Read only_replica_corrupted_statistics

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

Contents

Issue

Many cusstomers reported that executing a select query against a read-only replica fails with one of the following errors, when the same query can be executed against the primary replica successfully.

Typical error messages reported are:

> **Error 1:** Msg 9105, Level 16, State 27, Line 15 The provided statistics stream is corrupt.

> **Error 2:** Msg 9122, Level 16, State 201, Line 10 The statistics ‘%.*ls' is corrupt.

> **Error 3:** Msg 2767, Level 16, State 1, Procedure AspenDB-KPRDSB-PD.sys.sp_table_statistics2_rowset, Line 105 [Batch Start Line 0] Could not locate statistics '_WA_Sys_xxxxxxxxxxx' in the system catalogs. The same query can be executed against the primary replica successfully.

Investigation/Analysis

This is a known bug around statistics metadata and how they get invalidated on the Readable Secondaries that has existed since SQL Server 2012 in any Always-on AG setup whether on-prem/Azure. This bug is the result of a race condition, it requires specific timing between the statistics creation and queries on the secondary. If both of these actions are done frequently on any database, there is a higher likelihood that the bug will be hit. There is no other known reason why the database would be more susceptible to this issue.

Regardless of the known bug mentioned above, the error indicates a corruption, please make sure to double check the DB health before proceeding with the mitigation:

1. Identify the read-only replica from ASC’s Read Scale out section or from Kusto:

```
MonDmRealTimeResourceStats
| where  (originalEventTimestamp between (datetime(start_time)..datetime(end_time)))
| where LogicalServerName =~ 'ServerName' and database_name =~ 'DBName'
                    | order by NodeName, originalEventTimestamp asc
                    | serialize
                    | extend prevNodeName = prev(NodeName)
                    | extend nextNodeName = next(NodeName)
                    | extend prevReplicaType = prev(replica_type)
                    | extend nextReplicaType = next(replica_type)
                    | extend isFirstNodeName = (NodeName != prevNodeName)
                    | extend isLastNodeName = (NodeName != nextNodeName)
                    | extend isFirstReplicaType = (prevReplicaType != replica_type)
| extend isLastReplicaType =  (nextReplicaType != replica_type)
| where isFirstNodeName == true or isLastNodeName == true or isFirstReplicaType == true or
isLastReplicaType == true
| extend start = originalEventTimestamp, end = next(originalEventTimestamp)
                    | where isFirstNodeName == true or isFirstReplicaType == true
                    | extend replica_node = iff((replica_type == 0), 'Primary', 'Secondary')
                    | project NodeName, start, end, replica_type, replica_node
```

2. Check MonSQLSystemHealth and AlrSQLErrorsReported tables in Kusto for any errors reported that might indicate corruption: DB Corruption TSG :

```
let srv = "srv_name";
let db = "DB_name";
let startTime = datetime("2022-12-10 06:00:00");
let endTime = datetime("2022-12-22 07:56:00");
let timeRange = ago(7d);
let AppNames = MonAnalyticsDBSnapshot
 | where logical_server_name =~ srv
 | where logical_database_name =~ db
 | extend AppName = sql_instance_name
 | distinct AppName;
MonSQLSystemHealth
//| where TIMESTAMP > timeRange
| where  TIMESTAMP >= startTime
| where  TIMESTAMP <= endTime
| where AppName in~ ( AppNames )
| where
    message contains "Error: 823" or message contains "Error: 824" or message contains "Error: 825"
    or
    message contains "Error: 8646" or message contains "Error: 9003"
| summarize min(message) by AppName, AppTypeName



AlrSQLErrorsReported
| where TIMESTAMP > datetime("2022-12-18 06:00:00") and TIMESTAMP < datetime("2022-12-20 07:56:00")
| where LogicalServerName =~ "srv"
| where message contains "error"
| project TIMESTAMP , NodeName, message, code_package_version
| limit 1000
```

3. Check the data page integrity of the table that has the statistics from customer side:

```
DBCC CHECKTABLE ('Table_name');
```

Mitigation

There are two workarounds; you can go with one of them:

1. Drop the corrupted statistics, this can be done through the following steps:

    1. Identify the corrupted statistic(s) on the secondary replica (based on the table being fetched by the select query), the statistics should start with ***_WA_Sys*** since it was created by the auto create statistics:

```
With StatsTableList As (
  -- Statistics on Tables
  SELECT
    [Statistics] = stat.name,
    [Schema]     = SCHEMA_NAME(tab.schema_id),
    [Object]     = OBJECT_NAME(stat.object_id),
    [Column]     = COL_NAME(scol.object_id, scol.column_id)
  FROM
         sys.stats         stat (NOLOCK)
    Join sys.stats_columns scol (NOLOCK) ON stat.stats_id = scol.stats_id AND stat.object_id =
      scol.object_id
    Join sys.tables        tab  (NOLOCK) ON tab.object_id = stat.object_id),
StatsViewList As (
  -- Statistics on Views
  Select
     [Statistics] = stat.name,
     [Schema]     = SCHEMA_NAME(tab.schema_id),
     [Object]     = OBJECT_NAME(stat.object_id),
     [Column]     = COL_NAME(scol.object_id, scol.column_id)
  From
      sys.stats stat (NOLOCK)
    Join sys.stats_columns scol (NOLOCK) ON stat.stats_id = scol.stats_id AND stat.object_id = scol.
    Join sys.views         tab  (NOLOCK) ON tab.object_id = stat.object_id
),
StatsWithStatement As (
 -- Statistics on Tables and Views with drop statement
 Select
   *, [Statement]=Concat(N'Drop Statistics ', [Schema], '.', [Object], '.', [Statistics], ';')
 From
   StatsTableList
 Union All
 Select
   *, [Statement]=Concat(N'Drop Statistics ', [Schema], '.', [Object], '.', [Statistics], ';')
 From
   StatsViewList)
Select
    *
  From
    StatsWithStatement
  Where
    [Object] Like '%table name%' And
    [Statistics] like '_WA%' -- System statistics
  ORDER BY
    [Object]
```

    2. Drop the statistics on the primary:

```
Drop Statistics table_name._WA_Sys_xxxxxxxxxx;
```

    3. Run the user select query on the primary, which will auto-create the statistics and it will be replicated to the secondary replica.

    4. Test the query on the secondary replica.

2. Failover the secondary DB replica, this can be done:

   - Using the TSQL command:

     ```
     DBCC STACKDUMP('Failover')
     ```

   - Using the Powershell cmdlet: https://docs.microsoft.com/en-us/powershell/module/az.sql/invoke-azsqldatabasefailover

RCA Template

This defect is the result of a race condition, it requires specific timing between the statistics creation and queries on the secondary. If both of these actions are done frequently on any database, there is a higher likelihood that the defect will be hit. There is no other known reason why the database would be more susceptible to this issue.

There is an ongoing work item for this specific defect, although due to its complexity the fix is not expected to be delivered soon. The change requires new log records and a version bump - these need to be rolled out over two separate deployment trains, and each train can take several months to complete. Due to the number of occurrences hit, the team is treating this bug with high priority and sincerely apologize for any inconvenience which this has caused.

Internal reference

ICM: 279028108, 273805086,258354618
Work Item: InvalidateMDCache failed to clean outdated cache on RO replica thus new stats will not show up (you can follow the repro steps mentioned)

Root Cause Classification

Cases resolved by this TSG should be coded to the following root cause:
Azure SQL v3/Performance/SQL Server Engine bug/Other

How good have you found this content?