SCM enables lower latency and best response times for demanding applications. This PowerMax FC-NVMe feature delivers end-to-end NVMe and is compatible with Connectix

32 Gb NVMe switches and directors and 32 Gb NVMe server HBAs. The combination of end-to-end NVMe and SCM technology delivers true breakthrough performance.

Note: PowerMax 32 Gb I/O modules also support FC host connectivity.

Adaptive Compression Engine

PowerMax Adaptive Compression Engine (ACE) offers high performance and maximum storage efficiency for application environments. ACE compresses data and efficiently optimizes system resources to balance overall system performance. ACE features include:

- **Hardware acceleration**—The PowerMax system is equipped with data reduction hardware that is configured with one module per director or two modules per engine. The modules help reduce the processing overhead for supported data reduction functions.
- **Selective compressio**n—With ACE, you can decide, at the SG level, which data to compress and when to enable or disable compression. As an example, to prioritize workloads, you might decide not to use compression. Enabling compression for an SG compresses the candidate data in the background with the system still active. Likewise, disabling compression for an SG does not immediately start a decompression process; rather, it decompresses data when accessed and over time.
- **Activity-based compression**—Inline compression and advanced compression algorithms and hardware acceleration provide space savings, as you would expect. In addition, ACE uses activity-based compression (ABC) to determine which data to compress. ABC helps the system to mitigate compression overhead on the system and on frequently accessed data. ABC prevents constant compression and decompression of data that is frequently accessed.

  ABC marks the busiest data in the storage resource pool (SRP) to skip the compression flow regardless of the related SG compression setting. This function differentiates busy data from idle or less-busy data and only accounts for up to 20 percent of the allocations in the SRP. It marks up to 20 percent of the busiest allocations to skip the compression action, ensuring optimal response time and reducing the overhead that can result from the act of compressing data to save space.

  The mechanism that determines the busiest data does not add CPU load to the system. ABC uses statistics that are collected from front-end devices to determine which datasets are the best candidates for compression. The system can then maintain balance across the resources, providing an optimal environment for both the best possible compression savings and the best performance. Effectively, this balance avoids compression and decompression latency for the busiest data and reduces system overhead.

- **Fine-grained data packing**—ACE uses data reduction hardware to process incoming 128 KB I/O into four sections. Each section is compressed individually and in parallel, which maximizes the efficiency of the data reduction module.

  Fine-grained data packing offers performance benefits for both the compression function and the overall performance of the system. Included in this process is a

zero reclaim function that prevents the allocation of buffers with all zeros or no actual data. Pairing the zero reclaim function with fine-grained data packing enables the compression function to operate efficiently with minimal impact on performance. The 128 KB I/O are compressed in four buffers individually, and in parallel, enabling each section to be handled independently, although they are still part of the initial 128 KB I/O. The main benefit comes in the case of partial write updates or read I/O. If only one or two of the sections are updated or read, only that data is decompressed.

- **Extended Data Compression (EDC)** —With PowerMax EDC, data that is already compressed might qualify for additional compression savings based on background compression. Data that is part of a compression-enabled SG, is not already compressed by EDC, and meets a 30-day idle period requirement qualifies for additional compression savings.

Exclusive data deduplication

The PowerMax system further improves data reduction by introducing deduplication (dedupe). Dedupe improves storage utilization without compromising I/O performance. It works by generating unique hash IDs at the time of data ingress and comparing those IDs with existing hash IDs before storing the data on the disk. Dedupe is accomplished through a series of features including:

- **Hardware acceleration**—PowerMax data reduction hardware used by ACE also performs an inline dedupe function. All incoming data is passed through this hardware to generate a unique 32-byte hash ID for each 32k block of data using a Secure Hash Algorithm (SHA-2). This process occurs before data is stored on physical disks or even before data is compressed.
- **Hash table**—The hash table uses PowerMax system memory to store the unique hash ID. A dedupe relationship is generated whenever a matching hash ID for incoming data is found.
- **Dedupe Management Object (DMO)**—The DMO is a 64-byte object within system memory. DMOs only exist when dedupe relationships exist. These objects store and manage the pointers between front-end devices and the single instance of data that is stored at the back end of the array.

Host writes for any compression-enabled SGs go to persistent cache and are acknowledged immediately. Before the data is destaged to the physical media, the compression module generates the hash ID. This hash ID is checked for an existing entry in the hash table. If the match is found, only the pointers are updated to reference existing data. If the matching hash ID is not found, data is written and the thin device's pointers are updated.

The following figure shows the dedupe workflow for all writes for a compression-enabled SG:

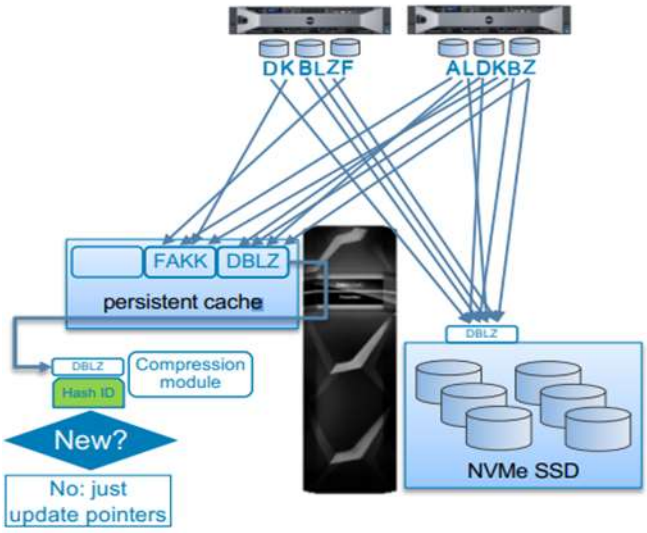

Figure 5. PowerMax dedupe workflow

PowerMaxOS automated data placement

On single-tier-based PowerMax all-flash systems, PowerMaxOS service levels enable performance management of all the data for each SG without having to move any of the data. The latest PowerMaxOS release includes support for a tiered storage model called automated data placement (ADP). ADP is enabled by the PowerMaxOS service levels and augmented by machine learning to place the most active data on faster storage drive technology.

PowerMaxOS uses real-time machine learning to model workload characteristics. This model provides a predictive function that enables PowerMaxOS to anticipate workload demand for a storage group. With these anticipated workload demands, PowerMax OS can adapt as necessary to changes in block size, write ratio, or I/O load. ADP movements occur as either promotion of active data to SCM or demotion from an SCM storage tier. The service levels are defined with target response time characteristics. Service levels also have either an upper response time limit or both upper and lower response time limits to avoid the over-allocation of shared resources for low-priority applications.

A storage group with a higher-priority service level that is affected by any lower-priority storage groups triggers response-time management to the lower-priority service levels. When the higher-priority storage group reaches its target response time, all lower storage groups continue to be managed until the lowest-priority storage groups reach their target response time. The management of any lower-priority service level is imposed by a response-time delay in I/O. The delay gradually increases over time to keep the higher-priority storage group within its respective target response time. The delay gradually decreases to ensure that the higher-priority storage group remains within its response time.

PowerMaxOS offers the following service levels and associated response times. It also sets service-level bias to apply promotion and demotion priorities.

- **Diamond**—This service level has the highest promotion priority and lowest response times. During optimal utilization, PowerMaxOS attempts to put all

Diamond-labeled data on SCM drives. Data is demoted when there is need for more-active Diamond data to be promoted or if the SCM usage exceeds pool-reserved capacity.

- **Platinum, Gold, and Optimized**—All these service levels have same priority. Active data extents using any of these service levels are promoted when possible given the space usage in an SCM tier. Demotion occurs when space is needed in an SCM tier for higher priority data or for more active data with the same priority.
- **Silver, and Bronze**—These service levels have lower priorities and higher target response times. SGs using these service levels are not chosen for promotion to an SCM tier.

The following figure illustrates PowerMaxOS service levels:

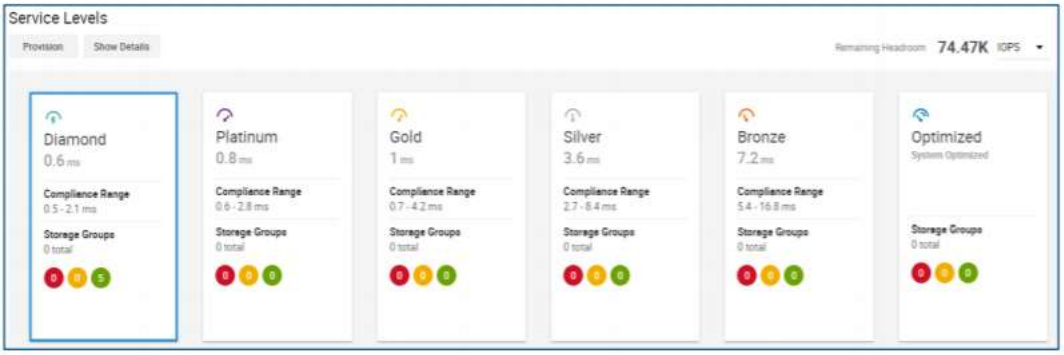

Figure 6. PowerMaxOS service levels

Key benefits of PowerMax systems for SQL Server databases

Several features of PowerMax systems provide exceptional benefits when used for SQL Server databases:

- Dynamic random-access memory (DRAM)-based cache is large, persistent, and mirrored, which enables all writes, including SQL Server log writes and batch loads, to be completed faster than writing directly to SSD.
- Dell EMC TimeFinder SnapVX replication provides storage-consistent snapshots. SnapVX technology provides high levels of scale, efficiency, and simplicity. It uses redirect-on-write for added performance and pointer-based management for dedupe-like data reduction. Snapshots have names, versions, dates, and automatic expiration dates for easy identification and management. Snapshots are protected, so they can be reused regardless of changes by the application. They can also be cascaded any number of times. With SnapVX technology, you can create SQL Server database replicas for gold copies, test or development environments, and backups. Snapshots can be restored in seconds and read/write access to snapshot data is always immediate.
- Dell EMC Symmetrix Remote Data Facility (SRDF) offers many DR topologies for SQL Server databases, including those for two, three, and four sites. SRDF can replicate in synchronous and asynchronous modes and offers multisite cascaded and star configurations. SRDF is closely integrated with SnapVX software to offer a variety of HA and DR solutions including fast recovery from remote copies and backup offload.

- Dell EMC AppSync data protection software offers self-service automated copy management for Microsoft SQL Server and other application environments. It provides automatic application discovery and mapping, storage provisioning and selection for local and remote replications, and easy repurposing for testing, development, and reporting. Copies can be used for instantly restoring a production database. Database recovery from such copies can be started immediately after initiating the restore, improving recovery time object (RTO) greatly for mission-critical applications.

Configuration best practices and design considerations

Introduction

PowerMax systems have simplified storage provisioning. This section describes configuration best practices and design considerations for storage connectivity and provisioning with PowerMax systems and Microsoft SQL Server.

