Enhancements that will make your SQL database engine roar Part 2

Pedro Lopes (@sqlpto)
Senior Program Manager
Data Group

SQL Server Tiger Team

Microsoft

Pedro Lopes

@sqlpto

pedro.lopes@microsoft.com

Senior Program Manager

Focused on SQL Server Relational Engine

7 years at Microsoft

Agenda

- Part #1
  - Performance and Scale
  - Diagnostics and Management
- Part #2
  - Diagnostics and Management
  - Query Execution and Performance

Diagnostics and Management (Cont.)

Backup and Restore tracing

- Long running tasks
- Not enough insight into progress
- Trace flag (3004) output is cryptic and unformatted
- Errorlog
  - TF 3014 = TF 3014 + TF 3004 + TF 3212 (buffer config details)
  - Formatted messages - Backup(dbname) and Restore(dbname)
  - All errors (currently sent to the client, which can be lost)
- Extended Event
  - backup_restore_progress_trace

Demo

backup_restore_progress_trace extended event

Database Recovery tracing

- Limited information available during database recovery activities such as Analysis, Redo and Undo
- Errorlog
  - Does not output during Analysis phase
  - “Recovery of database ‘%’ is xx% complete (approximately yy seconds remaining)
- Three new Extended Events:
  - database_recovery_progress_report
  - database_recovery_times
  - database_recovery_trace

Database Recovery progress

- Progress and time estimates for various phases

name	timestamp	database_name	phase	estimated_remaining_time_sec	percent_complete	total_elapsed_time_sec
database_recovery_progress_report	2015-09-23 15:40:24....	LargeDB	Analysis	49	0	0
database_recovery_progress_report	2015-09-23 15:40:24....	LargeDB	Analysis	37	0	0
database_recovery_progress_report	2015-09-23 15:40:24....	LargeDB	Redo	37	0	0
database_recovery_progress_report	2015-09-23 15:40:24....	LargeDB	Complete	0	100	1

- Recovery time for specific steps during database startup

name	timestamp	database_name	recovery_step	recovery_time
database_recovery_times	2015-09-23 15:40:24....	LargeDB	Total	3019
database_recovery_times	2015-09-23 15:40:24....	LargeDB	LogMgrPreRecoveryTime	2672
database_recovery_times	2015-09-23 15:40:24....	LargeDB	AnalysisRecTime	210
database_recovery_times	2015-09-23 15:40:24....	LargeDB	PhysicalRecoveryTime	283

Troubleshooting long running Recovery

SQL Server Tiger Team

name	timestamp	database_name	trace_message
database_recovery_trace	2015-09-23 15:40:21....	LargeDB	RecoveryUnit::PreRecovery
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Database LargeDB has more than 10000 virtual log files which is excessive. Too many virtual lo e log
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Virtual Log File Table summary: VLF count = 62.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	RecoveryUnit::PhysicalRecovery
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Starting database recovery.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Analysis phase starting.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Estimated log amount for ANALYSIS = 36677632 bytes.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Recovery of database 'LargeDB' (13) is 0% complete (approximately 49 seconds remain). Phase 1 of
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	ANALYSIS LogBlock Stats - 29987 blocks, 28 stalls, 198 ms.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	ANALYSIS Scan Stats - 35908096 bytes, 30040 blocks, 114 cache misses, 113 blocking, 205892 RA depth.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	ANALYSIS Consumer Stats - 0 LC, 30040 LogPool, 30040 disk, 0 pressures, 0 shrinks.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Recovery of database 'LargeDB' (13) is 0% complete (approximately 37 seconds remain). Phase 1 of 3. This is an informational message only. No user action is required.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	ANALYSIS LogBlock Stats - 29987 blocks, 28 stalls, 198 ms.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Dirty Page Table summary: page count = 12.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Analysis phase done.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Redo phase starting.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Estimate: REDO = 36468736 bytes of log, UNDO = 0 transactions and 0 log records. Buffer pool rampup OFF.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Recovery of database 'LargeDB' (13) is 0% complete (approximately 37 seconds remain). Phase 2 of 3. This is an informational message only. No user action is required.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Don't allow deferring failed transactions because no backup exists.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	REDO LogBlock Stats - 29987 blocks, 0 stalls, 37 ms.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	REDO Scan Stats - 35908096 bytes, 30040 blocks, 1 cache misses, 0 blocking, 17726122 RA depth.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	REDO Consumer Stats - 0 LC, 30040 LogPool, 0 disk, 0 pressures, 0 shrinks.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	REDO IO Stats - 20008 page fixes, 33 ms for read, 9994 wasted reads, 0 misses, 0 waits, 0 ms fo wait.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	REDO ReadAhead Stats - 12 RAs, 1 dups, 0 stalls, 0 ms for RA, 35908096 bytes processed.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	REDO EagerWrite Stats - 11 writes, 11 pages, 0 overflows.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	Redo phase done.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	9996 transactions rolled forward in database 'LargeDB' (13:0). This is an informational message only. No user action is required.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	RecoveryUnit::CompletePhysical: Completing physical recovery.
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	RecoveryMgr::PhysicalCompletion
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	RecoveryUnit::FixupPostRedo
database_recovery_trace	2015-09-23 15:40:24....	LargeDB	RecoveryMgr::TransitionToUpdateable: Preparing for the log to be updateable.

