

# Elastic Pool performance troubleshooting for customers

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:28 AM PST

---

## Contents

- [Resolve performance and capacity issues in Azure SQL Dat...](#)
- [Choosing the correct elastic pool size](#)
- [Monitoring an elastic pool and its databases](#)
- [Decide which databases should belong to the elastic pool](#)
- [Scaling guidance for mitigating resource contention](#)
- [Performance issues related to CPU and memory](#)
- [Error 10928: The request limit for the database has been re...](#)
- [Running out of storage space](#)
- [Resources](#)

This article is a copy of the Selfhelp - Common Solutions article which the customer will see when requesting help through the portal.

## Resolve performance and capacity issues in Azure SQL Database elastic pools

Azure SQL Database elastic pools are a cost-effective solution for managing many databases with varying resource usage. All databases in an elastic pool share a single SQL server instance and its resources (such as CPU, memory, worker threads, storage space, and tempdb) on the assumption that only a subset of databases in the pool will use these resources at any given time.

Within the pool, individual databases are given the flexibility to use resources within set parameters. Under an increasing workload, each database can request more resources from the pool to meet demand. If the sum of all requests exceeds the resource capacity of the pool, your applications may experience bad performance or resource-related error messages.

Scan and select one or more of the following sections for guidance to help resolve your performance and capacity issues in Azure SQL Database elastic pools.

---

## Choosing the correct elastic pool size

The best size for an elastic pool depends on the aggregated resources needed for all databases in the pool. There are two resource types to consider:



- Maximum compute resources which are utilized by all databases in the elastic pool; compute resources are indexed by either eDTUs or vCores depending on your choice of purchasing model.
- Maximum data storage which is utilized by all databases in the pool.

The following steps can help you estimate whether a pool is more cost-effective than single databases:

**(1) Estimate the eDTUs or vCores needed for the pool:**

- For the DTU-based purchasing model:  
 $\text{MAX}(\langle \text{total number of databases} \times \text{average DTU utilization per database} \rangle, \langle \text{number of concurrently peaking databases} \times \text{peak DTU utilization per database} \rangle)$
- For the vCore-based purchasing model:  
 $\text{MAX}(\langle \text{total number of databases} \times \text{average vCore utilization per database} \rangle, \langle \text{number of concurrently peaking databases} \times \text{peak vCore utilization per database} \rangle)$

**(2) Estimate the total storage space needed for the pool by adding the data size needed for all the databases in the pool.**



- For the DTU-based model, determine the eDTU pool size that provides this amount of storage by examining the [Resource limits for elastic pools using the DTU purchasing model](#) .
- For the vCore-based model, the maximum storage space increases with the number of vCores, as shown in [Resource limits for elastic pools using the vCore purchasing model](#) .

**(3) Decide on the minimum SLO for your elastic pool**

- For the DTU-based purchasing model, take the larger of the eDTU estimates from step 1 and step 2.
- For the vCore-based purchasing model, take the vCore estimate from step 1.\*\*

**(4) See the SQL Database pricing page on the portal and find the smallest pool size that's greater than the estimate from step 3.**

**(5) Compare the pool price from step 4 to the price of using the appropriate compute sizes for single databases.**

If the number of databases in a pool approaches the maximum supported, make sure to consider [Resource management in dense elastic pools](#) . Specifically consider the best practices described in the [Operational recommendations](#)  paragraph of the article.

## Monitoring an elastic pool and its databases


To avoid performance degradation due to resource contention, you should proactively monitor resource consumption and take timely action when increasing resource contention starts affecting workloads. Continuous monitoring is important because resource usage in a pool changes over time due to changes in user workload, changes in data volumes and distribution, changes in pool density, and changes in the Azure SQL Database service.



The easiest monitoring option is through the Azure portal when viewing the elastic pool resource:

- The "Overview" page for the elastic pool provides the resource utilization and data storage details at the pool level.
- The "Database Resource Utilization" page in the "Monitoring" section lists the performance summary for all databases in the pool, with the option to select and de-select individual databases. This will allow you to see what databases are contributing the highest workload and to simulate what impact removing a database would have for the overall pool capacity.

