



# SQL Server 2017

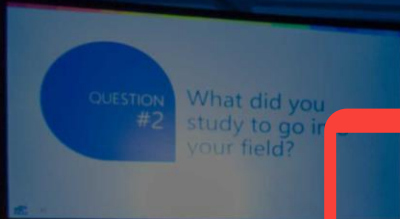
## Intelligence: Meet Database

Pedro Lopes, Program Manager, Microsoft  
Joe Sack, Program Manager, Microsoft





Please silence  
cell phones



# Explore everything PASS has to offer



24HOURS  
OF  
PASS

Free online webinar  
events



LOCAL  
GROUPS

Local user groups  
around the world



SQLSATURDAY  
PASS

Free 1-day local  
training events



VIRTUAL  
GROUPS

Online special  
interest user groups



BUSINESS  
ANALYTICS DAY  
PASS

Business analytics  
training



PASS  
VOLUNTEERS

Get involved

## Free Online Resources

PASS Blog  
White Papers  
Session Recordings

## Newsletter

PASS Connector  
BA Insights

[www.pass.org](http://www.pass.org)



# Session evaluations

Your feedback is important and valuable.

Submit by 5pm Friday, November 10<sup>th</sup> to win prizes. **3 Ways to Access:**



Go to [passSummit.com](https://passSummit.com)



Download the GuideBook App  
and search: PASS Summit 2017



Follow the QR code link  
displayed on session signage  
throughout the conference  
venue and in the program guide



# Pedro Lopes

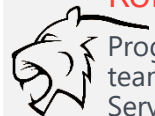
Program Manager,  
Microsoft



/pedroazevedolopes



@sqlpto



## Role

Program manager on the SQL Server Tiger team – owning all in-market versions of SQL Server

## Focus areas

Relational Engine – Query processing, performance tuning and optimization.

## History

Working with SQL Server since 2002.



# Joe Sack

Program Manager,  
Microsoft



/joesack



@JoeSackMSFT

## Role

Program manager on the SQL Server and Azure SQL Database product team.

## Focus areas

Query processing, performance and scalability.

## History

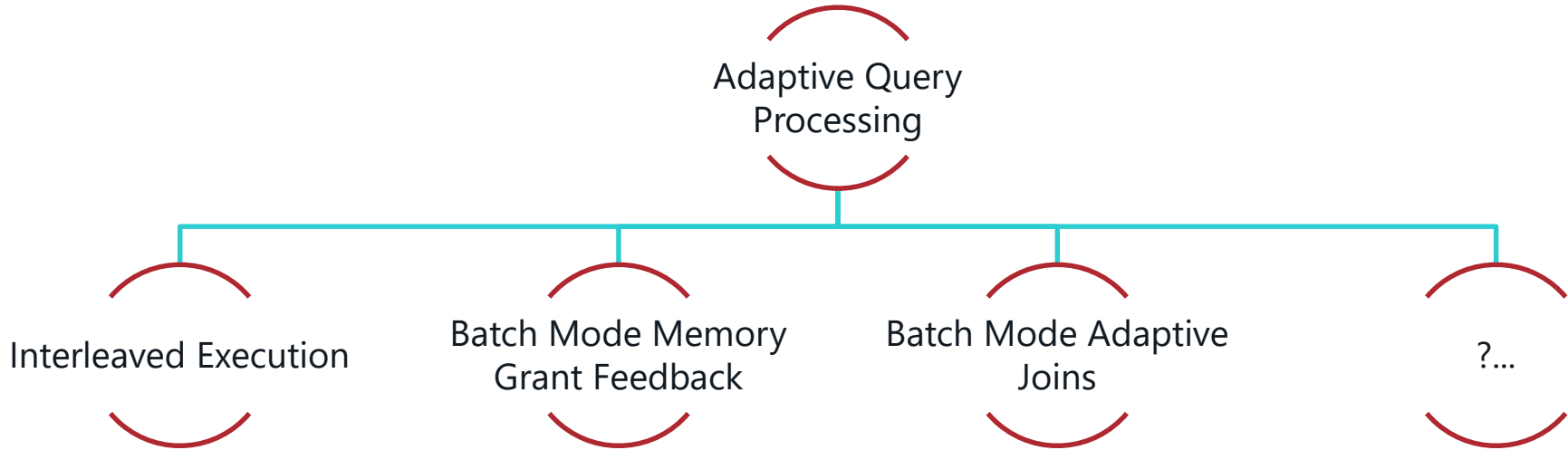
SQL Server professional since 1997.