Number of VLFs

Estimated log size

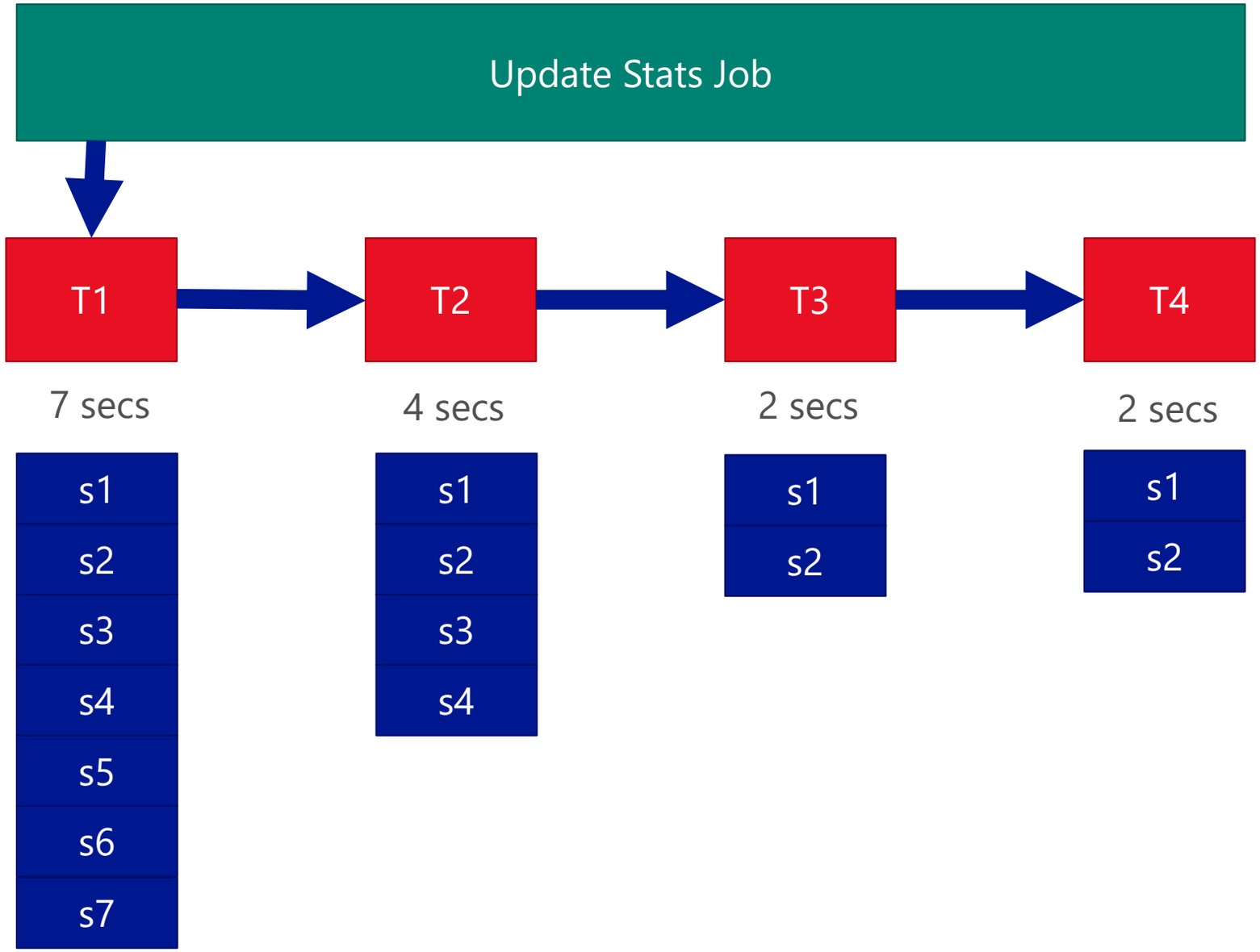
Number of transactions

Time spent in each phase

Index Usage DMV behavior updated

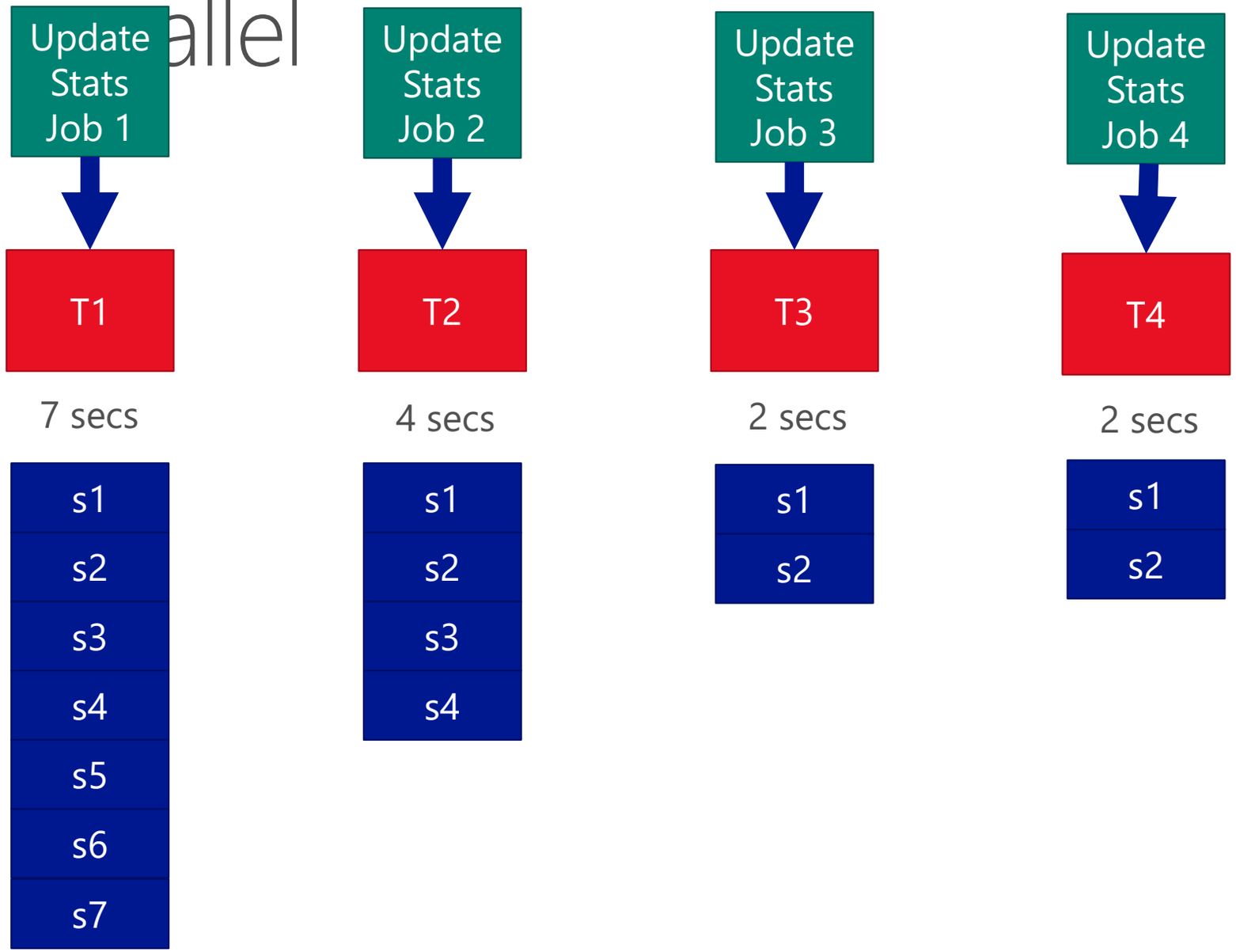
- Up until SQL Server 2008 R2, using index usage stats entries in sys.dm_db_index_usage_stats to make some assumptions over index design and workload patterns was common.
- In SQL Server 2012 and higher versions, entries in this DMV were reset with every index rebuild.
- In SQL Server 2016 and 2014 SP2, we are restoring the expected behavior with usage stats tracking in sys.dm_db_index_usage_stats, where entries will not be removed.

