Enhancements that will make your SQL database engine roar Part 1

Pedro Lopes (@sqlpto)
Senior Program Manager
Data Group

SQL Server Tiger Team

Microsoft

Pedro Lopes

@sqlpto

pedro.lopes@microsoft.com

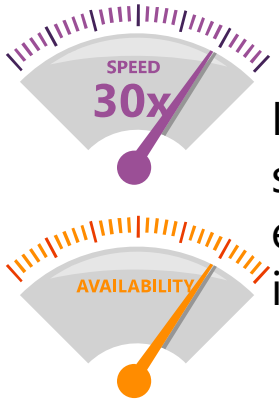
Senior Program Manager

Focused on SQL Server Relational Engine

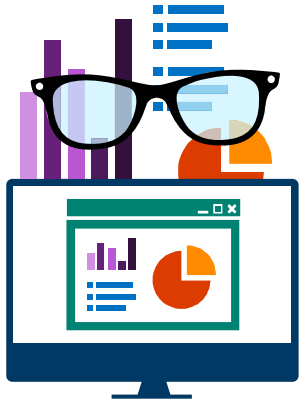
7 years at Microsoft

Session Objectives And Takeaways

SQL Server Tiger Team



It just works - performance and scale in SQL Server 2016 database engine and what is being added to in-market versions



Show new diagnostics improvements for SQL Server engine

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

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

Agenda

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

SQL Server is a Tier-1 RDBMS

Too many knobs

- Trace Flags
- Configuration Options

Downtime for database maintenance

- Database integrity checks
- Partition changes
- Schema changes

Repeated data collection

- Insufficient or unstructured diagnostics

Top customer feedback areas

Perf and Scale

- Tempdb
- Query Compilation
- SOS_RW Lock
- Memory allocation
- Auto Soft NUMA
- Large systems
- Multiple Log Writers

Availability

- Schema changes
- Partition management
- Always On

Diagnostics

- Integrity Checks
- Query Progress
- Query Execution
- Always On
- Backup and Restore
- Recovery
- SQLDumper
- xEvents
- Showplan

Performance and Scale

TempDB – the current experience

Allocation latch contention

- PFS, GAM, SGAM
- Solution: Trace flags (1117,1118) and Multiple files

Metadata latch contention

- High create/drop workloads
- LATCH_EX waits on underlying system tables
- Solution: Rewrite t-sql code to reduce temp tables

TempDB – the new experience

Trace flags removed

- 1117 and 1118 behavior will be enabled by default for tempdb

Improved scanning algorithms

- Reduces metadata contention
- Optimistic locking of system tables under shared latch

New defaults*

- Setup experience
- Size and auto growth

Autogrow and Allocations for user DBs

Trace flags removed

- New extensions in ALTER DATABASE commands

1118

- ALTER DATABASE <dbname> SET MIXED_PAGE_ALLOCATION { ON | OFF }
- Default value of the MIXED_PAGE_ALLOCATION is OFF
- New column in sys.databases (is_mixed_page_allocation)

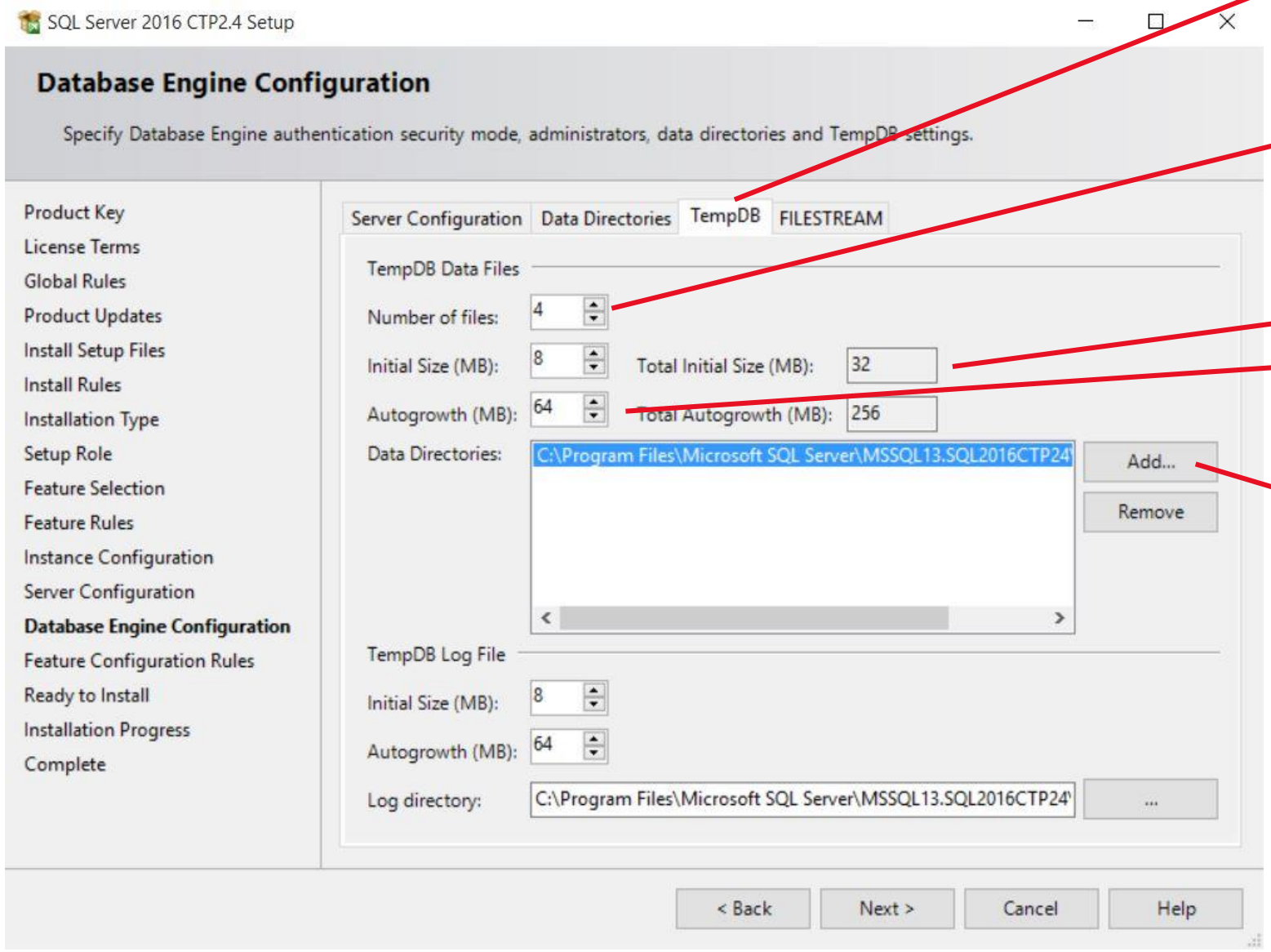
1117

- ALTER DATABASE <dbname> MODIFY FILEGROUP <filegroup> { AUTOGROW_ALL_FILES | AUTOGROW_SINGLE_FILE }
- Default value is AUTOGROW_SINGLE_FILE for all files in all filegroups
- New column in sys.filegroups (is_autogrow_all_files)