# Agenda

## Today we'll cover:

- Engine query processing features introduced in 2017
- Query Store enhancements
- Performance dashboard
- SSMS plan scenarios
- Troubleshooting Parallelism Waits
- Roadmap "what's next" (beyond 2017 RTM)

# Adaptability in SQL Server





# Query Processing and Cardinality Estimation

During optimization, the cardinality estimation (CE) process is responsible for estimating the number of rows processed at each step in an execution plan



CE uses a combination of statistical techniques and assumptions



When estimates are accurate (enough), we make informed decisions around order of operations and physical algorithm selection

# Common reasons for incorrect estimates



Missing statistics



Stale statistics



Inadequate statistics sample rate



Bad parameter sniffing scenarios



Out-of-model query constructs

- E.g. Multi-Statement TVFs, table variables, XQuery



Assumptions not aligned with data being queried

- E.g. independence vs. correlation

# Cost of incorrect estimates

Slow query  
response time due  
to inefficient plans

Excessive resource  
utilization (CPU,  
Memory, IO)

Spills to disk

Reduced  
throughput and  
concurrency

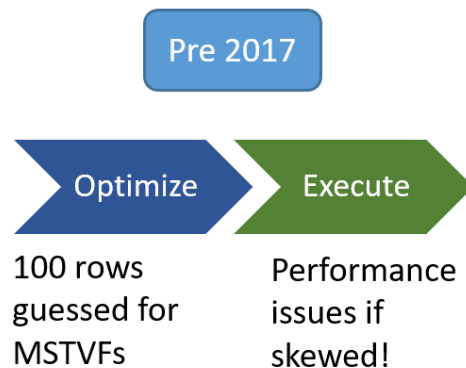
T-SQL refactoring  
to work around off-  
model statements

# Interleaved Execution for MSTVFs

Problem: Multi-statement table valued functions (MSTVFs) are treated as a black box by QP and we use a fixed optimization guess

Interleaved Execution will materialize and use row counts for MSTVFs

Downstream operations will benefit from the corrected MSTVF cardinality estimate

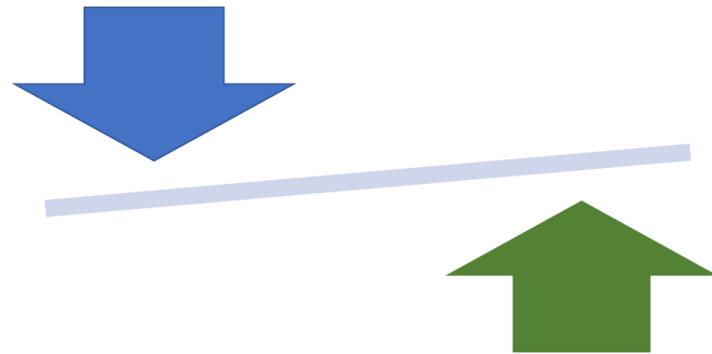


# Batch Mode Memory Grant Feedback (MGF)

Problem: Queries may spill to disk or take too much memory based on poor cardinality estimates

