SUMMIT2017
PASS

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

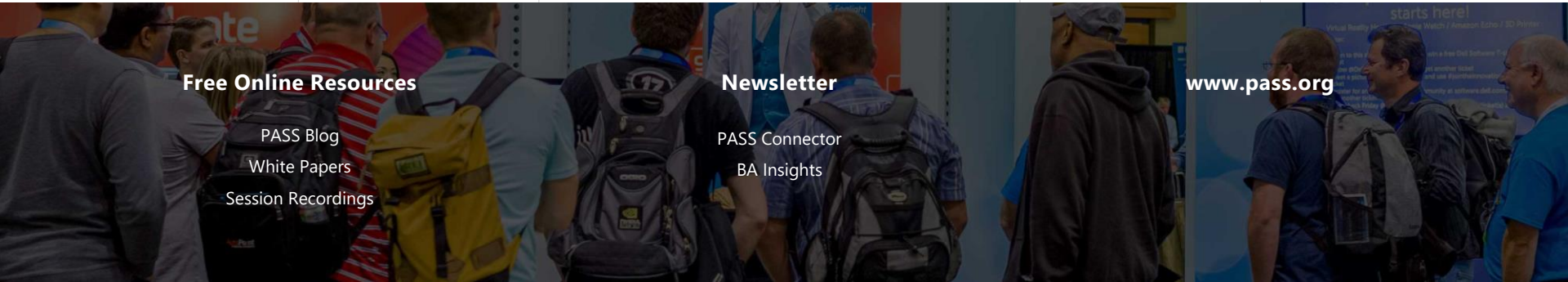
Online special interest user groups

BUSINESS ANALYTICS DAY PASS

Business analytics training

PASS VOLUNTEERS

Get involved



Free Online Resources

PASS Blog
White Papers
Session Recordings

Newsletter

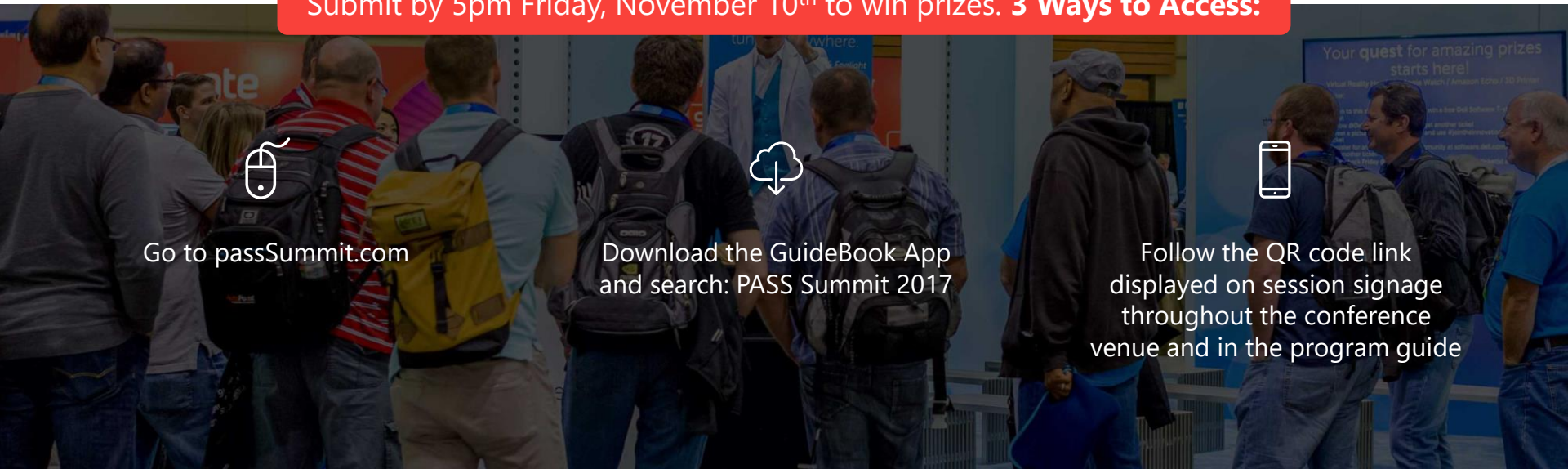
PASS Connector
BA Insights

www.pass.org

Session evaluations

Your feedback is important and valuable.

