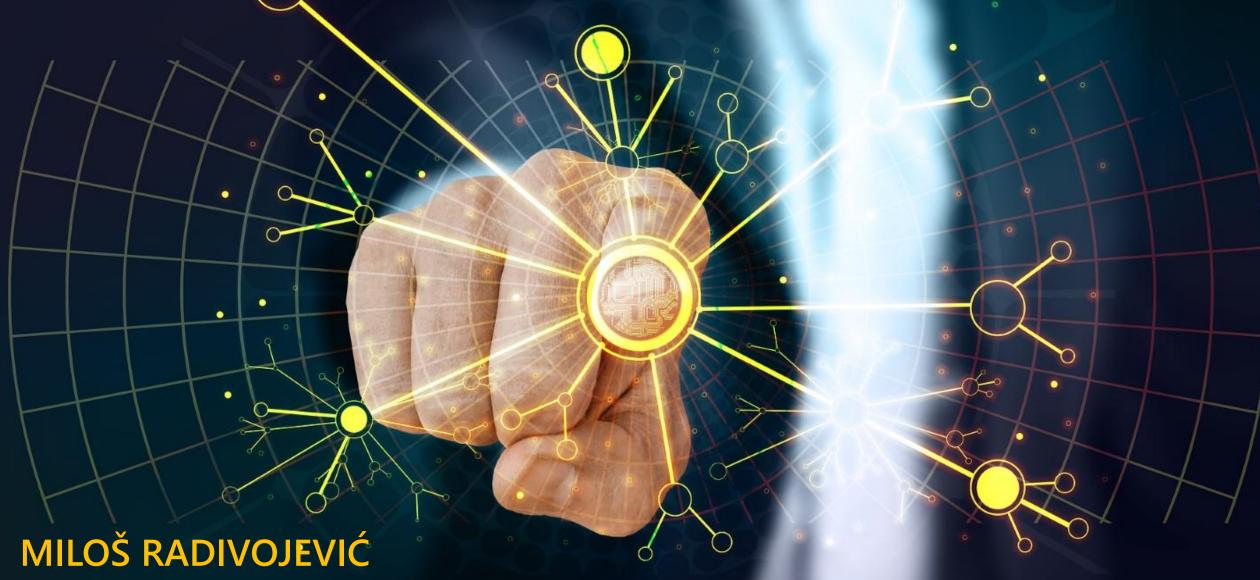
# INTELLIGENT QUERY PROCESSING IN SQL SERVER 2019



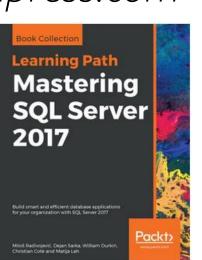
DATA PLATFORM MVP, BWIN GVC, AUSTRIA

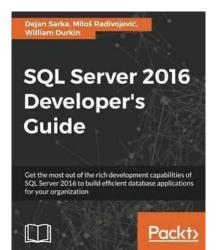
# Miloš Radivojević

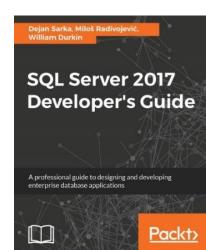
- Data Platform MVP
- Principal Database Consultant at bwin, Vienna, Austria
- Co-Founder: SQL Pass Austria
- Conference Speaker, Book Author















#### Slides and Code

https://bit.ly/2yUgISb



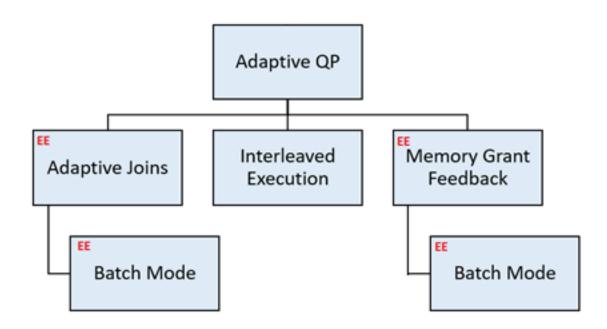
#### Inefficient Execution Plans in SQL Server

- Execution plans are sometimes suboptimal
- Affected Queries:
  - Queries using table variables
  - Queries with scalar user-defined functions
  - Queries referencing multi-statement table valued functions
  - Complex queries
  - Queries with tables with skew data distribution
- Issues:
  - Inappropriate operator choice (Nested Loops vs. Hash Match Join)
  - Memory Grant under- or overestimation

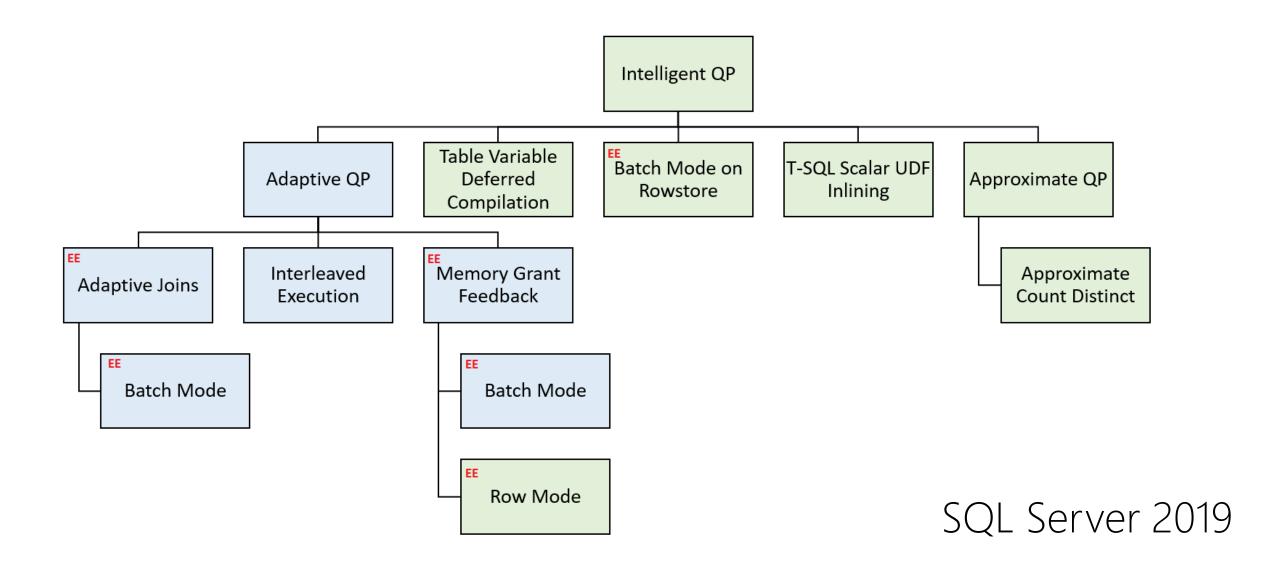
#### Inefficient Execution Plans in SQL Server

- SQL Server 2016 and prior
  - All plan decisions at the compile time (operators, memory)
  - Used "blindly" for consecutive query executions
  - No changes in cached plan (without recompiling)
- SQL Server 2017 Adaptive Query Processing
  - Creating a better plan for queries using MSTVF with the interleaved execution
  - Postponing decision about the join operator to the runtime (adaptive join)
  - Updating a part of the cached plan (memory grant)
- SQL Server 2019 Intelligent Query Processing
  - Additional adaptive improvements, but also some overall; therefore new name
     => Intelligent QP

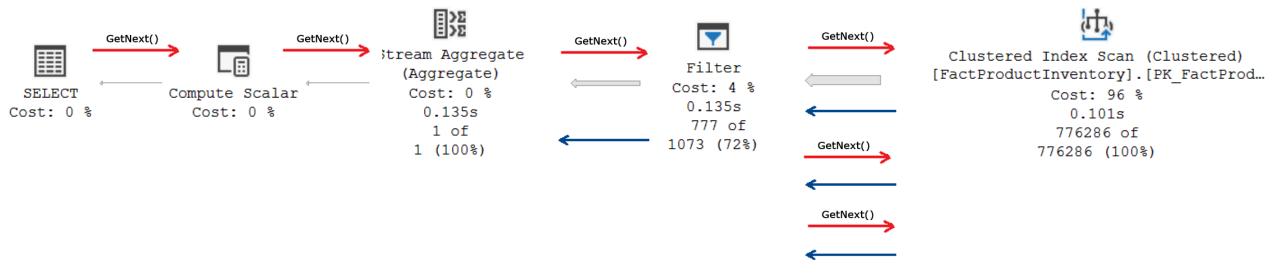
## Adaptive Query Processing



## Intelligent Query Processing

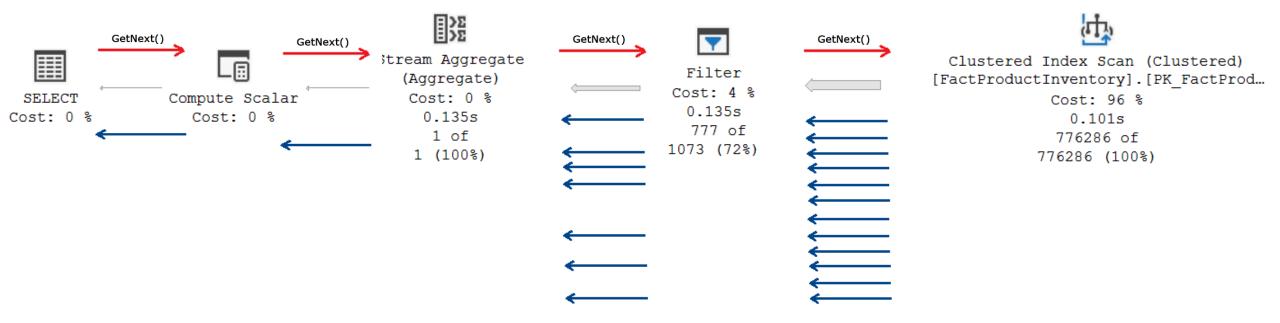


#### **ROW Mode**



Inefficient; the same instructions for every row, overhead of giving control to another operator and taking it back

#### **BATCH Mode**



batch of rows as working unit: up to 900 rows, depends on the number and size of columns and CPU L2 cache

#### What is Batch Mode?

- Batch mode allows query operators to work on a batch of rows, instead of just one row at a time
- At the CPU level multiple rows processed at once instead of one row
  - Number of processing instructions reduced
- Better CPU cache utilization and increased memory throughput
- Not exactly documented how to get the number of rows in batch (900 in all my tests)
- Can be beneficial for queries that are CPU bound

#### Batch Mode on Columnstore/Rowstore

- Batch Mode on columstore introduced in SQL Server 2012
  - Improvements up to 20x faster queries!
- Batch Mode on rowstore introduced in SQL Server 2019
  - Some queries could be significantly faster
  - In my examples 2-5x faster!

```
USE AdventureWorksDW2019;
     GO
     SET NOCOUNT ON SET STATISTICS TIME ON;
     GO
     SELECT COUNT(*), MAX(UnitsIn) FROM dbo.FactProductInventory OPTION (USE HINT ('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'));
     GO
     SELECT COUNT(*), MAX(UnitsIn) FROM dbo.FactProductInventory;
     GO
 Query 1: Query cost (relative to the batch): 53%
 SELECT COUNT(*), MAX(UnitsIn) FROM dbo.FactProductInventory OPTION (USE HINT ('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'))
                                                             Clustered Index Scan (Cluste ...
                                     Stream Aggregate
                                                             [FactProductInventory].[PK F...
                                        (Aggregate)
                Compute Scalar
                                        Cost: 11 %
  SELECT
                                                                      Cost: 89 %
                   Cost: 0 %
 Cost: 0 %
                                          0.174s
                                                                        0.099s
                                                                                               SOL Server Execution Times:
                                           1 of
                                                                      776286 of
                                                                                                 CPU time = 219 ms, elapsed time = 210 ms.
                                         1 (100%)
                                                                    776286 (100%)
 Query 2: Query cost (relative to the batch): 47%
 SELECT COUNT(*), MAX(UnitsIn) FROM dbo.FactProductInventory
                                         猖
                                                                     įŢ,
                                                        Clustered Index Scan (Cluste ...
                                      Hash Match
                Compute Scalar
                                                        [FactProductInventory].[PK F ...
                                      (Aggregate)
                   Cost: 0 %
                                      Cost: 1 %
  SELECT
                                                                 Cost: 99 %
                    0.000s
                                                                                              SQL Server Execution Times:
 Cost: 0 %
                                       0.001s
                                                                   0.040s
                     1 of
                                        1 of
                                                                 776286 of
                                                                                                CPU time = 46 ms, elapsed time = 50 ms.
                   1 (100%
                           Physical Operation
                                                     Hash Match
                                                               776286 (100%)
                           Logical Operation
                                                      Aggregate
                           Actual Execution Mode
                                                         Batch
                           Estimated Execution Mode
                                                         Batch
Physical Operation
                           Compute Scalar
                                             Physical Operation
                                                                            Clustered Index Scan
                           Compute Scalar
Logical Operation
                                             Logical Operation
                                                                            Clustered Index Scan
Actual Execution Mode
                                  Batch
                                             Actual Execution Mode
                                                                                      Batch
Estimated Execution Mode
                                  Batch
                                             Estimated Execution Mode
                                                                                      Batch
```

```
SELECT COUNT(*), MAX(UnitPrice) FROM dbo.FactInternetSales OPTION (USE HINT ('QUERY_OPTIMIZER_COMPATIBILITY_LEVEL_140'));
      GO
      SELECT COUNT(*), MAX(UnitPrice) FROM dbo.FactInternetSales;
Query 1: Query cost (relative to the batch): 50%
SELECT COUNT(*), MAX(UnitPrice) FROM dbo.FactInternetSales OPTION (USE HINT ('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'))
                                    Stream Aggregate
                                                             Clustered Index Scan (Clustered)
                                                           [FactInternetSales].[PK FactInterne...
                                       (Aggregate)
                                                                       Cost: 96 %
                Compute Scalar
                                       Cost: 4 %
Cost: 0 %
                  Cost: 0 %
                                        0.020s
                                                                         0.012s
                                                                                       SQL Server Execution Times:
                                         1 of
                                                                        60398 of
                                                                                        CPU time = 16 ms, elapsed time = 19 ms.
                                                                      60398 (100%)
                                       1 (100%)
Query 2: Query cost (relative to the batch): 50%
SELECT COUNT(*), MAX(UnitPrice) FROM dbo.FactInternetSales
                                                                          ďψ
                                         35
                                    Stream Aggregate
                                                             Clustered Index Scan (Clustered)
                                                           [FactInternetSales].[PK FactInterne...
                                      (Aggregate)
                                                                                      SQL Server Execution Times:
               Compute Scalar
                                       Cost: 4 %
                                                                       Cost: 96 %
                                                                                        CPU time = 16 ms, elapsed time = 18 ms.
```

0.015s

60398 of

60398 (100%)

Cost: 0 %

Cost: 0 %

0.024s

1 of

1 (100%)

