SQL Server 2016 Performance and Scalability Improvements

Marko Hotti
Sr. Technical Product Manager / SQL Server
Microsoft Corporation
marko.hotti@microsoft.com

Record-breaking performance

Leader in OLTP price/performance

#1 in TPC-E price/perf (44 cores) Windows Server 2012 R2 — FUJITSU

#2 in TPC-E price/perf (44 cores) Windows Server 2012 R2 — Lenovo

Leader in DW performance & price/performance

#1 in TPC-H 30TB (144 cores) **Windows Server 2016** — Lenovo

38% faster TPC-H 3TB (72 cores) Windows Server 2012 R2 — Lenovo

#1 and 40% faster in TPC-H 1TB (44 cores) Windows Server 2012 R2 — hp

Performance gains just by upgrading

Faster queries

3.6x faster DW queries on AlwaysOn replicas

15x faster queries Table valued parameters

271x faster queries Spatial line string

34x faster queries In-memory Columnstore

19x faster Spatial native Functions

3x faster Spatial index/ queries up to

2.6x faster DW queries with new CE

Faster operations

10x faster In-Memory temp objects

20% faster Auto NUMA partitioning

10x faster XEvent reader

7-10x faster DBCC CheckDB

20x faster Indirect checkpoint

100x faster Batch/Bulk operations

Faster throughput

7x faster throughput AlwaysOn

2.7x faster Batch request per sec

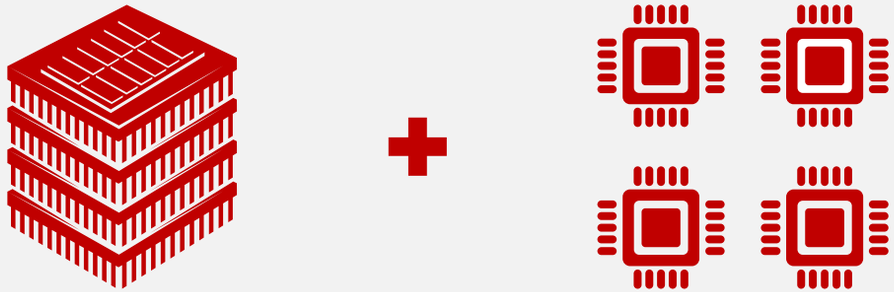
Game-changing app performance

PROS **1000%** faster scoring	Global ERP **700%** faster queries
Tableau **190%** faster queries	KPMG **250%** faster execution

TPC results as of April 2016 http://www.tpc.org/default.asp

SQL Server 2016 & Windows Server 2016 Better Together

Unparalleled scalability
with Windows Server 2016



12 TB of memory

WS 2016 max cores

Massive scale for in-memory performance

Simple, flexible HA and DR

No domain join needed

Unparalleled security

Fine-grained security controls

Built-in anti-malware

Microsoft Storage Spaces Direct

What is Storage Spaces Direct?

- Evolution of Storage Spaces
- Servers with local storage
- Highly available and scalable
- Storage for Hyper-V virtualization and private cloud

Why Storage Spaces Direct?

New device types

- Lower-cost flash storage with SATA SSDs
- Better flash performance with NVMe SSDs

Simplicity

- Ethernet/RDMA network as storage fabric
- No need for complex multi-initiator fabric
- Seamless capacity and performance expansion

Support for Windows Server Core

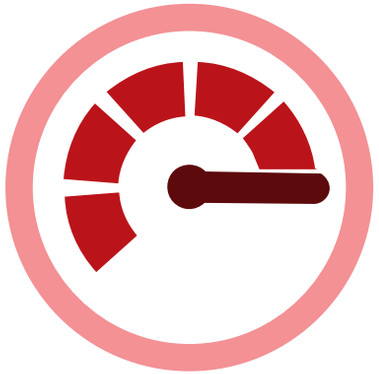
Windows Server edition with smallest footprint

- Reduced memory and disk requirements
- Fewer running processes and services: greater stability
- Simplified management

Requires less maintenance and fewer OS patches, greatly reduced downtime

50–60 percent less patching and fewer OS reboots

In-Memory OLTP enhancements



In-Memory OLTP

Benefits

Low latency

Up to 30 times the improvement in performance

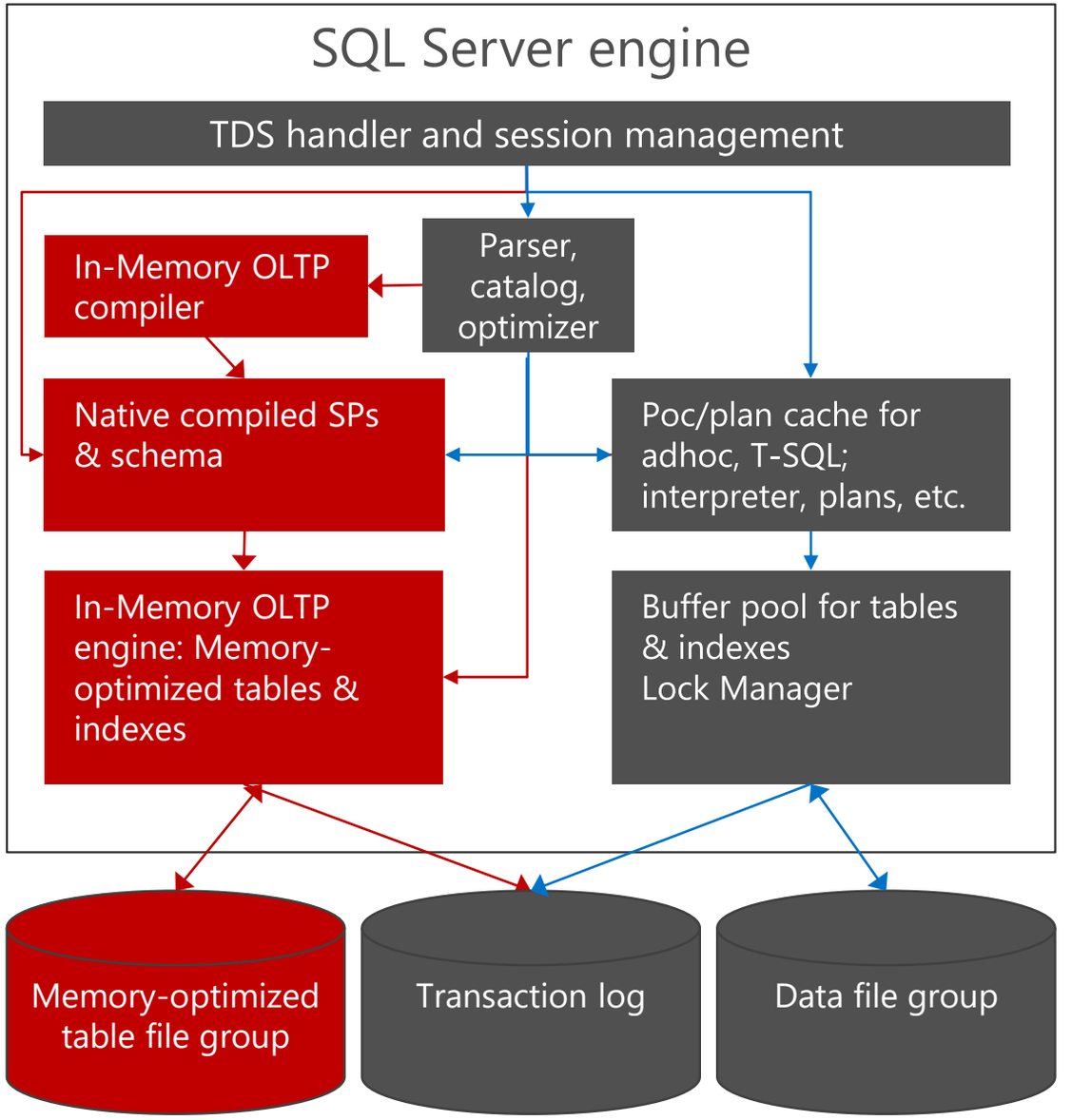
2 to 5 times the improvement in scalability

Takes advantage of investments in Microsoft SQL Server

How it works

New high-performance, memory-optimized online transaction processing (OLTP) engine integrated into SQL Server and architected for modern hardware trends

- Integrated into SQL Server relational database
- Full ACID support
- Memory-optimized
- Non blocking multi-version optimistic concurrency control (no locks or latches)
- T-SQL compiled to native code



Performance and Scaling Improvements

Supports up to 2 TB of user data in durable memory optimized tables in a single database.

Multiple threads to persist memory-optimized tables

Parallel Support

- Parallel scan for memory-optimized tables and HASH indexes
- Parallel plan support for accessing memory-optimized tables

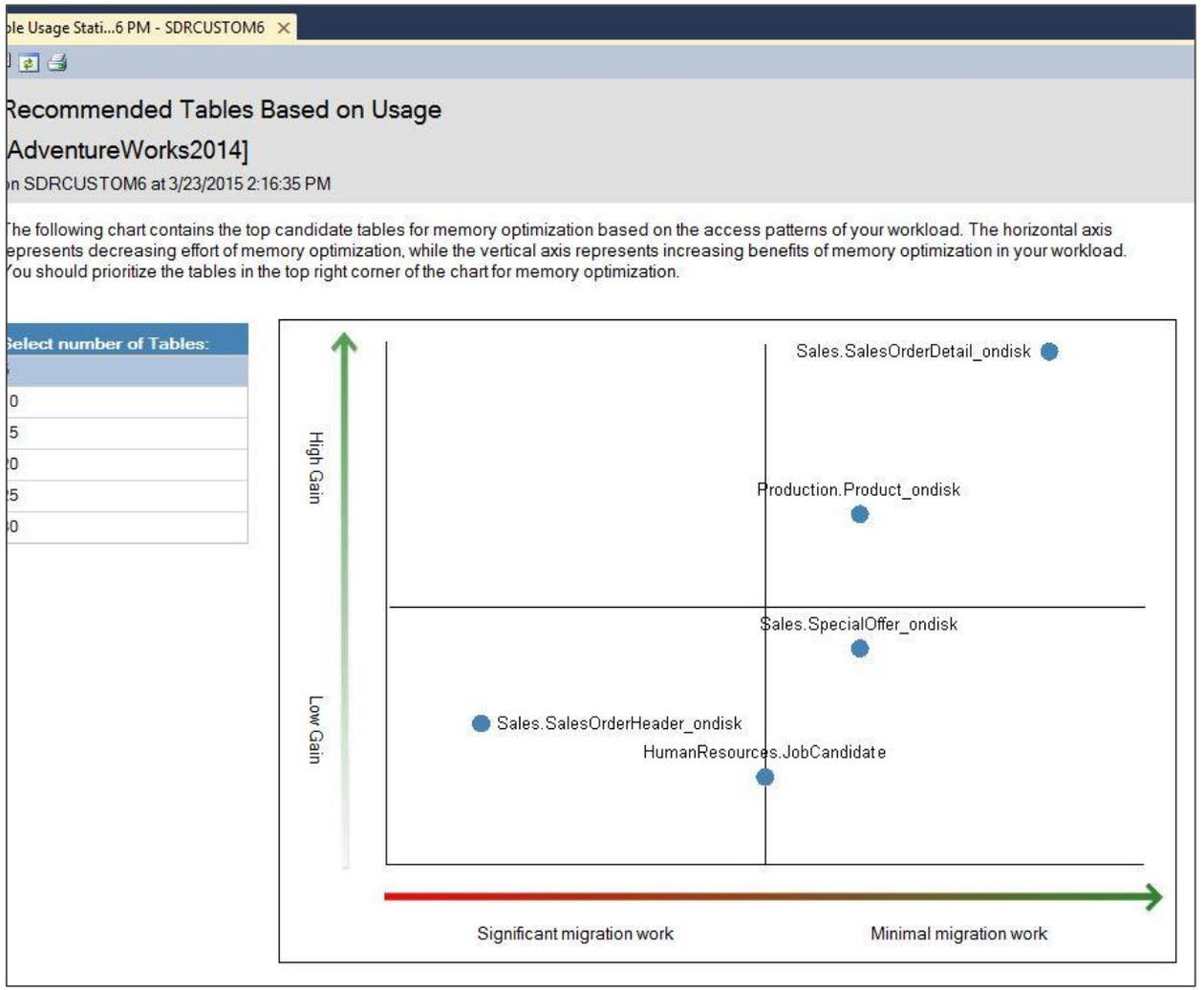
Improved scaling

Other enhancements include:

- Multiple threads to persist memory-optimized tables
- Multi-threaded recovery
- MERGE operation
- Dynamic management view improvements to `sys.dm_db_xtp_checkpoint_stats` and `sys.dm_db_xtp_checkpoint_files`
- `DBCC CHECKDB` performance changed to support 1 TB database check improvement by **7x**

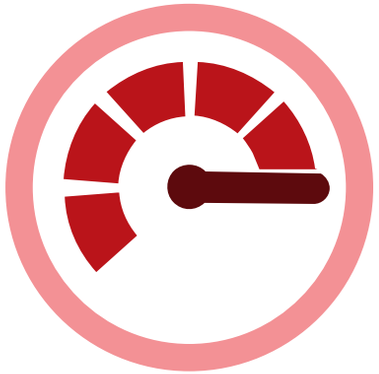
In-Memory OLTP engine has been enhanced to scale linearly on servers up to 4 sockets