- The "Monitoring" section also provides options to view additional metrics, define alerts, or stream the metrics to external consumers like Log Analytics and event hubs.

In DTU elastic pools, the eDTU metric at the pool level is not a MAX or a SUM of individual database utilization. It is derived from the utilization of various pool level metrics. Pool level resource limits may be higher than individual database level limits, so it is possible that an individual database can reach a specific resource limit (CPU, data IO, log IO, etc.), even when the eDTU reporting for the pool indicates no limit been reached.

For a step-by-step example regarding monitoring and alerts, see [Monitor resource usage using the Azure portal](#) .



Also see [Monitor an elastic pool and its databases](#)  and [Resource management in dense elastic pools](#)  for additional information.

## Decide which databases should belong to the elastic pool

Elastic pools are well suited for a large number of databases that have periods of activity followed by periods of inactivity. The workload of each database ideally has a low average utilization with infrequent utilization spikes. In a typical scenario, each database belongs to a different customer or application, each with varying and unpredictable resource consumption.

Conversely, a database with persistent medium-to-high utilization shouldn't be placed in the elastic pool if its workload might interfere with the infrequent utilization spikes of the other pool databases.

Use the steps from the section "Monitoring an elastic pool and its databases" above to identify databases that are interfering with the recommended usage pattern. Then either move these databases to a different pool with a higher performance level, or isolate them as stand-alone databases outside of the pool. This might also allow you to scale the pool down to a lower performance level and save overall cost.

To move a database into or out of the elastic pool, you need to change the database's service objective. You can achieve this through the "Compute and storage" database settings on the Azure portal, the [Set-AzSqlDatabase](#)  PowerShell cmdlet, or the [ALTER DATABASE](#)  Transact-SQL command.

Transact-SQL example:

```
-- check the current database edition options
SELECT Edition = DATABASEPROPERTYEX('AdventureWorks', 'Edition'),
       ServiceObjective = DATABASEPROPERTYEX('AdventureWorks', 'ServiceObjective'),
       MaxSizeInGB = cast(DATABASEPROPERTYEX('AdventureWorks', 'MaxSizeInBytes') as bigint) / (1024*1024*1024)

-- configure the database as standalone database (examples: Premium P1, General Purpose with 4 vCores)
ALTER DATABASE [AdventureWorks] MODIFY (EDITION = 'Premium', SERVICE_OBJECTIVE = 'P1', MAXSIZE = 500 GB);
ALTER DATABASE [AdventureWorks] MODIFY (EDITION = 'GeneralPurpose', SERVICE_OBJECTIVE = 'GP_Gen5_4', MAXSIZE =

-- move the database to a different elastic pool
ALTER DATABASE [AdventureWorks] MODIFY ( SERVICE_OBJECTIVE = ELASTIC_POOL (name = poolname ), MAXSIZE = 50 GB)
```

## Scaling guidance for mitigating resource contention

If your elastic pool is experiencing high utilization and resource contention, then use the steps from the section "Monitoring an elastic pool and its databases" above to identify the bottleneck and the databases that are related to it. The goal is to avoid that a single database, or a small subset of databases, impacts the performance of all other databases in the elastic pool.


Take the following action:

- If there is a clear relation to specific databases, then either move these databases to a different pool with a higher performance level, or isolate them as stand-alone databases outside of the pool. See the preceding section "Decide which databases should belong" for additional guidance.
- If there is no clear cause, and the workload can't be related to one or just a few databases, the quickest option is to scale the elastic pool to a higher performance level. See further below in this section for additional guidance.

**Do not move "hot" databases though.** If resource contention is primarily caused by a small number of highly utilized databases, don't move them while the databases still remain highly utilized. The move operation will further degrade performance, both for the database being moved and for the entire pool. Instead, either wait until high utilization subsides, or move less utilized databases to relieve resource pressure at the pool level.