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



Go to 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

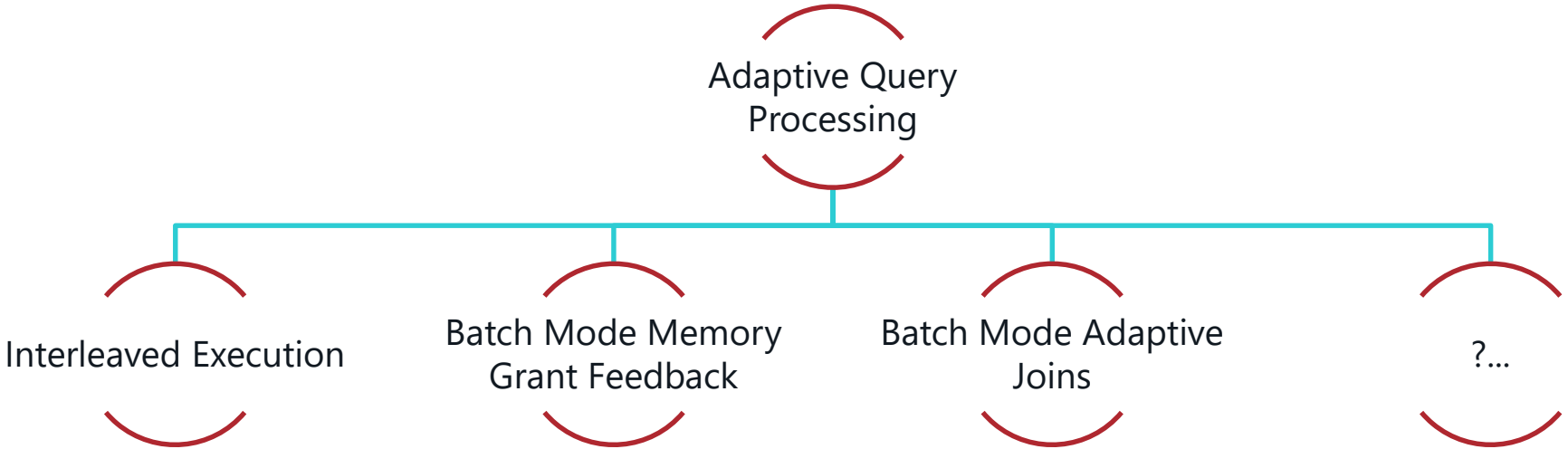
SQL Server professional since 1997.

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

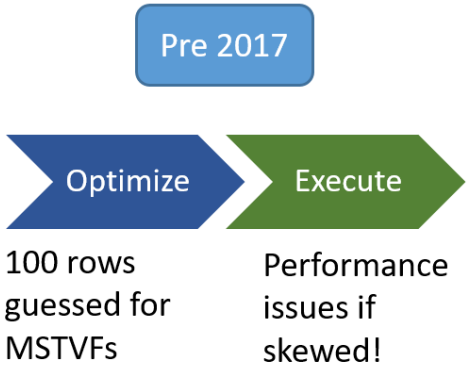
T-SQL refactoring to work around off-model statements

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

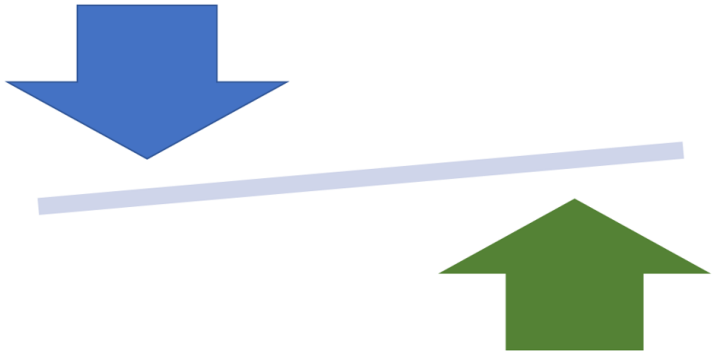


Batch Mode Memory Grant Feedback (MGF)