MGF will adjust memory grants based on execution feedback

MGF will remove spills and improve concurrency for repeating queries

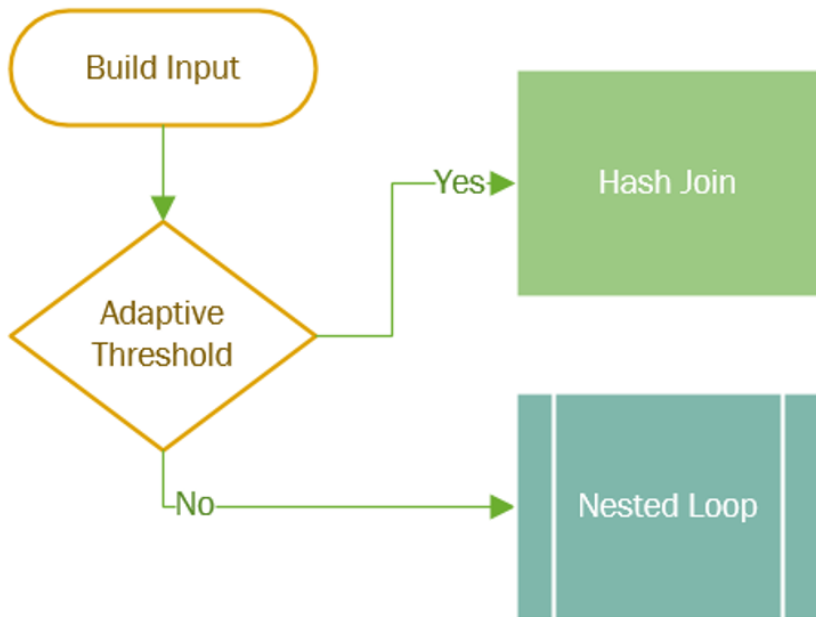


# Batch Mode Adaptive Joins (AJ)

Problem: If cardinality estimates are skewed, we may choose an inappropriate join algorithm

AJ will defer the choice of hash join or nested loop until after the first join input has been scanned

AJ uses nested loop for small inputs, hash joins for large inputs





# AQP

Demo



# Interleaved Execution Candidates

SELECT statements

140 compatibility level

MSTVF *not* used on the inside of a CROSS APPLY (unless w/runtime constants)

When not using plan forcing

When not using forced parameterization (query, database)

When not using USE HINT with DISABLE\_PARAMETER\_SNIFFING (or TF 4136)

# About Interleaved Execution

Expected overhead?	<ul style="list-style-type: none"><li>• Minimal, since we're already materializing MSTVFs</li></ul>
Cached plan considerations	<ul style="list-style-type: none"><li>• First execution cached will be used by consecutive executions</li></ul>
Plan attributes	<ul style="list-style-type: none"><li>• Contains Interleaved Execution Candidates</li><li>• Is Interleaved Executed</li></ul>
Xevents	<ul style="list-style-type: none"><li>• Execution status, CE update, disabled reason</li></ul>

# About Batch Mode Memory Grant Feedback

Expected overhead?

- If there is oscillation, we will disable the loop after multiple executions

XEvents

- Spill report, and updates by feedback

Expected decrease and increase size?

- For spills – spill size plus a buffer
- For overages – reduce based on waste, and add a buffer

RECOMPILE or eviction scenarios

- Memory grant size will go back to original

# Batch Mode Adaptive Joins

## Eligible statements

The join is eligible to be executed both by an indexed nested loop join or a hash join physical algorithm.

The hash join uses batch mode – either through the presence of a Columnstore index in the query overall or a Columnstore indexed table being referenced directly by the join.

# Adaptive Join Threshold





# About Batch Mode Adaptive Join

Expected overhead?	<ul style="list-style-type: none"><li>• We grant memory even for NL scenario, so if NL is * always * optimal, you have more overhead</li></ul>
Plan attributes	<ul style="list-style-type: none"><li>• Adaptive Threshold Rows, Estimated and Actual Join Type</li></ul>
xEvents	<ul style="list-style-type: none"><li>• Adaptive join skipped</li></ul>
Cached plan considerations	<ul style="list-style-type: none"><li>• Single compiled plan can accommodate low and high number of rows scenarios</li></ul>

