Comparing performance between 2 MIs

Last updated by | Radhika Shah | Aug 15, 2022 at 7:48 PM PDT

Issue

There might be customers who encounter performance discrepancy between databases on 2 different Managed Instances. Customer claims that they have 2 different MIs with same/similar data set, but one performs better than the other.

Investigation/Analysis

1. Collect actual query execution plans from both the executions and compare the plans. Verify that the data sets are indeed comparable and that the plans are different enough to cause performance difference that customer claims.

2. Verify if the query has required indexes: Find the most optimum query plan from the first Kusto results and check if it has any missing index hint in the query plan xml.

- Execute this query to get the query plan xml

```
select query_plan from sys.query_store_plan where plan_id = <>
```

- Save the result as .sqlplan and open in SSMS (Sql Server Management Studio); you can check the missing index hint by searching for keyword "missing" in any xml editor.

3. Look at the operators where most of the cost spend in the query plan; optimize the query accordingly to minimize the cost:

- If most of the cost is in Key Lookup operator, then those columns can be added as included columns to the index being seek'ed; this will remove the key lookup operator cost.

4. Check if the query is affected due to Parameter Sensitive Plan (PSP) issue:

- PSP is a common issue encountered when a DBMS caches a plan created with one set of parameters, and later reuses the plan for a different set of parameters for which it's suboptimal. One way to mitigate PSP issue is by specifying option(recompile), but then you will need to pay the compilation performance cost each time you run the query. Other hints generally don't help with PSP issues. It's possible that both environments have a PSP issue, but one environment is suffering worse than the other. More details and options can be found here: Queries that have parameter sensitive plan (PSP) problems

5. Remove the IS NULL OR anti- patterns:

- "IS NULL OR " should not be used in the Where clause, it will prevent from using the index on the column. The IS NULL check should be done in the application code not in TSQL.

6. If most of the cost of the Query plan is spend on the Join then:

- Divide the query into two small queries with strict filters set in the where clause (only use AND in the where clause).

- Union all the result set of the small sub queries instead of large joins on unnecessary data.

7. If query plan are the same, but performance is different, remember to check if there is code package change:

   ```
   MonSQLSystemHealth
   | where LogicalServerName =~ {ServerName}
   | where AppName == {AppName}
   | distinct code_package_version
   ```

How good have you found this content?