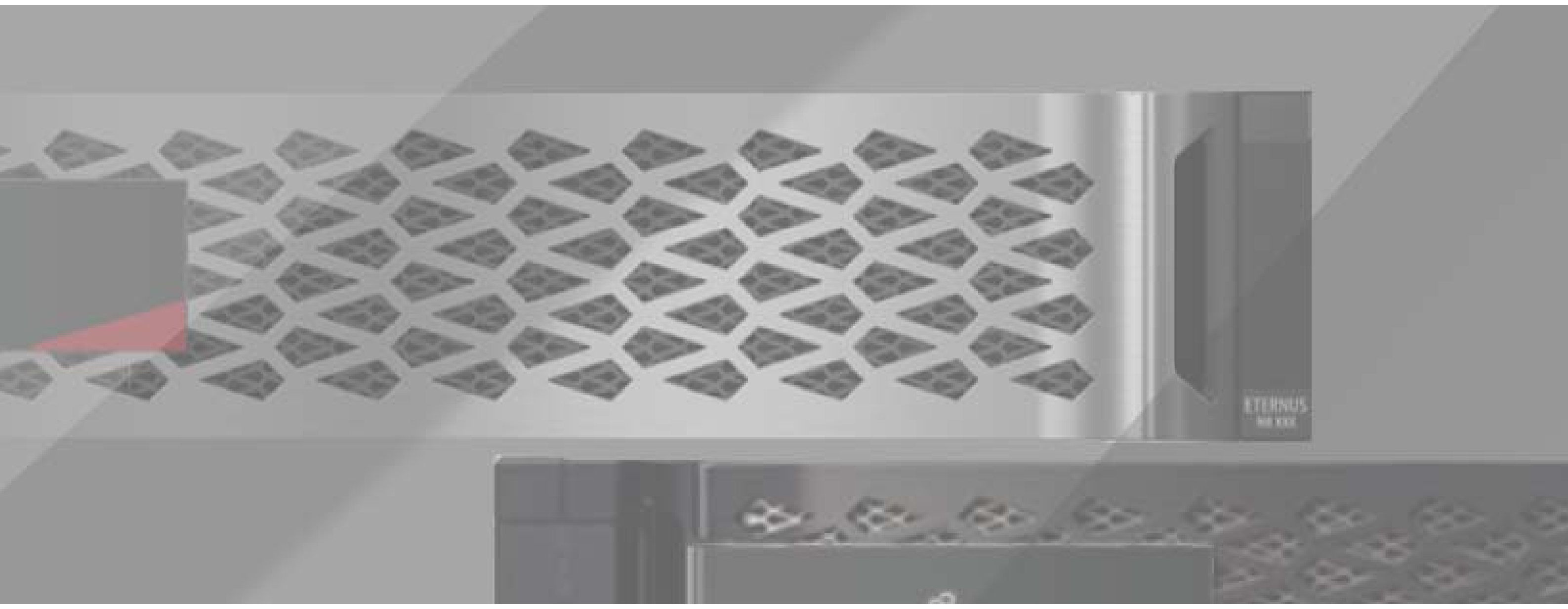
P3AG-6612-01ENZ0

FUJITSU Storage
ETERNUS AB series All-Flash Arrays

Best Practices Guide for Microsoft SQL Server with ETERNUS AB series



FUJITSU

Table of Contents

List of Figures

List of Tables

Preface

This document is intended to help storage administrators and database administrators successfully deploy Microsoft SQL Server on ETERNUS AB series storage.

First Edition
June 2022

Trademarks

About This Manual

Intended Audience

This document is intended for Fujitsu customers, partners, employees, and field personnel who are responsible for deploying a SQL Server database solution in a customer environment. It is assumed that the reader is familiar with the various components of the solution.

Related Information and Documents

The latest information for the ETERNUS AB is available at:
https://www.fujitsu.com/global/support/products/computing/storage/manuals-list.html

Document Conventions

■ Notice Symbols

The following notice symbols are used in this manual:

Caution	Indicates information that you need to observe when using the ETERNUS AB. Make sure to read the information.
Note	Indicates information and suggestions that supplement the descriptions included in this manual.

1. Executive Summary

In most OLTP systems, the processor, memory, and I/O subsystems in a server are well balanced and are not considered performance bottlenecks. The major source of performance issues in OLTP environments is typically related to storage I/O activity. The speed of existing HDD-based storage systems does not match the processing capabilities of the servers.

As a result, a powerful processor often sits idle, waiting for storage I/O requests to complete. This situation negatively affects user and business productivity. The loss of productivity delays return on investment (ROI) and increases overall TCO. Therefore, storage IOPS performance and latency become strategic considerations for business. It is critical to make sure that the response time goals are met and performance optimization is realized for other system resources (processor and memory).

The ETERNUS AB series flash array provides a robust system for delivering exceptional performance with submillisecond response times to mission-critical applications. The ETERNUS AB series flash array uses the latest in solid-state drive (SSD) technology along with a strong history of handling diverse workloads to provide superior business value by accelerating latency-sensitive and high-I/O environments.

The ETERNUS AB series flash array is exceptional not only for OLTP databases but also for data warehouse environments. The entry-level ETERNUS AB2100 is certified under Microsoft's Fast Track Data Warehouse program.

The ETERNUS AB series flash array leads the market in delivering high performance and low latency. AB6100 is the $/SPC-2 MBps leader, followed by AB5100.

NVMe has become the industry standard interface for Peripheral Component Interconnect Express (PCIe) SSDs. With a streamlined protocol command set and fewer clock cycles per I/O, NVMe supports up to 64K queues and up to 64K commands per queue. These attributes make it more efficient than SCSI-based protocols like SAS and SATA.

The introduction of NVMe over Fabrics (NVMe-oF) makes NVMe more scalable without affecting the low latency and small overhead that are characteristic of the interface. ETERNUS AB3100, AB5100, and AB6100 systems support NVMe over RoCE (NVMe/RoCE), NVMe over InfiniBand (NVMe/IB), and NVMe over Fibre Channel (NVMe/FC).

Both the AB3100 and AB6100 support iSCSI and the SCSI-based Fibre Channel Protocol (FCP), as well as NVMe/FC. Because little change is required in the standards to implement NVMe/FC, the introduction of NVMe/FC along with existing storage is easy, seamless, and noninvasive. Because NVMe/FC can use the same infrastructure components concurrently with other FC traffic, it is easy to migrate workloads at a pace that works for your organization. NVMe/FC also allows the efficient transfer of NVMe commands and structures end to end with no translations.

The leading-edge AB3100 and AB6100, which are built on end-to-end NVMe technology, not only provide NVMe-oF to reduce latency between the server and SAN but they also incorporate NVMe technology to accelerate access to your data.

The ETERNUS AB series array, available with up to 1.8PB of raw SSD capacity in the AB5100 and up to 367TB in the AB3100 and AB6100, provides the capacity and bullet-proof reliability to meet the requirements of the most demanding organizations. This technical report provides an overview of best practices for Microsoft SQL Server with ETERNUS AB series flash arrays.

Caveats

This document assumes that the database is either being relocated to an ETERNUS AB series storage system or being created on an ETERNUS AB series storage system to achieve high performance.

This document also assumes that you want to improve the performance of an OLTP application.

2. Introduction to ETERNUS AB series Storage Systems

ETERNUS AB series Hardware Overview

With high IOPS and sub-millisecond response times, ETERNUS AB series arrays enable business-critical applications to deliver faster results and improve customer experience.

This combination of high IOPS and ultra-low latency makes an ETERNUS AB series flash array a great choice for database-driven applications that require a dedicated, extreme-performance solution.

Table 1 provides an overview of ETERNUS AB series flash array performance.