#### No Batch mode!

#### Batch Mode on RowStore

- Native support
  - No tricks with fake columnstore indexes or other optimizer delusions
- Initial heuristics considers potential benefits of batch mode for operators
  - Interesting table (at least 131.072 rows)
  - Interesting batch operators: join, aggregate or window aggregate
  - At least one of the batch operator's input should have not less than 131.072 rows

#### Batch Mode on RowStore

sqlserver.batch\_mode\_heuristics Extended Event

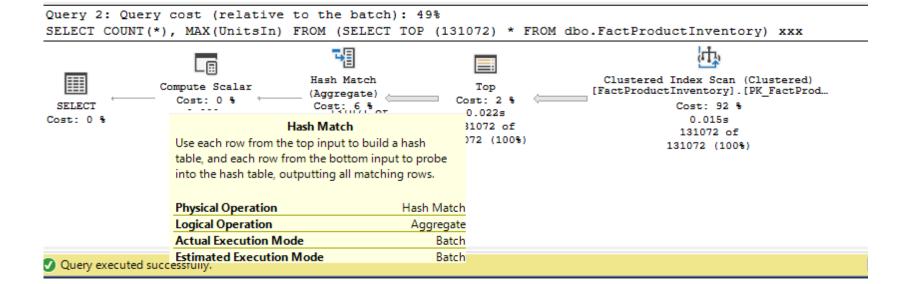
Displaying 4 Events			Displaying 4 Events		
name t	imestamp 🔺		name	timestamp 🔺	
batch_mode_heuristics 2	019-10-24 16:25:54.8502441		batch_mode_heuristics 2	019-10-24 16:25:54.8502441	
batch_mode_heuristics 2	019-10-24 16:25:54.9770007		batch_mode_heuristics 2	019-10-24 16:25:54.9770007	
batch mode heuristics 2	019-10-24 16:25:58.0653381		batch mode heuristics 2	019-10-24 16:25:58.0653381	
	019-10-24 16:25:58.3971789	<b>•</b>		019-10-24 16:25:58.3971789	
butch_mode_redusties 2	010 10 24 10.23.30.307 1703		butch_mode_neumatics	313 10 24 10.23.30.3371703	
vent:batch_mode_heuristics (2019-10-24	16:25:54.9770007)	Ever	nt:batch_mode_heuristics (2019-10-24	16:25:58.3971789)	
Details		Det	tails		
Details Field	Value		tails eld	Value	
	Value True	Fic		Value True	
Field		Fic	eld		
Field are_plan_affecting_actions_allowed	True	Fide are for	eld re_plan_affecting_actions_allowed	True	
Field are_plan_affecting_actions_allowed found_batch_operator_in_solution	True False	Fig. are for for for	eld re_plan_affecting_actions_allowed rund_batch_operator_in_solution	True	
Field are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate	True False False	File and for	eld re_plan_affecting_actions_allowed rund_batch_operator_in_solution rund_interesting_global_aggregate	True True True	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join	True False False False	File are for	eld re_plan_affecting_actions_allowed rund_batch_operator_in_solution rund_interesting_global_aggregate rund_interesting_join	True True True False	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_nary_join	True False False False False	Find an arm for	eld re_plan_affecting_actions_allowed rund_batch_operator_in_solution rund_interesting_global_aggregate rund_interesting_join rund_interesting_nary_join	True True True False False	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_nary_join found_interesting_table	True False False False False False False False False	Find an arm for	eld re_plan_affecting_actions_allowed rund_batch_operator_in_solution rund_interesting_global_aggregate rund_interesting_join rund_interesting_nary_join rund_interesting_table	True True True False False True False True	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_ioin found_interesting_nary_join found_interesting_table found_interesting_window_aggregate	True False False False False False False False False	Find any form for for form for for form for for form for for form for for form for	eld re_plan_affecting_actions_allowed re_plan_affecting_actions_allowed rund_batch_operator_in_solution rund_interesting_global_aggregate rund_interesting_join rund_interesting_nary_join rund_interesting_table rund_interesting_window_aggregate	True True True False False True False True	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_ioin found_interesting_nary_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so	True False	Figure 1 Fig	eld  re_plan_affecting_actions_allowed re_plan_affecting_actions_allowed rund_batch_operator_in_solution rund_interesting_global_aggregate rund_interesting_join rund_interesting_nary_join rund_interesting_table rund_interesting_window_aggregate rund_significant_batch_operator_in_solution recommended rund_significant_batch_operator_in_solution recommended rund_significant_batch_operator_in_solution recommended rund_significant_batch_operator_in_solution recommended rund_significant_batch_operator_in_solution recommended rund_significant_batch_operator_in_solution recommended r	True True False False False True False True False True True True True	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_nary_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heuristics	True False	Figure 1 Fig	eld  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_global_aggregate  rund_interesting_global_aggregate  rund interesting nary join  rund interesting table  rund_interesting_window_aggregate  rund_significant_batch_operator_in_sc  _batch_mode_enabled_by_heuristics	True True False False False True False True False True True True True	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_ioin found_interesting_nary_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heuristics is_batch_mode_enabled_unconditionally	True False	File are for for for for for for is, is, is,	eld  re_plan_affecting_actions_allowed  und_batch_operator_in_solution  und_interesting_global_aggregate  und_interesting_noin  und interesting nary join  und interesting table  und_interesting_window_aggregate  und_significant_batch_operator_in_sc _batch_mode_enabled_unconditional	True True False False False True False True False Jution True True True True y	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_mary_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heunistics is_batch_mode_enabled_unconditionally is_batch_processing_enabled	True False	File and for for for for for for for for for finding is,	eld  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_global_aggregate  rund_interesting_global_aggregate  rund_interesting_noin  rund interesting_noin  rund interesting_table  rund_interesting_window_aggregate  rund_significant_batch_operator_in_su_ batch_mode_enabled_by_heuristics  batch_mode_enabled_unconditional  batch_processing_enabled	True True False False False True False  True False Jution True True True True True True True	
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_ioin found_interesting_nary_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heuristics is_batch_mode_enabled_unconditionally is_batch_processing_enabled is_query_plan_using_batch_processing	True False	File are for for for for for is is is is is late	eld  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_global_aggregate  red_interesting_global_aggregate  red_interesting_ioin  red_interesting_nary_ioin  red_interesting_window_aggregate  red_interesting_interesting_window_aggregate  red_interesting_interesting_interesting_window_aggregate  red_interesting_interes	True True False False False Irue False True False Jution True True True True True True True True	- actProductInvent
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_nary_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heuristics is_batch_mode_enabled_unconditionally is_batch_processing_enabled is_query_plan_using_batch_processing last_optimization_level	True False	File are for	eld  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_global_aggregate  red_interesting_global_aggregate  red_interesting_join  red_interesting_nary_join  red_interesting_window_aggregate  red_interesting	True True True False False False Irue False Ulution True True True True True True True True	- actProductInvent
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heuristics is_batch_mode_enabled_unconditionally is_batch_processing_enabled is_query_plan_using_batch_processing last_optimization_level sql_text	True False	File are for	eld  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_global_aggregate  red interesting_global_aggregate  red interesting_nary_join  red interesting_table  red interesting_window_aggregate  red interesting_window_aggregate  red interesting_table	True True True False False False Inue False True False True True True True True True True Tru	- act Product Invent
Field  are_plan_affecting_actions_allowed found_batch_operator_in_solution found_interesting_global_aggregate found_interesting_join found_interesting_table found_interesting_window_aggregate found_significant_batch_operator_in_so is_batch_mode_enabled_by_heuristics is_batch_mode_enabled_unconditionally is_batch_processing_enabled is_query_plan_using_batch_processing last_optimization_level sql_text total_batch_cost	True False -1	File and for	eld  re_plan_affecting_actions_allowed  re_plan_affecting_actions_allowed  re_plan_affecting_global_aggregate  rund_interesting_global_aggregate  rund_interesting_ploin  rund interesting nary join  rund interesting window_aggregate  rund_significant_batch_operator_in_sc  _batch_mode_enabled_by_heuristics  _batch_mode_enabled_unconditional  batch_processing_enabled  _query_plan_using_batch_processing  st_optimization_level  al_text  tal_batch_cost	True True True False False True False True False True False True True True True  True True True Tr	-actProductInvento

```
USE AdventureWorksDW2019;
    GO
    --row mode
    SELECT COUNT(*), MAX(UnitsIn) FROM (SELECT TOP (131071) * FROM dbo.FactProductInventory) xxx;
    GO
    --batch mode
    SELECT COUNT(*), MAX(UnitsIn) FROM (SELECT TOP (131072) * FROM dbo.FactProductInventory) xxx;
    GO;
Query 1: Query cost (relative to the batch): 51%
SELECT COUNT(*), MAX(UnitsIn) FROM (SELECT TOP (131071) * FROM dbo.FactProductInventory) xxx
                                                                                         ďψ
                                       器
                                                            Stream Aggregate
                                                                             Clustered Index Scan (Clustered)
                    ....
                                                            Top
                                                                           [FactProductInventory].[PK_FactProd...
                                    (Aggregate)
                                                         Cost: 2 %
 SELECT
               Compute Scalar
                                    Cost: 11 %
                                                                                      Cost: 87 %
                                                           0.096s
Cost: 0 %
                 Cost: 0 %
                                      0.105s
                                                                                        0.089s
                                                         131071 of
                                       1 of
                                                                                      131071 of
                                                        131071 (100%)
```

1 (100%)

row mode

131071 (100%)



batch mode

```
USE WideWorldImporters;
      GO
                                    Warehouse.ColdRoomTemperatures Archive OPTION(USE HINT('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'));
      GO
                            FROM Warehouse.ColdRoomTemperatures Archive;
      SELECT COUNT(*)
Query 1: Query cost (relative to the batch): 68%
SELECT COUNT(*) FROM Warehouse.ColdRoomTemperatures Archive OPTION (USE HINT ('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'))
                      <u>Σ</u>
                                                                                    1
                Stream Aggregate
                                                           Parallelism
                                                                              Stream Aggregate
                                                                                                          Hash Match
                                                                                                                                                   Parallelism
                                                                                                                                                                                 Table Scan
                                      Concatenation
                                                                                                                                                                        [memory optimized history t
                   (Aggregate)
                                                         (Gather Streams)
                                                                                 (Aggregate)
                                                                                                     (Right Anti Semi Join)
                                                                                                                                               (Distribute Streams)
                                        Cost: 0 %
                   Cost: 0 %
                                                            Cost: 0 %
                                                                                 Cost: 3 %
                                                                                                          Cost: 54 %
                                                                                                                                                    Cost: 0 %
oute Scalar
                                                                                                                                                                                 Cost: 0 %
                                         0.546s
ost: 0 %
                    0.546s
                                                             0.546s
                                                                                   0.544s
                                                                                                            0.510s
                                                                                                                                                     0.000s
                                                                                                                                                                                   0.000s
                                          8 of
                     1 of
                                                                                   8 of
                                                                                                          3654736 of
                                                                                                                                                       0 of
                                                                                                                                                                                    0 of
                                                              8 of
                                        8 (100%)
                    1 (100%)
                                                            4 (200%)
                                                                                  4 (200%)
                                                                                                        3654730 (100%)
                                                                                                                                                     10 (0%)
                                                                                                                                                                                  10 (0%)
                                                                                          1
                                                         Stream Aggregate
                                                                                       Table Scan
                                                                                                                             SOL Server Execution Times:
                                                                              [memory optimized history ta ...
                                                           (Aggregate)
                                                            Cost: 0 %
                                                                                       Cost: 0 %
                                                                                                                                CPU time = 3842 ms, elapsed time = 615 ms.
                                                             0.000s
                                                                                        0.000s
                                                              0 of
                                                                                          0 of
                                                                                                                                                   3654736 of
                                                             4 (0%)
                                                                                        10 (0%)
                                                                                                                                                  3654740 (99%)
Query 2: Query cost (relative to the batch): 32%
SELECT COUNT(*) FROM Warehouse.ColdRoomTemperatures Archive
                                                                                                                                                       ÷
                                                                                                                                                                                    1
                                                        Parallelism
                                                                               Hash Match
                                                                                                          Hash Match
                                                                                                                                                   Parallelism
             Stream Aggregate
                                                                                                                                                                                 Table Scan
                                   Concatenation
                                                                                                                                                                        [memory optimized history t
                (Aggregate)
                                                      (Gather Streams)
                                                                           (Partial Aggregate)
                                                                                                     (Right Anti Semi Join)
                                                                                                                                               (Distribute Streams)
                                    Cost: 0 %
e Scalar
                 Cost: 0 %
                                                                                Cost: 0 %
                                                                                                           Cost: 8 %
                                                                                                                                                    Cost: 0 %
                                                                                                                                                                                 Cost: 0 %
                                Physical Operation
                                                               Hash Match
: 0 %
                  0.256s
                                                                                 0.008s
                                                                                                            0.018s
                                                                                                                                                      0.000s
                                                                                                                                                                                   0.000s
                  1 of
                                Logical Operation
                                                                Aggregate
                                                                                  1 of
                                                                                                          3654736 of
                                                                                                                                                       0 of
                                                                                                                                                                                    0 of
                 1 (100%)
                                                                                4 (25%)
                                                                                                        3654730 (100%)
                                                                                                                                                     10 (0%)
                                                                                                                                                                                  10 (0%)
                                Actual Execution Mode
                                                                    Batch
                                Estimated Execution Mode
                                                                    Batch
                                                                                       1
                                                            Ξ>Σ
                                                                                                                                           Clustered Index Scan (Cluste ...
                                                      Stream Aggregate
                                                                                    Table Scan
                                                                                                                                           [ColdRoomTemperatures Archiv...
                                                         (Aggregate)
                                                                           [memory optimized history ta ...
                                                                                                                                                    Coet . 01 &
                                                                                                                               SQL Server Execution Times:
```