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

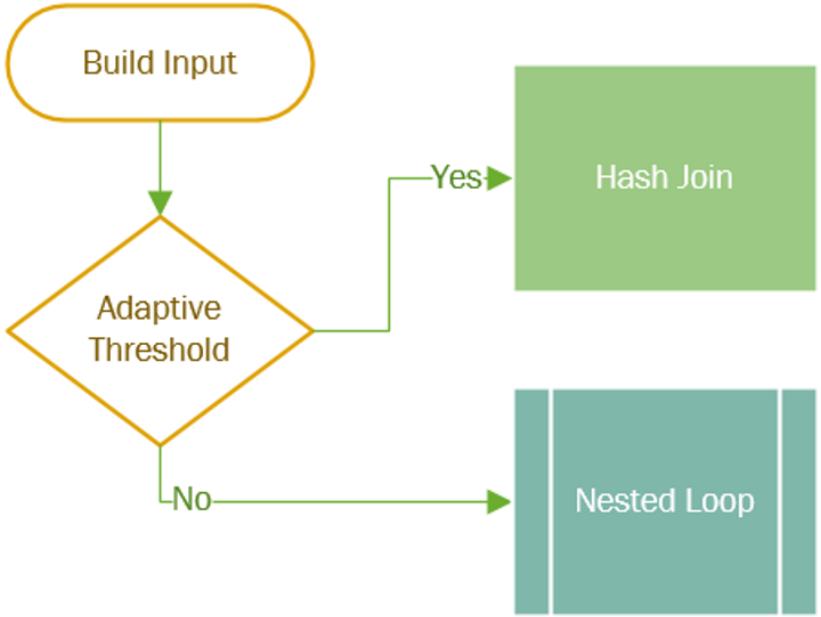


Batch Mode Adaptive Joins (AJ)

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?	• Minimal, since we're already materializing MSTVFs
Cached plan considerations	• First execution cached will be used by consecutive executions
Plan attributes	• Contains Interleaved Execution Candidates • Is Interleaved Executed
Xevents	• Execution status, CE update, disabled reason

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

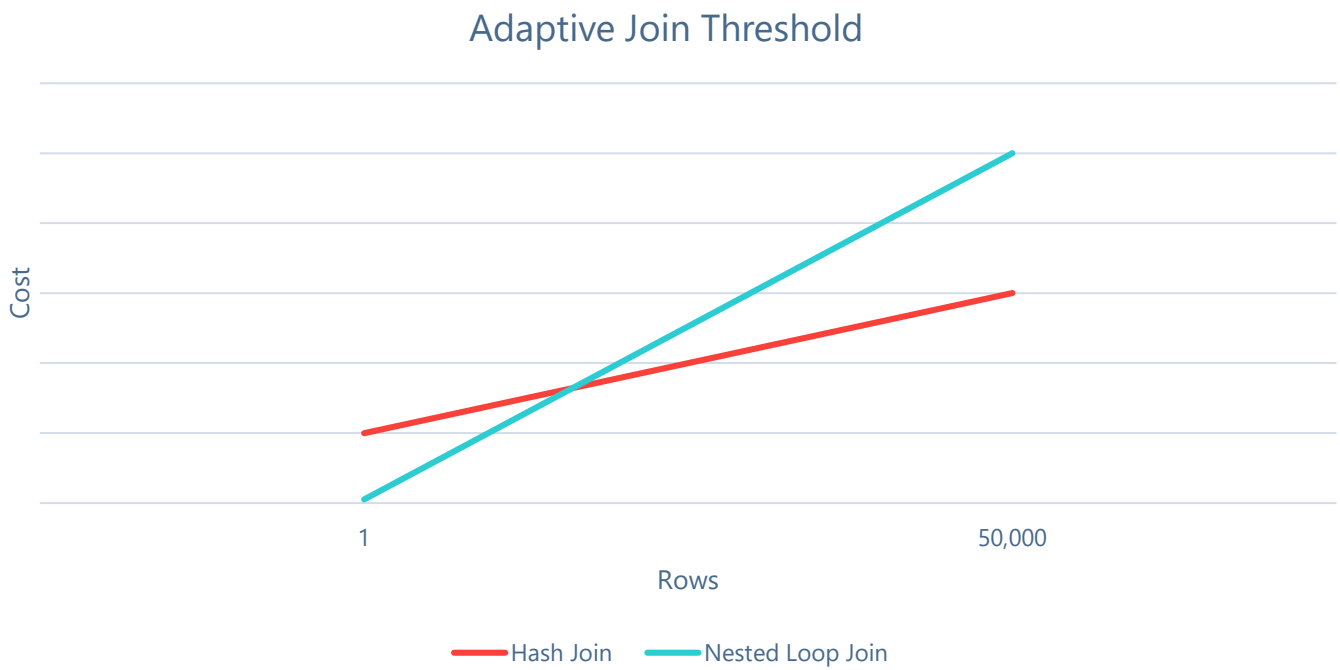
Batch Mode Adaptive Joins

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?	• We grant memory even for NL scenario, so if NL is * always * optimal, you have more overhead
Plan attributes	• Adaptive Threshold Rows, Estimated and Actual Join Type
xEvents	• Adaptive join skipped
Cached plan considerations	• Single compiled plan can accommodate low and high number of rows scenarios

