


Troubleshoot Out Of Memory errors

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:27 AM PST


Contents

- [Issue](#)
- [Investigation / Analysis](#)
 - [View out of memory events](#)
 - [View memory clerks](#)
 - [Investigate active queries](#)
 - [Use Query Store to investigate past query memory usage](#)
 - [Extended events](#)
 - [In-memory OLTP out of memory](#)
- [Mitigation](#)

Troubleshoot out of memory errors with Azure SQL Database

The content of this article is adapted from the public documentation at [Troubleshoot out of memory errors with Azure SQL Database](#) .

Issue

You may see error messages when the SQL database engine has failed to allocate sufficient memory to run the query. This can be caused by various reasons including the limits of selected service objective, aggregate workload memory demands, and memory demands by the query. For more information on the memory resource limit for Azure SQL Databases, see [Resource management in Azure SQL Database](#) .


Try the following avenues of investigation in response to:

- Error code 701 with error message "There is insufficient system memory in resource pool '%ls' to run this query."
- Error code 802 with error message "There is insufficient memory available in the buffer pool."

Investigation / Analysis

If out of memory errors persist in Azure SQL Database, consider at least temporarily increasing the service level objective of the database in the Azure portal. If out of memory errors persist, use the following queries to look for unusually high query memory grants that may contribute to an insufficient memory condition. Run the following example queries in the database that experienced the error (not in the `master` database of the Azure SQL logical server).

View out of memory events

If you encounter out of memory errors, review the output of [sys.dm_os_out_of_memory_events](#) . This DMV includes predicted out of memory cause information that is determined by a heuristic algorithm and is provided

with a finite degree of confidence. It provides some visibility to the events and causes of out of memory (OOM) events.

```
SELECT *
FROM sys.dm_os_out_of_memory_events
ORDER BY event_time DESC;
```

View memory clerks

Start with a broad investigation, if the out of memory error occurred recently, by viewing the allocation of memory to memory clerks. Memory clerks are internal to the database engine for this Azure SQL Database. The top memory clerks in terms of pages allocated might be informative to what type of query or feature of SQL Server is consuming the most memory.

```
SELECT [type], [name], pages_kb, virtual_memory_committed_kb
FROM sys.dm_os_memory_clerks
WHERE memory_node_id <> 64 -- ignore Dedicated Admin Connection (DAC) node
ORDER BY pages_kb DESC;
```

```
SELECT [type], [name], pages_kb, virtual_memory_committed_kb
FROM sys.dm_os_memory_clerks
WHERE memory_node_id <> 64 -- ignore Dedicated Admin Connection (DAC) node
ORDER BY virtual_memory_committed_kb DESC;
```

Notes regarding the output:

- Some common memory clerks, such as MEMORYCLERK_SQLQERESERVATIONS, are best resolved by identifying queries with large memory grants and improving their performance with better indexing and index tuning.
- While OBJECTSTORE_LOCK_MANAGER is unrelated to memory grants, it is expected to be high when queries claim many locks, for example, because of disabled lock escalation or very large transactions. See Wiki article [Objectstore Lock Manager](#) for further troubleshooting steps.
- Some clerks are expected to be the highest utilization: MEMORYCLERK_SQLBUFFERPOOL is almost always the top clerk, while CACHESTORE_COLUMNSTOREOBJECTPOOL will be high when columnstore indexes are used. Highest utilization by these clerks is expected.

For more information about memory clerk types, see [sys.dm_os_memory_clerks](#) .

Investigate active queries

In most cases, the query that failed is not the cause of this error.

The following sample query for Azure SQL Database returns important information on transactions that are currently holding or waiting for memory grants. Target the top queries identified for examination and performance tuning, and evaluate whether or not they are executing as intended. Consider the timing of memory-intensive reporting queries or maintenance operations.

```

--Active requests with memory grants
SELECT
  --Session data
  s.[session_id], s.open_transaction_count
  --Memory usage
  , r.granted_query_memory, mg.grant_time
  , mg.requested_memory_kb, mg.granted_memory_kb, mg.required_memory_kb, mg.used_memory_kb, mg.max_used_memo
  --Query
  , query_text = t.text
  , input_buffer = ib.event_info, query_plan_xml = qp.query_plan
  , request_row_count = r.row_count, session_row_count = s.row_count
  --Session history and status
  , s.last_request_start_time, s.last_request_end_time, s.reads, s.writes, s.logical_reads
  , session_status = s.[status], request_status = r.status
  --Session connection information
  , s.host_name, s.program_name, s.login_name, s.client_interface_name, s.is_user_process
FROM
  sys.dm_exec_sessions s
  LEFT OUTER JOIN sys.dm_exec_requests AS r ON r.[session_id] = s.[session_id]
  LEFT OUTER JOIN sys.dm_exec_query_memory_grants AS mg ON mg.[session_id] = s.[session_id]
  OUTER APPLY sys.dm_exec_sql_text (r.[sql_handle]) AS t
  OUTER APPLY sys.dm_exec_input_buffer(s.[session_id], NULL) AS ib
  OUTER APPLY sys.dm_exec_query_plan (r.[plan_handle]) AS qp
WHERE mg.granted_memory_kb > 0
ORDER BY mg.granted_memory_kb desc, mg.requested_memory_kb desc;

