Storage configuration

Enterprise applications

NetApp
March 13, 2024

Table of Contents

Storage configuration

Microsoft SQL Server database files and filegroups

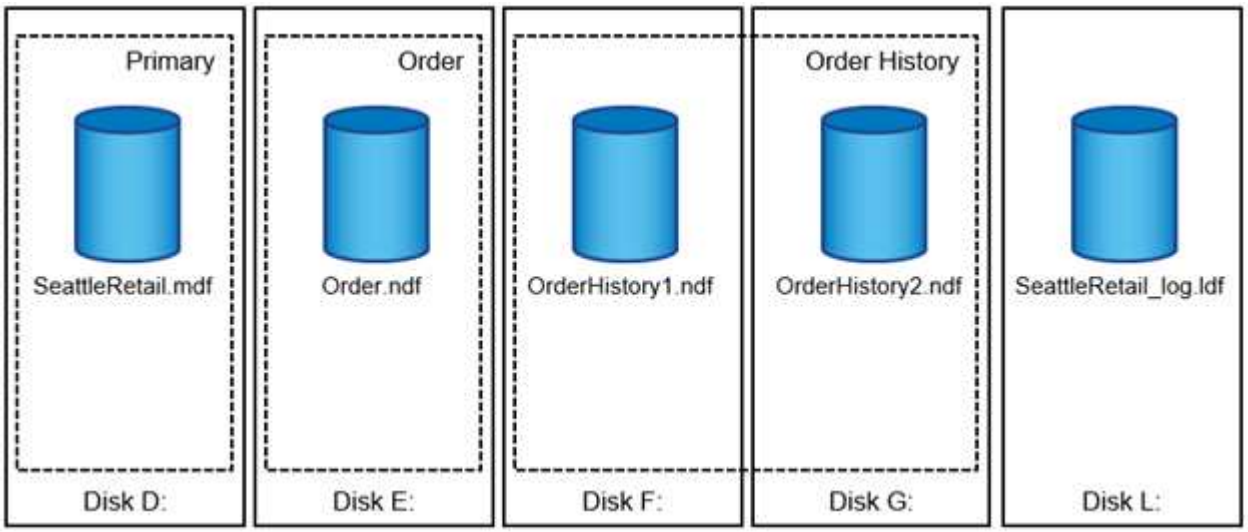
A SQL Server database is a collection of objects that allows you to store and manipulate data.

In theory, SQL Server (64-bit) supports 32,767 databases per instance and 524,272TB of database size, although the typical installation usually has several databases. However, the number of the databases SQL Server can handle depends on the load and hardware. It is not unusual to see SQL Server instances hosting dozens, hundreds, or even 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. A SQL Server transaction log is an integral 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 as soon as any change to the data page happens. 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, making 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. Those files, by default, have .ndf extensions.

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



The ability to put multiple data files inside the filegroup enables you to spread the load across different storage devices, which helps to improve the 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 in a sequential manner.

The separation between logical object placement in the filegroups 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 based on the underlying I/O configuration and the expected amount of data during the deployment stage. Those changes are transparent to the application developers, who are placing the database objects in the filegroups rather than database files.

NetApp recommends avoiding the use of the primary filegroup for anything but system objects. Creating a separate filegroup or set of filegroups for the user objects simplifies database administration and disaster recovery, especially in the case of large databases.

You can specify initial file size and autogrowth parameters at the time when you create the database or add new files to an existing database. SQL Server uses a proportional fill algorithm when choosing which 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.