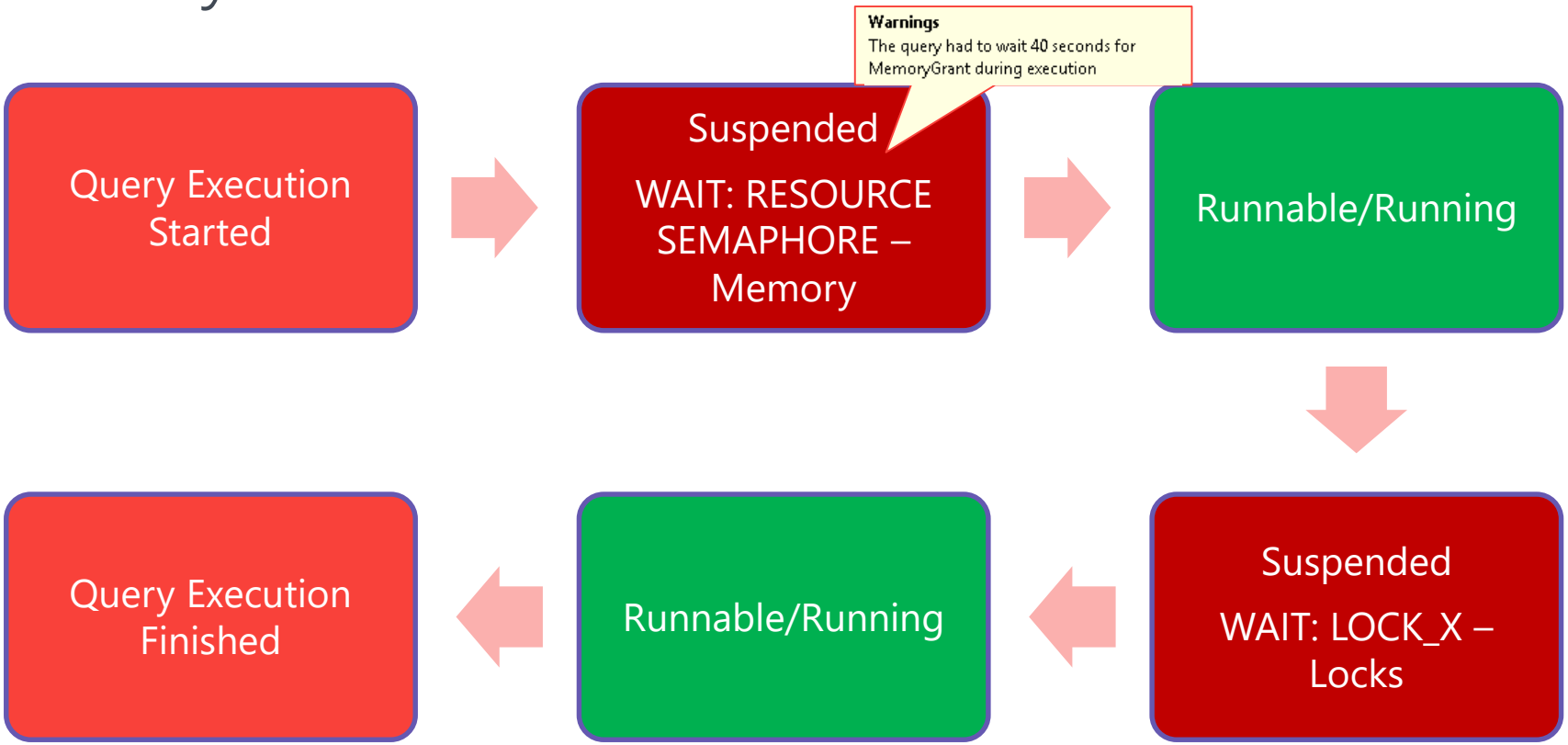
The middle-of-the-night call

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

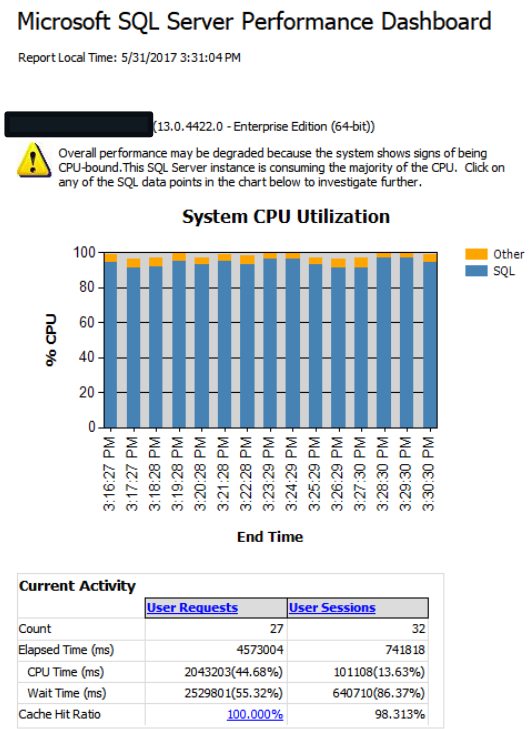
Another middle-of-the-night call

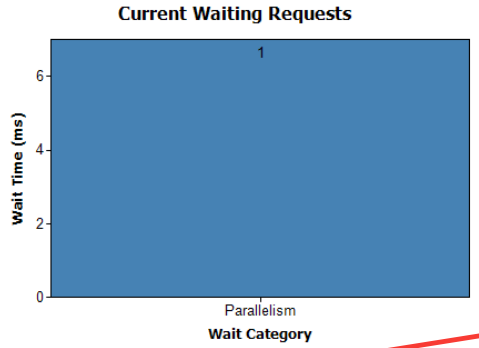
You're on call for supporting the data tier of a mission-critical SQL Server instance