New Transaction Performance Analysis Overview report

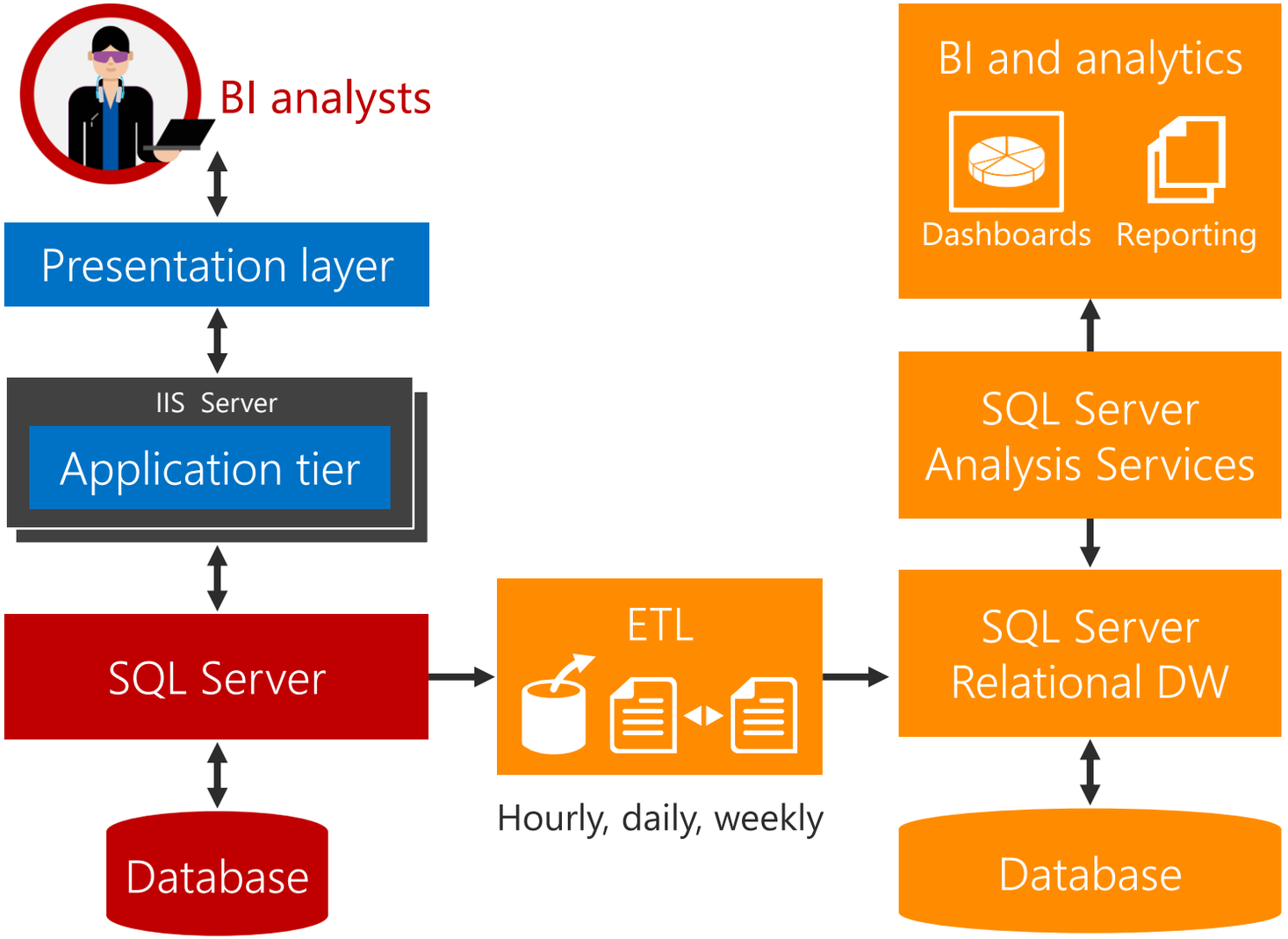


New report replaces the need to use the Management Data Warehouse to analyze which tables and stored procedures are candidates for in-memory optimization

Operational analytics: disk-based and in-memory tables



Traditional operational/analytics architecture



Key issues

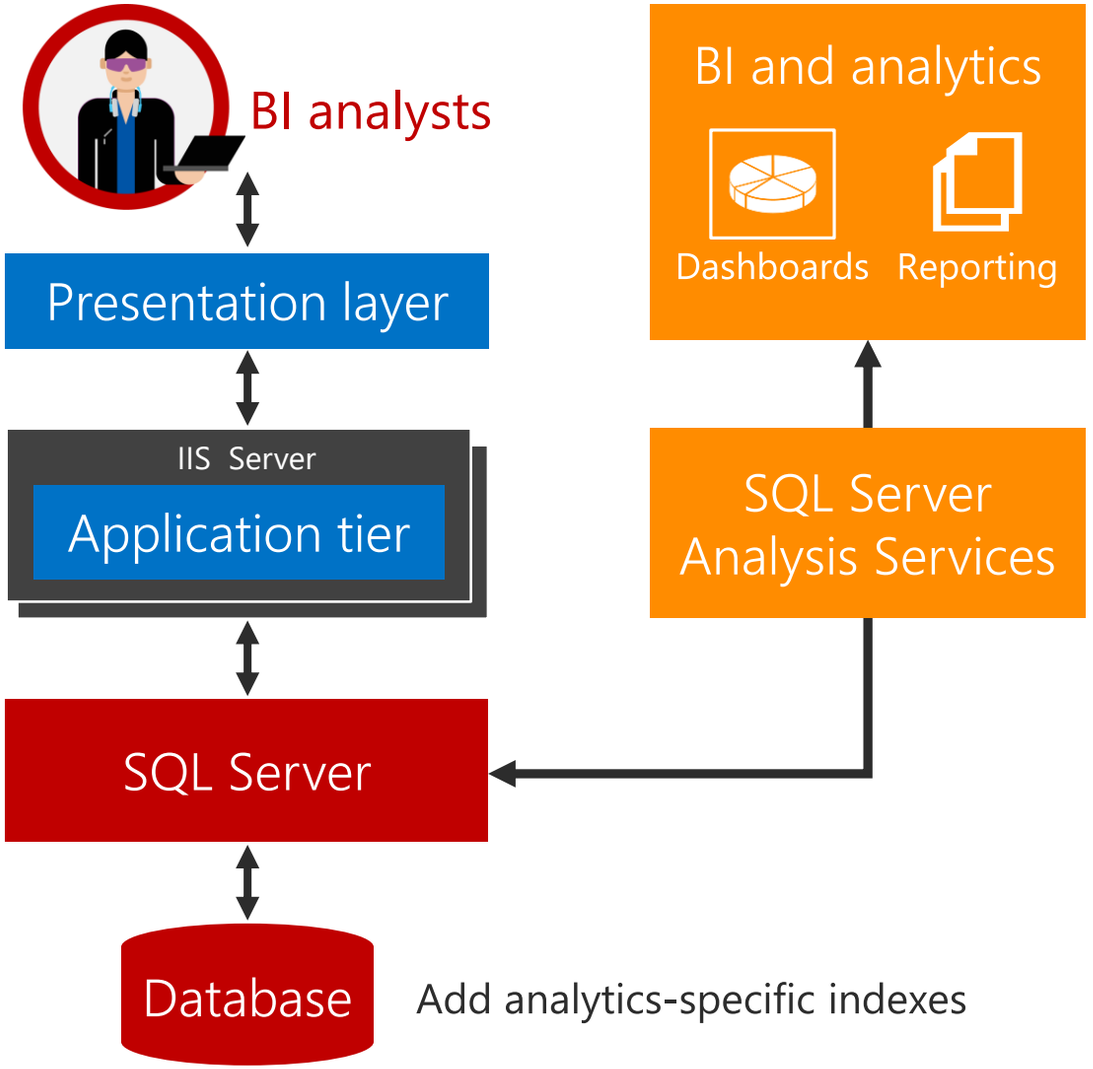
Complex implementation

Requires two servers (capital expenditures and operational expenditures)

Data latency in analytics

High demand;
requires real-time analytics

Minimizing data latency for analytics



Challenges

Analytics queries are resource intensive and can cause blocking

Minimizing impact on operational workloads

Sub-optimal execution of analytics on relational schema

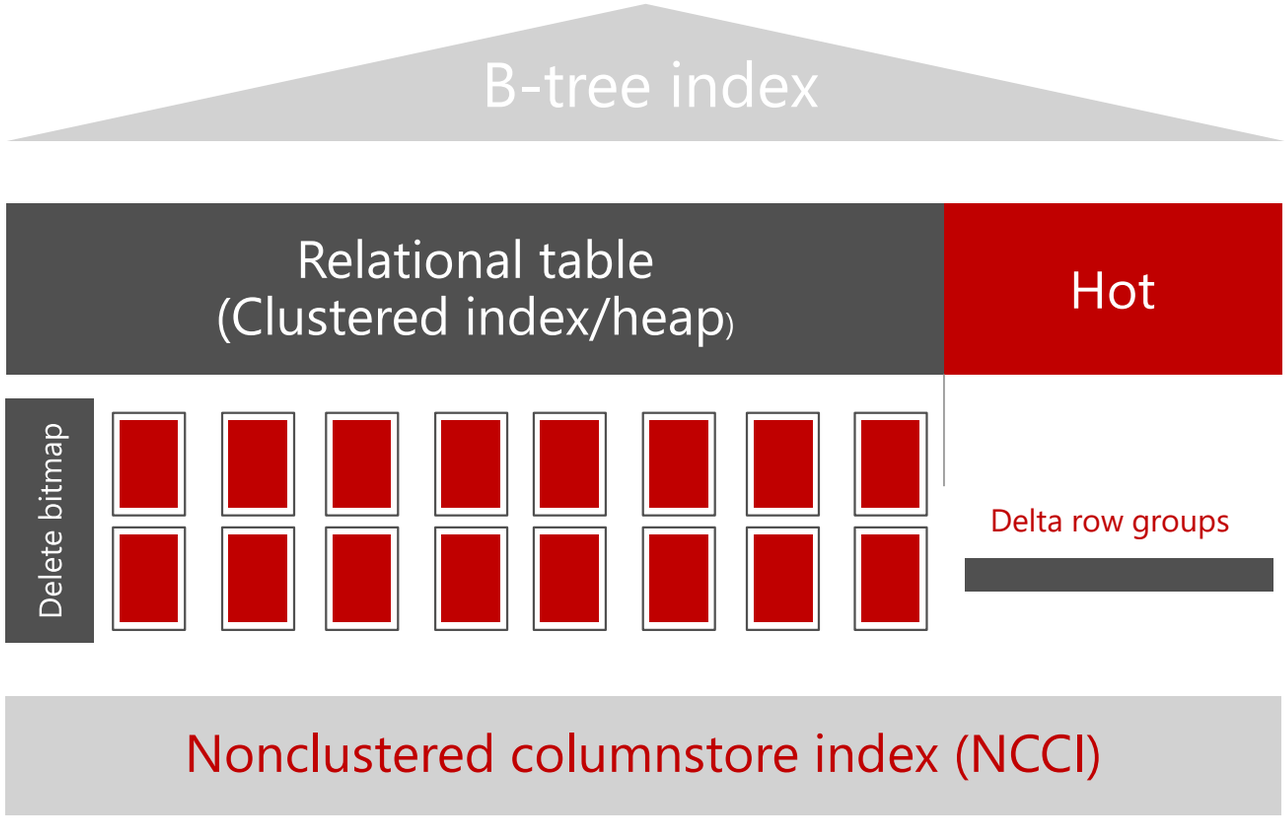
Benefits

No data latency

No ETL

No separate data warehouse

Operational analytics with columnstore index



Key points

Create an updateable NCCI for analytics queries

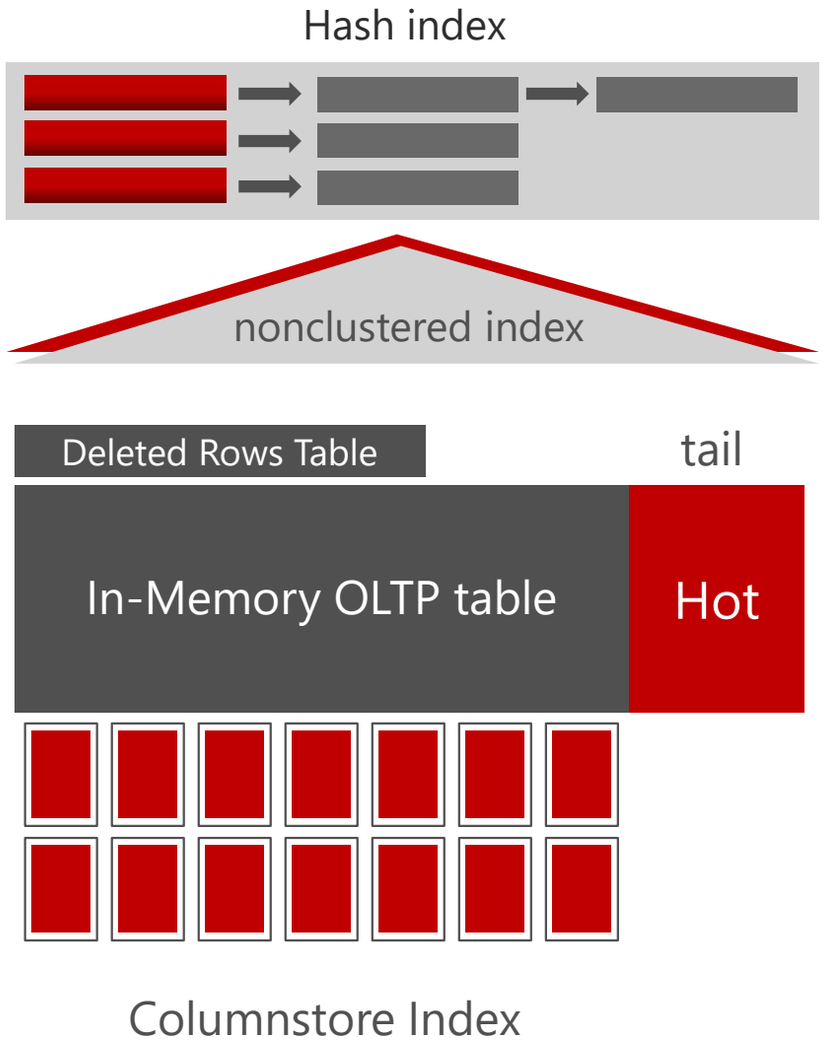
Drop all other indexes that were created for analytics