When scaling an elastic pool, a new compute instance is created with the requested service tier and compute size. For some combinations of service tier and compute size changes, a replica of each database must be created in the new compute instance, which involves copying data and can impact the overall latency. Similar to moving individual databases, you should wait until high utilization subsides, if possible, to avoid additional impact caused by the scaling operation.

Regardless, the databases remain online during either step, and connections continue to be directed to the databases in the original compute instance while either operation is progressing. There's no downtime except for a brief period at the end of the operation when database connections are dropped before being redirected to the new compute instance.

See [Scale elastic pool resources in Azure SQL Database](#)  for a more detailed discussion, the commands for initiating the operation, and additional information about expected scaling latency.


---

## Performance issues related to CPU and memory

If you experience high utilization and resource contention in an elastic pool, consider the following actions to mitigate it:

- Run through the usual query performance troubleshooting steps. Tune the queries to reduce overall resource consumption.
- Spread application resource consumption across multiple databases over time so that their peak utilization don't overlap.
- Regroup databases into separate pools so that peak workloads in different databases do not overlap in the same pool.
- Reduce pool density by moving some databases to another pool, or by making them stand-alone databases.
- Scale up the pool to get more resources. Also evaluate the eDTU against the vCore pricing model to decide if changing would bring you any benefit in performance and resource capacity.

Consider moving the problem databases out of the elastic pool and continue troubleshooting while they run as stand-alone database. This avoids that a single database, or a small subset of databases, impacts the performance of all other databases in the pool.

Use [Query Performance Insight](#)  on the stand-alone database to identify the bottleneck of the queries that are causing the issues.

See the preceding section "Scaling guidance for mitigating resource contention" above for further details.



---

## Error 10928: The request limit for the database has been reached



The full error message for error 10928 is:

Error 10928: Resource ID : 1: The request limit for the database has been reached

The request limit is referring to the number of worker threads. Worker threads are the SQL threads in the database that are actively processing queries. The maximum number of workers depends on your elastic pool's service tier.

**For an immediate solution**, scale your elastic pool to a larger service tier that is sufficient to handle the workload. See the values for "Max concurrent workers per pool" on [Resource limits for elastic pools using the DTU purchasing model](#)  and [Resource limits for elastic pools using the vCore purchasing model](#) .

**For a longer-term solution**, consider the following items:


- Identify and optimize long running or resource intensive queries to reduce the resource utilization of each query. Use [Query Performance Insight](#)  on the elastic pool's portal page to narrow down the queries that are causing the issues.
  - Reduce the max degree of parallelism of the database. Set it between a minimum of 1 and a maximum of 8. It is an online operation without a service restart.
  - Optimize or reduce the elastic pool's query workload to reduce the number of errors.
  - Review the article [Understand and resolve Azure SQL Database blocking problems](#)  to identify and reduce query blocking, if blocking is present.
- 

## Running out of storage space

### Immediate solution if you have exhausted the available storage for the elastic pool

The typical error message for this situation is "The elastic pool has reached its storage limit". If you see this error for one or several of your databases in the pool, then increase the storage size of the elastic pool:

- For DTU-based elastic pools, scale to a higher service tier.
- For vCore-based elastic pools, increase the maximum storage, and add more vCores if the upper storage limit for the vCores has been reached.

See [Change elastic pool storage size](#)  for more information on pool storage sizes. Also see the preceding sections "Decide which databases should belong to the elastic pool" and "Scaling guidance for mitigating resource contention" above for additional guidance.

## Troubleshooting storage space issues

There are two ways you may run out of storage space in an elastic pool: by reaching the maximum configured size of a database, or by exhausting the available storage space on the elastic pool itself. You should proactively monitor storage consumption and take timely action before it starts affecting workloads. Continuous monitoring is important because storage usage in a pool changes over time due to changes in user workload, changes in data volumes, and changes in the pool density.