Summary for Autogrow and Allocations

Database	TF 1117	TF 1118
TempDB	Not required (default)	Not required (default)
User Databases	Default behavior will grow single file. Use ALTER DATABASE <dbname> MODIFY FILEGROUP [PRIMARY] AUTOGROW_ALL_FILES to grow all files in the filegroup.	Not required (default). Use ALTER DATABASE <dbname> SET MIXED_PAGE_ALLOCATION ON to go back to using mixed extents
System Databases	N.A.	Allocations use mixed page extents, cannot be changed.

TempDB – setup



Separate tab for tempdb in setup

Number of data files – max (8, number of cores)

Recommend initial – 32MB and autogrow – 64MB

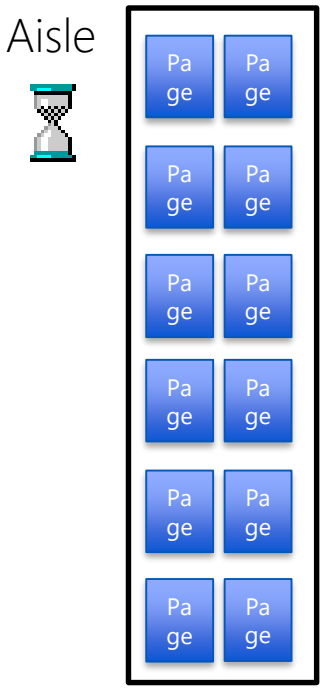
Specify multiple volumes, setup will round-robin the data files

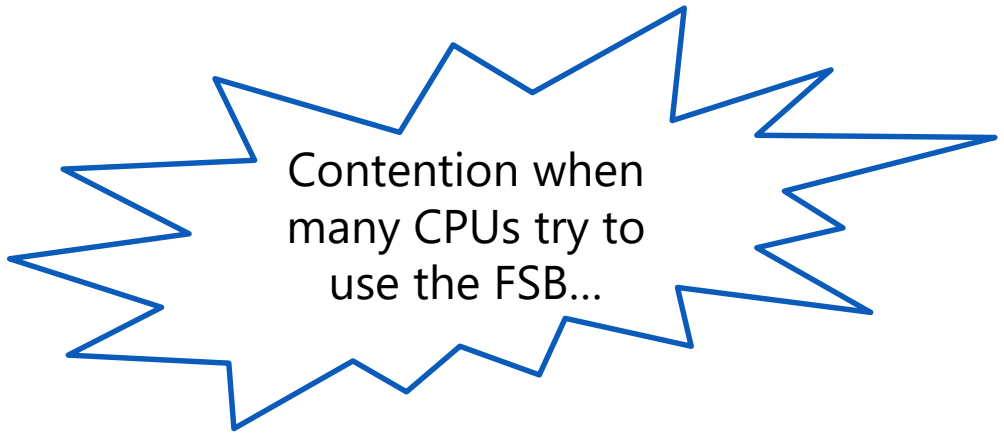
Instant File Initialization requires permission SE_MANAGE_VOLUME

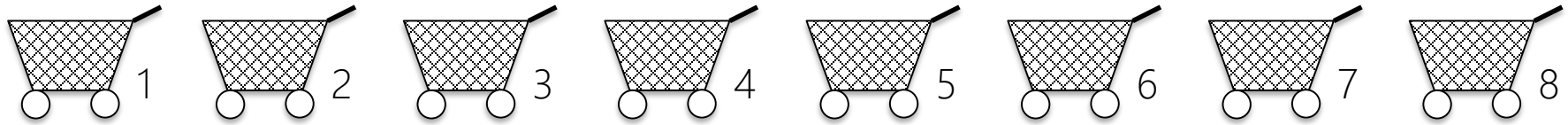
SMP in a nutshell

Imagine a supermarket with a single **aisle** (front-side bus), which has all the **products** (memory pages), and all **shopping carts** (CPUs) have to queue to get their designated product.

How fast can **shopping carts** access **products**?



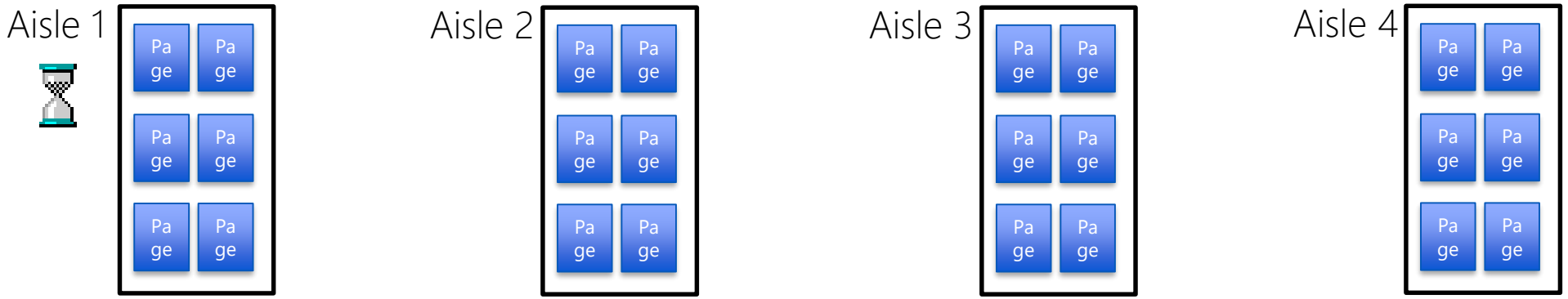




NUMA in a nutshell

Now imagine a supermarket with 4 **aisles** (1 bus per NUMA node, 4 nodes), with the **products** (memory pages) evenly distributed.