CPU time = 1642 ms, elapsed time = 306 ms.

```
WITH cte AS(
 SELECT ColdRoomTemperatureID, ROW NUMBER() OVER(PARTITION BY ColdRoomTemperatureID ORDER BY ColdRoomTemperatureID) rn
 FROM WarehoUSE.ColdRoomTemperatures Archive
 SELECT * FROM cte WHERE rn > 1 OPTION (USE HINT ('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'));
 GO
  WITH cte AS(
 SELECT ColdRoomTemperatureID, ROW NUMBER() OVER(PARTITION BY ColdRoomTemperatureID ORDER BY ColdRoomTemperatureID) rn
 FROM WarehoUSE.ColdRoomTemperatures Archive
 SELECT *
              FROM cte WHERE rn > 1:
Query 1: Query cost (relative to the batch): 85%
WITH cte AS( SELECT ColdRoomTemperatureID, ROW NUMBER() OVER(PARTITION BY ColdRoomTemperatureID ORDER BY ColdRoomTemperatureID) rn FROM Warehouse.ColdRoomTemperatures Archive ) SELECT
                                                                                                Parallelism
                                                    Sequence Project
                                                                                             Merge Join
                                                                                                                     Merge Join
                                                                                                                                                              Parallelism
                                    Filter
                                                                            Segment
                                                                                                                                              Sort
                                                                                                                                                                                  [memory or
               (Gather Streams)
                                                    (Compute Scalar)
                                                                                            (Concatenation)
                                                                                                                 (Right Anti Semi Join)
                                                                                                                                                           (Distribute Streams)
                                   Cost: 0 %
                                                                           Cost: 0 %
                                                                                                                                             Cost: 0 %
  SELECT
                 Cost: 1 %
                                                       Cost: 0 %
                                                                                                                     Cost: 2 %
                                                                                                                                                               Cost: 0 %
                                    1.562s
                                                                            1.445s
                                                                                                                                             0.000s
Cost: 0 %
                  1.563s
                                                       1.530s
                                                                                               1.377s
                                                                                                                      1.333s
                                                                                                                                                                0.000s
                                                                          3654736 of
                                       0 of
                                                                                                                                              0 of
                                                      3654736 of
                                                                                             3654736 of
                                                                                                                     3654736 of
                                                                                                                                                                 0 of
                     0 of
                                  1096420 (0%)
                                                                         3654740 (99%)
                                                                                                                                             10 (0%)
                1096420 (0%)
                                                     3654740 (99%)
                                                                                            3654740 (99%)
                                                                                                                   3654730 (100%)
                                                                                                                                                                10 (0%)
                                                                                                                                                                 Parallelism
                                                                                                                                              Sort
                                                                                                                                                              (Repartition Streams)
                                                                                                                                            Cost: 84 %
                                                                                                                                                                  Cost: 3 %
                                                                                                                                             1.295s
                                                                                                                                                                   0.602s
                                                                                                                                            3654736 of
                                                                                                                                                                  3654736 of
 SOL Server Execution Times:
                                                                                                                                           3654740 (99%)
                                                                                                                                                                 3654740 (99%)
CPU time = 8514 ms, elapsed time = 1574 ms.
                                                                                                                                                                        1
                                                                                                                                       Parallelism
                                                                                                                                                                     Table Scan
                                                                                                                        Sort
                                                                                                                                    (Distribute Streems)
                                                                                                                                                           [memory optimized history table 117...
Query 2: Query cost (relative to the batch): 15%
WITH cte AS( SELECT ColdRoomTemperatureID, ROW NUMBER() OVER(PARTITION BY ColdRoomTemperatureID ORDER BY ColdRoomTemperatureID) rn FROM WarehoUSE.ColdRoomTemperatures Archive ) SELECT
                Parallelism
                                                                                                                    Hash Match
                                                                                                                                                              Parallelism
                                                                            Sort
                                    Filter
                                                    Window Aggregate
                                                                                            Concatenation
                                                                                                                                                                                 [memory_op
               (Gather Streams)
                                                                                                               (Right Anti Semi Join)
                                                                                                                                                           (Distribute Streams)
                                   Cost: 0 %
                                                       Cost: 0 %
                                                                          Cost: 48 %
                                                                                             Cost: 0 %
  SELECT
                                                                                                                    Cost: 4 %
                                                                                                                                                               Cost: 0 %
                 Cost: 5 %
                                    0.002s
                                                                            0.116s
                                                                                              0.000s
                                                        0.004s
Cost: 0 %
                  0.298s
                                                                                                                     0.017s
                                                                                                                                                                0.000s
                                       0 of
                                                      3654736 of
                                                                          3654736 of
                                                                                             3654736 of
                     0 of
                                                                                                                   3654736 of
                                                                                                                                                                 0 of
                                  1096420 (0%) Actual Execution Mode
                                                                         3654740 (99%)
                                                                                           3654740 (99%)
                1096420 (0%)
                                                                                                                  3654730 (100%)
                                                                                                                                                               10 (0%)
                                            Estimated Execution Mode
                                                                    Batch
                                                                                                           Actual Execution Mode
                                                                                                                                   Batch
                                                                 Actual Execution Mode
                                                                                         Batch
                                                                                                           Estimated Execution Mode
                                                                                                                                   Batch
                                                                 Estimated Execution Mode
                                                                                         Batch
                                                                                                                                                     Clustered Index Scan (Clustered)
                                                                                                                                                    [ColdRoomTemperatures_Archive].[ix_...
  SOL Server Execution Times:
                                                                                                                                                              Cost: 43 %
     CPU time = 2061 ms, elapsed time = 333 ms.
                                                                                                                                                                0.181s
                                                                                                                                                              3654736 of
                                                                                                                                                             3654740 (99%)
```

```
GO
ALTER DATABASE Statistik SET COMPATIBILITY LEVEL = 150;
GO
SELECT COUNT_BIG(DISTINCT pid) from dbo.A OPTION (USE HINT ('QUERY_OPTIMIZER_COMPATIBILITY_LEVEL_140'));
GO
SELECT COUNT BIG(DISTINCT pid) from dbo.A;
Query 1: Query cost (relative to the batch): 52%
SELECT COUNT BIG (DISTINCT pid) from dbo.A OPTION (USE HINT ('QUERY OPTIMIZER COMPATIBILITY LEVEL 140'))
             Stream Aggregate
                                   Parallelism
                                                     Stream Aggregate
                                                                            Hash Match
                                                                                                   Parallelism
                                                                                                                        Stream Aggregate
                                                                                                                                             Index Scan (NonClustere
                                 (Gather Streams)
                (Aggregate)
                                                       (Aggregate)
                                                                            (Aggregate)
                                                                                               (Repartition Streams)
                                                                                                                          (Aggregate)
                                                                                                                                                    [A].[ix1]
 SELECT
                Cost: 0 %
                                    Cost: 0 %
                                                        Cost: 0 %
                                                                             Cost: 0 %
                                                                                                    Cost: 0 %
                                                                                                                           Cost: 7 %
                                                                                                                                                   Cost: 92 %
Cost: 0 %
                 9.118s
                                     9.118s
                                                         9.118s
                                                                              9.062s
                                                                                                     5.429s
                                                                                                                            5.681s
                                                                                                                                                    3.361s
                  1 of
                                      8 of
                                                          8 of
                                                                            9896294 of
                                                                                                   9896376 of
                                                                                                                          9896376 of
                                                                                                                                                  100000000 of
                 1 (100%)
                                    4 (200%)
                                                        4 (200%)
                                                                           73722 (13423%)
                                                                                                  294888 (3355%)
                                                                                                                         294888 (3355%)
                                                                                                                                                100000000 (100%)
                                                                                        SQL Server Execution Times:
                                                                                            CPU time = 43405 ms, elapsed time = 9136 ms.
Query 2: Query cost (relative to the batch): 48%
SELECT COUNT BIG (DISTINCT pid) from dbo.A
                Parallelism
                                 Hash Match
                                                  Index Scan (NonClustered)
              (Gather Streams)
                                 (Aggregate)
                                                        [A].[ix1]
                                                                                      SQL Server Execution Times:
 SELECT
                                  Cost: 1 %
                Cost: 0 %
                                                        Cost: 99 %
                                                                                         CPU time = 17171 ms, elapsed time = 2619 ms.
Cost: 0 %
                  2.542s
                                   1.442s
                                                         1.345s
                  1 of
                                    1 of
                                                       100000000 of
                 1 (100%)
                                  1 (100%)
                                                     100000000 (100%)
```

**USE** Statistik;

```
SELECT * FROM dbo.M
INNER JOIN dbo.O ON O.Mid = M.Id
INNER JOIN dbo.P ON P.Oid = O.Id
WHERE M.c1 = 2462782;
SELECT * FROM dbo.M
INNER JOIN dbo.O ON O.Mid = M.Id
INNER JOIN dbo.P ON P.Oid = O.Id
Query 1: Query cost (relative to the batch): 91%
         Nested Loops
                            Nested Loops
          (Inner Join)
                            (Inner Join)
 SELECT
          Cost: 0 %
                             Cost: 0 %
Cost: 0 %
           0.000s
                             0.000s
                              34 of
          36658 (0%)
                             36658 (0%)
```

### Regressions?

25x

```
WHERE M.c1 = 2462782 OPTION (USE HINT ('QUERY_OPTIMIZER_COMPATIBILITY_LEVEL_150'));
SELECT * FROM dbo.M INNER JOIN dbo.O ON O.MId = M.Id INNER JOIN dbo.P ON P.OId = O.Id WHE...
                                                                        Nested Loops
                                                                                                    Nested Loops
                                                                                                                               Nested Loops
                                                                        (Inner Join)
                                                                                                    (Inner Join)
                                                                                                                               (Inner Join)
                                                                        Cost: 0 %
                                                                                                    Cost: 0 %
                                                                                                                                Cost: 0 %
                                                                         0.000s
                                                                                                     0.000s
                                                                                                                                 0.000s
                                                                                                                                  4 of
                                                                                                                                           SQL Server Execution Times:
                                                                        136 (19%)
                                                                                                    136 (19%)
                                                                                                                                15 (26%)
                                                                                                                                               CPU time = 0 ms, elapsed time = 2 ms.
                                                                                                                                  П.
                                                                                                                          Index Seek (NonClustered)
                                                                                                                              [0].[UQ_tab0
                                                                                                                                Cost: 0 %
                                                                                                                                 0.000s
                                                                                                                                  26 of
                                                                                                                                136 (19%)
Query 2: Query cost (relative to the batch): 9%
SELECT * FROM dbo.M INNER JOIN dbo.O ON O.MId = M.Id INNER JOIN dbo.P ON P.OId = O.Id WHE...
                                                          Index Seek (NonClustered)
             Hash Match
                             Nested Loops
                                                               [M].[IX_tabM.
             (Inner Join)
                             (Inner Join)
 SELECT
                                                               Cost: 0 %
                                                                                                                                           SQL Server Execution Times:
              Cost: 0 %
                             Cost: 0 %
Cost: 0 %
               0.000s
                              0.000s
                                                                0.000s
                               4 of
                                                                 4 of
                                                                                                                                               CPU time = 47 ms, elapsed time = 50 ms.
             36658 (0%)
                              15 (26%)
                                                                15 (26%)
                                                          Key Lookup (Clustered)
[ M].[PK tabM ...
                                                               Cost: 0 %
                                                                0.000s
                                                                                                                  OPTION (USE HINT('DISALLOW BATCH MODE'));
                                                                 4 of
                                                                15 (26%)
                            Nested Loops
                                                               Nested Loops
                                                                                          Clustered Index Scan (Clustered)
                             (Inner Join)
                                                               (Inner Join)
                                                                                                 [0]. [PK_tab0] [0]
                                                                                                  Cost: 76 %
                             Cost: 0 %
                                                                Cost: 1 %
```

#### Enabling/Disabling Batch Mode

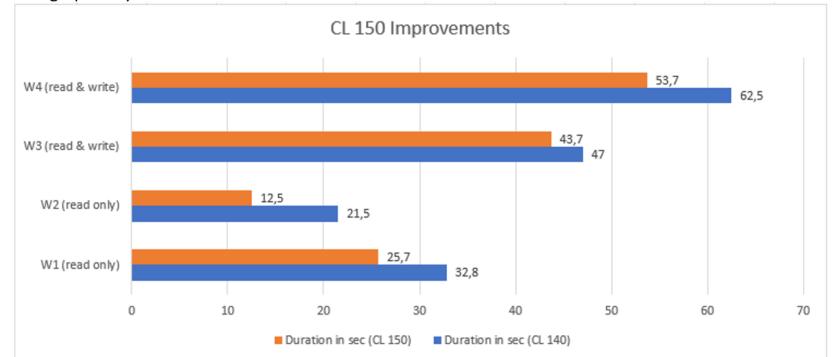
#### Enabling

```
ALTER DATABASE current SET COMPATIBILITY_LEVEL = 150;
ALTER DATABASE SCOPED CONFIGURATION SET BATCH_MODE_ON_ROWSTORE = ON;
OPTION (USE HINT('ALLOW_BATCH_MODE'));
 Disabling
ALTER DATABASE SCOPED CONFIGURATION SET BATCH MODE ON ROWSTORE = OFF;
OPTION (USE HINT('DISALLOW BATCH MODE'));
```

### Our tests - with a pure OLTP workload

Workload	Duration in sec (CL 140)	Duration in sec (CL 150)	Improvement
W1 (read only)	32,8	25,7	21,6 %
W2 (read only)	21,5	12,5	41,8%
W3 (read & write)	47,0	43,7	7 %
W4 (read & write)	62,5	53,7	14,1 %

an overall improvement – 17%



#### Batch Mode on Rowstore - Limitations

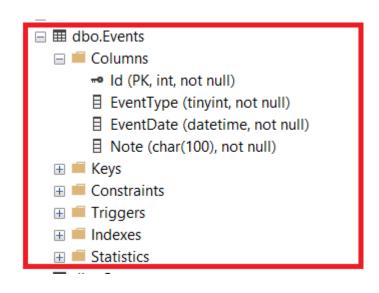
- reading from memory-optimized tables will be always done in row mode
- queries use table has (B)LOB, XML or sparse columns in the SELECT or WHERE clause
- queries using full-text or cursors

#### Batch Mode on Rowstore - Conclusion

- Very promising feature
  - Improvements with no efforts
  - It could be a reason for upgrade for some companies
- First version, probably will not optimize all queries, where you would expect the optimization
- Possible regressions, but you can enable/disable feature at two levels
- It brings benefits for queries with large tables and datasets

- Adjusting memory grant parameter in the execution plan AFTER the plan is generated (after a few query executions)
- Memory is adjusted for a query when
  - It used less than 50% of granted memory
  - Is spilling out to tempdb
- In SQL Server 2017 requires a columnstore index on the affected table