Storage design

PowerMax systems are preconfigured with your specified capacity, connectivity options, and storage RAID protection. The SSD drives are spread across the back end and provisioned into logical devices called thin data devices (TDATs). The devices are placed in RAID groups and combined to become an SRP. When host thin devices (TDEVs) are later created, their capacity is consumed in the SRP. Most systems have a single SRP unless a specialized configuration is required.

PowerMax system architectures are based on units known as bricks. A brick includes an engine, which consists of two directors. Each director includes cache, ports, and emulations based on the services and functions that the storage was designed to provide. For example, each director includes ports supporting Fibre Channel (FC) or gigabit Ethernet, and emulations, such as front-end, back-end, iSCSI, eNAS, and SRDF. Each director also includes CPU cores that are pooled to be used with all emulations and serve all director ports. While the default balanced core allocation is usually best, Dell EMC support can deploy other methods if core use across emulations is constantly unbalanced.

PowerMax systems come preconfigured, so that when they are powered up, activities can immediately focus on physical connectivity to hosts, zoning, and storage provisioning to the database servers. Factory pre-configuration makes the deployment fast, easy, and focused on the application needs, rather than on the storage configuration.

Storage connectivity best practices

When planning storage connectivity for performance and availability, connecting storage ports across different engines and directors is better than using all the ports on a single director. In this way, even if a component fails, the storage can continue to service host I/Os.

PowerMax systems use dynamic core allocation. Each director provides services such as front-end connectivity, back-end connectivity, or data management. Each such service has its own set of cores on each director. The cores are combined to provide CPU resources that can be allocated as necessary. For example, even if host I/Os arrive through a single front-end port on the director, the front-end pool with all its CPU cores is available to service that port. Because I/Os arriving at other directors have their own core pools, we recommend connecting each host to ports on different directors before using

additional ports on the same director. This practice ensures high performance and availability.

Virtual Provisioning and thin devices

All PowerMax system host devices use Virtual Provisioning technology. The devices are a set of pointers to capacity allocated at 128 KB extent granularity in the SRPs; however, to the host they look and respond just like regular LUNs. Using pointers also increases capacity and efficiency for SnapVX local replication by sharing extents when data does not change between snapshots.

Virtual Provisioning technology offers a choice of whether to fully allocate the host device capacity or to allow allocation on demand.

- A fully allocated device consumes all its capacity in the SRP on creation, and therefore, there is no risk that future writes might fail because the SRP has no remaining capacity.
- Allocation on demand enables over-provisioning. Although the storage devices are created and appear to the host as available with their full capacity, actual capacity is allocated in the SRP only when host writes occur. This is a common cost-saving practice.

Use allocation on demand when:

- You do not know the capacity growth rate for an application.
- You do not want to commit large amounts of storage ahead of time because it might not be used.
- You do not want to disrupt host operations at a later time by adding more devices or expanding the capacity of the existing devices.

If allocation on demand is used, capacity is only physically assigned as needed to meet application requirements.

When data files are created, Microsoft SQL Server pre-allocates capacity by writing to every page with contiguous zeros. When allocation on demand is used, it is best to increase database capacity over time, based on actual need. For example, with SQL Server provisioned with a thin device of 2 TB, the DBA—rather than immediately creating data files of 2 TB and consuming all its space—could use an auto-growth feature that consumes only the needed capacity.

Note: When Windows Instant File Initialization (IFI) is used, allocation of data files occurs in a thin-pool-friendly way. Areas of a disk under which a sparse file is defined, as created by IFI, are not zeroed. As table and index information is written to a fully initialized data file, areas of the database become allocated and used by non-zero user data. SQL Server automatically uses Windows IFI if the service account under which the SQL Server service is running has Perform volume maintenance tasks permission under the local security policy. By default, only administrators have this permission. Information about the IFI is provided in the Microsoft SQL Server Books online product documentation. Transaction logs remain fully allocated even with IFI to avoid marginal performance penalties that might result during additional thin pool allocations.

Considerations for PowerMax data reduction with SQL Server

The following considerations apply when you use PowerMax data reduction for SQL Server:

- **Efficiency using thin provisioning**—PowerMax systems are fully thin provisioned storage systems providing the highest levels of storage efficiency. SQL server SGs consume storage capacity only for the data that is actually written by the host and grow as needed with host writes. Storage efficiency is already realized when SQL Server SGs are deployed on PowerMax devices. All PowerMax data services continue to offer highly efficient data copy and capacity utilization even when you use PowerMax SnapVX for periodic snapshots or SRDF for remote replication.
- **Efficiency using PowerMax compression**—PowerMax ACE splits the 128 KB track into four 32 KB buffers and acts on them independently, making the compression process highly efficient. SQL Server I/O ranges between 8 KB and 256 KB with 8 KB to 32 KB write sizes most common for OLTP workloads; therefore, contiguous writes to the same track are considered for the compression analysis. In addition, data is decompressed at the same 32 KB granularity, improving the latency for activities with high locality of reference because data might be already available in PowerMax cache from prior activity on the track. Data that is already compressed and does not exceed the activity thresholds assigned by PowerMax compression algorithms is compressed in line and written to the thin pools, resulting in excellent efficiency.
- **Efficiency using PowerMax dedupe**—PowerMax dedupe uses the hash IDs that are generated on 128 KB track writes. Microsoft Windows NTFS uses 64 KB alignment. Due to the misalignment with extent size and NTFS allocation unit, the likelihood of dedupe on subsequent writes to the same device is low. However, multiple copies of SQL databases are always created on the PowerMax system for backup, test, dev, and reporting instances. These additional copies would certainly realize the dedupe benefit because the hash IDs for the copies would match either the source or some of the target extents that are associated with other copies. When such matching IDs are found in the hash table, a dedupe relationship is created.
- **Dynamically changing compression settings on SGs**—PowerMax compression is configured system-wide, but it is enabled or disabled at the SG level and can be changed dynamically. However, because PowerMax compression is activity-based, the effect might not be immediately noticeable. To minimize the impact on other workloads, compression-related movements occur at lower priority. Once the compression ratio is set on the SG, disabling it is possible but not necessary. The PowerMax compression engine compresses and decompresses data as needed to meet application performance requirements even when accessing a compressed dataset. PowerMax dedupe also relies on the compression setting on the SG; therefore, disabling compression on an SG also prevents the SG devices from realizing the dedupe benefit, and degrades the data efficiency.
- **Expectation of the compression ratio**—Compressibility greatly depends on the dataset. Most live SQL Server databases exhibit compression ratios from 1.3 to 3.0, with compressible datasets on the PowerMax system likely resulting in a higher compression ratio due to better hardware acceleration. Unisphere for PowerMax can provide an indication of SG compressibility for PowerMax systems. With tools

such as Dell EMC Live Optics, you can estimate compression ratio targets by collecting and analyzing statistics on random samples of the devices. After you configure the PowerMax system for compression, you can configure individual SGs for compression, and they will achieve their target compression ratio over time. Higher compressibility can potentially be achieved when the dataset has large blocks of fillers or blob data with a lot of cyclic/repeated patterns. PowerMax ACE determines the compressibility of the SG by scanning the contents and transparently moving the application data to highly compressible storage pools, improving the compression ratio of the SG. When the workload is run, compression targets are maintained because ACE decompresses the active dataset and assigns scores to the extents for further compressibility analysis. Less active extents remain on the compression storage pools that meet their potential compressibility.

- **Compressibility of logs**—SQL Server active log devices generally have lower compressibility, but as the logs fill up and are archived, the inactive nature of the archived logs results in higher compressibility of the log devices.
- **SQL Server data and log devices in cascaded configuration**—ACE collects statistics on 32 KB data extents for compressibility. Although SQL Server storage provisioning uses child SGs for data and log devices for data protection and manageability, these devices can be part of the same parent SG that has compression enabled. ACE still operates at the device level and achieves compression of various SQL Server objects in the most optimal fashion. Thus, best practices for storage provisioning with PowerMax compression are the same as the best practices associated with other PowerMax data services such as SnapVX and SRDF.
- **SQL Server compression and encryption**—SQL Server allows database table-level compression for rows and pages. Table-level compression is set up using the Data Compression Wizard. The compressibility is slightly better with SQL Server compression compared to PowerMax compression because SQL Server compression uses Microsoft proprietary technology that is based on SQL Server and it is aware of the block layout at the SQL Server level. However, SQL Server compression puts a greater demand on the host CPU to process compression and decompression on already compressed data that becomes active; thus, SQL Server-based compression affects the overall performance of the database and other applications running on the server. PowerMax compression is set up at the SG level and works within the PowerMax frame, benefitting all the database objects that are part of the SG, including the data, logs, indexes, and any support files associated with the application and database. When PowerMax compression is used, even external blob/object stores can be compressed without any impact to overall performance.

  Although SQL Server and PowerMax compression can co-exist, any application performance benefits are realized at the expense of host CPU overhead resulting from SQL Server compression. The benefits are also short-lived because SQL Server reads in more data to decompress to align with server application needs. As the data is aged and the cache needs to be refreshed, any benefits are diminished and performance lowers to the same level as using only PowerMax compression, which incurs no CPU overhead. For optimal performance, if you have a choice, use PowerMax compression rather than SQL Server compression.

You can also use PowerMax compression when SQL Server data, connections, and stored procedures use SQL Server encryption. The compressibility of the data depends on the amount of redundancy or repeated patterns, but there is no reason not to use PowerMax compression even when the database is encrypted. Even if compressibility of the data is lower due to obfuscated data patterns, there is no overhead associated with compression, so application performance is not affected. In addition, using PowerMax compression on the storage enables the use of Data at Rest (D@RE) encryption, with an integrated/external key manager, which allows storage-level encryption for the highest level of security.

Host connectivity

Host bus adapter (HBA) ports (initiators) and storage ports (targets) are connected to an FC or Ethernet switch based on the connectivity requirements. FC connectivity requires that you create zones on the switch and define initiator and target relationships. The zones create an I/O path between the host and storage. Zoning and paths strongly affect performance aspects of the host and database.