No application changes

Columnstore index is maintained just like any other index

Query optimizer will choose columnstore index where needed

Operational analytics: columnstore on in-memory tables



No explicit delta row group

Rows (tail) not in columnstore stay in In-Memory OLTP table

No columnstore index overhead when operating on tail

Background task migrates rows from tail to columnstore in chunks of 1 million rows

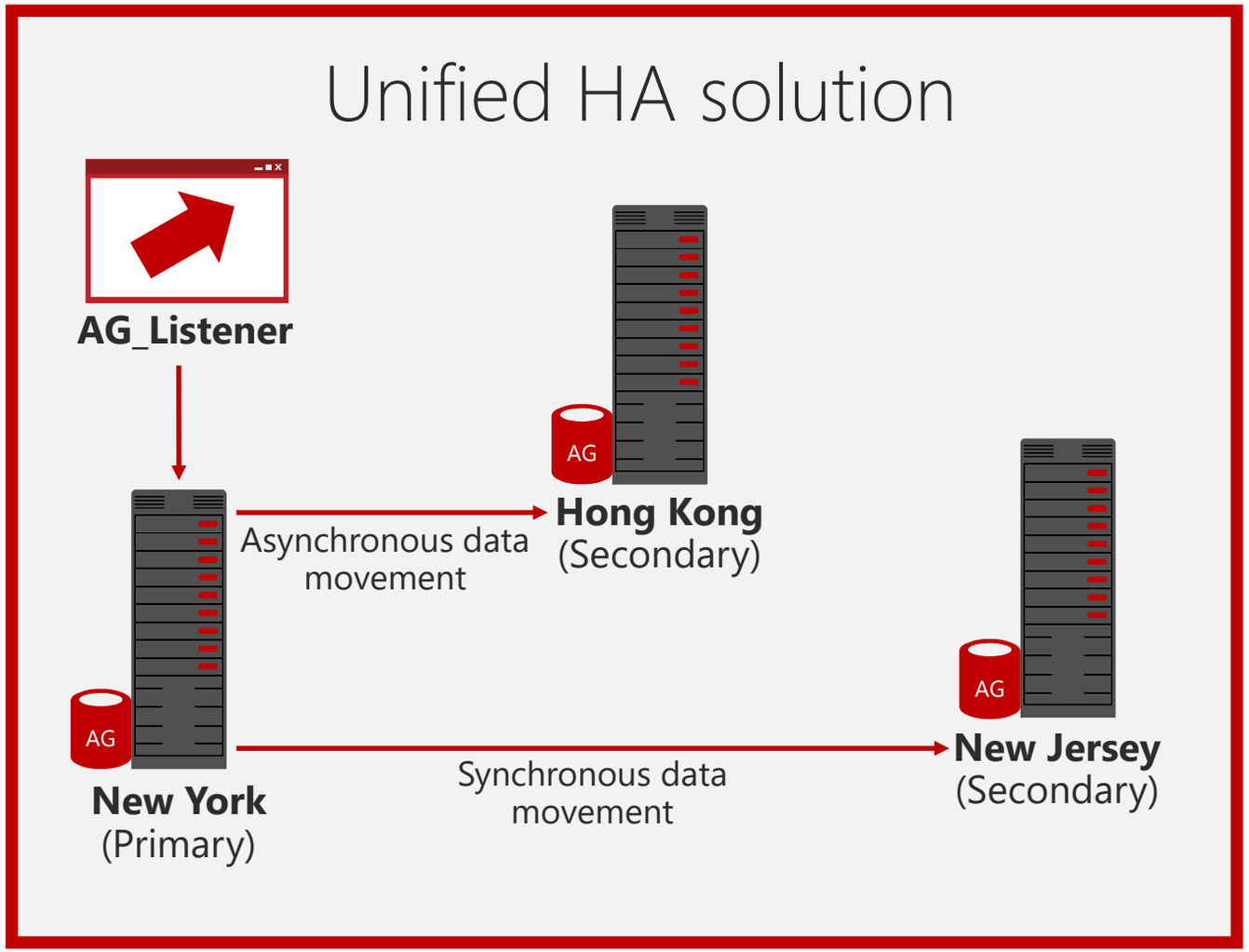
Deleted Rows Table (DRT) – Tracks deleted rows

Columnstore data fully resident in memory

Persisted together with operational data

No application changes required

Enhanced AlwaysOn Availability Groups



Greater scalability

Load-balancing readable secondaries

Increased number of automatic failover targets

Log transport performance

Supports Clustered ColumnStore

Improved manageability

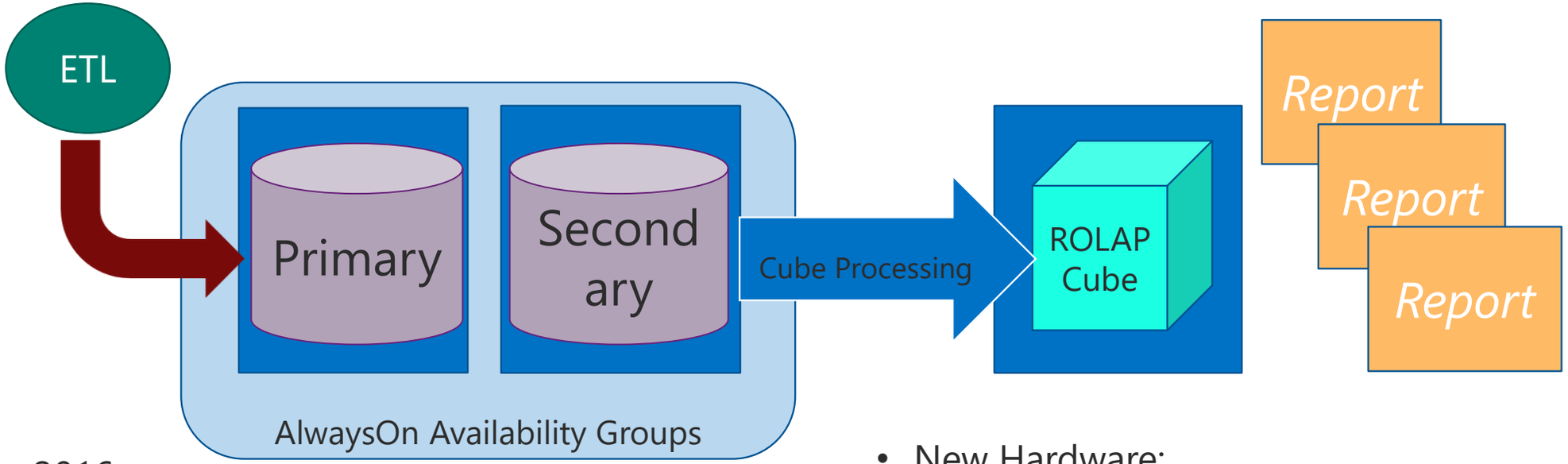
DTC support

Database-level health monitoring

Group Managed Service Account

Domain-independent Availability Groups

Solution Pattern: Leveraging Columnstore + AG for performance and scale



SQL Server 2016
Columnstore
AlwaysOn Availability Groups:
- ETL loading data into primary
- Queries / Cube processing against secondary
- HA + Read Scale Out

- New Hardware:
  - 16 cores/768GB RAM SSD
- Database Size:
  - Uncompressed: 12.5 TB
  - Compressed using Columnstore: 4.5 TB
- Dramatic performance improvement

Stretch Database

Ever-growing data, ever-shrinking IT

Massive tables (hundreds of millions/billions of rows, TBs size)

Users want/need to retain data indefinitely

Cold data infrequently accessed but must be online

Datacenter consolidation

Maintenance challenges

Business SLAs at risk

What to do?

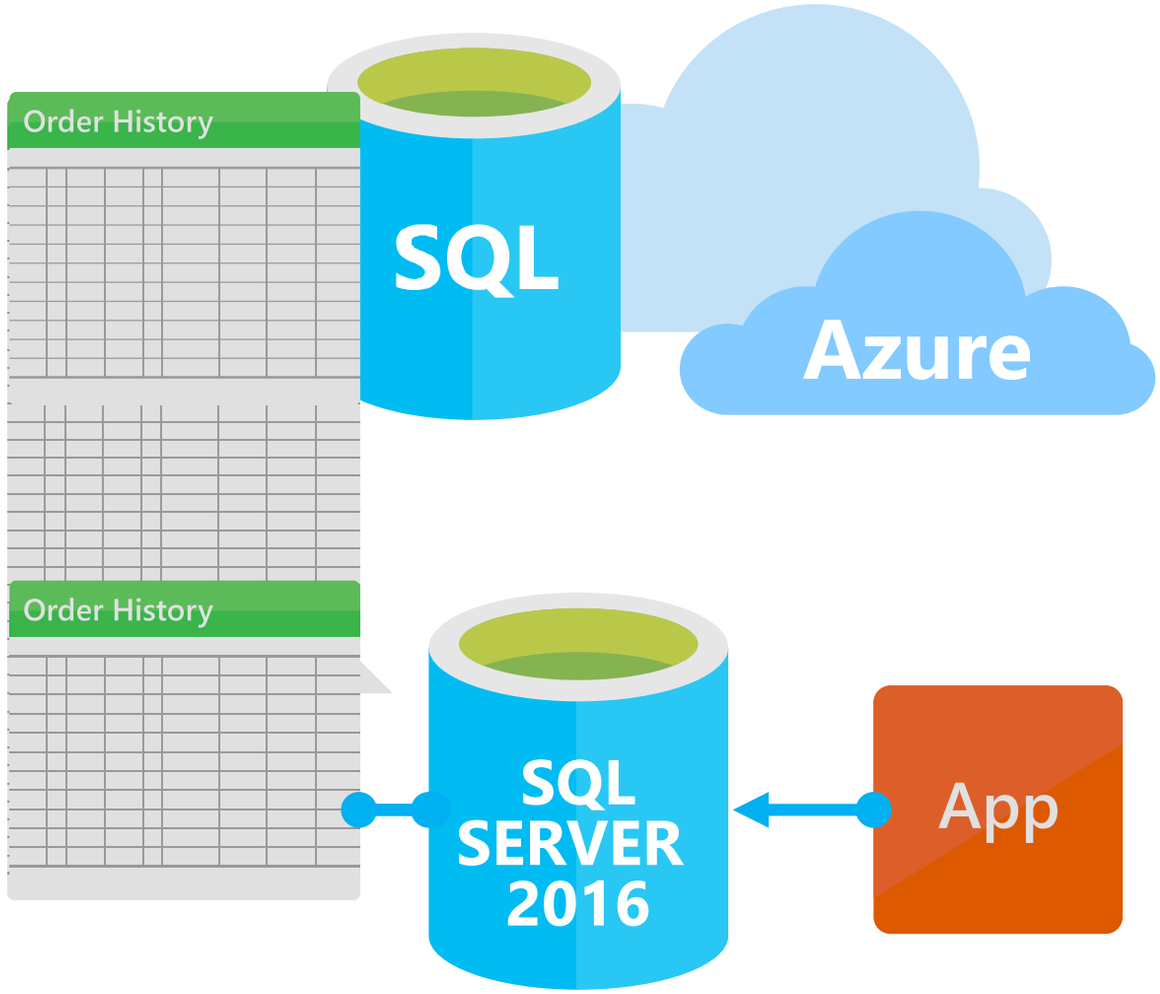
Expand server and storage

Move data elsewhere

Delete

Stretch SQL Server into Azure

Securely stretch cold tables to Azure with remote query processing



Capability

Stretch large operational tables from on-premises to Azure with the ability to query