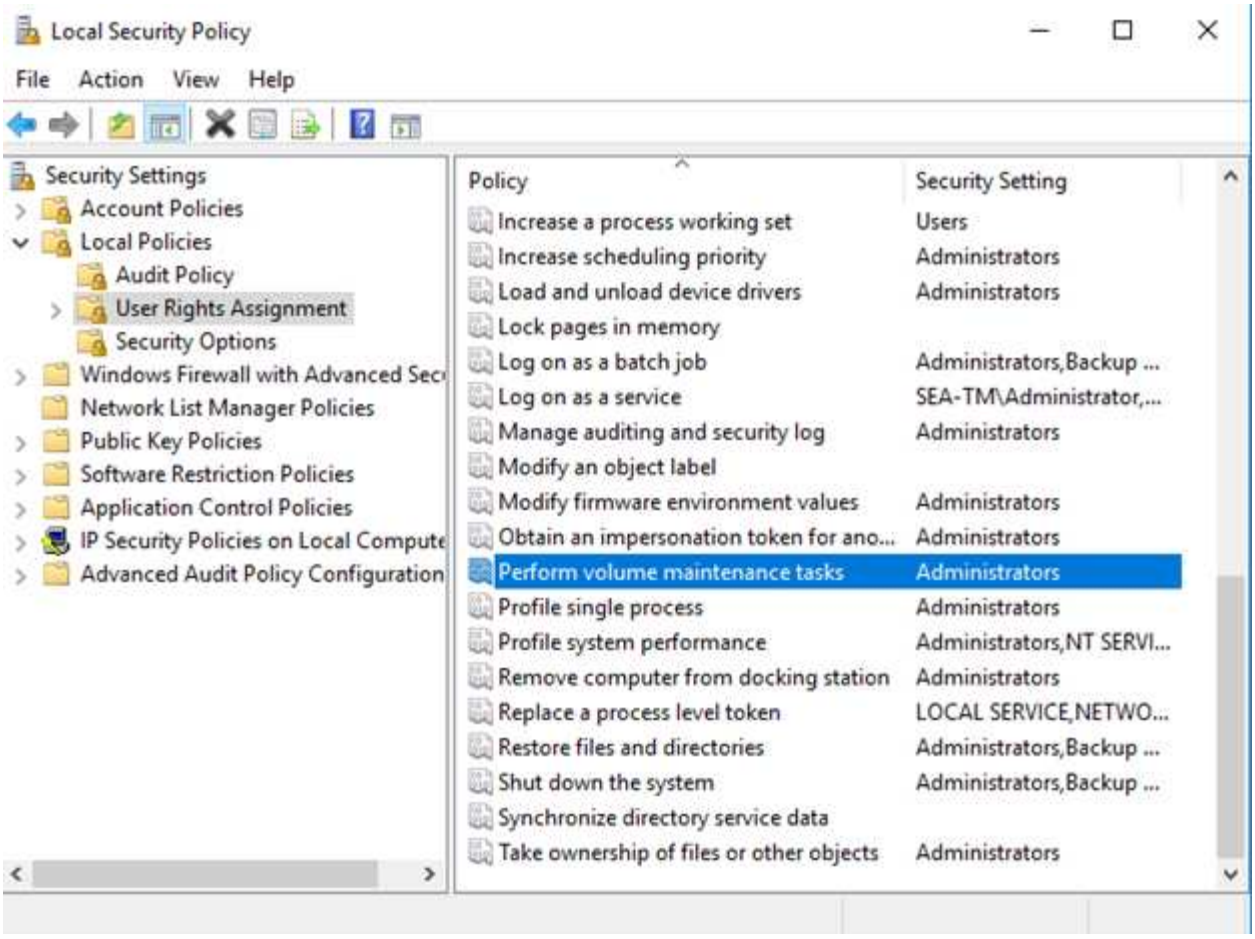
NetApp recommends that all files in the single filegroup have the same initial size and autogrowth parameters, with the grow size defined in megabytes rather than percentages. This helps the proportional fill algorithm evenly balance write activities across data files.

Every time SQL Server grows the files, it fills newly allocated space in the files with zeros. That process blocks all sessions that need to write to the corresponding file or, in case of transaction log growth, generate transaction log records.