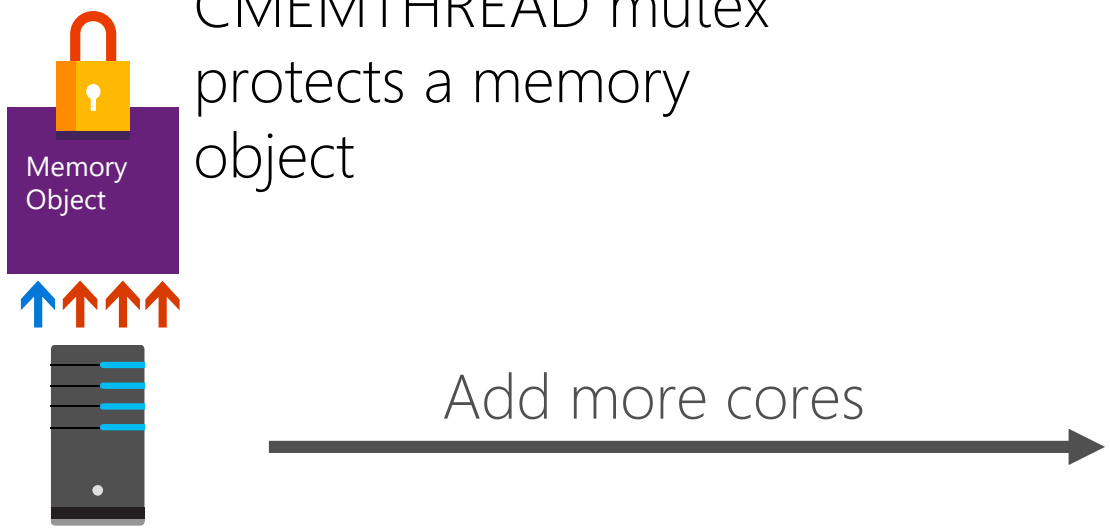
How fast can **shopping carts** access **products** now?

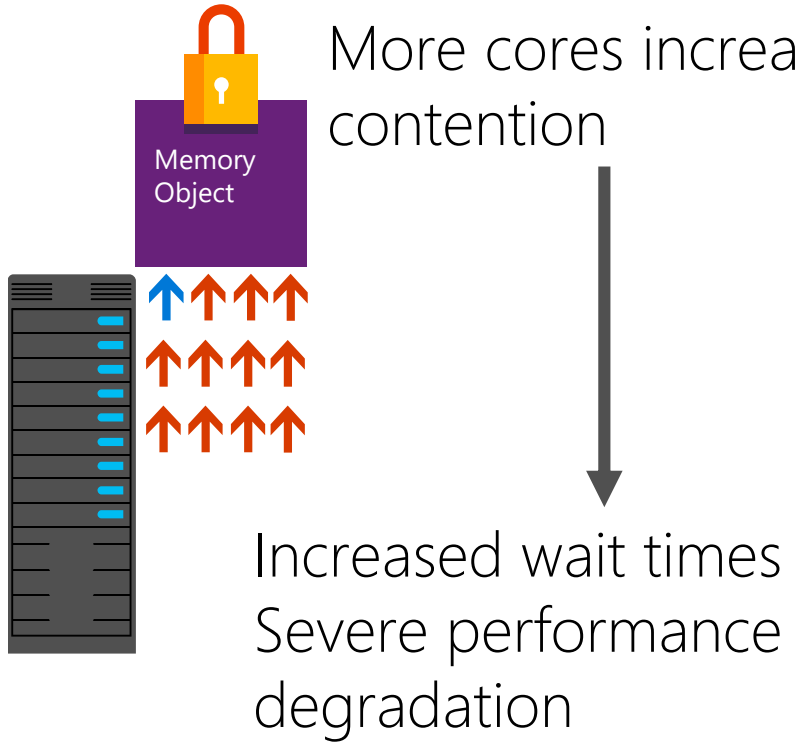


Large Systems support

- SQL Server 2016 and 2014 SP2
- Auto soft NUMA for large number of cores per socket
  - During startup, SQL Server interrogates the hardware layout and automatically configures Soft NUMA on systems reporting 8 or more CPUs per NUMA node.
  - Hyperthread (HT/logical processor) aware.
- SQL Error log
  - "Automatic soft-NUMA was enabled because SQL Server has detected hardware NUMA nodes with greater than 8 logical processors."
- DMV
  - New column in sys.dm_os_sys_info (softnuma_configuration_desc) can have one of the three values: OFF / ON / MANUAL

Memory allocation – current experience



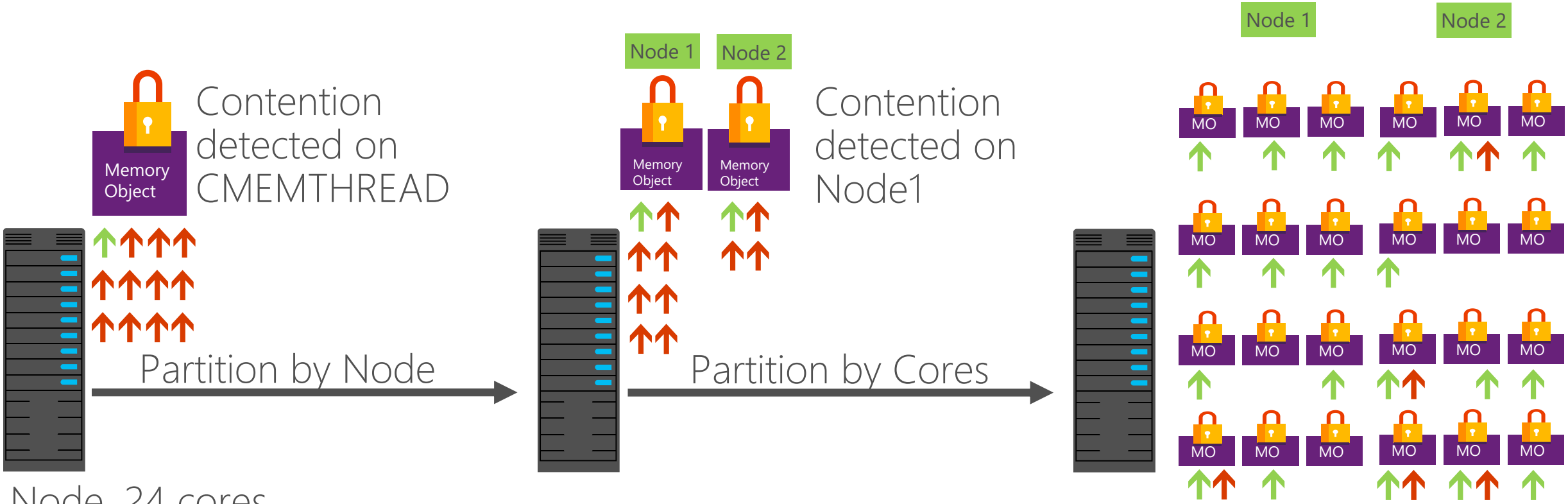


- Common contention objects
  - CMEMTHREAD (TF 8[illegible]8)
  - LOGPOOL (TF 90[illegible])

