

Configure SQL Server resources for optimal performance

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/configure-sql-server-resources-optimal-performance/1-introduction>

Introduction

Completed

One of the challenges administrators face in the cloud is balancing costs and performance. Both Azure and SQL Server provide many options for configuration to meet the needs of small and large workloads. Choosing the right storage and sizing your virtual machine are critical steps in meeting the performance needs of your applications and balancing cloud costs. Proper configuration of SQL Server resources like TempDB which can easily become a performance bottleneck, and Resource Governor, which can be used to manage multitenant workloads, are also important for properly maintaining your server performance.

Learning objectives

In this module, you will:

- Understand your options for configuration of Azure storage
- Learn how to configure TempDB data files in SQL Server
- Learn how to choose the right type of VM for SQL Server workloads
- Understand the use cases and configuration of Resource Governor in SQL Server

2. Explain how to optimize Azure storage for SQL Server virtual machines

Explain how to optimize Azure storage for SQL Server virtual machines

Completed

Storage performance is a critical component of an I/O heavy application like a database engine. Azure offers a broad array of storage options and can even build your storage solution to meet your workload requirements.

Azure Storage is a robust and secure platform designed to meet the diverse needs of various applications. It offers a wide range of scalable solutions, ensuring that all types of storage support encryption at rest. Users can choose between a Microsoft-managed encryption key or a user-defined encryption key for added security.

- **Blob Storage** - Blob storage is what is known as object-based storage and includes cold, hot, and archive storage tiers. In a SQL Server environment, blob storage will typically be used for database backups, using SQL Server's back up to URL functionality.
- **File Storage** - File storage is effectively a file share that can be mounted inside a virtual machine, without the need to set up any hardware. SQL Server can use File storage as a storage target for a failover cluster instance.
- **Disk Storage** - Azure managed disks offer block storage that is presented to a virtual machine. These disks are managed just like a physical disk in an on-premises server, except that they're virtualized. There are several performance tiers within managed disks depending on your workload. This type of storage is the most commonly used type for SQL Server data and transaction log files.

Azure managed disks

Azure managed disks are block-level storage volumes that are presented to Azure Virtual Machines. Block level storage refers to raw volumes of storage that are created and can be treated as an individual hard drive. These block devices can be managed within the operating system, and the storage tier isn't aware of the contents of the disk. The alternative to block storage is object storage, where files and their metadata are stored on the underlying storage system. Azure Blob Storage is an example of an object storage model. While object storage works well for many modern development solutions, most workloads running in virtual machines use block storage.

The configuration of your managed disks is important to the performance of your SQL Server workloads. If you're moving from an on-premises environment, it's important to capture metrics like

average disk seconds/read and **average disk seconds/write** from Performance Monitor as detailed earlier. Another metric to capture is the I/O Operations per Second, which can be captured using the **SQL Server: Resource Pool Stats Disk Read and Write IO/sec** counters, which show you how many IOPs SQL Server is serving at its peak. It's important to understand your workloads. You want to design your storage and virtual machine to meet the needs of those workload peaks without incurring significant latency. Each Azure Virtual Machine type has a limit on IOPs.

Azure managed disks come in four types:

Ultra disk - Ultra disks support high-IO workloads for mission critical databases with low latency.

Premium SSD - Premium SSD disks are high-throughput and low latency and can meet the needs of most database workloads running in the cloud.

Standard SSD - Standard SSDs are designed for lightly used dev/test workloads or web servers that do a small amount of IO, and require predictable latency.

Standard HDD - Standard HDDs are suitable for backups and file storage that is infrequently accessed.

Typically, production SQL Server workloads use either Ultra disk or Premium SSD, or some combination of the two. Ultra disks are typically used where you're looking for submillisecond latency in response time. Premium SSDs typically have single digit millisecond response time, but have lower costs, and more flexibility in design. Premium SSDs also support read-caching, which can benefit read-heavy database workloads by reducing the number of trips to the disk. The read cache is stored on the local SSD (the D:\ drive on Windows or /dev/sdb1/ on Linux) which can help reduce the number of round trips to the actual disk.

Striping disks for maximum throughput

One of the ways to get more performance and volume out of Azure disks is to stripe your data across multiple disks. This technique doesn't apply to Ultra disk, as you can scale IOPs, throughput, and maximum size independently on a single disk. However, with Premium SSDs it can be beneficial to scale both IOPs and storage volume. In order to stripe disks in Windows, you add the number of disks you would like to the VM, and then create a pool using Storage Spaces in Windows. Don't configure any redundancy for your pool (which would limit your performance) as the redundancy is provided by the Azure framework, which keeps three copies of all disks in synchronous replication to protect against a disk failure. When you create a pool, your pool has the sum of the IOPs and the sum of the volume of all the disks in your pool. For example, if you used 10 P30 disks that are each 1 TB and have 5000 IOPs per disk, you would have a 10-TB volume with 50,000 IOPs available.