Table 1 ETERNUS AB series performance for RAID 6

Performance Comparison	AB2100	AB3100	AB5100	AB6100
SSD count	24	24	48	24
Max read IOPS	300K<210µs	670K<250µs	1M<250µs	2M<250µs
Max write IOPS	45K<160µs	100K<190µs	185K<250µs	340K<190µs
Read bandwidth	10GB/sec	20GB/sec	21GB/sec	44GB/sec
Write bandwidth (cache mirroring enabled)	3.7GB/sec	7GB/sec	9GB/sec	12.5GB/sec
Low latency (reads)	55K<140µs	150K<200µs	100K<120µs	140K<110µs
Low latency (writes)	45K<160µs	50K<100µs	150K<100µs	200K<80µs

Table 2 provides a comparison of the ETERNUS AB series product line.

Table 2 ETERNUS AB series product comparison

Product Comparison	AB2100	AB3100	AB5100	AB6100
2U/24 Expansion shelves	3	none	4	none
Total drives	96	24	120	24
Raw capacity	1.45PB	367TB	1.8PB	367TB
Controller cache options	8GB or 32GB	16GB	16GB or 64GB	32GB or 128GB

Along with performance, the key to maximizing value is to maximize efficiency. Historically, companies have sacrificed efficiency to achieve extreme performance levels by overprovisioning their storage. But that is changing. The ETERNUS AB series system helps customers balance performance and efficiency by eliminating overprovisioning, resulting in dramatically reduced costs.

With the performance of more than 1,000 traditional drives, a single ETERNUS AB series flash array can meet extreme requirements with 95% less rack space, power, and cooling. This ability is a significant benefit to customers who are used to deploying partially filled disks to improve application performance.

In addition to cost efficiency, the ETERNUS AB series flash array provides application efficiency. By completing a higher volume of application operations, customers can dramatically increase efficiency and obtain better results.

The ETERNUS AB series flash array has a fully redundant I/O path that provides automated failover. Surprisingly, automated failover is not provided in many of the flash products available today, but it is an absolute requirement for enterprises that want to implement this type of technology.

All management tasks are performed while the ETERNUS AB series array remains online with complete read/write data access. This approach allows storage administrators to make configuration changes and conduct maintenance without disrupting application I/O.

The ETERNUS AB series flash array also offers advanced data protection common to enterprise storage to protect against data loss and downtime events. This protection is available locally with Snapshot technology and remotely with synchronous and asynchronous replication (except AB3100 and AB6100).

SANtricity OS

ETERNUS AB series flash array systems are managed by the SANtricity System Manager browser-based application. The SANtricity System Manager is embedded on the controller.

To create volume groups on the array, the first step is to assign a protection level during SANtricity configuration. This assignment is then applied to the disks selected to form the volume group. The ETERNUS AB series flash array storage systems support Dynamic Disk Pools (DDP) and RAID levels 0, 1, 5, 6, and 10. DDP was used for all configurations described in this document.

To simplify storage provisioning, SANtricity provides an automatic configuration feature. The configuration wizard analyzes the available disk capacity on the array. It then selects disks that maximize array performance and fault tolerance while meeting capacity requirements, hot spares, and any other criteria specified in the wizard.

For further information about SANtricity Unified Manager and SANtricity System Manager, see the Fujitsu manual site.

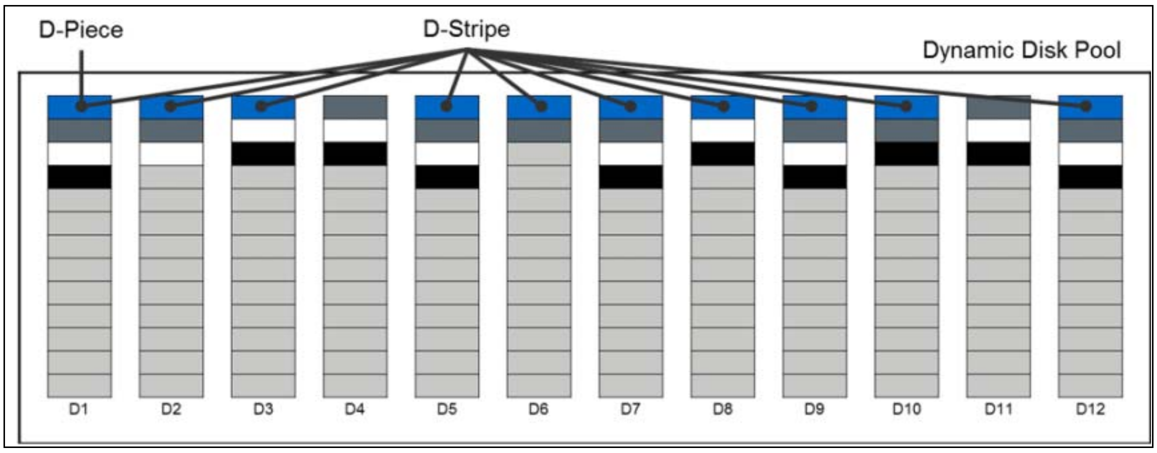
Dynamic Disk Pools

With seven patents pending, the DDP feature dynamically distributes data, spare capacity, and protection information across a pool of drives. These pools can range from a minimum of 11 drives to all the supported drives in a system. In addition to creating a single pool, storage administrators can mix traditional volume groups and DDP or even multiple pools, offering an unprecedented level of flexibility.

A pool is composed of several lower-level elements. The first of these elements is a D-piece. A D-piece consists of a contiguous 512MB section from a physical disk that contains 4,096 128KB segments. Within a pool, the system chooses 10 D-pieces by using an intelligent optimization algorithm from selected drives in the pool. Together, the 10 associated D-pieces are considered a D-stripe, which is 4GB of usable capacity. Within the D-stripe, the contents are similar to a RAID 6 8+2 scenario. Eight of the underlying segments potentially contain user data, one segment contains parity (P) information calculated from the user data segments, and one segment contains the Q value as defined by RAID 6.

Volumes are then created from an aggregation of multiple 4GB D-stripes as required to satisfy the defined volume size up to the maximum allowable volume size in a pool. Figure 1 shows the relationship between these data structures.

Figure 1 DDP components

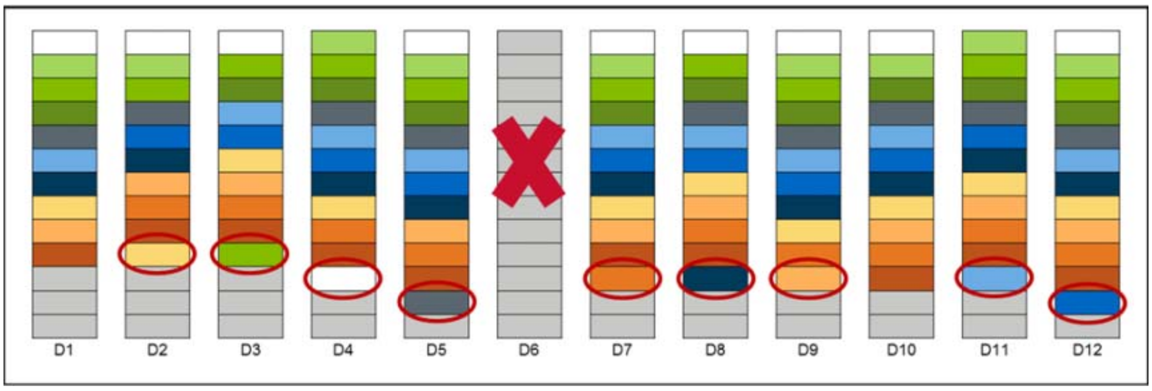


Another major benefit of a pool is that, rather than using dedicated stranded hot spares, the pool contains integrated preservation capacity to provide rebuild locations for potential drive failures. This approach simplifies management because individual hot spares no longer need to be planned or managed. The approach also greatly improves the time for rebuilds, if necessary, and enhances volume performance during a rebuild, as opposed to traditional hot spares.

