Buffer Pool Challenge

OLTP workload characteristics

- High volumes which access to small amount of data
- Generates many random IOPS to HDD

Buffer Pool Challenge

- Long wait times for PAGE_IO_LATCH

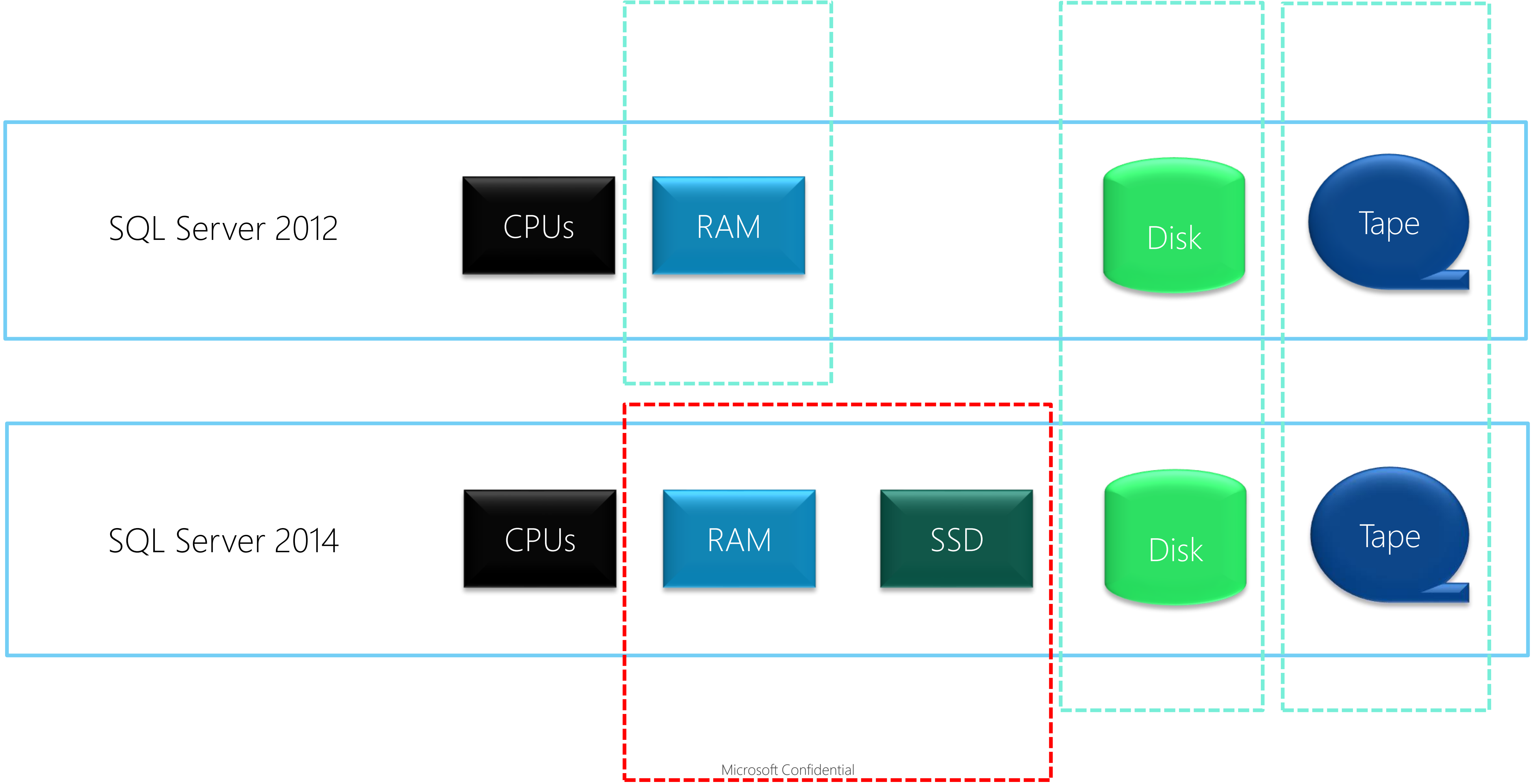
Why Buffer Pool Extensions?

OLTP workloads can have IO bottlenecks due to Random Disk access

Need an affordable way to increase OLTP workload performance

Adding memory can be prohibitive due to hardware limitations & cost

New Layer For Data Placement

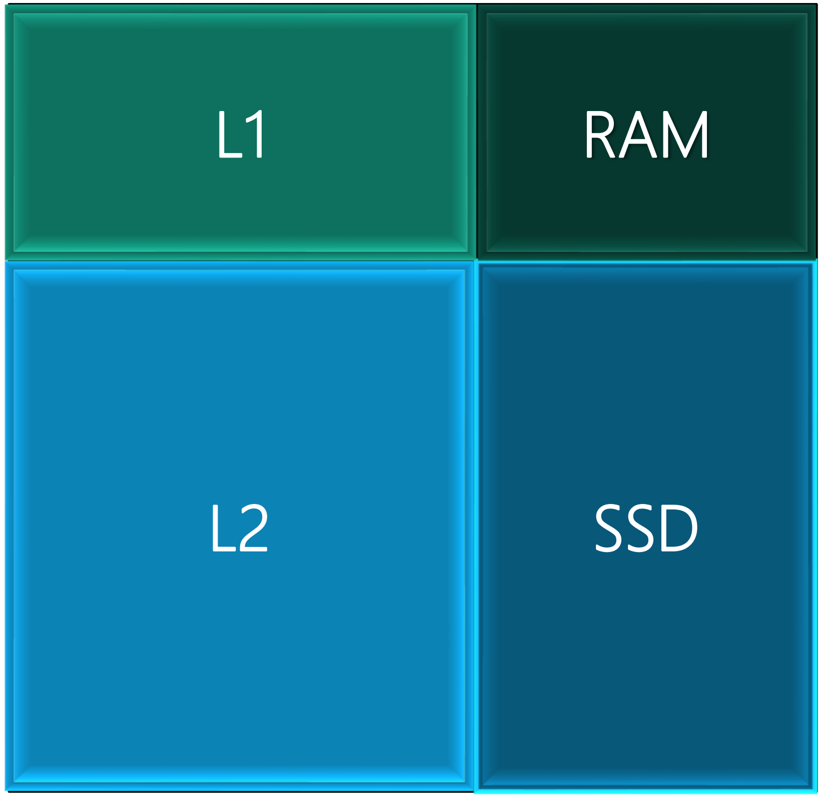


Terminology

L1 cache: Buffer Pool on RAM

L2 cache: Buffer Pool Extension on SSD

Buffer: An 8KB page in memory



L1

RAM

L2

SSD

Benefits

Use of non-volatile drives (SSD) to extend buffer pool

- Reduced IO Latency
- Increased Random IO throughput.

NUMA-Aware large page and BUF array allocation

Improve OLTP query performance with no application changes

No risk of data loss (using clean pages only)

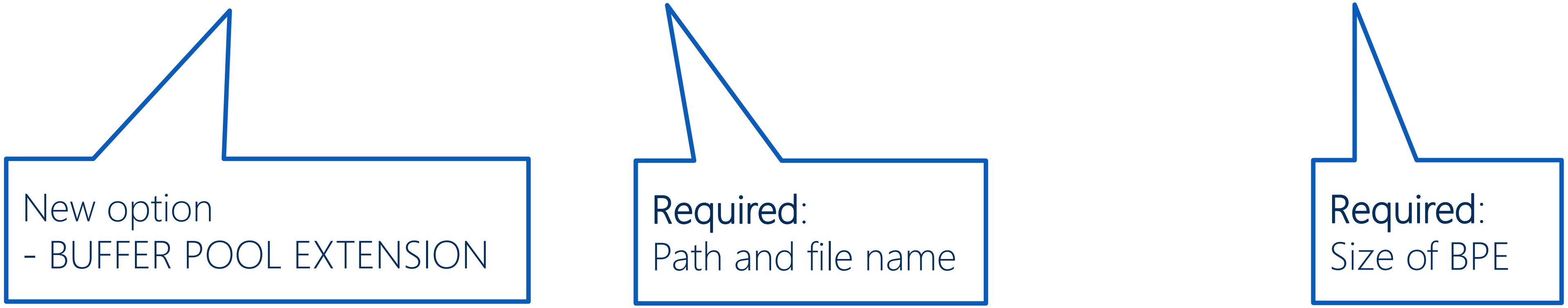
Optimized for OLTP workloads on commodity servers

Syntax

New configuration option - BUFFER POOL EXTENSION

Need to specify the size equal or greater than BUFFER POOL

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION ON (FILENAME = ‘os_file_path_and_name’, SIZE = {SIZE KB/MB/GB}) [;]
```



New option
- BUFFER POOL EXTENSION

Required:
Path and file name

Required:
Size of BPE

Commands

- Check if enabled

```
SELECT * FROM sys.dm_os_buffer_pool_extension_configuration
```

- Disable

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION OFF
```

- Which pages are in Buffer Pool Extensions?

```
SELECT * FROM sys.dm_os_buffer_descriptors
WHERE is_in_bpool_extension = 1
```

