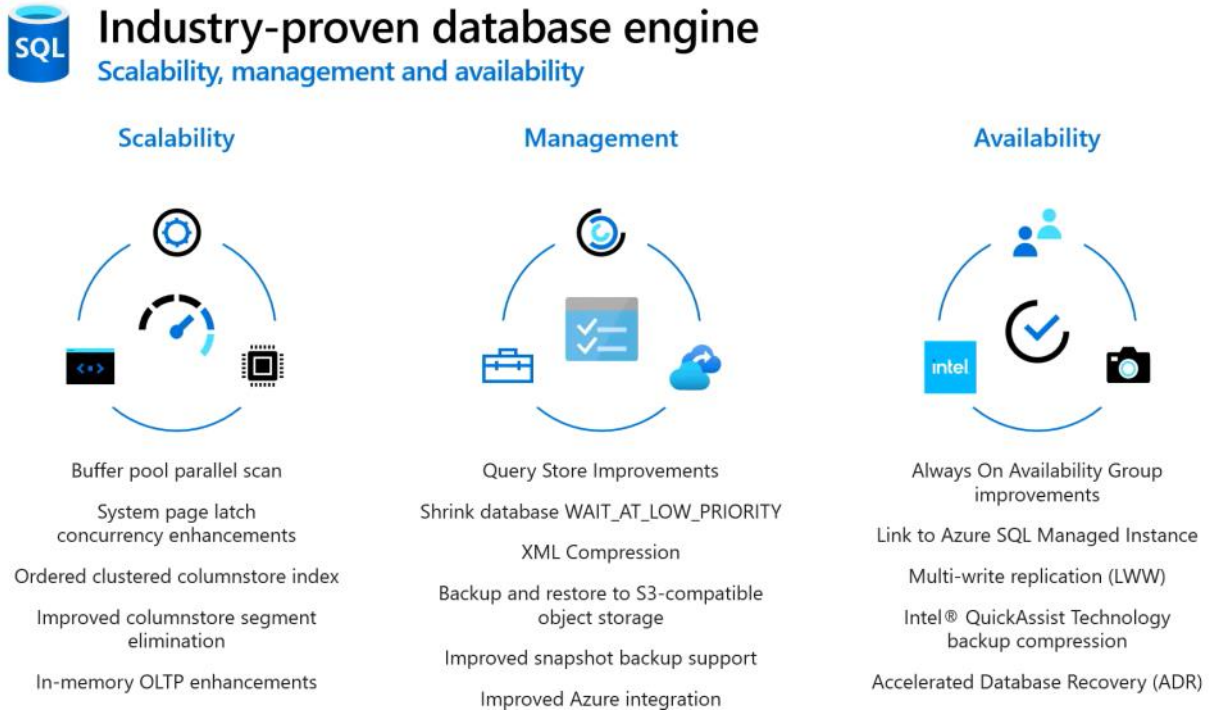
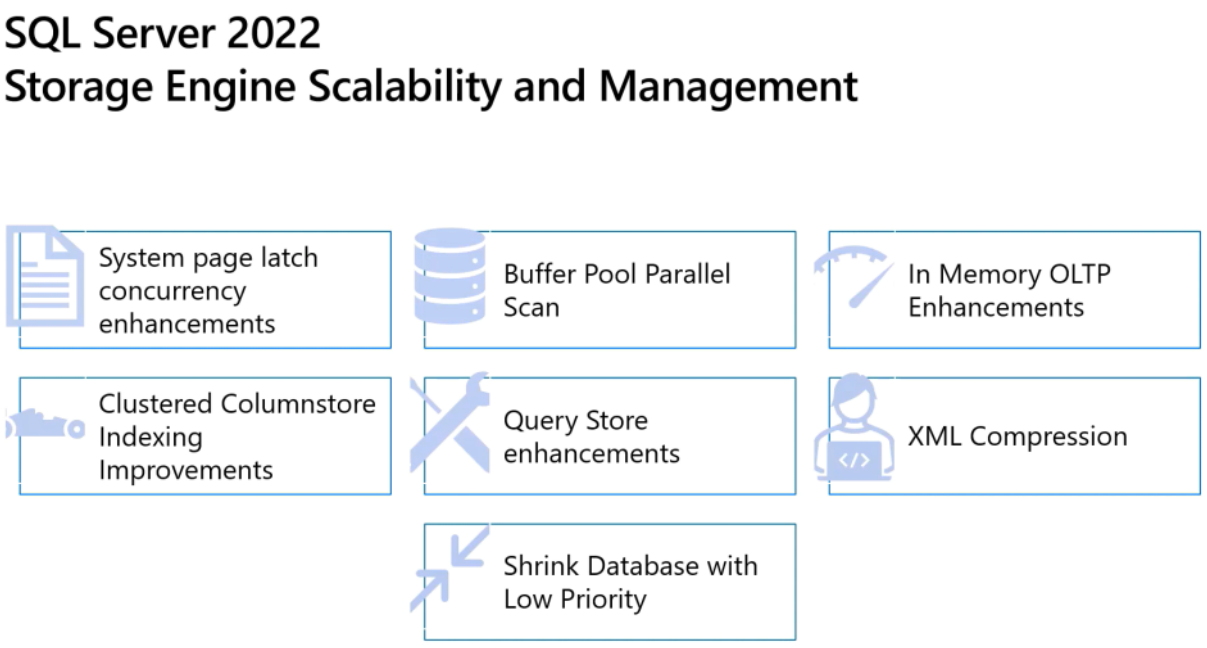
SQL Server 2022