FC connectivity best practices

Best practices for FC connectivity include:

- When zoning host initiators to storage target ports, ensure that each pair is on the same switch. Performance bottlenecks are often created when I/Os must travel through ISL (paths between switches), which are shared and limited.
- Use at least two HBAs for each database server to enable better availability and scale. Use multipathing software such as Dell EMC PowerPath or Microsoft Windows MPIO to balance loads and automatically failover or recover paths. Use commands such as `powermt display paths` or `multipath -l` to ensure that all paths are visible and active.
- Consider port speed and count when planning bandwidth requirements. Each 8 Gb FC port can deliver up to about 800 MB/sec. Therefore, a server with four ports cannot deliver more than about 3 GB/sec. Also consider that among the host initiator, storage port, and switch, the lowest speed supported by any of these components is negotiated and used for that path.
- Consider the number of paths that are available to the database storage devices. Each path between the host and storage ports creates additional SCSI representation for the database devices on the host.
- While more paths add I/O queues and the potential for more concurrency and performance, consider that server boot time is affected by the number of discovered SCSI devices (one for each path combination per device, plus a pseudo device). In addition, after connectivity needs are satisfied for performance and availability, additional paths do not add more value and only add more SCSI representations to the host.
- In most cases, for availability and performance, it is sufficient to have each HBA port zoned/masked to two or four PowerMax ports, preferably on different engines and directors.

iSCSI connectivity best practices

- Use VLANs dedicated to iSCSI setup. VLANs allow logical grouping of network endpoints, which minimize network bandwidth contention for iSCSI traffic and eliminate impact on iSCSI traffic due to noisy neighbors.
- If all network devices in the iSCSI communication paths support jumbo frames, using jumbo frames on Ethernet improves iSCSI performance.
- To minimize host CPU impact due to network traffic, ensure that Transmission Control Protocol (TCP) offloading is enabled on a host network interface card (NIC), which offloads processing of the TCP stack to the NIC and eases impact on the CPU.
- As with FC connectivity, using PowerPath software or native multipathing for Windows helps with load balancing and eases queuing issues for iSCSI traffic through the host NICs.

Number and size of host devices

- PowerMax systems use thin devices exclusively. Therefore, on creation, devices do not fully allocate their capacity in the SRP, and they only consume as much capacity as the application actually writes to them. For example, by default, a 1 TB device that has not been written to does not consume any storage capacity. This approach enables capacity savings because storage is consumed based on demand and not during device creation. However, if certain applications require a guarantee for their capacity, their devices can be fully created and allocated in the SRP. New devices can be created easily using Unisphere for PowerMax or by using CLI, as in the following example:

```
PS > symdev create -tdev -cap 500 -captype gb -N 4 -v #
create 4 x 500GB thin devices
```

- PowerMax host devices are natively striped at 128 KB across the data pools in the SRP. Although you can create only a few very large host devices, consider these factors:
  - As discussed previously, each path to a device creates a SCSI representation on the host. Each such representation provides a host I/O queue for that path. Each such queue can service a tunable, but limited (often 32), number of I/Os simultaneously. Provide enough database devices for concurrency (multiple I/O queues), but not so many that management overhead would be increased.
  - Another benefit of using multiple host devices is that internally the storage array can use more parallelism for operations such as data movement and local or remote replications. By performing more copy operations simultaneously, the overall operation takes less time.
  - While the size and number of host devices can vary, we recommend finding a reasonable, low number that offers enough concurrency, provides an adequate building block for capacity increments when additional storage is needed, and does not become too large to manage. For example, four or eight devices, sized at 250 GB to 1 TB (choose one size) can be a good design starting point for databases from 1 to 8 TB, if enough connectivity/concurrency exists.

Physical host connectivity best practices

For physical host connectivity, consider the number and speed of the HBA ports (initiators) and the number and size of host devices:

- **HBA ports**—Each HBA port (initiator) creates a path for I/Os between the host and the SAN switch and then to the PowerMax storage. If a host uses a single HBA port, it has a single I/O path that must serve all I/Os. Such a design is not advisable because a single path does not provide HA and risks a potential bottleneck during high I/O activity due to the unavailability of additional ports for load balancing.

  For a better design, provide each database server with at least two HBA ports, preferably on two separate HBAs. The additional ports provide more connectivity and enable multipathing software such as Dell EMC PowerPath or Microsoft Multipath I/O (MPIO) to balance loads and to fail over across HBA paths.

  Each path between the host and storage device creates a SCSI device representation on the host. For example, two HBA ports connected to two PowerMax front-end adapter ports with a 1:1 relationship create three presentations for each host device. One port is used for each path, and the multipathing software uses the other port to create a Dell EMC Symmetrix Multi-Path Disk Device (PowerPath System Devices). If each HBA port was zoned and masked to both FA ports (1: many relationship) there would be five SCSI device representations for each host device (one for each path combination plus pseudo device).

  While modern operating systems can manage hundreds of devices, it is not advisable or necessary, and it burdens the user with complex tracking and storage provisioning management overhead. We recommend that you:

  - Establish enough HBA ports to support workload concurrency, availability, and throughput
  - Use 1:1 relationship for storage front-end ports
  - Do not zone or mask each HBA port to all PowerMax front-end ports

  Following these suggestions will provide enough connectivity, availability, and concurrency, while reducing unnecessary complexity.

- **Number and size of host devices**—PowerMax can create host devices with capacity ranging from a few megabytes to multiple terabytes. With the native striping across the data pools that PowerMax provides, the DBA might be tempted to create only a few very large host devices. For example, a 1 TB Microsoft SQL Server database can reside on one 1 TB host device, or perhaps on ten 100 GB host devices; while either option satisfies the capacity requirement, you should use a reasonable number of host devices of appropriate size. In this example, if the database capacity was to rise above 1 TB, the DBA might want to add another device of the same capacity, even if 2 TB was not currently needed. Therefore, large host devices create very large building blocks when additional storage is needed.

  Each host device also creates its own host I/O queue for the host operating system. Each such queue can service a tunable, but limited, number of I/Os that can be transmitted simultaneously. If, for example, the host has four HBA ports and a single 1 TB LUN with multipathing software, it will have only four paths available to

queue I/Os. A high level of database activity generates more I/Os than the queues can service, resulting in artificially elongated latencies. In this example, two or more host devices are advisable to alleviate such an artificial bottleneck. Host software such as Dell EMC PowerPath or Windows PerfMon can help in monitoring host I/O queues to ensure that the number of devices and paths is adequate for the workload.

Another benefit of using multiple host devices is that, internally, the storage array can use more parallelism when operations such as FAST data movement or local and remote replications take place. By performing more copy operations simultaneously, the overall operation takes less time.

While there is no one magic number for the size and number of host devices, we recommend finding a reasonably low number that offers enough concurrency, provides an adequate building block for capacity when additional storage is needed, and does not become too large to manage.

Host I/O limits and multi-tenancy

The host I/O limits QoS feature was introduced in VMAX arrays, and it continues to provide the option to place specific IOPS or bandwidth limits on any SG. For example, assigning a specific host I/O limit for IOPS to an SG with low performance requirements can ensure that a spike in I/O demand will not saturate or overload the storage, affecting performance of more critical applications. As shown by test cases later in this paper, even though PowerMax systems can maintain high performance at low latency, using a host I/O limit may limit the impact of noisy neighbors on mission-critical applications.

Linux support with SQL Server 2017 and PowerMax with NVMeoF

Introduction

For the first time, the power of SQL Server has gone beyond Windows to support Linux and Docker containers. SQL Server 2017, with its support for Linux and Docker, is truly a platform with choice: choice of development languages, data types, and operating systems, all with support for Big Data and advanced analytics. With SQL Server now available on Linux, users can develop once and deploy SQL Server based applications anywhere, from on-premises to the cloud, with a consistent experience.

With the PowerMax Q3 2019 release, SQL Server running on Linux can also leverage supported 32 Gb HBAs for end-to-end NVMe for high demand host applications with best response times.

Advantages of SQL Server on Linux

Because SQL Server 2017 runs on Windows, Linux, and Docker containers, you can deploy your application on the platform of your choice, or on a combination of platforms that makes the most sense for your business.

Benefits of SQL Server with Linux support include:

- The platform is easily integrated with existing open source platforms.
- SQL Server with Linux supports an extended range of platforms on which to develop applications.
- Integration with Microsoft Active Directory is seamless, providing better and integrated control over security by using a single security platform for both Windows and Linux. You can use Active Directory authentication to centralize the identities of database users and other services in one location. In this manner, you can simplify permission management and avoid storing passwords.
- Using the same set of tools (SQL Server Management Studio) for both Linux and Windows reduces the cost of ownership by eliminating the need for additional hardware and software licenses.
- By not choosing a traditional Linux database, you can save three to six months of learning curve and gain strong security and reliability.
- Migrating data from Windows SQL Server to Linux is as simple as a backup and restore operation.
- Unlike MySQL and MariaDB, installation of third-party tools is not required.
- All enterprise-level features, such as data compression, column store, partitioning, high availability, and DR are included, so your organization can provide robust, data-driven applications to customers for a fraction of the cost of the competition.
- The installation process offers a native Linux experience for users through package-based installation methods such as Yum, apt-get, and RPM.
- Containers enable you to quickly deploy multiple database instances: First, make a copy of data and log files, then launch a new container and attach volumes to it.
- With support for Windows and Linux containers, SQL Server can run in container orchestration solutions such as Docker Swarm, Red Hat Open Shift, Mesosphere DC/OS, and Kubernetes.

The total cost of ownership is lower than that of other Linux-based enterprise databases.

Limitations of SQL Server on Linux

Limitations of SQL Server on Linux include:

- SQL Server Reporting Services (SSRS) is not available for installation on Linux.
- SQL Server Management Studio (SSMS) must be on the Microsoft Windows platform.
- SQL Server on Linux currently supports ext4 and XFS. Microsoft will add support for other file systems later, as needed.
- In our testing, SQL Server on Linux and Linux Docker containers appeared to consume CPU and memory resources similar to Windows 2017 for the same level of database performance. SQL Server on Linux consumed 11 percent more CPU compared to Windows for the same number of transactions per second. Minimum CPU and memory requirements for the Linux installation of SQL Server are higher compared to Windows. For up-to-date information on software and hardware requirements, see the latest SQL Server release notes.

SQL Server on PowerMax test cases

Introduction

We ran several test cases to demonstrate the capabilities of the PowerMax system for performance. For all these tests, we used a standard benchmarking tool to drive workloads from SQL Server instances. While the workloads ran, we monitored the database transaction rate (TPM) and Windows Server performance, then we charted and analyzed the results.

Our tests demonstrate a minimal performance configuration of a single-brick PowerMax 8000 with 30 NAND-Flash drives and 8 SCM drives. The results show how, in this small configuration, multiple SQL Server database workloads can achieve predictable performance when using PowerMax. PowerMax release Q3 2019, with end-to-end NVMe, 32 Gb FC and SCM drive support, provides the highest level of performance at very low response times for mission-critical SQL Server databases. We used the following SQL Server test cases:

- **Test case 1**—Quality of Service (QoS) test using a single SQL Server running at the Diamond service level. This test case shows how PowerMax with SCM drives provides the highest level of performance with very low response times.
- **Test case 2**—Scale test using multiple SQL Server databases running at Diamond service levels. This test case shows how PowerMax can effectively support multiple demanding applications while maintaining high performance and low latencies.
- **Test case 3**—Performance upgrade test for applications to support special performance requirements for quarter-end and other demanding work flows. This test case shows how effectively application service levels on PowerMax can be managed to support high performance requirements for special purposes.
- **Test case 4**—Onboard test for applications in a consolidated environment. This test case shows how effectively new upcoming applications can be consolidated on PowerMax alongside existing workloads.