SQL Server always zeroes out the transaction log, and that behavior cannot be changed. However, you can control whether data files are zeroing out by enabling or disabling instant file initialization. Enabling instant file initialization helps to speed up data file growth and reduces the time required to create or restore the database.

A small security risk is associated with instant file initialization. When this option is enabled, unallocated parts of the data file can contain information from previously deleted OS files. Database administrators can examine such data.

You can enable instant file initialization by adding the SA_MANAGE_VOLUME_NAME permission, also known as “perform volume maintenance task,” to the SQL Server startup account. You can do this under the local security policy management application (secpol.msc), as shown in the following figure. Open the properties for the “perform volume maintenance task” permission and add the SQL Server startup account to the list of users there.



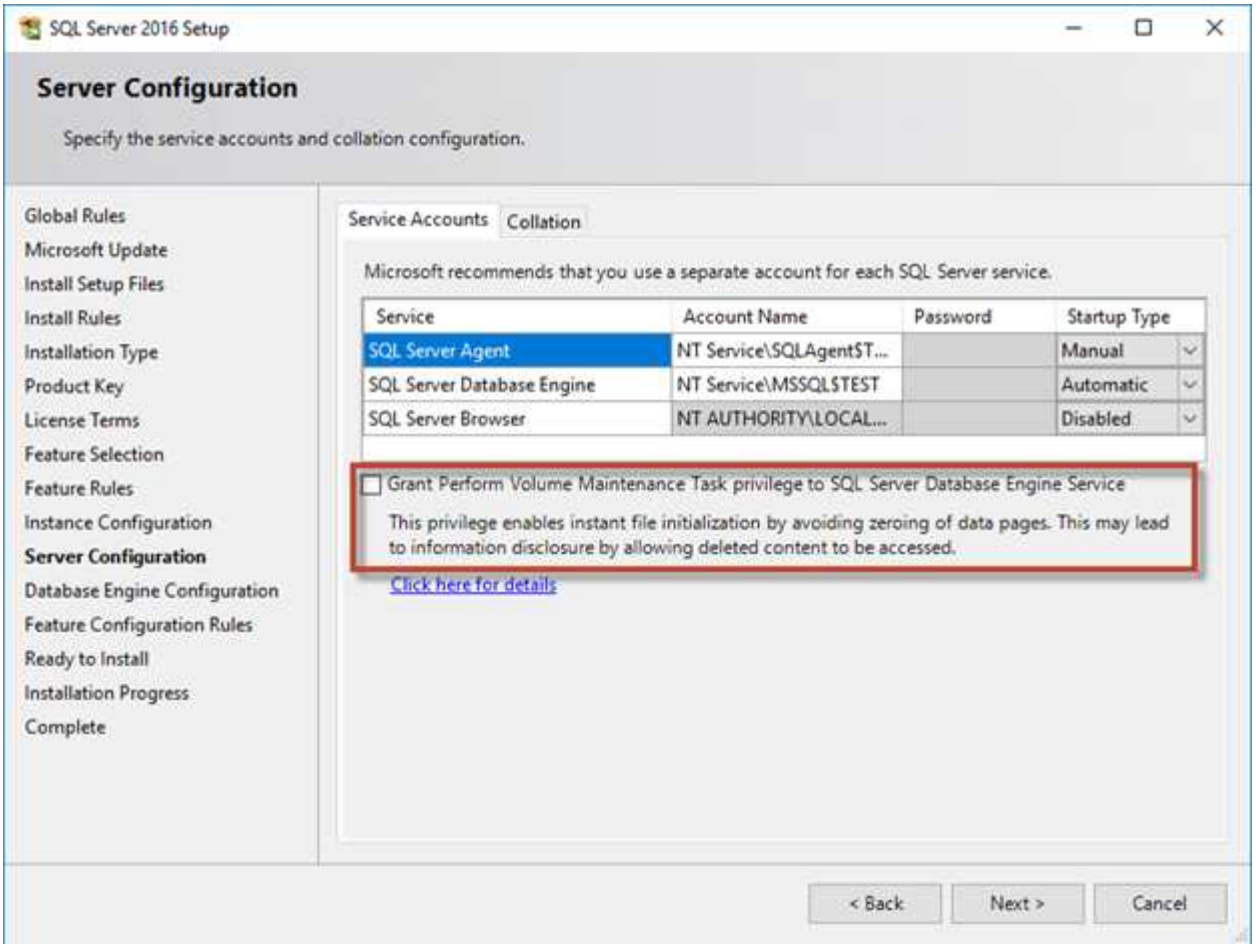
To check if the permission is enabled, you can use the code from the following example. This code sets two trace flags that force SQL Server to write additional information to the error log, create a small database, and read the content of the log.

```
DBCC TRACEON(3004,3605,-1)

GO

CREATE DATABASE DelMe

GO

EXECUTE sp_readerrorlog

GO

DROP DATABASE DelMe

GO

DBCC TRACEOFF(3004,3605,-1)

GO
```