# The middle-of-the-night call

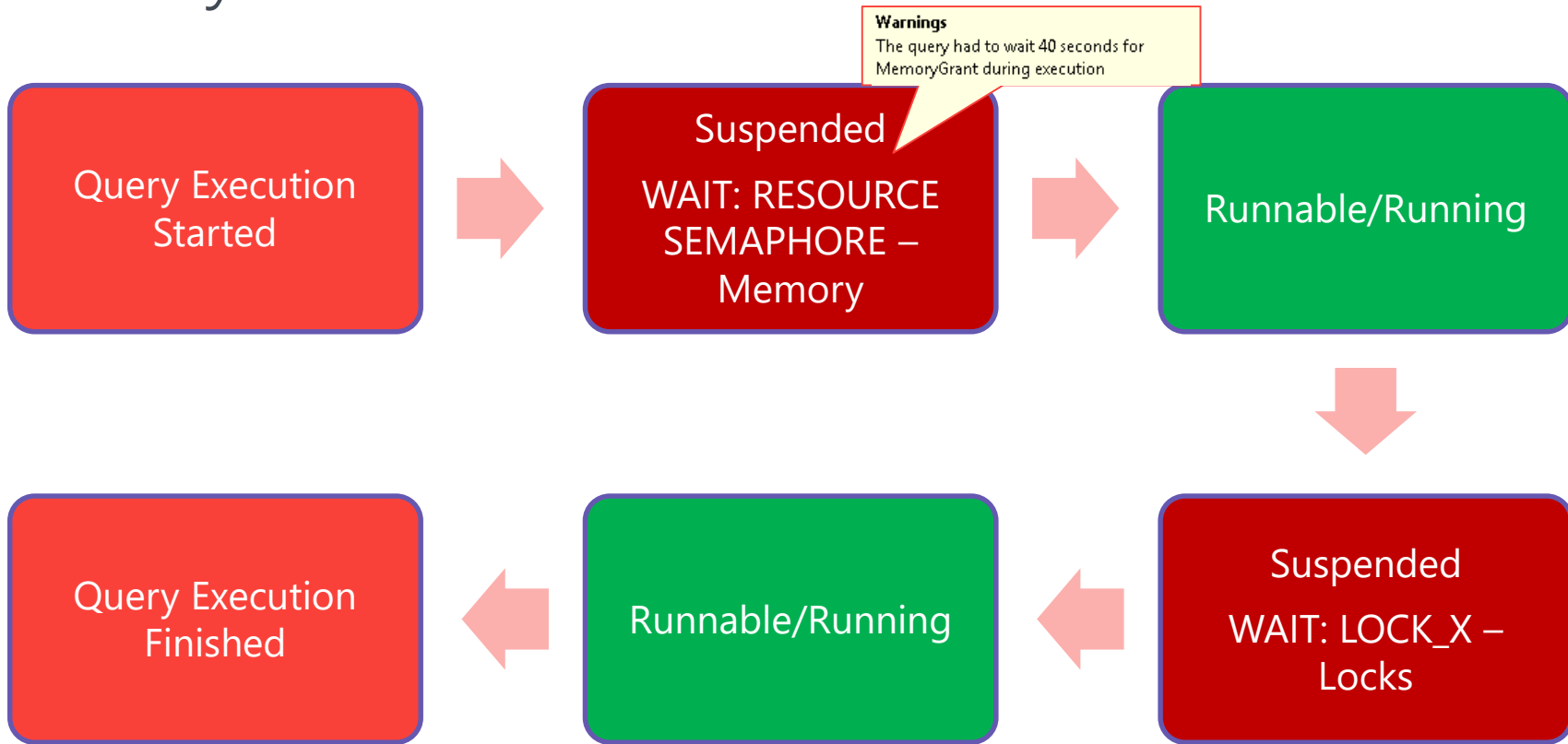
You're on call for supporting the data tier of a mission-critical SQL Server instance

Key business processes are being delayed.

You get a call asking to **mitigate** the issue and then determine the **root cause**.