SQL Server storage configuration best practices

There are few recommendations for best practices for SQL Server on Azure VMs and their storage configuration:

- Create a separate volume for data and transaction log files
- Enable read caching on the data file volume
- Don't enable any caching on the log file volume
- Plan for an extra 20% of IOPs and throughput when building your storage for your VM to handle workload peaks
- Use the D: drive (the locally attached SSD) for TempDB files because TempDB is recreated upon server restart, so there's no risk of data loss
- Enable instant file initialization to reduce the impact of file-growth activities.
- Move trace file and error log directories to data disks
- For workloads requiring storage latency under 1 millisecond, consider using Ultra disk over Premium SSD.

Azure Virtual Machine resource provider

One way to reduce the complexity of building storage for your SQL Server on an Azure Virtual Machine is to use the SQL Server templates in Azure Marketplace, which allow you to configure your storage as part of your deployment. You can configure the IOPs as needed and the template performs the work of creating your storage spaces pools within Windows.

Configure storage



Storage optimization ⓘ

General

Transactional processing

Data warehousing

Data storage

These disks will be attached to your virtual machine as data disks and will be stored in storage as page blobs.

Data drive location * ⓘ

F:\data



Disk type * ⓘ

Premium SSD



Disk type	Size (GiB)	Max IOPS	Max throughput	Number of disks
1024 GiB, Premium SSD (...)	1024	5000	200	1

ⓘ 1024 GiB, 5000 IOPS, 200 MB/s

Log storage

Transaction logs are a critical component of the database as they record all transactions and database modifications made by each transaction.

Shared drive space * ⓘ

Use a separate drive for lo...

Log drive location * ⓘ

G:\log



Disk type * ⓘ

Premium SSD



Disk type	Size (GiB)	Max IOPS	Max throughput	Number of disks
1024 GiB, Premium SSD (...)	1024	5000	200	1

ⓘ 1024 GiB, 5000 IOPS, 200 MB/s

TempDb storage

The tempDb system database is a global resource that is available to all users connected to the instance of SQL Server. It is used to store temporary user objects and internal objects created by the database engine.

Shared drive space * ⓘ

Use local SSD drive



TempDb drive location * ⓘ

D:\tempDb

This resource provider also supports adding TempDB to the local SSD drive and creates a scheduled task to create the folder on startup.

3. Describe virtual machine resizing

<https://learn.microsoft.com/en-us/training/modules/configure-sql-server-resources-optimal-performance/3-describe-virtual-machine-resizing>

Describe virtual machine resizing

Completed

There are many size options for Azure Virtual Machines. For SQL Server workloads the main characteristics to look for are the amount of memory available, and the number of input and output operations (IOPs) the virtual machine can perform.

Using constrained cores

Typically, SQL Server licenses are based on the number of cores, and Azure allocates a fixed ratio of CPU cores to memory. However, some workloads may require large amounts of memory without needing the default number of allocated CPUs. In such cases, using Azure's constrained cores can be beneficial.

With constrained cores, you can reduce the cost of software licensing while still getting the full amount of memory, storage, and I/O bandwidth. This is good for database workloads that aren't CPU-intensive and can benefit from high memory, storage, and I/O bandwidth, while using a constrained vCPU count.

Using general purpose virtual machines

Most SQL Server production workloads run on the general purpose or memory-optimized families of Azure Virtual Machines. Larger workloads requiring more memory and/or CPU resources land in memory-optimized virtual machines, but many production applications can run comfortably on general purpose virtual machines.

Resizing virtual machines

Azure supports resizing your virtual machine. This operation does require a restart; however, restarting a virtual machine is typically a fast process. In some cases, depending on what virtual machine type you're switching to and from, you may need to deallocate your virtual machine and then resize. This operation does extend the duration of the outage but shouldn't take more than a few minutes.

4. Optimize database storage

<https://learn.microsoft.com/en-us/training/modules/configure-sql-server-resources-optimal-performance/4-optimize-database-storage>

Optimize database storage

Completed

To optimize database storage, you should consider proportional fill and tempdb configuration.

Understand I/O performance

I/O performance can be critical to a database application. Azure SQL abstracts you from physical file placement, but there are methods to ensure you get the I/O performance you need.

Input/output per second (IOPS) might be important to your application. Be sure you've chosen the right service tier and vCores for your IOPS needs. Understand how to measure IOPS for your queries on-premises if you're migrating to Azure. If you have restrictions on IOPS, you might see long I/O waits. In the vCore purchasing model, you can scale up vCores or move to Business Critical or Hyperscale if you don't have enough IOPS. For production workloads, when using DTU, we recommend moving to the Premium tier.