### Memory Grant Feedback - Example



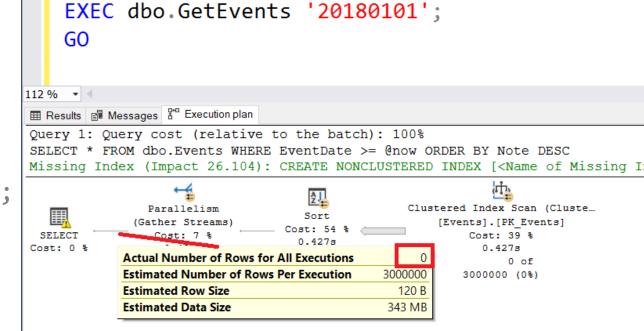
dbo.Events – 10M rows

CREATE OR ALTER PROCEDURE dbo.GetEvents @OrderDate DATETIME

AS BEGIN

**END** 

```
DECLARE @now DATETIME = @OrderDate;
SELECT * FROM dbo.Events
WHERE EventDate >= @now
ORDER BY Note DESC;
```



```
--ensure that the database runs under CL 130 (SQL Server 2016)
ALTER DATABASE TestDb SET COMPATIBILITY_LEVEL = 130;
GO
ALTER DATABASE SCOPED CONFIGURATION CLEAR PROCEDURE CACHE;
GO
EXEC dbo.GetEvents '20180101';
GO 3
    Query 1: Query cost (relative to the batch): 33%
    SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
    Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]. [Events
                                                                                                             SELECT
                      Parallelism
                                                            Clustered Index Scan (Clustered)
                                            Sort
                                                                                                  Cached plan size
                                                                                                                        24 KB
                    (Gather Streams)
                                                                  [Events].[PK Events]
                                          Cost: 54 %
                                                                                                  Estimated Operator Cost
                                                                                                                        0 (0%)
      SELECT
                      Cost: 7 %
                                                                      Cost: 39 %
                                           0.5189
                                                                                                  Degree of Parallelism
    Cost: 0 %
                        0.520s
                                                                        0.518s
                                               0 of
                                                                                                  Estimated Subtree Cost
                                                                                                                       329.792
                           0 of
                                         3000000 (0%)
                                                                                                                       591424
                                                                                                  Memory Grant
                     3000000 (0%)
                                                                     3000000 (0%)
                                                                                                                       3000000
                                                                                                  Estimated Number of Rows
    Query 2: Query cost (relative to the batch): 33%
    SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
    Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]. [Event
                                            A J
                                                                                                             SELECT
                      Parallelism
                                                            Clustered Index Scan (Clustered)
                                            Sort
                    (Gather Streams)
                                                                  [Events].[PK Events]
                                                                                                  Cached plan size
                                                                                                                        24 KB
                                          Cost: 54 %
      SELECT
                      Cost: 7 %
                                                                      Cost: 39 %
                                                                                                  Estimated Operator Cost
                                                                                                                        0 (0%)
                                           0.511s
    Cost: 0 %
                        0.511s
                                                                        0.511s
                                                                                                  Degree of Parallelism
                                               0 of
                                                                           0 of
                           0 of
                                                                                                  Estimated Subtree Cost
                                                                                                                       329,792
                                         3000000 (0%)
                     3000000 (0%)
                                                                     3000000 (0%)
                                                                                                                       591424
                                                                                                  Memory Grant
                                                                                                  Estimated Number of Rows
                                                                                                                       3000000
    Query 3: Query cost (relative to the batch): 33%
    SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
    Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Event.
                                                                                                             SELECT
                      Parallelism
                                                            Clustered Index Scan (Clustered)
                                            Sort
                                                                  [Events].[PK Events]
                    (Gather Streams)
                                          Cost: 54 %
                                                                                                  Cached plan size
                                                                                                                        24 KB
      SELECT
                      Cost: 7 %
                                                                      Cost: 39 %
                                           0.509s
                                                                                                  Estimated Operator Cost
                                                                                                                        0 (0%)
    Cost: 0 %
                        0.514s
                                                                        0.509s
                                               0 of
                                                                                                  Degree of Parallelism
                           0 of
                                                                           0 of
                                         3000000 (0%)
                                                                                                  Estimated Subtree Cost
                                                                                                                       329,792
                     3000000 (0%)
                                                                     3000000 (0%)
                                                                                                                       591424
                                                                                                  Memory Grant
                                                                                                  Estimated Number of Rows
                                                                                                                       3000000
```

#### The same memory grant

```
GO
EXEC dbo.GetEvents '20180101';
GO 3
  Query 1: Query cost (relative to the batch): 33%
  SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
  Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]. [Events]
                                                                                                                  SELECT
     Parallelism
                                                               Clustered Index Scan (Clustered)
                                              Sort
                                                                                                       Cached plan size
                                                                                                                               24 KB
                   (Gather Streams)
                                                                     [Events].[PK Events]
                                           Cost: 54 %
                                                                                                       Estimated Operator Cost
                                                                                                                               0 (0%)
    SELECT
                      Cost: 7 %
                                                                         Cost: 39 %
                                            0.446s
                                                                                                       Degree of Parallelism
  Cost: 0 %
                       0.447s
                                                                           0.446s
                                                0 of
                                                                                                       Estimated Subtree Cost
                                                                                                                              329,792
                           0 of
                                                                               0 of
                                          3000000 (0%)
                    3000000 (0%)
                                                                         3000000 (0%)
                                                                                                       Memory Grant
                                                                                                                              591424
                                                                                                       Estimated Number of Rows
                                                                                                                             3000000
  Query 2: Query cost (relative to the batch): 33%
  SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
  Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]. [Events]
                                                                                                                  SELECT
     Parallelism
                                                               Clustered Index Scan (Clustered)
                                                                                                                               24 KB
                                                                                                       Cached plan size
                                              Sort
                   (Gather Streams)
                                                                     [Events].[PK Events]
                                                                                                                               0 (0%)
                                                                                                       Estimated Operator Cost
    SELECT
                      Cost: 7 %
                                                                         Cost: 39 %
                                             0.416s
                                                                                                       Degree of Parallelism
                                                                           0.416s
  Cost: 0 %
                       0.416s
                                                0 of
                                                                                                       Estimated Subtree Cost
                                                                                                                              329,792
                           0 of
                                                                               0 of
                                          3000000 (0%)
                                                                                                                              591424
                                                                                                       Memory Grant
                     3000000 (0%)
                                                                         3000000 (0%)
                                                                                                       Estimated Number of Rows
                                                                                                                             3000000
  Query 3: Query cost (relative to the batch): 33%
  SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
  Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Events]
                                                                                                                  SELECT
     Parallelism
                                                               Clustered Index Scan (Clustered)
                                                                                                       Cached plan size
                                                                                                                               24 KB
                                              Sort
                                                                     [Events].[PK Events]
                   (Gather Streams)
                                                                                                       Estimated Operator Cost
                                                                                                                               0 (0%)
                                           Cost: 54 %
    SELECT
                      Cost: 7 %
                                                                         Cost: 39 %
                                                                                                       Degree of Parallelism
                                             0.413s
                       0.413s
  Cost: 0 %
                                                                           0.413s
                                                                                                       Estimated Subtree Cost
                                                                                                                              329,792
                                                 0 of
                           0 of
                                                                               0 of
                                                                                                       Memory Grant
                                                                                                                              591424
                                          3000000 (0%)
                     3000000 (0%)
                                                                         3000000 (0%)
                                                                                                       Estimated Number of Rows
```

--ensure that the database runs under CL 140 (SQL Server 2017)

ALTER DATABASE SCOPED CONFIGURATION CLEAR PROCEDURE CACHE;

ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 140;

GO

Batch mode Memory Grant in SQL Server 2017 requires a columnstore index

CREATE NONCLUSTERED COLUMNSTORE INDEX ixc ON dbo.Events(id, EventType, EventDate, Note) WHERE id = -4; GO ALTER DATABASE SCOPED CONFIGURATION CLEAR PROCEDURE CACHE; GO EXEC dbo.GetEvents '20180101'; GO 3 Query 1: Query cost (relative to the batch): 33% SELECT \* FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC Missing Index (Impact 41.6303): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]. [Events] ([Ev SELECT Clustered Index Scan (Clustered) Parallelism Sort Cached plan size 24 KB (Gather Streams) [Events].[PK Events] Cost: 27 % **Estimated Operator Cost** 0 (0%) SELECT Cost: 62 % Cost: 11 % 0.008s Degree of Parallelism Cost: 0 % 0.433s 0.432s 0 of **Estimated Subtree Cost** 206,811 0 of 0 of 670408 3000000 (0%) Memory Grant 3000000 (0%) 3000000 (0%) **Estimated Number of Rows** 3000000 Query 2: Query cost (relative to the batch): 33% SELECT \* FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC Missing Index (Impact 41.6303): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Events] ([Ev SELECT Parallelism Clustered Index Scan (Clustered) Sort (Gather Streams) [Events].[PK Events] Cached plan size 24 KB Cost: 27 % SELECT Cost: 62 % Cost: 11 % **Estimated Operator Cost** 0 (0%) 0.006s 0.424s 0.423s Cost: 0 % Degree of Parallelism 0 of 0 of 0 of Estimated Subtree Cost 206.811 3000000 (0%) 3000000 (0%) 3000000 (0%) 2440 Memory Grant Estimated Number of Rows 3000000 Query 3: Query cost (relative to the batch): 33% SELECT \* FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC Missing Index (Impact 41.6303): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Events] ([Ev SELECT Parallelism Clustered Index Scan (Clustered)

[Events].[PK Events]

Cost: 62 %

0.428s

3000000 (0%)

0 of

Sort

Cost: 27 %

0.006s

3000000 (0%)

0 of

(Gather Streams)

Cost: 11 %

0.430s

3000000 (0%)

0 of

SELECT

Cost: 0 %

Cached plan size

Memory Grant

**Estimated Operator Cost** 

Degree of Parallelism

Estimated Subtree Cost

**Estimated Number of Rows** 

24 KB

0 (0%)

206.811

3000000

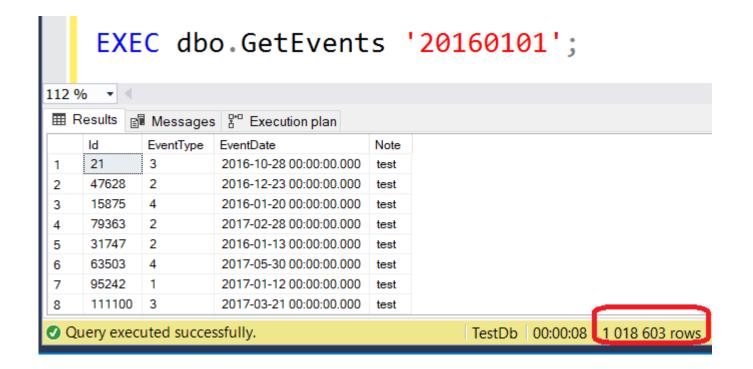
2440

```
ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 150;
G<sub>0</sub>
--remove the columnstore index
DROP INDEX ixc ON dbo.Events;
G<sub>0</sub>
ALTER DATABASE SCOPED CONFIGURATION CLEAR PROCEDURE_CACHE;
G<sub>0</sub>
EXEC dbo.GetEvents '20180101';
GO 3
Query 1: Query cost (relative to the batch): 33%
SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo]. [Events] ([Ever
                                        A J
                                                                                                             SELECT
                 Parallelism
                                                       Clustered Index Scan (Clustered)
                                       Sort
               (Gather Streams)
                                                             [Events].[PK Events]
                                                                                                  Cached plan size
                                                                                                                         24 KB
                                     Cost: 54 %
 SELECT
                                                                 Cost: 39 %
                                                                                                                         0 (0%)
                                                                                                  Estimated Operator Cost
                                       0.516s
Cost: 0 %
                   0.517s
                                                                   0.516s
                                          0 of
                                                                                                  Degree of Parallelism
                      0 of
                                                                      0 of
                                                                                                                        329,792
                                    3000000 (0%)
                                                                                                  Estimated Subtree Cost
                 3000000 (0%)
                                                                3000000 (0%)
                                                                                                                        661760
                                                                                                  Memory Grant
                                                                                                  Estimated Number of Rows
Query 2: Query cost (relative to the batch): 33%
SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Events] ([Ever
                                                                                                            SELECT
                 Parallelism
                                                       Clustered Index Scan (Clustered)
                                       Sort
                                                                                                 Cached plan size
                                                                                                                         24 KB
               (Gather Streams)
                                                             [Events].[PK Events]
                                     Cost: 54 %
                                                                                                 Estimated Operator Cost
                                                                                                                         0 (0%)
 SELECT
                  Cost: 7 %
                                                                 Cost: 39 %
                                      0.493s
                                                                                                 Degree of Parallelism
Cost: 0 %
                   0.494s
                                                                   0.493s
                                          0 of
                                                                                                 Estimated Subtree Cost
                                                                                                                        329,792
                      0 of
                                                                      0 of
                                    3000000 (0%)
                 3000000 (0%)
                                                                3000000 (0%)
                                                                                                                          5888
                                                                                                 Memory Grant
                                                                                                 Estimated Number of Rows
                                                                                                                       3000000
Query 3: Query cost (relative to the batch): 33%
SELECT * FROM dbo.Events WHERE EventDate >= @now ORDER BY Note DESC
Missing Index (Impact 26.1061): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Events] ([Ever
                                                                                                             SELECT
                                        A J
                 Parallelism
                                                       Clustered Index Scan (Clustered)
                                                                                                 Cached plan size
                                                                                                                         24 KB
                                        Sort
               (Gather Streams)
                                                             [Events].[PK Events]
                                                                                                 Estimated Operator Cost
                                                                                                                         0 (0%)
                                     Cost: 54 %
 SELECT
                  Cost: 7 %
                                                                 Cost: 39 %
                                                                                                 Degree of Parallelism
                                       0.499s
Cost: 0 %
                   0.500s
                                                                   0.499s
                                                                                                 Estimated Subtree Cost
                                                                                                                        329.792
                                          0 of
                      0 of
                                                                      0 of
                                                                                                                          5888
                                    3000000 (0%)
                                                                                                 Memory Grant
                3000000 (0%)
                                                                3000000 (0%)
                                                                                                 Estimated Number of Rows
```

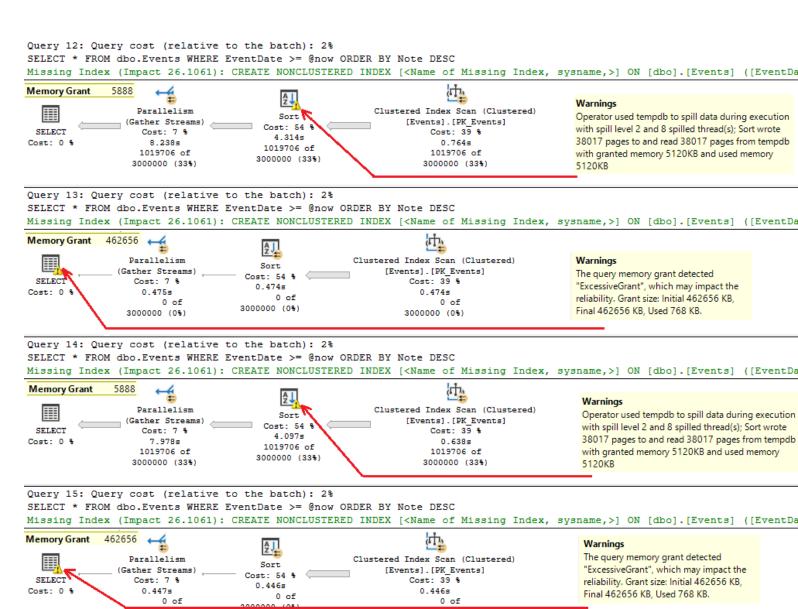
Row mode Memory Grant in SQL Server 2019 No columnstore index required

- New plan attributes in the XML plan
- It works with cached plans and memory grants > 1MB
- It does not work with OPTION (RECOMPILE)
- It is not persisted if the plan is removed from cache
- In SQL Server 2019, it works in both Batch and Row Mode
- If memory grant memory values oscillate, the feature is disabled

```
EXEC dbo.GetEvents '20180101';
EXEC dbo.GetEvents '20160101';
```



```
EXEC dbo.GetEvents '20180101';
EXEC dbo.GetEvents '20160101';
GO 30
```



• If memory grant memory values oscillate, the feature is disabled

```
EXEC dbo.GetEvents '20190901';
EXEC dbo.GetEvents '20160901';
GO 30
                                  Displaying 1 Events
                                                                              timestamp
                                        name
                                       memory grant feedback loop disabled
                                                                             2019-11-25 00:09:48.6804572
                                  Event:memory grant feedback loop disabled (2019-11-25 00:09:48.6804572)
                                   Details
                                    Field
                                                           Value
                                    sql_text
                                                           EXEC dbo.GetEvents '20190901'; EXEC dbo.GetEvents '20160901';
                                    total_execution_count
                                                           33
                                    total_update_count
                                                           31
```

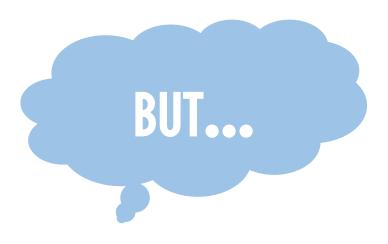
### Memory Grant Feedback - Conclusion

- Very useful feature
- In SQL Server 2019 works in both modes
- Will affect ALL queries having an operator that requires memory
- Almost no measurable overhead (\*)

• (\*) in all my tests

#### Scalar UDFs in SQL Server

- Code reuse, encapsulation and modularity
- Complex business rules or computations
- Single place change
- Written once, invoke from many modules
- Reduce network traffic



#### Scalar UDFs in SQL Server

Why do SQL Server Scalar-valued functions get slower?