# Query execution and wait statistics



# Wait statistics in Query Store

Demo



# Another middle-of-the-night call

You're on call for supporting the data tier of a mission-critical SQL Server instance

There has been a jump in CPU utilization on a key server, and one of the critical stored procedure calls is now running (much) more slowly than it used to?

You've been asked to **mitigate** the issue and then determine the **root cause**



# Perf Dashboard native in SSMS 17.2

## Microsoft SQL Server Performance Dashboard

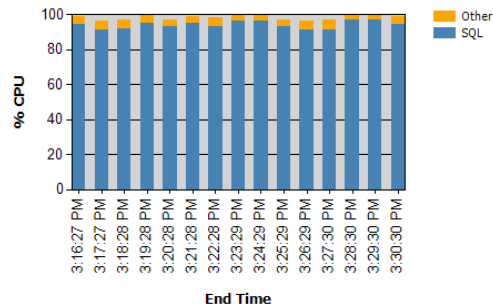
Report Local Time: 5/31/2017 3:31:04 PM

13.0.4422.0 - Enterprise Edition (64-bit)



Overall performance may be degraded because the system shows signs of being CPU-bound. This SQL Server instance is consuming the majority of the CPU. Click on any of the SQL data points in the chart below to investigate further.

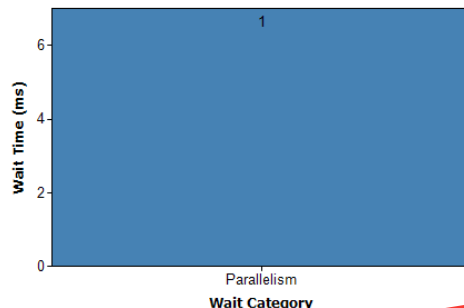
### System CPU Utilization



### Current Activity

	User Requests	User Sessions
Count	27	32
Elapsed Time (ms)	4573004	741818
CPU Time (ms)	2043203(44.68%)	101108(13.63%)
Wait Time (ms)	2529801(55.32%)	640710(86.37%)
Cache Hit Ratio	100.000%	98.313%

### Current Waiting Requests



No extra downloads!  
No new schema to deploy!  
Long standing request by  
CSS and customers

### Historical Information

<a href="#">Waits</a>	<a href="#">IO Statistics</a>
<a href="#">Latches</a>	
Expensive Queries	
<a href="#">By CPU</a>	<a href="#">By Duration</a>
<a href="#">By Logical Reads</a>	<a href="#">By Physical Reads</a>
<a href="#">By Logical Writes</a>	<a href="#">By CLR Time</a>

### Miscellaneous Information

<a href="#">Active Traces</a>	1
<a href="#">Active Xevent Sessions</a>	4
<a href="#">Databases</a>	16
<a href="#">Missing Indexes</a>	11

Categorized Wait  
stats page

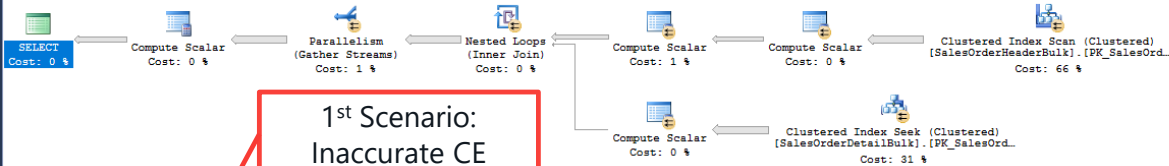
New categorized  
Latches page

Scoring added to  
Missing Index Report



# SSMS Plan Analysis

Query 1: Query cost (relative to the batch): 100%  
SELECT \* FROM Sales.SalesOrderHeaderBulk AS h INNER JOIN Sales.SalesOrderDetailBulk AS d ON h.SalesOrderID = d.SalesOrderID WHERE (h.OrderDate >= @StartOrderdate)



## Showplan Analysis

Multi Statement Scenarios

Each tab below shows details on a category of potential issues found in the plans.

