Geo-Replica Readable Secondary CPU troubleshooting

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

Contents

Initial note

This TSG applies to GEO-replication scenarios, in which QDS is not available at the readable secondary replicas. If the customer is using Failover Groups, the readable secondary database has QDS enabled but is in read-only state. It doesn't reflect the actual performance metrics from the secondary database.

Overview

Most backend telemetry collection is the same on the primary replica and on the read-only secondary replicas - the exception being that Query Store is not captured at the secondaries by default. Even if QDS is enabled, its value is greatly diminished because users can't turn on Query Store on a read-only database. All you get are the query hashes which can't be translated into the query text if you don't capture additional data.

This also implies that plan forcing on geo-secondary is not supported; it can't be done via QDS.

On the plus side, read-only databases eliminate a bunch of problems that read-write databases have. For example, user won't be able to cause blocking because they can't write to the database. It is still possible to have blocking at the secondaries - see Blocking on Read Replica and GeoDR - Replication lag check for reference.

The majority of the issue we experienced so far is related to high CPU.
This TSG will therefore focus on troubleshooting high CPU.

Workflow

Capture High CPU queries while the issue is occurring

Let the customer run the troubleshooting queries Q1 High CPU Queries from below for over 5 minutes.

The query will output two sections of data every 5 minutes. The first section of data points out the Top 10 CPU-consuming queries in the past 5 minutes (delta data). The second section of data returns all queries that were still running and consuming CPU when the troubleshooting query was run. See example output below at the end of this TSG.

After you identified the queries with the customer, work with the customer to tune those queries.

Root cause Analysis (RCA)

Set expectation that we can't do 100% RCA without additional collaboration because we don't have all the telemetry. Notably Query Store is not available to the customer and in the backend, we don't enable QDS by default. Even if we enable QDS, we only have the query hash but not the SQL text, and the customer still need to work with us to get the SQL query text.

Here are steps:

Determining user CPU usage

Use Q2 Kusto Real Time Resource Usage from below to confirm that the CPU usage was over 80% for the problem period. Optionally, if the customer shows you in the portal that CPU consumption was high, you can tell the customer that it's likely their queries which consumed the CPU. However, we won't be able to tell which queries consumed that CPU because the secondary replica doesn't have QDS. We will need to capture future data to troubleshoot further (next step).

Future data capturing

- Locate a client machine that has SQL Server client tools installed and make sure you can connect to your Azure SQL Database
- Save Q1 High CPU Queries (below) into a file called "CPU_query.sql"
- Run from the command line: `sqlcmd.exe -S %SERVERNAME% -U %USERNAME% -P %PASSWORD% -d %DBNAME% -w5000 -i CPU_Query.sql -o CPU_Query.sql.out`
- Wait until the problem occurs
- Stop the command sqlcmd.exe by Ctrl+C
- Examine the `cpu_query.sql.out` file and look at the period when you had high CPU. The queries in the output within the time window are the queries that consumed high CPU and need to be tuned.

Enabling backend QDS

(only applies to geo-readable secondaries before T45, and applies to all readable secondaries after T45)

You can ask PG to enable backend QDS.

However, you are no better than using te method above (Future Data Capturing) because the customer still doesn't have the Query Store to obtain the query text from. You just created additional work without much

benefit.

If you do choose this route and identify the CPU-consuming query hashes, then try to see if you can get the query text through a DMV by having the customer run Q3 DMV Query text from Query Hash. Note though that if the plan had been flushed already, then you won't have anything.