- Need to turn configuration OFF and ON to change value

Extension File and Device

No risk of data loss in extension file

- It only contains clean buffers

If SSD device is inaccessible

- SSD Buffer Pool Extension is disabled
- Need to re-enable

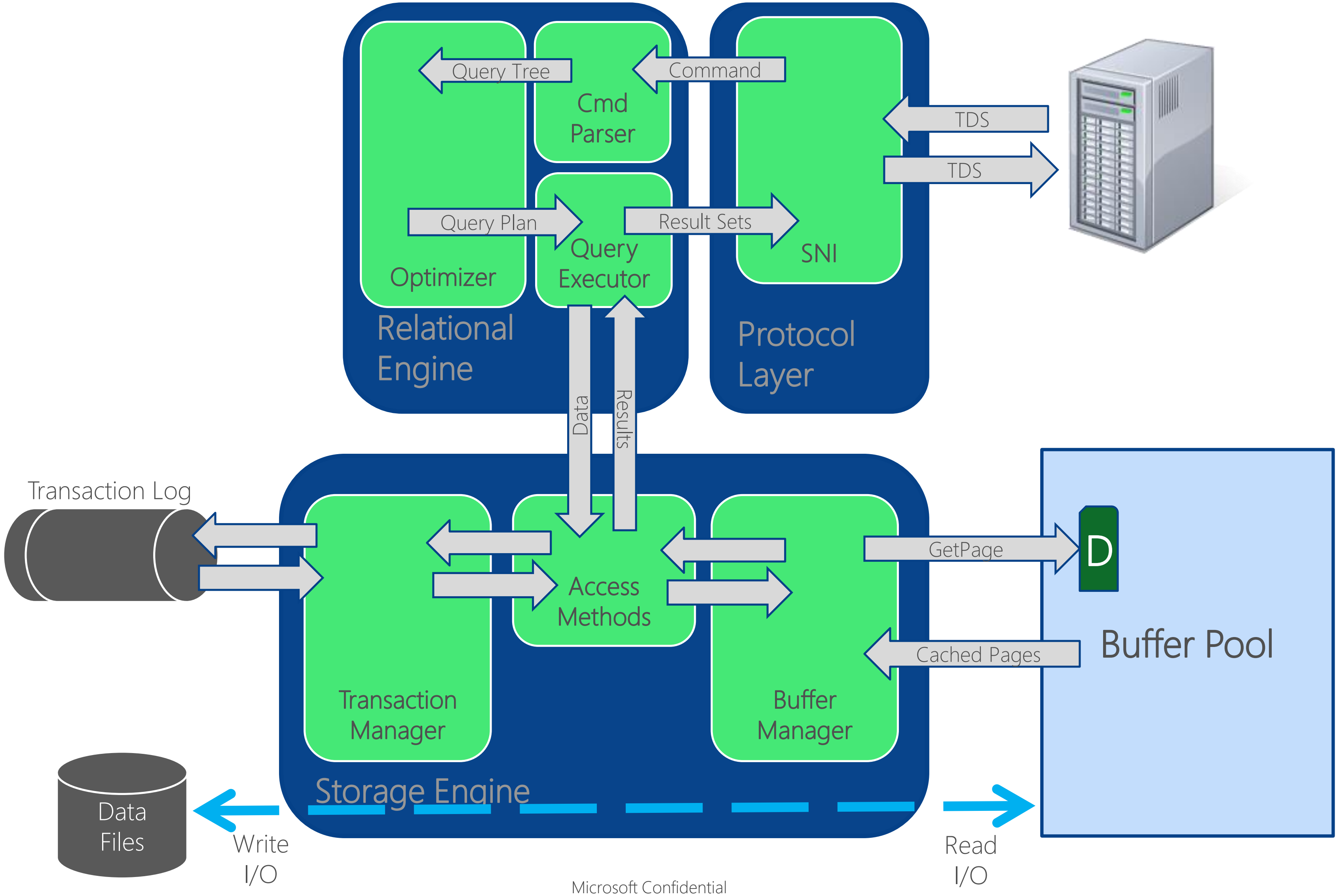
On a Cluster

- Ensure it is sized equally on both nodes

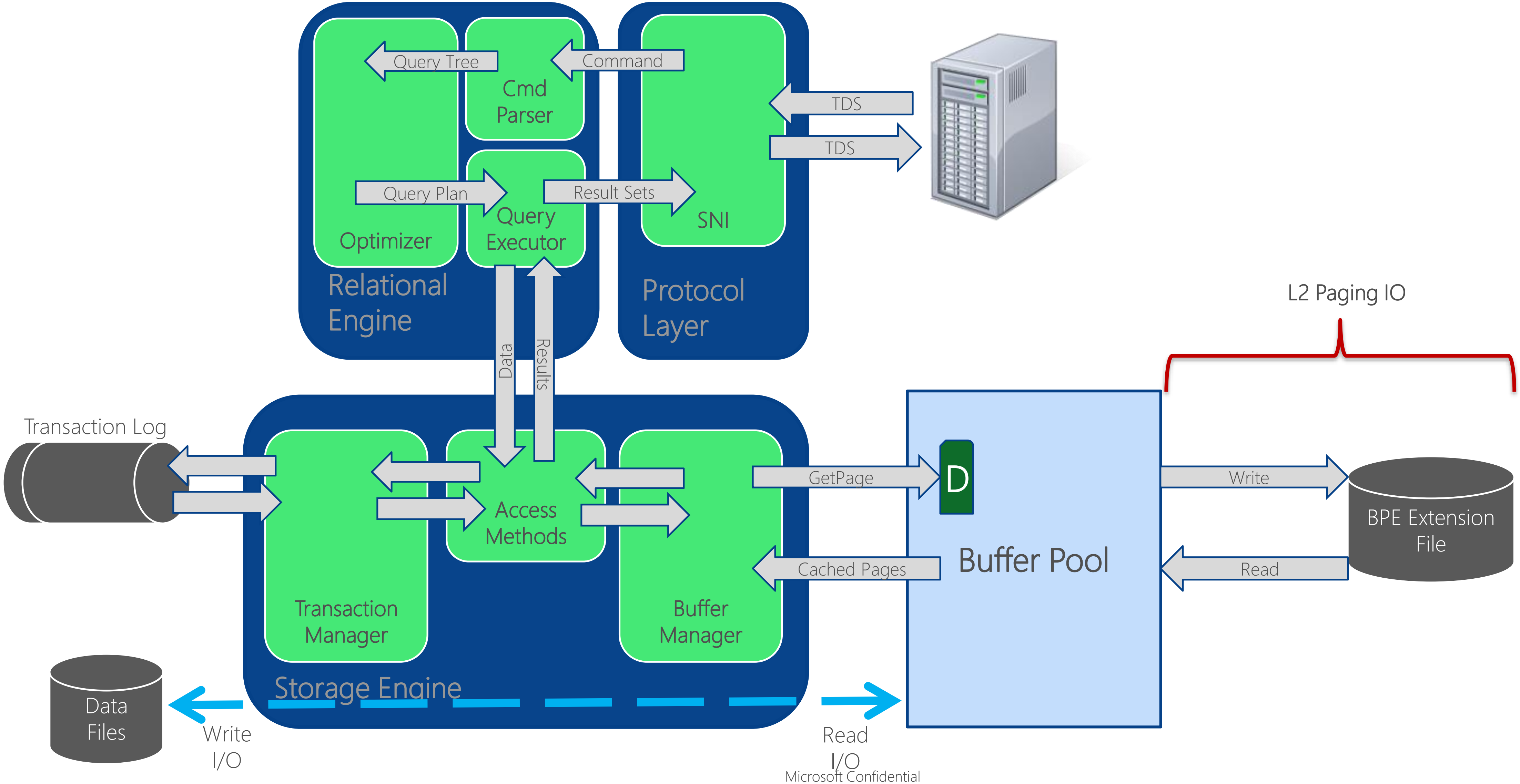
Limitations

- Memory Optimized Tables
- ColumnStore Index