Inaccurate Cardinality Estimation

Findings

< Back Forward >

	Difference %	Actual	Estimated	Node
1	14314	405300	2811.67	Clustered Index Scan (Clustered) [SalesOrderHeaderBulk].[PK_SalesOrderHeaderBulk]
2	12718	1389650	10840.7	Parallelism (Gather Streams)
3	12718	1389650	10840.7	Nested Loops (Inner Join)
4	12718	1389650	10840.7310854	Clustered Index Seek (Clustered) [SalesOrderDetailBulk].[PK_SalesOrderDetailBulk]

Finding Details

[Click here for more information about this scenario](#)

1) The predicate for this operator depends on parameter @StartOrderdate. The compile-time value was unknown or different from the runtime value, so the estimate may not be accurate for the run-time value. Refer to [here](#) for more details.

2) One of the common reasons for estimation differences is the use of different statistics. Check if statistics for table [SalesOrderHeaderBulk] are different or stale. Refer to [here](#) for more information.

## SELECT

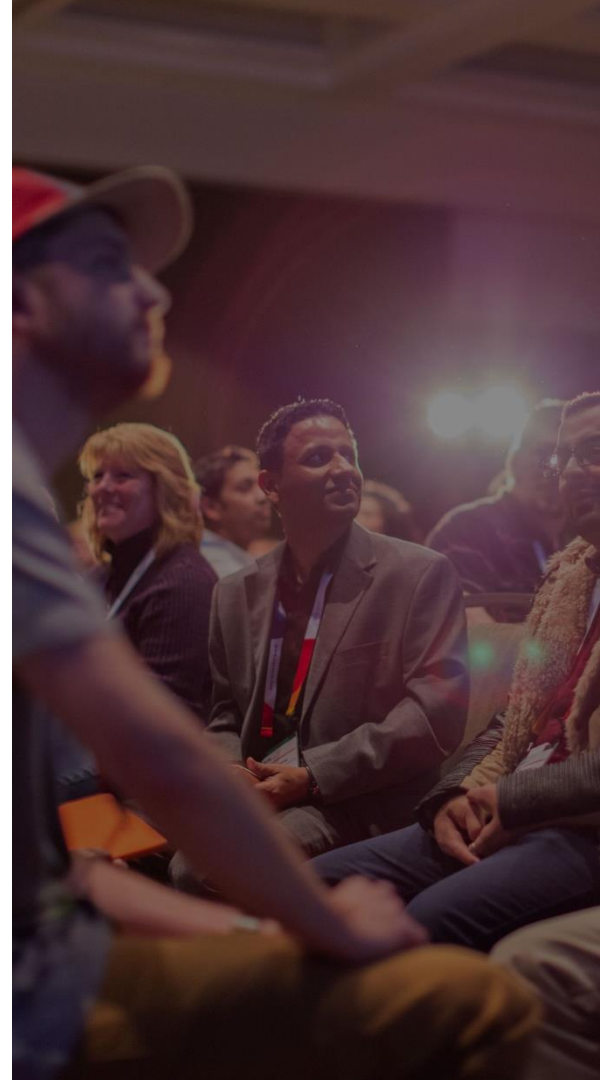
Misc	
Cached plan size	80 KB
CardinalityEstimationModelVersion	130
CompileCPU	6
CompileMemory	592
CompileTime	6
Degree of Parallelism	4
Estimated Number of Rows	10840.7
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	28.8878
Memory Grant	72
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDependentProperties	
Parameter List	@StartOrderdate
Column	@StartOrderdate
Parameter Compiled Value	'2004-07-31 00:00:00.000'
Parameter Data Type	datetime
Parameter Runtime Value	'2004-03-28 00:00:00.000'
QueryHash	0x504FBC112C813CD6
QueryPlanHash	0x99DC0A67C61621F4
QueryTimeStats	
RetrievedFromCache	true
SecurityPolicyApplied	False
Set Options	ANSI_NULLS: True, ANSI_PADDING: True, ANSI_WARNINGS: Off
Statement	SELECT * FROM Sales.SalesOrderHeaderBulk AS h
ThreadStat	
TraceFlags	
WaitStats	