# Refactor SQL Server scalar UDF to inline TVF to improve performance

Why SQL Server scalar functions are bad?

T-SQL Best Practices - Don't Use Scalar Value Functions in Column .

Are SQL Server Functions Dragging Your

Query Down?

SQL functions rarely perform well.

#### Scalar UDFs in SQL Server

- They are very slow
  - Iterative invocation
  - Overhead for invoking function once per row
- No cross-statement optimization
- Only serial execution plans possible



# Scalar UDF Inlining

#### Froid: Optimization of Imperative Programs in a Relational Database

Karthik Ramachandra Microsoft Gray Systems Lab

karam@microsoft.com

Alan Halverson Microsoft Gray Systems Lab

alanhal@microsoft.com

Kwanghyun Park Microsoft Gray Systems Lab

kwpark@microsoft.com

César Galindo-Legaria Microsoft

cesarg@microsoft.com

K. Venkatesh Emani<sup>3</sup>

venkateshek@cse.iitb.ac.in

Conor Cunningham Microsoft

conorc@microsoft.com

#### **ABSTRACT**

For decades, RDBMSs have supported declarative SQL as well as imperative functions and procedures as ways for users to express data processing tasks. While the evaluation of declarative SQL has received a lot of attention resulting in highly sophisticated techniques, the evaluation of imperative programs has remained naïve and highly inefficient. Imperative programs offer several benefits over SQL and hence are

expressing intent has on one hand provided high-level abstractions for data processing, while on the other hand, has enabled the growth of sophisticated query evaluation techniques and highly efficient ways to process data.

Despite the expressive power of declarative SQL, almost all RDBMSs support procedural extensions that allow users to write programs in various languages (such as Transact-SQL, C#, Java and R) using imperative constructs such as variable assignments, conditional branching, and loops

### Scalar UDF Inlining

- Goal improve queries where scalar UDFs are problem
- Scalar UDF Inlining feature (Froid framework):
  - Transforms scalar UDF into relational expressions or subqueries (IF => CASE WHEN)
  - Embeds them in the calling query by using APPLY operator
  - Optimize expressions or subqueries
- Result:
  - Performance improved (more efficient plan)
  - Execution plan could be parallel

```
DECLARE @val VARCHAR(10);
DECLARE @a INT = 2000;

IF @a > 1000
        SET @val = 'HIGH';
ELSE IF @a > 500
        SET @val = 'MEDIUM'
ELSE
        SET @val = 'LOW'

SELECT @val;
```

```
SELECT q5.v
FROM
         (SELECT 2000 AS a) AS q1
         OUTER APPLY
                  (SELECT CASE WHEN q1.a > 1000 THEN 'HIGH' END AS val)
AS q2
         OUTER APPLY
         (SELECT CASE WHEN q1.a > 500 THEN 'HIGH' END AS val) AS q3
         OUTER APPLY
                  (SELECT 'LOW' AS val) AS q4
         OUTER APPLY
                  (SELECT CASE WHEN q2.val IS NOT NULL
            THEN q2.val
            WHEN q3.val IS NOT NULL
            THEN q3.val
            ELSE q4.val
        END v) AS q5
);
```

```
DECLARE @val VARCHAR(10);

DECLARE @a INT = 2000;

IF @a > 1000
        SET @val = 'HIGH';

ELSE IF @a > 500
        SET @val = 'MEDIUM'

ELSE
        SET @val = 'LOW'

SELECT @val;
```

```
SELECT q5.v
FROM
         (SELECT 2000 AS a) AS q1
         OUTER APPLY
                  (SELECT CASE WHEN q1.a > 1000 THEN 'HIGH' END AS val)
AS q2
         OUTER APPLY
         (SELECT CASE WHEN q1.a > 500 THEN 'HIGH' END AS val) AS q3
         OUTER APPLY
                  (SELECT 'LOW' AS val) AS q4
         OUTER APPLY
                  (SELECT CASE WHEN q2.val IS NOT NULL
            THEN q2.val
            WHEN q3.val IS NOT NULL
            THEN q3.val
            ELSE q4.val
        END v) AS q5
);
```

```
DECLARE @val VARCHAR(10);
DECLARE @a INT = 2000;

IF @a > 1000
SET @val = 'HIGH';
ELSE IF @a > 500
SET @val = 'MEDIUM'
ELSE
SET @val = 'LOW'

SELECT @val;
```

```
SELECT q5.v
FROM
         (SELECT 2000 AS a) AS q1
         OUTER APPLY
                  (SELECT CASE WHEN q1.a > 1000 THEN 'HIGH' END AS val)
AS q2
         (SELECT CASE WHEN q1.a > 500 THEN 'HIGH' END AS val) AS q3
                  (SELECT 'LOW' AS val) AS q4
         OUTER APPLY
                  (SELECT CASE WHEN q2.val IS NOT NULL
            THEN q2.val
            WHEN q3.val IS NOT NULL
            THEN q3.val
            ELSE q4.val
        END v) AS q5
);
```

```
DECLARE @val VARCHAR(10);
DECLARE @a INT = 2000;

IF @a > 1000
SET @val = 'HIGH';
ELSE IF @a > 500
SET @val = 'MEDIUM'
ELSE
SET @val = 'LOW'

SELECT @val;
```

```
SELECT q5.v
FROM
         (SELECT 2000 AS a) AS q1
         OUTER APPLY
                  (SELECT CASE WHEN q1.a > 1000 THEN 'HIGH' END AS val)
AS q2
         (SELECT CASE WHEN q1.a > 500 THEN 'HIGH' END AS val) AS q3
                  (SELECT 'LOW' AS val) AS q4
         OUTER APPLY
                  (SELECT CASE WHEN q2.val IS NOT NULL
            THEN q2.val
            WHEN q3.val IS NOT NULL
            THEN q3.val
            ELSE q4.val
        END v) AS q5
);
```

```
DECLARE @val VARCHAR(10);
DECLARE @a INT = 2000;

IF @a > 1000
SET @val = 'HIGH';
ELSE IF @a > 500
SET @val = 'MEDIUM'
ELSE
SET @val = 'LOW'

SELECT @val;
```

```
SELECT q5.v
         (SELECT 2000 AS a) AS q1
         OUTER APPLY
                  (SELECT CASE WHEN q1.a > 1000 THEN 'HIGH' END AS val)
AS q2
         (SELECT CASE WHEN q1.a > 500 THEN 'HIGH' END AS val) AS q3
                  (SELECT 'LOW' AS val) AS q4
         OUTER APPLY
                  (SELECT CASE WHEN q2.val IS NOT NULL
            THEN q2.val
            WHEN q3.val IS NOT NULL
            THEN q3.val
            ELSE q4.val
        END v) AS q5
);
```

```
CREATE OR ALTER FUNCTION dbo.GetOrderItemStatus(
@Quantity INT, @UnitPrice DECIMAL(10,2))
RETURNS VARCHAR(20)
AS
                                                   SET STATISTICS TIME ON;
BEGIN
   DECLARE @Ret VARCHAR(20) = '';
                                                   ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 140;
   DECLARE @Amount DECIMAL(10,2) =
                                                   GO
@Quantity * @UnitPrice;
                                                   SELECT *, dbo.GetOrderItemStatus(Quantity,UnitPrice) ItemStatus FROM dbo.OrderDetails;
   IF @Amount > 1000
       SET @Ret = 'TOP 1000'
                                                   ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 150;
   ELSE IF @Amount > 500
       SET @Ret = 'TOP 500'
                                                   SELECT *, dbo.GetOrderItemStatus(Quantity,UnitPrice) ItemStatus FROM dbo.OrderDetails;
   RETURN @Ret;
END
           Query 1: Query cost (relative to the batch): 50%
           SELECT *, dbo.GetOrderItemStatus(Quantity,UnitPrice) ItemStatus FROM dbo.OrderDetails
                                Clustered Index Scan (Cluste ...
                           Compute Scalar
                                                [OrderDetails].[PK OrderDeta...
                             Cost: 2 %
            SELECT
                                                         Cost: 98 %
                              23.350s
           Cost: 0 %
```

```
1.2663
                                             _{3000000} of CPU time = 18656 ms, elapsed time = 24306 ms.
                  3000000 of
                3000000 (100%)
                                            3000000 (100%)
Query 2: Query cost (relative to the batch): 50%
SELECT *, dbo.GetOrderItemStatus(Quantity,UnitPrice) ItemStatus FROM dbo.OrderDetails
                                    Clustered Index Scan (Cluste...
                                    [OrderDetails].[PK OrderDeta...
                Compute Scalar
                                             Cost: 98 %
                                                         CPU time = 2047 \text{ ms}, elapsed time = 2075 \text{ ms}.
                  Cost: 2 %
                                               0.697s
Cost: 0 %
                                             3000000 of
                                            3000000 (100%)
```

```
CREATE OR ALTER FUNCTION dbo.GetOrderItemStatus(
@Quantity INT, @UnitPrice DECIMAL(10,2))
RETURNS VARCHAR (20)
AS
                                                             SET STATISTICS TIME ON;
BEGIN
                                                             GO
    DECLARE @Ret VARCHAR(20) = '';
                                                             ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 140;
    DECLARE @Amount DECIMAL(10,2) =
                                                             GO
@Quantity * @UnitPrice;
                                                             SELECT *, dbo.GetOrderItemStatus(Ouantity,UnitPrice) ItemStatus FROM dbo.OrderDetails;
    IF @Amount > 1000
         SET @Ret = 'TOP 1000'
                                                             ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 150;
    ELSE IF @Amount > 500
         SET @Ret = 'TOP 500'
                                                             SELECT *, dbo.GetOrderItemStatus(Ouantity,UnitPrice) ItemStatus FROM dbo.OrderDetails;
    RETURN @Ret;
END
 Ouery 1: Query cost (relative to the batch): 50%

☐ Defined Values

                                                                                                                                                    X
 SELECT *, dbo.GetOrderItemStatus(Quantity,UnitPrice)
                                                         [Expr1002] = Scalar Operator([TestDb].[dbo].[GetOrderItemStatus]([TestDb].[dbo].
                          [OrderDetails].[Quantity].[TestDb].[dbo].[OrderDetails].[UnitPrice]))
                                           Clustered Index Scan (Clus
                    Compute Scalar
                                           [OrderDetails].[PK OrderDe
                      Cost: 2 %
   SELECT
                                                     Cost: 98 %
                       23.350s
 Cost: 0 %
                                                       1.266s
                      3000000 of
                                                     3000000 of
                    3000000 (100%)
                                                   3000000 (100%)
                                                                                                                                             Close
 Query 2: Query cost (relative to the
                                                 P□ Defined Values
 SELECT *, dbo.GetOrderItemStatus(Qu
                                                 [Expr1010] = Scalar Operator(CONVERT_IMPLICIT(varchar(20), CASE WHEN CASE WHEN CONVERT_IMPLICIT(decimal
                                                 (10,2),CONVERT_IMPLICIT(decimal(10,0),[TestDb].[dbo].[OrderDetails].[Quantity],0)*[TestDb].[dbo].[OrderDetails].[UnitPrice],0)>(1000.00)
                                           Clus
                                                 THEN (1) ELSE (0) END = (0) AND CASE WHEN CONVERT_IMPLICIT(decimal(10,2),CONVERT_IMPLICIT(decimal(10,0),[TestDb].[dbo].
                          [Ord
                                                 [OrderDetails].[Quantity],0)*[TestDb].[dbo].[OrderDetails].[UnitPrice],0)>(500.00) THEN (1) ELSE (0) END = (1) THEN 'TOP 500' ELSE CASE
                    Compute Scalar
                                                 WHEN CASE WHEN CONVERT_IMPLICIT(decimal(10,2),CONVERT_IMPLICIT(decimal(10,0),[TestDb].[dbo].[OrderDetails].[Quantity],0)*
 Cost: 0 %
                      Cost: 2 %
                                                 [TestDb].[dbo].[OrderDetails].[UnitPrice].0)>(1000.00) THEN (1) ELSE (0) END = (1) THEN 'TOP 1000' ELSE "END END.0))
                                                                                                                                                       Close
```

```
CREATE OR ALTER FUNCTION dbo.GetOrderCnt (@CustomerId INT)
RETURNS INT
AS
BEGIN
     DECLARE @Cnt INT;
     SELECT @Cnt = COUNT(*) FROM dbo.Orders WHERE CustomerId = @CustomerId;
     RETURN @Cnt;
END
 ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 140;
 G<sub>0</sub>
 SELECT * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25;
 ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 150;
 G<sub>0</sub>
 SELECT * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25;
 Query 1: Query cost (relative to the batch): 1%
 SELECT * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId)>25
                                        ıψ
                 T
                             Clustered Index Scan (Cluste...
                Filter
                               [Customers].[PK Customers]
               Cost: 4 %
  SELECT
                                     Cost: 96 %
                8.970s
 Cost: 0 %
                                       0.140s
                  2 of
                                     100000 of
                                                                                       CPU time = 8610 ms, elapsed time = 8971 ms.
              30000 (0%)
                                    100000 (100%)
 Query 2: Query cost (relative to the batch): 99%
 SELECT * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId)>25
                                     T.
                                                                  Clustered Index Scan (Cluste...
                 Parallelism
                                                 Nested Loops
                                   Filter
               (Gather Streams)
                                                 (Inner Join)
                                                                    [Customers].[PK Customers]
                                  Cost: 0 %
  SELECT
                 Cost: 0 %
                                                  Cost: 0 %
                                                                           Cost: 1 %
                                   0.267s
 Cost: 0 %
                  0.267s
                                                   0.260s
                                                                            0.015s
                                      2 of
                     2 of
                                                  100000 of
                                                                          100000 of
                                                                                         CPU time = 1860 ms, elapsed time = 271 ms.
                                 30000 (0%)
                                                100000 (100%)
                 30000 (0%)
                                                                         100000 (100%)
                                                                                                BΈ
                                                                                           Stream Aggregate
                                                                                                                                Stream Aggregate
                                                                                                                                                     Index Seek (NonClustered)
                                                                                             (Aggregate)
                                                                                                                                   (Aggregate)
                                                                                                                                                         [Orders].[ix1]
                                                                        Compute Scalar
                                                                                              Cost: 0 %
                                                                                                              Compute Scalar
                                                                                                                                   Cost: 2 %
                                                                                                                                                           Cost: 96 %
                                                                                               0.232s
                                                                           Cost: 0 %
                                                                                                                Cost: 0 %
                                                                                                                                    0.220s
                                                                                                                                                            0.202s
                                                                                              100000 of
                                                                                                                                   100000 of
                                                                                                                                                           1000000 of
                                                                                            100000 (100%)
                                                                                                                                 100000 (100%)
                                                                                                                                                         1001640 (99%)
```

```
CREATE OR ALTER FUNCTION dbo.GetDaysFromLastOrder(@CustomerId INT)
RETURNS INT
AS
BEGIN
   DECLARE @Days INT;
   DECLARE @LastOrder DATETIME;
   SET @LastOrder = (SELECT TOP (1) OrderDate FROM dbo.Orders WHERE CustomerId = @CustomerId ORDER BY OrderDate DESC);
   SELECT @Days = DATEDIFF(day, @LastOrder, GETDATE());
   RETURN @Days;
END
                                                                             Scalar UDF Inline does not work
                                                                             with GETDATE() function
  SELECT * FROM dbo.Customers WHERE dbo.GetDaysFromLastOrder(CustomerId) > 365;
Query 1: Query cost (relative to the batch): 50%
SELECT * FROM dbo.Customers WHERE dbo.GetDaysFromLastOrder(CustomerId) > 365
                                             ĺψ,
                                Clustered Index Scan (Cluste...
                  Filter
                                  [Customers].[PK Customers]
                                          Cost: 98 %
                  6.553s
Cost: 0 %
                                            0.002s
                 1000 of
                                                          CPU time = 12531 ms, elapsed time = 12648 ms.
                                           1000 of
                300 (333%)
                                         1000 (100%)
Query 2: Query cost (relative to the batch): 50%
SELECT * FROM dbo.Customers WHERE dbo.GetDaysFromLastOrder(CustomerId) > 365
                   Y
                                 Clustered Index Scan (Cluste...
                  Filter
                                  [Customers].[PK Customers]
                Cost: 2
                                          Cost: 98 %
                  7.212s
                                                           CPU time = 12485 ms, elapsed time = 13079 ms.
Cost: 0 %
                                           0.003s
                 1000 of
                                           1000 of
                300 (333%)
                                         1000 (100%)
```