When a drive in a pool fails, the D-pieces from the failed drive are reconstructed to potentially all other drives in the pool by using the same mechanism normally used by RAID 6. During this process, an algorithm internal to the controller framework verifies that no single drive contains two D-pieces from the same D-stripe. The individual D-pieces are reconstructed at the lowest available Logical Block Access (LBA) range on the selected drive.

In Figure 2, drive 6 (D6) is shown to have failed. Next, the D-pieces that previously resided on that disk are recreated simultaneously across several other drives in the pool. Because there are multiple disks participating in the effort, the overall performance effect of this situation is lessened, and the length of time needed to complete the operation is dramatically reduced.

Figure 2 DDP drive failure



When multiple disk failures occur in a pool, priority for reconstruction is given to any D-stripes missing two D-pieces to minimize data availability risk. After those critically affected D-stripes are reconstructed, the remainder of the necessary data is reconstructed.

From a controller resource allocation perspective, there are two user-modifiable reconstruction priorities in DDP:

- Degraded reconstruction priority is assigned to instances in which only a single D-piece must be rebuilt for the affected D-stripes; the default for this value is high.
- Critical reconstruction priority is assigned to instances in which a D-stripe has two missing D-pieces that need to be rebuilt; the default for this value is highest.

For large disk pools with two simultaneous disk failures, only a relatively small number of D-stripes are likely to encounter the critical situation in which two D-pieces must be reconstructed. As discussed previously, these critical D-pieces are identified and reconstructed initially at the highest priority. Doing so returns the pool to a degraded state quickly so that further drive failures can be tolerated.

In addition to improving rebuild times and providing superior data protection, DDP can also greatly improve the performance of the base volume under a failure condition compared with the performance of traditional volume groups.

For more information about DDP, see "ETERNUS AB/HB series SANtricity OS Dynamic Disk Pools Feature Description and Best Practices" in the Fujitsu manual site.

Provisioning the ETERNUS AB series Flash Array

SANtricity DDP technology allows storage administrators to simplify RAID management, improve data protection, and maintain predictable performance under all conditions. DDP evenly distributes data, protection information, and spare capacity across the entire ETERNUS AB series pool of drives, simplifying setup and maximizing use. Its next-generation technology minimizes the performance effect of a drive failure and can return the system to optimal condition up to eight times more quickly than traditional RAID. With shorter rebuild times and patented technology to prioritize reconstruction, DDP significantly reduces exposure to multiple disk failures, offering a level of data protection that simply cannot be achieved with traditional RAID.

With SANtricity software, all management tasks can be performed while the storage remains online with complete read/write data access. Storage administrators can make configuration changes, conduct maintenance, or expand the storage capacity without disrupting I/O to attached hosts. SANtricity software's online capabilities include the following:

- Dynamic volume expansion allows administrators to expand the capacity of an existing volume.
- Dynamic segment size migration enables administrators to change the segment size of a given volume.
- Dynamic RAID-level migration changes the RAID level of a RAID group on the existing drives without requiring the relocation of data. Supported RAID levels are 0, 1, 5, 6, and 10.
- Nondisruptive controller firmware upgrades are supported with no interruption to data access.

For the detailed information on provisioning ETERNUS AB series flash array, see the Fujitsu manual site. If there is repeated volume creation and deployment, REST API commands can be used to automate this task.

■ Best Practice

Fujitsu recommends using Dynamic Disk Pools for all storage configuration.

3. SQL Server Overview

The SQL Server database platform can support a wide variety of applications. Before deploying SQL Server, you must understand the database workload requirements of the applications that your SQL Server instances support. Each application has different requirements for capacity, performance, and availability, so each database should be designed to optimally support those requirements. Many organizations classify databases into multiple management tiers, using application requirements to define SLAs. SQL Server database workloads can be described as follows:

- OLTP databases are often the most critical databases in an organization. These databases back customer-facing applications and are considered essential to the company's core operations. Mission-critical OLTP databases and the applications they support often have SLAs that require high levels of performance; these databases are sensitive to performance degradation and availability. They might also be candidates for clustering with Windows failover clusters or Always On availability groups. The I/O mix of these types of databases is characterized with 75% to 90% random read and 25% to 10% write.
- Decision support system (DSS) databases can be also referred to as data warehouses. These data warehouses are mission critical in many organizations that rely on analytics for their business. These databases are sensitive to CPU utilization and read operations from disk when queries are being run. In many organizations, DSS databases are the most critical at the end of a month, quarter, or year. This workload typically has a 100% read I/O mix.

SQL Server I/O Overview

SQL Server is sensitive to I/O latency issues because of the concurrent transactional nature of the SQL Server engine. SQL Server is built on a complicated system of row, page, extent, and table locks that provides transactional consistency throughout the SQL Server system. A poor I/O structure (for example, I/O takes too long to respond) causes resources to be held longer than necessary, resulting in blocking within the system. When blocking occurs, it is typically not obvious that the I/O subsystem is the root cause.

- **SQL Server reads**
  When reading data from SQL Server, the client first goes to the buffer cache. If the data is not in the buffer cache, SQL Server goes to the I/O subsystem to retrieve the data. The statement does not complete until 100% of the data is read; the user connection or process remains in an I/O wait state until completion.
- **SQL Server writes**
  The user writes to the transaction log and the buffer cache. If the data to be modified is not already in the buffer cache, then it must be read into the buffer cache from the I/O subsystem. The buffer manager enables the transaction log to be written to first, before changes are written to the database. This technique is known as write-ahead logging (WAL). When the user makes the change and the commit is executed, a log write is displayed about the change that took place, allowing the commit to complete. After the commit is complete, the user process can continue to the next stage or command without waiting for the changes to be written to the disk. Rollback transaction follows the same process as the commit, but in reverse. The buffer manager moves the data from the cache to the disk. It tracks log sequence numbers (LSNs) for each log record.
- **Transaction log**
  The SQL Server transaction log is a write-intensive operation that is sequential. The transaction log is used for data recovery if the database or instance fails.

The OLTP database system in the SQL Server environment is most dependent on getting the greatest number of transactions through the system in the least amount of time. Examples of different types of OLTP systems include web order systems and manufacturing tracking systems. OLTP systems can have large volumes of transactions per second (TPS), and, for the OLTP system, it is all about throughput. For these transactions to take place, SQL Server relies on an efficient I/O subsystem. According to this Microsoft SQL Server best practices article, an OLTP transaction profile has the following attributes:

- OLTP processing is random for both reads and writes issued against data files.
- I/O activity is approximately 80% read and 20% write.

- Usually, read activity is consistent and uses point queries; it does not consist of large time-consuming queries.
- Write activity to the data files occurs during checkpoint operations (frequency is determined by recovery interval settings).
- Log writes are sequential with a varying size that depends on the nature of the workload (sector aligned up to 60KB).
- Log reads are sequential (sector aligned up to 120KB).

Windows Volume Mount Points

Fujitsu storage solutions and Microsoft SQL Server support mount points. Mount points are directories on a volume that can be used to mount a different volume. Mounted volumes can be accessed by referencing the path of the mount point. Mount points eliminate the Windows 26-drive-letter limit and offer greater application transparency when you move data between LUNs, move LUNs between hosts, and unmount and mount LUNs on the same host. This is because you can move the underlying volumes around without changing the mount point path name.