Today, Update Statistics executed serially



Total time for the Job is 15 seconds

Today, Update Statistics executed in parallel

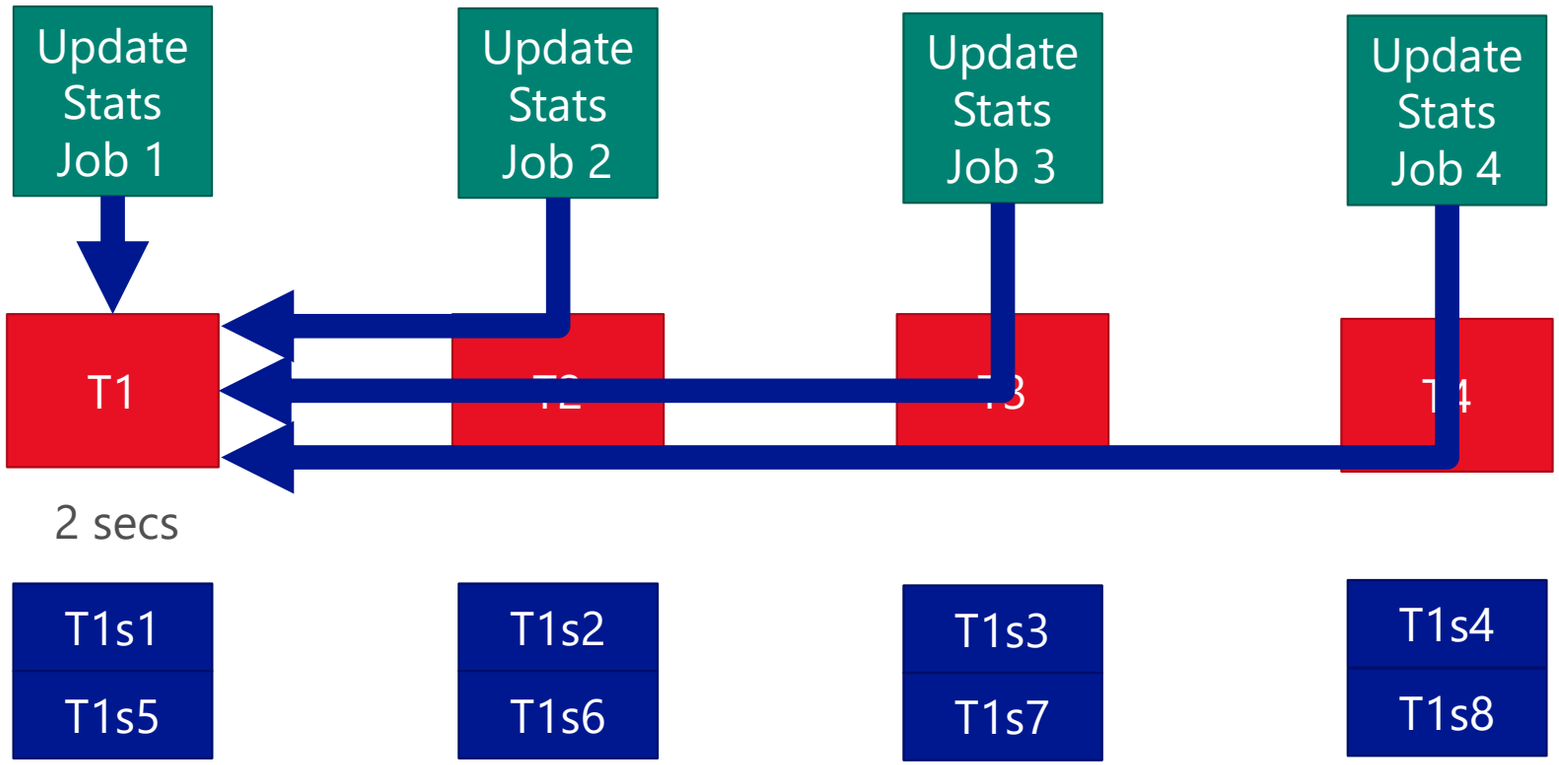


Total time for the Job is 7 seconds

Simultaneous Update Statistics?

- For VLDB scenarios, running UPDATE STATISTICS on the entire database can take considerable time to complete.
- UPDATE STATISTICS uses parallelism but cannot run simultaneously over different Statistics objects in the same table.
- Let's look at a few scenarios…