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

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

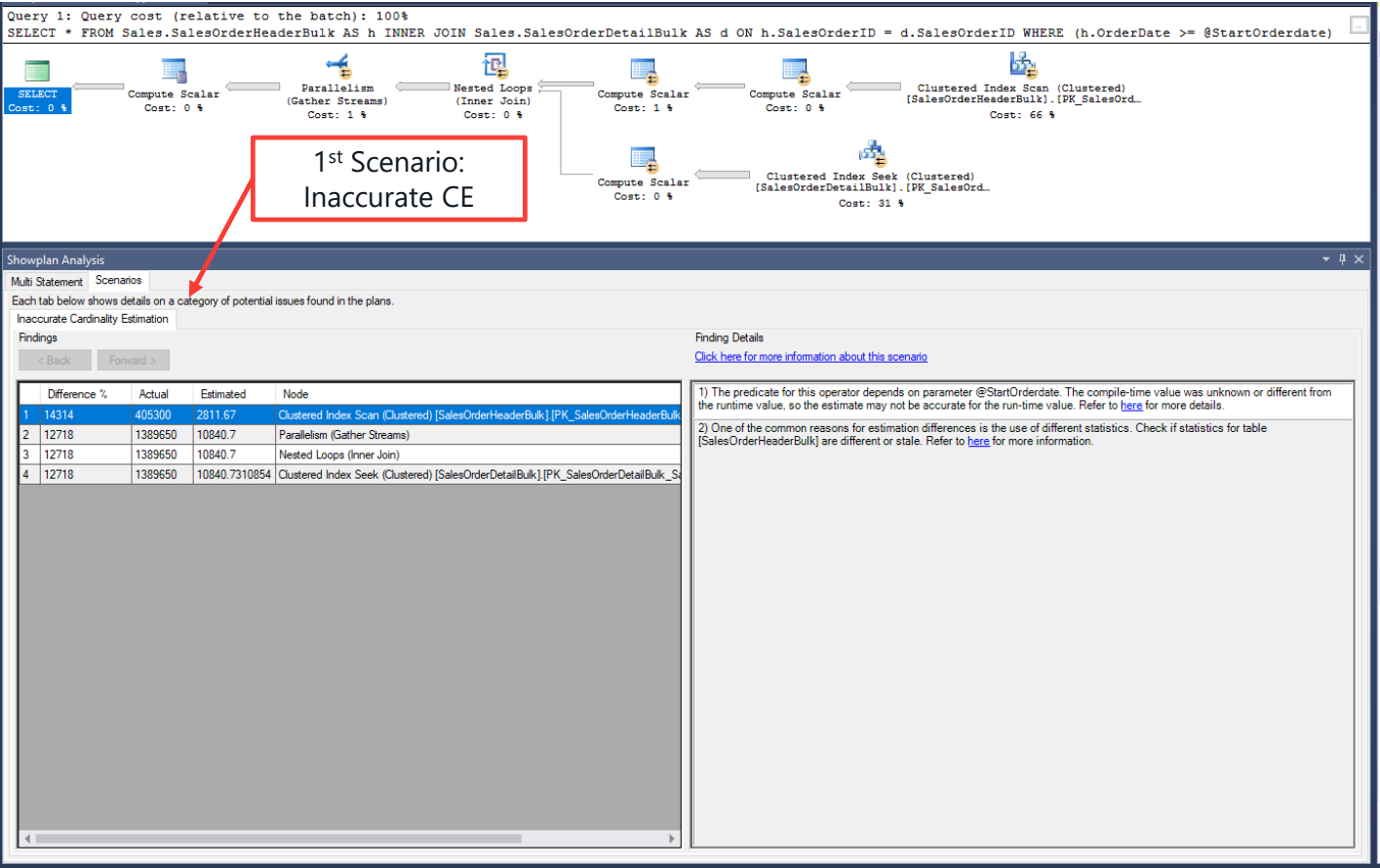
Categorized Wait stats page

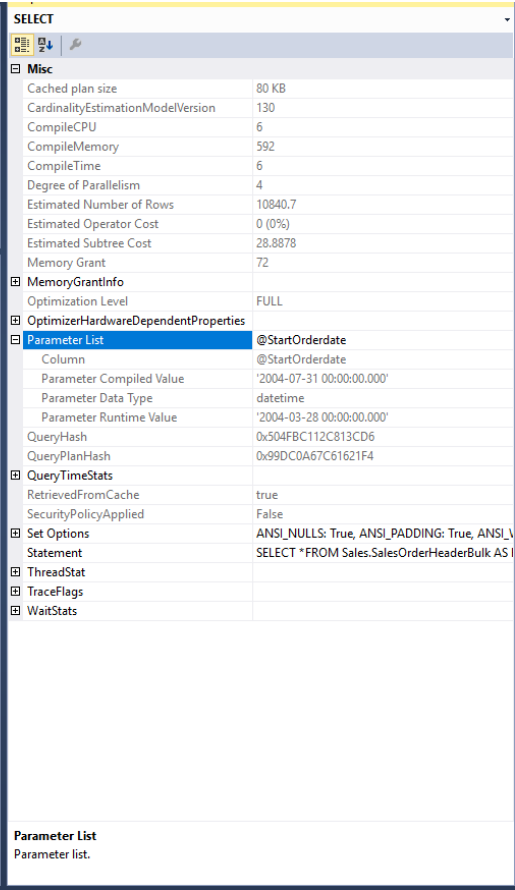
New categorized Latches page