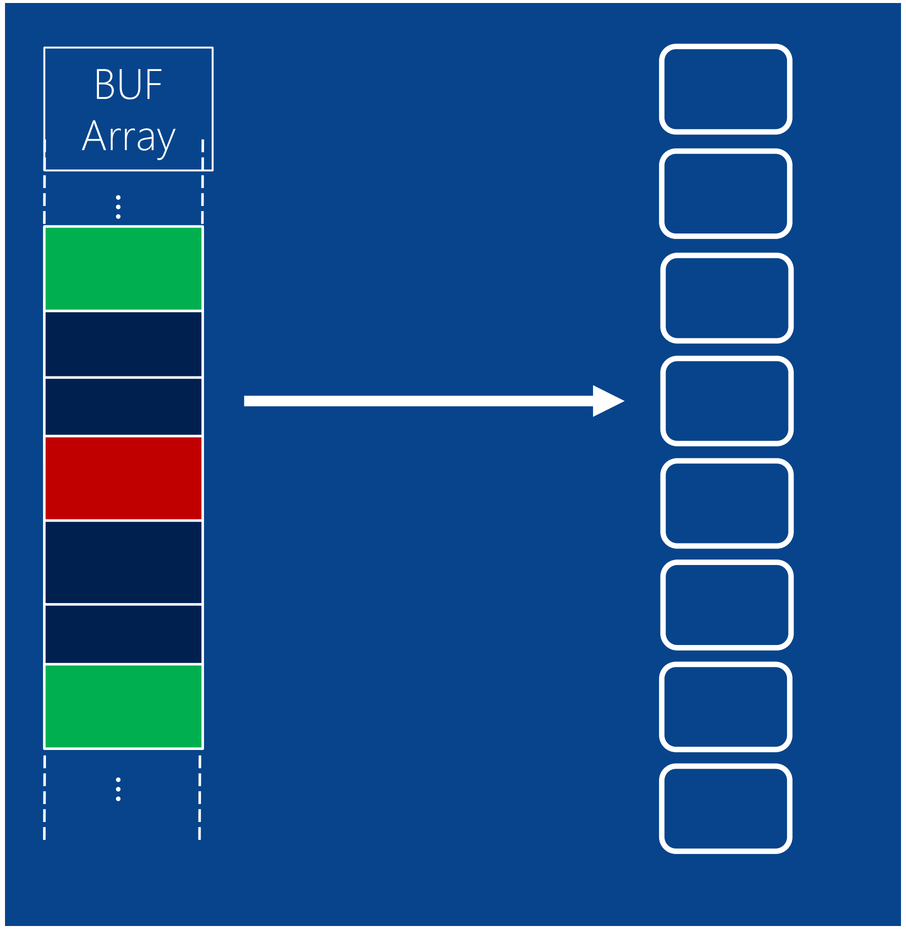
Buffer Pool Manager



Buffer Pool Manager – With Extensions



SSD Buffer Pool Extension Internals

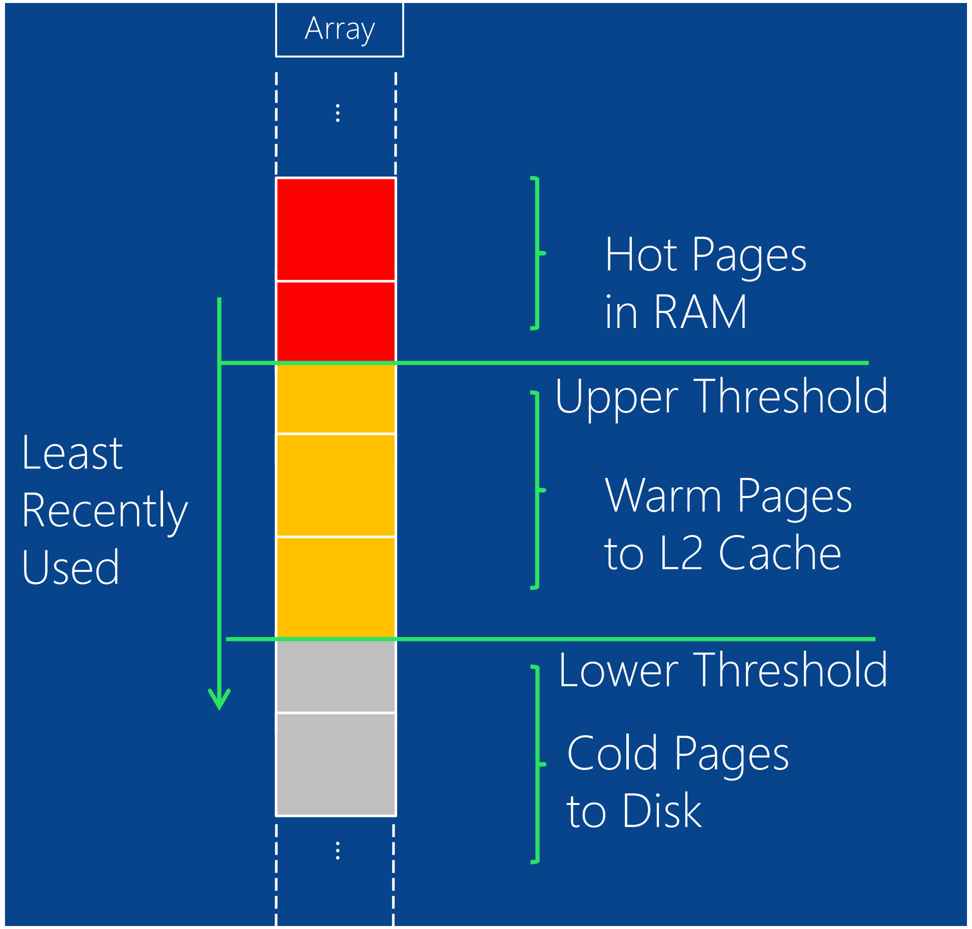


Managing the pages on the SSD buffer pool extension

It is managed as a 8 KB page

Using BUF Array

Data Rotation Algorithm



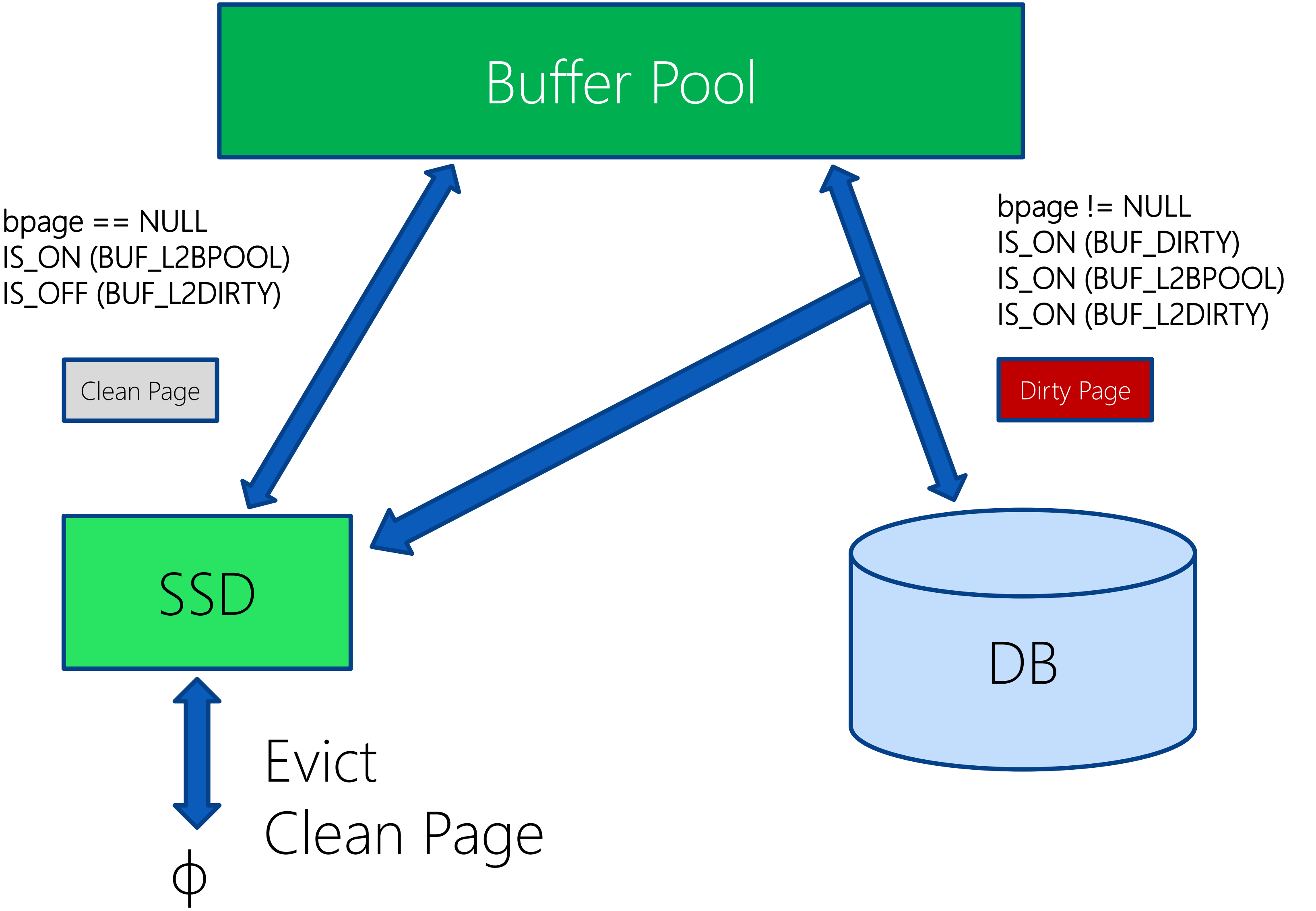
Data placement corresponding to usage frequency
- LRU algorithm

Upper Threshold > Reference counts
- Buffer Pool on RAM (L1 cache)

