SQL Server: Temporary Objects

Module 4: Tempdb and Temporary Object Considerations

Joe Sack
Joe@SQLskills.com

pluralsight
hardcore developer training

Module Introduction

- **When making heavy use of temporary objects you need to have a strategy for configuring, supporting and troubleshooting tempdb**
- **You also need to understand the various advantages and disadvantages of temporary object capabilities so that you can leverage them in your code appropriately and avoid introducing performance and scalability issues**
- **This module will cover:**
  - Identification and troubleshooting of tempdb contention
  - A discussion of temporary object advantages, disadvantages and appropriate usage scenarios

Planning for Tempdb Usage

- **Both temporary tables and table variables make use of the tempdb system database**
- **There is only *one* tempdb system database per SQL Server instance, so with a high enough volume of temporary object usage, this database can become a central point of contention**
- **Additionally, there is only *one data file* allocated to tempdb by default, which means that a heavy amount of small temporary object creation and deletion can cause allocation page latch contention on the single data file**
  - Latch waits on tempdb resources like 2:1:1 (first PFS page) and 2:1:3 (first SGAM page)
    - PFS tracks allocated pages and heap page free space
    - GAM tracks allocated extents
    - SGAM tracks which mixed extents have one or more unallocated pages

Addressing Allocation Page Contention

- **Validate number of equally sized tempdb data files**
  - CSS guidance is:
    - <= 8 cores, #files = #cores
    - >8 cores, #files = 8 (add in 4-file chunks / monitor for contention)
    - Where a core is a logical processor core
      - E.g. a server with two CPUs, and each CPU having 4 physical cores, has 8 logical cores, and 16 logical cores if hyper-threading is enabled
- **Consider enabling TF 1118 enabled for SGAM contention, allocating full extents to each temporary object**
  - See Paul Randal's blog post, "Misconceptions around TF 1118", http://bit.ly/Wp1d2t

Temporary Object Advantages

- **Temporary tables**
  - Provide intermediate result-sets that can reduce query plan complexity
    - Not a guaranteed solution, but a valid tool for query plan quality issues
  - Ability to isolate a table's data "per user" execution
  - Have column-level statistics like permanent tables
  - Can use TRUNCATE, SELECT INTO, CREATE INDEX, ALTER TABLE, IDENTITY_INSERT, ROLLBACK TRANSACTION
- **Table variables**
  - Can be used within scalar and multi-statement table-valued functions
  - Inherits the current database's collation implicitly
  - Will not directly impact recompilations (although you may actually want this in some circumstances)
  - Can be passed as input parameters to stored procedures

Temporary Object Disadvantages

- **Temporary tables**
  - Heavy tempdb usage via object creates/drops can lead to latch contention
  - Not always optimal compared to inline, single-statement alternative
  - Not supported in user-defined functions
  - Inherits tempdb collation for non-contained databases
  - System caching of temporary tables and associated statistics may cause unexpected query plan shapes and incorrect statistics
    - For an advanced, nuanced overview of this subject, see the Paul White (MVP) blog post, “Temporary Tables in Stored Procedures”, http://bit.ly/145DXNw
- **Table variables**
  - Heavy tempdb usage via object creates/drops can lead to latch contention
  - No column-level statistics, cardinality estimate skews
  - Cannot TRUNCATE, SELECT INTO, CREATE INDEX, ALTER TABLE, IDENTITY_INSERT, ROLLBACK TRANSACTION
  - Parallelism inhibited for non-SELECT operations

Key Factors Influencing Temporary Object Usage

- **Functionality**
  - Does it meet my requirements?
    - Interim result-sets
    - Concurrent, isolated result-sets without having to create permanent tables
- **Performance and scalability**
  - For complex queries, sometimes breaking the query down into steps with temporary objects can improve query plan quality and increase performance
- **Result-set sizes**
  - Consider the lack of column-level statistics for table variables and the impact on query plan quality
    - Small result-sets in table variables may produce fine query execution plans, but what about large result-sets?

Course Summary

- **We covered:**
  - Temporary tables
  - Table variables
  - Tempdb and temporary object considerations

- **Thanks for watching!**