Benefits

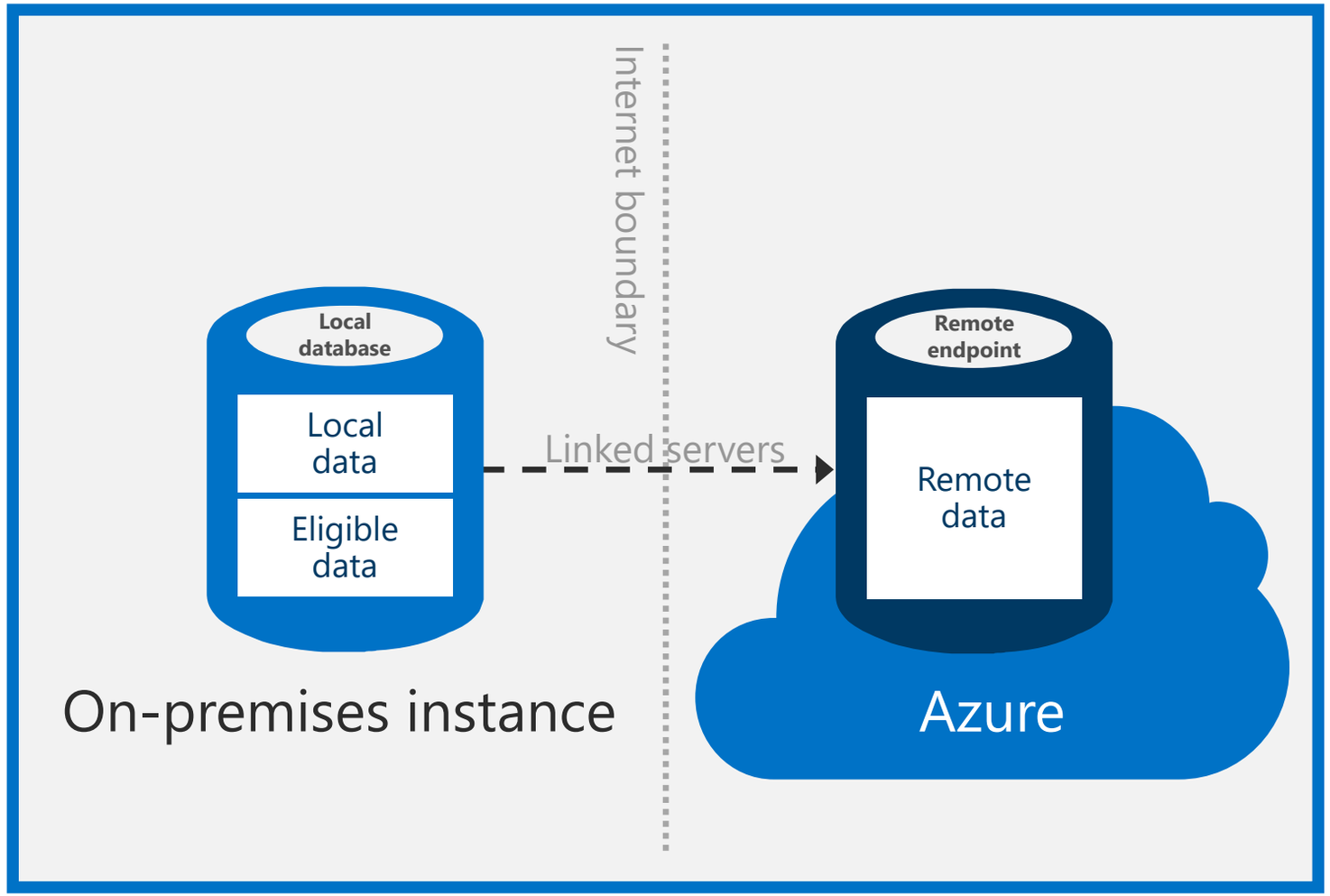
Cost-effective online cold data

Entire table is online and remains queryable from on-premises apps

No application changes

Support for Always Encrypted and Row-Level Security

Stretch Database architecture



How it works

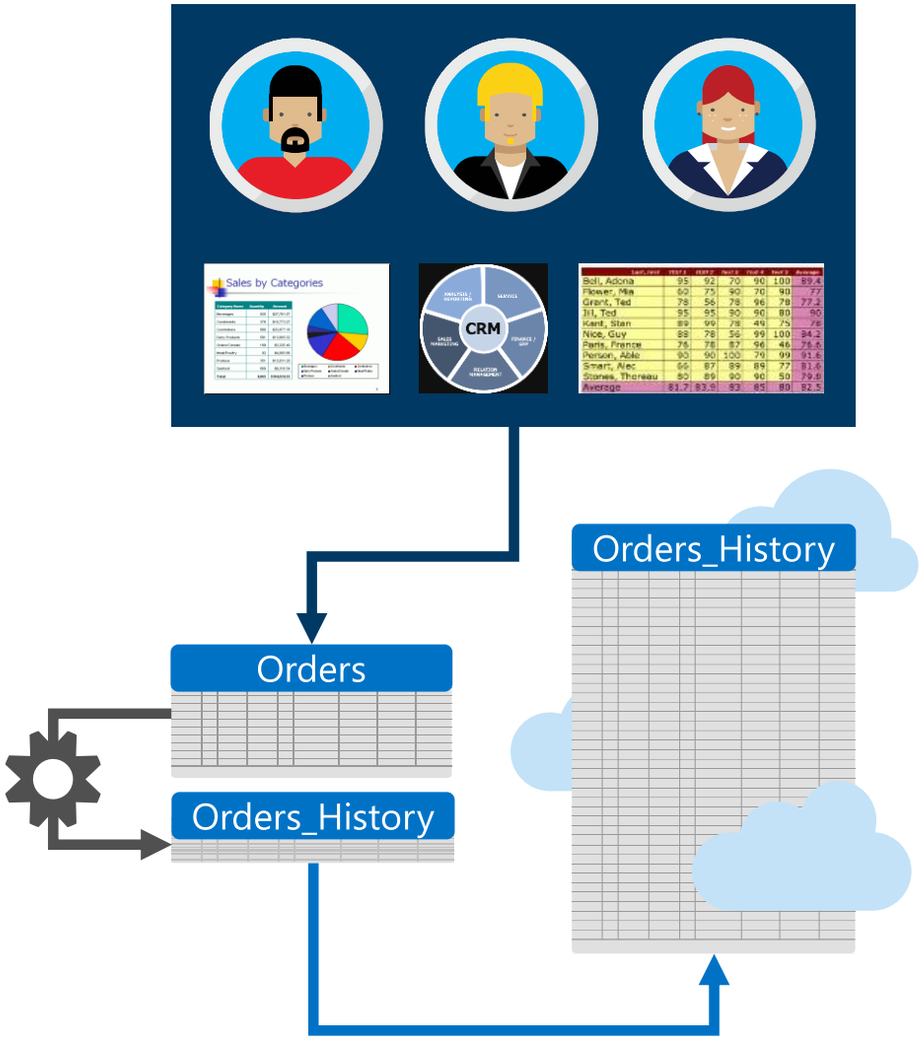
Creates a secure linked server definition in the on-premises SQL Server

Targets remote endpoint with linked server definition

Provisions remote resources and begins to migrate eligible data, if migration is enabled

Queries against tables run against both local database and remote endpoint

Queries continue working

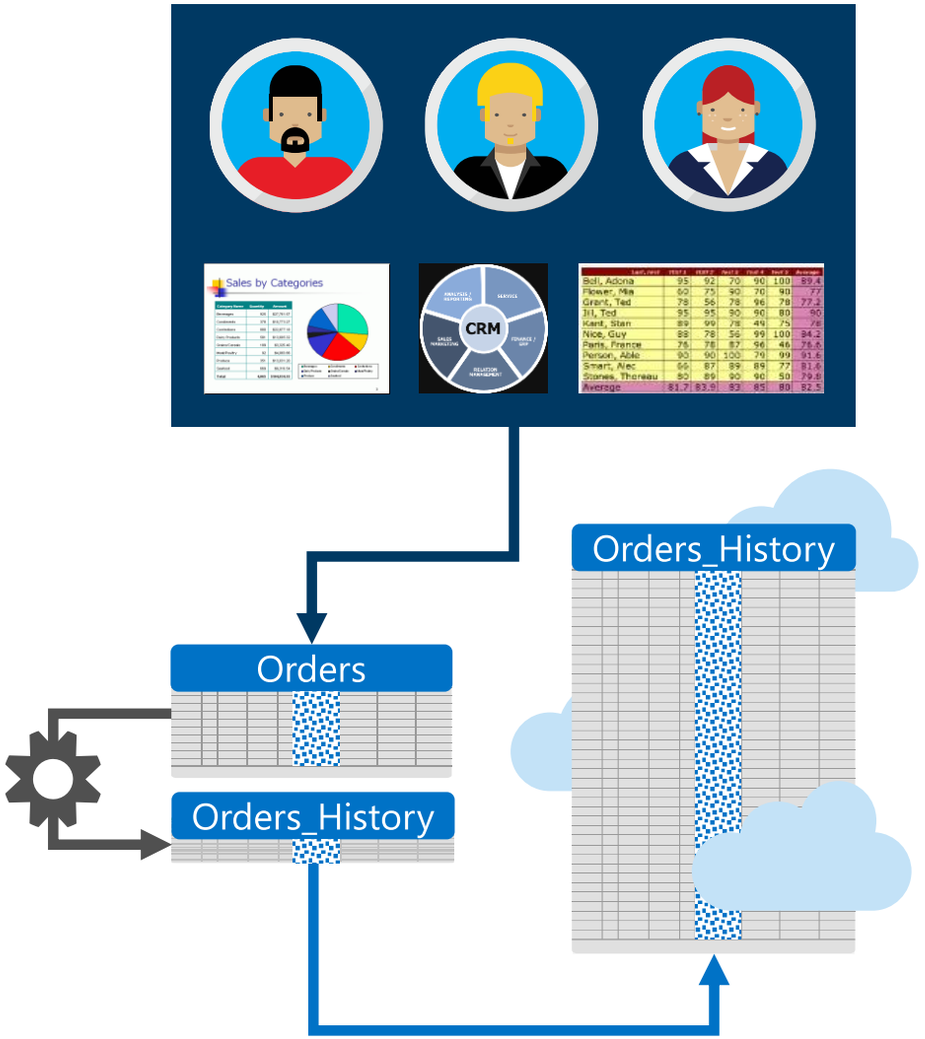


Business applications continue working without disruption

DBA scripts and tools work as before (all controls still held in local SQL Server)

Developers continue building or enhancing applications with existing tools and methods

Advanced security features supported



Data in motion always via secure channels (TLS 1.1/1.2)

Always Encrypted supported if enabled by user (encryption key remains on-premises)

Row-Level Security already working

SQL Server and SQL Azure auditing already working

Improvements in SQL Server 2016 that make it run faster without enabling new features

“It Just Works Faster”

Query Optimizer Improvements

There are two key changes

1. SQL Server now leverages parallelism when sample statistics are created either explicitly or as part of autostats
2. SQL Server now uses a sub linear threshold to trigger AUTO_UPDATE_STATISTICS computation to address the auto statistics updates especially for large tables. Until now, SQL Server used a fixed % of number of changes (i.e. delete, insert or updates) to a column irrespective of the size of the table. With this change, stats computation will get triggered at much lower % for larger tables.

Encryption enhancements

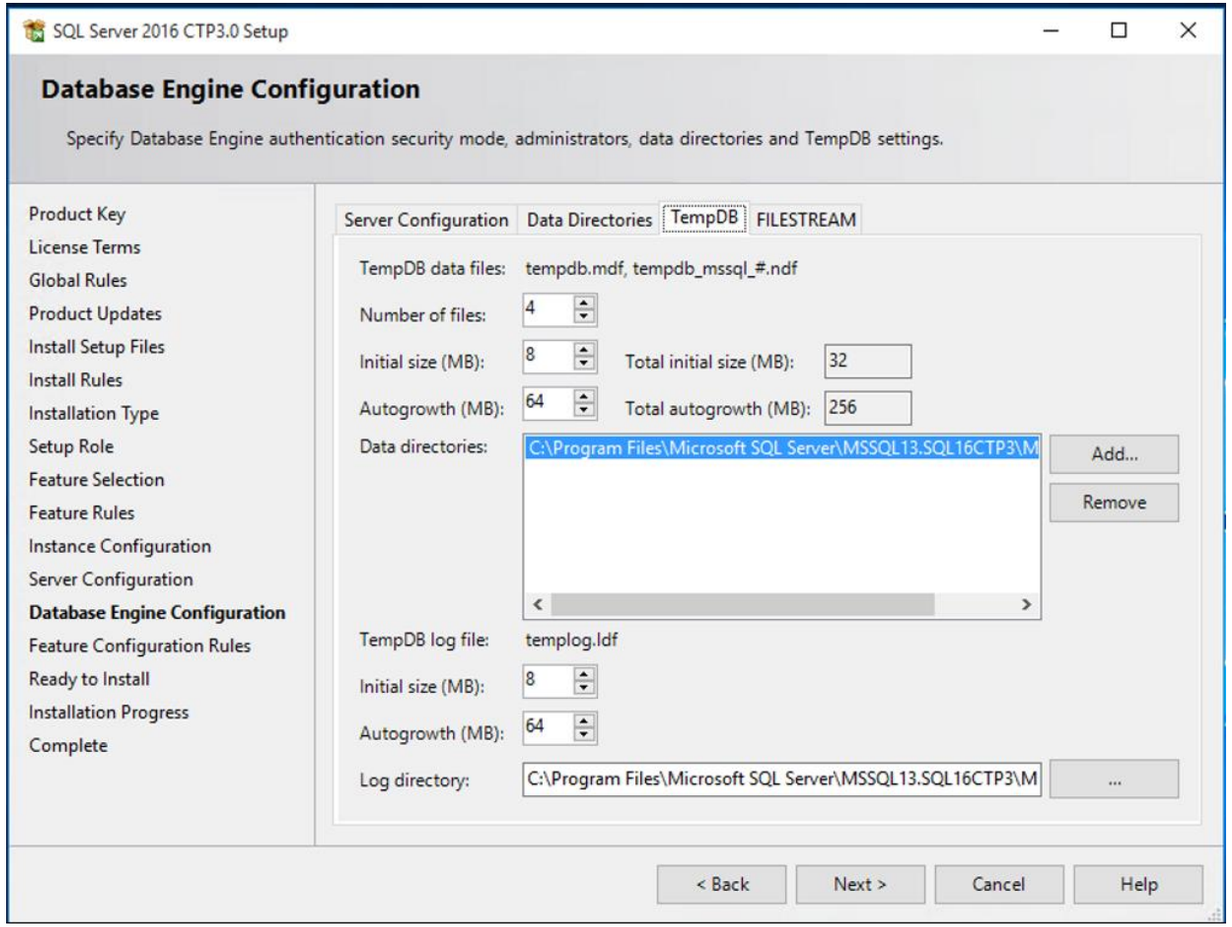
Hardware accelerated encryption/decryption for TDE

- Implements next generation of Microsoft cryptography
- Takes advantage of specialized microprocessor instructions
- Improves performance as much as 3x to 10x

Parallelizable decryption

- Decryption now supported as parallelizable (used to be sequential only)
- Dramatically improved response times for queries with encrypted data columns

Automatic TEMPDB Configuration



- Number of files will default to the lower of 8 or number of logical cores as detected by setup
- Initial size & Autogrowth
- TF 1117 is enabled by default for TEMPDB
- Enable Instant File Initializationif you specify a very large initial size or autogrowth value
- Specify multiple folders/drives to spread the datafiles across several volumes. Each file will be placed in a round-robin manner

- For Log File Autogrowth default value of 64MB is provided to so that the number of Virtual Log Files (VLFs) during initial creation is a small and manageable number and with appropriate size so that the unused log space can be reclaimed easily

-T1117 and –T1118 changes for TEMPDB and user databases

TEMPDB
One of these changes is TEMPDB always assumes -T1117 and -T1118 behavior.