When instant file initialization is not enabled, the SQL Server error log shows that SQL Server is zeroing the mdf data file in addition to zeroing the ldf log file, as shown in the following example. When instant file initialization is enabled, it displays only zeroing of the log file.

	LogDate	ProcessInfo	Text
365	2017-02-09 08:10:07.660	spid53	Ckpt dbid 3 flush delta counts.
366	2017-02-09 08:10:07.660	spid53	Ckpt dbid 3 logging active xact info.
367	2017-02-09 08:10:07.750	spid53	Ckpt dbid 3 phase 1 ended (8)
368	2017-02-09 08:10:07.750	spid53	About to log Checkpoint end.
369	2017-02-09 08:10:07.880	spid53	Ckpt dbid 3 complete
370	2017-02-09 08:10:08.130	spid53	Starting up database 'DelMe'.
371	2017-02-09 08:10:08.150	spid53	FixupLogTail(progress) zeroing C:\Program Files\Micros
372	2017-02-09 08:10:08.160	spid53	Zeroing C:\Program Files\Microsoft SQL Server\MSSQ
373	2017-02-09 08:10:08.170	spid53	Zeroing completed on C:\Program Files\Microsoft SQL
374	2017-02-09 08:10:08.710	spid53	Ckpt dbid 6 started
375	2017-02-09 08:10:08.710	spid53	About to log Checkpoint begin.

The perform volume maintenance task is simplified in SQL Server 2016 and is later provided as an option during the installation process. This figure displays the option to grant the SQL Server database engine service the privilege to perform the volume maintenance task.



Another important database option that controls the database file sizes is autoshrink. When this option is enabled, SQL Server regularly shrinks the database files, reduces their size, and releases space to the operating system. This operation is resource intensive and is rarely useful because the database files grow again after some time when new data comes into the system. Autoshrink must never be enabled on the database.

Microsoft SQL Server tempdb files

NetApp recommends proactively inflating tempdb files to their full size to avoid disk fragmentation.

Page contention can occur on lobal allocation map (GAM), shared global allocation map (SGAM), or page free space (PFS) pages when SQL Server must write to special system pages to allocate new objects. Latches protect (lock) these pages in memory. On a busy SQL Server instance, it can take a long time to get a latch on a system page in tempdb. This results in slower query run times and is known as latch contention. See the following best practices for creating tempdb data files:

- For < or = to 8 cores: tempdb data files = number of cores
- For > 8 cores: 8 tempdb data files