Monday, April 24, 2023 8:13 PM

[SQL Server 2022 Blogging Series Archives - Microsoft SQL Server Blog]





tempdb Performance Critical for Scalability

SQL Server 2016 Improvements

- Setup experience has improved
- Trace Flag 1117 and 1118 are no longer required

SQL Server 2017 Improvements

- Round robin PFS
- Improvements to optimistic latching

SQL Server 2019 Improvements

- Memory-optimized tempdb metadata
- Concurrent PFS updates

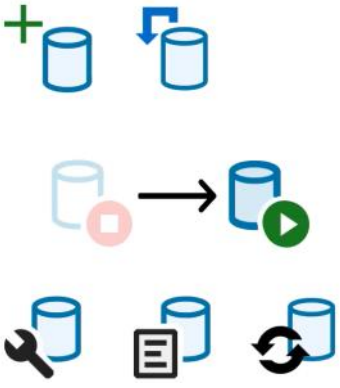
SQL Server 2022 Improvements

- System page latch concurrency enhancements
  - Concurrent Global Allocation Map (GAM) Updates
  - Concurrent Shared Global Allocation Map (SGAM) Updates

```
--GET OBJECT INFO FROM PAGE RESOURCE SQL 2019
-- RESOLVE WAIT RESOURCE TO OBJECT INFORMATION
-- NEW FUNCTIONS AVAILABLE IN SQL SERVER 2019
USE master GO
SELECT
er.session_id, er.wait_type, er wait_resource,
OBJECT_NAME(page_info.[object_id],page_info.database_id) as [object_name],
er.blockingsessionid,er.command,
SUBSTRING(st.text, (er.statement_start_offset/2) * 1,
((CASE er.statement_end_offset WHEN -1 THEN DATALENGTH(st.text)
ELSE er.statement_end_offset
END      er.statement_start_offset)/2) + 1) AS statement_text,
page_info.database_id,page_info.[file_id], page_info.page_id, page_info.[object_id],
page_info.index_id, page_info.page_type_desc FROM sys.dm_exec_requests AS er
CROSS APPLY sys.dm_exec_sql_text(er.sql_handle) AS st CROSS APPLY sys.fn_PageResCracker
(er.page_resource) AS r
CROSS APPLY sys.dm_db_page_info(r.[db_id], r.[file_id], r.page_id, 'DETAILED') AS page_info WHERE
er.wait_type like '%page%'
```

Current Buffer Pool Scan Operations

- Operations that scan the buffer pool can be slow, especially on large memory machines, such as:
  - Creating a new databases
  - File drop operations
  - Backup/restore operations
  - Always On failover events
  - DBCC CHECKDB
  - Log restore operations
  - Internal operations (e.g., checkpoint)



SQL Server 2022 - Buffer Pool Parallel Scan

- Buffer Pool Scans are parallelizing by utilizing multiple cores
- Benefits to both small and large databases on large-memory machines
- Improvement adds Buffer Pool scan diagnostics and telemetry for supportability and insights.
  - Long Buffer Pool scans will be visible by the ERRORLOG
  - Extended events will capture scan start/complete, errors, FlushCache, etc.
- Customers running mission critical OLTP, and data warehouse environments will see the most benefit

operations on large memory machines by utilizing multiple CPU cores

References:

Improve scalability with Buffer Pool Parallel Scan in SQL Server 2022
https://aka.ms/bpps

Buffer Pool Parallel Scan Diagnostics

Extended Events

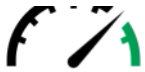
event name	channel
buffer_pool_scan_start	Debug
buffer_pool_scan_complete	**Analytic**
buffer_pool_scan_task_start	Debug
buffer_pool_scan_task_complete	Debug
buffer_pool_scan_task_error	Debug
buffer_pool_scan_stats	Analytic
buffer_pool_flush_cache_start	Debug
buffer_pool_flush_cache	**Analytic**
buffer_pool_flush_io_throttle	Debug
buffer_pool_flush_io_throttle_status	Debug