- **Test case 5**—Noisy neighbor test for Predictable performance of mission critical applications. This test case shows the effectiveness of PowerMax systems in managing noisy neighbors with minimal or no impact to mission-critical applications.
- **Test case 6**—End-to-end NVMe test for SQL Server running on Linux. This test shows high performance at the best response times for the applications.
- **Test case 7**—Bandwidth test for SQL Server DSS environments running on PowerMax. This test demonstrates the high level of sequential read bandwidth available on the PowerMax platform for decision support systems and data warehouse environments.

Test environment

The following figure is a high-level depiction of the test environment. The environment consists of multiple Dell EMC PowerEdge R740 servers. Each host uses two dual-port HBAs, so four initiators per server are connected to the SAN switches. The database uses an 8 KB block size and a very small buffer pool to generate as many I/Os as possible to test the PowerMax storage capabilities.

The following table provides the details of the testbed's hardware and software components.

Table 1. Hardware and software components

Category	Type	Quantity/size	Version/release
Storage system for testing	Dell EMC PowerMax 8000	• 1 PowerMax brick • 512 GB usable cache • 30 NVMe NAND flash drives in RAID 5 • 8 SCM drives in RAID 5	PowerMaxOS 5978 based on Q3 2019 release
Database servers	Dell EMC PowerEdge R740	2 stand-alone servers	Windows Server 2016
Databases	SQL Server on stand-alone servers	2 instances	Microsoft SQL Server 2017

Test case 1: Quality of Service (QoS) test

Objective

Test case 1 demonstrates exemplary Quality of Service (QoS) for a demanding SQL Server database running an OLTP workload on the PowerMax.

Test configuration

We ran a single SQL Server database OLTP workload with a SQL Server buffer pool size of 4 GB. TheSQL Server database was set to the Diamond service level. While the test ran, we monitored host IOPS, SQL Server Transactions Per Minute (TPM), and read and write response times.

Test results

As the test results in the following figure show, the SQL Server database at the Diamond service level serviced by SCM drives coupled with the PowerMax cache provided very high performance at unrivaled low latencies.

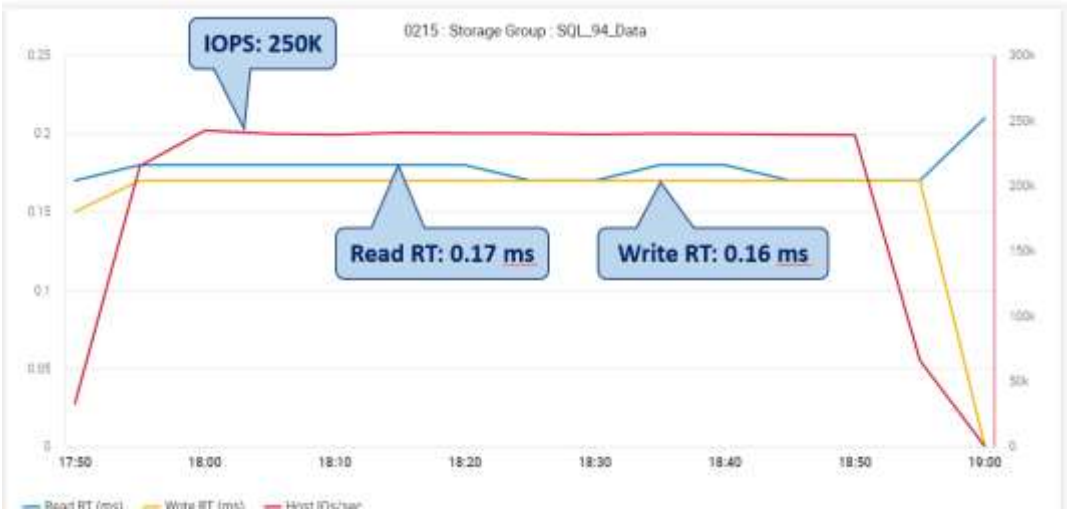

Figure 7. QoS Test for single SQL Server database

The test produced an unprecedented read response time of 0.17 ms and a write response time of 0.16 ms. All components of the storage system were still well balanced and capable of supporting other workloads. Due to the very low response times, SQL Server database on PowerMax allowed for host IOPS of 250 K with a SQL server database OLTP TPM of 2.5 M.

Test conclusion

The PowerMax system with SCM drives and other performance features can provide unparalleled levels of IOPS at very low latencies for mission-critical applications.

Test case 2: Scale test

Objective

Test case 2 demonstrates the ability of PowerMax to scale for SQL Server databases supporting two different applications that are running OLTP workloads.

Test configuration

We ran two OLTP workloads on SQL Server databases at Diamond service levels running on two identical servers. While the test ran, we monitored host IOPS, SQL Server TPM, and read and write response times.

Test results

As the test results in the following figure show, PowerMax supported both highly demanding workloads very effectively. Both databases performed in a balanced fashion with overall cache read hits of 47%.

Figure 8. Scale test for two independent SQL Server databases

Test case 2 demonstrated read and write response times of around 0.4 ms for both databases. Even with such low response times, SQL Server databases on PowerMax allowed for host IOPS of 430,000 for both SQL server databases and OLTP TPM of over 2 M each. The total TPM for both databases was close to 4.3 M.

Test conclusion

The PowerMax system can support multiple highly demanding workloads simultaneously while providing consistent performance. Comparing the results with Test Case 1, we can see that the PowerMax system scales well. Even with a limited number of SCM drives (8 in our test case), multiple competing workloads can still achieve high performance at low latencies.

Test case 3: Performance upgrade test

Objective

Test case 3 demonstrates that application performance can be upgraded on demand using PowerMax service levels.

Test configuration

We ran a single SQL Server database OLTP workload with a SQL Server buffer pool size of 4 GB. Starting at the Silver service level, we gradually upgraded the service level to Gold and then to Diamond while the test ran. We monitored host IOPS, SQL Server TPM, and read and write response times during the test.

Test results

As the test results in the following figure show, as application demand changed, the service level was upgraded to Gold which allowed PowerMaxOS to promote very active extents to SCM.As demand increased even further, these workloads attained the highest level of priority when the service level was changed to Diamond. At the Diamond service level featuring the availability of larger SCM capacity, the highest level of performance was achieved with unrivaled response times.

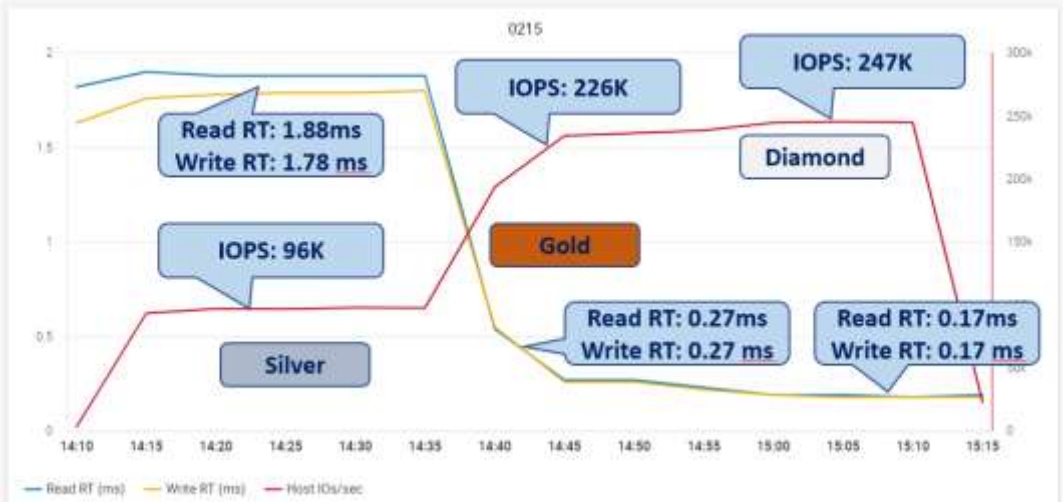

Figure 9. Single SQL Server database performance upgrade test

Table 2 below shows the response times, IOPS, and TPM for each of the three service levels tested.

Table 2. Test case 3: Service levels and response time

Service level	Read response time (ms)	Write response time (ms)	Host IOPS	SQL Server TPM
Silver	1.88	1.78	96K	890K
Gold	0.27	0.27	226K	2.2M
Diamond	0.17	0.17	247K	2.3M

PowerMax automated data placement uses real time ML and AI which effectively prioritized the workloads as the workload demand changed. Therefore, when the service level was changed, PowerMax used the limited SCM drive capacity very effectively for a highly active workload. Because of this ability, the performance benefits were realized very quickly.

Test conclusion

The PowerMax system with automated data placement using AI/ML and service levels effectively manage workload performance of demanding applications.

Test case 4: Onboarding application test

Objective

Test case 4 demonstrates the effective management of workload profiles while onboarding new applications.

Test configuration

We ran two OLTP workloads on SQL Server databases:

- An already-active mission-critical workload running at the Diamond service level
- A low-priority workload for a newly deployed application running at the Silver service level.

As the newly deployed application aged, it was promoted first to the Gold service level and then to the Diamond service level to reflect more users and an increasingly active and important workload profile for that onboarding application. While the test ran, we monitored host IOPS, SQL Server TPM, and read and write response times.

Test results

As the test results in the following figure show, PowerMax prioritized workloads quite well as the service level changed. As the new application became more important and its performance demands increased, PowerMax continued to provide high performance with low latencies while maintaining the Diamond service level performance profile for the existing workload.

Figure 10. Onboarding application test alongside existing mission critical application

Table 3 below shows the response times, IOPS, and TPM for each of the three service levels tested for both databases.

Table 3. Test case 4: Service levels and response time

Period	Database	Service Level	Read response time (ms)	Write response time (ms)	Host IOPS	SQL Server TPM
1	DB1	Diamond	0.25	0.22	223K	2.3M
	DB2	Silver	1.92	1.89	91K	850K
2	DB1	Diamond	0.30	0.30	214K	2.0M
	DB2	Gold	0.89	0.89	204K	1.96M
3	DB1	Diamond	0.4	0.35	207K	1.93M
	DB2	Diamond	0.4	0.35	207K	1.93M

This test shows that the PowerMaxOS is able to identify the increasing activity level of onboarding application as the service levels change. As a result PowerMax appropriately prioritizes the onboarding application over the existing application and tries to balance

performance. When both applications run on Diamond service levels at the similar activity levels PowerMax resources are nicely balanced across both applications.

Test conclusion

The PowerMax system automated data placement recognizes active and demanding applications. It leverages the limited SCM tier effectively by keeping highly active extents from both applications in SCM to optimize the performance of both applications simultaneously.

Test case 5: Noisy neighbor test

Objective

This test demonstrates how PowerMaxOS maintains the performance levels needed for a mission-critical application by managing a “noisy neighbor” – the temporary but demanding application competing with mission-critical application for the system resources.