Problem with Update Statistics executed simultaneously



Job1

Grant Sch-M Lock on T1s<…>

Job2

Deny Sch-M Lock on T1s<…>

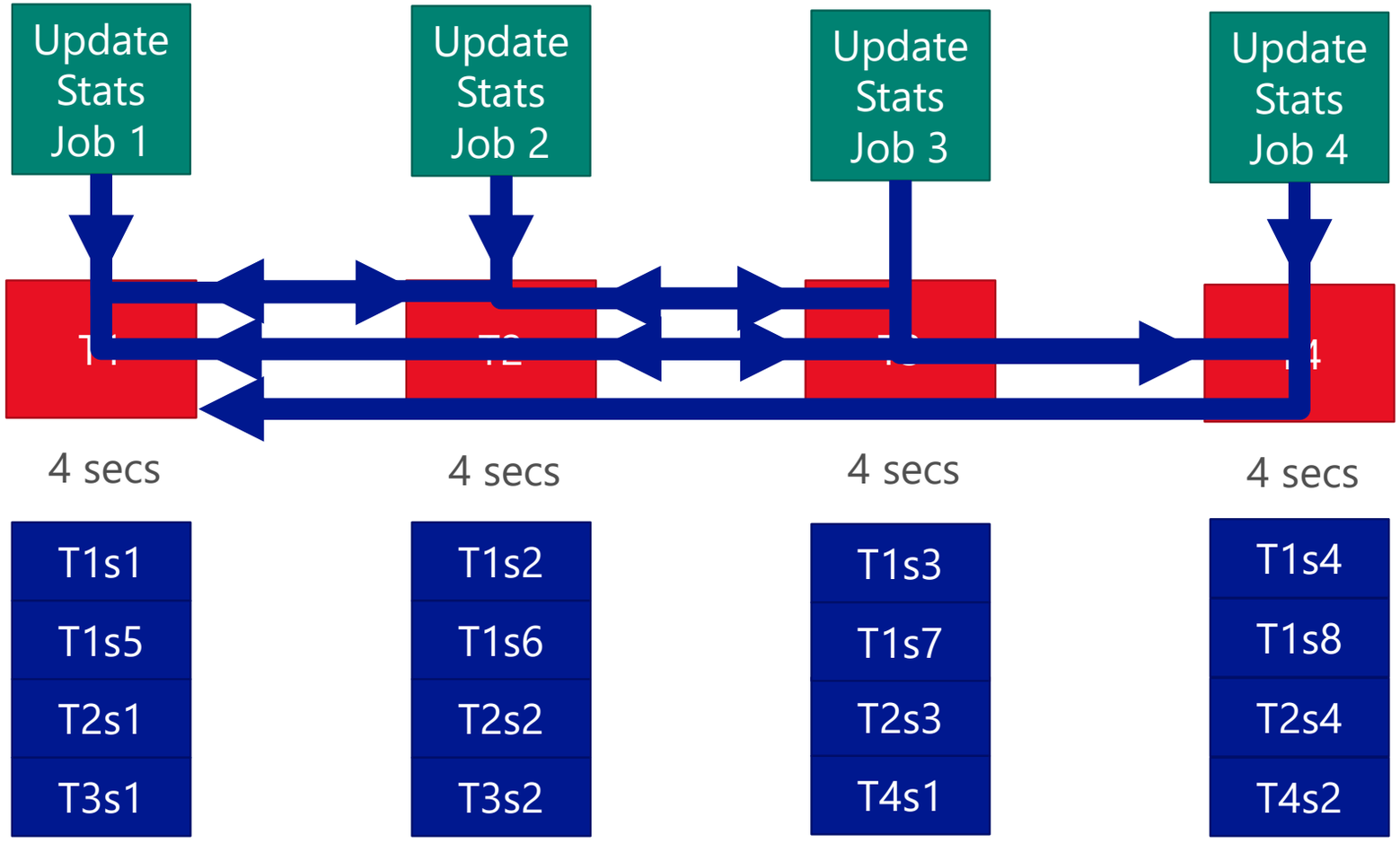
Job3

Deny Sch-M Lock on T1s<…>

Job4

Deny Sch-M Lock on T1s<…>

Update Statistics executed simultaneously



Total time for the Job is 4 seconds

Grant U Lock on T1s<...>

Grant U Lock on T2s<...>

Grant U Lock on T3s<...>

Grant U Lock on T4s<...>

KB 3156157

In SQL Server 2014 SP1 CU6, using TF 7471

Query Execution and Performance

Database Compat Level and TF 4199

- History
  - TF 4199 introduced in SQL Server 2005 SP3 CU6 to collect all optimizer fixes in a single trace flag
  - These fixes were not enabled by default in next major version (continued under 4199)
- Going forward:

SQL Compat. Level	Trace Flag 4199	Optimizer fixes before SQL Server 2016 RTM	Optimizer fixes after SQL Server 2016 RTM
120	Off	Disabled	Disabled
120	On	Enabled	Disabled
130	Off	Enabled by compatibility level	Disabled
130	On	Enabled by compatibility level	Enabled by compatibility level

Note: Setting no. 3 is recommended for customers who are newly upgrading to SQL Server 2016.

min and max query memory grant option

- User control over min and max grant size in percentages
  - OPTION (MAX_GRANT_PERCENT=0.1), meaning 0.1% of max allowed query memory under Resource Governor
  - The valid value is between 0 and 100%
  - MAX_GRANT_PERCENT >= MIN_GRANT_PERCENT
- Why use a floating point value?
  - On a high end machine with 1 TB of memory, 1% can be already 10GB
- SQL Server 2016 and SQL Server 2014 SP2

Grants in DMV and Showplan

- New columns in sys.dm_exec_query_stats

total_grant_kb	last_grant_kb	min_grant_kb	max_grant_kb	total_used_grant_kb	last_used_grant_kb
783288	783288	783288	783288	0	0

min_used_grant_kb	max_used_grant_kb	total_ideal_grant_kb	last_ideal_grant_kb	min_ideal_grant_kb	max_ideal_grant_kb
0	0	28592000	28592000	28592000	28592000

- Showplan extended to include grant usage per thread and iterator

Memory Grant	783288
⊟ MemoryGrantInfo	
DesiredMemory	28592000
GrantedMemory	783288
GrantWaitTime	0
MaxUsedMemory	0
RequestedMemory	783288
RequiredMemory	4064
SerialDesiredMemory	28588448
SerialRequiredMemory	512

Demo

Memory grant Extended Events

Addressing large memory grant requests

Optimized Nested Loops (or Batch Sort)

- Optimization aimed at minimizing I/O during a Nested Loop when the inner side table is large
- Nested Loop join may try to reorder the input rows to improve I/O performance

Issue

- Extreme memory grants found when the outer table of the Nested Loop join has a predicate that filters the result to a small input, but the batch sort appears to be using an estimate for cardinality that is equivalent to the entire outer table

Option

- Disable this feature globally using Trace Flag 2340
- Use new query hints (more on that coming)

Side-effects

- OOM conditions, memory pressure for plan cache eviction, or unexpected RESOURCE_SEMAPHORE waits

Demo

Optimized Nested Loops and Memory grants

Per-operator level performance stats

- Need insight on the performance of each node and thread
- Showplan extended to include RunTimeCountersPerThread
  - Up to SQL Server 2016

```
<RunTimeInformation> <RunTimeCountersPerThread Thread="0" ActualRows="8001"
ActualEndOfScans="1" ActualExecutions="1" /> </RunTimeInformation>
```