Gone in SQL Server 2016

Memory allocation – new experience

SQL Server 2016 and 2014 SP2



2 Node, 24 cores

12 cores per node

Dynamic Partitioning of Memory Objects

- Contention factor
  - Current contention/temperature of the CMemThread
  - If average is 1 or more outstanding waits per each of the last 100,000 allocations, then promote
- Diagnostics
  - Wait_type ‘CMEMTHREAD’ in sys.dm_os_wait_stats DMV
  - Columns in sys.dm_os_memory_objects DMV (contention_factor, partition_type, exclusive_allocations_count, waiting_tasks_count)
  - New sqlos.pmo_promotion Extended Event (fired during a promotion)

Demo

Dynamic Partitioning of Memory Objects

Note: Video demo removed for space saving

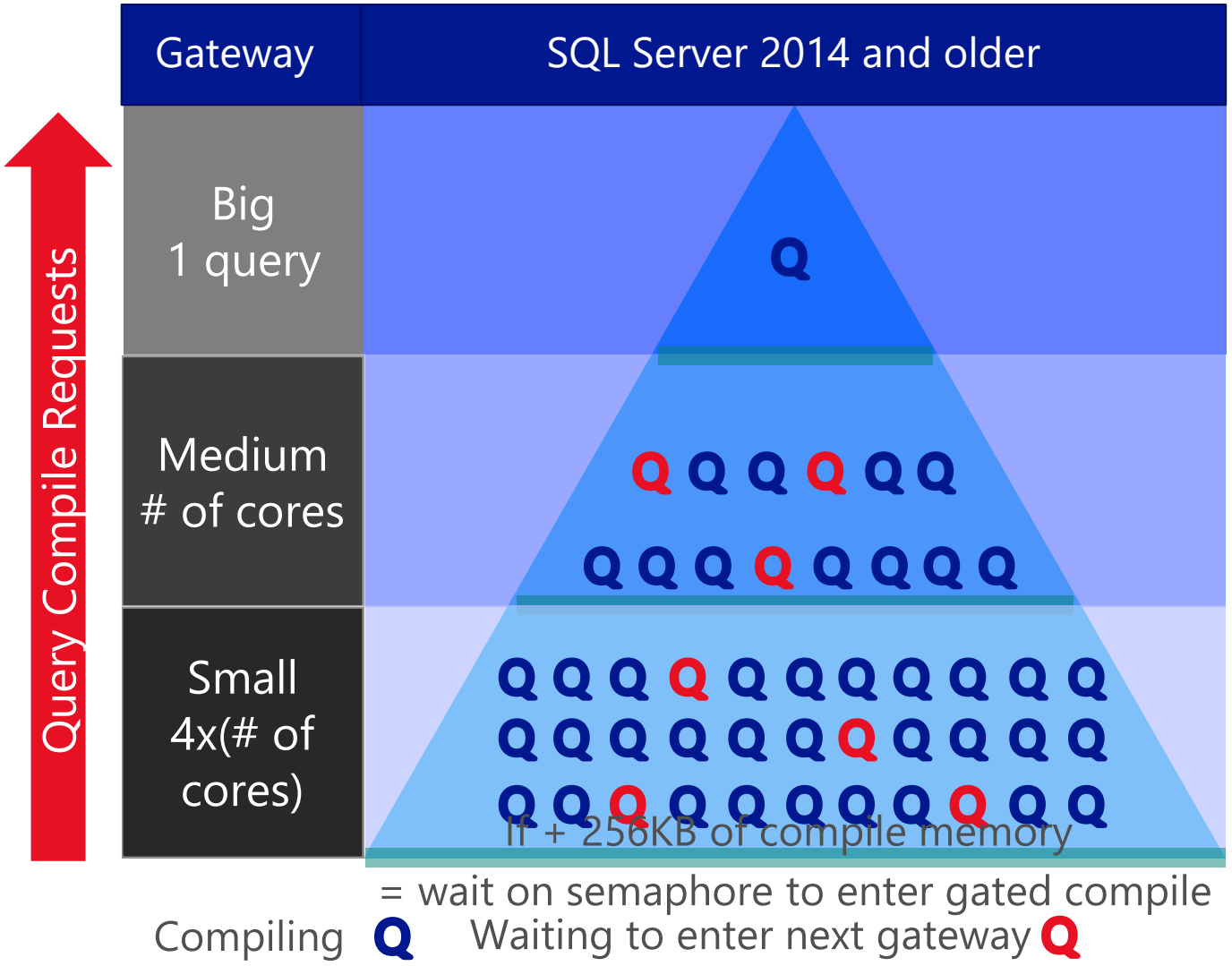
Online Operations

- ALTER COLUMN
  - Table is unavailable due to blocking
  - Massive amount of log - size of data operation
  - Leverages Online Index Build (OIB) infrastructure
  - Only needs schema lock at the end
  - Rollback from failure as simple as dropping the new version

```
CREATE TABLE dbo.doc_exy (C1 INT, C2
varchar(50) NULL, C3 decimal (4,2));
--Change Type
ALTER TABLE dbo.doc_exy
ALTER COLUMN C1 DECIMAL (5, 2) WITH
(ONLINE = ON)
--Change collation
ALTER TABLE dbo.doc_exy
ALTER COLUMN C2 varchar(50) COLLATE
Latin1_General_BIN WITH (ONLINE = ON)
 -- Increase the scale and precision of
the decimal column.
ALTER TABLE dbo.doc_exy ALTER COLUMN C3
decimal (10,4) WITH (ONLINE = ON)
--Others (Nullability, Sparseness)

TRUNCATE TABLE PartitionTable1 WITH
(PARTITIONS (6 TO 8));
```

Query Compilation – Big Gateway



- Servers with large amount of RAM
- RESOURCE_SEMAPHORE_QUERY_COMPILE waits
- Concurrent large compilation requests blocked - current Big Gateway policy of 1 big query