Test configuration

We ran a single SQL Server database OLTP workload with a SQL Server buffer pool size of 4 GB. The SQL Server database was running at a Diamond service level for the duration of the test. We started a “noisy neighbor” application at a Silver service level and after some time, we terminated that application. While the test ran, we monitored host IOPS, SQL Server TPM, and read and write response times.

Test results

As the test results in the following figure show, PowerMax maintained the performance of the mission-critical application at the Diamond service level while managing a noisy neighbor at a Silver service level for a limited amount of time.

Figure 11. No impact to mission critical application due to noisy neighbor

The SQL Server database at the Diamond Service level achieved read and write response times of 0.17 ms with host IOPS of 231 K and SQL Server TPM of over 2.1 M. As the noisy neighbor application ran, it achieved 1.84 and 1.7 ms read and write response times respectively with SQL Server host IOPS 30 K and TPM of 295 K. Even when it was running there was only marginal impact on the mission-critical application, which continued with 229 K

host IOPS and 2.1 M SQL server TPM at the latencies of 0.2 ms for read and write operations. And when the noisy neighbor was terminated, the mission-critical application returned to its original level of operation.

Test conclusion

The PowerMax system with SCM drives and other performance features can provide unprecedented levels of IOPS at very low latencies for mission-critical applications. It can maintain that performance level even when some low-priority applications are added to the workload.

Test case 6: End-to-end NVMe test for SQL Server on Linux

Objective

Test case 6 demonstrates the end-to-end NVMe support of PowerMax for SQL Server running on Linux servers.

Test configuration

We ran a single SQL Server database OLTP workload with a SQL Server buffer pool size of 4 GB on Linux server. While the test ran, we monitored host IOPS, SQL Server TPM, and read and write response times.

Test results

As the test results in the following figure show, PowerMax provided end-to-end NVMe support when using 32 Gb FC-NVMe HBAs on the Linux server.

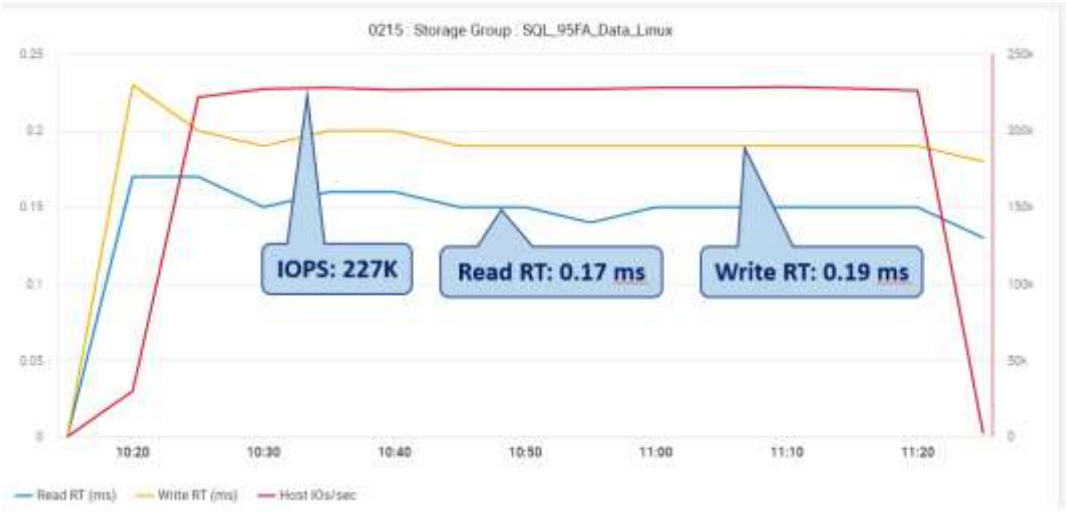

Figure 12. End-to-end NVMe test for SQL Server running on Linux

This testing achieved read and write response times of 0.15 ms and 0.19 ms respectively. This performance level was unparalleled, while all components of the storage system were still quite balanced and able to support additional workloads. PowerMax supported host IOPS of over 227 K with a SQL Server Database OLTP transaction rate of 1.8 M.

Test conclusion

The end-to-end NVMe support on PowerMax accommodates an unrivaled level of performance for SQL Server on Linux.

Test case 7: DSS bandwidth test

Objective

Test case 7 demonstrates the high level of sequential read bandwidth that is available on the PowerMax platform for decision support systems and data warehouse environments.

Test configuration

We ran a single SQL Server database DSS workload at the Diamond service level with a SQL Server buffer pool size of 4 GB. While the test ran, we monitored host MB/S, as well as read and write response times.

Test results

As the test results in the following figure show, dual-ported smart RAID, 32 Gb FC-NVMe, and SCM drives coupled with the PowerMax cache provided a tremendous amount of bandwidth and unprecedented low latencies while supporting a SQL Server database OLTP workload at the Diamond service level.

Figure 13. SQL Server DSS sequential read bandwidth on PowerMax

The test showed an unprecedented read response time of 0.33 ms and a write response time of 0.21 ms while driving the DSS workload at almost 30 GB/s.

Test conclusion

The PowerMax system with SCM drives and all other performance features can provide unparalled levels of bandwidth at very low latencies for decision support systems.

Dell EMC documentation references

The following Dell EMC documentation provides additional and relevant information for PowerMax. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell EMC representative.

- PowerMax and VMAX technical documents and videos
- PowerMax and VMAX All Flash technical document library for application papers

Appendix A: Operating system support and Windows/Linux test case

Introduction

This appendix provides an overview of supported operating systems for SQL Server 2017 and higher, including Linux support, and a test case that compares OLTP workload performance on Linux and Windows deployments.

Supported operating systems

SQL Server 2017 and higher support the following operating systems:

- Linux:
  - Red Hat Enterprise Linux 7.3
  - SUSE Linux Enterprise Server 12 SP2
  - Ubuntu 16.04
- Docker:
  - Docker Engine 1.8+ on any supported Linux distribution or Docker for Mac/Windows. For more information, see the Docker installation and configuration guide.
  - Minimum of 3.5 GB of disk space and 2 GB of RAM for container

Linux system configuration

The Linux system configuration includes:

- 3.25 GB RAM
- XFS or EXT4 file system

Management tools for SQL Server on Linux

After the operating system is installed, connect to the SQL Server instance on your Linux machine. You can connect locally or remotely and with a variety of tools and drivers. One or more of the following tools can be used to manage SQL Server:

You can use native tools that are used on SQL Server on Windows to manage SQL Server instances on Linux and Docker containers. Supported tools include:

- sqlcmd command-line tool
- Visual Studio Code (VS Code)
- SQL Server Management Studio (SSMS)
- SQL Server Data Tools (SSDT)

Conclusion

For cloud-native applications, Linux and containerized SQL Server versions provide good alternatives to Windows versions. Users can use the standard set of tools that are available to them on those platforms to manage SQL Server without significant performance differences.

Appendix B: Useful commands for SQL Server on Linux

Using Windows share

A Windows shared drive can be made available to Linux-based installations as follows:

1. Grant domain ID DOMAIN\USERNAME full control on the share folder.
2. Create a mount point in Red Hat.
3. Change mssql:mssql as owner of the mount point.

```
sudo mount -t cifs //server1/newshare /var/opt/mssql/data -o
vers=3.0,username=user1,dom=DOMAIN1,uid=mssql,gid=mssql
```

Joining AD Domain from RHEL Linux Server

RHEL Server can join Windows AD for centralized authentication:

```
sudo realm join contoso.com -U 'user@CONTOSO.COM' –v
```

Installing SQL Server 2017 for Linux

You can download SQL Server 2017 from the Microsoft SQL Server website and install it easily:

```
sudo curl -k -o /etc/yum.repos.d/mssql-server.repo
https://packages.microsoft.com/config/rhel/7/mssql-server-2017.repo
sudo /opt/mssql/bin/mssql-conf setup
```

Starting, stopping and checking status of SQL Server 2017 services on Linux

Use standard Linux service control commands for starting, stopping, and checking the status of SQL Server:

```
systemctl start mssql-server
systemctl status mssql-server
systemctl stop mssql-server
```

Appendix C: SQL Server on Kubernetes with PowerMax persistent volumes

This appendix describes in detail how IT professionals and application developers can leverage the advantages of Docker, Kubernetes, SQL Server and Dell EMC servers and PowerMax storage and to deploy SQL servers with efficiency.

A SQL Server instance on Kubernetes with persistent storage on PowerMax provides resiliency and storage efficiency. Kubernetes plays the role of the cluster orchestrator. When a SQL Server instance in a container fails, the orchestrator bootstraps another instance of the container that attaches to the same persistent storage from PowerMax. If a node has failed, Kubernetes creates a pod on another healthy node.

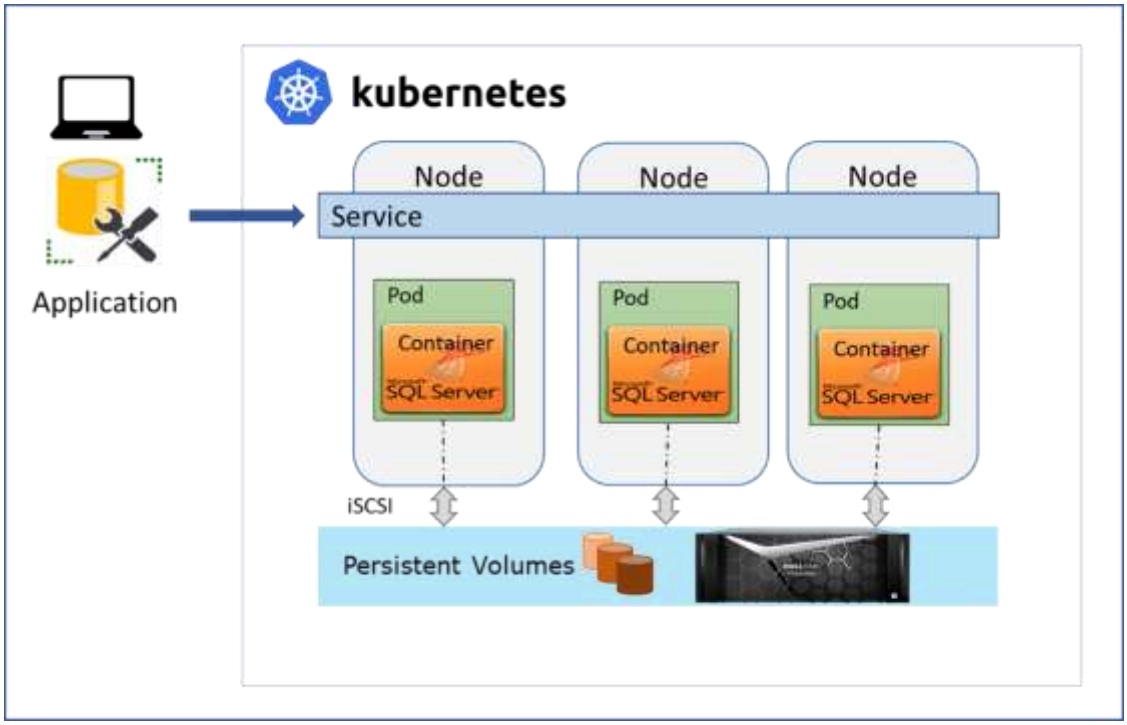

Figure 14. SQL server on Kubernetes architecture

In the preceding figure, mssql-server is a SQL Server instance (container) in a pod. The figure shows three instances of SQL Server running. Applications connect to the service to access the SQL Server. The service maps the external IP address and TCP port to the pod's internal IP address and port. The service could be the Load Balancer type or the Cluster IP type, so that the IP address used to connect to SQL Server remains the same even in the event of a node or pod failure. When a node hosting a SQL Server instance pod fails, Kubernetes bootstraps a new pod with a SQL Server instance on a different, healthy node and attaches it to the same persistent storage.

