One specific query runs slower than on-premises

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

Contents

Issue

We often got complaints from customers that they found some specific query runs slower in Azure SQL DB than on-premises. Here are some steps that I normally check:

Investigation / Analysis

Step 1: Table schema and indxes, data, execution parameters, platform resources

As a first step, you need to know if and how the on-premise query scenario can be compared to the Azure database:

- Are you using the same parameters when running the query in both environments?
- Are the database/table size and row counts exactly the same?
- Are the table schema and indexes the same or not? Consider to export the database to the other environment to compare if possible.
- Review what resources were available to the on-premise SQL Server and how these compare to the resources allocated to the Azure SQL database.

Step 2: Actual execution plan

In a second step, get and compare the actual query execution plans, if possible, from both on-premise and Azure SQL Database. Use the steps from article How to Capture the Actual Execution Plan.

Step 2a: Actual execution plans are different

- Run `UPDATE STATISTICC ... WITH FULLSCAN` against all the tables that are touched by the problematic query. Then check the performance and query plan again.
- Compare the compatibility level on both environments: `SELECT name, compatibility_level FROM sys.databases;`

- Check the Azure SQL server configuration options: `SELECT * FROM sys.database_scoped_configurations` and evaluate the options that appear different from its on-premise equivalent.
- If your on-premise SQL Server is running on a relatively old version, consider turning on the `LEGACY_CARDINALITY_ESTIMATION` (ALTER DATABASE SCOPED CONFIGURATION) for testing.
- Compare the MAXDOP settings, which can be seen in the actual execution plan. See Max_DOP_Issues for background information.

In addition, the query execution plan can also change over time. You can look into Query Store for this.

Step 2b: Actual execution plans are the same

We need to confirm if there is any problematic wait:

- Get the wait type and wait time from the actual execution plan
- Or check the query wait type during query execution while the slow query is running. Use the steps in article Verifying waitstats.

Quick summary of examples:

- If the wait type is `LCK_*`, use the mitigation steps from Troubleshooting Blocking and Blocking.
- If the wait type is `SOS_SCHEDULER_YIELD`, then the CPU is the bottleneck. Use the steps from CPU_Troubleshooting for further troubleshooting.
- If the wait type is `ASYNC_NETWORK_IO`, ensure your application and Azure SQL Database are in the same region. This wait type indicates that the SQL side is waiting on the application to retrieve its results. Check resource consumption on the client side and possibly investigate the network latency and speed (network trace, psping)

You may see other wait types as well and troubleshoot based on what you see. Check article Wait type for the meaning of the wait type.

Public Doc Reference

Blog article: One specific query runs slower in Azure DB than on-premises SQL Server?!

How good have you found this content?