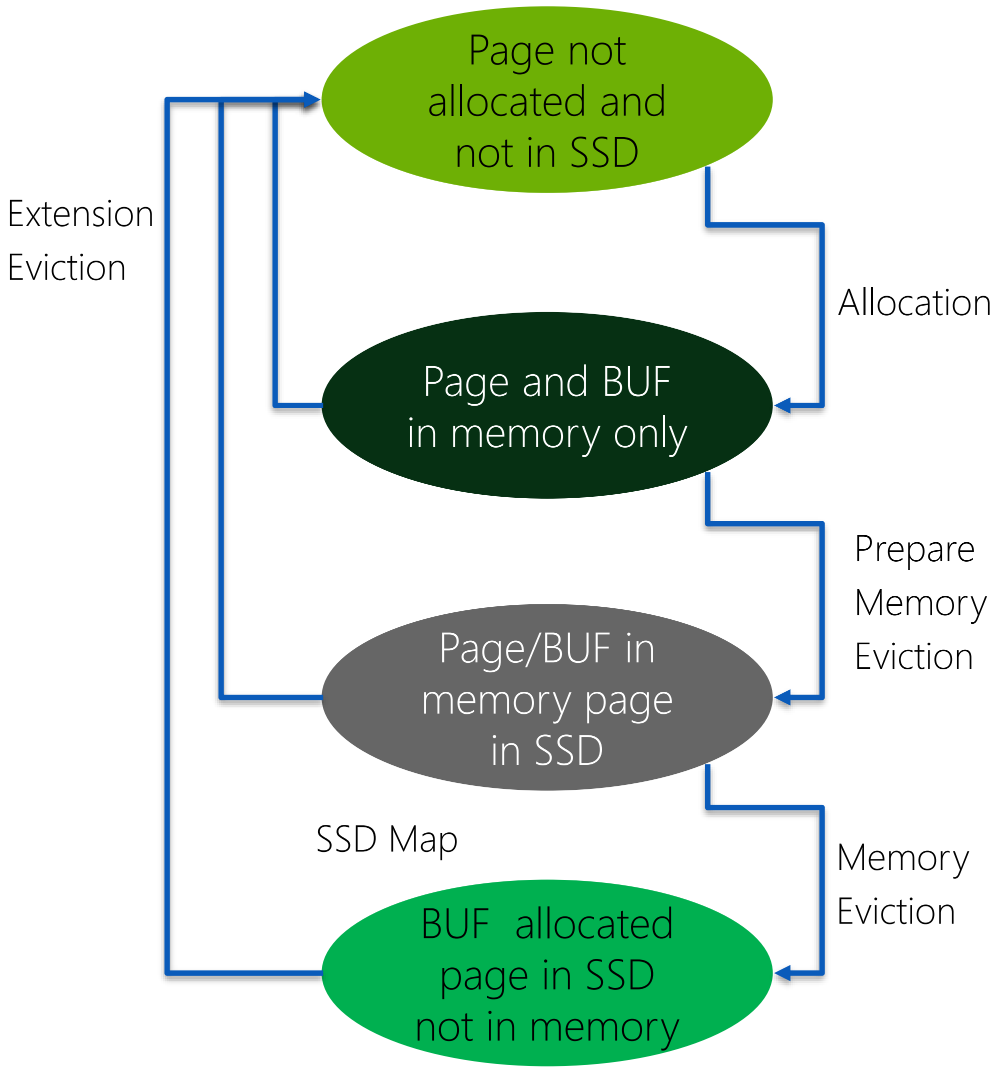
Upper Threshold > Reference counts > Lower Threshold
- Buffer Pool on SSD (L2 cache)

Lower Threshold < Reference counts
- On mechanical disk

Implementation



Page Eviction


Page not allocated and not in SSD
Extension Eviction
Allocation
Page and BUF in memory only
Prepare Memory Eviction
Page/BUF in memory page in SSD
SSD Map
Memory Eviction
BUF allocated page in SSD not in memory

Memory Eviction:

- Migrates a page from virtual memory to the L2 Cache

Extension Eviction:

- Writes dirty pages out to spinning media, frees page from RAM and SSD

Performance Counters

SQL Server Buffer Manager

- Extension page writes/sec
- Extension page reads/sec
- Extension outstanding IO counter
- Extension page evictions/sec
- Extension allocated pages
- Extension free pages
- Extension page unreferenced time
- Extension in use as percentage

xEvents and DMVs

xEvents:

- buffer_pool_extension_pages_written
- buffer_pool_extension_pages_read
- buffer_pool_extension_pages_evicted
- buffer_pool_page_threshold_recalculated

DMV:

- sys.dm_os_buffer_descriptions
- sys.dm_os_buffer_pool_extension_configuration

Demonstration: Buffer Pool Extensions

Best Practices

Enough on-board memory

- No benefit from Buffer Pool Extension

Target Workload

- OLTP - Read heavy preferred
- A large workload
- Multiple smaller workloads (consolidation)
- Systems that have memory pressure
- Sweet spot: SSD storage sized 4x-10x times of RAM size

Managed Lock Priority - What is it trying to solve?

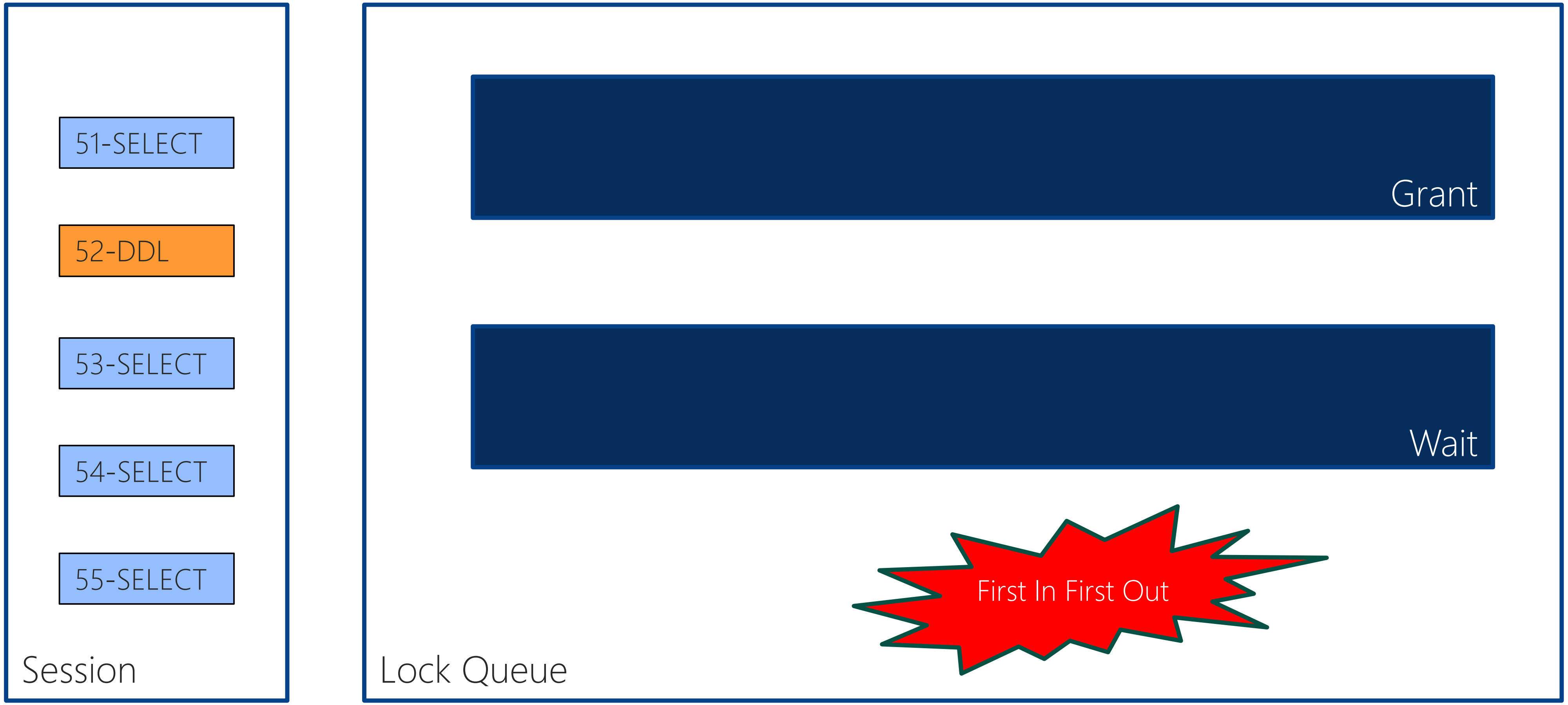
Blocking by online DDL

- Partition switching - short duration Sch-M lock acquired
- Online index rebuilds - short table S and Sch-M lock acquired

Concurrency and availability

- Blocking transactions need to complete before DDL
- Switch will block new transactions
- Timeouts, workload slowdowns
- Impact to mission critical OLTP workloads

Without Managed Lock Priority



Syntax

New clause in existing T-SQL DDL for ALTER TABLE and ALTER INDEX

Syntax

```
<low_priority_lock_wait>::=
  {
    WAIT_AT_LOW_PRIORITY ( MAX_DURATION =
<time>[MINUTES],

        ABORT_AFTER_WAIT = { NONE | SELF | BLOCKERS } )
  }
      NONE - current behavior
      SELF  -  abort DDL
      BLOCKERS – abort user blockers
```

```
ALTER TABLE stgtab SWITCH PARTITION 1 TO parttab
PARTITION 1
      WITH (WAIT_AT_LOW_PRIORITY (MAX_DURATION=
60 minutes,
        ABORT_AFTER_WAIT=BLOCKERS))

 ALTER INDEX clidx ON parttable REBUILD
   WITH ( ONLINE=ON (WAIT_AT_LOW_PRIORITY
(MAX_DURATION= 300,
        ABORT_AFTER_WAIT=SELF)) )
```

Examples

ABORT_AFTER_WAIT Summary

Kill All Blockers (BLOCKERS)

- Blocking user transactions killed
- MAX_DURATION in minutes

Requires ALTER ANY CONNECTION permission to kill sessions

Switch to Normal Queue (None)