Best Practice

Fujitsu recommends using NTFS mount points instead of drive letters to exceed the 26-drive-letter limitation in Windows. When you use volume mount points, the name given to the volume label and the name given to the mount point must be the same.

Database Files and File Groups

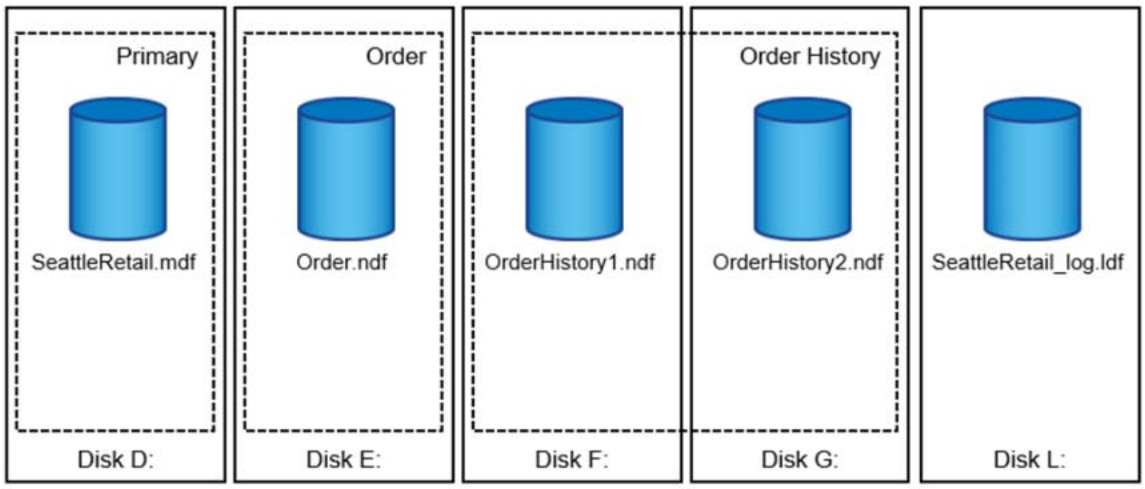
A SQL Server database is a collection of objects that allow one to store and manipulate data. In theory, SQL Server (64-bit) supports 32,767 databases per instance and 524,272TB of database space, although the typical installation usually has a few databases. However, the number of databases SQL Server can handle depends on the load and hardware. It is not unusual to see SQL Server instances hosting dozens or even hundreds or thousands of small databases.

Each database consists of one or more data files and one or more transaction log files. The transaction log stores the information about database transactions and all data modifications made by each session. Every time the data is modified, SQL Server stores enough information in the transaction log to undo (roll back) or redo (replay) the action. SQL Server’s transaction log is a part of SQL Server’s reputation for data integrity and robustness. The transaction log is vital to the atomicity, consistency, isolation, and durability (ACID) capabilities of SQL Server. SQL Server writes to the transaction log when any change to the data page occurs. Every data manipulation language (DML) statement (for example, select, insert, update, or delete) is a complete transaction, and the transaction log makes sure that the entire set-based operation takes place. In this way, the log makes sure of the atomicity of the transaction.

Each database has one primary data file, which, by default, has the .mdf extension. In addition, each database can have secondary database files, which have .ndf extensions by default. See Figure 3.

All database files are grouped into file groups. A file group is the logical unit, which simplifies database administration. This approach allows the separation between logical object placement and physical database files. When you create the database objects tables, you specify what file group they should be placed in without worrying about the underlying data file configuration.

Figure 3 Example of primary, secondary, and log files in each drive



The ability to put multiple data files inside the file group allows you to spread the load across different storage devices, which helps to improve I/O performance of the system. The transaction log, in contrast, does not benefit from the multiple files because SQL Server writes to the transaction log sequentially.

The separation between logical object placement in the file groups and physical database files allows you to fine-tune the database file layout, getting the most from the storage subsystem. For example, independent software vendors (ISVs) who are deploying their products to different customers can adjust the number of database files according to underlying I/O configuration and expected amount of data during the deployment stage. Those changes are transparent to the application developers, who are placing the database objects in the file groups rather than database files.

Microsoft recommends that you avoid using the primary file group for anything but system objects. Creating a separate file group or set of file groups for the user objects simplifies database administration and disaster recovery, especially if databases are large.

You can specify initial file size and autogrowth parameters when you create the database or add new files to an existing database. SQL Server uses a proportional fill algorithm when choosing what data file it should write data into. It writes an amount of data proportionally to the free space available in the files. The more free space in the file, the more writes it handles.

Best Practice

Fujitsu recommends that all files in the single file group have the same initial size and autogrowth parameters, with the grow size defined in megabytes rather than percentages. This approach helps the proportional-fill algorithm balance write activities across data files evenly.

SQL Server Database Files

You can provision database files on an ETERNUS AB series flash array in two ways:

- To create a database that has database files residing on an ETERNUS AB series LUN, you can use the following Transact-SQL (T-SQL) script during database creation:

```
-- Assuming C:\MSSQL\Data and C:\MSSQL\Log is the mount points of EF-Series LUNs
USE master;
GO
CREATE DATABASE Sales
ON
( NAME = Sales_dat,
 FILENAME =
 'C:\MSSQL\Data\saledat.mdf', SIZE =
 10,
 MAXSIZE = 50,
 FILEGROWTH =
 5 )
LOG ON
( NAME = Sales_log,
 FILENAME =
 'C:\MSSQL\Log\salelog.ldf', SIZE =
 5MB,
 MAXSIZE = 25MB,
 FILEGROWTH =
 5MB ) ;
GO
```

- To move the database files from non-ETERNUS AB series LUNs to ETERNUS AB series LUNs, you must detach the database files. After you detach the database files, you can copy the files to the path or mount points that reside on the ETERNUS AB series LUNs. After the files are copied, you can attach the database files at the new location.

The common best practice is to separate data, transaction logs, and tempdb files in separate logical LUNs. The origin of this recommendation lies with the separation of the types of workload between different physical storages. This recommendation is still a valid for environments in which you can have that separation. However, it is common to see customers deploying SQL Server in a shared storage environment where physical separation is much harder to achieve and usually is not even necessary for performance reasons.

It is still a good idea to maintain separation to help with manageability so that potential problems are easier to isolate. For example, separating tempdb onto its own logical disk means that you can presize it to fill the disk without worrying about space requirements for other files. The more separation you implement, the easier it is to correlate logical disk performance to specific database files.

Database Files for Tempdb

The tempdb system database is a global resource that is available to all users connected to the SQL Server instance, and it is used to hold the following:

- Temporary user objects that are explicitly created, such as global or local temporary tables, temporary stored procedures, table variables, or cursors.
- Internal objects that are created by the SQL Server Database Engine, such as worktables to store intermediate results for spools or sorting.
- Row versions that are generated by data modification transactions in a database that uses read-committed row versioning isolation or Snapshot isolation transactions.
- Row versions that are generated by data modification transactions for features, such as online index operations, multiple active result sets (MARS), and AFTER triggers.

Operations within tempdb are minimally logged, enabling transactions to be rolled back. Tempdb is re-created every time SQL Server is started so that the system starts with a clean copy of the tempdb database. Temporary tables and stored procedures are dropped automatically on disconnect, and no connections are active when the system is shut down. Therefore, there is nothing in tempdb to be saved from one session of SQL Server to another. Backup and restore operations are not allowed on tempdb.

Every SQL Server instance has a shared database named tempdb that is used by temporary objects and, because there is only one tempdb database per instance it often causes a bottleneck for systems that use it heavily. Typically, this bottleneck occurs because of PAGELATCH, an in-memory latch contention on the allocation bitmap pages inside the data files.