Query References

Q1 High CPU Queries

```
-- High CPU queries (that are either finished or still running)
--
if  object_id ('tempdb.dbo.#query_stats') is not null drop table #query_stats

SELECT getdate() runtime, *
INTO #query_stats
FROM
   (SELECT query_stats.query_hash,
    SUM(query_stats.total_worker_time) 'total_worker_time',
    SUM(query_stats.execution_count) 'execution_count',
    SUM(total_logical_reads) 'total_logical_reads',
    REPLACE (REPLACE (MIN(query_stats.statement_text),  CHAR(10), ' '), CHAR(13), ' ') AS 'statement_text'
  FROM
     (SELECT QS.*,
      SUBSTRING(ST.text, (QS.statement_start_offset/2) + 1,
      ((CASE statement_end_offset WHEN -1 THEN DATALENGTH(ST.text) ELSE QS.statement_end_offset END - QS.state
      FROM sys.dm_exec_query_stats AS QS
      CROSS APPLY sys.dm_exec_sql_text(QS.sql_handle) AS ST) AS query_stats
      GROUP BY query_hash) t

while 1 = 1
begin
   WAITFOR DELAY '00:05:00'
   INSERT INTO #query_stats

   SELECT getdate() runtime, *
   FROM
      (SELECT query_stats.query_hash,
       SUM(query_stats.total_worker_time) 'total_worker_time',
       SUM(query_stats.execution_count) 'execution_count',
       SUM(total_logical_reads) 'total_logical_reads',
       REPLACE (REPLACE (MIN(query_stats.statement_text),  CHAR(10), ' '), CHAR(13), ' ') AS 'statement_text'
     FROM
        (SELECT QS.*,
         SUBSTRING(ST.text, (QS.statement_start_offset/2) + 1,
         ((CASE statement_end_offset WHEN -1 THEN DATALENGTH(ST.text) ELSE QS.statement_end_offset END - QS.st
         FROM sys.dm_exec_query_stats AS QS
         CROSS APPLY sys.dm_exec_sql_text(QS.sql_handle) AS ST) AS query_stats
         GROUP BY query_hash) t

   print 'For the past 5 minutes, the following are top 10 CPU consuming queries.  Stats are captured for fini
   print '--High CPU Queries (Delta)--'
   SELECT TOP 10
      t2.runtime, t2.query_hash,
      CAST((t2.total_worker_time - (CASE WHEN t1.total_worker_time IS NULL THEN 0 ELSE t1.total_worker_time EN
      (t2.total_logical_reads - (CASE WHEN t1.total_logical_reads IS NULL THEN 0 ELSE t1.total_logical_reads E
      (t2.execution_count - (CASE WHEN t1.execution_count IS NULL THEN 0 ELSE t1.execution_count END)  ) 'Tota
      t2.Statement_Text
   FROM
      (SELECT * FROM #query_stats  WHERE runtime = (SELECT MAX(runtime) FROM #query_stats)) t2
   LEFT JOIN
     (SELECT * FROM #query_stats  WHERE runtime = (SELECT MIN(runtime) FROM #query_stats)) t1
      on t2.query_hash=t1.query_hash
   ORDER BY (t2.total_worker_time - (CASE WHEN t1.total_worker_time IS NULL THEN 0 ELSE t1.total_worker_time E

   RAISERROR (' ', 0, 1) WITH NOWAIT
   print 'The following are top 10 CPU consuming queries that have not finished running at the time of this sn
   print '--Active CPU Consuming Queries--'
   SELECT TOP 10 getdate()
      runtime, query_hash,
      cpu_time ,
      logical_reads ,
      start_time ,
      substring (REPLACE (REPLACE ((txt.text),  CHAR(10), ' '), CHAR(13), ' '), 1, 256) AS "Statement_Text"
   FROM sys.dm_exec_requests req
   CROSS APPLY sys.dm_exec_sql_text(req.sql_handle) txt
   ORDER BY cpu_time desc
   RAISERROR (' ', 0, 1) WITH NOWAIT
```

```
    DELETE #query_stats where runtime < (select max(runtime) from #query_stats)

end
```

Q2 Kusto Real Time Resource Usage

```
MonDmRealTimeResourceStats
| where TIMESTAMP >= datetime(2022-11-24 13:00:00)
| where TIMESTAMP <= datetime(2022-11-27 13:00:00)
| where LogicalServerName =~ "servername"
| where database_name =~ "databasename"
| where replica_type == 0
| summarize avg(avg_cpu_percent) , avg(avg_data_io_percent), avg(avg_log_write_percent), avg(avg_memory_usage_
| render timechart
```