```

You may decide to use the KILL statement to stop a currently executing query that is holding or waiting for a large memory grant. Use this statement carefully, especially when critical processes are running. For more information, see [KILL \(Transact-SQL\)](#).

Use Query Store to investigate past query memory usage

While the previous sample query reports only live query results, the following query uses the Query Store to return information on past query execution. This can be helpful in investigating an out of memory error that occurred in the past.


The following sample query for Azure SQL Database return important information on query executions recorded by the Query Store. Target the top queries identified for examination and performance tuning, and evaluate whether or not they are executing as intended. Note the time filter on `qsp.last_execution_time` to restrict results to recent history. You can adjust the TOP clause to produce more or fewer results depending on your environment.

```

SELECT TOP 10 PERCENT --limit results
    a.plan_id, query_id, plan_group_id, query_sql_text
    , query_plan = TRY_CAST(query_plan as XML)
    , avg_query_max_used_memory
    , min_query_max_used_memory
    , max_query_max_used_memory
    , last_query_max_used_memory
    , last_execution_time
    , query_count_executions
FROM (
    SELECT
        qsp.plan_id, qsp.query_id, qsp.plan_group_id, qsp.query_plan, qsqt.query_sql_text
    , last_execution_time = MAX(qsp.last_execution_time)
    , query_count_executions = SUM(qsrs.count_executions)
    , avg_query_max_used_memory = AVG(qsrs.avg_query_max_used_memory)
    , min_query_max_used_memory = MIN(qsrs.min_query_max_used_memory)
    , max_query_max_used_memory = MAX(qsrs.max_query_max_used_memory)
    , last_query_max_used_memory = MAX(qsrs_latest.last_query_max_used_memory) --only from latest result
FROM sys.query_store_plan AS qsp
INNER JOIN sys.query_store_query AS qsq ON qsp.query_id = qsq.query_id
INNER JOIN sys.query_store_query_text AS qsqt ON qsq.query_text_id = qsqt.query_text_id
INNER JOIN sys.query_store_runtime_stats AS qsrs ON qsp.plan_id = qsrs.plan_id
INNER JOIN (SELECT plan_id
    , last_query_max_used_memory
    , rownum = ROW_NUMBER() OVER (PARTITION BY plan_id ORDER BY last_execution_time DESC)
    FROM sys.query_store_runtime_stats qsrs) AS qsrs_latest
ON qsrs_latest.plan_id = qsp.plan_id
AND qsrs_latest.rownum = 1 --use latest last_query_max_used_memory per plan_id
WHERE DATEADD(hour, -24, sysdatetime()) < qsp.last_execution_time --past 24 hours only
AND qsrs_latest.last_query_max_used_memory > 0
GROUP BY qsp.plan_id, qsp.query_id, qsp.plan_group_id, qsp.query_plan, qsqt.query_sql_text
) AS a
ORDER BY max_query_max_used_memory DESC, avg_query_max_used_memory DESC;

```

Extended events

In addition to the previous information, it may be helpful to capture an [Extended Events](#)  trace of the activities on the server to thoroughly investigate an out of memory issue in Azure SQL Database. SSMS provides an Extended Events subfolder under each Azure SQL database in Object Explorer. Use an Extended Events session to capture these useful events, and identify the queries generating them:

- Category Errors:
 - error_reported
 - exchange_spill
 - hash_spill_details
- Category Execution:
 - excessive_non_grant_memory_used
- Category Memory:
 - query_memory_grant_blocking
 - query_memory_grant_usage

The capture of memory grant blocks, memory grant spills, or excessive memory grants could be potential clue to a query suddenly taking on more memory than it had in the past, and a potential explanation for an emergent out of memory error in an existing workload.

In-memory OLTP out of memory

You may encounter Error code 41805: There is insufficient memory in the resource pool '%ls' to run this operation if using In-Memory OLTP. Reduce the amount of data in memory-optimized tables and memory-optimized table-valued parameters, or scale up the database to a higher service objective to have more memory. For more information on out of memory issues with SQL Server In-Memory OLTP, see the Wiki article [Memory - In Memory](#).

Mitigation

The individual steps in the "Investigation / Analysis" section above contain some mitigation hints. See above.

How good have you found this content?