The easiest monitoring option is through the Azure portal when viewing the elastic pool resource:

- The "Overview" page for the elastic pool provides the resource utilization and the data storage details at the pool level, including maximum storage size, allocated space, and used space.
- The "Database Resource Utilization" page in the "Monitoring" section lists the performance summary for all databases in the pool, with the option to select and de-select individual databases. Make sure to select the additional metrics for including the storage values in the result.
- The "Monitoring" section also provides options to view additional metrics, define alerts, or stream the metrics to external consumers like Log Analytics and event hubs.

Run the following Transact-SQL queries to compare the allocated vs. used vs. unused data storage sizes at the elastic pool and the database level. These provide the following information:

- Maximum data storage size of the elastic pool and a selected database. The maximum database size should be smaller than the maximum pool size.
- The allocated data space and how much of it is used and unused. If the pool size is reaching its limit and there is a lot of unused allocated space, you can try to shrink the related databases to return this unused space from the database to the pool.

```

-- connect to master database
-- elastic pool data space details for all elastic pools at the server
SELECT elastic_pool_name,
       'time' = max(end_time),
       'avg_storage_used_percent' = max(avg_storage_percent),
       'avg_storage_free_percent' = (100.0 - max(avg_storage_percent)),
       'elastic_pool_storage_limit_GB' = max(elastic_pool_storage_limit_mb) / 1024.0,
       'elastic_pool_data_storage_used_GB' = (max(avg_storage_percent) / 100.0) * (max(elastic_pool_storage_limit
       'elastic_pool_data_storage_free_GB' = (100.0 - max(avg_storage_percent)) * (max(elastic_pool_storage_limit
FROM sys.elastic_pool_resource_stats
WHERE end_time > dateadd(minute, -15, getdate())
GROUP BY elastic_pool_name;

-- connect to user database
-- database data storage allocated vs. used/unused at the file level
-- values for current database
SELECT 'database_name' = DB_NAME(),
       'snapshot_time' = getdate(),
       'max_data_space_GB' = cast(DATABASEPROPERTYEX(DB_NAME(), 'MaxSizeInBytes') as bigint) / (1024*1024*1024),
       'allocated_disk_space_GB' = SUM(size/128.0) / 1024.0,
       'allocated_used_data_space_GB' = SUM(CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0) / 1024.0,
       'allocated_unused_data_space_GB' = SUM(size/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0) /
FROM sys.database_files
GROUP BY type_desc
HAVING type_desc = 'ROWS';

-- connect to user database
-- elastic pool data storage allocated vs. used/unused
-- values include all databases in the pool to the which current database belongs
SELECT TOP 1 'database_name' = DB_NAME(),
SELECT TOP 1 'database_name' = DB_NAME(),
       snapshot_time,
       'max_data_space_GB' = max_data_space_kb / 1024.0 / 1024.0,
       'allocated_disk_space_GB' = allocated_disk_space_kb / 1024.0 / 1024.0,
       'allocated_used_data_space_GB' = used_data_space_kb / 1024.0 / 1024.0,
       'allocated_unused_data_space_GB' = (allocated_disk_space_kb - used_data_space_kb) / 1024.0 / 1024.0,
       avg_storage_percent,
       avg_allocated_storage_percent
FROM sys.dm_resource_governor_resource_pools_history_ex
WHERE name LIKE 'SloSharedPool%'
ORDER BY snapshot_time desc;

```

When running out of storage space, it is possible that the databases have allocated the data space but are not using it. Freeing up the unused data space could help with increasing the available pool storage. See the guidance on DBCC SHRINKDATABASE and DBCC SHRINKFILE at [Reclaim unused allocated space](#) for steps and considerations. You should also be aware of the potential negative performance impact of shrinking database files, as described in [Index maintenance after shrink](#).

For more information regarding allocated vs. used vs. unused data space, see [Understanding types of storage space for an elastic pool](#).

## Resources

- [Manage elastic pools in Azure SQL Database](#)
- [Scale elastic pool resources in Azure SQL Database](#)
- [Resource management in dense elastic pools](#)
- [Resource limits for elastic pools using the DTU purchasing model](#)

- [Resource limits for elastic pools using the vCore purchasing model](#) 

**How good have you found this content?**

