Columnstore and B+ tree – Are Hybrid Physical Designs Important?

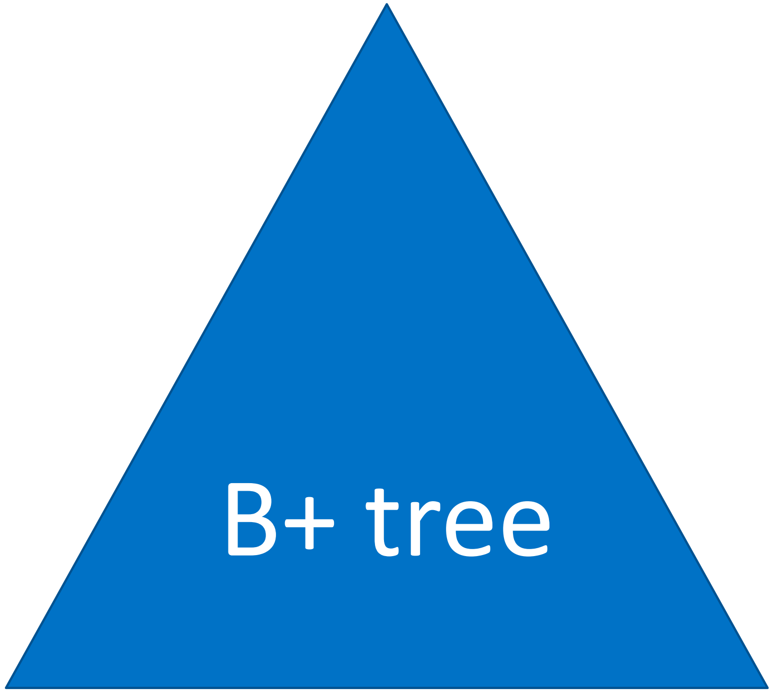
Adam Dziedzic, Jingjing Wang, Sudipto Das,

