24 HOURS OF PASS

SQL Server 2017 Intelligence: Meet Database

Pedro Lopes, Senior Program Manager, Microsoft

@sqlpto

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

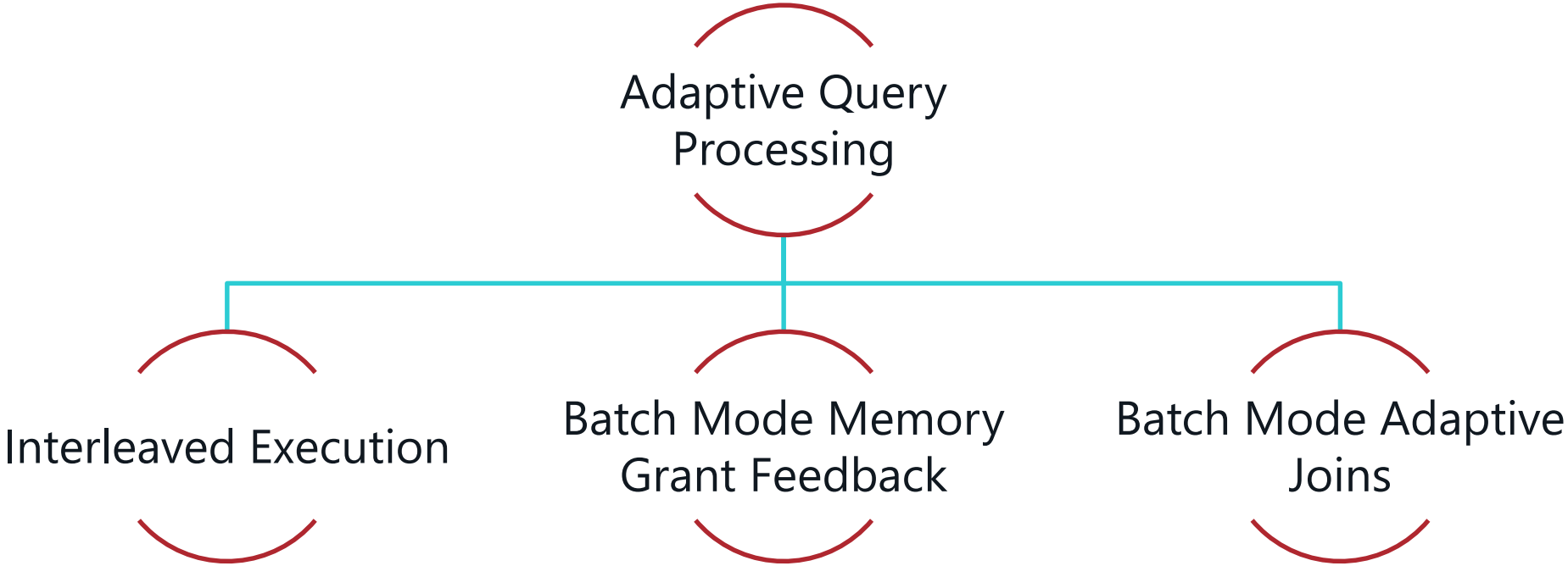
Working with SQL Server since 2002.

Agenda

Today we'll cover:

- Engine query processing features introduced in 2017
- Query Store enhancements
- Performance dashboard
- SSMS plan scenarios
- Troubleshooting Parallelism Waits

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

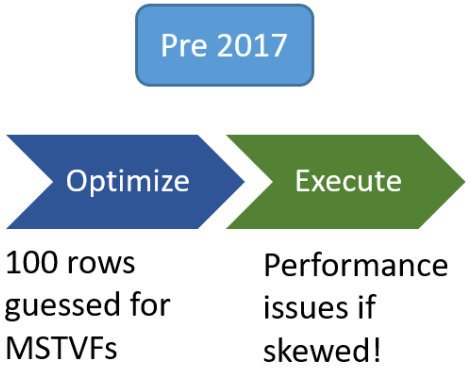
T-SQL refactoring to work around off-model statements