  - SQL Server 2016 and SQL Server 2014 SP2

```
<RunTimeInformation> <RunTimeCountersPerThread Thread="0" ActualRows="8001"
ActualRowsRead="10000000" Batches="0" ActualEndOfScans="1" ActualExecutions="1"
ActualExecutionMode="Row" ActualElapsedms="965" ActualCPUms="965"
ActualScans="1" ActualLogicalReads="26073" ActualPhysicalReads="0"
ActualReadAheads="0" ActualLobLogicalReads="0" ActualLobPhysicalReads="0"
ActualLobReadAheads="0" /> </RunTimeInformation>
```

Per-operator level performance stats

- New Extended Event query_thread_profile
  - Showplan time scale = milliseconds
  - xEvent time scale = microseconds for CPU and total time.

Name	Category	Channel
query_thread_profile	execution	Debug

query_thread_profile

Reports the performance of each node and thread of a query plan after execution

Event Fields	Description
actual_batches	Number of batches processed by this thread
actual_execution_mode	Execution mode of the thread. 0 indicates row mode, 1 indicates batch mode
actual_logical_reads	Number of logical pages read
actual_physical_reads	Number of physical pages read
actual_ra_reads	Number of read-ahead pages read
actual_rebinds	Number of rebinds for this thread
actual_rewinds	Number of rewinds for this thread
actual_rows	Number of rows processed by this thread
actual_writes	Number of pages written
cpu_time_us	CPU time in microseconds
io_reported	Is IO reported?
node_id	The ID of the node in the query plan
thread_id	The ID of the thread running in this node
total_time_us	Cumulative time in microseconds, including waits

Demo

Per-operator level performance stats

What is Predicate Pushdown?

- Parts of the filter predicate may match an index key and may therefore be used to run an index seek or range scan.
  - Remaining parts of the predicate are known as "residual" and must be evaluated for each row output by the scan or range operation.
  - This would be a filter operator.
- To improve performance, SQL Server can push such a filter down to the table access operator itself.
  - In the case of an inaccurate cardinality estimation that is related to parameter sensitivity, the scan-below filter may be processing a larger number of rows than expected.
  - This is hidden from an actual execution plan: actual number of rows returned are rows after the residual predicate is applied and not the actual number of rows that are scanned from a table or index.

Predicate Pushdown in Showplan

- If a row is filtered out by the Storage Engine due to an non-sargable predicate, can you find that?
- Can be confusing to see high CPU or large logical reads (from tracing), but the query plan doesn't reflect that.
- New Showplan attribute *Number of Rows Read*.
  - This attribute provides information about how many rows were read by the operator before the residual predicate was applied.
- In SQL Server 2016, 2014 SP2 and 2012 SP3

Demo

Predicate Pushdown in Showplan

Basic stats terms

- Density = 1/Distinct Value Count
- Frequency = Row Count * Density
- Selectivity of Predicate = Row Count satisfying Predicate

The CE assumptions

Uniformity

- Distinct values are evenly spaced and that they all have the same frequency.
- More precisely, within each histogram step, distinct values are evenly spread and each value has same frequency.

Containment

- Users query for data that exists.
- For equi-join of two tables, we factor in the predicates selectivity in each input histogram before joining histograms and come up with the JOIN selectivity.

Independence

- Data distributions on different columns are assumed to be independent, unless correlation information is available and usable.

Inclusion

- For filter predicates where Col = Const, the constant is assumed to actually exist for the associated column.
- If a corresponding histogram step is non-empty, one of the step's distinct values is assumed to match the value from the predicate.

What assumptions changed in new CE?

Simple Containment becomes Base Containment

- Users might query for data that does not exist, so we use probabilistic approach.
- For equi-join of two tables, we use the base tables histograms to come up with the JOIN selectivity, and then factor in the predicates selectivity.
- Trace Flag 2301 to enable Base Containment in previous versions.

Independence becomes Correlation

- The combination of the different column values are not necessarily independent.
- This may resemble more real-life data querying.

New CE using multi-column stats

- new CE derives cardinality for correlated columns by sorting the filters according to their density, where the smallest density value is first, using single-column statistics.
  - Since the density is expressed in a range between 0.xxxx and 1 the smaller values means lower density, better selectivity or more different values.
  - We then use only the first four most selective filters to calculate the combined density, using **Exponential Backoff**, expressed like $p_0 \times p_1^{1/2} \times p_2^{1/4} \times p_3^{1/8} \times Tc$
- For SQL Server 2016, if multi-column statistics over the predicate columns are available, those will be used to derive **frequency base** estimations.