Q3 DMV Query text from Query Hash

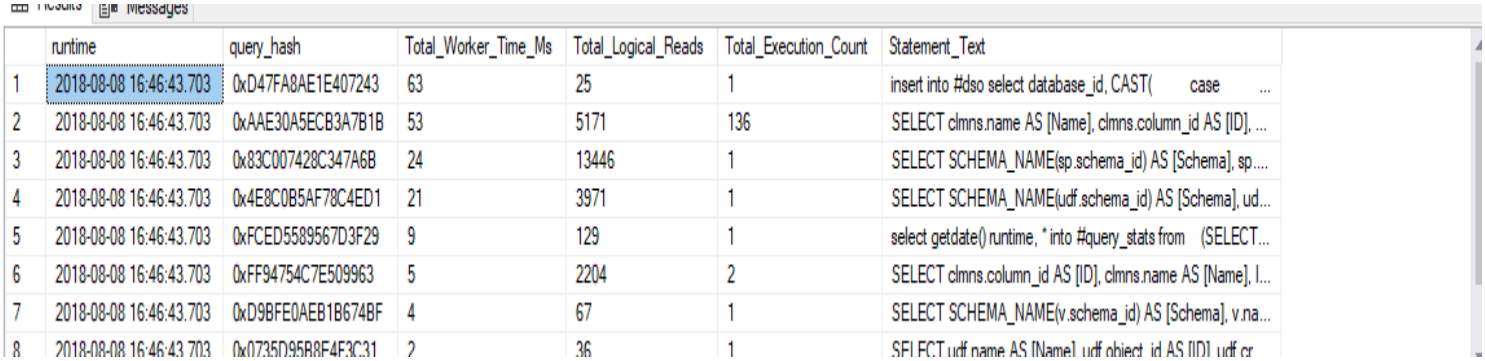
```
-- Insert desired <query_hash> below
SELECT getdate() runtime, * FROM
(SELECT query_stats.query_hash,
    SUM(query_stats.total_worker_time) 'total_worker_time',
    SUM(query_stats.execution_count) 'execution_count',
    SUM(total_logical_reads) 'total_logical_reads',
    REPLACE (REPLACE (MIN(query_stats.statement_text),  CHAR(10), ' '), CHAR(13), ' ') AS "Statement_Text"
FROM
    (SELECT
       QS.*,
       SUBSTRING(ST.text, (QS.statement_start_offset/2) + 1,
       ((CASE statement_end_offset WHEN -1 THEN DATALENGTH(ST.text)
         ELSE QS.statement_end_offset END
        - QS.statement_start_offset)/2) + 1) AS statement_text
     FROM sys.dm_exec_query_stats AS QS
     CROSS APPLY sys.dm_exec_sql_text(QS.sql_handle) AS ST) AS query_stats
     GROUP BY query_hash) t
     WHERE query_hash = <query_hash>)
--   example:
--   WHERE query_hash=0xAB8FD7BE9E25F74F
```

Example output

Example output from Q1 High CPU Queries

For the past 5 minutes, the following are top 10 CPU consuming queries. Stats are captured for finished queries only. For Running queries, see next snapshot

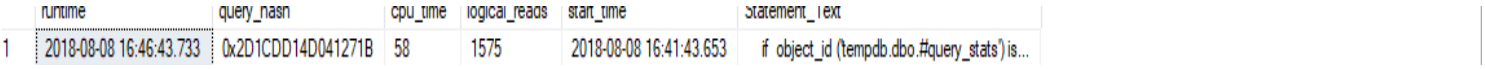
--High CPU Queries (Delta)--



	runtime	query_hash	Total_Worker_Time_Ms	Total_Logical_Reads	Total_Execution_Count	Statement_Text
1	2018-08-08 16:46:43.703	0xD47FA8AE1E407243	63	25	1	insert into #dso select database_id, CAST(case ...
2	2018-08-08 16:46:43.703	0xAAE30A5ECB3A7B1B	53	5171	136	SELECT clmns.name AS [Name], clmns.column_id AS [ID], ...
3	2018-08-08 16:46:43.703	0x83C007428C347A6B	24	13446	1	SELECT SCHEMA_NAME(sp.schema_id) AS [Schema], sp....
4	2018-08-08 16:46:43.703	0x4E8C0B5AF78C4ED1	21	3971	1	SELECT SCHEMA_NAME(udf.schema_id) AS [Schema], ud...
5	2018-08-08 16:46:43.703	0xFCED5589567D3F29	9	129	1	select getdate() runtime, * into #query_stats from (SELECT...
6	2018-08-08 16:46:43.703	0xFF94754C7E509963	5	2204	2	SELECT clmns.column_id AS [ID], clmns.name AS [Name], l...
7	2018-08-08 16:46:43.703	0xD9BFE0AEB1B674BF	4	67	1	SELECT SCHEMA_NAME(v.schema_id) AS [Schema], v.na...
8	2018-08-08 16:46:43.703	0x0735D95B8E4F3C31	2	36	1	SELECT udf.name AS [Name], udf.object_id AS [ID], udf.cr...

The following are top 10 CPU consuming queries that have not finished running at the time of this snapshot capture

--Active CPU Consuming Queries--



	runtime	query_hash	cpu_time	logical_reads	start_time	Statement_Text
1	2018-08-08 16:46:43.733	0x2D1CDD14D041271B	58	1575	2018-08-08 16:41:43.653	if object_id ('tempdb.dbo.#query_stats') is...

How good have you found this content?