It is possible to reduce the contention on the in-memory pages by adding extra data files to tempdb with the same initial size and autogrowth configuration. This approach works because SQL Server uses a round-robin, proportional-fill algorithm to stripe the writes across the data files. When multiple data files exist for a database, all the writes to the files are striped to those files. The writes to any particular file are based on the proportion of free space that the file has compared to the total free space across all the files. Thus, writes are proportionally distributed to the files according to their free space, regardless of their size, to make sure that they fill at the same time.

Microsoft recommends up to a one-to-one mapping between the number of files and logical CPUs. During testing of massive workloads, Microsoft has seen performance benefits, even with hundreds of data files.

A more pragmatic approach, however, is to have a one-to-one mapping between files and logical CPUs up to eight. You can then add files if you continue to see allocation contention or if you must push the I/O subsystem harder.

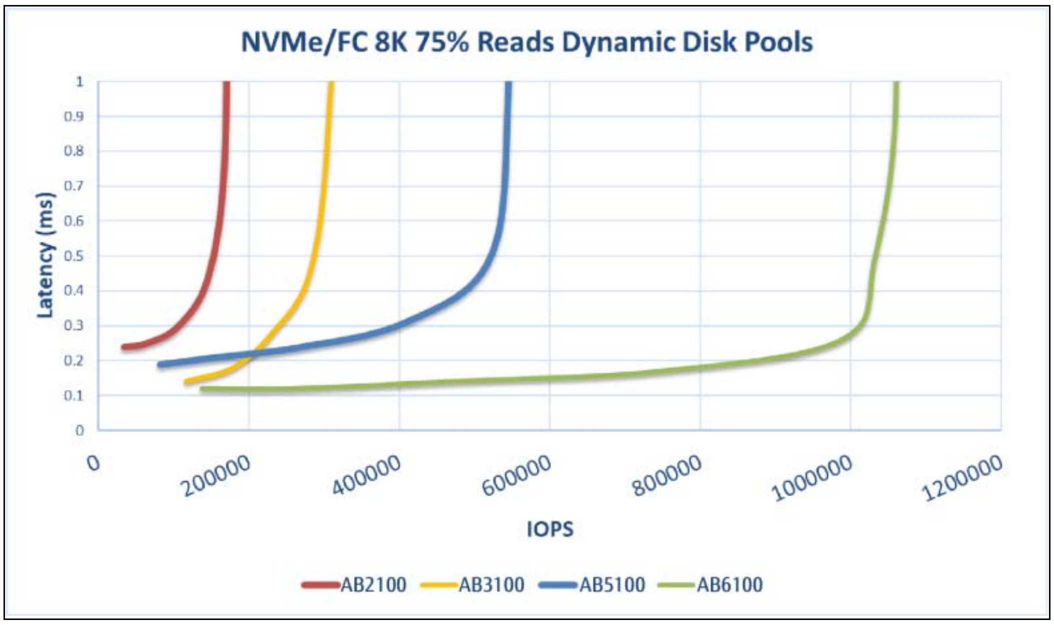
Because the ETERNUS AB series flash array provides a robust system for delivering exceptional performance, you can create multiple tempdb files and place them on the ETERNUS AB series flash array. You can accomplish this by using the following T-SQL Server script.

```
select *
from    sys.database_files

use master
go
-- Change logical tempdb file name first since SQL Server shipped with logical file name called
tempdev
alter database tempdb modify file (name = 'tempdev', newname = 'tempdev01');

-- Change location of tempdev01 and log file to C:\MSSQL\Tempdb path
alter database tempdb modify file (name = 'tempdev01', filename =
'C:\MSSQL\Tempdb\tempdev01.mdf');
alter database tempdb modify file (name = 'templog', filename = 'C:\MSSQL\Tempdb\templog.ldf');

-- Assign proper size for tempdev01
ALTER DATABASE [tempdb] MODIFY FILE ( NAME = N'tempdev01', SIZE = 2GB, FILEGROWTH = 100 MB );
ALTER DATABASE [tempdb] MODIFY FILE ( NAME = N'templog', SIZE = 4GB, FILEGROWTH = 100 MB);

-- Add more tempdb files
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev02', FILENAME =
N'C:\MSSQL\Tempdb\tempdev02.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev03', FILENAME =
N'C:\MSSQL\Tempdb\tempdev03.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev04', FILENAME =
N'C:\MSSQL\Tempdb\tempdev04.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev05', FILENAME =
N'C:\MSSQL\Tempdb\tempdev05.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev06', FILENAME =
N'C:\MSSQL\Tempdb\tempdev06.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev07', FILENAME =
N'C:\MSSQL\Tempdb\tempdev07.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev08', FILENAME =
N'C:\MSSQL\Tempdb\tempdev08.ndf' , SIZE = 2GB , FILEGROWTH = 100 MB);
```

4. ETERNUS AB series OLTP Performance

Figure 4 shows ETERNUS AB series performance results for the entry-level AB2100, the midrange AB5100, and the end-to-end NVMe AB3100 and AB6100 configured with DDP for a typical OLTP workload. Testing was done with the array configured to use 24 SSDs in a pool, the FC host protocol or NVMe/FC for the AB3100 and AB6100, and a workload of 75% read and 25% write. At approximately 300μs of latency, the AB2100 produces 100,000 IOPS, the AB5100 produces 400,000 IOPS, while the NVMe-based AB3100 and AB6100 deliver 240,000 and over 1,000,000 IOPS, respectively.

Figure 4 Performance results comparison for OLTP configuration



5. High Availability

ETERNUS AB series Systems and SANtricity OS

The ETERNUS AB series storage system has been architected for high reliability and high availability with the following features:

- Dual active controller with automated I/O path failover
- RAID levels 0, 1, 5, 6, and 10 or DDP
- Redundant, hot-swappable controllers, disks, power supplies, and fans
- Automatic drive failover detection and rebuild using global hot spares
- Mirrored data cache with battery backup and destage to memory
- Nondisruptive controller firmware upgrades
- Proactive drive-health monitoring
- Background media scan with autoparity check and correction

All components are fully redundant, and you can swap them without powering off the system or even halting operation. Redundant components include controllers, disks, power supplies, and fans. The ETERNUS AB series power supplies offer an 80-plus efficiency rating. The ETERNUS AB series flash array features several functions designed to protect data in every circumstance. Multiple RAID levels are available for use with varying levels of redundancy. If a connection is lost, failover from one path to another is also automatically included with the system. Within the shelf, each drive has a connection to each controller so that even internal connection issues can be quickly overcome. Volumes on the system are available for host I/O from the moment they are created and can even have significant properties altered without stopping I/O.

Other features of the ETERNUS AB series flash array that protect data include mirroring and backing up controller cache. If power is lost while the system is operating, onboard batteries destage the data from cache memory to internal controller flash so that it is available when power is restored. The RAID algorithms allow the system to recreate any lost data in the rare case of drive failure. You can also confirm data with RAID parity and even continue a rebuild if you hit an unreadable sector.

Behind the scenes, the system performs other tasks that protect data at all times. The optional media scan feature looks for inconsistencies even on sectors not currently accessed by any host. The ETERNUS AB series array proactively tracks SSD wear and flags drives that are approaching the end of their expected life. All types of diagnostic data are constantly collected for use later by Fujitsu support, if necessary.

As already described, the ETERNUS AB series array offers many reliability and availability features. In addition, SANtricity software makes it possible to maximize availability. For example, SANtricity:

- Enables high-speed, high-efficiency Snapshot technology
- Protects data in seconds
- Reduces flash consumption by storing only changed blocks
- Provides robust disaster recovery protection
- Supports synchronous mirroring for no-data-loss protection of content
- Supports asynchronous mirroring for long-distance protection and compliance
- Maximizes ROI with flexible protection
- Supports flash, near-line SAS (NL-SAS), or a mix of recovery targets based on cost and performance needs
- Delivers speed without breaking budgets

SQL Server HA Options

The high availability of SQL Server solutions masks the effects of a hardware or software failures and maintains the availability of applications to minimize the perceived downtime for users. SQL Server provides several high-availability (HA) solutions.