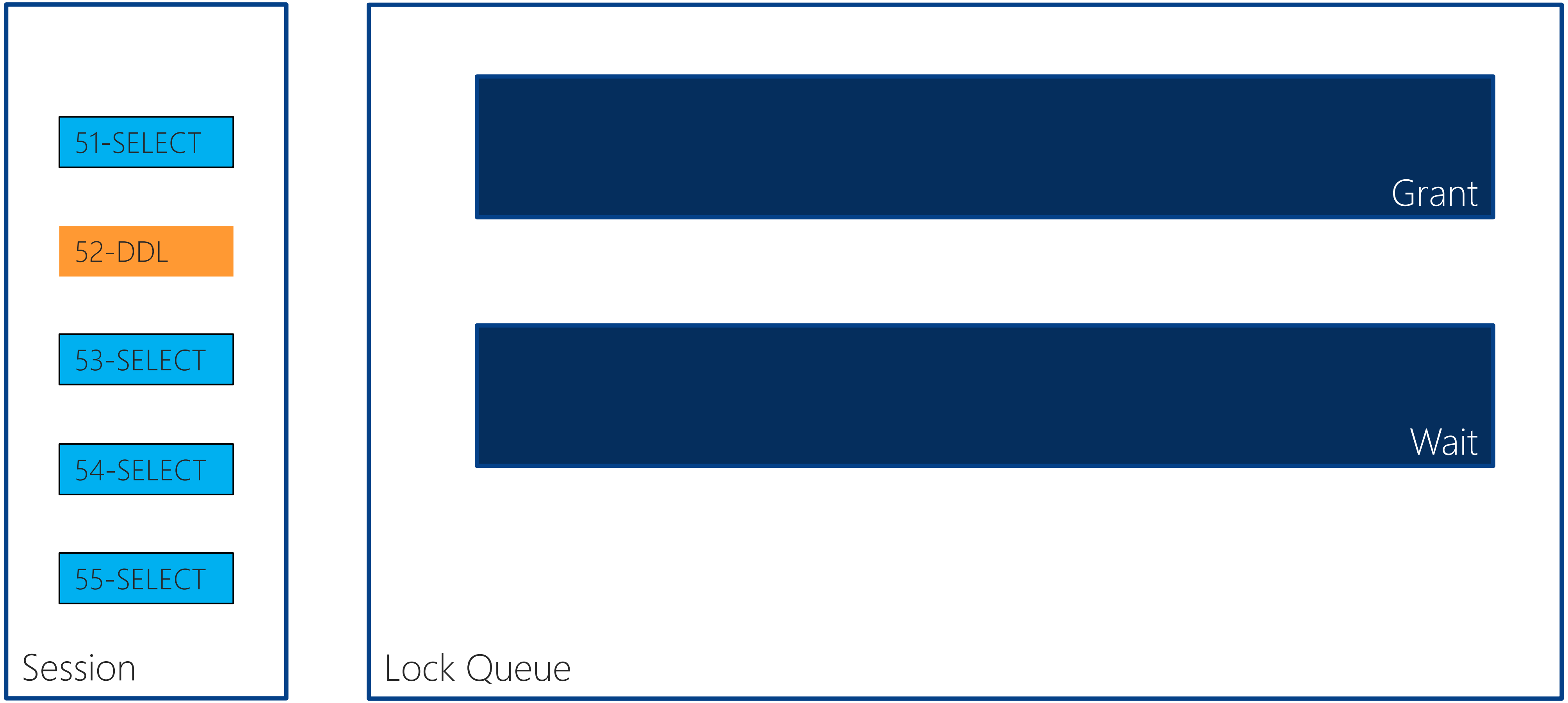
- Wait for blockers
- MAX_DURATION in minutes
- Regular FIFO lock queue

Exit DDL after wait (SELF)

- Wait for blockers
- MAX_DURATION in minutes
- Terminates DDL after wait time

If no blockers, lock granted immediately, DDL completes

Behavior with Managed Lock Priority



Managed Lock Priority Summary

Benefits

- Managed by DBA for both partition switch and Online Index Rebuild
- Lock request placed in lower priority queue
- Decision to wait or kill self or blockers
- Executed immediately if no blockers

Partition Switch

- Blockers killed at source and destination
- Sch-M lock (source and destination)

Online Index Rebuild

- MAX_DURATION applies to every lock request
- Time reset for every S & Sch-M lock
- Only Sch-M lock conflict for read only workloads

Best Practices

ABORT_AFTER_WAIT=BLOCKERS	• Long running queries/transactions • May not be beneficial in workloads with short transactions
Locking	• SWITCH acquires locks on source and target tables • Wait on the second lock affects concurrency on first • Dormant tables
DDL Transaction	• Nested transaction holds locks longer
Transaction Log Growth	• Log truncation affected by MAX_DURATION • Extra logging Online Index Rebuild DDL operation • Can lead to system performance issues

Single Partition Online Index Rebuild

Accessibility

- One or more partitions can be rebuilt
- Table accessible for DML and query operations
- Short term locks beginning and end of the index rebuild

Lock Priority

- Use Managed Lock Priority with Single Partition Online Index Rebuild

Availability

- Reduced down time for mission critical workloads

Resource savings

- CPU, memory and disk space
- Log space usage reduced

Managed Lock Priority Diagnostics

DMVs

- sys.dm_tran_locks “request_status” →LOW_PRIORITY_CONVERT, LOW_PRIORITY_WAIT, or ABORT_BLOCKERS
- sys.dm_os_wait_stats “wait_type” → ...LOW_PRIORITY and ..ABORT_BLOCKERS

Extended Events

- lock_request_priority_state
- process_killed_by_abort_blockers
- ddl_with_wait_at_low_priority
- sqlserver.progress_report_online_index_operation

Error Log

- Abort session diagnostics in SQL Server Error Log
- Deadlock diagnostics in deadlock graph

SQL Server & Statistics

What are Statistics?

- Distribution of values
- One or more columns

Why are they important?

- Plan choices
- Query performance

How are they maintained?

- Automatic update triggered by at least 20% of the data change
- For Larger Tables, TF 2371 (Auto Update Stats Changes - http://blogs.msdn.com/b/saponsqlserver/archive/2011/09/07/changes-to-automatic-update-statistics-in-sql-server-traceflag-2371.aspx)

Challenges Prior to SQL Server 2014

- Statistics become stale
- SCANs or samples the entire table
- Ascending keys
- One stats blob for whole table

Incremental Statistics

KB 2970386 - Metadata Corruption Fix

A feature targeted for partitioned tables

- Assuming most updates in newest partitions

CREATE STATISTICS

- One statistical object for each partition
- Binary merge of partial statistics
- A complete tree of statistics persisted

UPDATE STATISTICS

- Rebuild statistics on a subset of partitions
- Merge with rest of the statistics tree

Enabling Incremental Statistics

```
CREATE STATISTICS stat ON tbl (col1, col2)
WITH INCREMENTAL = ON

UPDATE STATISTICS tbl (stat)
WITH RESAMPLE ON PARTITIONS (1, 3, 5)

UPDATE STATISTICS tbl (stat)
WITH INCREMENTAL = { ON | OFF }

ALTER DATABASE DbName
SET AUTO_CREATE_STATISTICS ON (INCREMENTAL = { ON | OFF })

CREATE INDEX myind ON tbl(col)
WITH (STATISTICS_INCREMENTAL = ON)
```

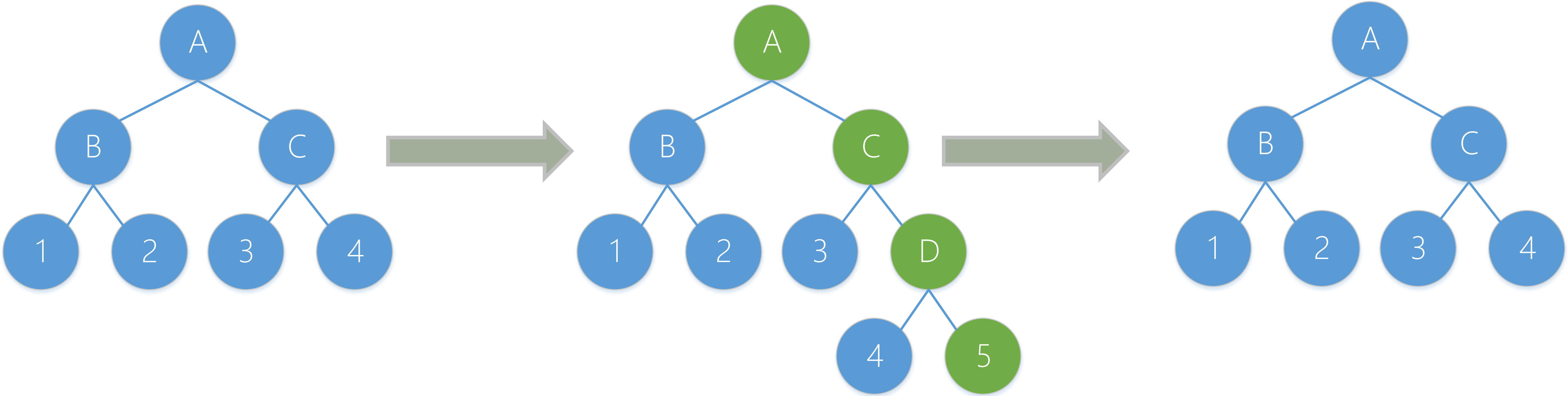
Not enabled by default

Incremental Stats Example

Create Incremental Statistics on a 4 partition table

Add partition 5 and update statistics

Merge partitions 4 and 5, and update statistics



Query compilation

Still uses global table stats for query optimization

- No per-partition statistics for optimization yet
- Consider Filtered Statistics