New in SQL Server 2016 and 2014 SP2

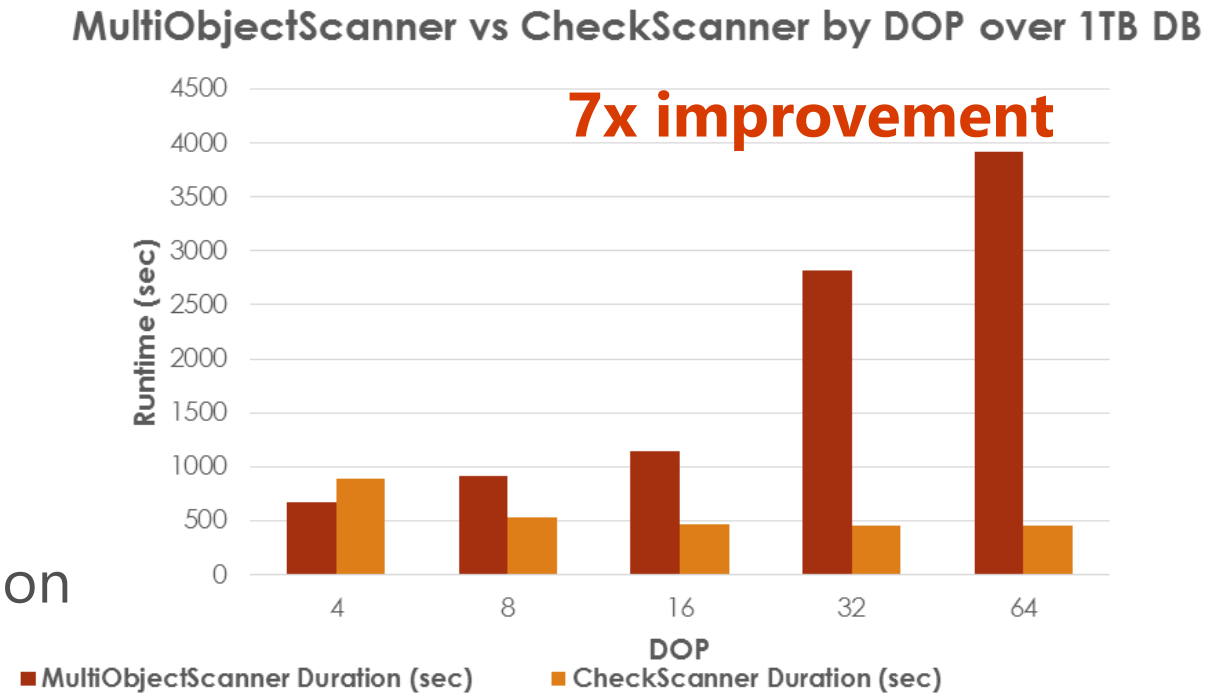
- Dynamically adjust Big Gateway threshold
- Allows concurrent big query compiles on large memory systems
- `sys.dm_exec_query_optimizer_memory_gateways`

pool_id	name	max_count
1	Small Gateway	96
1	Medium Gateway	24
1	Big Gateway	5 *

Diagnostics and Management

CHECKDB

- Slow consistency checks
  - Expensive Expression evaluation
- Data types moved to EXTENDED_LOGICAL_CHECKS
  - Filtered indexes
  - Persisted Computed columns
  - UDT columns and UDT columns on based on CLR assemblies
- New techniques for faster checks
  - CheckScanner using lock free design (similar to Hekaton)
  - MAXDOP option (SQL Server 2016 and 2014 SP2)



New Memory Grant Showplan Warning

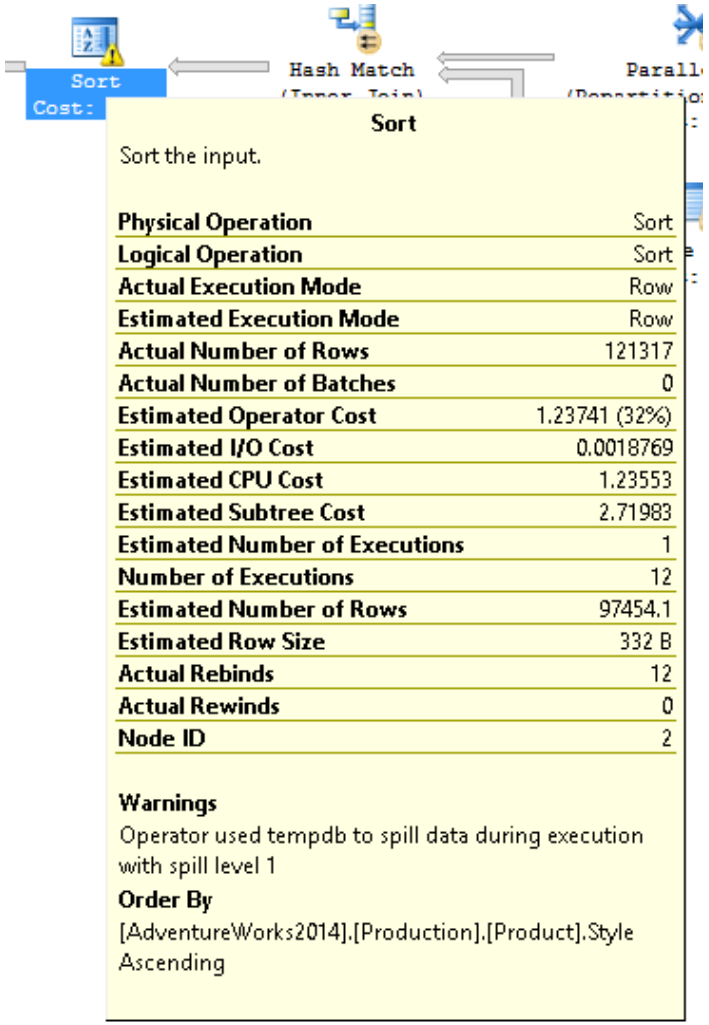
- 3 conditions:
  - Under used: when the max used size is too small compared to the grant size. This can cause blocking and less efficient usage.
  - Over used: when the used size exceeds the grant size. This can cause out of memory on the server.
  - Excessive growth: when the dynamic grant increases too much. This can cause server instability and unpredictable workload performance.
- SQL Server 2016 post-RTM and SQL Server 2014 SP2

