SQL Sentry Portal & SentryOne Monitor Dashboards

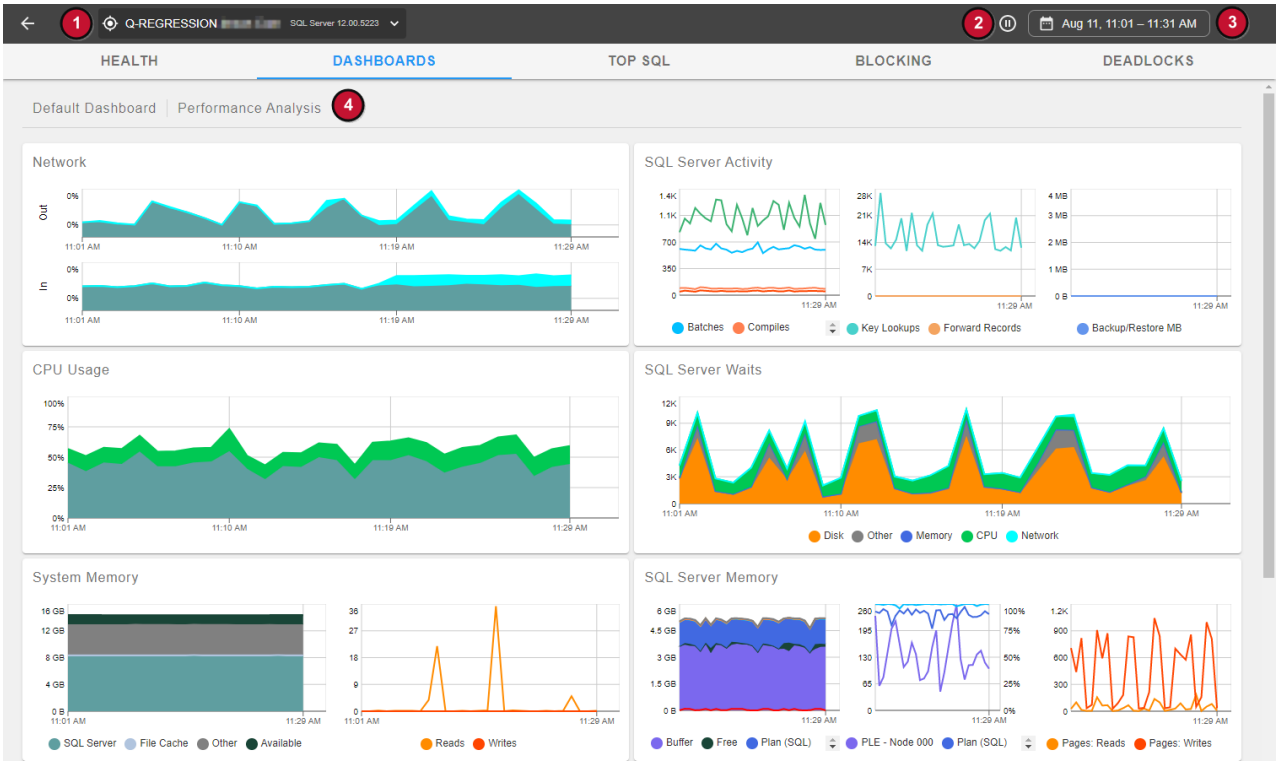
Last Modified on 14 March 2022

Applies to the following products and features: The SentryOne Monitor product.

The **Dashboards** view displays the performance charts for monitored targets. Select a target from the sidebar, then select **Dashboards** from the feature navigation menu at the top.

Dashboard Navigation

Note: In the SQL Sentry Portal feature for SQL Sentry, versions 2021.1 and later, Custom Dashboards have been replaced by Custom Charts and are not part of the Performance Analysis Dashboard section.



Within the **Dashboard** view, the options and information displayed include:

1. A drop-down menu to switch between monitored targets.
2. A pause button (which flips to a play button) to toggle the view of live data.
3. A date selector.
4. **Default Dashboard | Performance Analysis** for the default dashboard or **Custom Dashboard |** for a Custom Dashboard.