Scoring added to Missing Index Report

PASS

SSMS Plan Analysis

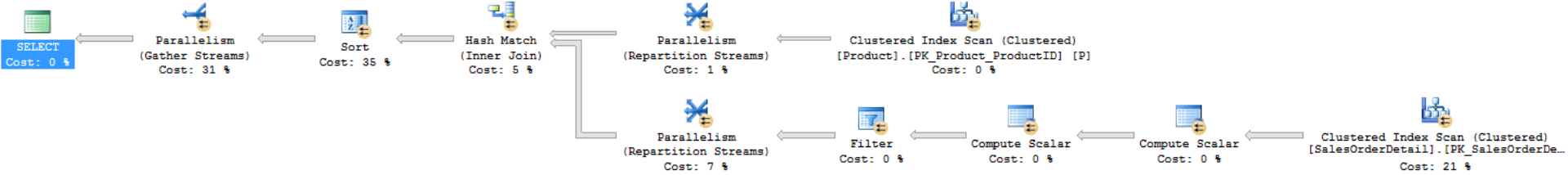




SSMS Tools

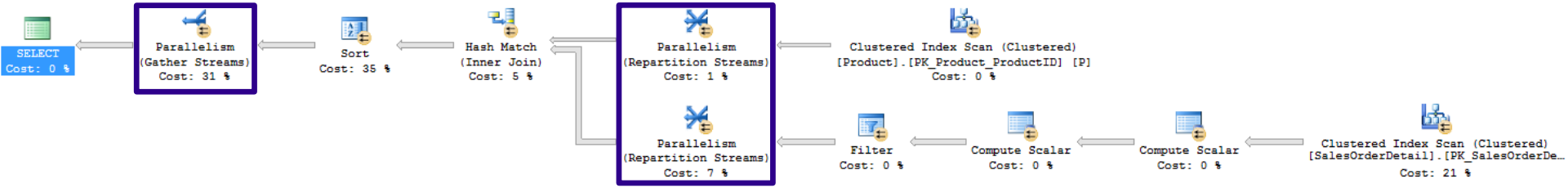
Demo

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

PASS

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.