Note: SQL Server 2019 supports availability groups on containers in a Kubernetes cluster. For availability groups, deploy the SQL Server Kubernetes operator to your Kubernetes cluster. The operator helps package, deploy, and manage SQL Server instances and the availability group in a cluster. SQL Server 2019 deployment with availability groups is not covered in this document.

CSI Plugin

To effectively address the challenges of persistent storage, PowerMax provides a CSI plugin that delivers persistent storage for container-based applications on premises for both development and production scale. The Kubernetes CSI plugin for PowerMax is CSI 1.0 compliant and allows containerized applications in Kubernetes clusters to use block storage from PowerMax over an iSCSI interface.

Prerequisites

Ensure that the CSI Driver for Dell EMC PowerMax has been installed on the Kubernetes cluster and is operational. See the *Powermax CSI Driver Product Guide*. Make sure that your setup meets all the prerequisites described in this document.

Deployment of SQL server infrastructure

Deploying a stateless container into a Kubernetes Cluster is quite simple, but SQL Server is stateful. Therefore, we need a way to persist the created data across pod restarts. Kubernetes provides us with the means to implement persistent storage. This section covers the deployment of SQL Server on Kubernetes under different scenarios including restoring an existing database backup from the PowerMax volume, as well by using a snapshot of the existing data volumes on PowerMax.

Create namespace

Kubernetes namespaces are intended for use in environments with many users spread across multiple teams or projects. Namespaces provide a scope for names and provide a way to divide cluster resources among multiple users (via resource quota). It is recommended to have a separate namespace for SQL. Create a Kubernetes **namespace** with the name “mssql” using this command:

```
# kubectl create namespace mssql
```

Create SA password

Create a password for SQL user SA before you create the SQL server container. The recommended method is to create a Kubernetes secret and use it for the SA password. The following command creates a password for the SA account. Set SA_PASSWORD to that password that you would like to use with this command:

```
# kubectl create secret generic mssql --namespace mssql --from-
literal=SA_PASSWORD="MySecretP@ssw0rd"
```

Note: SA password should be at least 8 characters long and must contain a combination of uppercase letters, lowercase letters, 0 - 9 numerals and non-alphanumeric characters. If minimum password requirements are not met, SQL Server container will fail to come up. Detailed password requirements can be found at *Password Policy*.

Secrets can also be created by using manifest files as described in the Kubernetes documentation about using Secrets. When a secret is being used for SQL SA password, the deployment manifest for SQL Server will have password setting as shown below:

```
env:
        - name: MSSQL_PID
          value: "Developer"
        - name: ACCEPT_EULA
          value: "Y"
```

```
        - name: MSSQL_SA_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mssql
              key: SA_PASSWORD
```

Instead of using a Kubernetes secret, the password can be specified in the manifest file for the deployment. This method is the simplest one, but it is not recommended as it exposes the password clearly in text:

```
env:
        - name: MSSQL_PID
          value: "Developer"
        - name: ACCEPT_EULA
          value: "Y"
        - name: MSSQL_SA_PASSWORD
          value: "MySecretP@ssw0rd"
```

Persistent storage creation

Kubernetes pods are ephemeral by nature, so the data does not survive through the restart/re-scheduling of a pod. The Kubernetes persistent volume (PV) framework allows administrators to provision persistent storage for a cluster. Using persistent volume claims (PVCs), developers can request storage resources defined by a Storage Class (SC) without having specific knowledge of the underlying storage infrastructure.

Once the PowerMax CSI plugin has been installed on a Kubernetes cluster, it creates a default Storage Class using parameters from the *myvalues.yaml* file that was created during plugin installation. You can also create your own storage class by specifying parameters that determine how storage is provisioned on the Dell EMC PowerMax array. Some of the important parameters that you need to define while creating Storage Class are:

- Mandatory parameters:
  - SYMID – Dell EMC PowerMax ID on which volumes are to be created
  - SRP – Storage Resource Pool. If you are not aware of which SRP to use, set it to SRP_1.
- Optional parameters:
  - ServiceLevel – Service Level for the volume. If not specified, the driver takes the Optimized service level as default. See *DellEMC PowerMax: Service Levels for PowerMaxOS* to select a Service Level based on your applications' performance requirements.
  - Application Prefix – Used to group volumes belonging to the same application.

Note: For valid values for parameters SYMID, SRP and ServiceLevel, please consult your PowerMax administrator as these values are specific to your environment. To better understand various Service Levels please see *DellEMC PowerMax: Service Levels for PowerMaxOS*.

Below are two examples of Storage Class. The first example specifies only mandatory parameters.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: pmax-8000-449
provisioner: csi-powermax
parameters:
  SRP: "SRP_1"
  ServiceLevel: "Diamond"
  SYMID: "000197600358"
```

The second example specifies File System type as XFS and reclaim policy as Retain. Setting reclaim policy to Retain will not delete the PowerMax volume when the associated PV ID is deleted. This is useful when you want to preserve the data on the volume for future use.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: pmax-retain
parameters:
  SRP: SRP_1
  SYMID: "000197600358"
  ServiceLevel: Diamond
  FsType: xfs
provisioner: csi-powermax
reclaimPolicy: Retain
volumeBindingMode: Immediate
```

You can create Persistent Volumes (PV) and Persistent Volume Claims (PVC) using these storage classes. These PVC names can be used in the pod manifests where you can specify which containers need these volumes and where they must be mounted.

In general, a container's root filesystem is not suitable for storing persistent data. The containers you run on Kubernetes Engine are typically disposable entities, and the cluster manager may delete, evict, or reschedule any containers that become unavailable due to node failure or other causes. In such an occurrence, all data saved to a container's root filesystem is lost.

To deploy SQL server in Kubernetes cluster with persistent volumes from PowerMax, create at least three PVCs with the appropriate Storage Class using a manifest. One volume is for the SQL Server installation which will be mounted as /var/opt/mssql so that the SQL Server configuration can persist. The other two volumes are for data and log respectively. The number of data and log volumes can be increased as needed. It is recommended to have at least two different volumes: one volume for data and one for log. Having data and log on the same volume is not recommended as described earlier in this paper. A sample manifest for creating PVCs to be used by SQL server is given below:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
  name: mssql-server-1
  namespace: mssql
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: pmax-358
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mssql-data-1
  namespace: mssql
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1024Gi
  storageClassName: pmax-358
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mssql-log-1
  namespace: mssql
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 250Gi
  storageClassName: pmax-358
```

Create PVCs using the manifest. It will create a PVC and an associated PV.

```
# kubectl create -f pvc-449-mssql-server-data-log.yaml
persistentvolumeclaim/mssql-server-1 created
persistentvolumeclaim/mssql-data-1 created
persistentvolumeclaim/mssql-log-1 created
```

Use Kubectl command to list PVCs created by manifest and their associated PVs. The volume capacity may be larger that requested in the manifest. This is not a problem.

```
# kubectl get pvc -n mssql
```

```
NAME             STATUS   VOLUME            CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mssql-data-1     Bound    pmax-5f45ddbfb2   1025Gi     RWO            pmax-358       29s
mssql-log-1      Bound    pmax-5f464d68b2   251Gi      RWO            pmax-358       29s
mssql-server-1   Bound    pmax-5f451fb8b2   11Gi       RWO            pmax-358       29s

# kubectl get pv -n mssql
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS        CLAIM     STORAGECLASS   REASON   AGE
pmax-5f451fb8b2   11Gi       RWO            Delete           Bound         sql/mssql-server-1       pmax-449               8m53s
pmax-5f45ddbfb2   1025Gi     RWO            Delete           Bound         sql/mssql-data-1         pmax-449               8m53s
pmax-5f464d68b2   251Gi      RWO            Delete           Bound         sql/mssql-log-1          pmax-449               9m7s
```

Using existing PowerMax volumes to create PV/PVC

If a containerized SQL Server needs to restore backup from an existing PowerMax volume or needs to use existing PowerMax volumes for data and log, you need to create a PV using the existing PowerMax volume. Create this PV with Storage Class with ReclaimPolicy set to Retain. Use the kubectl command to verify the ReclaimPolicy setting for the Storage Class:

```
# kubectl describe sc pmax-retain-358

Name:                  pmax-retain-358
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           csi-powermax
Parameters:
SRP=SRP_1,SYMID=000197600358,ServiceLevel=Bronze
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Retain
VolumeBindingMode:     Immediate
Events:                <none>
```

Identify the PowerMax volume that needs to be used for PV and find its volume identifier. If there is no volume identifier for the volume, set a unique identifier using Unisphere or SymCLI. The command `symdev -sid <sym> list -identifier device_name` displays the list of volumes with identifier.