Dashboard Options

Date Selector

Use the date selector to use an available **Range** or define a **Custom Range** of time for data to view on the dashboard. Use the **Days**, **Weeks**, and **Months** options to select an entire day, week, or month range at once.

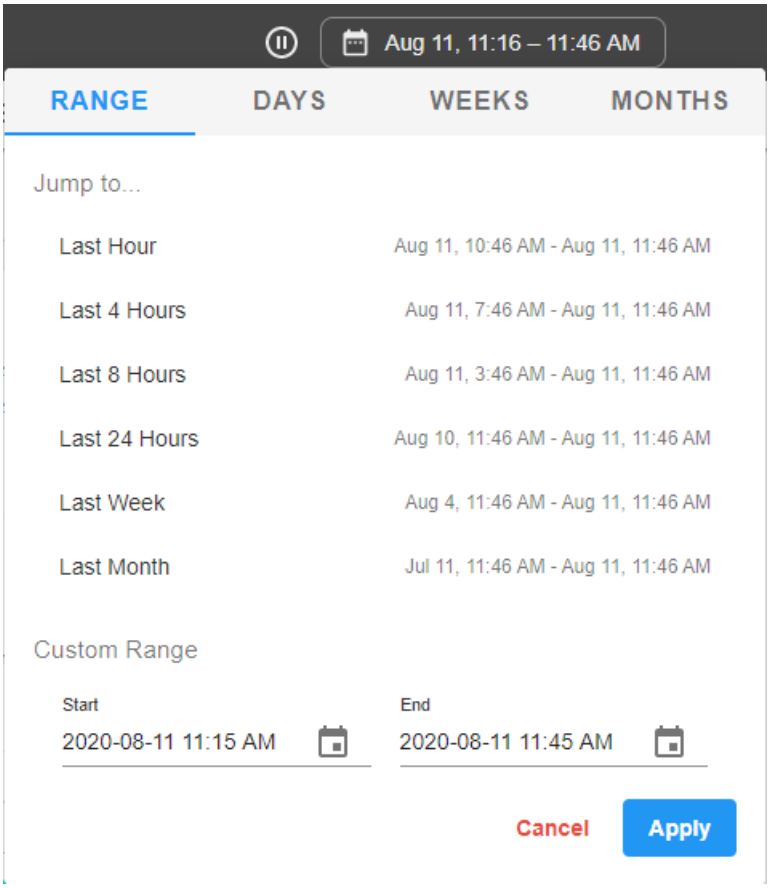
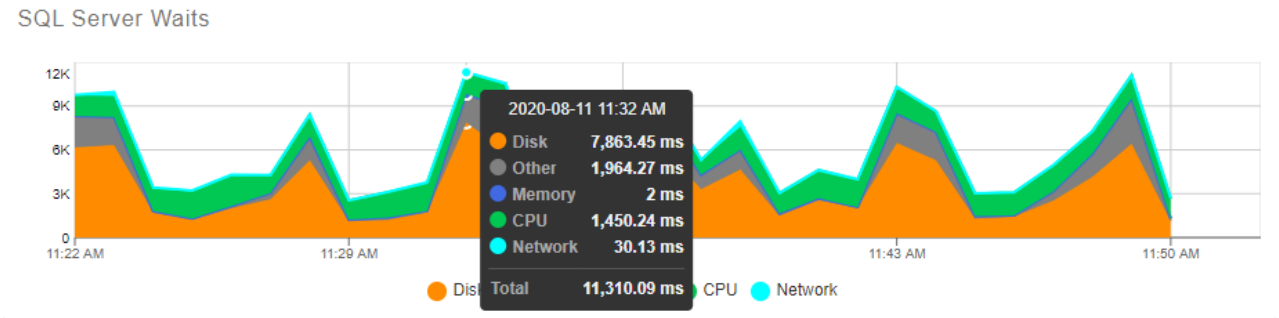