### Scalar UDF Inlining

- Not all scalar UDFs can be inlined.
- Check whether a function can be inlined:

```
SELECT CONCAT(SCHEMA_NAME(o.schema_id),'.',o.name), is_inlineable
FROM sys.sql_modules m INNER JOIN sys.objects o ON o.object_id = m.object_id
WHERE o.type = 'FN';
```

- is\_inlineable = 1 does not imply that it will always be inlined
- Decision is made when the query referencing a scalar UDF is compiled

#### Scalar UDF Inlining - Limitations

- UDF does not invoke any intrinsic function that is either timedependent or has side effects such as GETDATE() or NEWSEQUENTIALID
- The UDF does not reference table variables, table-valued parameters or user-defined types
- UDF is not natively compiled (interop is supported)
- UDF is not used in a computed column or a check constraint definition
- The UDF is not a partition function
- Full list of limitations: https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/scalar-udf-inlining?view=sql-server-ver15

### Regressions

```
ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 140;
GO
SELECT TOP (10) * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25;
GO
ALTER DATABASE TestDb SET COMPATIBILITY LEVEL = 150;
GO
SELECT TOP (10) * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25;
Query 1: Query cost (relative to the batch): 0%
SELECT TOP (10) *, dbo.GetOrderCnt(CustomerId) FROM dbo.Customers
                                                       įψ,
                                            Clustered Index Scan (Cluste...
             Compute Scalar
                                 Top
                                             [Customers].[PK Customers]
                               Cost: 0
 SELECT
                                                   Cost: 100 %
                                0.000s
                                                                                CPU time = 5922 \text{ ms}, elapsed time = 873 \text{ ms}.
                                                     0.000s
Cost: 0 %
                 10 of
                                10 of
                                                      10 of
               10 (100%)
                               10 (100%)
                                                    10 (100%)
Query 2: Query cost (relative to the batch): 100%
SELECT TOP (10) *, dbo.GetOrderCnt(CustomerId) FROM dbo.Customers
Missing Index (Impact 99.8008): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Orders] ([CustomerID])
                                 Nested Loops
                                                            Clustered Index Scan (Cluste..
                                 Top
                                            (Inner Join)
                                                              [Customers].[PK Customers]
                               Cost: 0 %
 SELECT
             Compute Scalar
                                             Cost: 0 %
                                                                     Cost: 0 %
                                                                                  CPU time = 4735 ms, elapsed time = 4753 ms
                                2.557s
Cost: 0 %
               Cost: 0 %
                                              2.557s
                                                                      0.000s
                                10 of
                                               10 of
                                                                      10 of
                              10 (100%)
                                                                     10 (100%)
                                             10 (100%)
                                                                                          35
                                                                                                                                             Clustered Index Scan (Cluste...
                                                                                    Stream Aggregate
                                                                                                                         Stream Aggregate
                                                                                       (Aggregate)
                                                                                                                           (Aggregate)
                                                                                                                                                 [Orders].[PK Orders]
                                                                                                                                                     Cost: 75 %
                                                                   Compute Scalar
                                                                                       Cost: 0 %
                                                                                                       Compute Scalar
                                                                                                                            Cost: 25 %
                                                                     Cost: 0 %
                                                                                        2.557s
                                                                                                         Cost: 0 %
                                                                                                                             2.557s
                                                                                                                                                       2.556s
                                                                                         10 of
                                                                                                                             10 of
                                                                                                                                                      20292 of
                                                                                        11 (90%)
                                                                                                                            11 (90%)
                                                                                                                                                     21980 (92%)
```

#### Regressions

```
ALTER DATABASE TestDb SET COMPATIBILITY_LEVEL = 140;

GO
SELECT TOP (10) * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25;

GO
ALTER DATABASE TestDb SET COMPATIBILITY_LEVEL = 150;

GO
SELECT TOP (10) * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25;
```

#### Solution

```
SELECT TOP (10) * FROM dbo.Customers WHERE dbo.GetOrderCnt(CustomerId) > 25
OPTION (USE HINT('DISABLE_TSQL_SCALAR_UDF_INLINING'));
```

### Scalar UDF Inlining - Configuration

• Enable

```
ALTER DATABASE current SET COMPATIBILITY_LEVEL = 150;

ALTER DATABASE SCOPED CONFIGURATION SET TSQL_SCALAR_UDF_INLINING = ON;

CREATE OR ALTER FUNCTION dbo.getMaxOrderDate(@CustID INT) RETURNS DATETIME WITH INLINE = ON
```

Disable

```
ALTER DATABASE SCOPED CONFIGURATION SET TSQL_SCALAR_UDF_INLINING = OFF;

OPTION (USE HINT('DISABLE_TSQL_SCALAR_UDF_INLINING'));

CREATE OR ALTER FUNCTION dbo.getMaxOrderDate(@CustID INT) RETURNS DATETIME WITH INLINE = OFF
```

### Scalar UDF Inlining - Conclusion

- Very promising feature
  - Improvements with no efforts
  - Part of the Standard Edition
- Many limitations (GETDATE(), table variables...)
- Very useful for small and medium companies (not enough people to rewrite UDFs) 3rd party tools
- it's a first version
- As far I know, it is still not available in Azure

#### Problems with Table Variables

```
DECLARE @t TABLE(id INT)
INSERT INTO @t SELECT message_id FROM sys.messages;
SELECT * FROM @t;
 Query 2: Query cost (relative to the batch): 2%
 SELECT * FROM @t
                    Table Scan
                       [@t]
                    Cost: 100 %
 Cost: 0
                     1 (30940800%)
```

Actual Number of Rows309408Number of Rows Read309408Estimated Number of Rows1Estimated Row Size11 BEstimated Data Size11 B

- Inappropriate operators in the execution plan
- Insufficient memory grants

#### Problems with Table Variables

- Queries using table variables with large number of rows (> 10K) can have an inefficient plan
- Inappropriate operators in the execution plan
  - mostly Nested Loop Join instead of Hash Join
- Insufficient memory grants
  - Number of processing rows is usually underestimated => less Memory Grant reserved for the query => spills to tempdb

### Table Variable Deferred Compilation

```
DECLARE @T AS TABLE (ProductID INT);
                                             to
INSERT INTO @T SELECT ProductID
                                             t<sub>1</sub>
FROM Production Product
                                                        SQL Server 2017 and all previous versions
WHERE ProductLine IS NOT NULL;
                                                           • Content of the table variable unknown at the compile time =>
                                                             cardinality 1
SELECT * FROM @T t
                                             t<sub>2</sub>
INNER JOIN Sales Sales Order Detail od
ON t.ProductID = od.ProductID
                                                        SQL Server 2019
INNER JOIN Sales Sales Order Header h
ON h.SalesOrderID = od.SalesOrderID

    Content of the table variable known at the compile time =>

                                                             cardinality = actual number of rows
ORDER BY od.UnitPrice DESC;
```

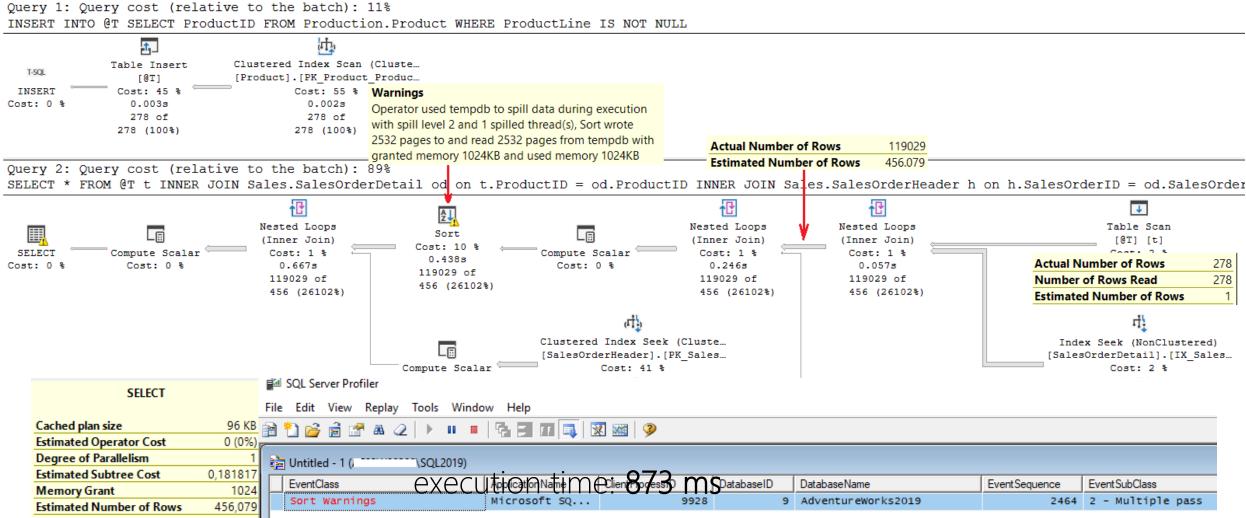
DECLARE @T AS TABLE (ProductID INT);
INSERT INTO @T SELECT ProductID FROM Production.Product WHERE ProductLine IS NOT NULL;

SELECT \* FROM @T t
INNER JOIN Sales.SalesOrderDetail od on t.ProductID = od.ProductID
INNER JOIN Sales.SalesOrderHeader h on h.SalesOrderID = od.SalesOrderID
ORDER BY od.UnitPrice DESC;

: Query cost (relative to the batch): 11%

SQL Server 2017

456 (26102%)



```
DECLARE @T AS TABLE (ProductID INT);
                           INSERT INTO @T SELECT ProductID FROM Production. Product WHERE ProductLine IS NOT NULL;
                           SELECT * FROM @T t
                                                                                                                                                                                                                                                                                                                                                      SOL Server 2019
                           INNER JOIN Sales.SalesOrderDetail od on t.ProductID = od.ProductID
                           INNER JOIN Sales.SalesOrderHeader h on h.SalesOrderID = od.SalesOrderID
                           ORDER BY od. UnitPrice DESC;
Query 2: Query cost (relative to the batch): 100%
SELECT * FROM @T t INNER JOIN Sales.SalesOrderDetail od on t.ProductID = od.ProductID INNER JOIN Sales.SalesOrderHeader h on h.SalesOrderID = od.SalesOrderDetail od on t.ProductID = od.ProductID = od.P
Missing Index (Impact 18.7112): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Sales].[SalesOrderDetail] ([ProductID]) INCLUDE ([CarrierTrackingNumber],[Orde...
                                    Parallelism
                                                                                                                    Hash Match
                                                                                                                                                                                                                                                          Parallelism
                                                                                                                                                                                                                                                                                                          Clustered Index Scan (Clustered)
                                                                                   Sort
                                                                                                                                                                                                                                                                                                       [SalesOrderHeader].[PK_SalesOrderHe...
                                 (Gather Streams)
                                                                                                                   (Inner Join)
                                                                                                                                                                                                                                                   (Repartition Streams)
                                                                              Cost: 26 %
                                     Cost: 24 %
                                                                                                                    Cost: 16 %
                                                                                                                                                                Compute Scalar
                                                                                                                                                                                                         Compute Scalar
                                                                                                                                                                                                                                                            Cost: 3 %
                                                                                                                                                                                                                                                                                                                              Cost: 6 %
                                                                                 0.1725
                                                                                                                                                                    Cost: 0 %
                                        0.2635
                                                                                                                      0.1085
                                                                                                                                                                                                             Cost: 0 %
                                                                                                                                                                                                                                                              0.016s
                                                                                                                                                                                                                                                                                                                                0.0125
                                                                               119029 of
                                      119029 of
                                                                                                                                                                                                                                                             31465 of
                                                                                                                                                                                                                                                                                                                              31465 of
                                                                                                                    119029 of
                                                                             126790 (93%)
                                                                                                                                                                                                                                                         31465 (100%)
                                                                                                                                                                                                                                                                                                                           31465 (100%)
                                   126790 (93%)
                                                                                                                  126790 (93%)
                                                                                                                                                                                                                                                                                                                                         Actual Number of Rows
                                                                                                                                                                                                                                                                                                                                                                                                     278
                                                                   Actual Number of Rows
                                                                                                                          119029
                                                                                                                                                                                                                                                                                                                                                                                                     278
                                                                                                                                                                                                                                                                                                                                         Number of Rows Read
                                                                   Estimated Number of Rows
                                                                                                                          126790
                                                                                                                                                                   Parallelism
                                                                                                                                                                                                                                                               Parallelism
                                                                                                                                                                                                                 Hash Match
                                                                                                                                                                                                                                                                                                          Table Scan
                                                                                                                                                                                                                                                                                                                                                                                                     278
                                                                                                                                                                                                                                                                                                                                         Estimated Number of Rows
                                                                                                                                                           (Repartition Streams)
                                                                                                                                                                                                                (Inner Join)
                                                                                                                                                                                                                                                       (Repartition Streams)
                                                                                                                                                                                                                                                                                                             [@T] [t]
                                                                                                                                                                                                                                                                                                                                                                                                    11 B
                                                                                                                                                                                                                                                                                                                                         Estimated Row Size
                                                                                                                                                                    Cost: 5 %
                                                                                                                                                                                                                  Cost: 4 %
                                                                                                                                                                                                                                                                 Cost: 0 %
                                                                                                                                                                                                                                                                                                            Cost: 0 %
                                                                                                                                                                      0.0735
                                                                                                                                                                                                                    0.0885
                                                                                                                                                                                                                                                                   0.0015
                                                                                                                                                                                                                                                                                                              0.0005
```