Bolin Ding, Vivek R. Narasayya, Manoj Syamala

Microsoft THE UNIVERSITY OF CHICAGO UNIVERSITY of WASHINGTON

Physical designs for diverse workloads

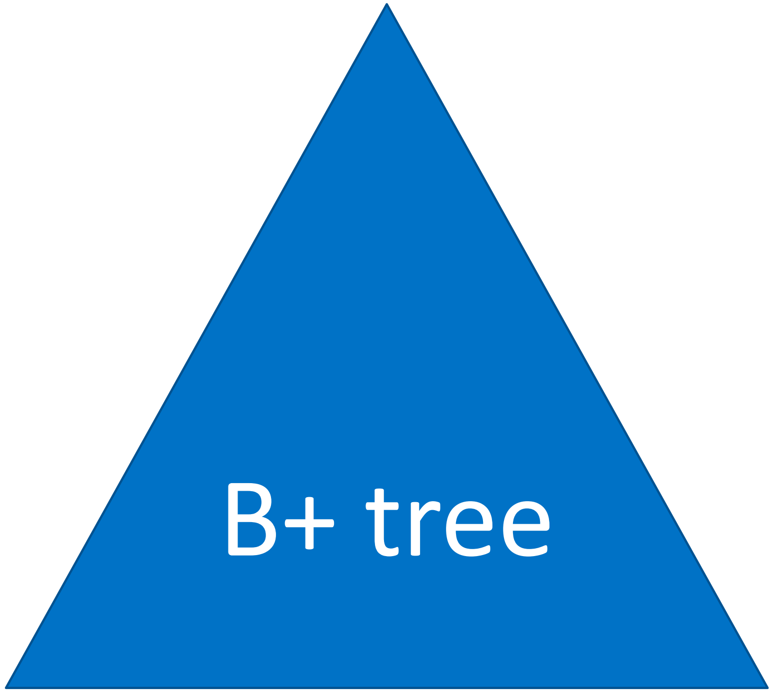
OLTP

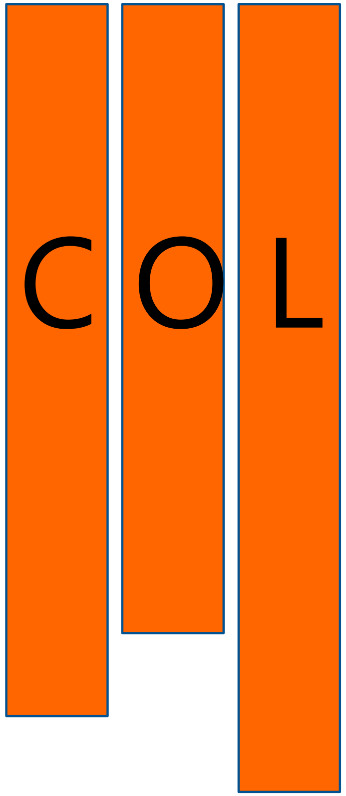


Physical designs for diverse workloads

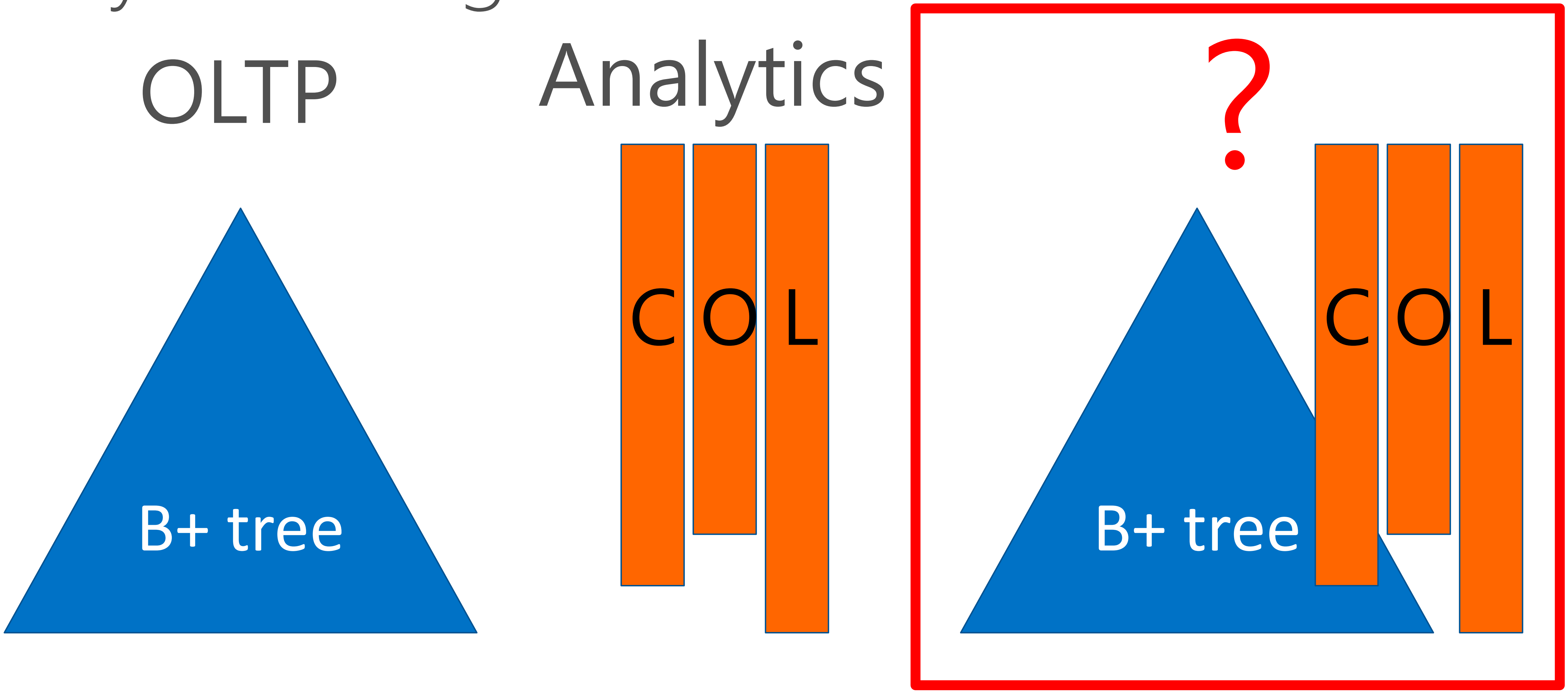
OLTP