I/O latency is another key component for I/O performance. For faster I/O latency for Azure SQL Database, consider Business Critical or Hyperscale. For faster I/O latency for SQL Managed Instance, move to Business Critical or increase the file size or the number of files for the database. Improving transaction log latency might require you to use multistatement transactions.

Files and filegroups

SQL Server professionals often use files and filegroups to improve I/O performance through physical file placement. Azure SQL doesn't allow users to place files on specific disk systems. However, Azure SQL has resource commitments for I/O performance regarding rates, IOPS, and latencies. In this way, abstracting the user from physical file placement can be a benefit.

Azure SQL Database only has one database file (Hyperscale typically has several), and the maximum size is configured through Azure interfaces. There's no functionality to create more files.

Azure SQL Managed Instance supports adding database files and configuring sizes, but not physical placement of files. You can use the number of files and file sizes for SQL Managed Instance to improve I/O performance. In addition, user-defined filegroups are supported for SQL Managed Instance for manageability purposes.

Describe proportional fill

When inserting 1 gigabyte of data into a SQL Server database with two data files, you might expect each file to increase by approximately 512 megabytes. However, this isn't always the case. SQL Server distributes data based on the size of each file. For instance, if both data files are 2 gigabytes, the data would be evenly distributed. But if one file is 10 gigabytes and the other is 1 gigabyte, around 900 MB would go into the larger file and 100 MB into the smaller one. This behavior is common in any database, but in the write-intensive tempdb, an uneven write pattern can create a bottleneck in the largest file, as it handles more writes.

Configure Tempdb in SQL Server

SQL Server detects the number of available CPUs during setup and configures the appropriate number of files, up to eight, with even sizing. Additionally, the behaviors of trace flags 1117 and 1118 are integrated into the database engine, but only for `tempdb`. For `tempdb`-heavy workloads, it may be beneficial to increase the number of `tempdb` files beyond eight, matching the number of CPUs on your machine.

You use `tempdb` in the same way for both SQL Server and Azure SQL. Note, however, that your ability to configure `tempdb` is different, including the placement of files, the number and size of files, and `tempdb` configuration options.

SQL Server uses `tempdb` for various tasks beyond just storing user-defined temporary tables. It's used for work tables that store intermediate query results, sorting operations, and the version store for row versioning, among other purposes. Due to this extensive utilization, it's crucial to place `tempdb` on the lowest latency storage available and to properly configure its data files.

The database files of `tempdb` are always automatically stored on local SSD drives, so I/O performance shouldn't be an issue.

SQL Server professionals often use more than one database file to partition allocations for `tempdb` tables. For Azure SQL Database, the number of files is scaled with the number of vCores (for example, two vCores equals four files) with a maximum of 16. The number of files isn't configurable through T-SQL against `tempdb`, but you can configure it by changing the deployment option. The maximum size of `tempdb` is scaled per number of vCores. You get 12 files with SQL Managed Instance, independent of vCores.

The database option `MIXED_PAGE_ALLOCATION` is set to `OFF`, and `AUTOGROW_ALL_FILES` is set to `ON`. You can't configure this, but, as with SQL Server, these are the recommended defaults.

The `tempdb` metadata optimization feature introduced in SQL Server 2019, which can alleviate heavy latch contention, isn't currently available in Azure SQL Database or Azure SQL Managed Instance.

Database configuration

Commonly, you configure a database with the T-SQL `ALTER DATABASE` and `ALTER DATABASE SCOPED CONFIGURATION` statements. Many of the configuration options for performance are available for Azure SQL. Consult the [ALTER DATABASE](#) and [ALTER DATABASE SCOPED CONFIGURATION](#) T-SQL reference for the differences between SQL Server, Azure SQL Database, and Azure SQL Managed Instance.

In Azure SQL Database, the default recovery model is full recovery, which ensures that your database can meet Azure service-level agreements (SLAs). This means that minimal logging for bulk operations isn't supported, except for `tempdb`, where minimal logging is allowed.

MAXDOP configuration

Max degree of parallelism (MAXDOP) can affect the performance of individual queries. SQL Server and Azure SQL handle `MAXDOP` in the same way. When `MAXDOP` is set to a higher value, more

parallel threads are used per query, potentially speeding up query execution. However, this increased parallelism requires extra memory resources, which can lead to memory pressure and affect storage performance. For example, when compressing rowgroups into a columnstore, parallelism requires more memory, which can result in memory pressure and rowgroup trimming.

Conversely, setting MAXDOP to a lower value can reduce memory pressure, allowing the storage system to perform more efficiently. This is important in environments with limited memory resources or high storage demands. By carefully configuring MAXDOP, you can balance query performance and storage efficiency, ensuring optimal use of both CPU and storage resources.