SELECT **Estimated Data Size** 3058 B 119029 of 119029 of 278 of 278 of 126790 (93%) 126790 (93%) 278 (100%) 278 (100%) Cached plan size 208 KB **Estimated Operator Cost** 0 (0%) Degree of Parallelism Parallelism Clustered Index Scan (Clustered) Estimated Subtree Cost 9,32107 (Repartition Streams) [SalesOrderDetail].[PK\_SalesOrderDe Compute Scalar Compute Scalar Cost: 4 % Cost: 10 % Memory Grant 140864 Cost: 0 % Cost: 0 % 0.0825 0.0175 Estimated Number of Rows 126790 121317 of 121317 of 121317 (100%) 121317 (100%)

execution time: 484 ms

SELECT

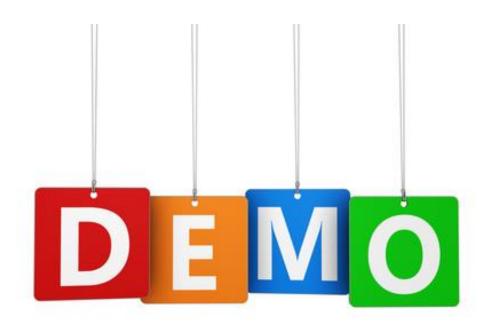
Cost: 0 %



# Table Variable Deferred Compilation

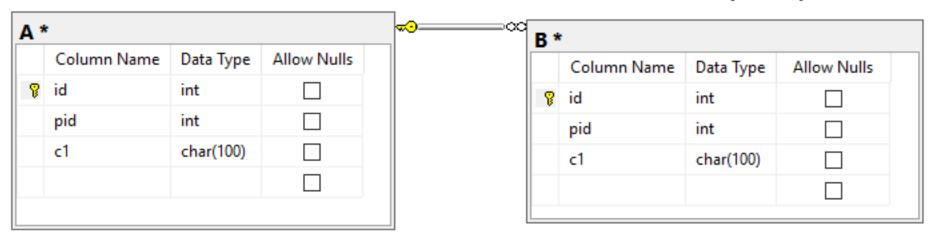
- Improves plan quality and overall performance for queries referencing table variables
- Cardinality estimates are based on actual table variable row counts
- This accurate row count information will be used for optimizing downstream plan operations

### Table Variable Deferred Compilation

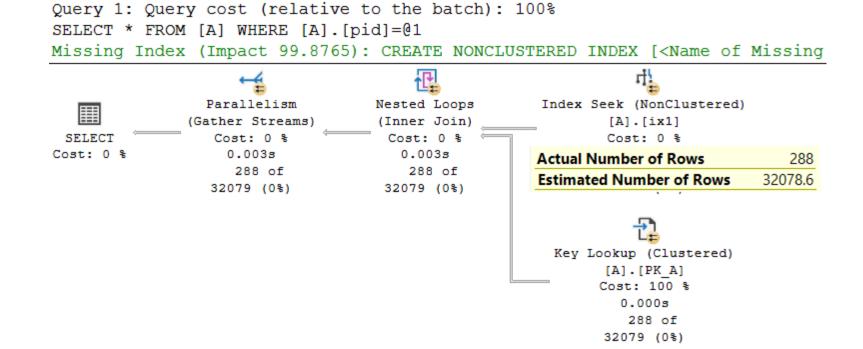


#### 100M (13 GB)

335 M (42 GB)



SELECT \* FROM A WHERE A.pid = 413032;

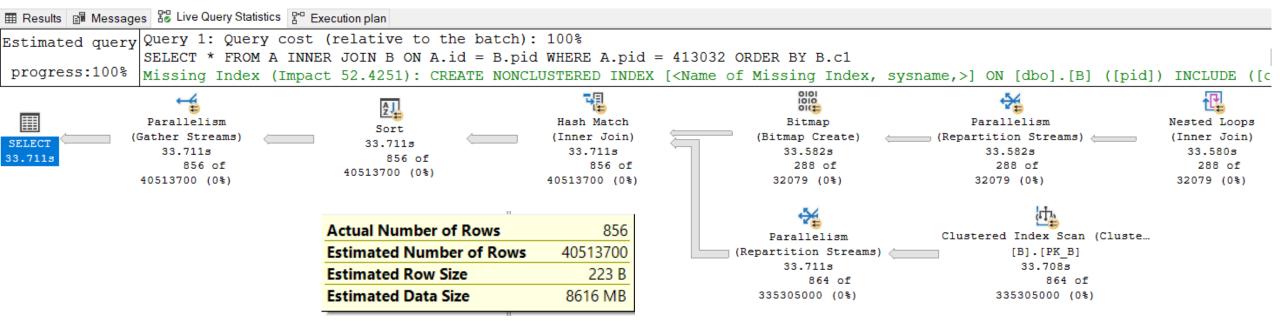


SELECT \* FROM A

INNER JOIN B ON A.id = B.pid

WHERE A.pid = 413032 ORDER BY B.c1;

#### Query returns 856 rows



© Executing query... 100% | SQL2019 (15.0 RTM) | mradivojevic (71) | Statistik | 00:00:34 | 0 rows

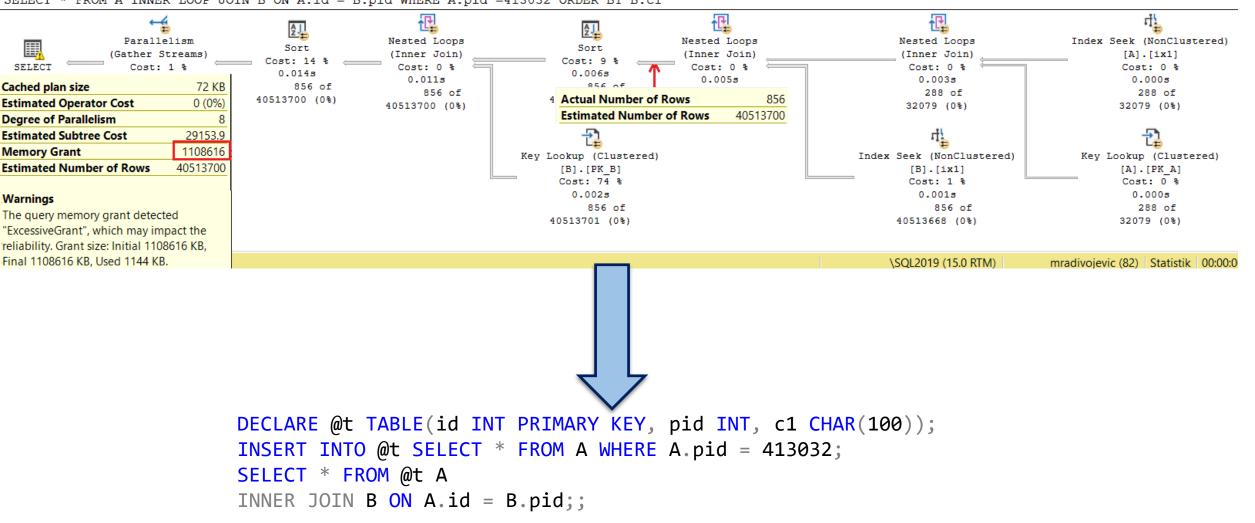
```
SELECT * FROM A

INNER LOOP JOIN B ON A.id = B.pid

WHERE A.pid = 413032 ORDER BY B.c1;
```

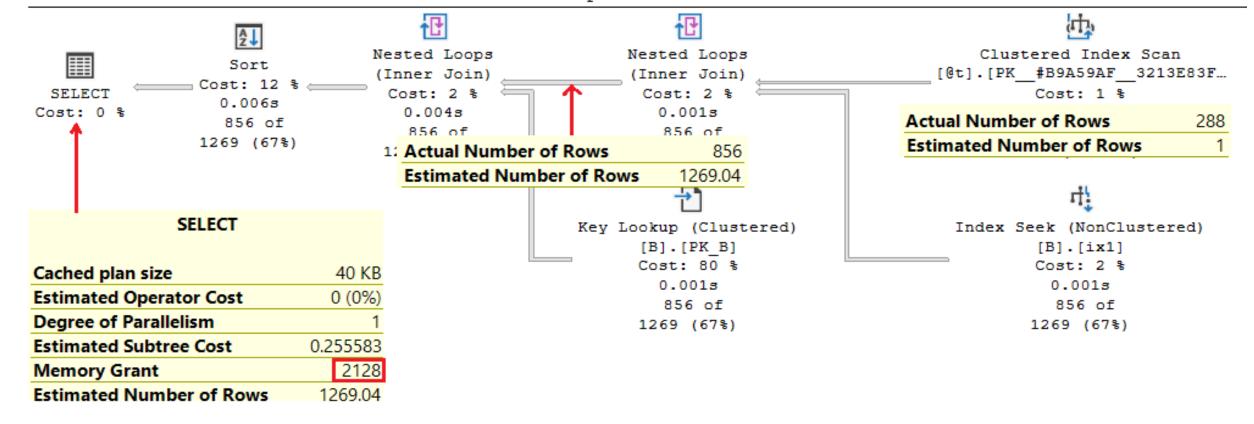
#### Query returns 856 rows

Query 1: Query cost (relative to the batch): 100% SELECT \* FROM A INNER LOOP JOIN B ON A.id = B.pid WHERE A.pid =413032 ORDER BY B.c1

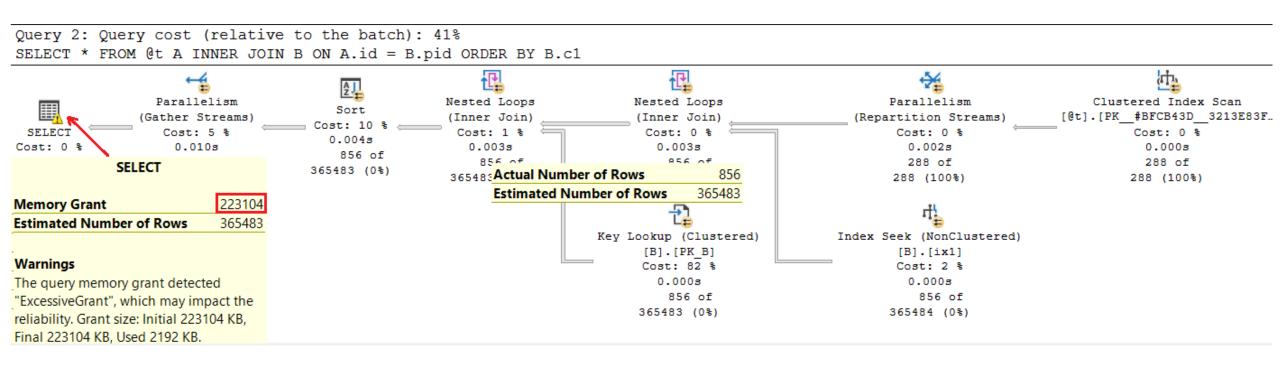


```
DECLARE @t TABLE(id INT PRIMARY KEY, pid INT, c1 CHAR(100));
INSERT INTO @t SELECT * FROM A WHERE A.pid = 413032;
SELECT * FROM @t A
INNER JOIN B ON A.id = B.pid;
```

Query 2: Query cost (relative to the batch): 0% SELECT \* FROM @t A INNER JOIN B ON A.id = B.pid ORDER BY B.c1



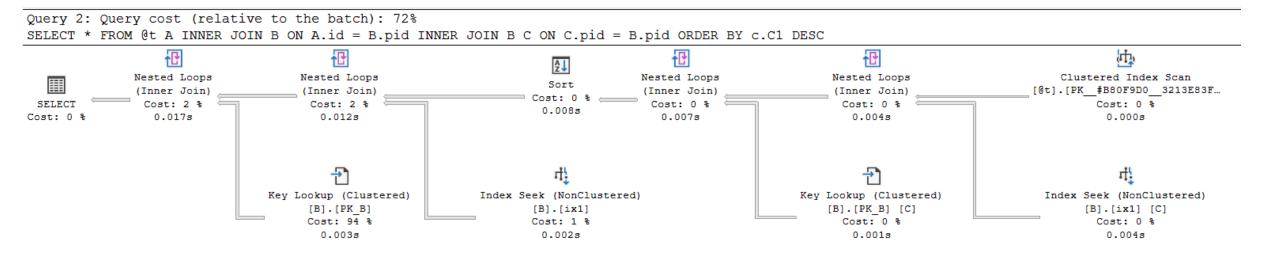
```
DECLARE @t TABLE(id INT PRIMARY KEY, pid INT, c1 CHAR(100));
INSERT INTO @t SELECT * FROM A WHERE A.pid = 413032;
SELECT * FROM @t A
INNER JOIN B ON A.id = B.pid;
```



Parallel plan, higher estimations, higher memory grant

```
DECLARE @t TABLE(id INT PRIMARY KEY, pid INT, c1 CHAR(100));
INSERT INTO @t SELECT * FROM A WHERE A.pid = 413032;
SELECT * FROM @t A
INNER JOIN B ON A.id = B.pid
INNER JOIN B C ON C.pid = B.pid
ORDER BY C.c1 DESC;
```