Analytics



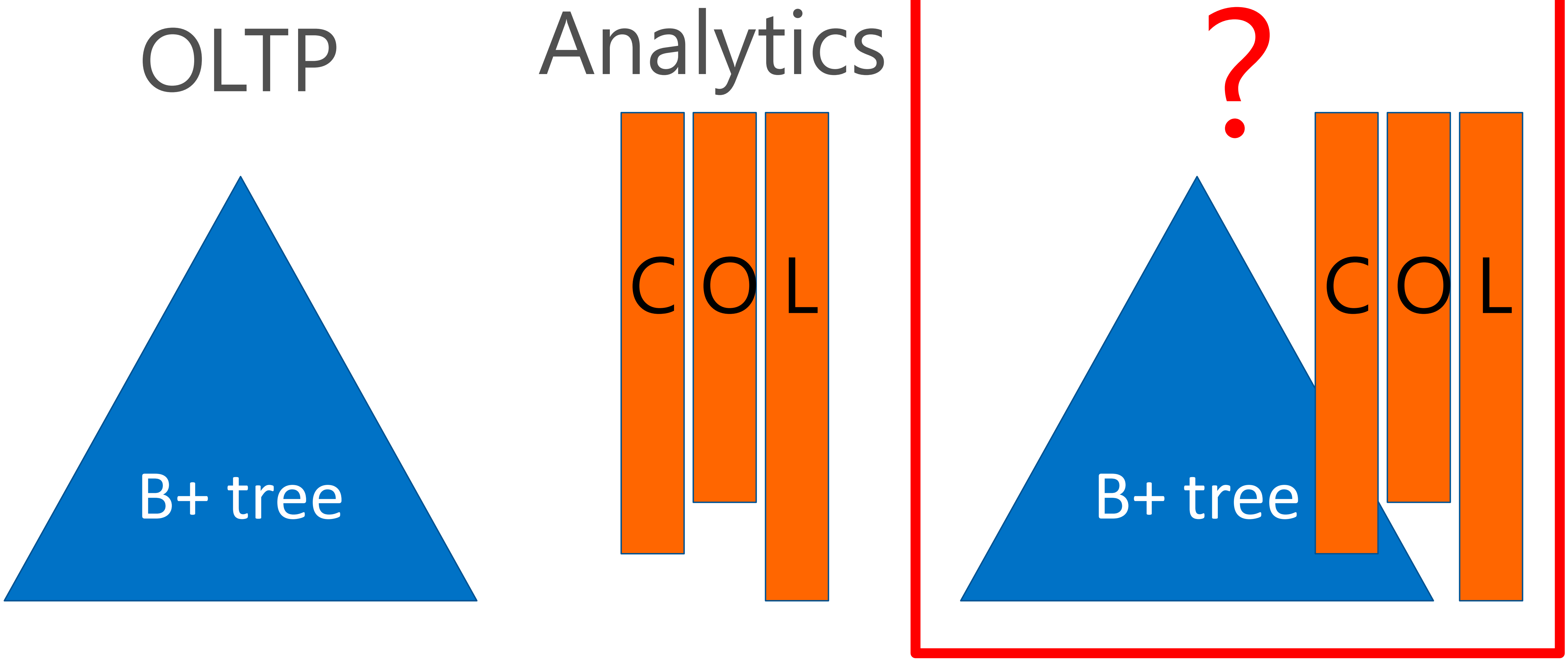


Physical designs for diverse workloads



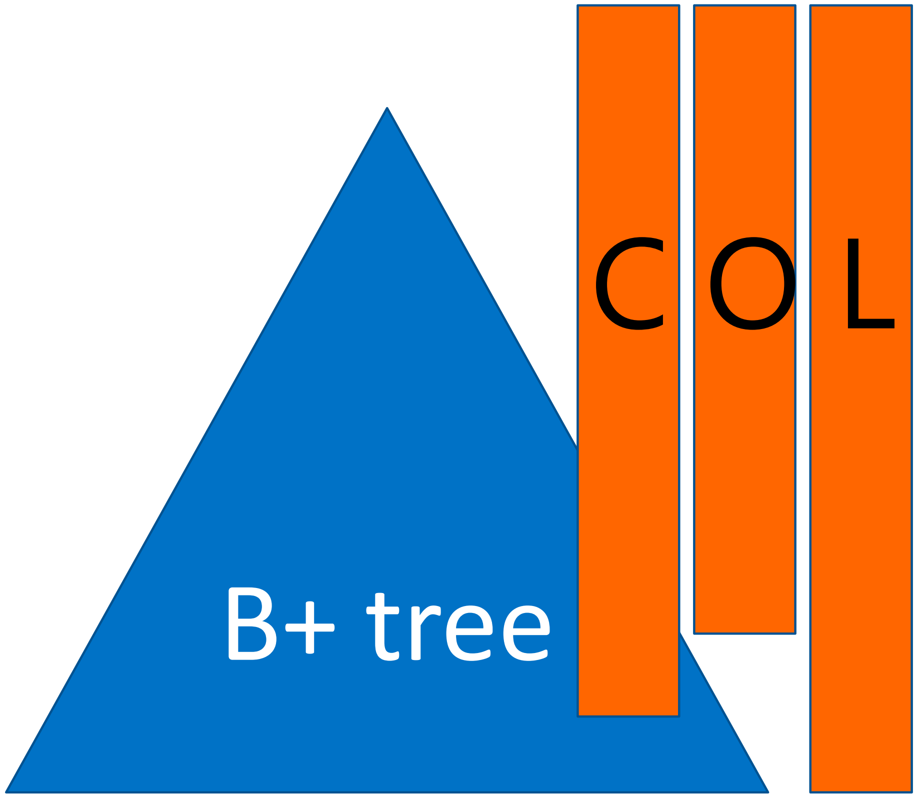
B+ tree & Columnstore on same table = Hybrid design

Physical designs for diverse workloads

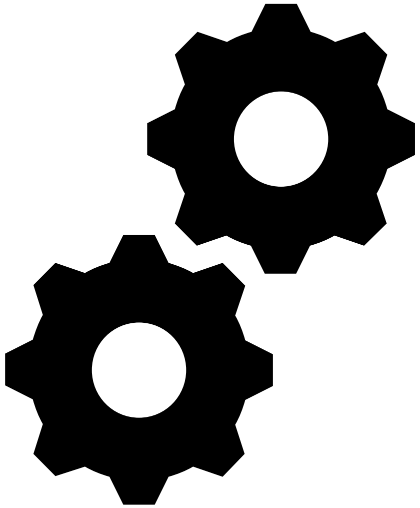


Are Hybrid Designs important and which workloads can benefit?