timestamp	name	database_id	elapsed_time_ms	elapsed_time_sec	command	operation	scanned_buffers	total_iterated_buffers	sql_text
2020-04-21 08:32:03.7232160	buffer_pool_scan_complete	9	14249	14	CREATE DATABASE	FlushCache	115	204640239	create database testdb1
2020-04-21 08:35:57.7701936	buffer_pool_scan_complete	9	14277	14	DBCC	FlushCache	321	204640239	DBCC FLUSH('data', 'testdb1')
2020-04-21 08:36:59.6669784	buffer_pool_scan_complete	9	11422	11	DBCC	MarkBuffersCopyOnWrite	321	204640238	dbcc checkdb('testdb1')
2020-04-21 08:37:25.4008359	buffer_pool_scan_complete	10	14244	14	DBCC	FlushCache	26	204640239	dbcc checkdb('testdb1')
2020-04-21 08:37:48.5725842	buffer_pool_scan_complete	9	11545	11	DBCC CHECKCATALOG	MarkBuffersCopyOnWrite	329	204640238	dbcc checkdb('testdb1')
2020-04-21 08:38:00.0099010	buffer_pool_scan_complete	10	11429	11	DBCC CHECKCATALOG	RemoveDatabase	383	204640238	dbcc checkdb('testdb1')
2020-04-21 08:38:53.0288060	buffer_pool_scan_complete	9	11424	11	DBCC TABLE CHECK	MarkBuffersCopyOnWrite	329	204640238	dbcc checktable('testdb1.dbo.t1')
2020-04-21 08:39:18.7317759	buffer_pool_scan_complete	10	14220	14	DBCC TABLE CHECK	FlushCache	26	204640239	dbcc checktable('testdb1.dbo.t1')
2020-04-21 08:39:41.6384020	buffer_pool_scan_complete	9	11417	11	DBCC	MarkBuffersCopyOnWrite	329	204640238	dbcc checktable('testdb1.dbo.t1')
2020-04-21 08:39:53.0755504	buffer_pool_scan_complete	10	11423	11	DBCC	RemoveDatabase	157	204640238	dbcc checktable('testdb1.dbo.t1')
2020-04-21 08:42:33.3964545	buffer_pool_scan_complete	9	11461	11	ALTER DATABASE	RemoveDatabaseByFile	352	204640238	alter database testdb1 remove file data2
2020-04-21 08:43:46.6099952	buffer_pool_scan_complete	9	14232	14	BACKUP DATABASE	FlushCache	352	204640239	backup database testdb1 to disk='nul'
2020-04-21 08:45:12.0283396	buffer_pool_scan_complete	10	14280	14	CREATE DATABASE	FlushCache	115	204640239	create database testdb3 alter database testdb3 set ...
2020-04-21 08:46:23.6216234	buffer_pool_scan_complete	10	14249	14	BACKUP DATABASE	FlushCache	306	204640239	create database testdb3 alter database testdb3 set ...
2020-04-21 08:47:10.3404686	buffer_pool_scan_complete	10	11814	11	DROP DATABASE	RemoveDatabase	306	204640238	create database testdb3 alter database testdb3 set ...
2020-04-21 08:47:36.5430815	buffer_pool_scan_complete	10	14470	14	RESTORE DATABASE	FlushCache	4	204640239	create database testdb3 alter database testdb3 set ...
2020-04-21 08:48:13.7775513	buffer_pool_scan_complete	10	14223	14	RESTORE LOG	FlushCache	4	204640239	create database testdb3 alter database testdb3 set ...
2020-04-21 08:48:51.0432028	buffer_pool_scan_complete	10	14244	14	RESTORE LOG	FlushCache	5	204640239	create database testdb3 alter database testdb3 set ...

In-memory Feature Family

- In-Memory OLTP (SQL 2014+/2022)
- Memory-Optimized tempdb Metadata (SQL 2019/2022)
- Memory Protection Keys (SQL 2019/2022)
- Persisted Log Buffer ("Tail of Log Caching") (SQL 2016+/2019)
- Data file "Enlightenment" (SQL 2019/2022)
- Hybrid Buffer Pool (SQL 2019/2022)
  - Read caching (SQL 2019)
  - Direct write (SQL 2022)

In-memory OLTP memory enhancements



- Improved supportability with insights on memory usage related to memory consumers

  sys.dm_xtp_system_memory_consumers
- Improved the ability to manage and make memory shrinkable
- Release as much memory as possible in case of memory pressure or on demand
- Provide a method to release unused memory on demand

  sys.sp_xtp_force_gc

Columnstore Index Improvements

- Ordered Clustered ColumnStore Index - Ordered (CCI) sorts the existing data in memory before the index builder compresses the data into index segments
  - More efficient segment elimination
  - Reduced number of segments to read from disk
- Improved ColumnStore segment elimination - Beginning in SQL Server 2022, segment elimination capabilities extend to string, binary, GUID data types, and the datetimeoffset data types
- Improved optimization - SQL Server 2022 leverages new hardware capabilities including the Advanced Vector Extension (AVX) 512 extension to improve batch mode operations.

XML Compression

- XML data type is commonly used to store unstructured data
- Data compression only applies to in-row scenarios (row, page)
- XML Compression will compress the XML data type in Azure SQL and SQL Server 2022
- XML Compression can be specified during CREATE and ALTER of TABLE and INDEX statements
- sp_estimate_data_compression_savings will be expanded to estimate XML savings

```
ALTER TABLE Sales.StoreBIGXMLCopy REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = PAGE, XML_COMPRESSION = ON)
```

