Unable to release memory used for inmemory table

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

Contents

Issue

The customer is using in-memory tables and encounters the following error message:

> Msg 41823, Level 16, State 109, Line 1
> Could not perform the operation because the database has reached its quota for in-memory tables. This error may be transient. Please retry the operation. See 'http://go.microsoft.com/fwlink/?LinkID=623028' for more information.

They try to delete the data in the affected table, but are still unable to release the memory and cannot insert more rows.

Investigation / Analysis

This article describes an issue related to in-memory tables having a clustered columnstore index. See the steps below to confirm if the customer is affected by this issue.

If you find that the customer has a different issue, see Wiki article Memory - In Memory for more general troubleshooting steps.

Verify the recent memory usage for the database

In-memory object storage is represented by the `xtp_storage_percent` column, which indicates the storage utilization for In-Memory OLTP as a percentage of the pool limit at the end of the reporting interval. This includes memory used for storage of memory-optimized tables, indexes, and table variables, and the memory used for processing ALTER TABLE operations on memory-optimized tables.

```
SELECT
    end_time,
    xtp_storage_percent,
    avg_memory_usage_percent, avg_cpu_percent, avg_data_io_percent, avg_log_write_percent
FROM sys.dm_db_resource_stats

/*
end_time                 xtp_storage_percent  avg_memory_usage_percent  avg_cpu_percent  avg_data_io_percent
------------------------ -------------------- ------------------------- ---------------- --------------------
2022-10-19 15:02:19.860  61.69                31.40                     0.00             0.00
2022-10-19 15:02:04.817  61.69                31.40                     0.00             0.00
2022-10-19 15:01:49.770  61.69                31.40                     0.00             0.00
*/
```

Check if the index used space is the main memory consumer

The issue would present itself by high unused (= allocated but not used) index memory value, or by a high used memory value for an empty table.

```
-- memory consumption for all user tables, indexes, and system objects
SELECT
    object_name(object_id) AS name,
    memory_allocated_for_table_kb / 1024 AS memory_allocated_for_table_MB,
    memory_used_by_table_kb / 1024 AS memory_used_by_table_MB,
        (memory_allocated_for_table_kb - memory_used_by_table_kb) / 1024 AS memory_unused_by_table_MB,
    memory_allocated_for_indexes_kb / 1024 AS memory_allocated_for_indexes_MB,
    memory_used_by_indexes_kb / 1024 AS memory_used_by_indexes_MB,
        (memory_allocated_for_indexes_kb - memory_used_by_indexes_kb) / 1024 AS memory_unused_by_indexes_MB
FROM sys.dm_db_xtp_table_memory_stats
ORDER BY memory_used_by_table_MB DESC;

/* Sample output - note the huge allocated vs. used difference:
name    memory_allocated_for_table_MB  memory_used_by_table_MB  memory_unused_by_table_MB  memory_allocated_fo
------- ------------------------------ ------------------------ -------------------------  -------------------
table1  4358                           0                        4358                       1112
*/
```

Check if there is a clustered columnstore index enabled on the in-memory table

```
SELECT i.name AS indexname, t.name AS tablename, i.type_desc
FROM sys.indexes i
INNER JOIN sys.tables t ON i.object_id = t.object_id
WHERE i.type = 5 -- CLUSTERED COLUMNSTORE

/*
indexname   tablename  type_desc
----------- ---------- --------------------
table1_CCI  table1     CLUSTERED COLUMNSTORE
*/
```

Mitigation

A clustered columnstore index might be using the memory irrespective of whether there is data in the table or not. It will remain allocated to the table and index after deleting the data from the table, until the memory is

requested again either by the same table or another table. The clustered columnstore index is supposed to release the unused allocations, but that can apparently fail (exact conditions yet unknown though).

To mitigate such an issue, you can remove the clustered columnstore index and put it back on. This allows that the clustered columnstore index memory will be released after data deletion.

Use the following SQL statements to drop and recreate the clustered columnstore index on an in-memory table:

```
ALTER TABLE [dbo].[table1] DROP INDEX [table1_CCI];
ALTER TABLE [dbo].[table1] ADD INDEX [table1_CCI] CLUSTERED COLUMNSTORE;
```

Internal Reference

- Memory - In Memory

Public Doc Reference

- In-memory columnstore

How good have you found this content?