The following example script modifies tempdb by creating eight tempdb files and moving tempdb to the mount point `C:\MSSQL\tempdb` for SQL Server 2012 and later.

```
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

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');

GO

-- Assign proper size for tempdev01

ALTER DATABASE [tempdb] MODIFY FILE ( NAME = N'tempdev01', SIZE = 10GB );

ALTER DATABASE [tempdb] MODIFY FILE ( NAME = N'templog', SIZE = 10GB );

GO

-- Add more tempdb files

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev02', FILENAME =
N'C:\MSSQL\tempdb\tempdev02.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev03', FILENAME =
N'C:\MSSQL\tempdb\tempdev03.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev04', FILENAME =
N'C:\MSSQL\tempdb\tempdev04.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev05', FILENAME =
N'C:\MSSQL\tempdb\tempdev05.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev06', FILENAME =
N'C:\MSSQL\tempdb\tempdev06.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev07', FILENAME =
```

```
N'C:\MSSQL\tempdb\tempdev07.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev08', FILENAME =
N'C:\MSSQL\tempdb\tempdev08.ndf' , SIZE = 10GB , FILEGROWTH = 10%);

GO
```

Beginning with SQL Server 2016, the number of CPU cores visible to the operating system is automatically detected during installation and, based on that number, SQL Server calculates and configures the number of tempdb files required for optimum performance.

Microsoft SQL Server storage considerations

The combination of NetApp storage solutions and Microsoft SQL Server enables the creation of enterprise-level database storage designs that can meet today's most demanding application requirements.

To optimize both technologies, it is vital to understand the SQL Server I/O pattern and characteristics. A well-designed storage layout for a SQL Server database supports the performance of SQL Server and the management of the SQL Server infrastructure. A good storage layout also allows the initial deployment to be successful and the environment to grow smoothly over time as the business grows.

Data storage design

For SQL Server databases that do not use SnapCenter to perform backups, Microsoft recommends placing the data and log files on separate drives. For applications that simultaneously update and request data, the log file is write intensive, and the data file (depending on your application) is read/write intensive. For data retrieval, the log file is not needed. Therefore, requests for data can be satisfied from the data file placed on its own drive.

When you create a new database, Microsoft recommends specifying separate drives for the data and logs. To move files after the database is created, the database must be taken offline. For more Microsoft recommendations, see Place Data and Log Files on Separate Drives.

Aggregates

Aggregates are the primary storage containers for NetApp storage configurations and contain one or more RAID groups consisting of both data disks and parity disks. NetApp has performed various I/O workload characterization tests using shared and dedicated aggregates with data files and transaction log files separated. The tests show that one large aggregate with more RAID groups and spindles optimizes and improves storage performance and is easier for administrators to manage for two reasons:

- One large aggregate makes the I/O capabilities of all spindles available to all files.
- One large aggregate enables the most efficient use of disk space.

For high availability (HA), place the SQL Server Always On Availability Group secondary synchronous replica on a separate storage virtual machine (SVM) in the aggregate. For disaster recovery purposes, place the asynchronous replica on an aggregate that is part of a separate storage cluster in the DR site, with content replicated by using NetApp SnapMirror technology. NetApp recommends having at least 10% free space available in an aggregate for optimal storage performance.

Volumes

NetApp FlexVol volumes are created and reside inside aggregates. Many volumes can be created in a single aggregate, and each volume can be expanded, shrunk, or moved between aggregates with no user downtime.

Volume design considerations

Before you create a database volume design, it is important to understand how the SQL Server I/O pattern and characteristics vary depending on the workload and on the backup and recovery requirements. See the following NetApp recommendations for flexible volumes:

- Use flexible volumes to store SQL Server database files and avoid sharing volumes between hosts.
- Use NTFS mount points instead of drive letters to surpass the 26-drive-letter limitation in Windows. When using volume mount points, it is a general recommendation to give the volume label the same name as the mount point.
- When appropriate, configure a volume autosize policy to help prevent out-of-space conditions. 17 Best practice guide for Microsoft SQL Server with ONTAP © 2022 NetApp, Inc. All rights reserved.
- Enable read reallocation on the volume when the SQL Server database I/O profile consists of mostly large sequential reads, such as with decision support system workloads. Read reallocation optimizes the blocks to provide better performance.
- If you install SQL Server on an SMB share, make sure that Unicode is enabled on the SMB/CIFS volumes for creating folders.
- Set the NetApp snapshot copy reserve value in the volume to zero for ease of monitoring from an operational perspective.
- Disable storage Snapshot™ copy schedules and retention policies. Instead, use SnapCenter to coordinate Snapshot copies of the SQL Server data volumes.
- Place the SQL Server system databases on a dedicated volume or VMDK.
- tempdb is a system database used by SQL Server as a temporary workspace, especially for I/O intensive DBCC CHECKDB operations. Therefore, place this database on a dedicated volume with a separate set of spindles. In large environments in which volume count is a challenge, you can consolidate tempdb into fewer volumes and store it in the same volume as other system databases after careful planning. Data protection for tempdb is not a high priority because this database is recreated every time SQL Server is restarted.
- Place user data files (.mdf) on separate volumes because they are random read/write workloads. It is common to create transaction log backups more frequently than database backups. For this reason, place transaction log files (.ldf) on a separate volume or VMDK from the data files so that independent backup schedules can be created for each. This separation also isolates the sequential write I/O of the log files from the random read/write I/O of data files and significantly improves SQL Server performance.

LUN

- Make sure that the user database files and the log directory to store log backup are on separate volumes to prevent the retention policy from overwriting snapshots when these are used with SnapVault technology.
- Make sure that SQL Server databases reside on LUNs separate from LUNs that have non-database files, such as full-text search-related files.
- Placing database secondary files (as part of a filegroup) on separate volumes improves the performance of the SQL Server database. This separation is valid only if the database's .mdf file does not share its LUN with any other .mdf files.

- If you create LUNs with DiskManager or other tools, make sure that the allocation unit size is set to 64K for partitions when formatting the LUNs.
- See the Microsoft Windows and native MPIO under ONTAP best practices for modern SAN to apply multipathing support on Windows to iSCSI devices in the MPIO properties.

Log directory

Log directory is specified in SQL Server to store transaction log backup data at the host level. If you are using SnapCenter to backup log files then each SQL Server host used by SnapCenter must have a host log directory configured to perform log backups. SnapCenter has a database repository, so metadata related to backup, restore, or cloning operations is stored in a central database repository.

The sizes of the host log directory is calculated as follows: Size of host log directory = ((maximum DB LDF size x daily log change rate %) x (snapshot retention) ÷ (1 - LUN overhead space %) The host log directory sizing formula assumes a 10% LUN overhead space

Place the log directory on a dedicated volume or LUN. The amount of data in the host log directory depends on the size of the backups and the number of days that backups are retained. SnapCenter allows only one host log directory per SQL Server host. You can configure the host log directories at SnapCenter -→ Host -→ Configure Plug-in.

NetApp recommends the following for a host log directory:

- Make sure that the host log directory is not shared by any other type of data that can potentially corrupt the backup snapshot data.
- Do not place user databases or system databases on a LUN that hosts mount points.
- Create the host log directory on the dedicated FlexVol volume to which SnapCenter copies transaction logs.
- Use SnapCenter wizards to migrate databases to NetApp storage so that the databases are stored in valid locations, enabling successful SnapCenter backup and restore operations. Keep in mind that the migration process is disruptive and can cause the databases to go offline while the migration is in progress.
- The following conditions must be in place for failover cluster instances (FCIs) of SQL Server:
  - If you are using a failover cluster instance, the host log directory LUN must be a cluster disk resource in the same cluster group as the SQL Server instance being backed up SnapCenter.
  - If you are using a failover cluster instance, user databases must be placed on shared LUNs that are physical disk cluster resources assigned to the cluster group associated with the SQL Server instance.

ONTAP Storage efficiency with Microsoft SQL Server

Storage efficiency is the ability to store and manage SQL Server data in a way that consumes the least amount of storage space with little or no effect on the overall performance of the system.