Database Compat Level and CE version

DB Compat Level	110 or below	120	130
No TF	70	120	130
TF 9481 enabled	70	70	70
TF 2312 enabled	120	120	130

Demo

Estimation using multi-column stats

Query progress estimation

Limited feedback mechanism for query execution progress

Very difficult to get insight into long running queries

Estimated Query plans do not contain any run-time information

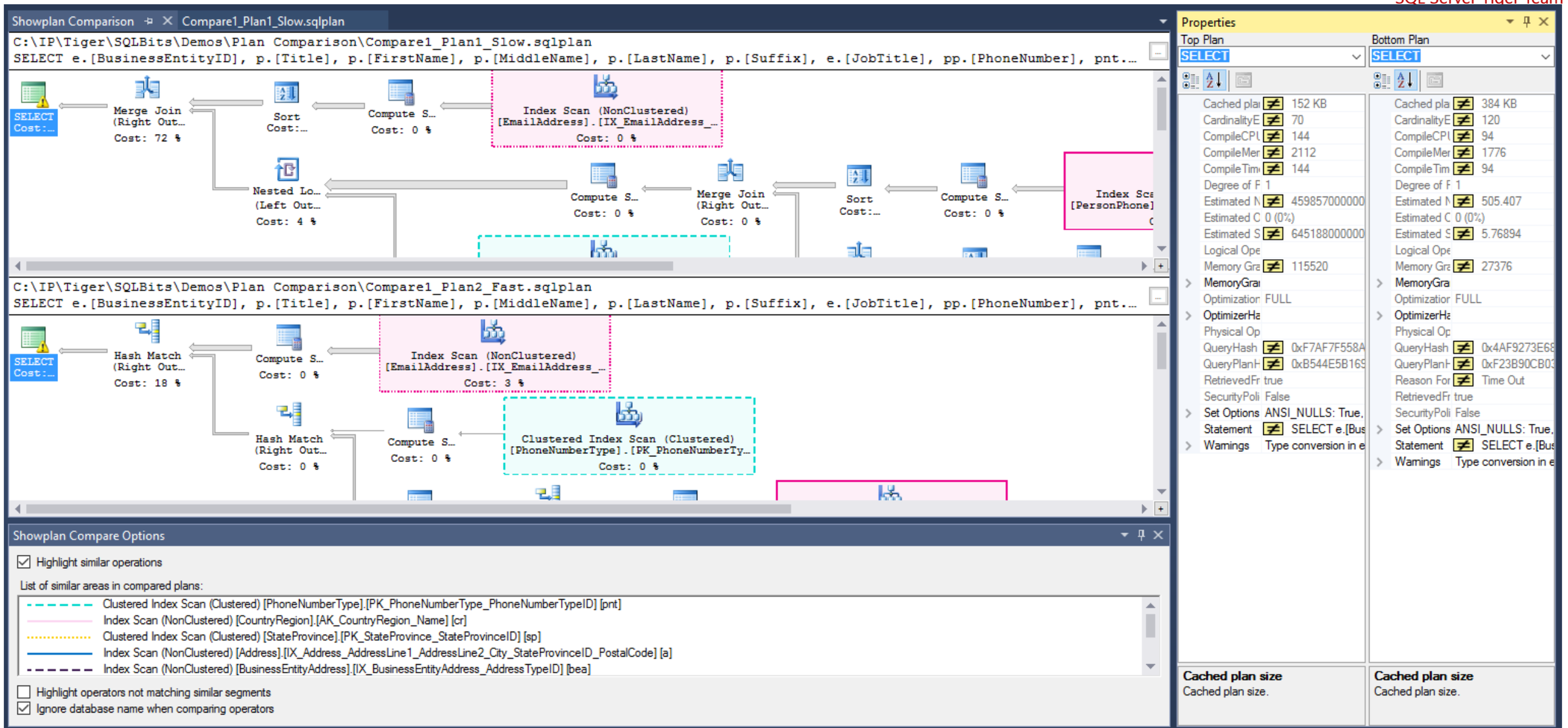
Actual row counts, memory grant usage, execution time warnings