Hybrid design = B+ tree & Columnstore



EVALUATE



AUTO
RECOMMEND

Hybrid design = B+ tree & Columnstore

1. Micro-benchmarks

2. Auto Recommend Hybrid Designs

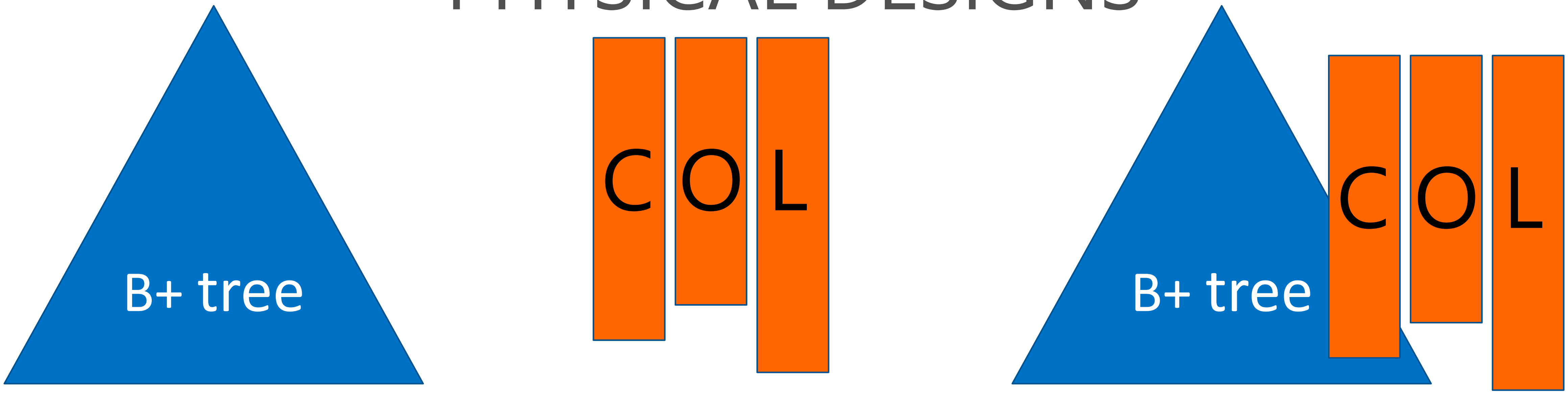
3. End-to-end evaluation

Micro-benchmarking HYBRID DESIGNS

QUERIES

- Selectivity
- Sort order
- Updates
- Mix: Scans & Updates
- Concurrency

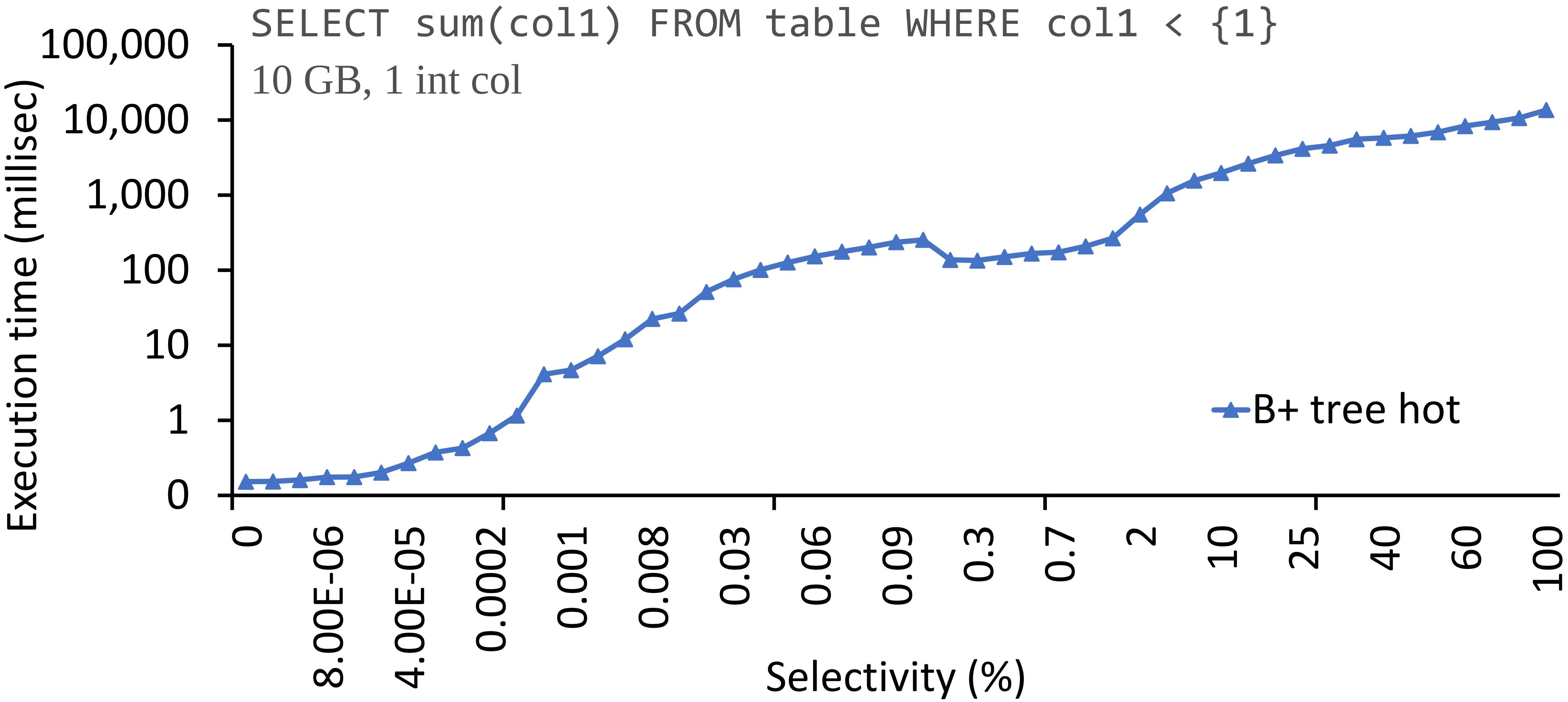
PHYSICAL DESIGNS



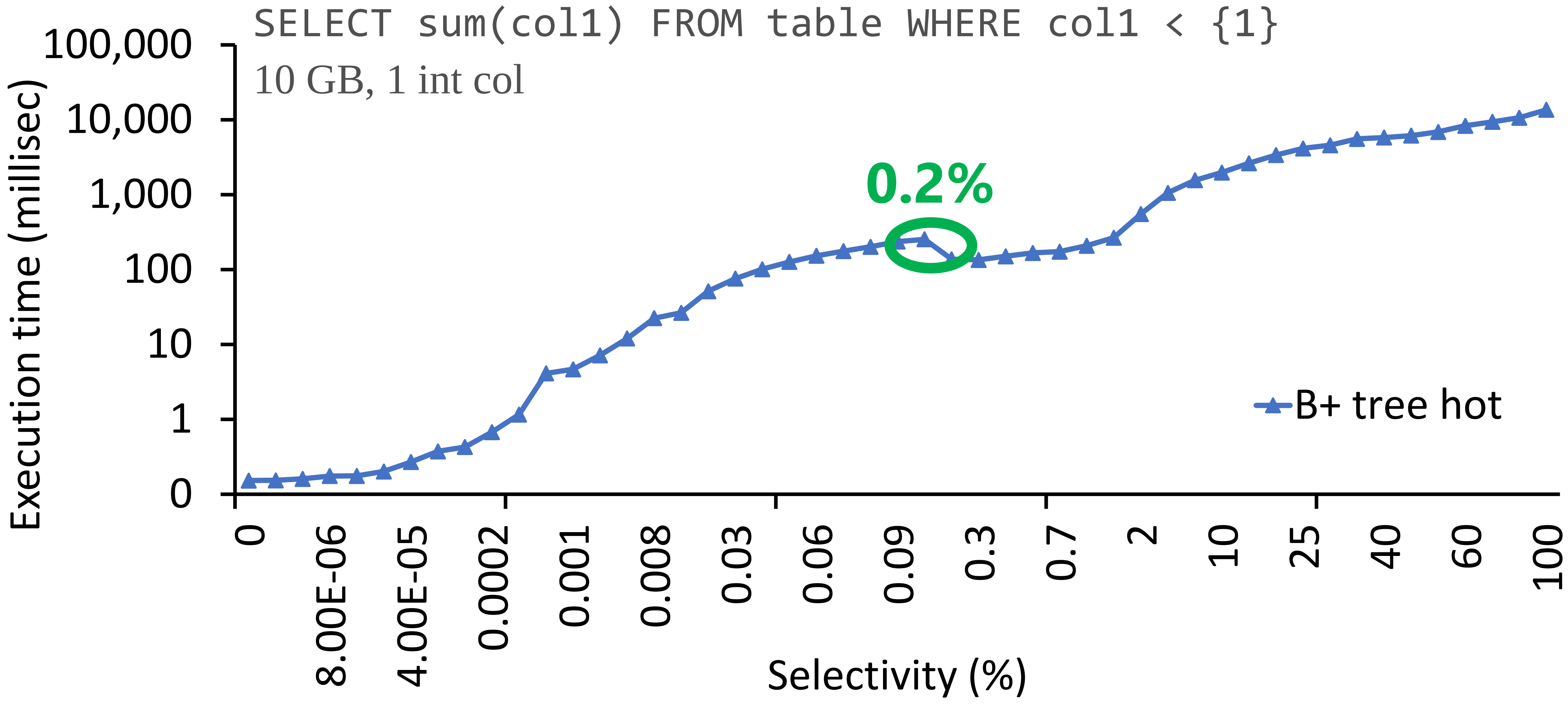
Experimental setup for micro-benchmarks

- Synthesized queries and data sets + TPC-H data
- Pre-release version of SQL Server 2017
- Server: on-premise
  - **Dual socket** Intel® Xeon® CPU E5−2660v2, **10 cores per socket**, 2 threads per core, clocked at 2.20 GHz, 64 KB L1 cache per core, 256 KB L2 cache per core, and 25 MB L3 cache shared
  - **384 GB main memory**
  - **18 TB HDD in RAID-0** configuration (throughput of ~1 GB/sec for reads and ~400 MB/sec for writes)