## Parameter List

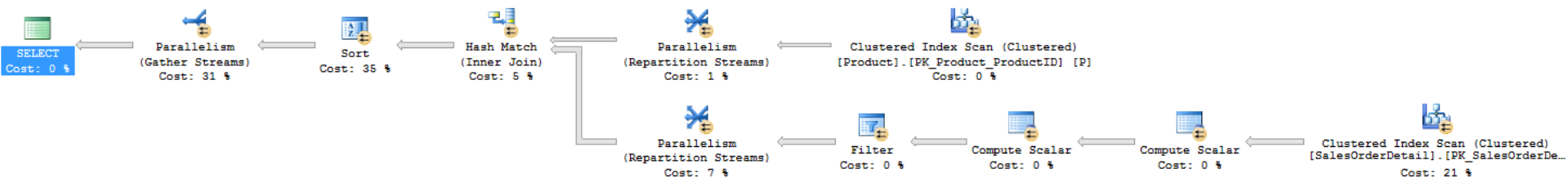
Parameter list.

# SSMS Tools

Demo

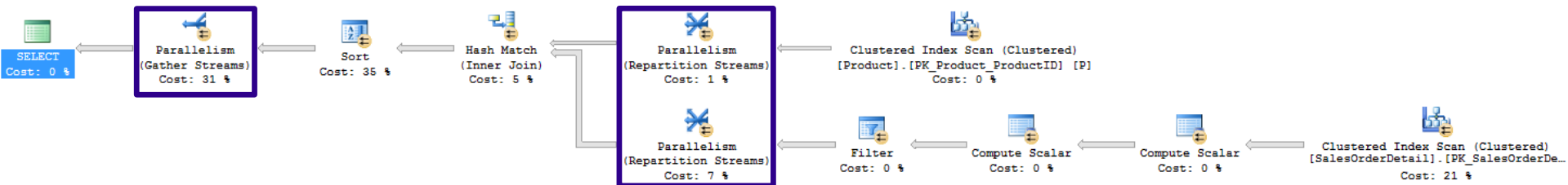


# Defining parallelism



- The Parallelism operator, a.k.a. Exchange Iterator, actually implements parallelism in query execution.
- Moves streams (rowsets) between threads (bound to available DOP).
- It's really two operators:
  - **Producers** that push data to consumers.
  - **Consumers** that may have to wait for data from producers.

# How it implements logical operations



Type		# producer threads	# consumer threads
Gather Streams		DOP	1
Repartition Streams		DOP	DOP
Distribute Streams		1	DOP

# Making parallelism waits actionable



From Docs:

- Occurs when trying to synchronize the query processor exchange iterator.
- Consider lowering the DOP if contention on this wait type becomes a problem.

# Making parallelism waits actionable

SQL Server 2016 SP2

SQL Server 2017 CU3



## CX Packet

- Occurs when trying to synchronize the query processor exchange iterator.
- **Actionable:** consider lowering the DOP if contention on this wait type becomes a problem.
- Now in Showplan.



## CX Consumer

- Occurs with parallel query plans when a consumer thread waits for a producer thread to send rows.
- **Negligible:** this is a normal part of parallel query execution.
- Ignored in Showplan.