-T1117 - When growing a data file grow all files at the same time so they remain the same size. Reducing allocation contention points.
-T1118 - When doing allocations for user tables always allocate full extents. Reducing contention of mixed extent allocations

In summary, SQL Server 2016 no longer requires one to turn on TF 1117 or 1118.

Trace Flags 1117 and 1118: User Databases and TempDB

User Database

For User Databases, trace flags 1117 and 1118 have been replaced with new extensions in ALTER DATABASE commands. Use the ALTER DATABASE syntax to enable or disable the desired trace flag behavior at a database level.

-- Trace Flag 1118

Trace flag 1118 for user databases is replaced by a new ALTER DATABASE setting - MIXED_PAGE_ALLOCATION.
Default value of the MIXED_PAGE_ALLOCATION is OFF meaning allocations in the database will use uniform extents.
The setting is opposite in behavior of the trace flag (i.e. TF 1118 OFF and MIXED_PAGE_ALLOCATION ON provide the same behavior and vice-versa).

Syntax:

ALTER DATABASE <dbname> SET **MIXED_PAGE_ALLOCATION** { ON | OFF }
For more information see https://msdn.microsoft.com/en-US/library/bb522682.aspx

-T1117 and –T1118 changes for TEMPDB and user databases

Example:
--Default value is OFF so all allocations in AdventureWorks will use uniform extents. To disable and use mixed extents turn the setting to on.
ALTER DATABASE AdventureWorks SET **MIXED_PAGE_ALLOCATION** ON;

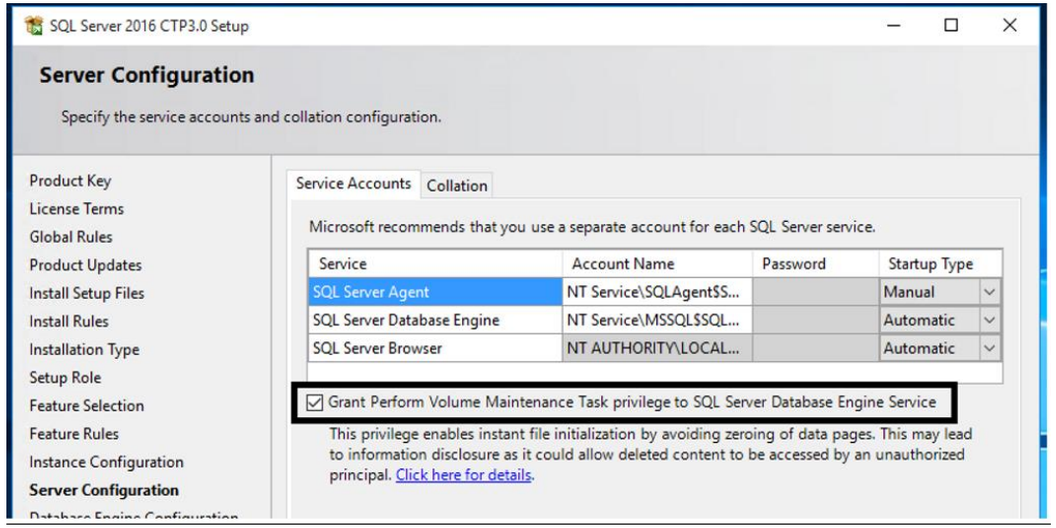
Catalog changes:
A new column is_mixed_page_allocation_on is added to DMV sys.databases that shows which allocation type (uniform or mixed) is being used. For more information see, https://msdn.microsoft.com/en-us/library/ms178534.aspx

Instant File Initialization

Database Instant File Initialization was added several SQL Server releases ago. The instant file initialization feature scales the creation and expansion (growth) of database, DATA files. The 'Manage Volume Privilege' option is off by default preventing many SQL Server installations from taking advantage of the feature.

SQL Server 2016 Setup provides the option to enable 'Perform Volume Maintenance Task' privilege to the SQL Server Service SID. This privilege enables instant file initialization by avoiding zeroing of data pages. For security and performance considerations see Database Instant File Initialization topic.

For Failover Cluster instance, each node will be configured individually for this option since the privilege belongs to local security policy. The option will show and can be enabled when adding each node.



If you are installing SQL Server using command line or a configuration file, set the **SQLSVCINSTANTFILEINIT** parameter to True to enable instant file initialization for SQL Server service account.

Core engine scalability

Dynamic partitioning of thread-safe memory objects by non-uniform memory access (NUMA) node or by CPU

Enables greater scalability of high-concurrency workloads running on NUMA hardware

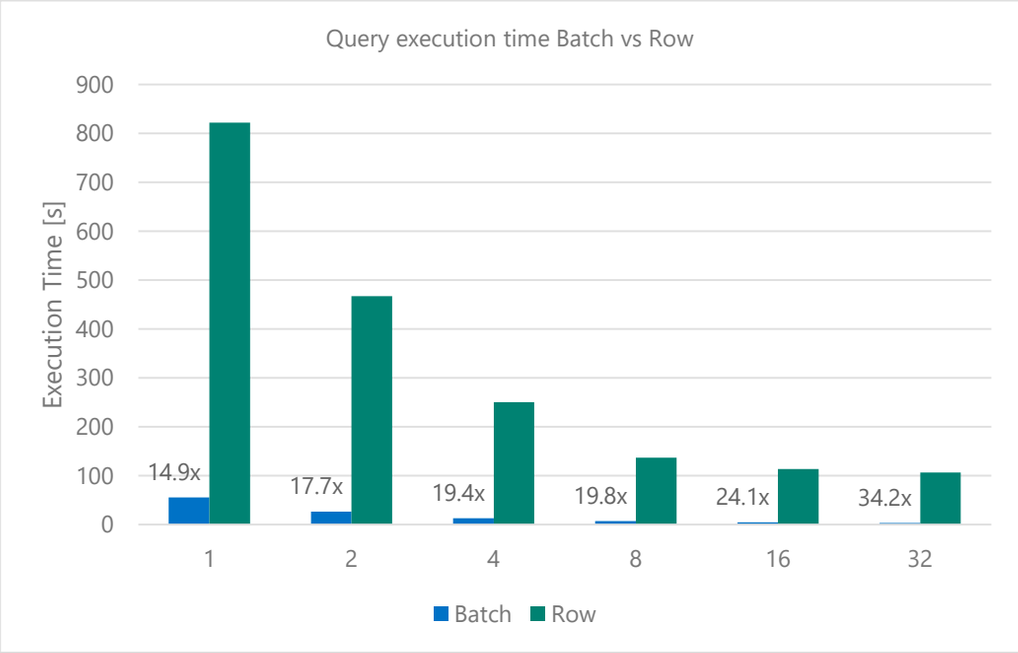
Dynamically promotes CMemThread to be partitioned by NUMA node or by CPU based on workload characteristics and contention factors

Eliminates need for trace flag, but also dynamically determines partition based on contention

SQL Server 2016 Runs Faster On Same Hardware

- A bold statement that any SQL Server professional can stand behind with confidence.
- No application changes needed, just works

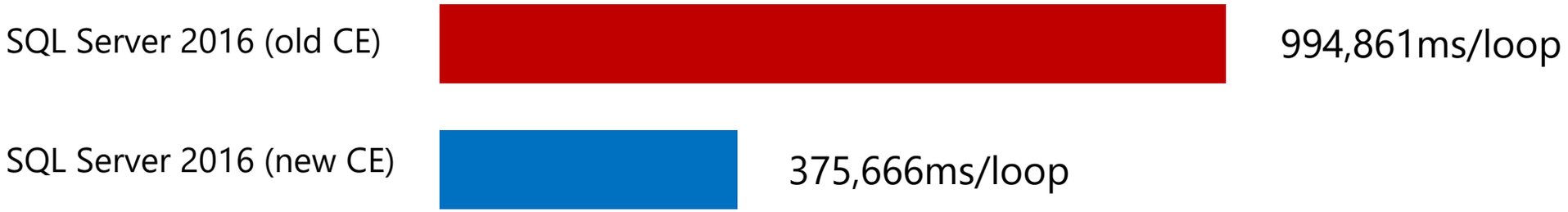
Columnstore **34X** faster



AlwaysOn **7X** faster

	Throughput MB/s	Avg CPU% (secondary)
SQL 2014	82	17
SQL 2016	**540**	**36**

TPC-H queries with new cardinality estimation



2.6x

faster

Tests running the 22 queries that make up the TPC-H benchmark test using a 3,000 warehouse workload (900 million rows in order_line table). Used traceflag 9481 to force the old cardinality estimation algorithm and compared the results using the new cardinality estimation.

Tests performed on Intel Xeon CPU E7-8890v3 @ 2.50GHz, 1.5 TB RAM, and Tegile Flash Storage Array.

DBCC CheckDB

7x

faster

Machine	32GB RAM, 4 Core Hyper-threaded enabled 2.8Ghz, SSD Storage
SQL Server	Out of the box, default installation

https://blogs.msdn.microsoft.com/psssql/2016/02/25/sql-2016-it-just-runs-faster-dbcc-scales-7x-better/

DBCC CheckDB

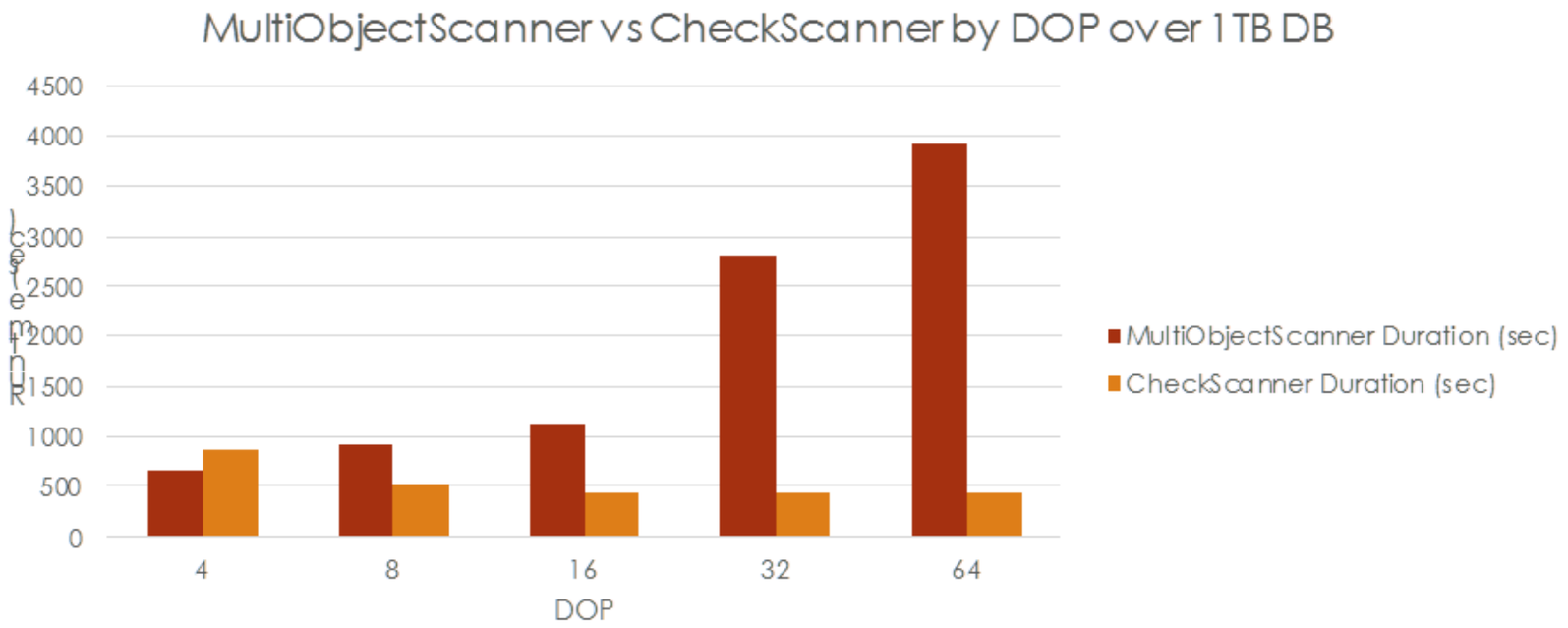
Internally DBCC CHECK* uses a page scanning coordinator design (MultiObjectScanner).