Storage efficiency is a combination of RAID, provisioning (overall layout and utilization), mirroring, and other data protection technologies. NetApp technologies like Snapshot copies, Thin provisioning, FlexClone helps in creating cost benefits by optimizing existing storage in the infrastructure and deferring or avoiding future

storage expenditures. The more you use these technologies together, the larger the savings.

Thin provisioning

Thin provisioning comes in many forms and is integral to many features that ONTAP offers to an enterprise application environment. Thin provisioning is also closely related to efficiency technologies for the same reason: efficiency features allow more logical data to be stored than what technically exists on the storage system.

Almost any use of snapshots involves thin provisioning. For example, a typical 10TB database on NetApp storage includes around 30 days of snapshots. This arrangement results in approximately 10TB of data visible in the active file system and 300TB dedicated to snapshots. The total 310TB of storage usually resides on approximately 12TB to 15TB of space. The active database consumes 10TB, and the remaining 300TB of data only requires 2TB to 5TB of space because only the changes to the original data are stored.

Cloning is also an example of thin provisioning. A major NetApp customer created 40 clones of an 80TB database for use by development. If all 40 developers using these clones overwrote every block in every datafile, over 3.2PB of storage would be required. In practice, turnover is low, and the collective space requirement is closer to 40TB because only changes are stored on the drives.

Space management

Some care must be taken with thin provisioning an application environment because data change rates can increase unexpectedly. For example, space consumption due to snapshots can grow rapidly if database tables are reindexed, or wide-scale patching is applied to VMware guests. A misplaced backup can write a large amount of data in a very short time. Finally, it can be difficult to recover some applications if a file system runs out of free space unexpectedly.

Fortunately, these risks can be addressed with careful configuration of `volume-autogrow` and `snapshot-autodelete` policies. As their names imply, these options enable a user to create policies that automatically clear space consumed by snapshots or grow a volume to accommodate additional data. Many options are available and needs vary by customer.

See the logical storage management documentation for a complete discussion of these features.

LUN thin provisioning

The efficiency of thin provisioning of active LUNs in a file system environment can be lost over time as data is deleted. Unless that deleted data is either overwritten with zeros or the space is released with TRIM/UNMAP space reclamation, the "erased" data occupies more and more unallocated whitespace in the file system. Furthermore, thin provisioning of active LUNs is of limited use in many database environments because datafiles are initialized to their full size at the time of creation.

Careful planning of LVM configuration can improve efficiency and minimize the need for storage provisioning and LUN resizing. When an LVM such as Veritas VxVM or Oracle ASM is used, the underlying LUNs are divided into extents that are only used when needed. For example, if a dataset begins at 2TB in size but could grow to 10TB over time, this dataset could be placed on 10TB of thin-provisioned LUNs organized in an LVM diskgroup. It would occupy only 2TB of space at the time of creation and would only claim additional space as extents are allocated to accommodate data growth. This process is safe as long as space is monitored.

Fractional reservations

Fractional reserve refers to the behavior of a LUN in a volume with respect to space efficiency. When the option `fractional-reserve` is set to 100%, all data in the volume can experience 100% turnover with any

data pattern without exhausting space on the volume.

For example, consider a database on a single 250GB LUN in a 1TB volume. Creating a snapshot would immediately result in the reservation of an additional 250GB of space in the volume to guarantee that the volume does not run out of space for any reason. Using fractional reserves is generally wasteful because it is extremely unlikely that every byte in the database volume would need to be overwritten. There is no reason to reserve space for an event that never happens. Still, if a customer cannot monitor space consumption in a storage system and must be certain that space never runs out, 100% fractional reservations would be required to use snapshots.