```
# symdev -sid 358 list -identifier device_name

Symmetrix ID: 000197600358
                                      Device
-------------------------------------------------------------
Sym   Config          Attr Device Name
----- --------------- ---- ------------------------------
000F7 TDEV                 SQLbackup1TB
000FC TDEV                 csi-K8S-pmax-2a908fccb3
```

Volume

```
000FF TDEV                csi-K8S-pmax-88f40510b4
00101 TDEV                csi-K8S-pmax-88f4c8f6b4
00108 TDEV                csi-K8S-pmax-706fba4db9
```

Unisphere shows the volume identifier in the volume details next to the device ID:

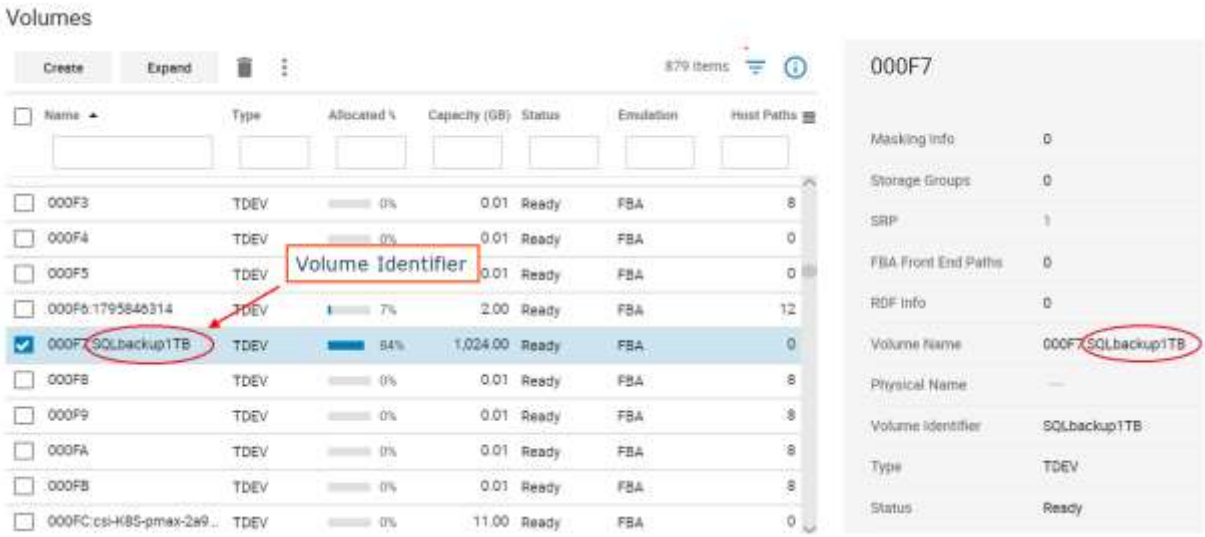

Figure 15. Locating volume identifier for a PowerMax volume using Unisphere

If the volume identifier is not set or needs to be changed, use SymCLI or Unisphere. Do not change the volume identifier if the volume is in use by the CSI driver, because the volume identifier is validated for CSI operations. A volume is in use by CSI driver if a Persistent Volume (PV) is already mapped to the volume.

```
# symdev -sid 358 set 000F7 -device_name sqlbackup2
```

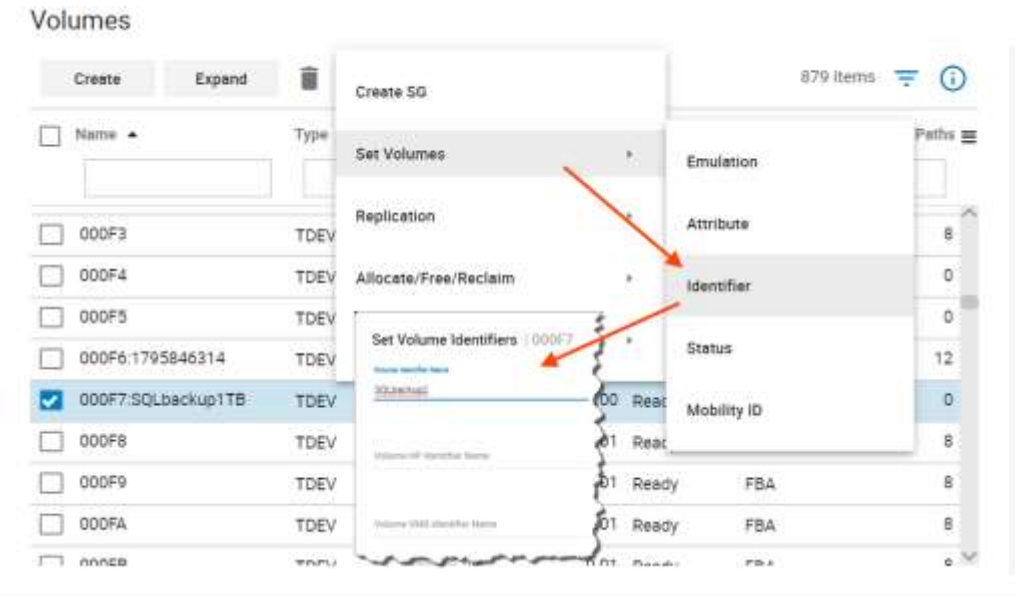

Figure 16. Setting volume identifier using Unisphere

A volume identifier is used to specify the volumeHandle for PV. The volume handle needs to be in the format <volumeIdentifier>-<SymmID-<device ID>. For example, if volume is 000F7 on PowerMax 000197600358 and identifier is set to “SQLbackup1TB”, volumeHandle would be SQLbackup1TB-000197600358-000F7 as shown in the manifest for PV and PVC:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mssql-backup
  namespace: mssql
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 1024Gi
  csi:
    driver: csi-powermax.dellemc.com
    volumeHandle: SQLbackup1TB-000197600358-000F7
  persistentVolumeReclaimPolicy: Retain
  storageClassName: pmax-retain
  volumeMode: Filesystem
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mssql-backup
  namespace: mssql
```

```
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1024Gi
  storageClassName: pmax-retain-358
  volumeMode: Filesystem
  volumeName: mssql-backup
apiVersion: apps/v1beta1
```

Deployment of SQL server

Once the required PVCs are in place, you are ready to deploy SQL Server. All created PVCs will be mounted to their respective directories in the SQL container. In this document example, the container hosting the SQL Server instance is described as a Kubernetes deployment object. The deployment creates a replica set. The replica set creates the pod.

Create a manifest to describe the container based on the SQL Server mssql-server-linux Docker image. The manifest references the mssql-server persistent volume claim, and the mssql secret that is already applied to the Kubernetes cluster. The manifest also describes a service, which is a load balancer. The load balancer guarantees that the IP address persists after the SQL Server instance is recovered.

```
kind: Deployment
metadata:
  name: mssql-deployment-1
  namespace: mssql
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: mssql
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: mssql
        image: mcr.microsoft.com/mssql/server:2017-latest
        ports:
        - containerPort: 1433
        env:
        - name: MSSQL_PID
          value: "Developer"
        - name: ACCEPT_EULA
          value: "Y"
        - name: MSSQL_SA_PASSWORD
           valueFrom:
             secretKeyRef:
               name: mssql
```

```
              key: SA_PASSWORD
        volumeMounts:
        - name: mssqlsvr
          mountPath: /var/opt/mssql
        - name: mssqldata
          mountPath: /mssql/data
        - name: mssqllog
          mountPath: /mssql/log
        - name: mssqlbackup
          mountPath: /mssql/backup
      volumes:
      - name: mssqlsvr
        persistentVolumeClaim:
          claimName: mssql-server-1
      - name: mssqldata
        persistentVolumeClaim:
          claimName: mssql-data-1
      - name: mssqllog
        persistentVolumeClaim:
          claimName: mssql-log-1
      - name: mssqlbackup
        persistentVolumeClaim:
          claimName: mssql-backup
---
apiVersion: v1

kind: Service
metadata:
  name: mssql-deployment-1
  namespace: mssql
spec:
  selector:
    app: mssql
  ports:
    - protocol: TCP
      port: 1433
      targetPort: 1433
  externalIPs:
    - 10.228.247.248
```

Some of the fields and their values in the manifest are explained here:

- **MSSQL_PID value**: "Developer": Sets the container to run SQL Server Developer edition. Developer edition is not licensed for production data. If the deployment is for production use, set the appropriate edition (Enterprise, Standard, or Express).
- **persistentVolumeClaim**: This value requires an entry for claimName: that maps to the name used for the persistent volume claim. In this example four persistent volume claims are being used.

- **MSSQL_SA_PASSWORD**: Configures the container image to set the SA password. A Kubernetes secret is being used for password that was created in the earlier steps.

Create deployment using manifest file using the `kubectl create -f <manifest file name>` command:

```
# kubectl create -f deploy-mssql-server-1.yaml
deployment.apps/mssql-deployment-1 created
```

The deployment and service are created. The SQL Server instance is in a container, connected to persistent storage. To view the status of the pod, enter the `kubectl get pod` command.

```
# kubectl get pod -n mssql
NAME                                READY   STATUS    RESTARTS
AGE
mssql-deployment-1-7f754f6d7b-jss58   1/1     Running   0
75s
```

If the pod has a status of Running, it indicates that the container is ready. It may take several minutes for the pod to get to the Running state after being deployed.

```
# kubectl describe pod mssql-deployment-1-7f754f6d7b-jss58
Name:               mssql-deployment-1-7f754f6d7b-jss58
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               dsib3244/10.228.247.244
Start Time:         Thu, 01 Aug 2019 19:31:41 -0400
Labels:             app=mssql
                    pod-template-hash=7f754f6d7b
Annotations:        <none>
Status:             Running
IP:                 10.233.65.72
Controlled By:      ReplicaSet/mssql-deployment-1-7f754f6d7b
Containers:
  mssql:
    Container ID:
docker://439a0bb16317ccd42af75e38b10e0b479b3968025a77f1c8325c38e7f
ae192e5
    Image:          mcr.microsoft.com/mssql/server:2017-latest
    Image ID:       docker-
pullable://mcr.microsoft.com/mssql/server@sha256:29fb9c64b0efb5694
8864b4df9e8b1dc26ef3ecc552e64902c24d81519f6a15e
    Port:           1433/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 01 Aug 2019 19:32:19 -0400
    Ready:          True
```

The `kubectl get services` command returns services that are running, as well as the internal and external IP addresses for the services. Make note of the external IP address for the mssql-deployment service. Use this IP address to connect to SQL Server.

```
# kubectl get services -n mssql
NAME                  TYPE        CLUSTER-IP     EXTERNAL-IP
PORT(S)     AGE
mssql-deployment-8    ClusterIP   10.233.45.10   10.228.247.248
1433/TCP    2d
```

A static route may be required to be configured on the host accessing SQL Server. If required, set a static route for EXTERNAL-IP of the service with gateway as one of the nodes of the cluster.

Connecting to the SQL Server instance

Use the `sa` account and the external IP address (and port number if not default) for the service to connect to the SQL Server container. Use the password that you configured as the Kubernetes secret or password configured in the deployment manifest.

You can use the following applications to connect to the SQL Server instance:

sqlcmd

To connect with sqlcmd, run the following command from the node or from any other machine which has sqlcmd installed and has network access to the pod.

```
sqlcmd -S <External IP Address> -U sa -P <SA Password>
```

For example:

```
sqlcmd -S 10.233.26.89 -U sa
```

Password:

```
1>
```

SQL Server Management Studio

SQL Server Management Studio (SSMS) is part of a suite of SQL tools that Microsoft offers free of charge for your development and management needs. SSMS is an integrated environment to access, configure, manage, administer, and develop all components of SQL Server. It can connect to SQL Server running on any platform on-premises, in Docker containers, and in the cloud.