SQL Server 2016 changes the internal design (CheckScanner), applying no lock semantics and design similar to those used with In-Memory Optimized (Hekaton) objects, allowing DBCC operations to scale far better than previous releases.

The following chart shows the same 1TB database testing.

.

- MultiObjectScanner = Older design
- CheckScanner = New design



DBCC CHECKDB extended logical checks

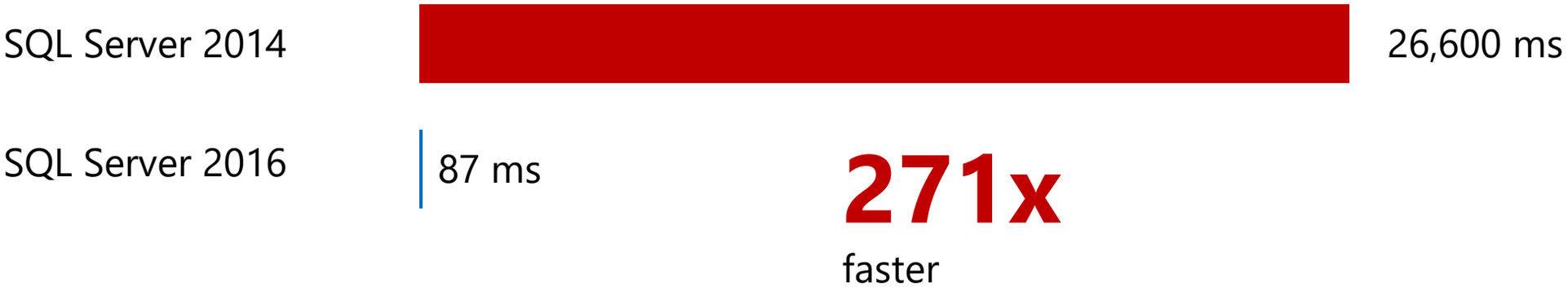
Starting with SQL Server 2016, additional checks on filtered indexes, persisted computed columns, and UDT columns will not be run by default to avoid the expensive expression evaluation. This will greatly reduce the time to run CHECKDB on databases that have these objects.

Physical consistency checks of these objects are still done . This means only when **EXTENDED_LOGICAL_CHECKS** option is specified, the expression evaluation is performed. This is in addition to other logical checks that are only performed (indexed view, XML indexes, and spatial indexes) when **EXTENDED_LOGICAL_CHECKS** option is specified.

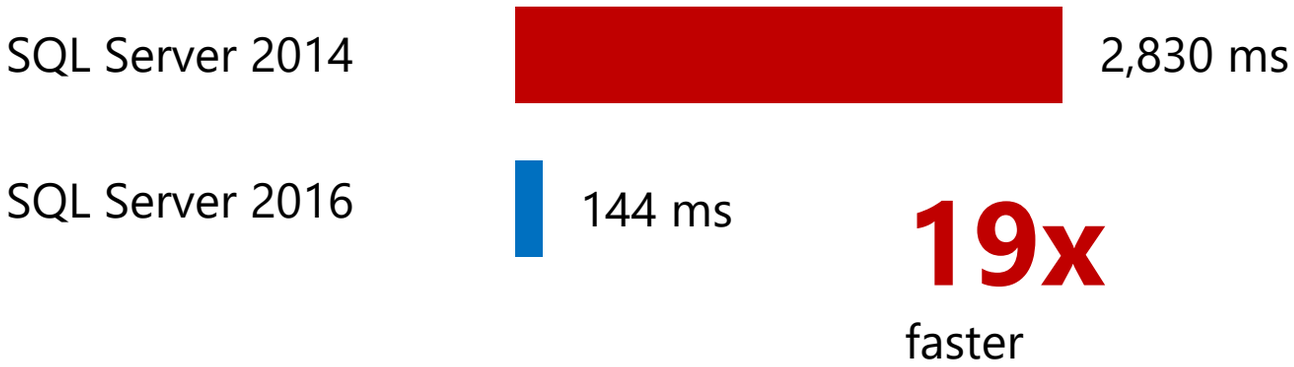
For filtered indexes, CHECKDB has also been improved to skip any data record that is not qualified for being indexed by target NC index.

https://blogs.msdn.microsoft.com/psssql/2016/03/01/sql-2016-it-just-runs-faster-dbcc-extended-checks/

Spatial Line String query improvements

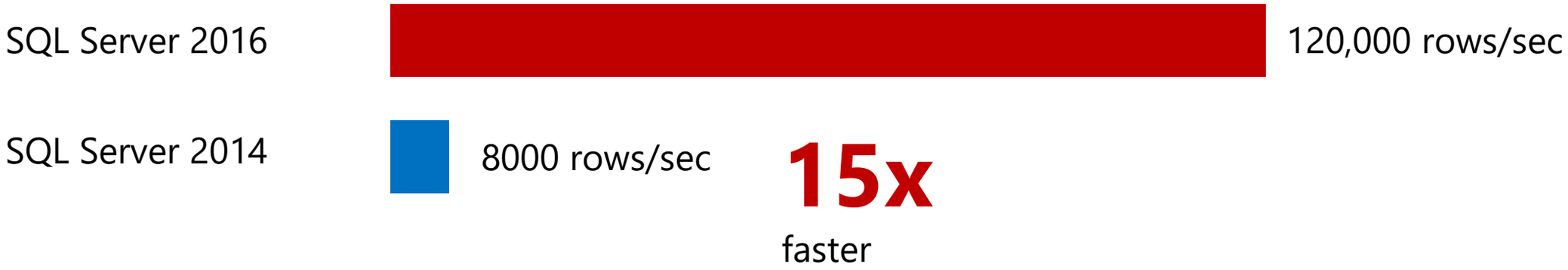


Spatial Native Function query improvements



Machine	32GB RAM, 4 Core Hyper-threaded enabled 2.8Ghz, SSD Storage
SQL Server	Out of the box, default installation

Table valued parameters using spatial columns



https://blogs.msdn.microsoft.com/psssql/2016/03/08/sql-2016-it-just-runs-faster-tvps-with-spatial-columns/

Table valued parameters using spatial columns

Table Valued Parameters (TVPs) can be used as input parameter(s) to stored procedures. A problem with TVP parameters, containing spatial columns, limits scalability. When a TVP parameter arrives at the SQL Server the rows are stored in TEMPDB. The problem caused the Spatial assembly to be reloaded as each spatial row and column was processed.

SQL Server 2016 corrects the scalability problem, using native spatial validation(s), increasing performance by 15 times or more.

TVP Before the Fix:	8000 rows/sec
TVP After the Fix:	120,000 rows/sec

https://blogs.msdn.microsoft.com/psssql/2016/03/08/sql-2016-it-just-runs-faster-tvps-with-spatial-columns/

Spatial Index Builds Faster

Index creation and tessellation can be intensive, spatial activities. Along with the native and TVP enhancements additional work to optimize index creation and tessellation was completed.

Testing reveals that building a spatial index on SQL Server 2016, with the improved design, can be more than 2 times faster than SQL Server 2012 or 2014 on the same data and hardware. It is common place for spatial tables to be 300 million or more rows. Reducing the index build time by a factor of 2x or more greatly reduces the need maintenance window(s.)

https://blogs.msdn.microsoft.com/psssql/2016/03/08/sql-2016-it-just-runs-faster-tvps-with-spatial-columns/

Automatic Soft NUMA

Soft NUMA can be used to divide a physical node into multiple logical nodes presenting a different layout to the entire SQL Server and adjusting the partitioning to optimize scalability and performance. Microsoft recommends use of Soft NUMA on the newer, large CPU NUMA system deployments to increase performance.

During startup, SQL Server 2016 interrogates the hardware layout and automatically configures Soft NUMA on systems reporting 8 or more CPUs per NUMA node. The partitioning triggers various adjustments throughout the database engine improving scalability and performance. The Automatic Soft NUMA logic considers logical CPU ratios, total CPU counts and other factors, attempting to create soft, logical nodes containing 8 or fewer CPUs each.

Your mileage may vary but, here is a testing results from the SQL Server 2016 test harness:
"With HT aware auto soft-NUMA, we get up-to 30% gain in query performance when DOP is set to the number of physical cores on a socket (12 in this case) using Automatic Soft NUMA."

Updated Scheduling Algorithms

SQL Server 2016 gets a scalability boost from scheduling updates. Testing uncovered issues with the percentile scheduling based algorithms in SQL Server 2012 and 2014. A large, CPU quantum worker and a short, CPU quantum worker can receive unbalanced access to the scheduling resources.

Take the following example. Worker 1 is a large, read query using read ahead and in-memory database pages and Worker 2 is doing shorter activities. Worker 1 finds information already in buffer pool and does not have to yield for I/O operations. Worker 1 can consume its full CPU quantum.

On the other hand, Worker 2 is performing operations that require it to yield. For discussion let's say Worker 2 yields at 1/20th of its CPU, quantum target. Taking resource governance and other activities out of the picture the scheduling pattern looks like the following.

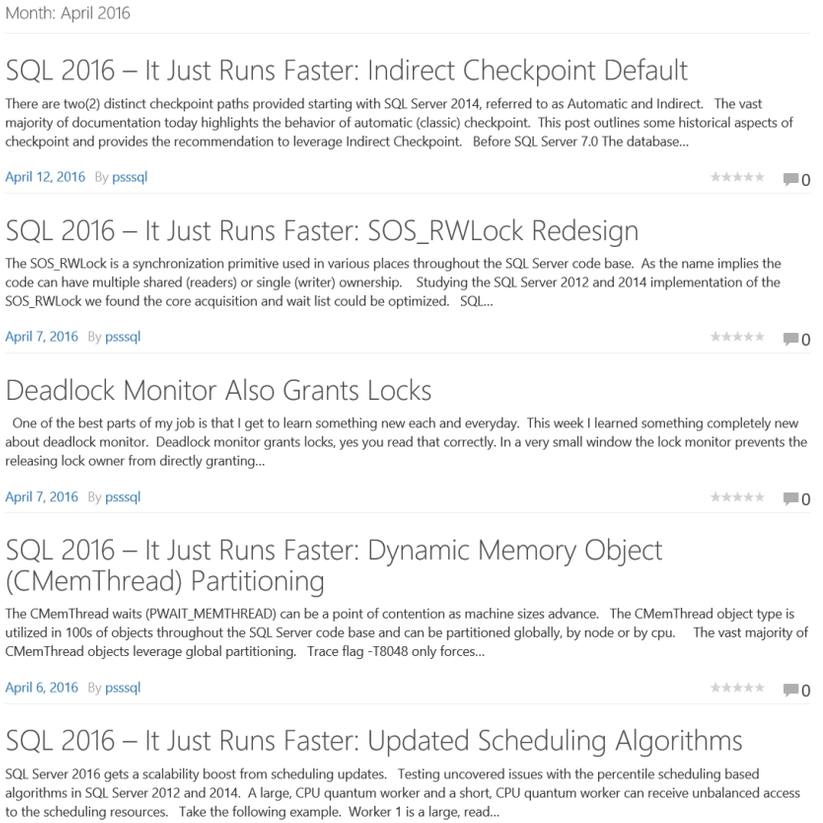
Worker 1 is getting ~5 times more CPU cycles than Worker 2. In testing we found issues with various workloads and system tasks. If Worker 2 is the log writer it takes longer to harden log records, which holds locks, which can lead to blocking and throughput issues.

SQL Server 2016 and Windows Azure SQL Database (WASD) monitors the quantum usage patterns allowing all workers to get fair treatment. The same pattern described above looks like the following on SQL Server 2016. In this simplistic example Worker 2 is allowed to consume repeated quantum's preventing Worker 1 from monopolizing the scheduler in an unfriendly pattern.

Note: The scheduler changes were deployed to Windows Azure SQL Server Database in March of 2014.