Demo

Live Query Statistics

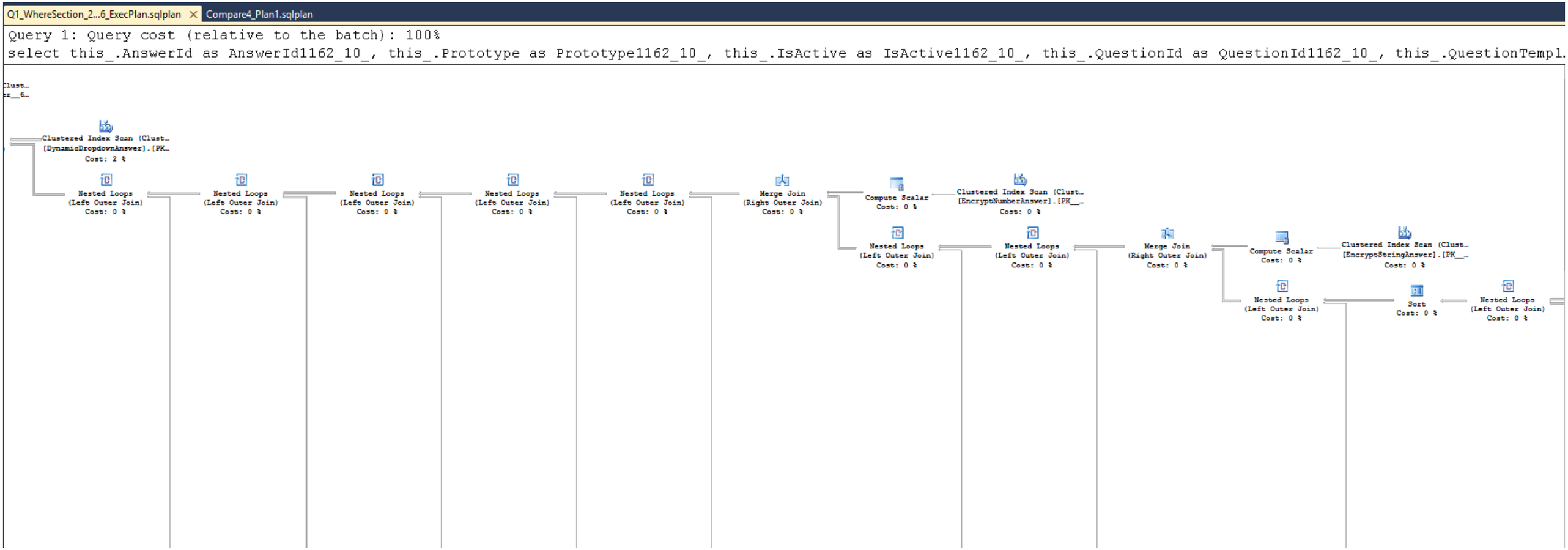
SSMS Plan Comparison

SQL Server Tiger Team



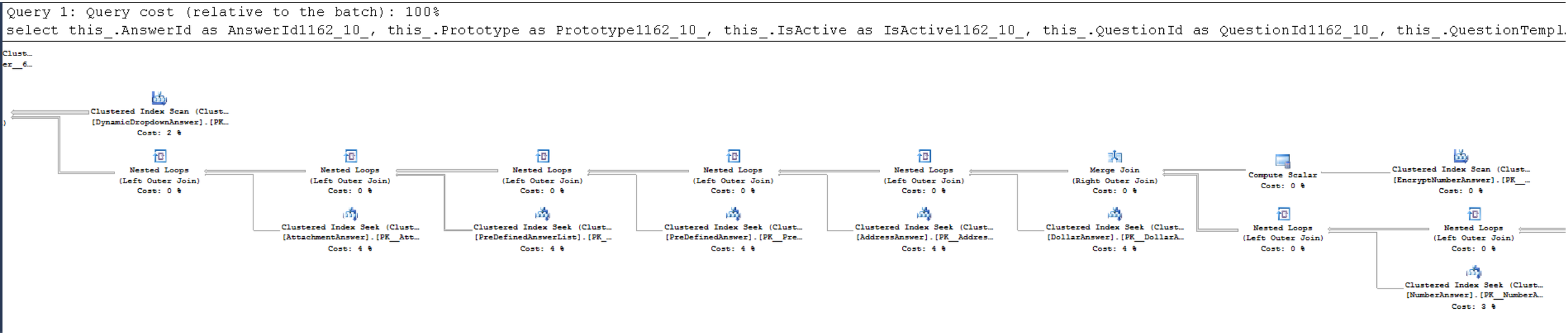
Real estate usage - Up to SSMS 2016

- Vertical lines limit plan visibility

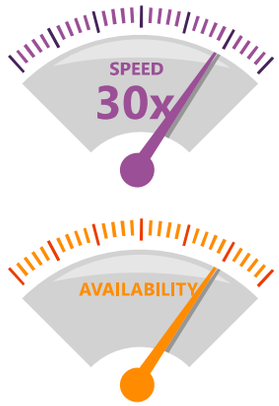


Real estate usage - New

- Optimized layout algorithm render plans in a more condensed view, so more of the plan fits on the screen without having to zoom out



In Review: Objectives And Takeaways

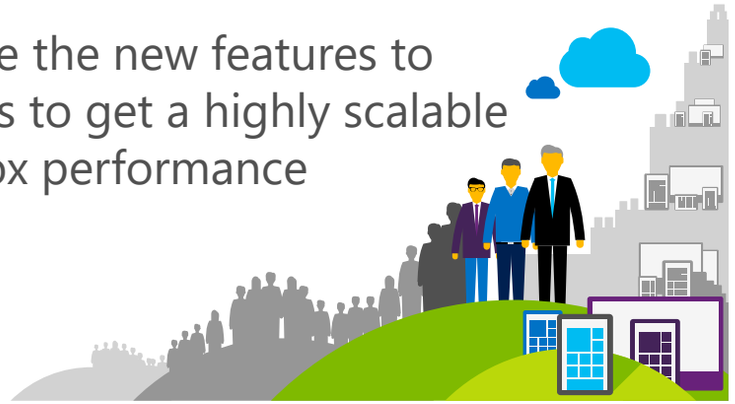


It just works - performance and scale in SQL Server 2016 database engine

Learn about new diagnostics improvements for SQL Server engine

Learn how to use the new diagnostics to troubleshoot common performance issues

Evangelize the new features to customers to get a highly scalable out-of-box performance



SQL Server Tiger Team

Survey, decks and demos:
Part 2: http://speakerscore.com/Roar2

Microsoft