Chart Details

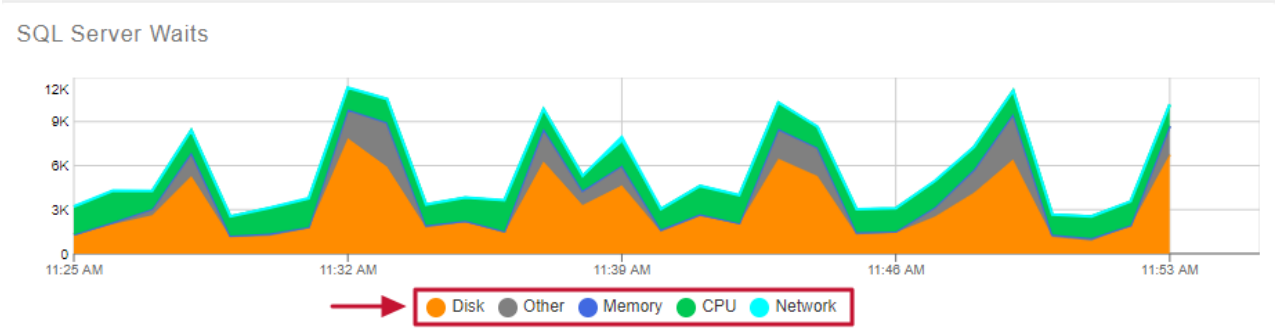
Hover over a point in a chart to view additional details.



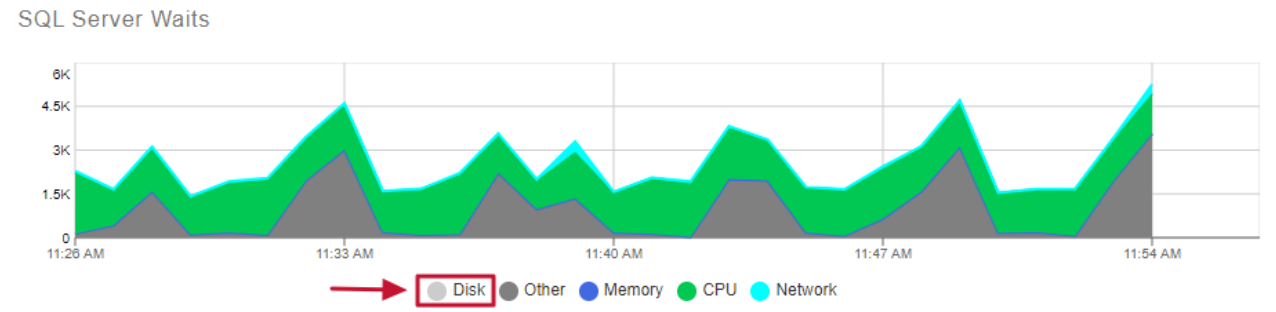
> **Note:** The chart details tool-tips vary by chart.

Filter Dots

Use the filter dots (e.g. **Disk**, **Other**, **Memory**, **CPU**, and **Network** in the image below) to customize what appears on the chart from the available options:

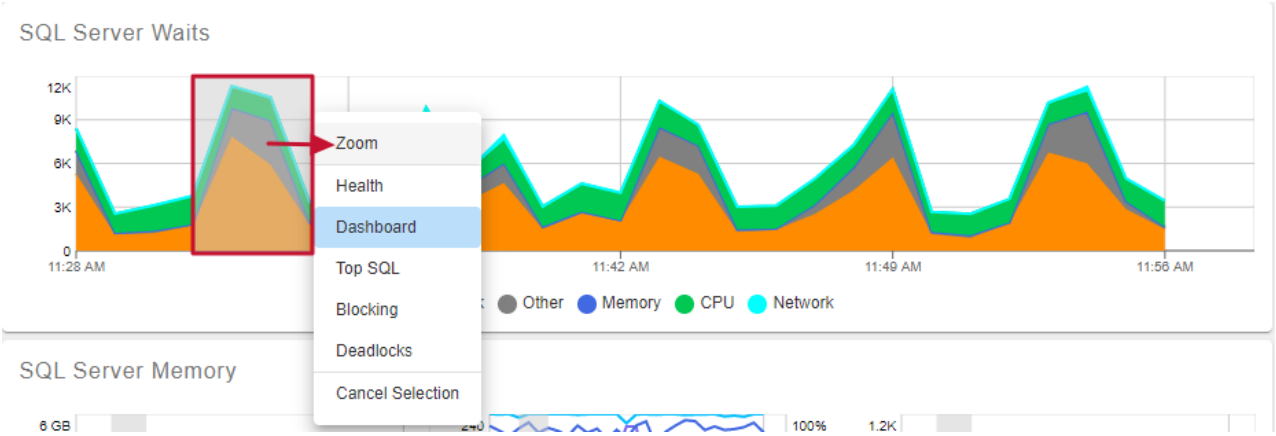


The same chart with **Disk** filtered out:



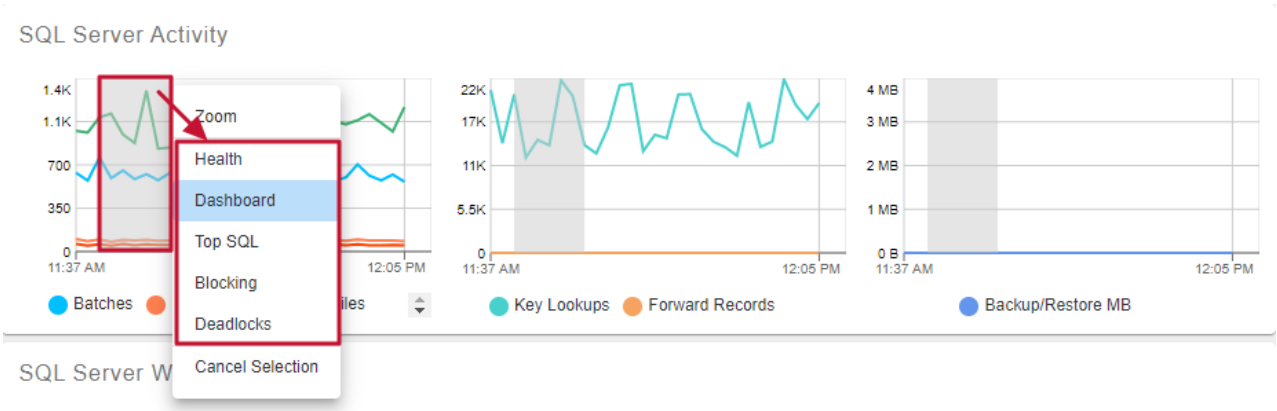
Zoom

Highlight an area on the chart for options to **Zoom** the Dashboard charts into that time selection.



Jump To

Highlight an area on the chart for options to jump to other feature views (e.g. **Health**, **Dashboard**, **Top SQL**, **Blocking**, or **Deadlocks**) for the selected time period. For example, use this troubleshoot performance issues by correlating a spike to a SQL query.



View Additional Metrics

Some charts, such as **SQL Server Activity** and **SQL Server Memory** may have additional metrics available in the legend and filter dots. Use the arrows to scroll through them.

Dashboard Charts

Note: The available charts displayed vary by target type (e.g. SQL Server vs. Azure SQL database). For example, an Azure SQL Managed Instance target will have SQL Server charts, but not Windows charts displayed.

Windows

Network

The **Network** chart displays the total network traffic on the server as well as the network utilization on each of the adapter present on the monitored target.

CPU Usage

The **CPU Usage** chart displays the total CPU Usage for the server as well as information on context switching, user time, kernel time, and more. The total processor time percentage across all processors on the server. A sustained value greater than 80 percent generally indicates a CPU bottleneck.