You can configure MAXDOP in Azure SQL, similar to SQL Server, by using the following techniques:

- `ALTER DATABASE SCOPED CONFIGURATION` to configure MAXDOP is supported for Azure SQL.
- The stored procedure `sp_configure` for "max degree of parallelism" is supported for SQL Managed Instance.
- MAXDOP query hints are fully supported.
- Configuring MAXDOP with Resource Governor is supported for SQL Managed Instance.

5. Control SQL Server resources

<https://learn.microsoft.com/en-us/training/modules/configure-sql-server-resources-optimal-performance/5-control>

Control SQL Server resources

Completed

While some SQL Servers or Azure SQL managed instances are dedicated to a single application's databases, a configuration often seen in mission-critical applications, many servers support databases for multiple applications with varying performance requirements and peak workload cycles. Balancing these differing requirements can be challenging for administrators. One effective way to manage server resources is by using Resource Governor, introduced in SQL Server 2008.

Resource Governor is a feature in SQL Server and Azure SQL managed instances that allow granular control over CPU, physical I/O, and memory resources for incoming application requests. When enabled at the instance level, Resource Governor uses a classifier function to define how connections are treated, subdividing sessions into workload groups. Each workload group is configured to use a specific pool of system resources.

Resource pools

A resource pool represents the physical resources available on the server. SQL Server always has two pools: default and internal, even when Resource Governor isn't enabled. The internal pool is reserved for critical SQL Server functions and can't be restricted. The default pool, along with any resource pools you explicitly define, can be configured with limits on the resources they can use. For each noninternal pool, you can specify the following limits:

- Min/Max CPU percent
- Cap of CPU percent
- Min/Max memory percent
- NUMA node affinity
- Min/Max IOPs per volume

Note

Changes to a resource pool only impact new sessions, not those already in progress. Therefore, modifying a pool won't restrict the resources of a long-running process. The exception to this rule is external pools used with SQL Server Machine Learning Services, which can be limited by a pool change even for ongoing sessions.

All resource pool settings, except for the minimum and maximum CPU percentage, represent hard limits that can't be exceeded. The min/max CPU percentage only applies when there's CPU contention. For instance, if you set a maximum of 70%, the workload may use up to 100% of available CPU cycles when there's no contention. However, if other workloads are running, the workload will be restricted to 70%.

Workload group

A workload group serves as a container for session requests, classified by the classifier function. Similar to resource pools, there are two built-in groups: default and internal. Each workload group is associated with a single resource pool, but a resource pool can host multiple workload groups. By default, all connections are directed to the default workload group unless the classifier function assigns them to a user-defined group. The default workload group utilizes the resources allocated to the default resource pool.

Classifier function

The classifier function is run at the time a connection is established to the SQL Server instance and classifies each connection into a given workload group. If the function returns a NULL, default, or the name of the nonexistent workload group the session is transferred into the default workload group. Since the classifier is run at every connection, it should be tested for efficiency. The following image shows a sample classifier function that classifies users based on their user name.

```
CREATE FUNCTION dbo.RGClassifier()  
RETURNS SYSNAME
```

```
WITH SCHEMABINDING
AS
BEGIN
DECLARE @WorkloadGroup AS SYSNAME
IF(SUSER_NAME() = 'ReportUser')
    SET @WorkloadGroup = 'ReportServerGroup'

ELSE IF (SUSER_NAME() = 'PrimaryUser')
    SET @WorkloadGroup = 'PrimaryServerGroup'
ELSE
    SET @WorkloadGroup = 'default'

RETURN @WorkloadGroup
END
```

You can increase the complexity of the function definition shown in the example, but you should verify that the more complex function doesn't impact the user performance.

Resource Governor use cases

Resource Governor is used primarily in multitenant scenarios where a group of databases share a single SQL Server instance, and performance needs to be kept consistent for all users of the server. You can also use Resource Governor to limit the resources used by maintenance operations like consistency checks and index rebuilds, to try to guarantee sufficient resources for user queries during your maintenance windows.

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/configure-sql-server-resources-optimal-performance/6-knowledge-check>

Module assessment

Completed

7. Summary

Summary

Completed

Proper storage configuration is crucial for the performance of your Azure Virtual Machines. To ensure optimal performance, SQL Server should run on premium disk storage or ultra disk. The Resource Provider for SQL Server can automate the creation of storage for your SQL Servers on Azure Virtual Machines, simplifying the setup process. Additionally, Resource Governor can be used to manage to differ workloads within the same SQL Server, allowing you to allocate resources efficiently and maintain balanced performance across various applications. This combination of premium storage and resource management tools ensures that your SQL Server operates smoothly and efficiently in an Azure environment.