B+ tree Range Scans vs. Col Full Scans

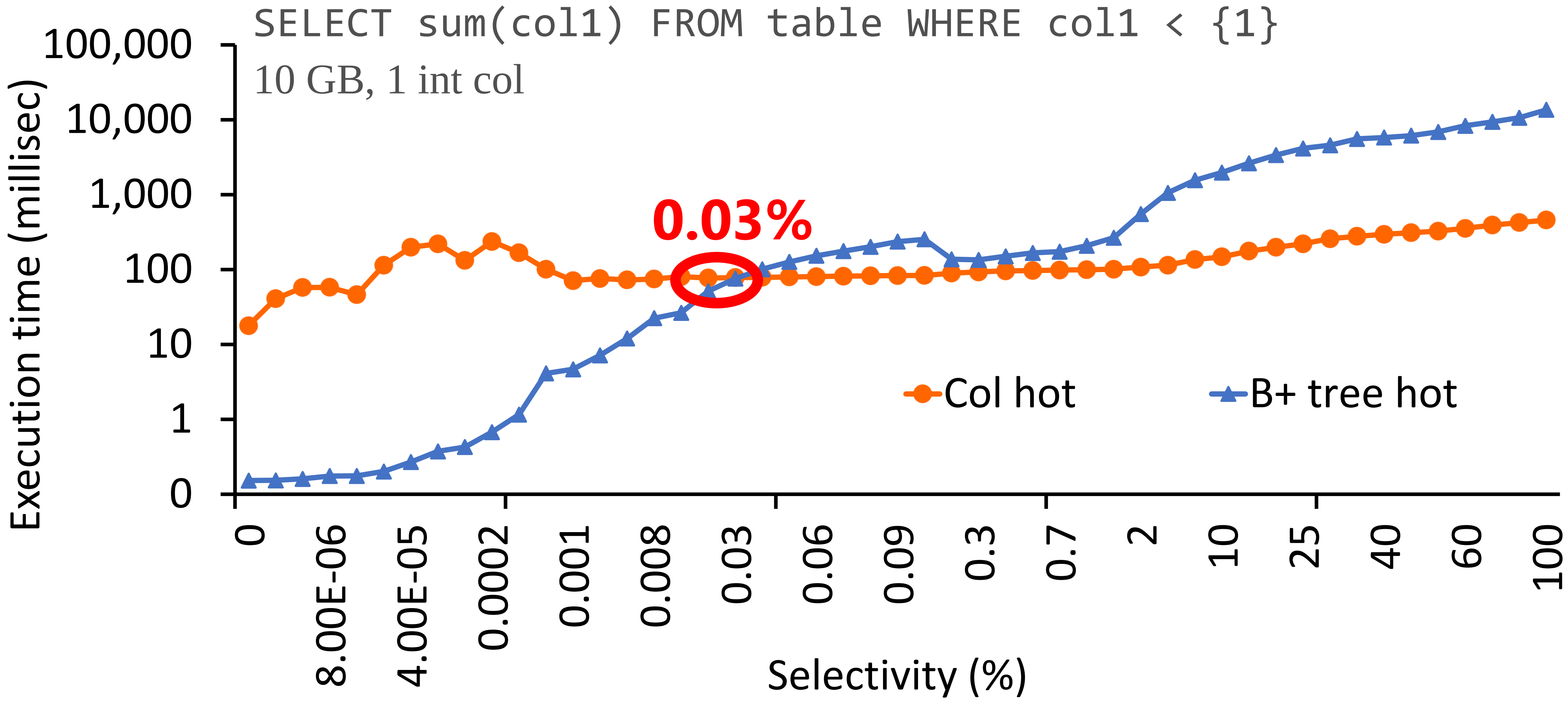


B+ tree Range Scans vs. Col Full Scans



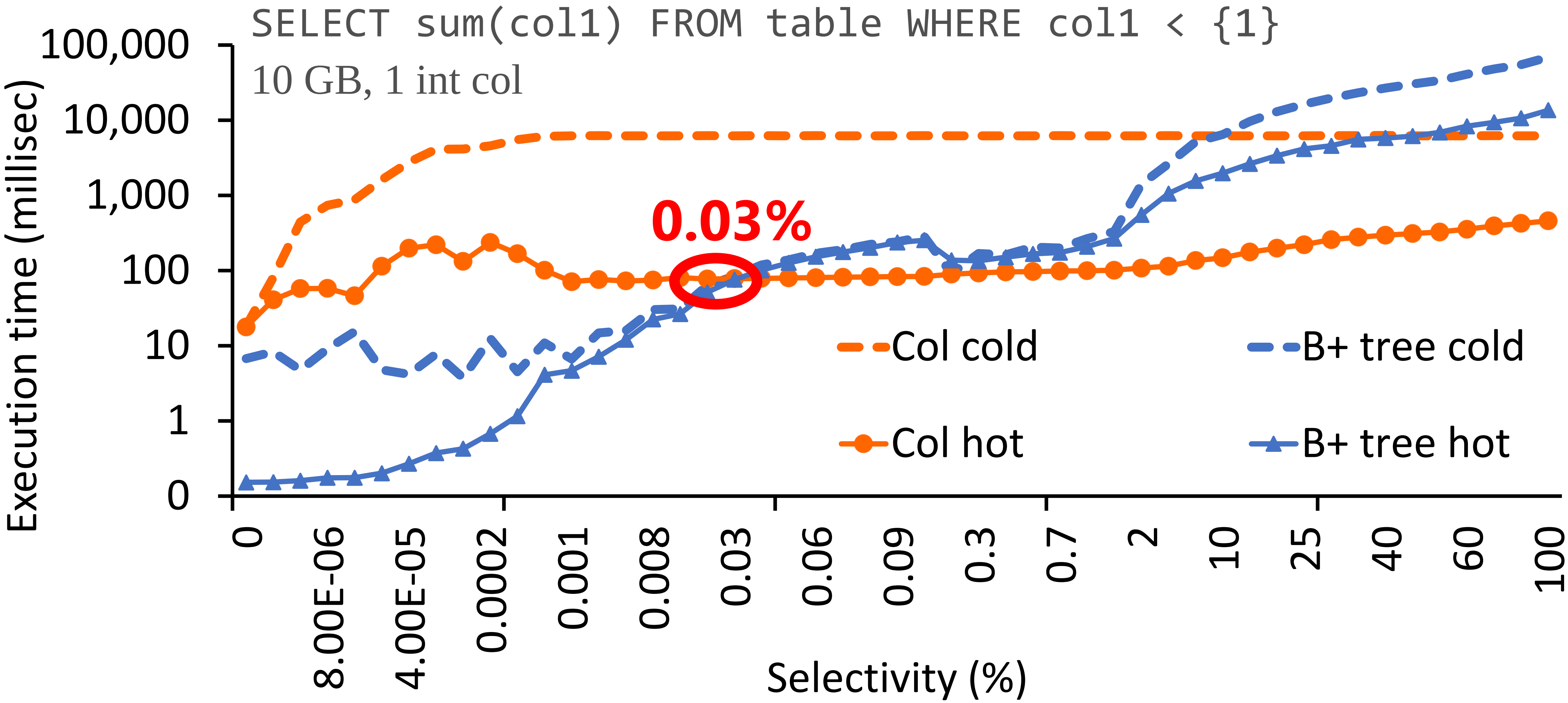
Skip data effectively & run single-threaded

B+ tree Range scans vs. Col Full scans



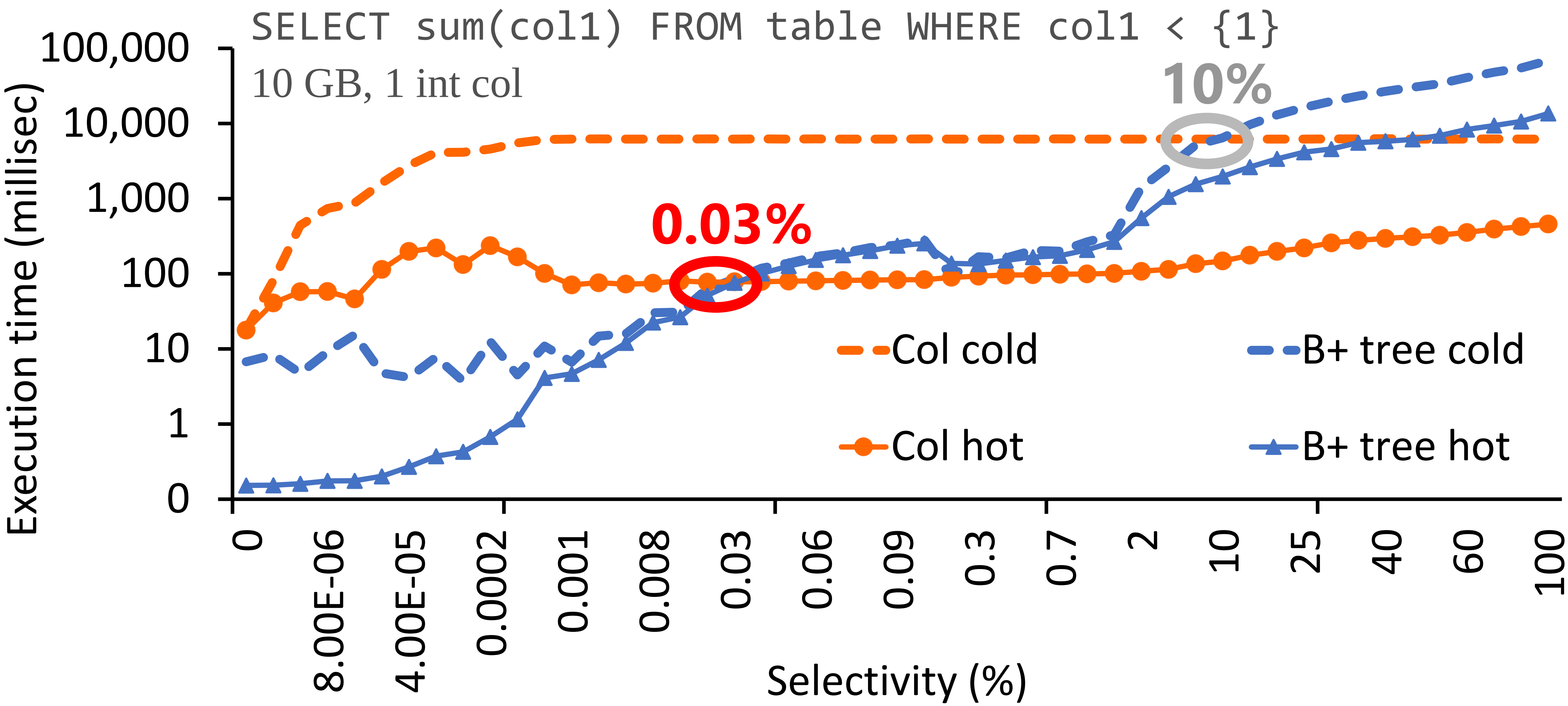
Superior performance of Columnstore scans