System Memory

The **System Memory** chart displays information about the amount of memory being used by different processes on the server as well as page faults and page file usage.

SQL Server

The amount of physical memory used by each SQL Server. Important for determining whether available memory is being used effectively, and whether there's memory contention between multiple instances on the same server.

File cache

The amount of physical memory currently allocated to the system file cache.

Other

The amount of physical memory used by all processes on the server other than SQL Server or SSAS.

Disk I/O

The **Disk IO** chart displays the read and write latency for each of the physical disks on the server.

ms/Read

The average time in milliseconds each physical disk read is taking.

Disk latency is the only disk measurement for which there are generally accepted ranges that represent good and bad performance from a SQL Server perspective. Disk queue metrics, for example, are not accurate for many SAN systems, and there are also no universally agreed upon good and bad ranges for SQL Server. The following ranges can be used as a general guideline to determine whether disk latency is acceptable:

- Less than 10ms - Fast *
- Between 10ms - 20ms - Acceptable
- Between 20ms - 50ms - Slow
- Greater than 50ms - Critical

* For transaction log writes, between 0ms and 2ms is desirable.

ms/Write

The average time in milliseconds each physical disk write is taking.

SQL Server

SQL Server Activity

The **SQL Server Activity** chart displays information about what the SQL Server instance is doing.

Batches

The total number of select, insert, or delete statements per second, including those inside a stored procedure. The name is somewhat misleading since it doesn't represent batches (groups of multiple statements) in the traditional sense. It's one of the best measures of overall activity on a SQL Server.

Over 1000 Mb per second is generally considered moderate to high activity. A 100Mb network can reach saturation at around 3000 Mb per second.

Compiles

The total number of initial compiles and recompiles per second. The value should generally be < 10 percent of

batches per second. Higher values indicate plan reuse is low, and will generally correlate with high CPU, since plan compilation is a CPU intensive operation. It may also correlate with low cache hit ratios for object and/or SQL plans.

It can also be a strong indicator of memory pressure, since there may not be enough room to keep all plans in cache.

Recompiles

The number of recompiles per second. The value should generally be < 10 percent of initial compiles per second.

Transactions

The total number of transactions per second across all databases on the server. A transaction can be either a user-defined statement block surrounded by a BEGIN TRAN and END TRAN, or an individual DML statement (insert, update or delete).

Compare with batches per second. On systems with high DML you typically want to see a low ratio of transactions to batches. A low ratio indicates that the individual statements are being bundled together, and can result in dramatically higher throughput and reduced IO due to log flushes.

Key Lookups

The number of times per second that the query processor had to perform a key lookup, across all queries. Lookups occur when the index being used is non-covering, meaning it doesn't include all of the columns required by the query. For each row returned by the index operation, the query processor has to go back to either the clustered index to perform a key lookup, or the base table to perform a RID lookup in the case of a heap.

Lookups are a high overhead operation, especially when large number of rows are involved, because each lookup incurs a random I/O and additional processing. This often correlates with higher CPU usage and page reads. Lookups can be eliminated by using a covering index, adjusting joins to reduce the set so the lookup isn't needed, or using multiple indexes (intersection).

Forwarded Records

The number of times per second that the query processor had to lookup forwarded records, across all queries. Forwarded records occur in tables with no clustered index (heaps) when rows become too large to fit on the page and have to be relocated. Over time, this can cause severe fragmentation and queries to incur much higher than normal I/Os, specifically random reads. This can correlate with high SQL Server page reads, and high SQL Server disk wait time, data file and physical disk latency if the disk system isn't keeping up with the additional reads. On many systems it's not unusual for this counter to stay at zero if all tables have a clustered index, any heaps aren't fragmented, or they just aren't accessed frequently.

Backup/Restore MB

The data rate in MB/sec for any backup operations taking place on the server.

SQL Server Waits

The **SQL Server Waits** chart displays information about the classes and categories of waits that occurred as

well as the duration of milliseconds that the waits were in effect during that time period.