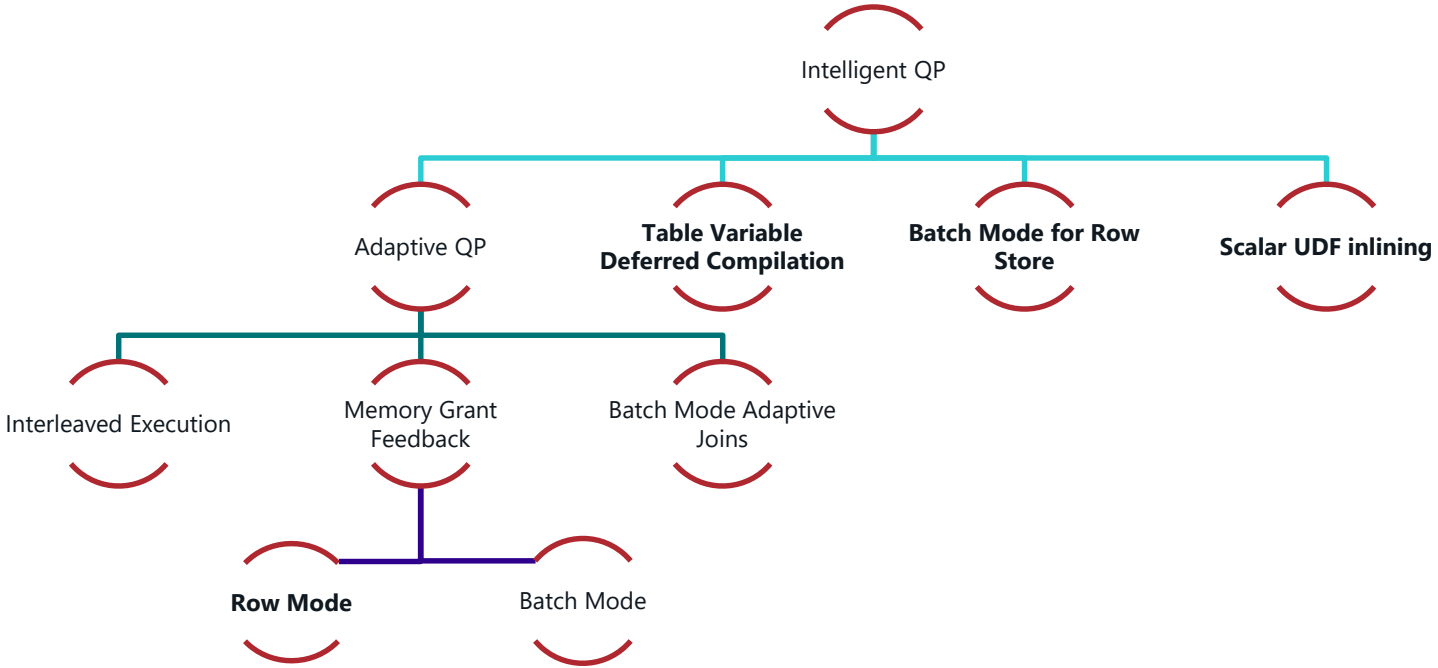
CX Consumer

- 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

PASS

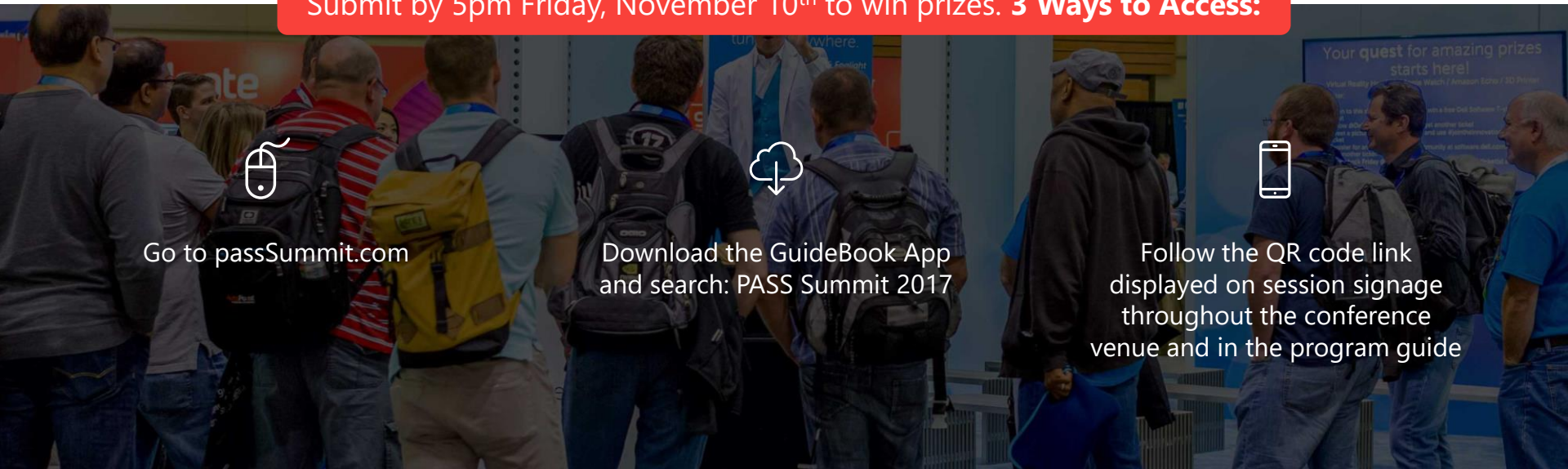
Useful links

- 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