Query returns 3 222 rows



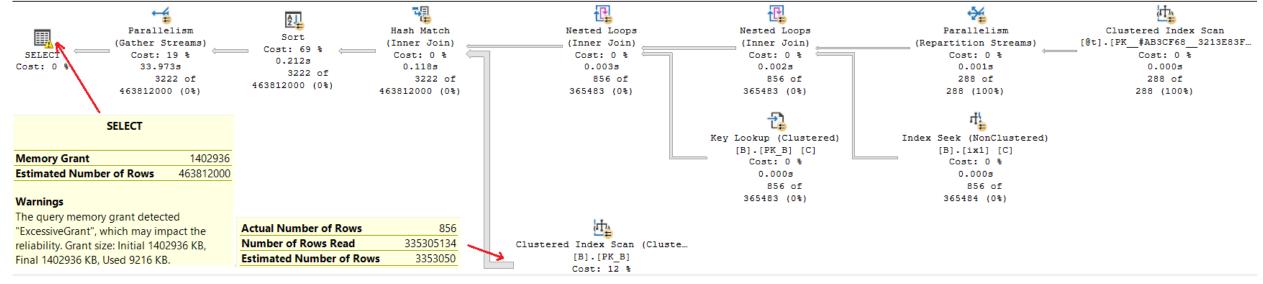
SQL Server Execution Times: CPU time = 31 ms, elapsed time = **116** ms.

```
DECLARE @t TABLE(id INT PRIMARY KEY, pid INT, c1 CHAR(100));
INSERT INTO @t SELECT * FROM A WHERE A.pid = 413032;
SELECT * FROM @t A
INNER JOIN B ON A.id = B.pid
INNER JOIN B C ON C.pid = B.pid
ORDER BY C.c1 DESC;
```

Query 2: Query cost (relative to the batch): 100%

SELECT \* FROM @t A INNER JOIN B ON A.id = B.pid INNER JOIN B C ON C.pid = B.pid ORDER BY c.C1 DESC

Missing Index (Impact 12.5716): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[B] ([pid]) INCLUDE ([c1])

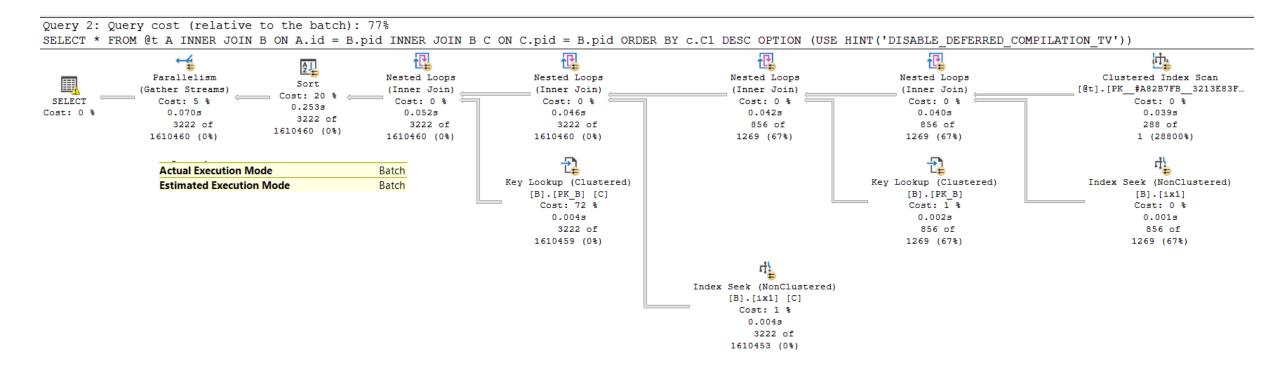


SQL Server Execution Times:

CPU time = 43999 ms, elapsed time = 34482 ms.



```
DECLARE @t TABLE(id INT PRIMARY KEY, pid INT, c1 CHAR(100));
INSERT INTO @t SELECT * FROM A WHERE A.pid = 413032;
SELECT * FROM @t A
INNER JOIN B ON A.id = B.pid
INNER JOIN B C ON C.pid = B.pid
ORDER BY c.C1 DESC
OPTION (USE HINT('DISABLE_DEFERRED_COMPILATION_TV'));
```



SQL Server Execution Times: CPU time = 416 ms, elapsed time = **109** ms

### Table Variable Deferred Compilation

- Designed to address cardinality issues caused by fixed estimation:
  - Nested Loop Joins where Hash Joins are more appropriate
  - Memory grant underestimation issues
- Better estimation for execution plans for new queries
- Prone to parameter sniffing
- Can break existing workarounds

- New operator Adaptive Join
- Allows to choose between Hash Join and Nested Loop Join at runtime
- It starts as Hash Join and if after input scanning
  - Estimated number of rows < threshold => switches to Nested Loop Join
  - Estimated number of rows >= threshold => continues as Hash Join
- It will better handle queries with variety of parameters, but it won't solve all issues caused by wrongly chosen Join operator
- It works only in Batch Mode

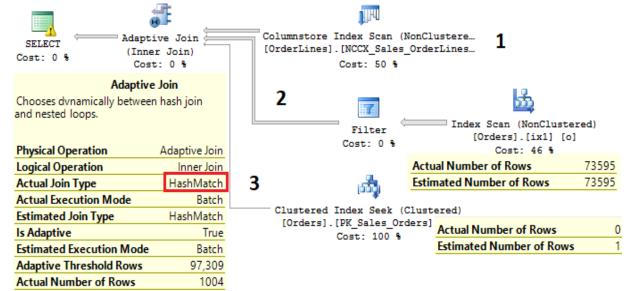
EXEC dbo.GetOrderDetails 1;

**EXEC** dbo.GetOrderDetails 112:

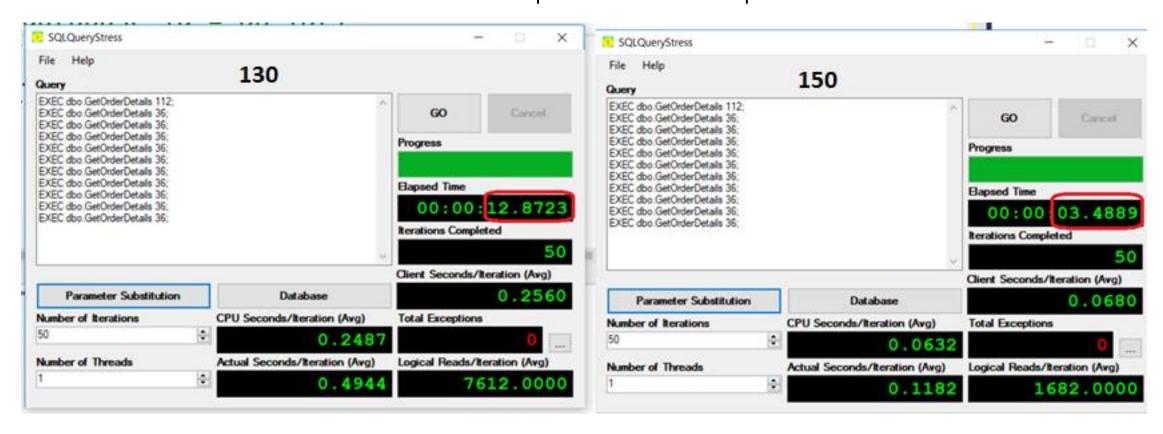
SELECT o.OrderID, o.OrderDate, ol.OrderLineID, ol.Quantity, ol.UnitPrice FROM Sales Missing Index (Impact 49.4219): CREATE NONCLUSTERED INDEX [<Name of Missing Index, Columnstore Index Scan (NonClustere... Adaptive Join SELECT [OrderLines].[NCCX Sales OrderLines... (Inner Join)

Cost: 0 % Cost: 0 % Cost: 50 % Adaptive Join Chooses dynamically between hash join and nested loops. Index Scan (NonClustered) Filter [Orders].[ix1] [o] Cost: 0 % Physical Operation Adaptive Join Cost: 46 % Logical Operation Inner Join Actual Number of Rows Actual Join Type NestedLoops Estimated Number of Rows 73595 Batch Actual Execution Mode Clustered Index Seek (Clustered) Estimated Join Type HashMatch [Orders].[PK Sales Orders] [o] Is Adaptive True Cost: 100 % Actual Number of Rows Estimated Execution Mode Batch Adaptive Threshold Rows 97.309 Estimated Number of Rows Actual Number of Rows 32

SELECT o.OrderID, o.OrderDate, ol.OrderLineID, ol.Quantity, ol.UnitPrice FROM Sale: Missing Index (Impact 49.4219): CREATE NONCLUSTERED INDEX [<Name of Missing Index,



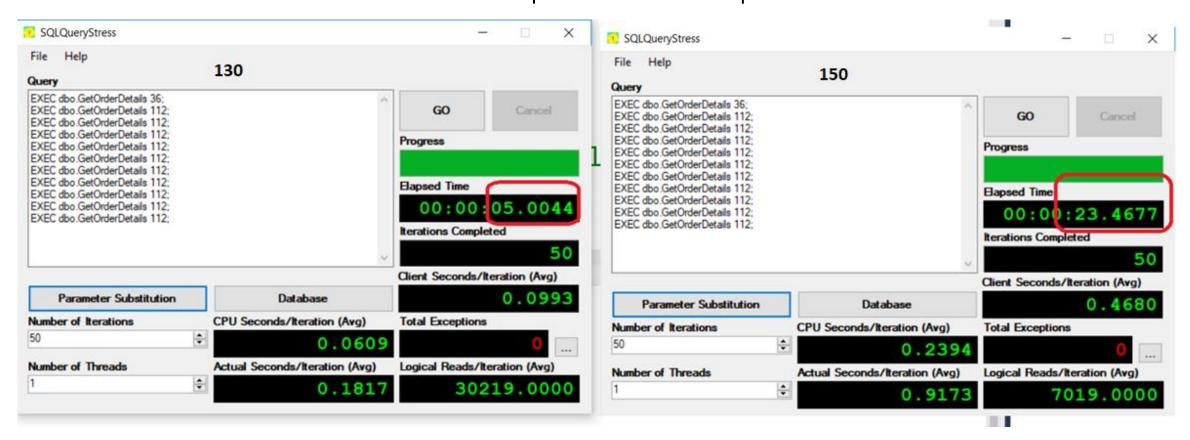
Q: Is it better with new Adaptive Join operator?



A: Yes!!! Under the CL 150, the query runs 4x faster!



Q: Is it better with new Adaptive Join operator?



A: Actually NO - under the CL 150, the query runs 5x slower!

### Batch Mode Adaptive Join - Conclusion

- It can bring benefits for some queries
- It can also bring regressions
- You should test, test and test it with your workload!
- Can handle basic parameter sniffing cases
- It requires operators in batch mode

#### Interleaved Execution

- Related to queries with multi statement table valued functions (MTVF)
  - Breaks the optimization process
  - Executes a part of the query with function call and get actual cardinality
- Epilogue: More appropriate plan (correct cardinality instead of cardinality 100)
- Costs: Increased CPU compile time
- Limits: It works with fixed parameters only

#### Interleaved Execution

```
DECLARE @d DATETIME = SYSDATETIME();
SELECT ol.OrderID, ol.UnitPrice, ol.StockItemID FROM Sales.Orderlines olINNER JOIN dbo.SignificantOrders() f1
ON f1.Id = ol.OrderID WHERE PackageTypeID = 7;
PRINT CONCAT('Execution time: ', DATEDIFF(millisecond, @d, SYSDATETIME()), ' ms');
GO 5
```

#### dbo.SignificantOrders() cardinality: 74K rows

#### Compatibility Level 130

```
Beginning execution loop
Execution time: 777 ms
Execution time: 763 ms
Execution time: 781 ms
Execution time: 765 ms
Execution time: 772 ms
Batch execution completed 5 times.
```

Plan: Nested Loop Join

#### Compatibility Level 140

```
Beginning execution loop
Execution time: 385 ms
Execution time: 363 ms
Execution time: 368 ms
Execution time: 361 ms
Execution time: 364 ms
Batch execution completed 5 times.
Plan: Hash Join
```

**2**x

#### APPROX\_DISTINCT\_COUNT

- It uses significantly less memory resources
- Error from the precise COUNT DISTINCT equivalent within 2% for most workloads
- Great performance for data sets with a high number of distinct values
- Implemented the HyperLogLog algorithm
- It's more about used resources then about the speed

# HyperLogLog Algorithm

- It hashes each element to make the data distribution more uniform
- After hashing all the elements, it looks for the binary representation of each hashed element
- HLL looks number of leading zero bits in the hash value of each element and finds maximum number of leading zero bits
- Number of distinct elements=2^(k+1)
- where k is maximum number of leading zeros in data set
- Documentation: http://algo.inria.fr/flajolet/Publications/FlFuGaMe07.pdf

--Table A 100M rows

SELECT COUNT(DISTINCT(pid)) FROM dbo.A;

SELECT APPROX COUNT DISTINCT(pid) FROM dbo.A;

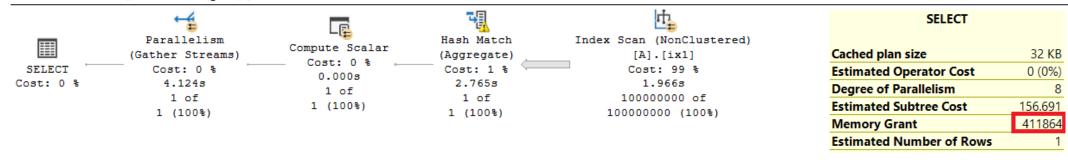
SQL Server Execution Times:

CPU time = 15735 ms, elapsed time = **2493 ms**.

SQL Server Execution Times:

CPU time = 12360 ms, elapsed time = 1988 ms.

Query 1: Query cost (relative to the batch): 50% SELECT COUNT(DISTINCT(pid)) FROM dbo.A



Query 2: Query cost (relative to the batch): 50% SELECT APPROX COUNT DISTINCT(pid) FROM dbo.A

1 of 101 1 of 10000000 of 1 (100%) 1 (100%) 1 (100%) 10000000 (100%)	SELECT Cost: 0 %		Compute Scalar Cost: 0 % 0.000s 1 of 1 (100%)		
--	------------------	--	---	--	--

SELECT	
Cached plan size	40 KB
<b>Estimated Operator Cost</b>	0 (0%)
Degree of Parallelism	8
<b>Estimated Subtree Cost</b>	156.691
Memory Grant	24776
Estimated Number of Rows	1

# Intelligent Query Processing - Conclusion

- It will affect OLTP workload, significantly more than SQL Server 2017
- Many issues will be solved, but for specific query patterns
- Benefits can be even huge, depending on workload
- Regressions are possible, but you can easily mitigate them by disabling features at different levels
- Expectations for improvement in existing OLTP workloads
  - HIGH: Batch mode on rowstore and Memory Grant, Scalar UDF Inlining
  - LOW: Table Variable Deferred Compilation and Adaptive Joins