And many more….

https://blogs.msdn.microsoft.com/psssql/2016/02/23/sql-2016-it-just-runs-faster-announcement/



Month: April 2016

SQL 2016 – It Just Runs Faster: Indirect Checkpoint Default

There are two(2) distinct checkpoint paths provided starting with SQL Server 2014, referred to as Automatic and Indirect. The vast majority of documentation today highlights the behavior of automatic (classic) checkpoint. This post outlines some historical aspects of checkpoint and provides the recommendation to leverage Indirect Checkpoint. Before SQL Server 7.0 The database…

April 12, 2016 By psssql ★★★★★ 0

SQL 2016 – It Just Runs Faster: SOS_RWLock Redesign

The SOS_RWLock is a synchronization primitive used in various places throughout the SQL Server code base. As the name implies the code can have multiple shared (readers) or single (writer) ownership. Studying the SQL Server 2012 and 2014 implementation of the SOS_RWLock we found the core acquisition and wait list could be optimized. SQL…

April 7, 2016 By psssql ★★★★★ 0

Deadlock Monitor Also Grants Locks

One of the best parts of my job is that I get to learn something new each and everyday. This week I learned something completely new about deadlock monitor. Deadlock monitor grants locks, yes you read that correctly. In a very small window the lock monitor prevents the releasing lock owner from directly granting…

April 7, 2016 By psssql ★★★★★ 0

SQL 2016 – It Just Runs Faster: Dynamic Memory Object (CMemThread) Partitioning

The CMemThread waits (PWAIT_MEMTHREAD) can be a point of contention as machine sizes advance. The CMemThread object type is utilized in 100s of objects throughout the SQL Server code base and can be partitioned globally, by node or by cpu. The vast majority of CMemThread objects leverage global partitioning. Trace flag -T8048 only forces…

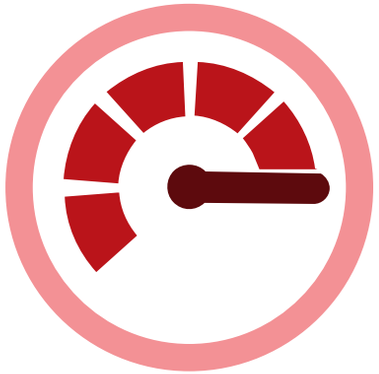
April 6, 2016 By psssql ★★★★★ 0

SQL 2016 – It Just Runs Faster: Updated Scheduling Algorithms

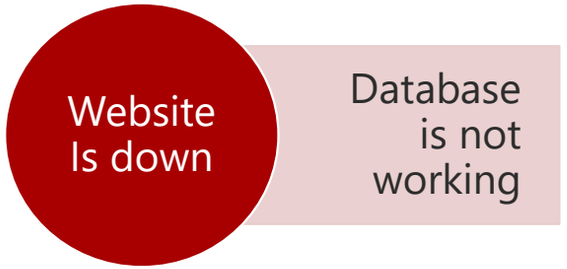
SQL Server 2016 gets a scalability boost from scheduling updates. Testing uncovered issues with the percentile scheduling based algorithms in SQL Server 2012 and 2014. A large, CPU quantum worker and a short, CPU quantum worker can receive unbalanced access to the scheduling resources. Take the following example. Worker 1 is a large, read…

Query Store

Your flight data recorder
for your database



Problems with query performance



Fixing query plan choice regressions is difficult

- Query plan cache is not well-suited for performance troubleshooting

Long time to detect the issue (TTD)

- Which query is slow? Why is it slow?
- What was the previous plan?

Long time to mitigate (TTM)

- Can I modify the query?
- How to use plan guide?

Plan choice change can cause these problems

The solution: Query Store

Dedicated store for query workload performance data

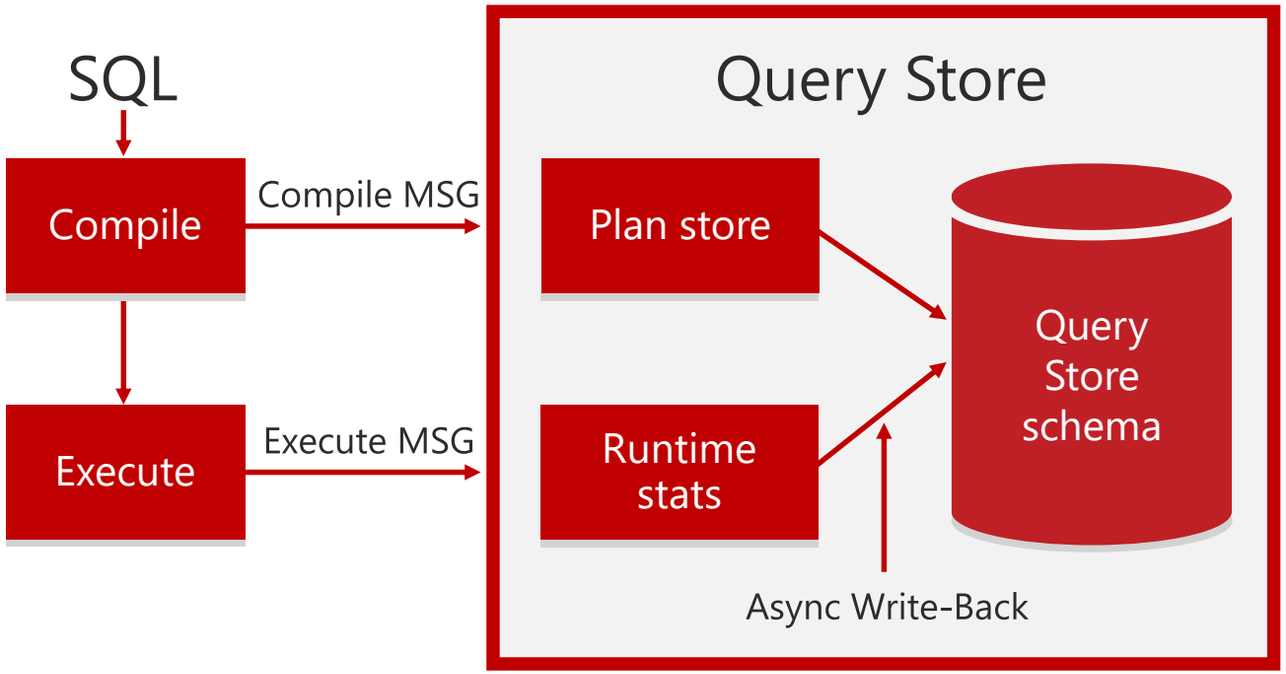
Captures the history of plans for each query
Captures the performance of each plan over time
Persists the data to disk (works across restarts, upgrades, and recompiles)

Significantly reduces TTD/TTM

Find regressions and other issues in seconds
Allows you to force previous plans from history

DBA is now in control

Query data store



Durability latency controlled by DB option
DATA_FLUSH_INTERNAL_SECONDS

Collects query texts (plus all relevant properties)

Stores all plan choices and performance metrics

Works across restarts / upgrades / recompiles

Dramatically lowers the bar for performance troubleshooting

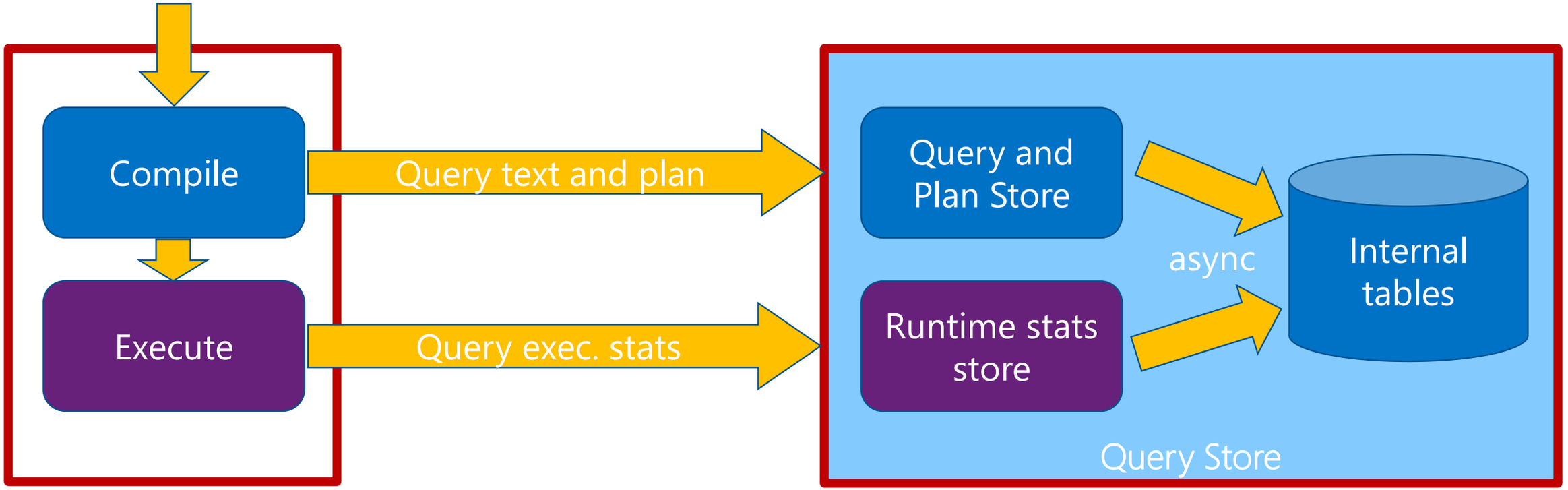
New Views

Intuitive and easy plan forcing

Query Store write architecture

Query Store captures data in-memory to minimize I/O overhead

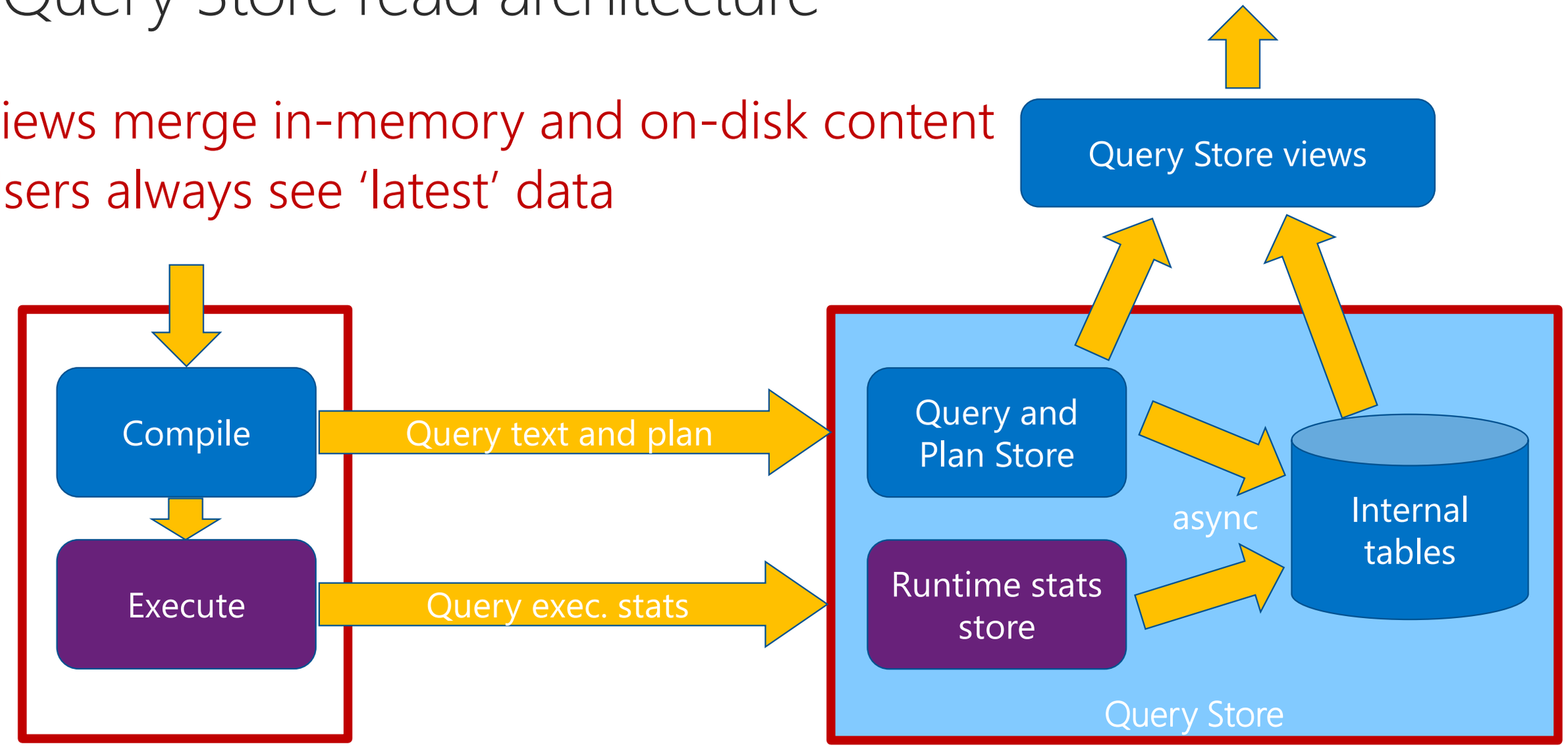
Data is persisted to disk asynchronously in the background