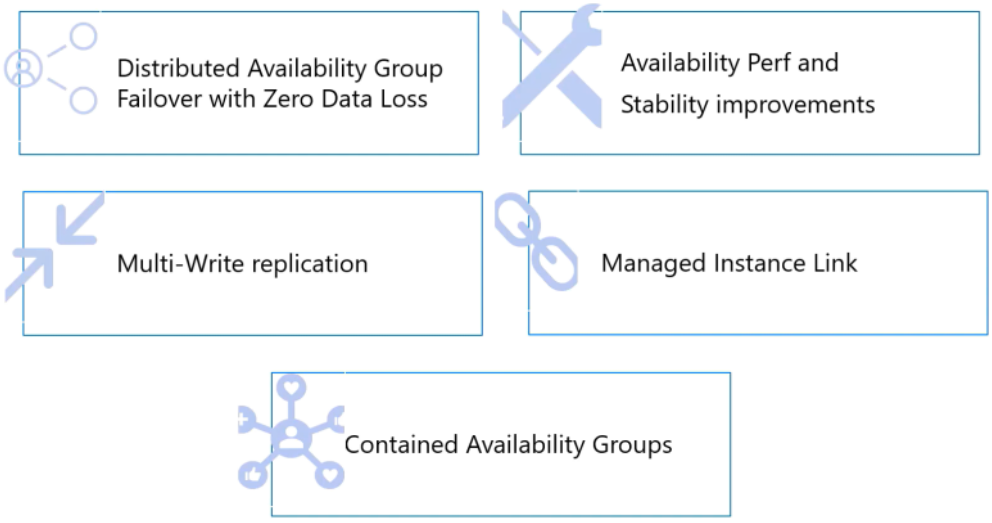
Shrink Database with Low Priority

- Customers often need to reclaim data space
- Common for hosted environments (new database per customer)
- Shrink Database operations can cause concurrency issues
- Shrink Database WLP addresses this problem by waiting with less restrictive locking
- Similar to ALTER INDEX WAIT_AT_LOW_PRIORITY

```
DBCC SHRINKDATABASE (2, 20, NOTRUNCATE)
WITH WAIT_AT_LOW_PRIORITY (ABORT_AFTER_WAIT = SELF))

DBCC SHRINKFILE (5, 7) WITH WAIT_AT_LOW_PRIORITY
(ABORT_AFTER_WAIT = BLOCKERS))
```

SQL Server 2022
High Availability New Feature Overview



Distributed Availability Groups

- Intended for remote locations
- Resolves quorum issues between sites
- Improves WAN network efficiency
- Enables mixed Windows versions

Distributed Availability Groups

- Challenges:
  - No central cluster manager
  - No coordination to synchronize failovers
  - **Very difficult to guarantee zero data loss in failovers.**

Zero Data loss Distributed AG Failover

- Add the ability to apply REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT at the Distributed AG level
- Ensures that no transaction is partially committed when failover happens.
- Takes a very complex error-prone operation and makes it simple and success-prone.

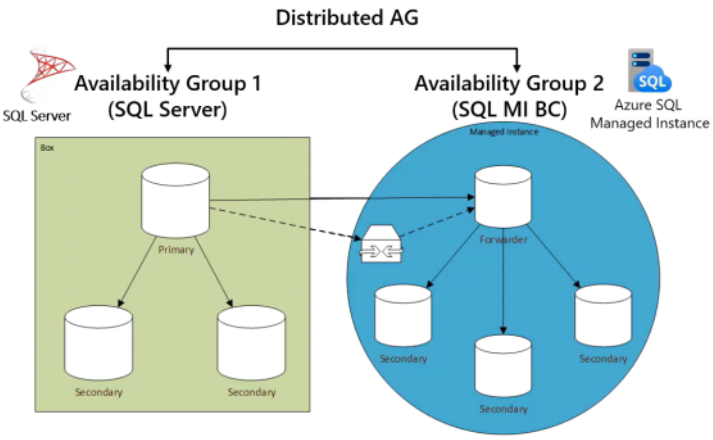
What is link and why use it?

- A common capability in the SQL Server and SQL Managed Instance that enables rich **hybrid connectivity scenarios** between the two.
- Empowers you to immerse into Azure at your **own pace**.
- **Modernize** in Azure while keep running on your **existing SQL Server**.
- When and if ready, **migrate** to Azure at your own pace with **minimum downtime**.
- Use Managed Instance as a **Disaster Recovery (DR)** site (**SQL Server 2022 only**)*

Link is based on distributed Availability Groups

Distributed Availability Groups

- Leveraging the proven Availability Group technology stack
- Connection between SQL Server and SQL MI is bridged using distributed AG
- Distributed AG spans two separate Availability Groups, across domains and across platforms
- WSFC is not required SQL Server 2017, 2019 and 2022.
- Single-node WSFC is required for SQL Server 2016.
- You can use mixed versions of SQL Servers to connect to SQL MI.



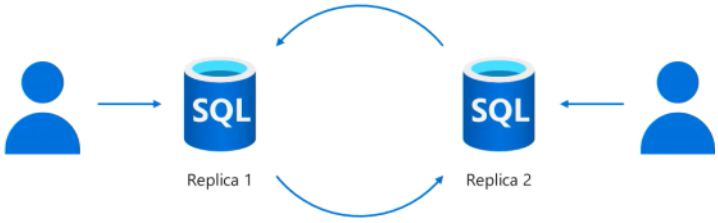
AG Perf and Stability improvements

- Two TCP channels for log replication in Distributed AG environments
  - Helps compensate for high latency
  - Improves throughput
- Redo Thread Pool enhancements
  - Current: Statically assign threads to databases as they start up (100 threads/DB) Remaining DBs get one thread each.
  - NEW: Global thread pool for redo work. Any database can post a work item to the pool and get a thread in FIFO order. Makes for more fair and efficient usage of execution threads.
- AlwaysOn_Health : Capture sp_server_diagnostics XEvent when STATE=3 (ERROR) To Diagnose HADR Health Events
- DCR: Add REVERTING progress to errorlog Just like SQL Already Reports Recover Progress
- DCR: Improve AlwaysOn_health and system_health logs, add XEvents that are helpful to understand database state, recovery and reverting progress
- Introduce unfair mutex into UCS session registration to improve AG data movement performance under multi-db scenario

Multi-write replication

Multi-master writes for users across multiple locations

- Globally distributed database replicas for geo-localized writes
- Enhanced conflict detection for inserts and updates with Last Writer Wins (LWW) capabilities
- Ensures the last update is persisted across all replicas based on the UTC time of the operation.