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

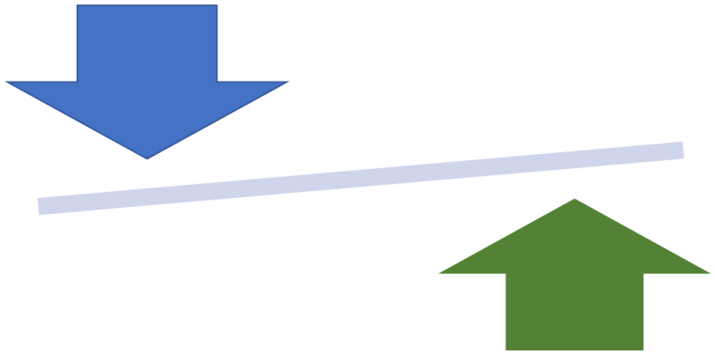


Batch Mode Memory Grant Feedback (MGF)

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

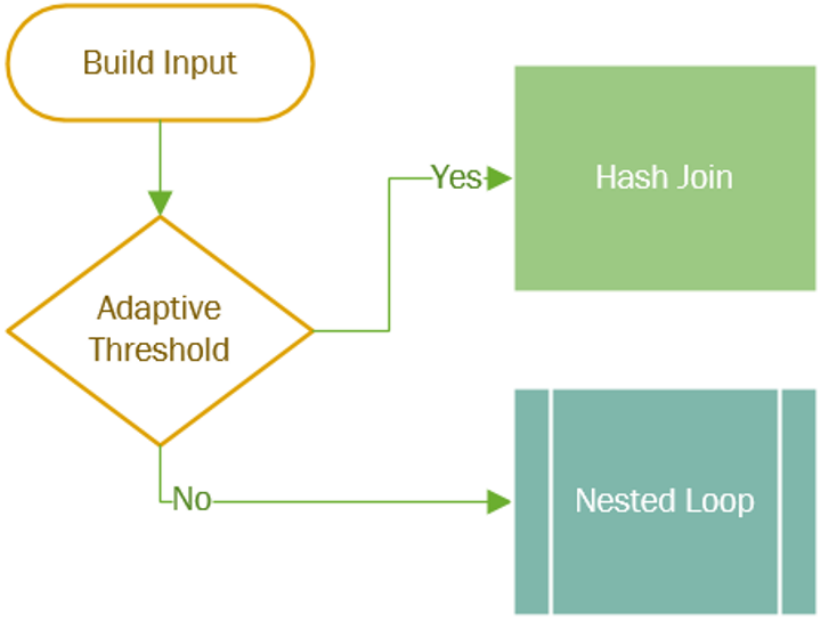


Batch Mode Adaptive Joins (AJ)

