

Unplanned failover running query with spatial UDT

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

Contents

- [Issue](#)
- [Investigation / Analysis](#)
 - [ASC](#)
 - [Search IcM for recent occurrences](#)
 - [Kusto telemetry](#)
- [Mitigation](#)
- [More information](#)
- [Internal Doc References](#)
- [Public Doc Reference](#)

Issue

The customer is running a query using user-defined data types (UDT) based on spatial data types. Either always or only under specific conditions, the connection is dropped while the query executes and the database appears to be unavailable for a short time. The main symptom for the customer is dropped connections and database unavailability.

Investigation / Analysis

The issue is related to limitations with UDTs and spatial data types. If the query is creating large spatial aggregates and/or large result sets, it may run out of memory. This OOM situation is then causing an instance-level exception, which is triggering a SQL dump. The SQL instance is restarted and failing over after the dump has been written. It is this failover that is causing the disconnect and the database unavailability.

ASC

Check the ASC troubleshooter for unplanned failovers and SQL dumps. For this issue and cause, the failover will be related to out-of-memory (OOM) conditions and the `MEMORYCLERK_SPATIAL` memory clerk. Check the memory section on the troubleshooter to see which memory clerk is the main consumer.

Search IcM for recent occurrences

The failover might have triggered an LSI noting the dump and failover. Search the IcM portal for the server and database name; there might be some matches that can help you with assessing the frequency of the issue.

If you consider opening an IcM for the RCA, these LSI references will be helpful.

Kusto telemetry

According to [sys.dm_os_memory_clerks \(Transact-SQL\)](#), the memory clerk `MEMORYCLERK_SPATIAL` is used for [Spatial Data](#) memory allocations. See [sys.dm_os_memory_clerks \(Transact-SQL\)](#) for a list of memory clerks and their meaning.

The main indicator for this issue is the increase of `MEMORYCLERK_SPATIAL` while the offending query is running. The following Kusto query over `MonSqlMemoryClerkStats` returns the same values as the "sys.dm_os_memory_clerks" DMV and can help you find the memory type that is related to the out-of-memory issue:

```
// Identify the set of AppName and NodeName values that might exist after several failovers
let srv = "servername";
let db = "databasename";
let startTime = datetime(2022-10-19 07:00:00Z);
let endTime = datetime(2022-10-19 23:00:00Z);
let timeRange = ago(7d);
MonDmDbHadrReplicaStates
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
// | where TIMESTAMP >= timeRange
| where LogicalServerName =~ srv
| where logical_database_name =~ db
| where is_primary_replica == 1
| where AppName notcontains "b-"
| summarize min(TIMESTAMP) by AppName, NodeName
| join kind=inner
(
    MonSqlMemoryClerkStats
    | where TIMESTAMP >= startTime
    | where TIMESTAMP <= endTime
    // | where TIMESTAMP >= timeRange
    | where LogicalServerName =~ srv
    | extend memory_MB = round(pages_kb/1024.0)
    | extend vm_committed_MB = round(vm_committed_kb/1024.0)
    | where ['memory_clerk_type'] in ("MEMORYCLERK_SPATIAL")
    //| where ['memory_clerk_type'] in ("MEMORYCLERK_SQLCLR", "MEMORYCLERK_SQLBUFFERPOOL", "CACHESTORE_SQLCP",
    | project TIMESTAMP, NodeName, LogicalServerName, AppName, memory_clerk_type, memory_clerk_name, memory_MB
) on AppName, NodeName
| summarize memory_MB = sum(memory_MB) + sum(vm_committed_MB) by ['memory_clerk_type'], bin(TIMESTAMP, 5min)
| render timechart
```

Mitigation

This issue is caused by a known limitation. It is by design and there is nothing we can do at the SQL Server and platform level to avoid it. The restart occurs automatically to clear the memory, thus causing the unavailability. Any ICM can only confirm the RCA, but it cannot help with avoiding the issue.

To avoid the issue, the customer needs to look into the query design and execution at their side. The goal is to reduce the required memory while performing the spatial operations.

Possible options are:

1. Reduce the data set.
2. Try batching the queries, split the operation into several transactions and batches.

3. Try using a higher SLO (won't help in most cases though, as failing queries usually require a lot more memory).

More information


RCA - MS-Internal, do not share with the customer

Spatial data types are internally using CLR.



When we reach a OOM limit, a dump is triggered that causes these LSI's to be filled. We hit dump as there is a "Dump On First CLR OOM" being set, and we cannot disable it, as hitting OOM in CLR during JIT/Unload actions could cause critical failures, so it's still safer to keep it this way.

For Spatial use case, PG is slowly moving CLR implementations to the native path (out of CLR), which should prevent us from hitting similar issues in the future. But we do not have an ETA on this as this is a complicated long term fix.

Internal Doc References

- [IcM 339224255](#) 
- Defect: SQLBUVSTS #9373596 (broken link for reference:
http://sqlbuvs01:8080/Main/SQL%20Server/_workitems/edit/9373596)

Public Doc Reference

- [Spatial Data](#) 
- [Spatial data types](#) 
- [Static Aggregate Geography Methods](#) 
- [UDT](#) 

How good have you found this content?



-