B+ tree Range Scans vs. Col Full Scans



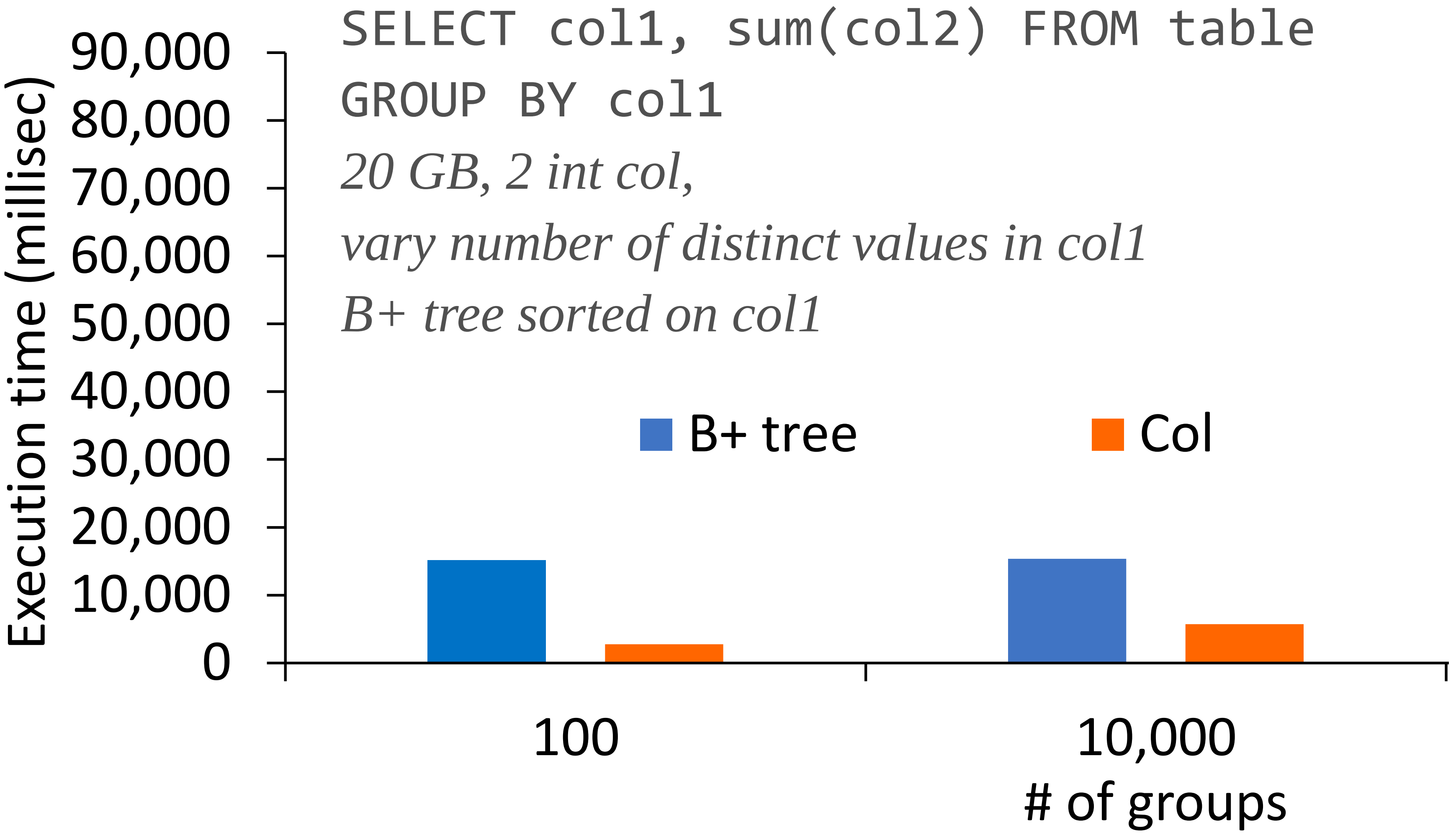
B+ tree helps for low selectivity & slower storage