Compression and deduplication

Compression and deduplication are both forms of thin provisioning. For example, a 50TB data footprint might compress to 30TB, resulting in a savings of 20TB. For compression to yield any benefits, some of that 20TB must be used for other data, or the storage system must be purchased with less than 50TB. The result is storing more data than is technically available on the storage system. From the data point of view, there is 50TB of data, even though it occupies only 30TB on the drives.

There is always a possibility that the compressibility of a dataset changes, which would result in increased consumption of real space. This increase in consumption means that compression must be managed as with other forms of thin provisioning in terms of monitoring and using `volume-autogrow` and `snapshot-autodelete`.

Compression and deduplication are discussed in further detail in the section xref:./mssql/efficiency.html

Compression and fractional reservations

Compression is a form of thin provisioning. Fractional reservations affect the use of compression, with one important note; space is reserved in advance of the snapshot creation. Normally, fractional reserve is only important if a snapshot exists. If there is no snapshot, fractional reserve is not important. This is not the case with compression. If a LUN is created on a volume with compression, ONTAP preserves space to accommodate a snapshot. This behavior can be confusing during configuration, but it is expected.

As an example, consider a 10GB volume with a 5GB LUN that has been compressed down to 2.5GB with no snapshots. Consider these two scenarios:

- Fractional reserve = 100 results in 7.5GB utilization
- Fractional reserve = 0 results in 2.5GB utilization

The first scenario includes 2.5GB of space consumption for current data and 5GB of space to account for 100% turnover of the source in anticipation of snapshot use. The second scenario reserves no extra space.

Although this situation might seem confusing, it is unlikely to be encountered in practice. Compression implies thin provisioning, and thin provisioning in a LUN environment requires fractional reservations. It is always possible for compressed data to be overwritten by something uncompressible, which means a volume must be thin provisioned for compression to result in any savings.

NetApp recommends the following reserve configurations:

- Set `fractional-reserve` to 0 when basic capacity monitoring is in place along with `volume-autogrow` and `snapshot-autodelete`.
- Set `fractional-reserve` to 100 if there is no monitoring ability or if it is impossible to exhaust space under any circumstance.

Efficiency

Space efficiency features, such as compression, compaction, and deduplication are designed to increase the amount of logical data that fits on a given amount of physical storage. The result is lower costs and management overhead.

At a high level, compression is a mathematical process whereby patterns in data are detected and encoded in a way that reduces space requirements. In contrast, deduplication detects actual repeated blocks of data and removes the extraneous copies. Compaction allows multiple logical blocks of data to share the same physical block on media.

See the sections below on thin provisioning for an explanation of the interaction between storage efficiency and fractional reservation.

SQL Server also have feature to compress and efficiently manage data. SQL Server currently supports two types of data compression: row compression and page compression.

Row compression changes the data storage format. For example, it changes integers and decimals to the variable-length format instead of their native fixed-length format. It also changes fixed-length character strings to the variable-length format by eliminating blank spaces. Page compression implements row compression and two other compression strategies (prefix compression and dictionary compression). You can find more details about page compression in Page Compression Implementation.

Data compression is currently supported in the Enterprise, Developer, and Evaluation editions of SQL Server 2008 and later. Although compression can be performed by the database itself, this is rarely observed in a SQL Server environment.

Here are the recommendation for managing space for SQL Server data files

- Use thin provisioning in SQL Server environments to improve space utilization and to reduce the overall storage requirements when the space guarantee functionality is used.
- Use autogrow for most common deployment configurations because the storage admin only needs to monitor space usage in the aggregate.
- Advice not to enable deduplication on any volumes containing SQL Server data files unless the volume is known to contain multiple copies of the same data, such as restoring database from backups to a single volume.

Space reclamation

Space reclamation can be initiated periodically to recover unused space in a LUN. With SnapCenter, you can use the following PowerShell command to start space reclamation.

```
Invoke-SdHostVolumeSpaceReclaim -Path drive_path
```

If you need to run space reclamation, this process should be run during periods of low activity because it initially consumes cycles on the host.

Copyright information

Trademark information