Threshold for auto stats update and query recompile

- To update global : 500 + 20% of **average** partition size (non-empty)
- To update per-partition : 20% data change in a partition

Default sample rate for incremental stats

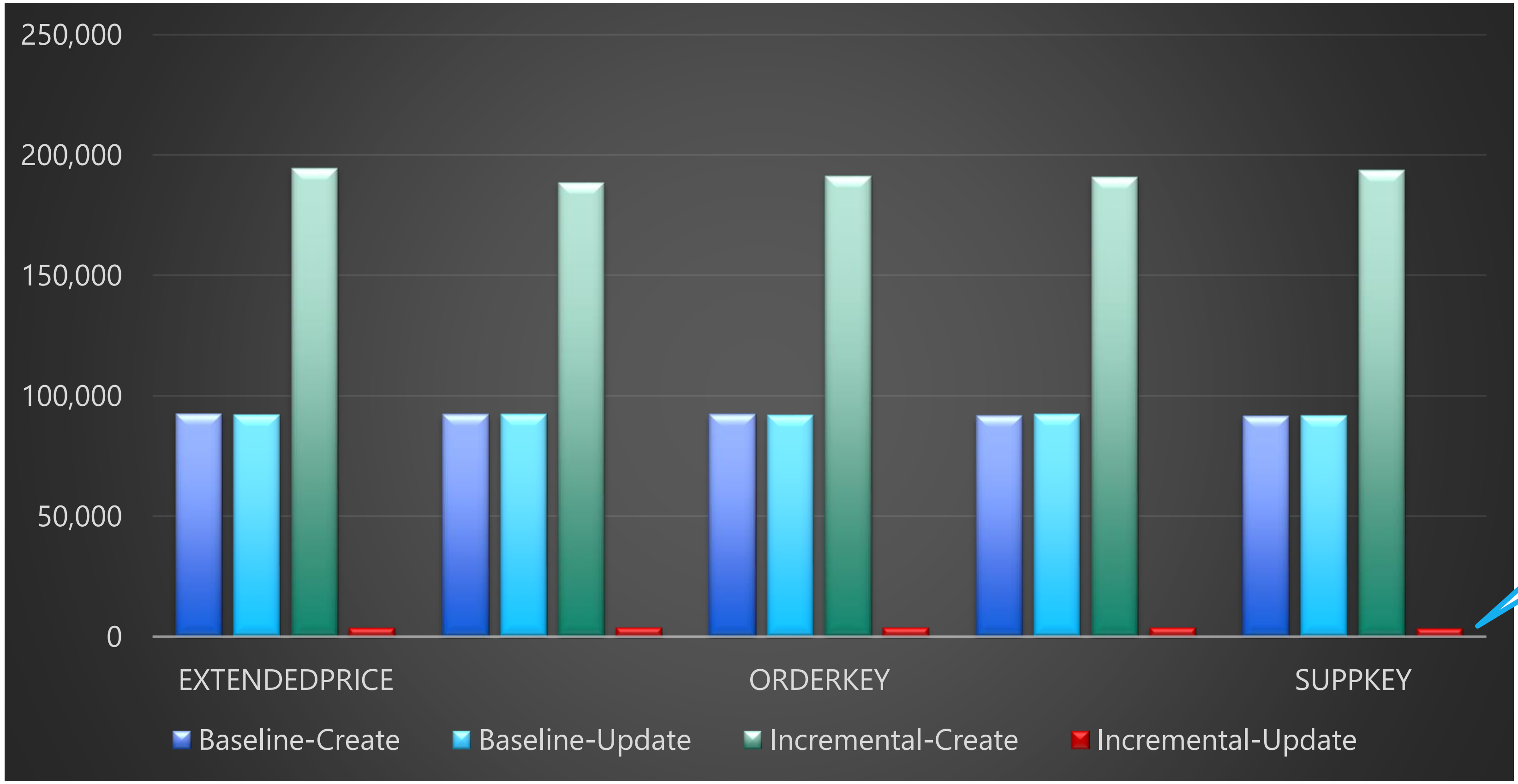
- Auto create : same default sample rate as regular stats
- Auto update : using most recent sample rate

Performance Results- Sample

Database **TPCH 100G**

Table **LINEITEM** **600 million** rows in **2,500** partitions

Measure elapsed time for create/update statistics with **sample size = 1 million rows**



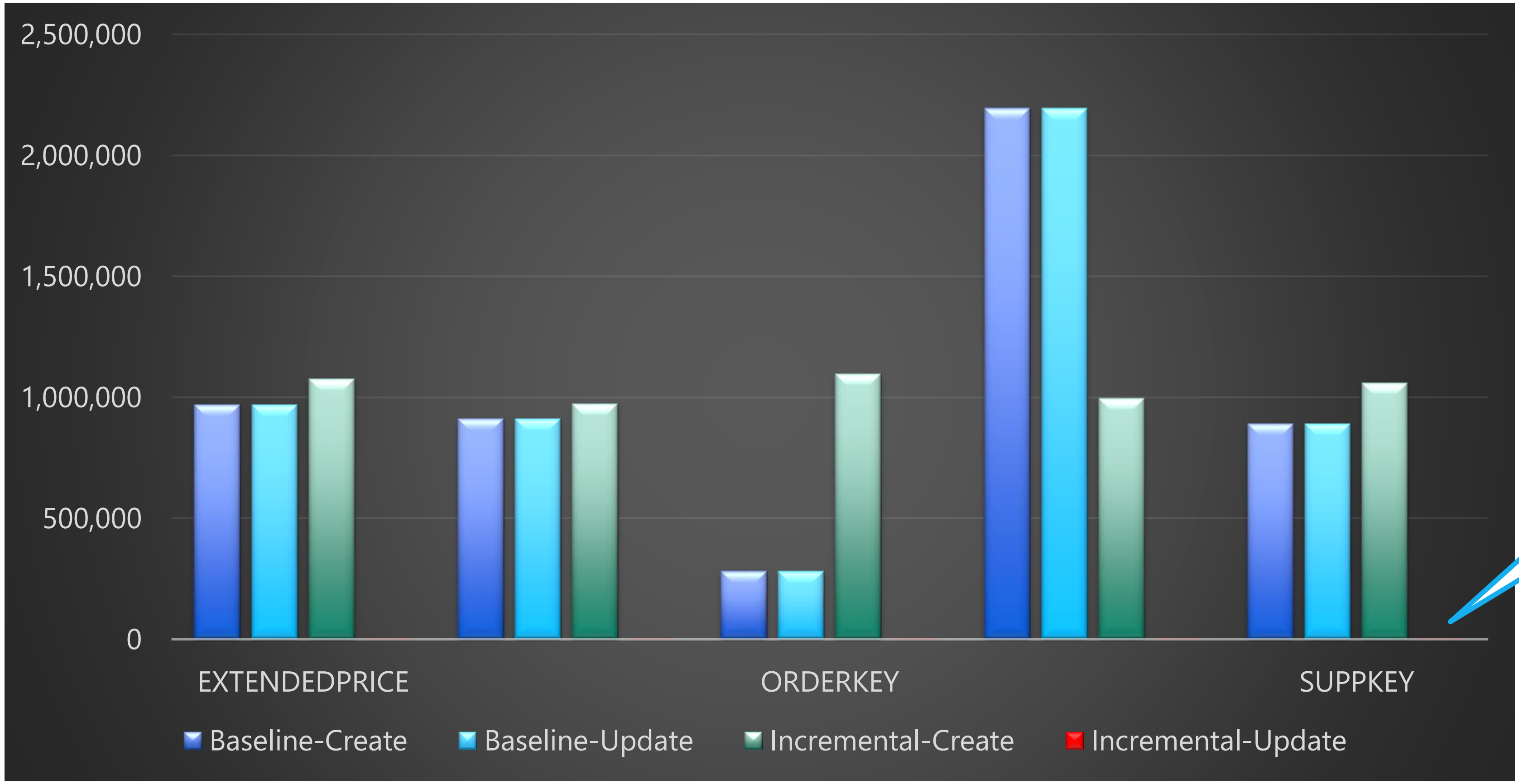
Incremental Update Statistics

Performance Results - FULLSCAN

Database **TPCH 100G**

Table **LINEITEM** **600 million** rows in **2,500** partitions

Measure elapsed time for create/update **FULLSCAN** statistics



Incremental Update Statistics

Limitations

- Indexes that are not partition-aligned with the base table
- AlwaysOn readable secondary databases
- Read-only databases
- Filtered indexes
- Views
- Internal tables
- Spatial indexes or XML indexes

Incremental Statistics Diagnostics

sys.stats

- sys.stats has a column *is_incremental* to indicate whether a statistic is incremental

sys.databases

- sys.databases has a column *is_auto_create_statistics_incremental* to indicate whether auto created stats will be incremental

sys.dm_db_stats_properties

- Accurately track the number of rows changed in a table

What is Cardinality Estimation?