Contained Availability Groups

Problem Statement

- Objects or settings related to an AG do not get replicated across the instances in the AG
  - Users/Logins
  - Permissions
  - Jobs
  - Settings
- Applications need a consistent execution context to function properly
- Workarounds involve complex and fragile scripts to force synchronization or manual processes

Goals

- Provide a unified experience across all instances in the Contained AG
- Nearly all operations within the AG require no syntax change
  - The exception being operations specifically impacting a contained AG
- Preserve the operations and management of the individual instances in the Contained AG
- Enable multiple Contained AGs to coexist on the same set of instances without interference

Contained AG Syntax

(Only the syntax in RED is new)

```
CREATE AVAILABILITY GROUP MyAg
        WITH (CONTAINED, [REUSE_SYSTEM_DATABASES,] AUTOMATED_BACKUP_PREFERENCE =
SECONDARY, FAILURE_CONDITION_LEVEL = 3,                    HEALTH_CHECK_TIMEOUT = 600000 )
        FOR DATABASE ThisDatabase, ThatDatabase
        REPLICA ON
                'COMPUTER01' WITH ( ENDPOINT_URL = 'TCP://COMPUTER01:5022',
                        AVAILABILITY_MODE = SYNCHRONOUS_COMMIT)),
                'COMPUTER02' WITH ( ENDPOINT_URL = 'TCP://COMPUTER02:5022',
                        AVAILABILITY_MODE = SYNCHRONOUS_COMMIT))

ALTER AVAILABILITY GROUP [MyAg] ADD LISTENER 'MyAgListenerIvP6' ( WITH IP (
('2001:db88:f0:f00f::cf3c'),('2001:4898:e0:f213::4ce2') ) , PORT = 60173 ); GO
```

Managing the Contained AG

- NOTE: Contained Availability Groups do NOT represent a security boundary.
- To manage or interact with the Contained AG, connect to the contained AG:
  - Using the Contained AG Listener
  - Specifying a database within the Contained AG
    - USE MyAG_MSDB
- Database names must be unique across all instances

System Databases

- We create <AG-Name>_Master and <AG-Name>_MSDB in each Contained AG.
- They are initially empty but contain the subset of the instance master which we replicate. (Logins, users, permissions, jobs, etc.)
- They are automatically seeded to new AG replicas

[SQL Server Contained Availability Groups Configuration (mssqltips.com)]

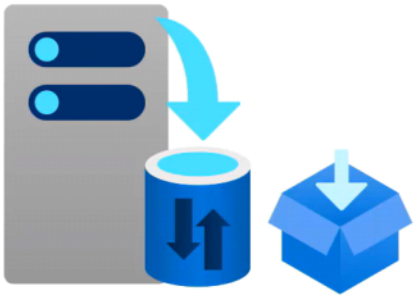
SQL Server 2022 - Backup Compression and Offloading with Intel® QuickAssist Technology (QAT)

Hardware acceleration for backup compression

Why compress backups?

- Reduce capacity
- Improve performance
  - Faster backup
  - Faster restore

- Existing backup compression products
  - Software based compression adds CPU overhead
  - Backup speed is variable
  - Depends on available CPU resources
  - More likely to impact workload



Intel® QuickAssist Technology (QAT) compression

- Intel QuickAssist Technology improves performance across applications, including symmetric encryption and authentication, asymmetric encryption, digital signatures, RSA, DH, and ECC, and lossless data compression.
- Intel QuickAssist Technology (QAT) provides a compression technology that integrates hardware acceleration of compute-intensive workloads on Intel platforms.

Database Backup Improvements for SQL Server 2022 with Intel QuickAssist Technology:

Up to 2.3x
speedup in database backup time[1]

Up to 6%
reduction in backup storage capacity[1]

SQL Server 2022 and Intel® QAT
Enabling Hardware Offloading

- Leverages Intel® QuickAssist Technology (Intel® QAT) for improved backup performance
- Free-up processor cycles by offloading backup compression
- Reduce demands on processor
- Improves backup speed

```
Enable the hardware offload feature for
SQL Server 2022:
sp_configure 'hardware offload enabled', 1
GO
RECONFIGURE
GO
```

```
Enable Intel® QuickAssist Technology (QAT)
hardware mode:
ALTER SERVER CONFIGURATION
SET HARDWARE_OFFLOAD = ON
(ACCELERATOR = QAT)
```

```
Enable Intel® QuickAssist Technology (QAT)
software mode:
ALTER SERVER CONFIGURATION
SET HARDWARE_OFFLOAD = ON
(ACCELERATOR = QAT, MODE = SOFTWARE)
```

```
Backup database with QAT compression:

BACKUP DATABASE [TicketReservations]
TO DISK = 'D:\backups\QAT-DEFLATE.bak'
WITH FORMAT, COMPRESSION (ALGORITHM = QAT_DEFLATE)
GO
```