Although there are hundreds of wait types, only the wait types that can be definitively attributed to one of the physical resource categories (**Disk**, **Memory**, **CPU**, and **Network**) are included in the calculations for this chart. The **Other** category is for a few other important wait types that can either affect performance in more than one major category, or cannot be directly attributed to any category with absolute certainty, such as backups and parallelism respectively.

SQL Server Waits is one of the most important charts on the dashboard, because it provides an instant profile of the SQL Server and where it's spending the most time waiting for physical resources. If SQL Server are consistently low, then what the other dashboard charts are showing is less important. For example, if CPU and SQL Server Activity: Batches look unusually high, but CPU waits are low, then the server hardware is effectively handling the load.

Total waits of less than 200ms is excellent. Between 200ms and 1000ms is average. Greater than 1000ms likely requires some attention to determine where the bottleneck lies. Over 5000ms may indicate severe bottle-necking.

The total wait time may be higher by virtue of a large number of processes (spids) active on the server, because wait time is summed across all processes, it isn't a per process average. This can be especially applicable to the **Other** category, because several processes experiencing parallelism at the same time can cause it to spike to high levels.

Additional Information:

- SQL Server Best Practices Article
- What to do (or not do) about top wait stats

SQL Server Memory

The **SQL Server Memory** chart displays information about how the Server instance is using memory that has been allocated to it.

Buffer

The current size of the buffer cache (in MB). You want this to be as large as possible for maximum performance, and on a dedicated SQL Server it should consume most of the SQL Server memory and physical memory.

Plan (SQL)

The current size of the cache used for query plans (in MB). This includes ad-hoc, auto-parameterized, and prepared plans. A high value in proportion to the buffer cache may indicate query plans aren't being effectively reused.

Plan (Objects)

The current size of the cache used for object plans (in MB). This includes stored procedures, functions, and triggers. A high value in proportion to the buffer cache may indicate query plans aren't being effectively reused.

Additional Information: Caching Mechanisms

Columnstore

The current size of the Columnstore index on the SQL Server (in MB). This includes both clustered and nonclustered columnstore indexes.

In-Mem OLTP

The current amount of memory (in MB) dedicated to In-Memory OLTP. This includes Memory-optimized tables, non-durable tables, and natively compiled T-SQL modules.

Other

The current size of the cache used for all other plans (in MB). This includes bound trees, extended stored procedures, temporary tables, and table variables. This cache size should be low in proportion to the other plan caches. If it goes over roughly 10 percent of the object or SQL plan size, further investigation may be needed.

PLE (sec)

The average lifespan of a data page. If this value is less than 600, it's an indicator of memory pressure. Ideally, it should be much higher than 600 if ample memory is available. In general, the larger the buffer cache size, the higher it should be. This is the best universal indicator of memory pressure.

Plan (SQL)

The ratio of hits to lookups for the query plan cache. This value should stay above 90 percent.

Plan (Object)

The ratio of hits to lookups for the object plan cache. This value should stay above 90 percent.

Pages: Reads

The average number of buffer data pages read from disk per second. Ideally, this value should be at or near zero most of the time. If it's above zero, it means that the data wasn't found in the buffer cache, and so it had to be retrieved from disk. If spikes in page reads correlate with high disk latency, the disk system may not be keeping up.

Querying newly created temp tables will also show up as page reads, as well as activity from internal tempdb objects. This includes hash joins, hash aggregates, sort, and query spool operations. This means that you can still see high paging from tempdb due to query activity, even though you aren't explicitly using temp tables.

When page reads and page writes correlate closely, it's a strong indicator that it's related to tempdb activity, because pages are being written to disk when the objects are created, then immediately read back in to memory for use by querying operations.

If lazy writes > zero and track closely with page reads, and page life expectancy < 600, it's a strong indicator of memory pressure, because data is being moved out of buffer to make room for new data coming in.

Lazy writes also cause page writes, but generally much less than tempdb activity. If you see high page reads, and relatively low lazy writes and page writes, it's likely memory pressure and not tempdb activity.