To connect to SQL server running in a container, enter the following information:

Setting	Description
Server type	The default is database engine; do not change this value
Server name	External IP address assigned to service and the TCP port number
Authentication	For SQL Server on Linux, use SQL Server Authentication
Login	Name of a user with access to a database on the server, default is SA as configured in manifest

Setting	Description
Password	Password or Secret configured in Pod manifest

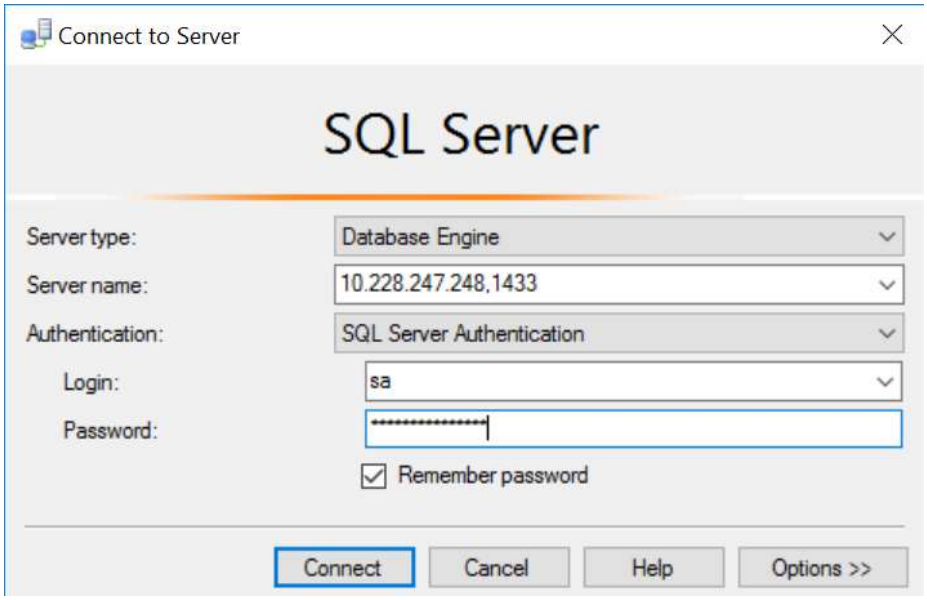

Figure 17. SQL Server Management Studio (SSMS)

Restore database

If the deployment was created with a backup volume mounted, the database can be restored from backup using sqlcmd or SSMS.

An example of a script used to restore a database is shown below.

```
USE [master]
RESTORE DATABASE [tpcc215_Mod_8Dev1TB] FROM  DISK = N'/sql/backup/tpccMod_MyFile.bak'
WITH REPLACE,FILE = 1
, MOVE N'tpcc215_org_8Dev1TB_fg1' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg1.mdf'
, MOVE N'tpcc215_org_8Dev1TB_fg2' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg2.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg3' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg3.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg4' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg4.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg5' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg5.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg6' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg6.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg7' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg7.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg8' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg8.ndf'
, MOVE N'tpcc215_org_8Dev1TB_log' TO N'/mssql/log/tpcc215_Mod_8Dev1TB_log.ldf'
, NOUNLOAD,  STATS = 5

GO
```

Figure 18. Sample SQL script to restore database

You can run the script SSMS or using by using sqlcmd:

```
sqlcmd -S <Server IP Address> -U sa -P <SA Password> -i <script
file> -o <result file>
```

For example:

```
# sqlcmd -S 10.228.247.248 -U sa -P MySecretP@ssw0rd -i
restoredb.sql -o restoreout.txt
```

Migrating existing SQL database to Kubernetes

Migrating an existing SQL database which exists on a PowerMax device is fast and efficient. There is no need to go through the time-consuming process of database backup and restore. The most efficient method is to create a snapshot of the existing data and log volumes and then use those snapshots to create Kubernetes PVs to be used by a SQL Server container.

PowerMax array's local replication data service Timefinder SnapVX creates very low-impact snapshots. SnapVX provides the ability to manage consistent point-in-time copies for storage groups with a single operation. This allows multiple persistent volumes used by a pod to be managed together.

A point-in-time snapshot can be accessed from Kubernetes nodes by linking it to a host-accessible volume referred to as a target. Target volumes are standard PowerMax devices. Up to 1,024 target volumes can be linked to the snapshot(s) of a single source volume. By default, targets are linked in a no-copy mode. This no-copy mode eliminates the need to perform a full volume copy of the source volume during the unlink operation in order to continue to use the target volume for host I/O.

Make a copy of existing data and log

To make a copy of the existing data and log, follow these steps:

1. Identify the PowerMax volumes that contain the existing SQL server and put them in a Storage Group. If the volumes are already in a storage group, this step is not required.

```
symsg -sid 358 symsg  -sid 358 create  mssql-deployment-1
symsg  -sid 358 create  mssql-deployment-1
symsg -sid 358 -sg  mssql-deployment-1 add dev 000FF
symsg -sid 358 -sg  mssql-deployment-1 add dev 00101
```

2. Create a snapshot of the storage group which will be then used to create Kubernetes persistent volumes.

```
symsnapvx -sid 358 -sg  mssql-deployment-1 -name sql-snap-
1 establish
```

3. Create a new Storage Group with volumes that match in number and size with the original storage group, and link the snapshot to this new storage group. Make sure that device names do not contain "-" in the name.

```
symsg -sid 358 create sql_LNK_SG_001

symdev -sid 358 create -tdev -emulation fba -v -nop -cap  1048577 -captype mb -
device_name sqldatasnap001 -sg sql_LNK_SG_001

symdev -sid 358 create -tdev -emulation fba -v -nop -cap 256001 -captype mb -
device_name sqllogsnap001 -sg sql_LNK_SG_001

symsnapvx -sid 358 -sg mssql-deployment-1 -lnsg sql_LNK_SG_001 -snapshot_name
sqlsnap1  link -copy
```

Create PV and PVC using existing volumes

To create a PV using a specific existing PowerMax volume, use a volume identifier to specify the volumeHandle for PV. The volume handle must be in the format `<volumeIdentifier>-<SymmID-<device ID>.`

For example, if volume is 00104 on PowerMax with SYMID 000197600358 and identifier is set to “sqldatasnap001”, volumeHandle would be “sqldatasnap001-000197600358-00104” as shown in the manifest for PV and PVC below. Create similar manifests for all the volumes required for the SQL server (Data and Log volumes). Field *labels* can be used to uniquely define a PV so that while creating a PVC, the *selector* parameter can be used to pick a specific PV. In this example, label matching is being used to pick a specific PV while creating PVC. Create PV for data and log volumes.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: sql-data-snap-001
  labels:
    device-name: sqldatasnap001
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 1024Gi
  csi:
    driver: csi-powermax.dellemc.com
    volumeHandle: sqldatasnap001-000197600358-00104
  persistentVolumeReclaimPolicy: Retain
  storageClassName: powermax-358
  volumeMode: Filesystem
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mssql-data-snap-1
spec:
  accessModes:
```

```
  - ReadWriteOnce
  resources:
    requests:
      storage: 1024Gi
  selector:
    matchLabels:
      device-name: sqldatasnap001
  storageClassName: powermax-358
```

For the SQL server volume, create a dynamic PVC of required size.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mssql-server-1
  labels:
    device-name: sql-server-1
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: powermax-358
```

Deploy SQL server and attach database

Deploy SQL Server using the PVC created in the previous step. The manifest will have one newly created volume for SQL Server, plus data and log volumes from existing PowerMax devices. Once the pod is operational, use SQL Server Management Studio to attach the database to SQL Server using the .mdf file for the database. In SSMS, after connecting to the SQL Server Database Engine through Object Explorer, expand the server, right-click on Databases, and select Attach.

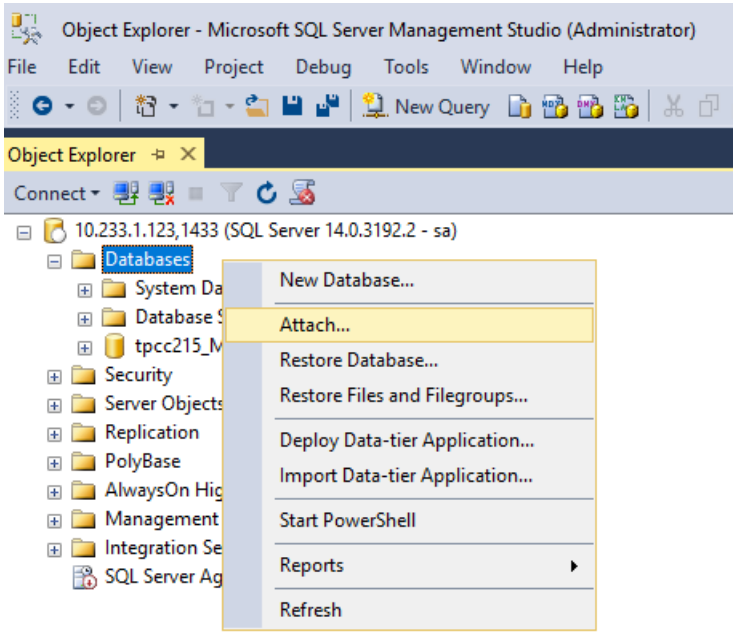

Figure 19. Attaching SQL Server database using SSMS

Create a script file with the restore commands as shown in the script example below. This script example has eight data files and one log file to be restored. Use SSMS or sqlcmd to execute the script. Run sqlcmd with all the required options. Option -I specifies the input file which was created in the previous step and -o option specifies the output file where output of the command will be stored. If -o option is not specified, output is written on a standard output device which is normally the terminal screen.

```
# sqlcmd -S 10.228.247.248 -U sa -P MySecretP@ssw0rd -i
attachdb.sql -o attachout.txt
```

```
USE [master]
RESTORE DATABASE [tpcc215_Mod_8Dev1TB] FROM  DISK = N'/sql/backup/tpccMod_MyFile.bak' WITH
REPLACE,FILE = 1
, MOVE N'tpcc215_org_8Dev1TB_fg1' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg1.mdf'
, MOVE N'tpcc215_org_8Dev1TB_fg2' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg2.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg3' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg3.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg4' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg4.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg5' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg5.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg6' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg6.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg7' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg7.ndf'
, MOVE N'tpcc215_org_8Dev1TB_fg8' TO N'/mssql/data/tpcc215_Mod_8Dev1TB_fg8.ndf'
, MOVE N'tpcc215_org_8Dev1TB_log' TO N'/mssql/log/tpcc215_Mod_8Dev1TB_log.ldf'
, NOUNLOAD,  STATS = 5

GO
```

Planned failover

If the node running SQL Server needs to be taken out of service, there is no need to shut down the server. The SQL Server will automatically be migrated to another node. The server will be unavailable while the persistent storage volumes are mapped to the new node. The failover time depends on the number of volumes mapped to the container and the volume un-map/map process is sequential in nature. Table 1 shows SQL server failover time and the number of persistent volumes.

Table 4. SQL server downtime and number of PVCs

SQL server migration time using Node drain	
Number of PVCs	**SQL server down time (Seconds)**
1	64
4	84
8	143
12	148

If the container dies and is restarted on the same node, the downtime is only 10 to 12 seconds because no volume un-map/map or volume discovery is required in this scenario.