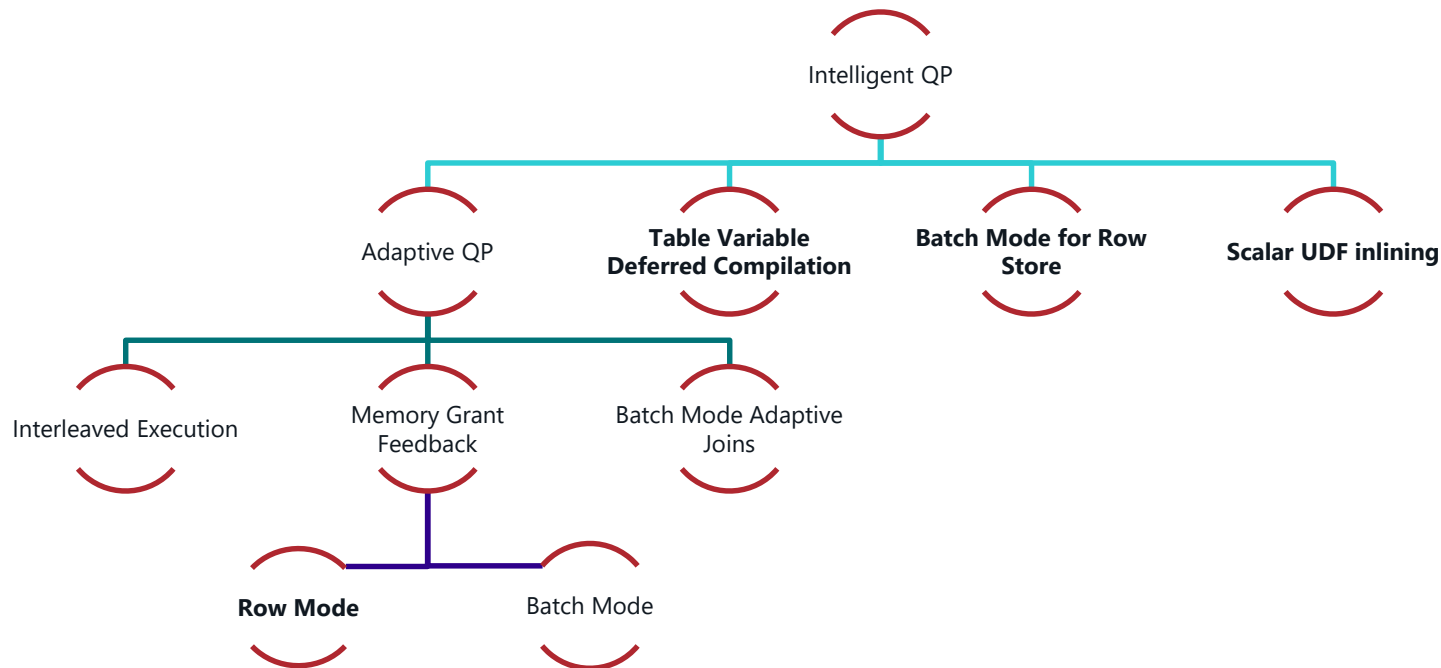


# Parallelism Waits

Demo



# Beyond 2017... Intelligent QP



Interested in testing new QP features? Email [joe.sack@microsoft.com](mailto:joe.sack@microsoft.com)

# Useful links

- Monitoring performance by using the Query Store:  
<http://docs.microsoft.com/sql/relational-databases/performance/monitoring-performance-by-using-the-query-store>
- Query Processing Architecture Guide: <http://aka.ms/sqlserverguides>
- Adaptive query processing in SQL databases  
<http://docs.microsoft.com/sql/relational-databases/performance/adaptive-query-processing>
- Craig Freedman's blog series on Parallelism:  
<https://blogs.msdn.microsoft.com/craigfr/tag/parallelism/>
- SQL Server Tiger team blog series on SSMS-based tools:  
[https://blogs.msdn.microsoft.com/sql\\_server\\_team/tag/ssms](https://blogs.msdn.microsoft.com/sql_server_team/tag/ssms)

# Session evaluations

Your feedback is important and valuable.

Submit by 5pm Friday, November 10<sup>th</sup> to win prizes. **3 Ways to Access:**



Go to [passSummit.com](https://passSummit.com)



Download the GuideBook App  
and search: PASS Summit 2017



Follow the QR code link  
displayed on session signage  
throughout the conference  
venue and in the program guide



# Thank You

Learn more from Pedro Lopes, Joe Sack



@sqlpto



@JoeSackMSFT