Pages: Writes

The average number of buffer data page writes to disk per second.

Page writes can be caused by checkpoints, lazy writes, and tempdb activity. To calculate the approximate amount of writes related to tempdb, for any given interval, subtract checkpoints and lazy writes from total page writes.

If high page writes correlate with high latency, the disk system may not be keeping up.

Database I/O

The **Database IO** chart displays information about the read and write latency for the databases.

ms/Read

The average time in milliseconds each physical disk read is taking for a particular database file. The top 10 database files (data and transaction log) with the highest latency for the specified date range are shown.

ms/Write

The average time in milliseconds each physical disk write is taking for a particular database file.

Log Flushes

Log flushes occur with every DML operation, and are a normal part of SQL Server activity. It's important to note that log writes to physical disk from updates to buffer pages happen immediately upon transaction commit, whereas writes to physical disk from the changed buffer pages is delayed until the next checkpoint occurs. It's critical that the physical disk system where the transaction log resides is fast enough to keep up with activity. If not, it can slow down all DML operations occurring in the database.

Ideally each busy transaction log should have its own dedicated disks, so that writes can happen sequentially, which will minimize latency. If log flushes are high and latency is high for a transaction log file, then the disk system is likely under-powered for the current load.

Checkpoint Pages

The average pages per second written to disk by the checkpoint process. Checkpoints flush all dirty buffer pages for a given database to disk and are a normal part of SQL Server operations. The frequency of checkpoints and volume of checkpoint pages is dictated directly by the Recovery Interval server option. SQL Server uses checkpoints to batch writes to disk, which is generally more efficient. However, if the volume of each checkpoint is too high and you see a correlation with high disk latency, it may indicate that the disk system isn't keeping up.

Lazy Writes

The average number of writes per second by the lazy writer. The lazy writer periodically scans the buffer and evicts pages that have low use counts in order to maintain a certain number of pages on the free list. Ideally, this value should be at or near zero most of the time. When there is no memory pressure, the lazy writer will

generally leave data pages in memory, even those with low use counts. However, when pressure exists, the lazy writer will continually be working to make room for new data coming into the buffer.

An indicator of memory pressure is ongoing lazy writes > zero with page reads/writes > zero and page life expectancy< 600.

Azure SQL Database

Resource Usage

A DTU represents the power of the database engine as a blended measure of CPU, memory, and read and write rates. This measure helps you assess the relative power of the SQL Database performance levels. Each service tier, which sets pricing and usage limits for an Azure SQL Database, expresses the amount of resource limits as a number of DTUs. The more DTUs an Azure SQL Database is allocated the more resources the database will have to service the workload.

> **Additional Information:** See the What is Azure SQL Database? article on Microsoft Docs for more information regarding DTUs, purchasing models, and service tiers.

Total DTU %

If your database is seeing high Total DTU percentage usage it may benefit from adjusting to the next highest service tier to improve performance. If you're consistently seeing very low total DTU percentage usage you may save some money by scaling down to the next lower service tier.

Data I/O

This metric is the average Data I/O percentage based on the limit of the service tier. This is one of the metrics that makes up DTU.

Log I/O

This metric is the average log I/O percentage based on the limit of the service tier. This is one of the metrics that makes up DTU.

CPU %

This metric is the average CPU percentage based on the limit of the service tier. This is one of the metrics that makes up DTU.

Memory Usage

Allocated Memory Usage

Each service tier has a maximum amount of memory allowed for the Azure SQL Database to use. This metric provides the percentage of the allowed memory being used for the database.

It will be very common for this metric to be high. If much of the data your applications need is in memory it means better performance because the database doesn't have to read from the physical disk to return the data.

Database Size

Each service tier has a maximum allowed size for the Azure SQL Database. This chart uses that tier to determine the total space available.

Used Space

The space used by the database in MB.

Free Space

Amount of space remaining (in MB) from the total space allowed for the tier.