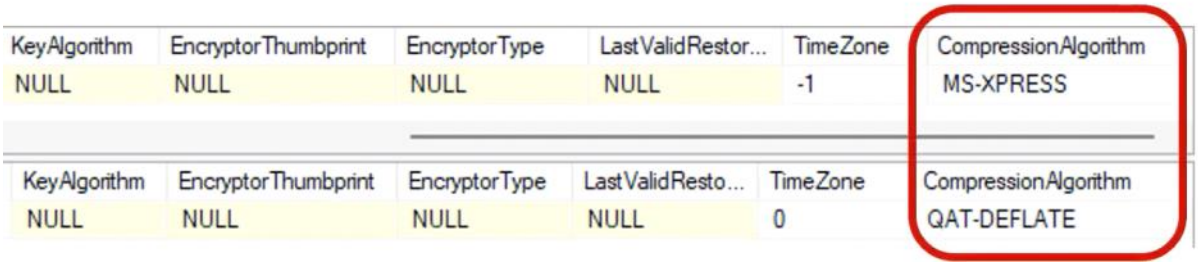
```
Backup database with the default compression:

BACKUP DATABASE [TicketReservations]
TO DISK = 'D:\backups\MS-XPRESS.bak'
WITH FORMAT
GO
```

SQL Server 2022 and Intel® QAT – Restore Headeronly

- Restore headeronly will display the compression algorithm
- Running Restore headeronly requires that the Intel® QuickAssist Technology (Intel® QAT) drivers are available

```
RESTORE HEADERONLY
FROM DISK = 'D:\backups\QAT-Deflate.bak'
GO
```



KeyAlgorithm	EncryptorThumbprint	EncryptorType	LastValidRestor...	TimeZone	CompressionAlgorithm
NULL	NULL	NULL	NULL	-1	MS-XPRESS

KeyAlgorithm	EncryptorThumbprint	EncryptorType	LastValidResto...	TimeZone	CompressionAlgorithm
NULL	NULL	NULL	NULL	0	QAT-DEFLATE

SQL Server 2022 and Intel® QAT – Restore Database

- SQL Server backups compressed using QAT_DEFLATE will support all T-SQL RESTORE operations
- Intel® QAT compressed backups require the server running the RESTORE DATABASE command to have the Intel® QAT drivers installed

```
Restore database command:
RESTORE DATABASE [testdb1]
FROM DISK = 'D:\backups\QAT-Deflate.bak'
GO
```

SQL Server 2022 and Intel® QAT - Troubleshooting
sys.dm_server_accelerator_status

- The SQL Server sys.dm_server_accelerator_status dynamic management view can be used to verify the configuration state
- The mode description reason is a valuable for troubleshooting
- Configured libraries and driver versions are visible

```
The DMV below will expose offloading potentials
and the configuration, one row per accelerator:

SELECT * FROM sys.dm_server_accelerator_status
GO
```

SQL Server 2022 and Intel® QAT - Troubleshooting

- Intel® QuickAssist Technology Landing Page https://developer.intel.com/quickassist
- SQL Server error log will show if hardware has been detected along with the accelerator mode

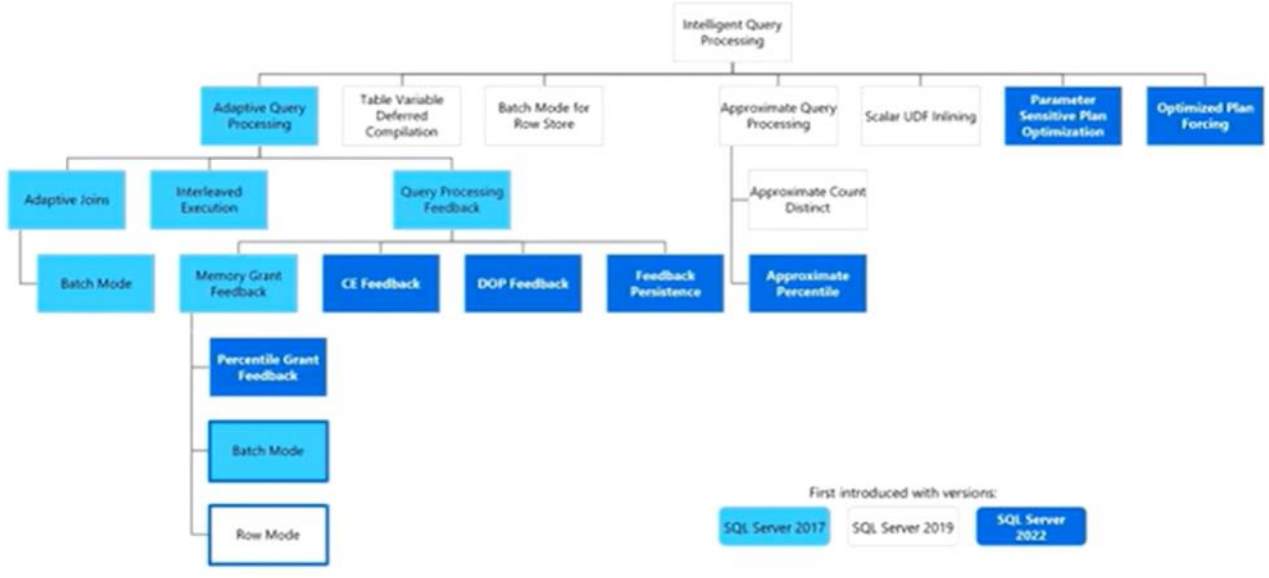
	LogDate	ProcessInfo	Text
1	2022-08-16 23:35:05.420	Server	Detected Intel(R) QuickAssist Compression Library QATZip: 1.8.0.10.
2	2022-08-16 23:35:05.420	Server	Detected Intel(R) QuickAssist Kernel Driver icp_qat: 3.2.0.9.
3	2022-08-16 23:35:05.420	Server	Detected Intel(R) Storage Acceleration Library ISA-L: 2.30.0.0.
4	2022-08-16 23:35:05.420	Server	Intel(R) QuickAssist Technology (QAT) initialization succeeded.
5	2022-08-16 23:35:05.420	Server	Intel(R) QuickAssist Technology (QAT) hardware detected on the system.
6	2022-08-16 23:35:05.420	Server	Intel(R) QuickAssist Technology (QAT) will be used in hardware mode.
7	2022-08-16 23:35:05.420	Server	Intel(R) QuickAssist Technology (QAT) session sucessfully created.