Estimate of number of rows returned by each operation

- Based on statistics metadata
- Fixed estimates (assumptions) when no histograms are available

Query optimization is inherently unpredictable and very sensitive to cardinality estimations

Cardinality estimation influences the cost, which is an indicator of response time

Poor estimations can result in inefficient plans

Why Change Cardinality Estimation?

Business Case

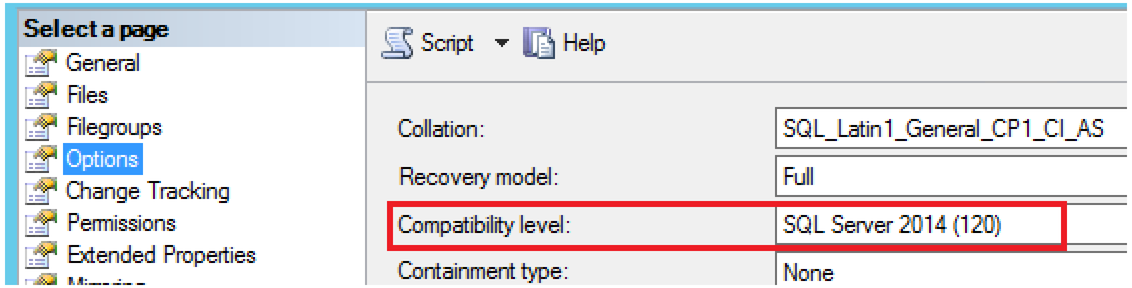
- CE engine was the same since 7.0
- Many changes under trace flag (Trace flag 4199 for Optimizer)
- Goal was to architecturally enhance the quality of cardinality estimation for a broad range of queries and workloads: OLTP, DW and DSS
- Create better plans for most cases, especially complex queries
- Make performance smoother and more predictable

Ramifications

- Can result in worse plans in edge cases
- Can be disabled, new CE is side-by-side

New Cardinality Estimation

New CE model



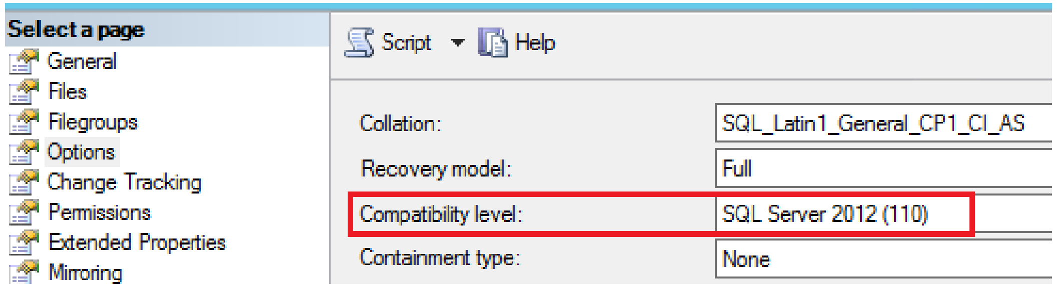
Collation: SQL_Latin1_General_CP1_CI_AS

Recovery model: Full

Compatibility level: SQL Server 2014 (120)

Containment type: None

Old CE model



Collation: SQL_Latin1_General_CP1_CI_AS

Recovery model: Full

Compatibility level: SQL Server 2012 (110)

Containment type: None

New Cardinality estimation – Trace Flags

New CE model

```
SELECT * FROM FactCurrencyRate
WHERE DateKey = 20101201
OPTION (QUERYTRACEON 2312)
```

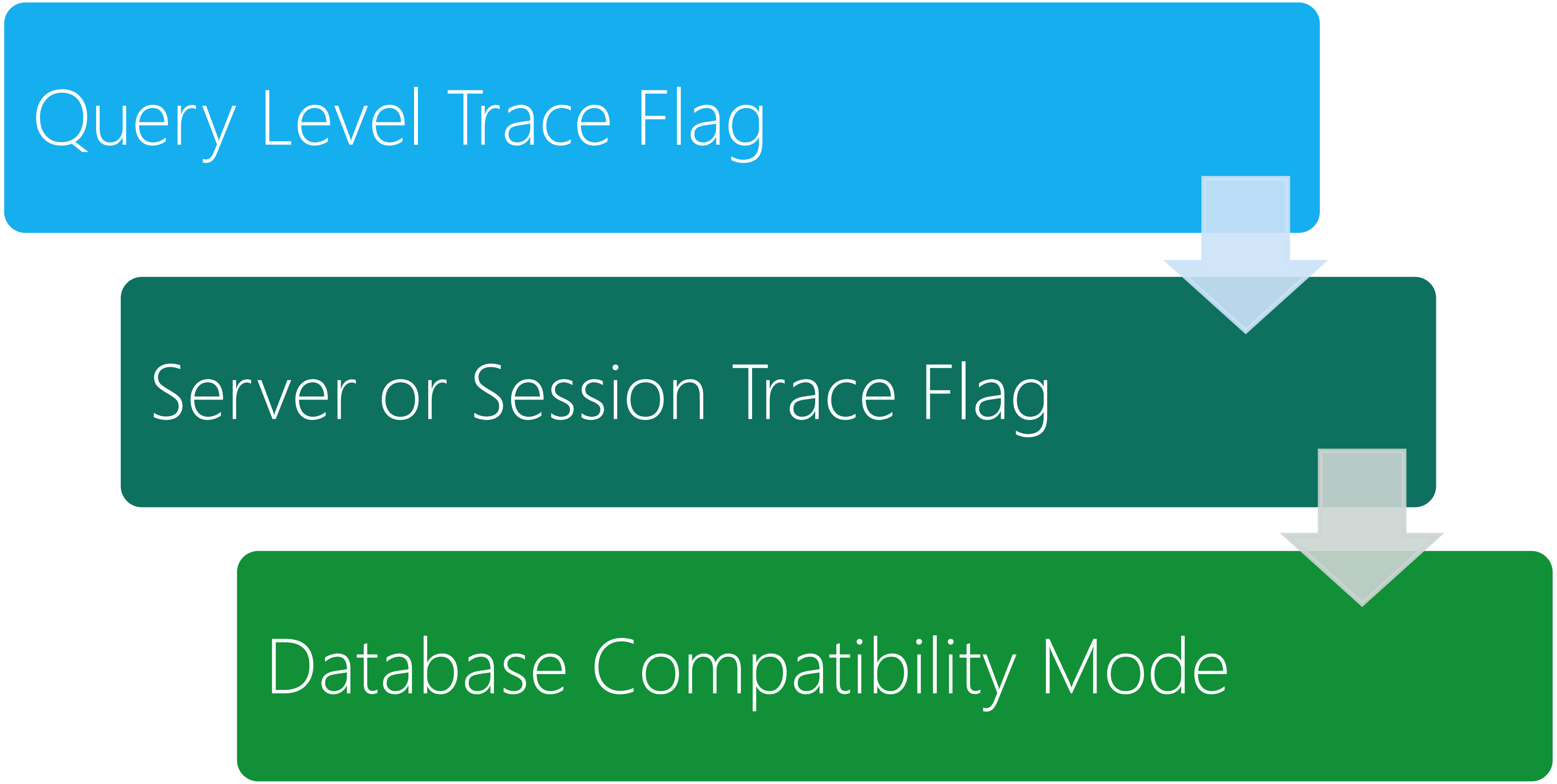
or Server Level TF 2312

Old CE model

```
SELECT * FROM FactCurrencyRate
WHERE DateKey = 20101201
OPTION (QUERYTRACEON 9481)
```

or Server Level TF 9481

Precedence



Query Level Trace Flag

Server or Session Trace Flag

Database Compatibility Mode

Ascending Key Problem

Example

- SELECT item, category, amount FROM dbo.Sales AS s WHERE Date = '2013-12-19'
- Stats last updated 18th December, 2012

Old CE

- Latest data inserted not in the histogram
- Cardinality estimation comes up as 1
- Trace flag 2389 can help (Ascending Keys and Auto Quick Corrected Statistics)

New CE

- Assumption is that the value exists
- Estimate cardinality using average frequency
- Trace flag 2389 no longer relevant

What’s New - Exponential Back Off

Change in calculating combined density with single-table predicates

1000 Rows. 200 “Microsoft” , 50 XRacers.

```
SELECT * FROM Cars
WHERE Manufacturer = ‘Microsoft’ AND Model =‘XRacer’
```

Old CE (Independence)	New CE (Exponential Backoff)
$p_0 \times p_1 \times p_2 \times p_3 \times \cdots$	$p_0 \times {p_1}^{1/2} \times {p_2}^{1/4} \times {p_3}^{1/8}$
0.05 * 0.2 * 1000 = 10	0.05 * sqrt(0.2) * 1000 = 22.36

