High RecompileDNR and TemptableChanged recompilations

Last updated by | Ricardo Marques | Mar 2, 2023 at 7:46 AM PST

Contents

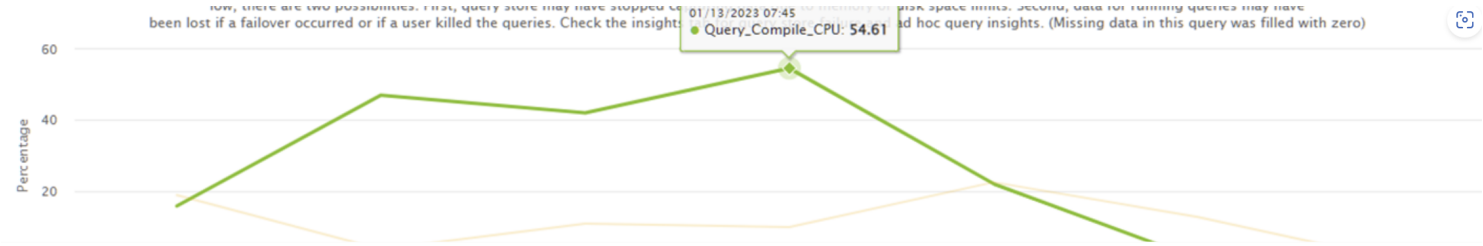
Issue

Customer complaining about query performance. There is high CPU consumption.

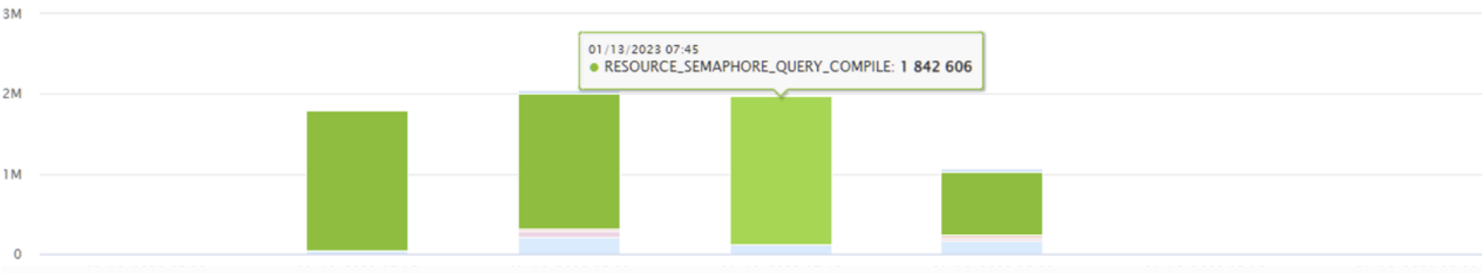
Investigation/Analysis

Note: for troubleshooting this kind of scenarios use the existing workflow - Workflow for High CPU troubleshooting

Looking at ASC it's possible to see that Compile CPU is high



Also that prevalent wait is RESOURCE_SEMAPHORE_QUERYCOMPILE.



When looking into the query compilation data it's possible to observe that there is a high recompile values with reason **RecompileDNR** and **TemptableChanged**

RecompileDNR - In Microsoft SQL Server, Deferred Name Resolution allows you to create database objects that reference other objects that may not exist yet. When you execute a query that references an object with

deferred name resolution, SQL Server defers checking the existence and permissions of the object until runtime. This can improve the efficiency of creating and compiling code.

If you modify the referenced object after the original object is compiled, it is possible that the original object may no longer function correctly, as the object's compiled code still references the original object. In this case, you may need to recompile the original object to update its compiled code with the changes made to the referenced object.

Deferred Name Resolution can simplify the process of creating and modifying database objects, especially in complex systems with many interdependent objects

TemptableChanged - When you create and compile code that references a temporary table in Microsoft SQL Server, the compiled code includes a reference to the temporary table. If the structure of the temporary table changes after the code is compiled, it can cause errors or unexpected behaviour when the code is executed.

To ensure that the compiled code reflects the current structure of the temporary table, you may need to recompile the code. Recompiling the code will update the compiled code with the current structure of the temporary table, and ensure that the code functions correctly with the modified table structure.

Note that temporary tables in SQL Server are specific to the session that creates them. This means that a temporary table created in one session is not visible to other sessions. If the code that references a temporary table is executed in a different session, the temporary table must be created in that session as well.

In short the issue is caused by the high usage of temporary table, in multiple stored procedures or in one procedure but executed multiple times per second.

Mitigation

The mitigation is done by the customer at code level.

Instead of temporary tables he can use table variables or derived tables.

Again, this will be e decision that has to be done and tested by the customer.

Public Doc reference

Table variables

Derived tables

Internal reference

360932102

How good have you found this content?