- Some Intel QAT Backup Compression Results
- How to Enable Intel QAT Backup Compression in SQL Server 2022
- How to Install the Intel QAT Driver

```
--#8. Using msdb and the compression_algonthm via backupset for backup performance history
SELECT bs.databasename,
backuptype = CASE
WHEN bs.type = 'D'
AND bs.is_copy_only 0 THEN 'Full Database'
WHEN bs.type = 'D'
AND bs.is_copy_only = 1 THEN 'Full Copy-Only Database'
WHEN bs.type = 'I' THEN 'Differential database backup'
WHEN bs.type = 'L' THEN 'Transaction Log'
WHEN bs.type = 'F' THEN 'File or filegroup'
WHEN bs.type = 'G' THEN 'Differential file'
WHEN bs.type = 'P' THEN 'Partial'
```

```
WHEN bs.type = 'Q' THEN 'Differential partial'
END + ' Backup',
CASE bf.device_type
WHEN 2 THEN 'Disk'
WHEN 5 THEN 'Tape'
WHEN 7 THEN 'Virtual device'
WHEN 9 THEN 'Azure Storage'
WHEN 105 THEN 'A permanent backup device'
ELSE 'Other Device'
END AS DeviceType,
bms.software_name AS backupsoftware,
bs.recovery_model,
bs.compatibility_level,
BackupStartDate = bs.Backup_Start_Date,
BackupFinishDate = bs.Backup_Finish_Date, DATEDIFF(ms, backup_start_date, backup_finish_date) AS
ms_taken,
LatestBackupLocation = bf.physical_device_name,
backup_size_mb = CONVERT(decimal(10, 2), bs.backup_size/1024./1024.),
compressed_backup_size_mb = CONVERT(decimal(10, 2), bs.compressed_backup_size/1024./1024.),
(CONVERT(decimal(10, 2), bs.backup_size/1024./1024.)) - (CONVERT(decimal(10, 2),
bs.compressed_backup_size/1024./1024.)) as compression_savings, bs.compressionalgorithm
FROM msdb.dbo.backupset bs
LEFT OUTER JOIN msdb.dbo.backupmediafamily bf ON bs.[media_set_id] = bf.[media_set_id]
INNER JOIN msdb.dbo.backupmediaset bms ON bs.[media_set_id] = bms.[media_set_id]
WHERE bs.backup_start_date > DATEADD(DAY, -72, sysdatetime()) --only look at last 12 hours
ORDER BY bs.Backup_Start_Date DESC, bs.databasename ASC
```

IQP Family Tree




```
Query Store - Custom capture policies
ALTER OATABASE [QueryStoreDB]
SET QUERY_STORE = ON
(
OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = ( STALE_QUERY_THRESHOLD_DAYS = 90 ),
DAT A_F LUSH_INTE RVAL_SE CONDS = 900,
MAX_STORAGE_SIZE_MB = 1000, INTERVAL_LENGTH_MINUTES = 60, SIZE_BASED_CLEANUP_MODE =
AUTO, MAX_PLANS_PER_QUERY = 200, WAIT_STATS_CAPTURE_MODE = ON,
QUERY_CAPTURE_MODE = CUSTOM,
QUERY_CAPTURE_POLICY = (
STALE_CAPTURE_POLICY_THRESHOLD = 24 HOURS, EXECUTION_COUNT = 30,
TOTAL_COMPILE_CPU_TIME_MS = 1000, TOTAL_EXECUTION_CPU_TIME_MS = 100 )
```

);

What are Query hints?

- Ideally the Query Optimizer selects an optimal execution plan
- Developers and DBAs often need to influence plan behavior but historically had few options outside of modifying application code
- Query hints are specified via the OPTION clause to influence the behavior of operators in a statement

Example:

```
SELECT COUNT(DISTINCT [WWI Order ID])
FROM [Fact].[OrderHistoryExtended]
OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE);
```

Applying Query hints Today

- Query hints help provide solutions to various performance related issues – but they do require a rewrite of the original query

- DBAs often cannot make changes directly to T-SQL code
  - T-SQL hard-coded into application
  - T-SQL automatically generated by the application

- DBA may have to rely on plan guides
  - Common feedback: "plan guides are complex to use and manage"