B+ tree Range Scans vs. Col Full Scans

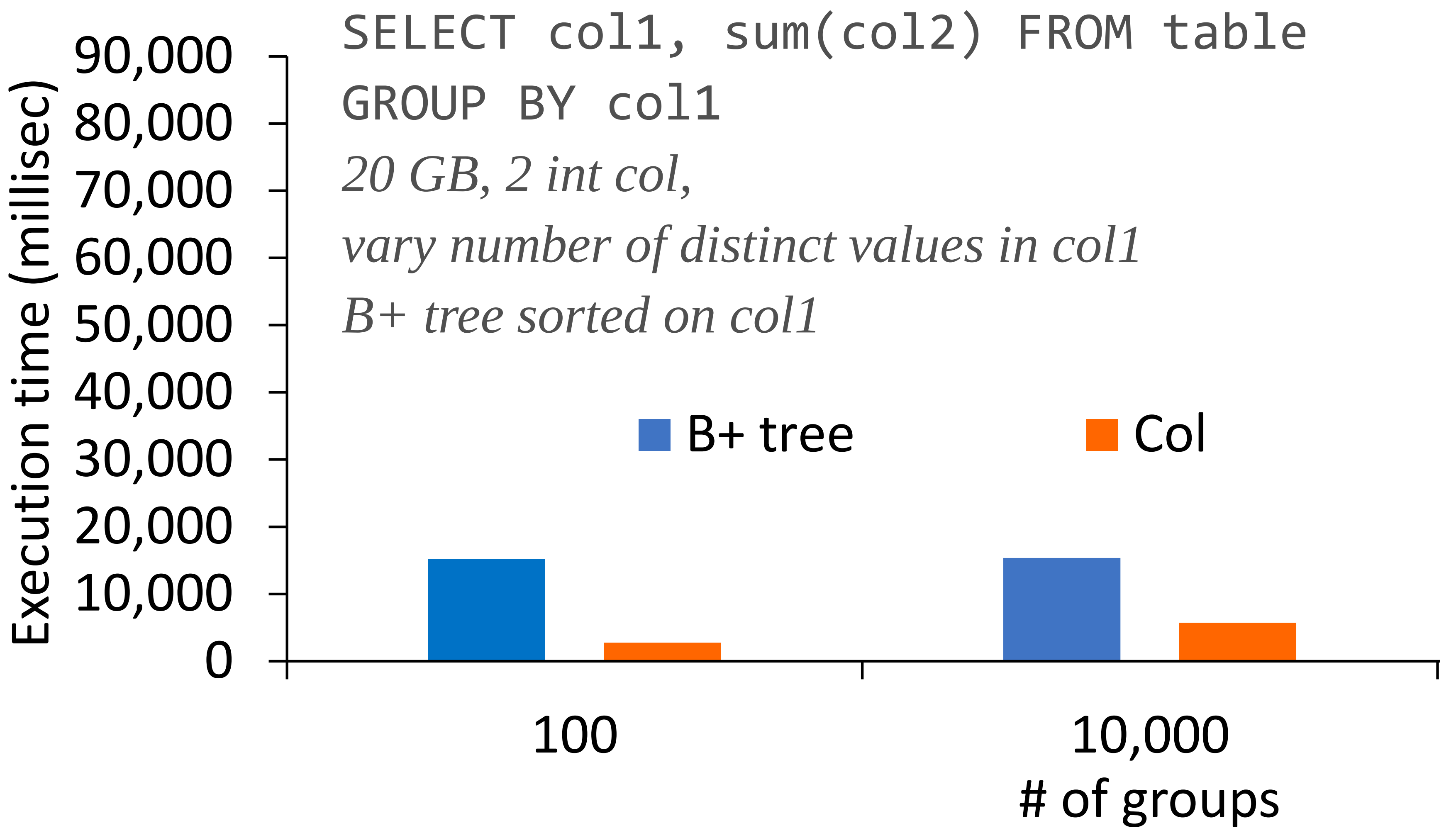


B+ tree helps for low selectivity & slower storage

Sorted B+ tree vs. Unsorted Col

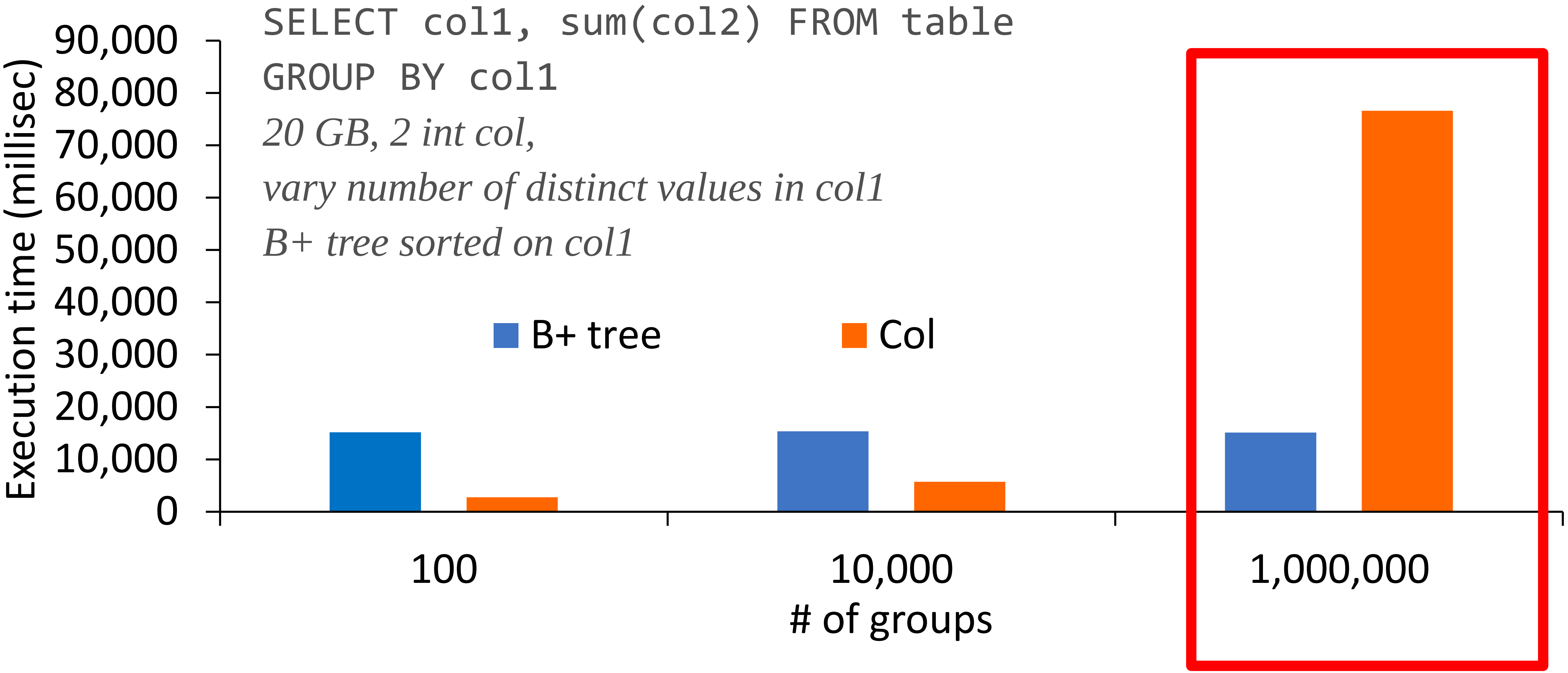


Sorted B+ tree vs. Unsorted Col



Scanning & hashing Col faster than reading sorted B+ tree

Sorted B+ tree vs. Unsorted Col

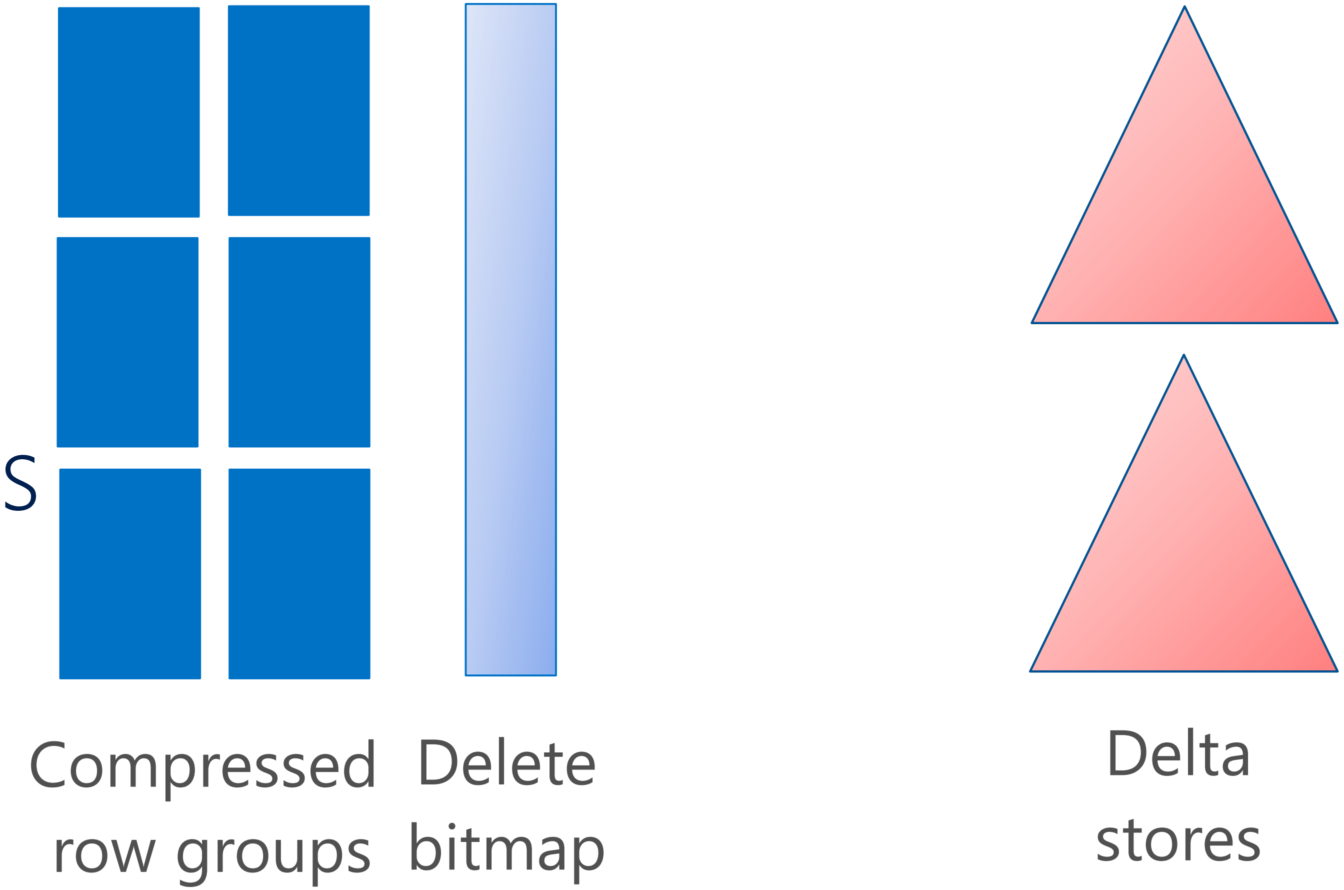


Sort order of B+ tree beneficial for scarce query memory

Two types of Columnstores in SQL Server

Primary Columnstores