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

PASS

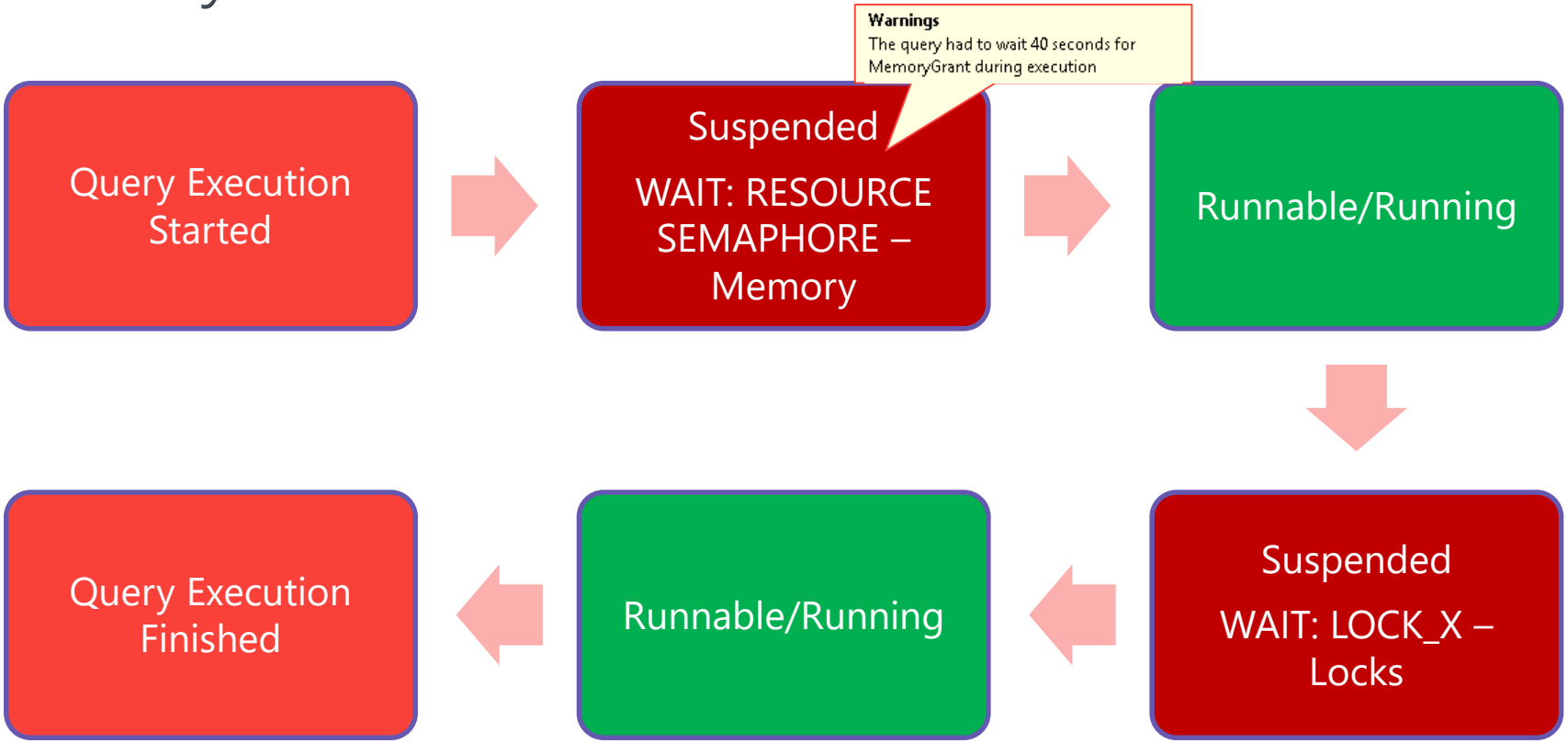
The middle-of-the-night call

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 then it used to?

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

Perf Dashboard native in SSMS 17.2

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

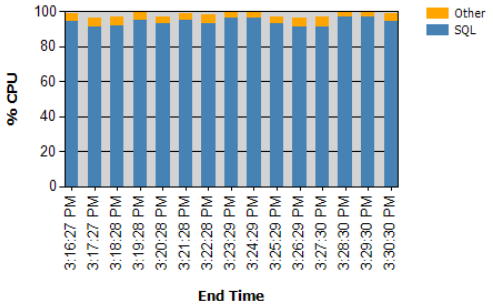
Microsoft SQL Server Performance Dashboard

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

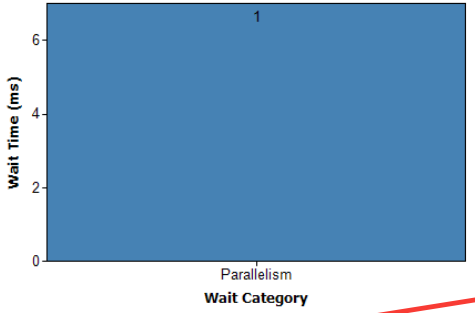
[redacted] (13.0.4422.0 - Enterprise Edition (64-bit))

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 Waiting Requests



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%

Historical Information

Waits — IO Statistics
Latches

Expensive Queries

By CPU — By Duration
By Logical Reads — By Physical Reads
By Logical Writes — By CLR Time