Introducing Query Store hints

Query Store hints provides a method to shape query plans without changing code

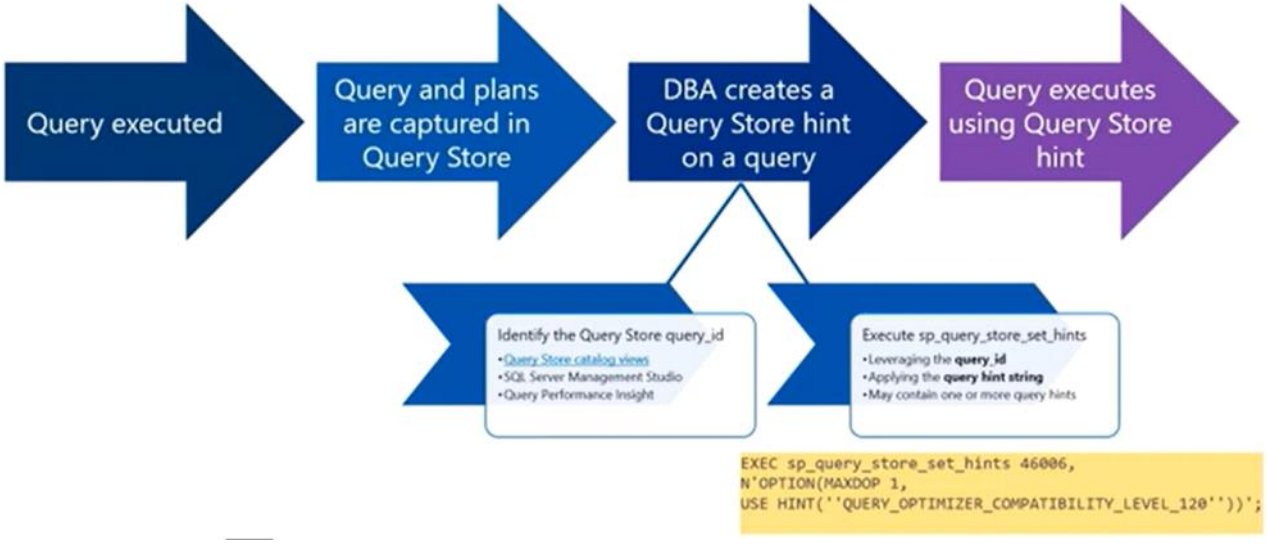
Query Store hints extends the power of Query Store

Query Store hints are persisted, surviving restarts

Override other hard-coded statement level hints and plan guides

Queries will always execute as opposing Query Store hints will be ignored

Using Query Store hints



[Query Store hints in SQL Server 2022 - Microsoft SQL Server Blog]

Feedback Workflow

User executes a query

If query qualifies for feedback, possible changes are suggested to the system

Changes are tried, and if they improve query performance, are verified and persisted

Cardinality Estimation – What is the model?

- Basic set of assumptions that determine how we interpret statistics and predicates used in the query
- There has historically been one "model" for CE, but not all queries perform best with this model
- CE Feedback allows us to modify the underlying assumptions for a specific query – changing the model for that query and improving performance

What is CE Feedback?

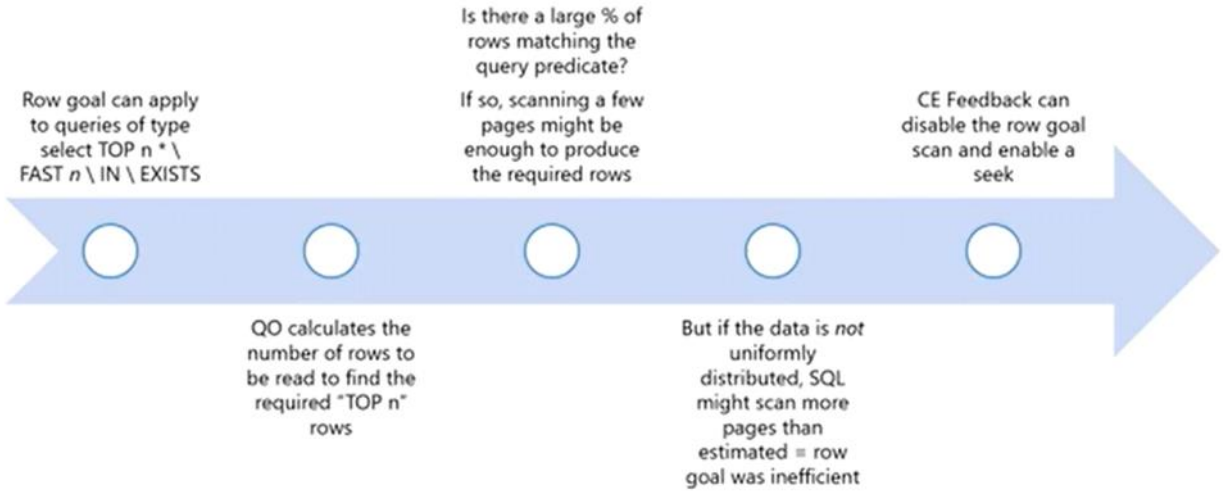
- Start with default model assumption
- Modify model assumption based on observations about query execution
- Validate that new model assumption is better

Model assumption 1: Uniformity

Uniformity is used when interpreting data from on-disk histograms

Data within histogram buckets is assumed to be uniform

CE Feedback – Row goal Analysis



Model assumption 2: Independence (or not…)

- Calculating the selectivity of a conjunction of predicates that are independent is done by multiplying their individual selectivities
- Quite often, predicates are not independent
- …and sometimes they are not completely correlated, either