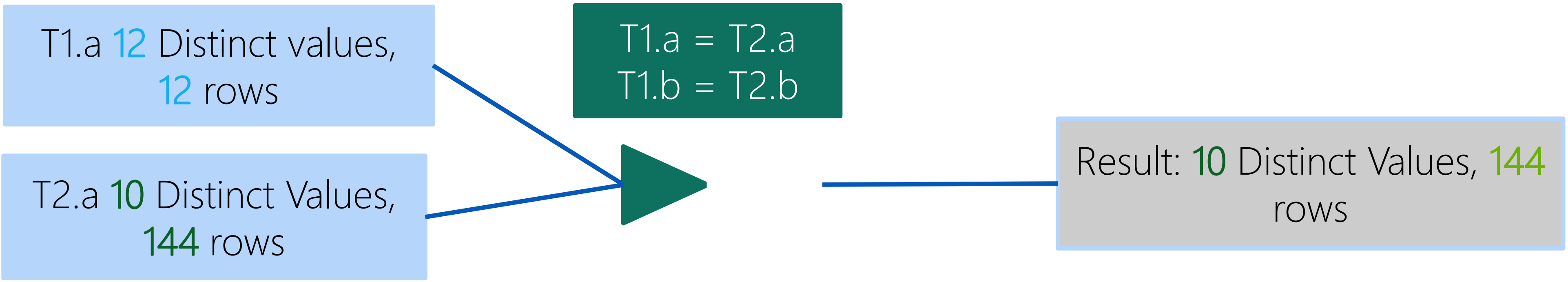
Density = 1/# Distinct Values

What’s New – Join Changes

```
SELECT t1.c, t1.d, t2.c
FROM t1 INNER JOIN t2
ON t1.a = t2.a AND t1.b = t2.b;
```

Equijoins with multiple predicates

- Take the lesser of the *Distinct Value Counts* between two inputs (10)
- Multiply by *Average Frequencies* from both inputs (10 * 1 * 14.4 = 144)



T1.a 12 Distinct values, 12 rows

T2.a 10 Distinct Values, 144 rows

T1.a = T2.a
T1.b = T2.b

Result: 10 Distinct Values, 144 rows

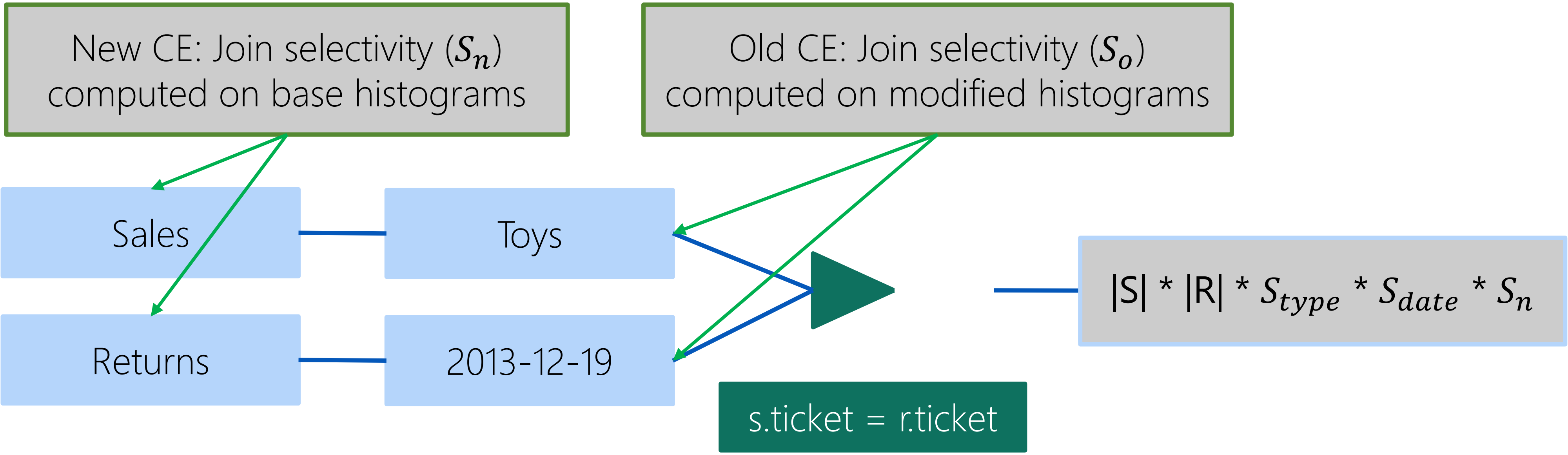
Joins use Base Containment in new CE v/s Simple containment in old CE

- Selectivity applied to Base tables
- Then selectivity of additional predicates applied

What’s New – Containment Assumption

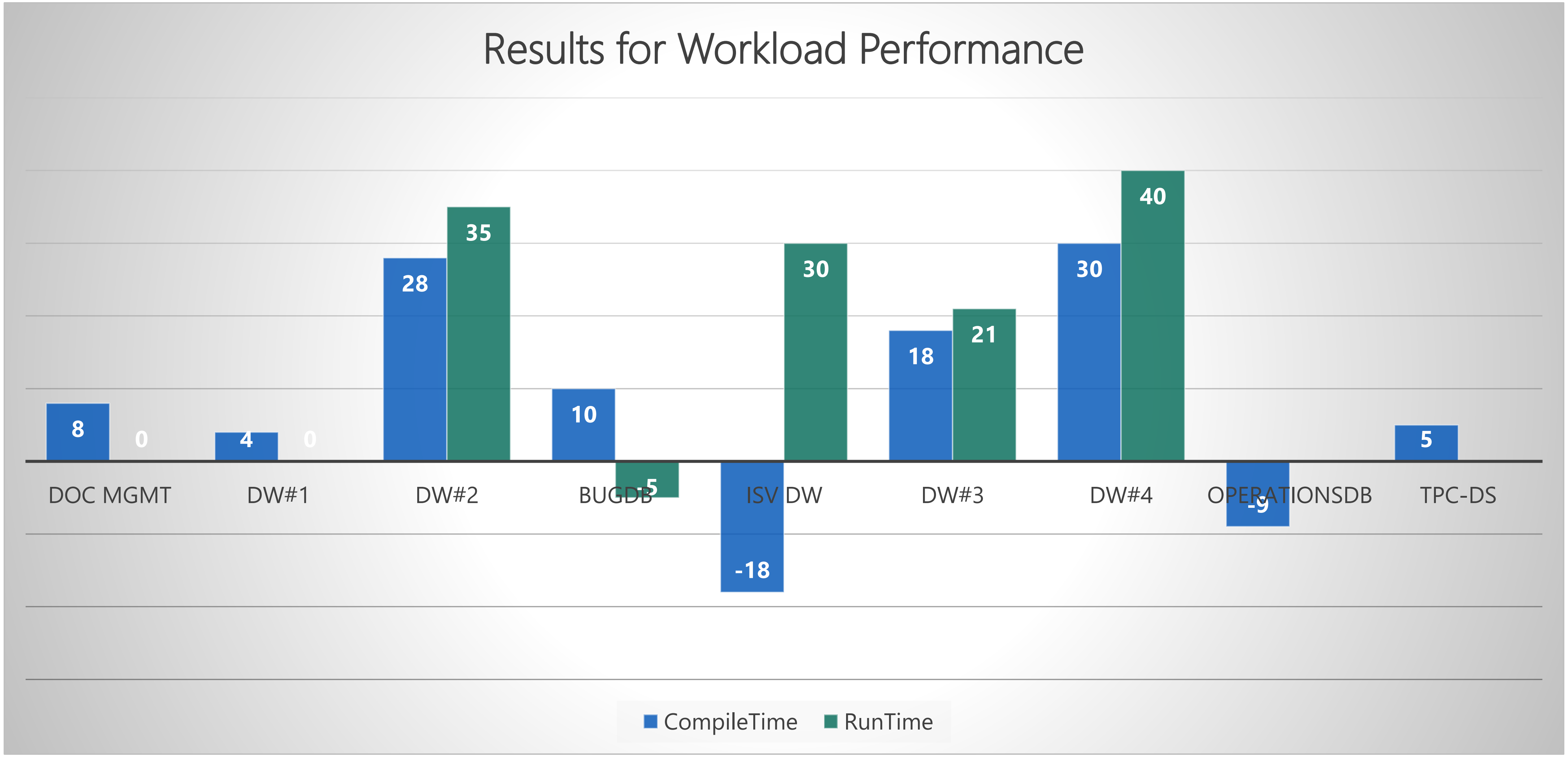
```
SELECT s.ticket, s.customer, r.store
FROM dbo.Sales AS s
INNER JOIN dbo.Returns AS r ON s.ticket = r.ticket
WHERE s.type = 'toy‘ AND r.date = '2013-12-19';
```

Equijoins with multiple predicates



New CE: Join selectivity (S_n) computed on base histograms
Old CE: Join selectivity (S_o) computed on modified histograms
Sales
Toys
Returns
2013-12-19
|S| * |R| * S_{type} * S_{date} * S_n
s.ticket = r.ticket

Impact of using New CE



When to use New CE?

Customers with query predictability issues

Data warehouse workloads can hugely benefit per testing

Test your workload with the new CE, there are edge cases!

Supportability features to help analyze Query Plans and Optimizer choices

- Extended Event *query_optimizer_estimate_cardinality*
- Extended Event *query_optimizer_force_both_cardinality_estimation_behaviors*

Diagnostics

Extended Events

- query_optimizer_estimate_cardinality
- query_optimizer_force_both_cardinality_estimation_behaviors

Interpreting Extended Event

- Stats collection
- Input_relation
- CalculatorName

XML Query Plans

- StatsID