Miscellaneous Information

Active Traces	1
Active Xevent Sessions	4
Databases	16
Missing Indexes	11

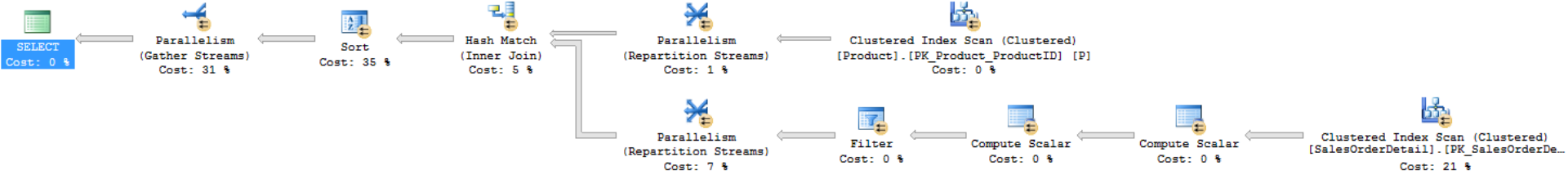
Categorized Wait stats page

New categorized Latches page

Scoring added to Missing Index Report

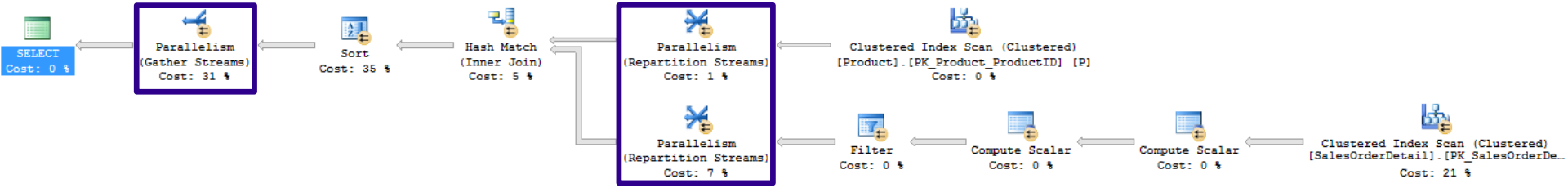
PASS

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

Making parallelism waits **actionable**

CX Packet

SQL Server 2016 SP2
SQL Server 2017 CU3

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.

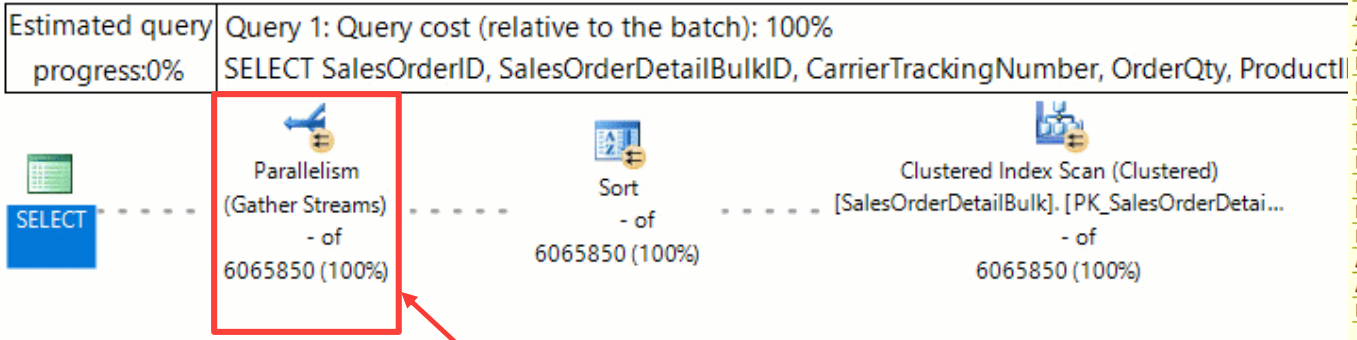
CX Packet

- 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.

Example of a Merging Exchange



Parallelism

An operation involving parallelism.

Physical Operation	Parallelism
Logical Operation	Gather Streams
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	121317
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0.79413 (24%)
Estimated Subtree Cost	3.31101
Estimated CPU Cost	0.794134
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	121317
Estimated Row Size	95 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0

Output List

[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].SalesOrderID,
[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].SalesOrderDetailID,
[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].CarrierTrackingNumber,
[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].OrderQty,
[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].ProductID,
[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].SpecialOfferID,
[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].UnitPrice,
[AdventureWorks2016CTP3].[Sa...

Order By

[AdventureWorks2016CTP3].[Sales].[SalesOrderDetail].ProductID Ascending

⊟ QueryTimeStats	
CpuTime	40068
ElapsedTime	72826

⊟ WaitStats	
⊞ [1]	
⊞ [2]	
⊞ [3]	
⊞ [4]	
⊟ [5]	
WaitCount	87939
WaitTimeMs	805707
WaitType	CXPACKET

Order preserving (merging) Exchange

PASS

Parallelism Waits Demo

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

24HOURS OF PASS

OBRIGADO POR PARTICIPAREM

PASS