Session evaluations

Your feedback is important and valuable.

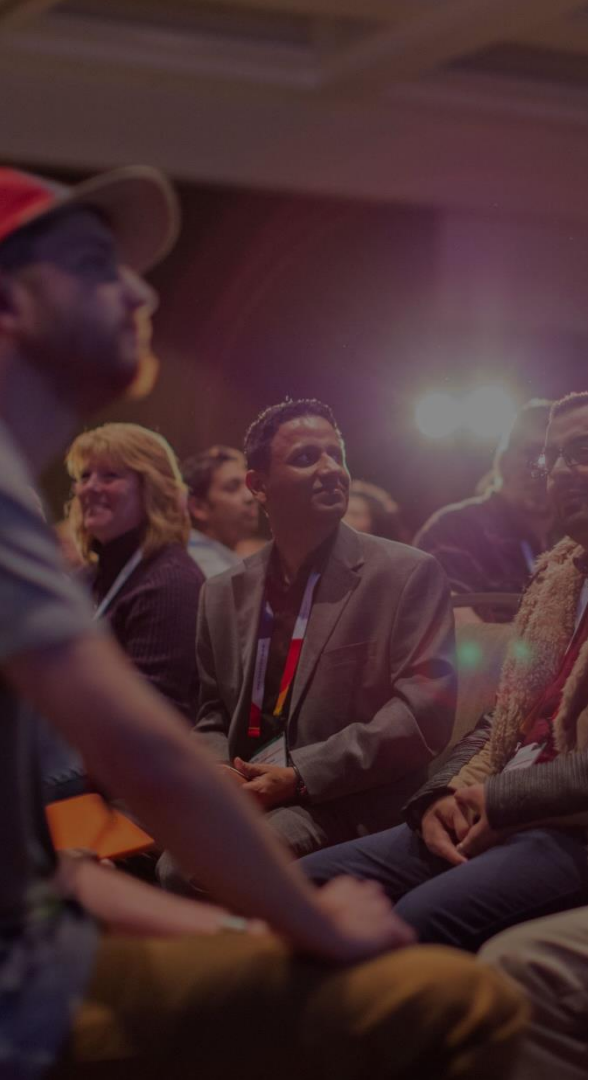
Submit by 5pm Friday, November 10th to win prizes. **3 Ways to Access:**



Go to 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

PASS