New Spills Warnings - Sort

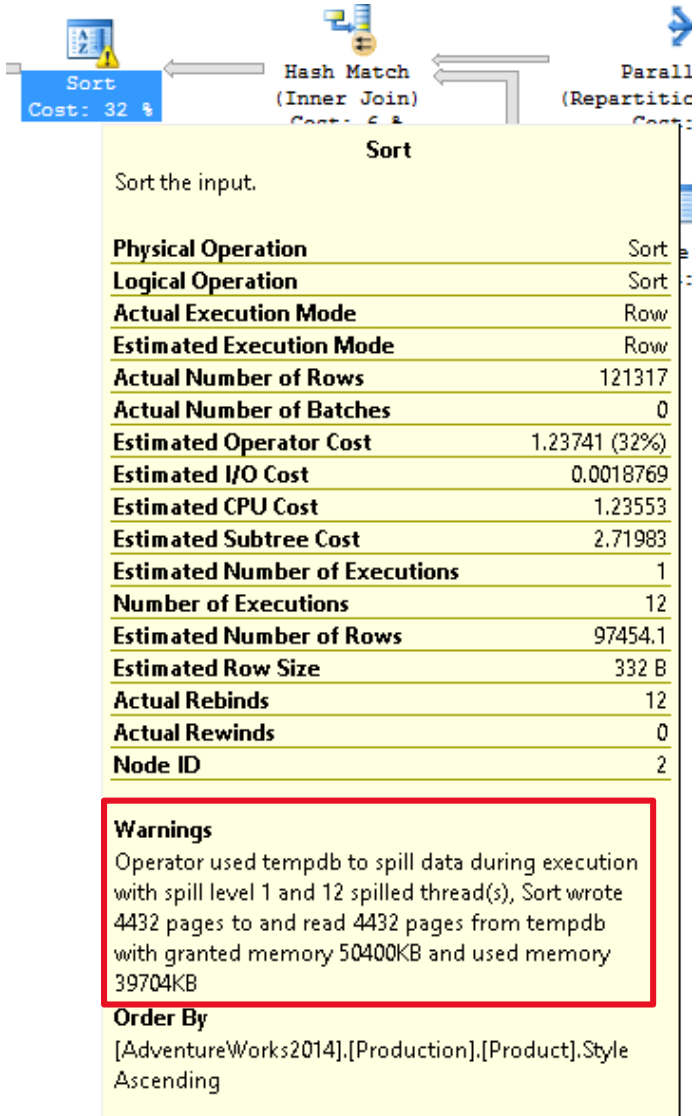
- Sort Spills = sort operations that do not fit into memory
  - Supported by a Worktable in TempDB
- Spill level 1
  - Means one pass over the data was enough to complete the sort.
- Spill level 2
  - Means multiple passes over the data are required to sort the data

New Spills Warnings - Sort

Up to SQL Server 2016



SQL Server 2016 and 2014 SP2

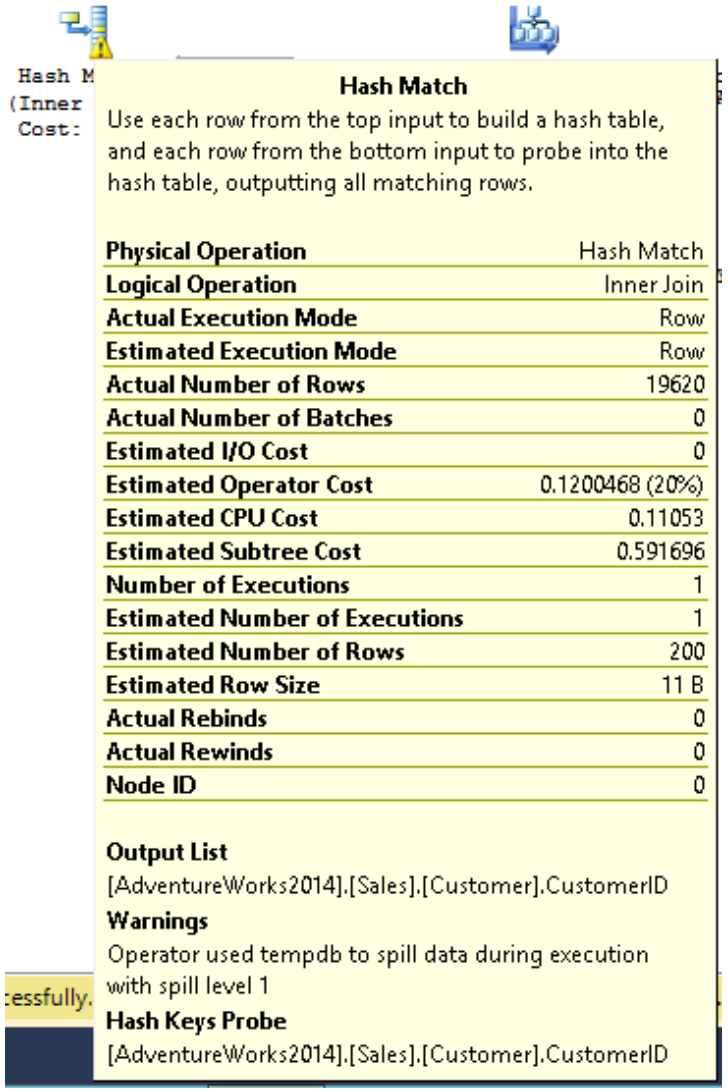


New Spills Warnings - Hash

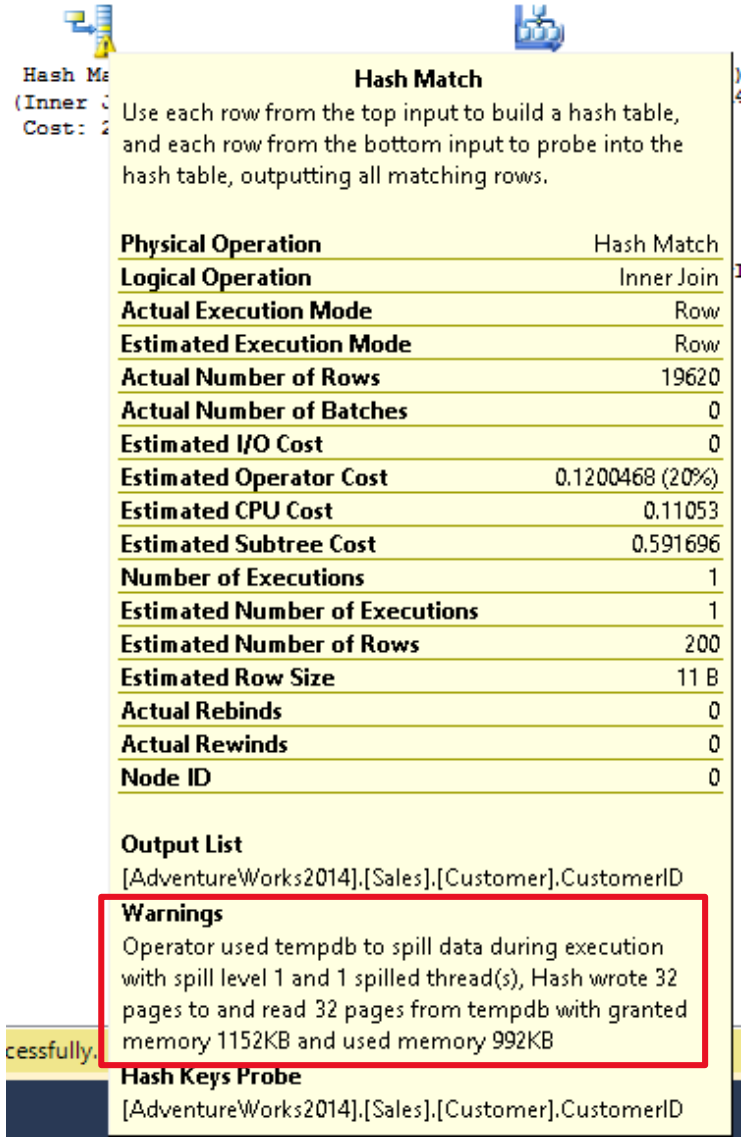
- Hash Spills = hash recursion or cessation of hashing (hash bailout) has occurred during a hashing operation
  - Supported by a Workfile in TempDB
- Spill level 1 = Hash recursion
  - Occurs when the build input does not fit into available memory, resulting in the split of input into multiple partitions that are processed separately.
  - If any of these partitions still do not fit into available memory, it is split into sub-partitions, which are also processed separately. This splitting process continues until each partition fits into available memory or until the maximum recursion level is reached.
- Spill level 2 = Hash bailout
  - Occurs when a hashing operation reaches its maximum recursion level and shifts to an alternate plan to process the remaining partitioned data.

