Chat Q&A Day 1

Q: Are there be any databases you would _not_ enable Query Store on?

A: The short answer: if the database has an ad-hoc workload (versus procedural) and is high-volume, you could see a degredation in performance with Query Store enabled.

Query Store Performance Overhead: What you need to know: https://www.sqlskills.com/blogs/erin/query-store-performance-overhead/

Query Store Settings: https://www.sqlskills.com/blogs/erin/query-store-settings/

Important Query Store Fixes - January 2019: https://www.sqlskills.com/blogs/erin/important-query-store-fixes-january-2019/

Q: Does that mean that you cannot compare cost from different servers?

A: Cost is determined independent of server hardware. Theoretically, if you run the same queries from the demos on your server or laptop, you would have the same cost numbers.

Q: What's good approach to find - small queries-adhoc running at high frequencies (query : host name and frequencies - aggregation)..if I want to use Extended Events. Is there performance overhead with XE if I am already running at the peak resource usage of sql server (2012)?

A: I would not recommend using Extended Events in that scenario (because you're at peak resource use), you can find those queries in the plan cache – Kimberly should cover this on Day 3. Assuming it's the same query that's running repeatedly but with different input values, then they should all have the same query_hash and you can aggregate on that to see what's running most often. If you were NOT resource bound, you could do this with Extended Events – with an event like sql_statement_completed you can add query_hash as an action – that's the easiest way to aggregate queries based on query text, even with different input parameters.

Q: What is the easiest way to see that a query is running slow due to issues with statistics? We have a few very big tables, which causes performance issues if statistics is updated with default values. What is the best way to find the sample rate sweet spot for big tables?

A: There is no easy way to find queries that are running slow *because* of an issue with statistics. You'll have to dig into query perf and the plan to see that. But, for the tables which require a larger sample for the optimizer to best understand the distribution of data, I tend to update those with FULLSCAN, as updating with anything above 50% sample doesn't seem to run any faster than a FULLSCAN (in my testing, YMMV). If a FULLSCAN and 200 step histogram still isn't giving the optimizer enough information about the distribution of data, then look at using Kimberly's scripts to find what columns have a lot of skew and might benefit from filtered statistics. She did a session at PASS in 2013 in

Charlotte, you can get the resources here: https://www.sqlskills.com/resources/conferences/PASS2013-DBA321DemoScripts.zip

Q: in above case..so optimizer decides...instead of nested loop for customerid and then picking all rest column from clustered index..it just scans the clustered index only..is that correct...can there be a case where optimizer goes with nested loop

A: Absolutely – that's a case where the query is parameter sensitive, and based on the value passed in during compilation and the selectivity of that value, it may go with a seek and key lookup into a nested loop, or for a non-unique value it might do a full scan of an index.

Q: sometimes I've seen where a query will take milliseconds to run but several seconds to compile a plan... what methods can be used or metrics to

A: You can get information about compilation duration from Query Store (in sys.query_store_query) if you're on SQL 2016 or higher. Within a query plan (actual plan, or the one in the plan cache) you can also see compilation information. For example, if the StatementOptmEarlyAbortReason attribute has a value of TimeOut, then it's likely a complex query that needs to be simplified. I take any large queries and break them down into smaller queries, and start running those and measuring compilation, and then start adding back on.

Q: Are the memory grants and the memory for sorting still part of buffer pool?

A: Yes, as of SQL Server 2012 everything comes from the memory manager and it's "any size page" allocator (e.g. buffer pool, plan cache, etc.), which manages the memory available to SQL Server based on the max server memory setting. Memory allocations for SQLCLR and for providers that execute out of process do not pull from the same memory. You can get information about memory grants (what's been granted, what's waiting to be granted in sys.dm_exec_query_memory_grants).

Q: how to find which sql query is recompling most in sql server 2008/2012

A: In 2008 use Trace and in 2012 use Extended Events. There is a recompile event, and you can capture the statement to see what query/queries are recompiling most frequently, as well as the reason.