Log Shipping

Log shipping operates at the database level. It can maintain one or more warm standby databases (referred to as secondary databases) for a single production database that is referred to as the primary database. For more information about log shipping, see About Log Shipping (SQL Server).

Database Mirroring

Database mirroring increases database availability by supporting almost instantaneous failover. Database mirroring can be used to maintain a single standby database or mirror database for a corresponding production database that is referred to as the principal database. For more information, see Database Mirroring (SQL Server).

Always On Failover Cluster Instances

Always On Failover Cluster Instances applies Windows Server Failover Clustering (WSFC) functionality to provide local high availability through redundancy at the server instance level: a failover cluster instance (FCI). An FCI is a single SQL Server instance that is installed across WSFC nodes and possibly across multiple subnets. On the network, an FCI appears to be a SQL Server instance running on a single computer, but the FCI provides failover from one WSFC node to another if the current node becomes unavailable. For more information, see Always On Failover Cluster Instances (SQL Server).

Always On Availability Groups

Always On Availability Groups are an enterprise-level high-availability and disaster recovery solution introduced in SQL Server 2012 for maximizing availability for one or more user databases. Always On Availability Groups require that SQL Server instances reside on WSFC nodes. For more information, refer to Always On Availability Groups (SQL Server).

Always On Availability Groups supports two availability modes: asynchronous-commit mode and synchronous-commit mode.

- Asynchronous-commit mode is a disaster recovery solution that works well when the availability replicas are distributed over considerable distances. For more information, see Asynchronous-Commit Availability Mode.
- Synchronous-commit mode emphasizes high availability over performance, at the cost of increased transaction latency. In synchronous-commit mode, transactions wait to send the transaction confirmation to the client until the secondary replica has hardened the log to disk. For more information, see Synchronous-Commit Availability Mode.

Automatic failover provides high availability by making sure that the database is quickly made available again after the loss of the primary replica. To configure an availability group for automatic failover, you must set both the current primary replica and one secondary replica to synchronous-commit mode with automatic failover.

6. Sizing

SQL Server performance has been centered on I/O. Traditionally, users improved this performance by either increasing the number of spindles or making the spindles go more quickly. With the advent of the ETERNUS AB series flash array, you can improve performance by using SSDs.

ETERNUS AB series I/O Overview

There are several factors that can affect the overall performance of an ETERNUS AB series storage system, including physical components, such as networking infrastructure, and the configuration of the underlying storage itself. Generically, storage system performance tuning can be defined as following a 40/30/30 rule: 40% of the tuning and configuration is at the storage system level, 30% is at the file system level, and the final 30% is at the application level. The following sections describe the 40% related to storage system specifics. For the file system and application level, some of the general considerations include the following:

- **I/O size**
  ETERNUS AB series storage systems are largely responsive systems. To complete an I/O operation, they require a host to request that operation. The I/O size of the individual requests from the host can have a significant effect on either the number of IOPS or throughput, described in megabytes per second (MBps) or gigabytes per second (GBps). Larger I/O operations typically lead to lower numbers of IOPS and larger MBps, and the opposite is true as well. This relationship is defined with the equation Throughput = IOPS x I/O size.
- **Read versus write requests**
  In addition to the I/O size, the percentage of read versus write I/O requests processed at the storage system level also has a potential effect on the storage system. You should consider this percentage when designing a solution.
- **Sequential versus random data streams**
  Host requests to the underlying disk media logical block addresses can be sequential or random, which has a significant effect on performance at the storage system level. The sequence influences the ability of physical media to respond effectively to the request with minimal latency; the sequence also influences the effectiveness of the storage system’s caching algorithms. An exception to increased latency of random requests is for SSDs, which do not have mechanically invoked latency.
- **Number of concurrent I/O operations**
  The number of outstanding I/O operations applied to a given volume can vary according to several factors, including whether the file system uses raw, buffered, or direct I/O. Generally, most volumes in an ETERNUS AB series storage system are striped across several drives. Providing a minimal amount of outstanding I/O to each individual disk can cause underutilization of the resources in the storage system, resulting in less than desired performance characteristics.

If you are new to ETERNUS AB series technology, it might be helpful to review the differences between RAID 10, RAID 5, RAID 6, and DDP technology. Table 3 compares the usable capacity for different RAID levels. For completeness, all RAID levels supported by ETERNUS AB series systems are shown.

Table 3 Comparison of usable capacity for different RAID levels

Desired Feature	RAID 0	RAID 1 RAID 10	RAID 5	RAID 6	DDP
Usable capacity	100%	50%	(N-1) / N where N is the selected drive count in the volume group	(N-2) / N where N is the selected drive count in the volume group	80% minus selected preservation capacity

Estimating I/O

Estimating the number of I/O operations required for a system is crucial when you size a database. This exercise helps you understand how to keep the database instance performing within acceptable limits. You must estimate I/O when you are unable to get the actual physical I/O numbers for the system. This typically happens for new systems that are in the process of being constructed. The following sections provide formulas for estimating I/O.

New OLTP Database System

The following items must be considered:

- Business transaction
- Duration of business transaction
- Acceptable latency for the application
- Approximate ratio of transaction and system transactions (I/O)

The easiest way to estimate sizing for SQL Server with ETERNUS AB series systems is by following this example:

Procedure ▶▶▶

1. Estimate the number of business transactions. For example, estimate the number of business transactions as 600,000,000 per day. The business runs 24/7.
2. Assume that one business transaction creates 25 I/O operations. Therefore, for our example, the I/O operations per day are 600,000,000 x 25 = 15,000,000,000 I/O per day.
3. Using standard storage sizing tools requires IOPS information. From the example, this is (15,000,000,000) / 86,400 = 173,612 IOPS.
4. Acceptable latency for the application should be considered. For example, the application might want less than 1 millisecond of latency.
5. You must also estimate the percentage of reads and writes. 80% reads and 20% writes are typical of an OLTP database.

◀◀◀

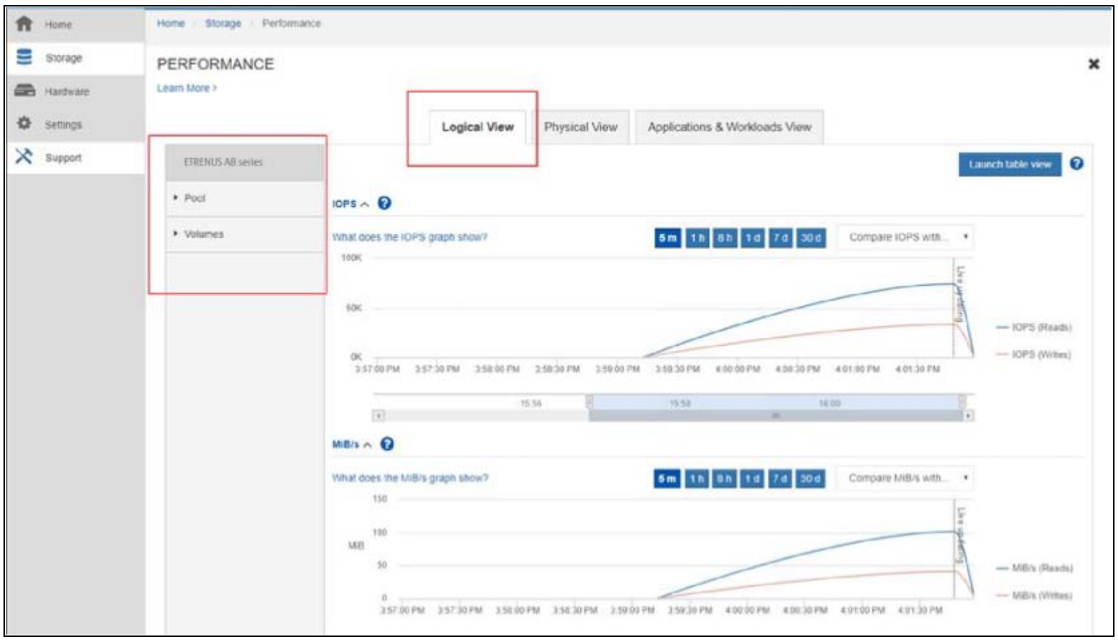
7. ETERNUS AB series Performance Monitoring Using SANtricity System Manager

During storage system operations, it can be useful to monitor storage system performance. Using SANtricity System Manager, you can view ETERNUS AB series performance data in both textual and graphical dashboard formats.