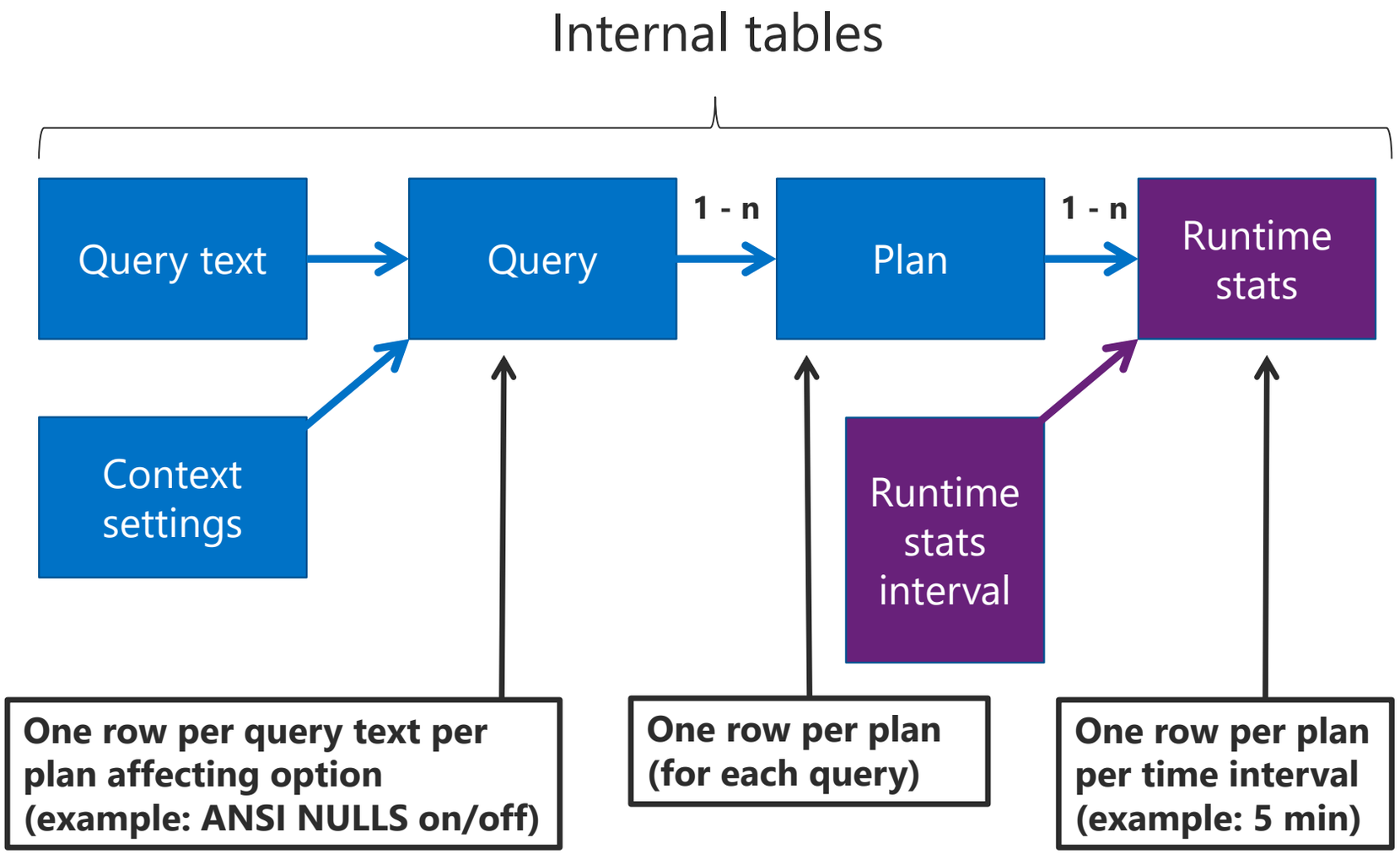
Query Store read architecture

Views merge in-memory and on-disk content

Users always see 'latest' data



Query Store schema explained



Exposed views

sys.

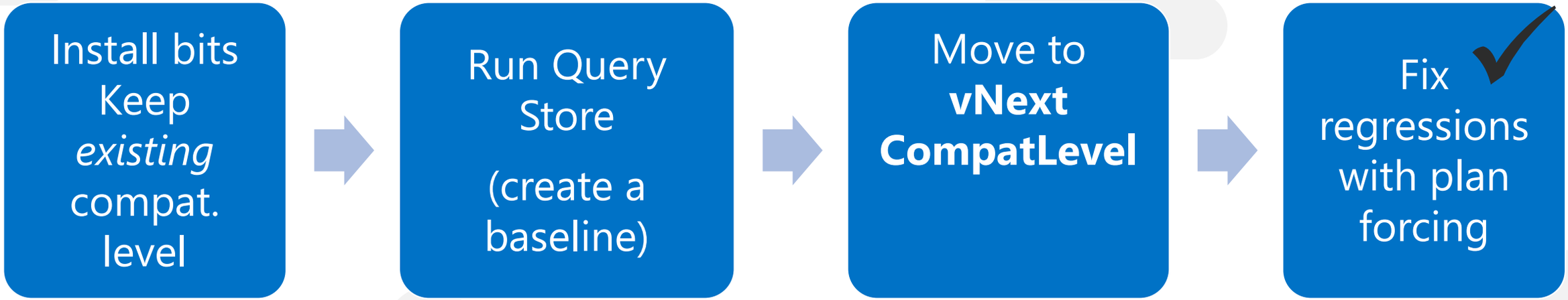
Compile stats:
- query_store_query_text
- query_context_settings
- query_store_query
- query_store_plan

Runtime stats:
- query_store_runtime_stats_interval
- query_store_runtime_stats

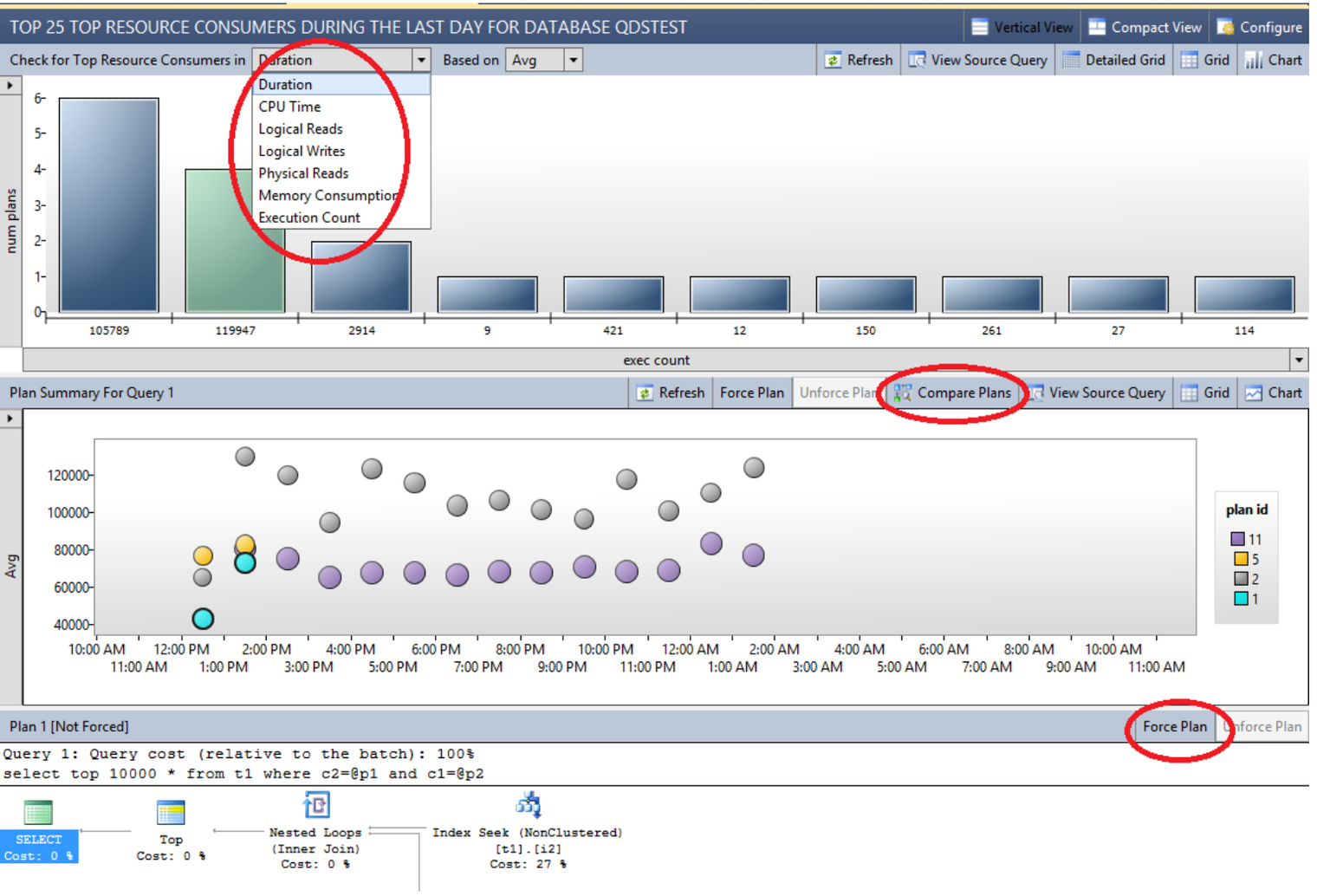
Keeping stability while upgrading to SQL Sever 2016

SQL Server 2016

QO enhancements tied to database compatibility level



Monitoring performance by using the Query Store



The Query Store feature provides DBAs with insight on query plan choice and performance

Working with Query Store

```
/* (6) Performance analysis using Query Store views*/
SELECT q.query_id, qt.query_text_id, qt.query_sql_text,
SUM(rs.count_executions) AS total_execution_count
FROM
sys.query_store_query_text qt JOIN
sys.query_store_query q ON qt.query_text_id =
q.query_text_id JOIN
sys.query_store_plan p ON q.query_id = p.query_id JOIN
sys.query_store_runtime_stats rs ON p.plan_id = rs.plan_id
GROUP BY q.query_id, qt.query_text_id, qt.query_sql_text
ORDER BY total_execution_count DESC

/* (7) Force plan for a given query */
exec sp_query_store_force_plan
12 /*@query_id*/, 14 /*@plan_id*/
```

```
);

/* (4) Clear all Query Store data */
ALTER DATABASE MyDB SET QUERY_STORE CLEAR;

/* (5) Turn OFF Query Store */
ALTER DATABASE MyDB SET QUERY_STORE = OFF;
```

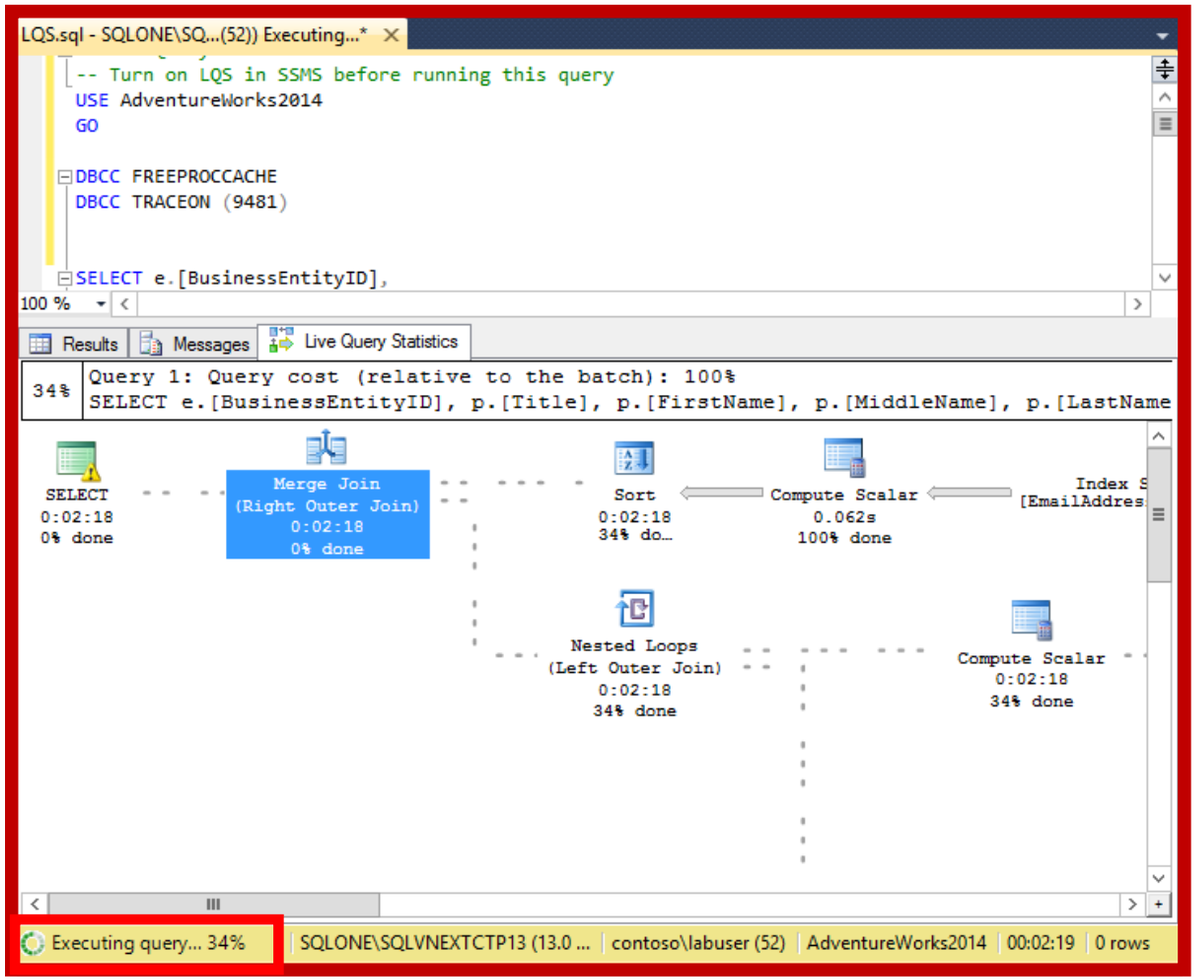
DB-level feature exposed through T-SQL extensions

ALTER DATABASE

Catalog views (settings, compile, and runtime stats)

Stored Procs (plan forcing, query/plan/stats cleanup)

Live query statistics



View CPU/memory usage, execution time, query progress, and more

Enables rapid identification of potential bottlenecks for troubleshooting query performance issues

Allows drill down to live operator level statistics:

Number of generated rows

Elapsed time

Operator progress

Live warnings

Summary: Query Store

Capability

Query Store helps customers quickly find and fix query performance issues

Query Store is a ‘flight data recorder’ for database workloads

Benefits

Greatly simplifies query performance troubleshooting

Provides performance stability across SQL Server upgrades

Allows deeper insight into workload performance