New Spills Warnings - Hash

Up to SQL Server 2016

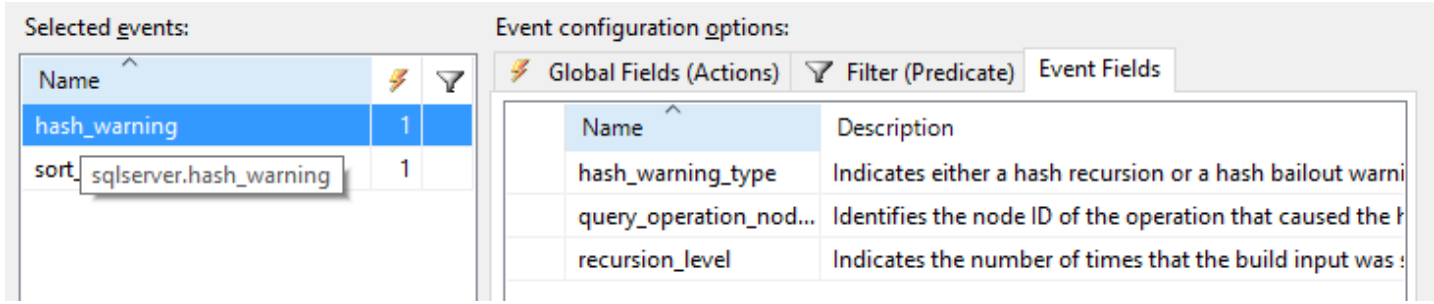


SQL Server 2016 and 2014 SP2

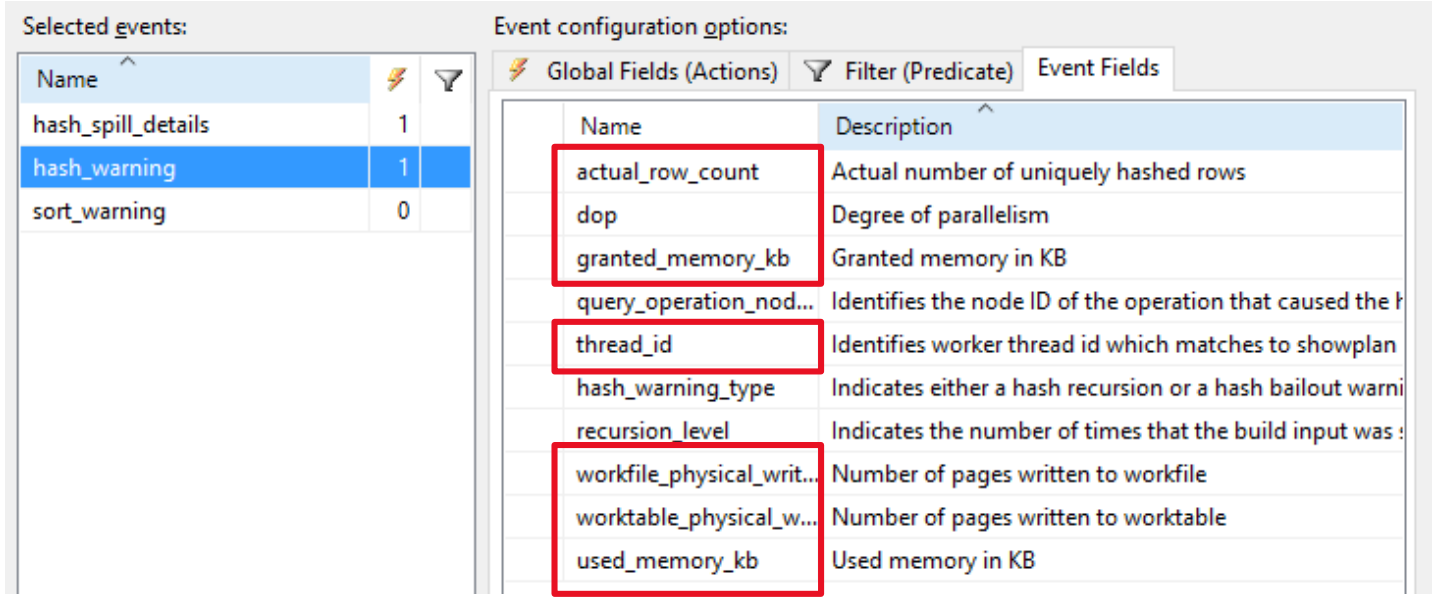


Spill xEvents - Hash Warning

Up to SQL Server 2016

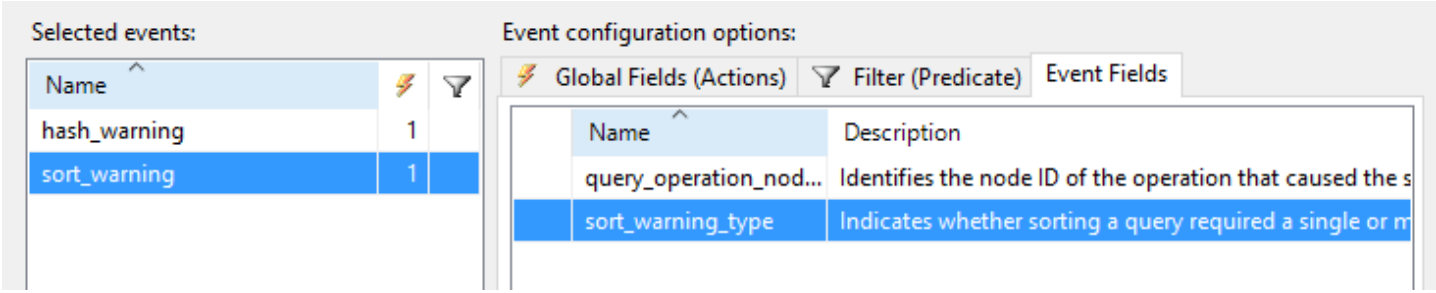


SQL Server 2016 SQL Server 2014 SP2

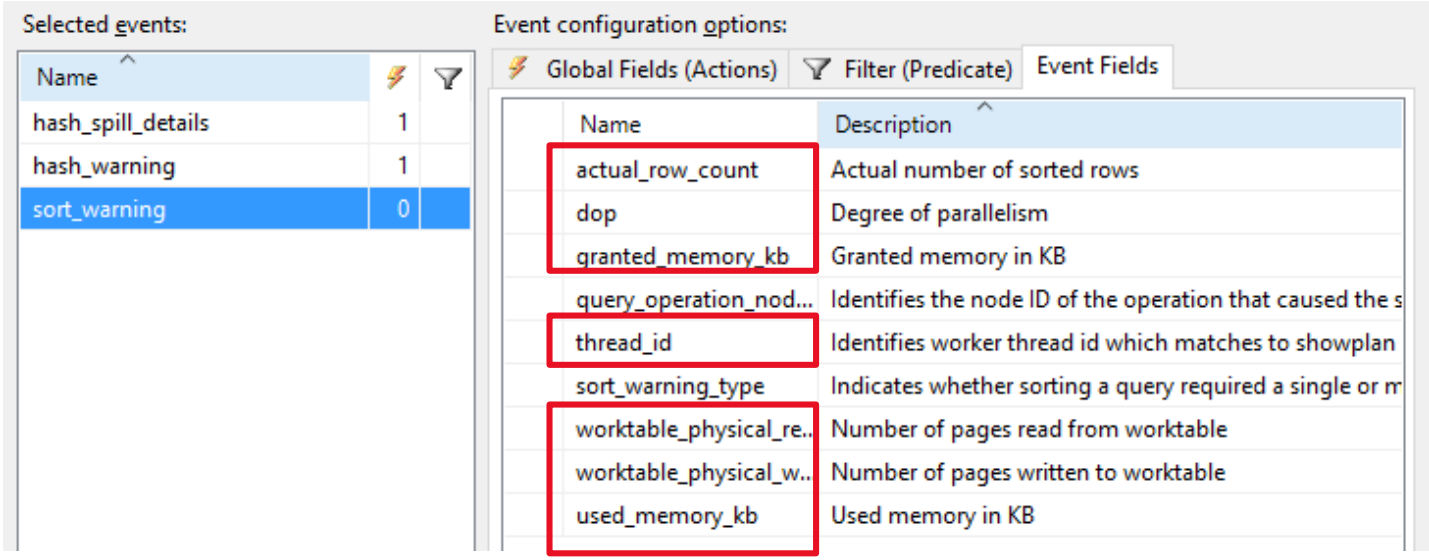


Spill xEvents - Sort Warning

Up to SQL Server 2016

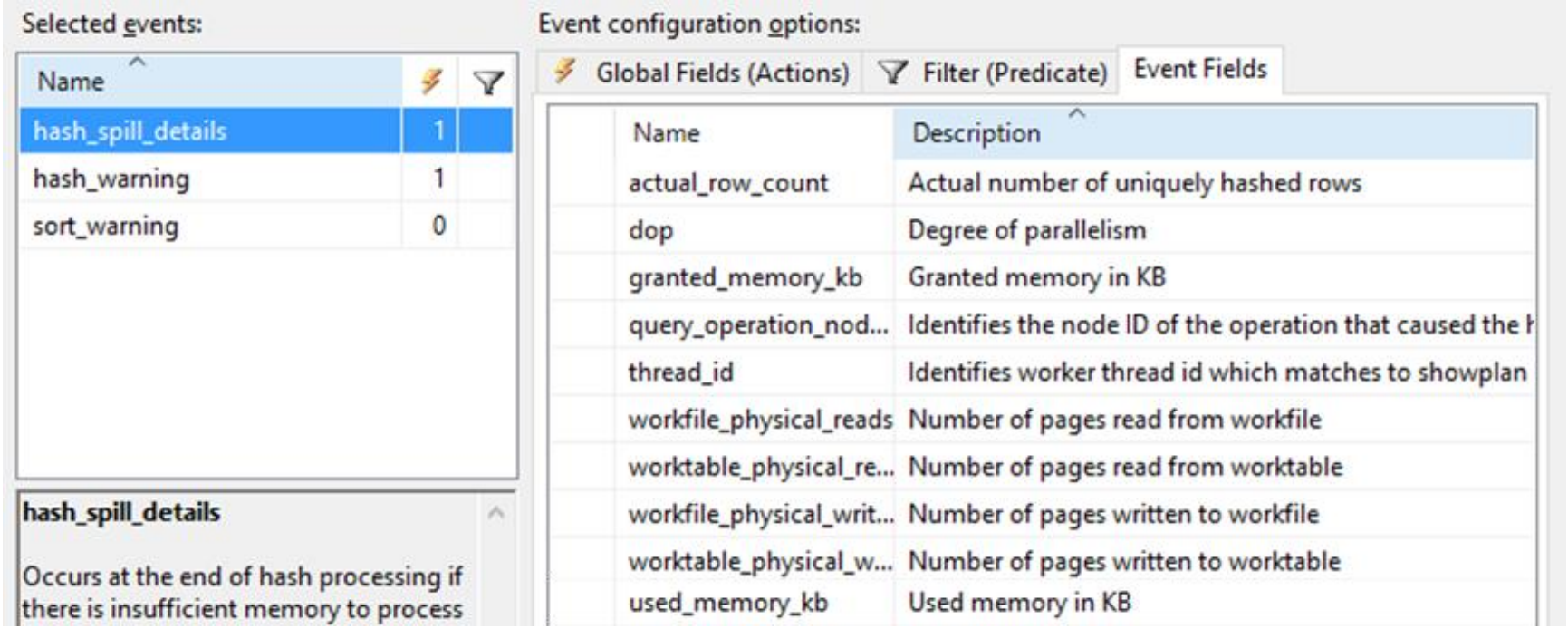


SQL Server 2016
SQL Server 2014 SP2



Spill xEvents - Hash details

- New Extended Event hash_spill_details
  - Triggered at the end of hash processing.
  - Use this together with any of the *query_pre_execution_showplan* or *query_post_execution_showplan* events to determine which operation in the generated plan is causing the hash spill.



Demo

Spill warnings and extended events

Survey, decks and demos:
Part 1: http://speakerscore.com/Roar1

There is a Part #2 – starting at 2pm