SANtricity System Manager provides exceptional performance monitoring that enables you to capture logical, physical, and application-level workloads. It adds new functionality, including application and workload tagging, enhanced performance data, an embedded monitor, and a graphical view of the volume usage. You can observe the performance monitor by selecting View Performance Details in SANtricity System Manager. You can see the performance of three main categories:

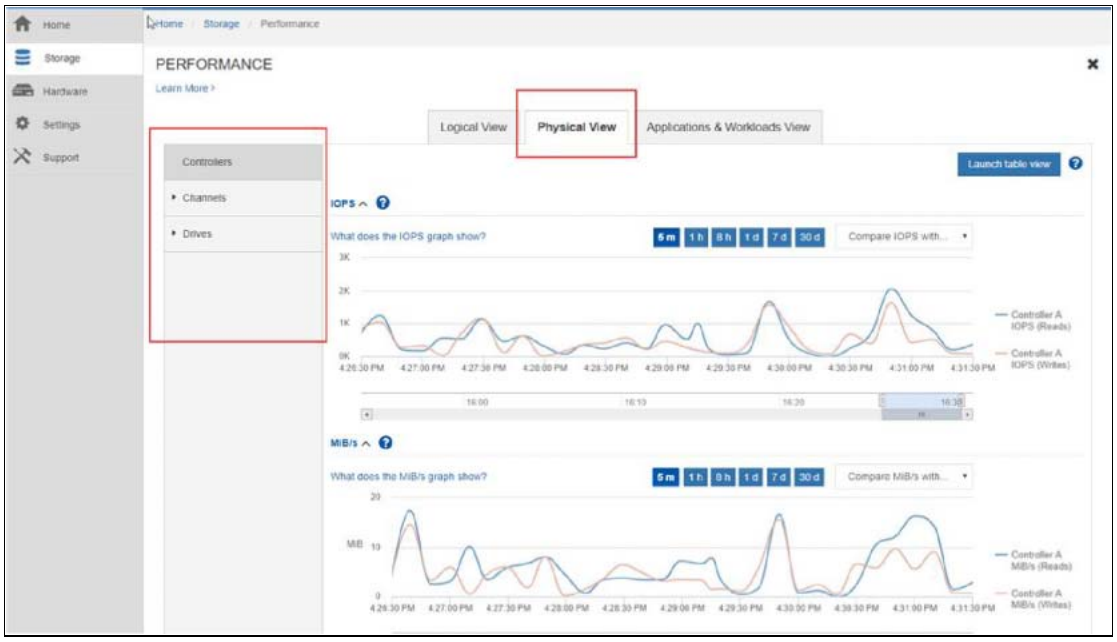
- The logical view enables you to filter information by pools and volume groups or volumes. See Figure 5.

Figure 5 Example of performance monitor with logical view



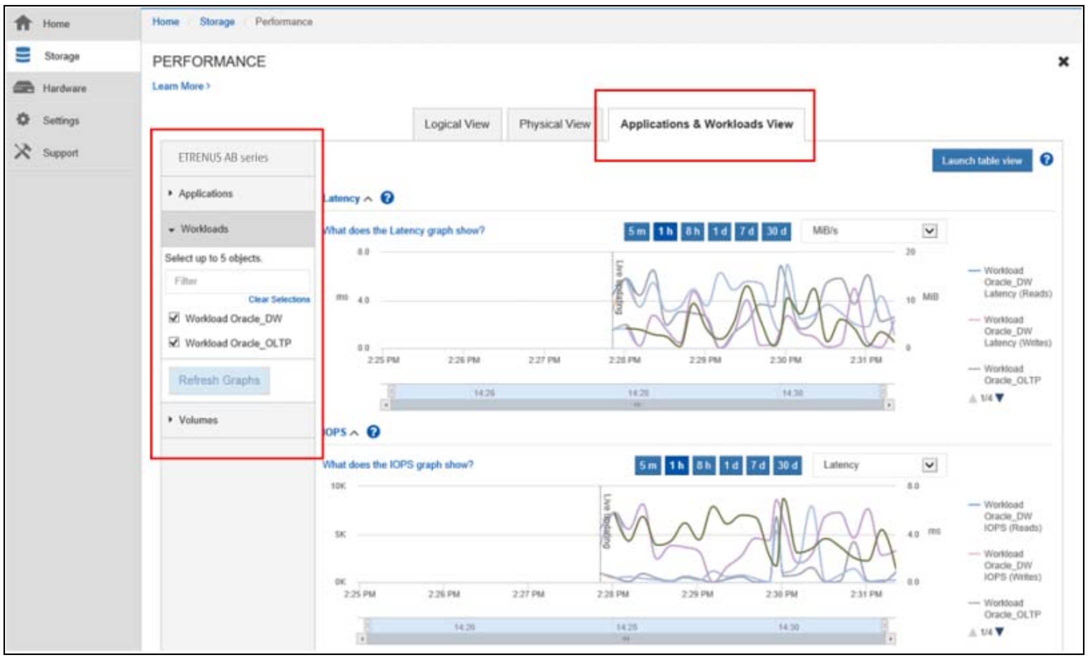
- The physical view enables you to filter information by host channels and drives. See Figure 6.

Figure 6 Example of performance monitor with physical view


Home
Storage
Hardware
Settings
Support
Home / Storage / Performance
PERFORMANCE
Learn More ›
Logical View
Physical View
Applications & Workloads View
Launch table view
Controllers
Channels
Drives
IOPS
What does the IOPS graph show?
5 m 1 h 8 h 1 d 7 d 30 d
Compare IOPS with...
Controller A IOPS (Reads)
Controller A IOPS (Writes)
MiB/s
What does the MiB/s graph show?
Compare MiB/s with...
Controller A MiB/s (Reads)
Controller A MiB/s (Writes)

- The applications and workloads view enables you to filter information by applications, workloads, and volumes within the workloads. See Figure 7.

Figure 7 Example of performance monitor with applications and workloads view



8. Conclusion

In this document, we present the ETERNUS AB series solution for high-performance MS SQL Server databases. This solution allows you to use best-in-class, end-to-end, modern SAN and NVMe technologies to deliver business-critical IT services today, while preparing for the future.

With the ETERNUS AB series, Fujitsu has created a SAN array that is future-ready, usable today, and easily implemented within your current operational processes and procedures. The ETERNUS AB series flash array is a market leader in delivering high performance, consistent low latency, and advanced HA features.

The array is easy to provision with the embedded System Manager. If you have many systems to prepare, you can also provision the array with a REST API. (See Appendix.)

The ETERNUS AB series flash array also has robust built-in monitoring capabilities that enable you to troubleshoot performance issues at the logical level, the physical level, and the application level.

In addition to solving extreme latency requirements, the ETERNUS AB series flash array also resolves SQL Server database challenges in the following ways:

- It dramatically boosts the performance of existing applications and lowers the cost per IOPS without requiring that you rearchitect the application.
- It increases SQL Server performance with RAID10 and DDP.
- Better response times increase user productivity, improving business efficiency.
- SQL Server Always On Availability Groups provide an alternative solution for high availability and disaster recovery.

A. Provisioning ETERNUS AB series Systems with the REST API

```
# EF-Series System Manager RESTAPI Volume Provisioning

# =========================
# Globals
# =========================

# bypass untrusted certificates -- only do this for sources you trust!
add-type @"
    using System.Net;
    using System.Security.Cryptography.X509Certificates;
    public class TrustAllCertsPolicy : ICertificatePolicy {
        public bool CheckValidationResult(
            ServicePoint srvPoint, X509Certificate certificate,
            WebRequest request, int certificateProblem)
            { return true;
        }
    }
"@
[System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy
[Net.ServicePointManager]::SecurityProtocol = "Tls12, Tls11, Tls, Ssl3"

# IPs and credentials
$EFSeriesC2A = "https://10.251.229.52:8443/devmgr/"
$username = "admin"
$pass = "Infinit1"

# persist web session after logging in
$session = New-Object Microsoft.Powershell.Commands.WebRequestSession

# headers
$headers = @{}
$headers.Add('accept','application/json')
$headers.Add('Content-Type','application/json')

# misc
$storageSystemId = "1"
```

```
# =========================
# Initiate Session
# =========================
# as alternative, can also store creds as plaintext in url:
# Invoke-RestMethod -Method Get -Uri
($EFSeriesC2A+"devmgr/utils/login?uid=admin&pwd=myPassword123&xsrf=false&onlycheck=false") -
Headers $headers

Function LoginAPI ($username, $pass) {
    $credentialsJSON = '
    {
      "userId": "' + $username + '",
      "password": "' + $pass + '"
    }'

    $restParams = @{
        Method     = 'Post'
        Uri        = $EFSeriesC2A+"utils/login"
        Headers    = $headers
        Body       = $credentialsJSON
        WebSession = $session
    }

    Invoke-RestMethod @restParams
}
# LoginAPI $username $pass
# =========================
# GET Functions
# =========================
# Ex endpoints:
# Invoke-RestMethod -Method Get -Uri ($EFSeriesC2A+"devmgr/utils/about") -Headers $headers
# Invoke-RestMethod -Method Get -Uri ($EFSeriesC2A+"devmgr/utils/buildinfo") -Headers $headers

Function GetStorageSystems {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems"
        Headers    = $headers
        WebSession = $session
    }

    try {
        Invoke-RestMethod @restParams
    }
    catch {
        CatchException $restParams
    }
}
# GetStorageSystems


Function GetEvents {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/events"
        Headers    = $headers
        WebSession = $session
    }

    try {
        Invoke-RestMethod @restParams
    }
    catch {
        CatchException $restParams
    }
}
# GetEvents

Function GetVolumes ($getIdsOnly) {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/volumes"
        Headers    = $headers
        WebSession = $session
    }

    try {
        if ($getIdsOnly -eq $true) {
            (Invoke-RestMethod @restParams).id
        }
        else {
            (Invoke-RestMethod @restParams)
        }
    }
    catch {
        CatchException $restParams
    }
}
# GetVolumes $true # '$false' returns all volume details


Function GetControllerId {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/controllers"
        Headers    = $headers
        WebSession = $session
    }

    try {
        (Invoke-RestMethod @restParams).controllerRef[0]
    }
    catch {
        CatchException $restParams
    }
```

```
        else {
            (Invoke-RestMethod @restParams)
        }
    }
    catch {
        CatchException $restParams
    }
}
# GetDriveIds $true # '$false' returns all drive details

Function GetSnapshots {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/snapshot-images"
        Headers    = $headers
        WebSession = $session
    }

    try {
        (Invoke-RestMethod @restParams)
    }
    catch {
        CatchException $restParams
    }
}
# GetSnapshots}
# GetControllerId

Function GetStoragePoolId {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/storage-pools"
        Headers    = $headers
        WebSession = $session
    }

    try {
        (Invoke-RestMethod @restParams).volumeGroupRef
    }
    catch {
        CatchException $restParams
    }
}
# GetStoragePoolId

Function GetHosts {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/hosts"
        Headers    = $headers
        WebSession = $session
    }

    try {
        Invoke-RestMethod @restParams
    }
    catch {
        CatchException $restParams
    }
}
# GetHosts

Function GetDriveIds ($getIdsOnly) {
    $restParams = @{
        Method     = 'Get'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/drives"
        Headers    = $headers
        WebSession = $session
    }

    try {
        if ($getIdsOnly -eq $true) {
            (Invoke-RestMethod @restParams).id
        }
```

```
# =========================
# POST Functions
# =========================

Function CreateVolume ($volName, $size) {
    $controllerId = GetControllerId
    $poolId = GetStoragePoolId
    $volumeDetails = '
    {
       "poolId": "' + $poolId + '",
       "name": "' + $volName + '",
       "sizeUnit": "gb",
       "size": "' + $size + '",
       "segSize": 0,
       "dataAssuranceEnabled": false,
       "owningControllerId": "' + $controllerId + '",
       "metaTags": [
        {
            "key": "string",
            "value": "string"
        }
       ]
    }'

    $restParams = @{
        Method     = 'Post'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/volumes"
        Headers    = $headers
        Body       = $volumeDetails
        WebSession = $session
    }

    try {
        Invoke-RestMethod @restParams
    }
  catch {
        CatchException $restParams
    }
}
# CreateVolume delThisVolume 25

Function MapVolume ($targetHostId, $targetVolId) {
    $mappingDetails = '
    {
        "mappableObjectId": "' + $targetVolId + '",
        "targetId": "' + $targetHostId + '"
    }'

    $restParams = @{
        Method     = 'Post'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/volume-mappings"
        Headers    = $headers
        Body       = $mappingDetails
        WebSession = $session
    }

    try {
        Invoke-RestMethod @restParams
    }
    catch {
        CatchException $restParams
    }
}
# MapVolume "84000000600A098000BF6F310030060D5C3C58CA" "02000000600A098000BF85F300001E555C41E60F"
# =========================
# DELETE Functions
# =========================

Function DeleteVolume ($targetVolId){
    $restParams = @{
        Method     = 'Delete'
        Uri        = $EFSeriesC2A+"v2/storage-systems/"+$storageSystemId+"/volumes/"+$targetVolId
        Headers    = $headers
        Body       = $mappingDetails
        WebSession = $session
    }

    try {
        Invoke-RestMethod @restParams
    }
    catch {
        CatchException $restParams
    }
}
# DeleteVolume 02000000600A098000BF85F300001E615C45CC56



# =========================
# Exception Handler
# =========================

Function CatchException ($restParams) {
    if ($_.Exception.Message -like "*401*") {
        Write-Host "Not logged in -- attempting log in now . . ."
        LoginAPI $username $pass
        Invoke-RestMethod @restParams
    }
    else {
        Write-Host ($_.Exception.Message)
    }
}
```

```
# =========================
# Function Calls
# =========================

# new session:
$session = New-Object Microsoft.Powershell.Commands.WebRequestSession

GetHosts
```

FUJITSU Storage
ETERNUS AB series All-Flash Arrays
Best Practices Guide for
Microsoft SQL Server with ETERNUS AB series

P3AG-6612-01ENZ0

Date of issuance: June 2022
Issuance responsibility: FUJITSU LIMITED