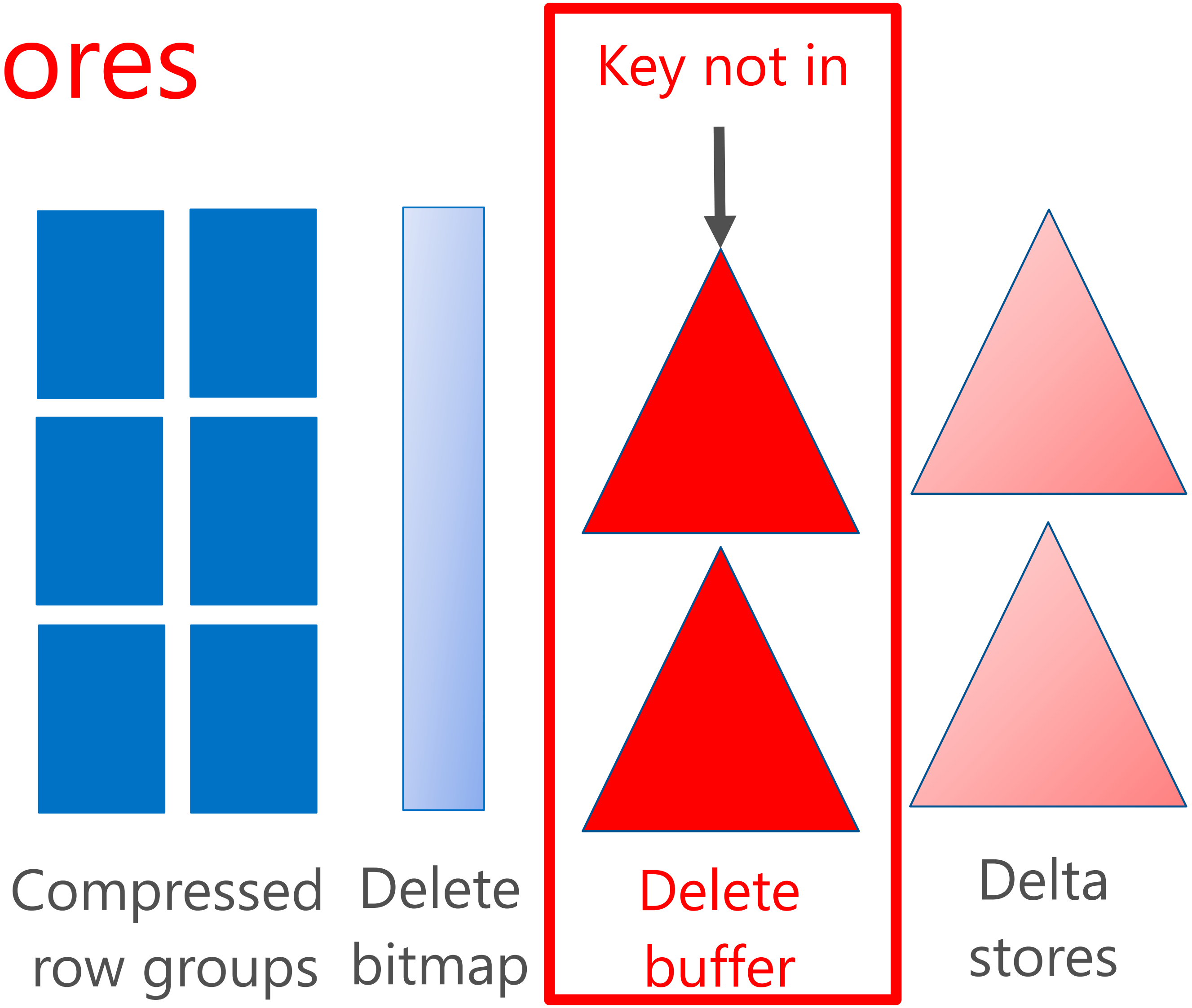
- Base table storage
- Optimize **large scans**
- **Complicated updates** on compressed columns
- **Updates** via the delete bitmap **are expensive**



Two types of Columnstores in SQL Server

Secondary Columnstores

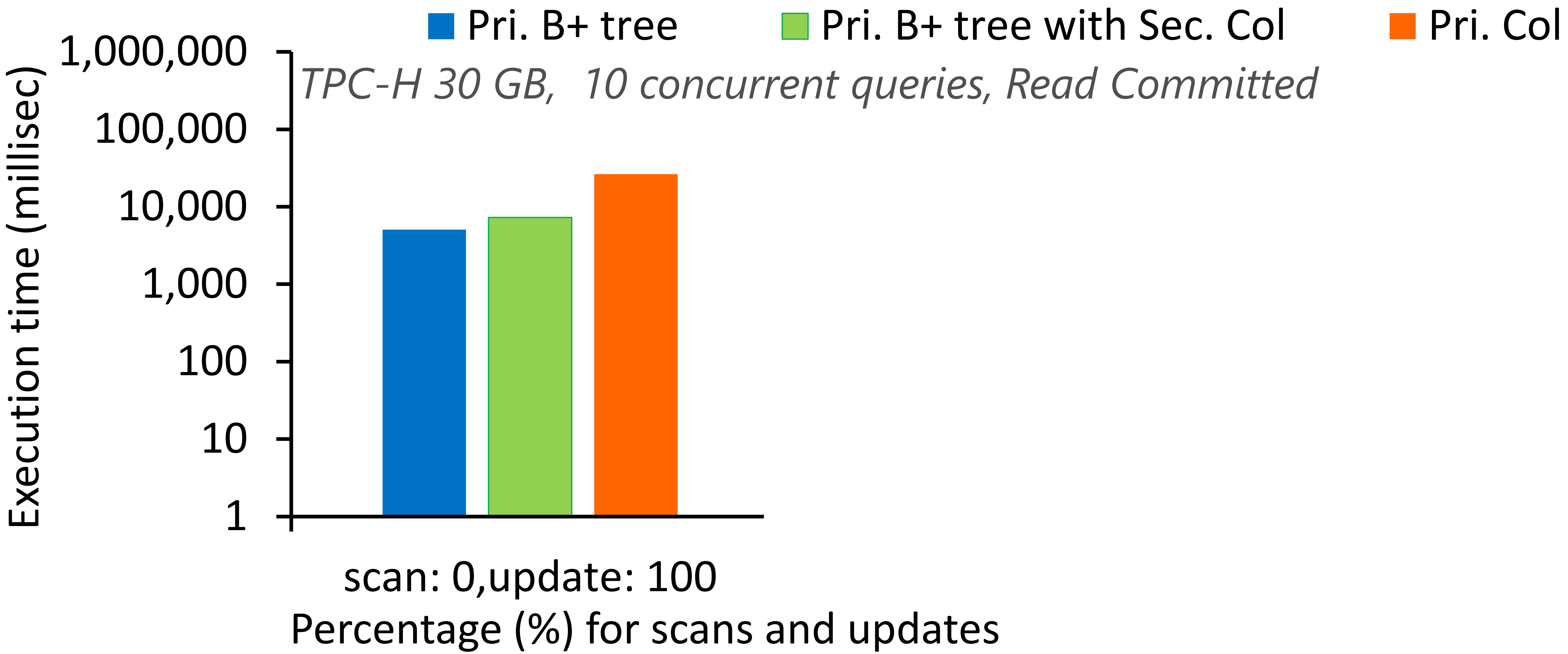
- Designed to balance performance of **long scans and updates**
- **Delete buffer** stores the logical row being deleted
- Requires anti-semi join



Mixed workload: large scans & small updates

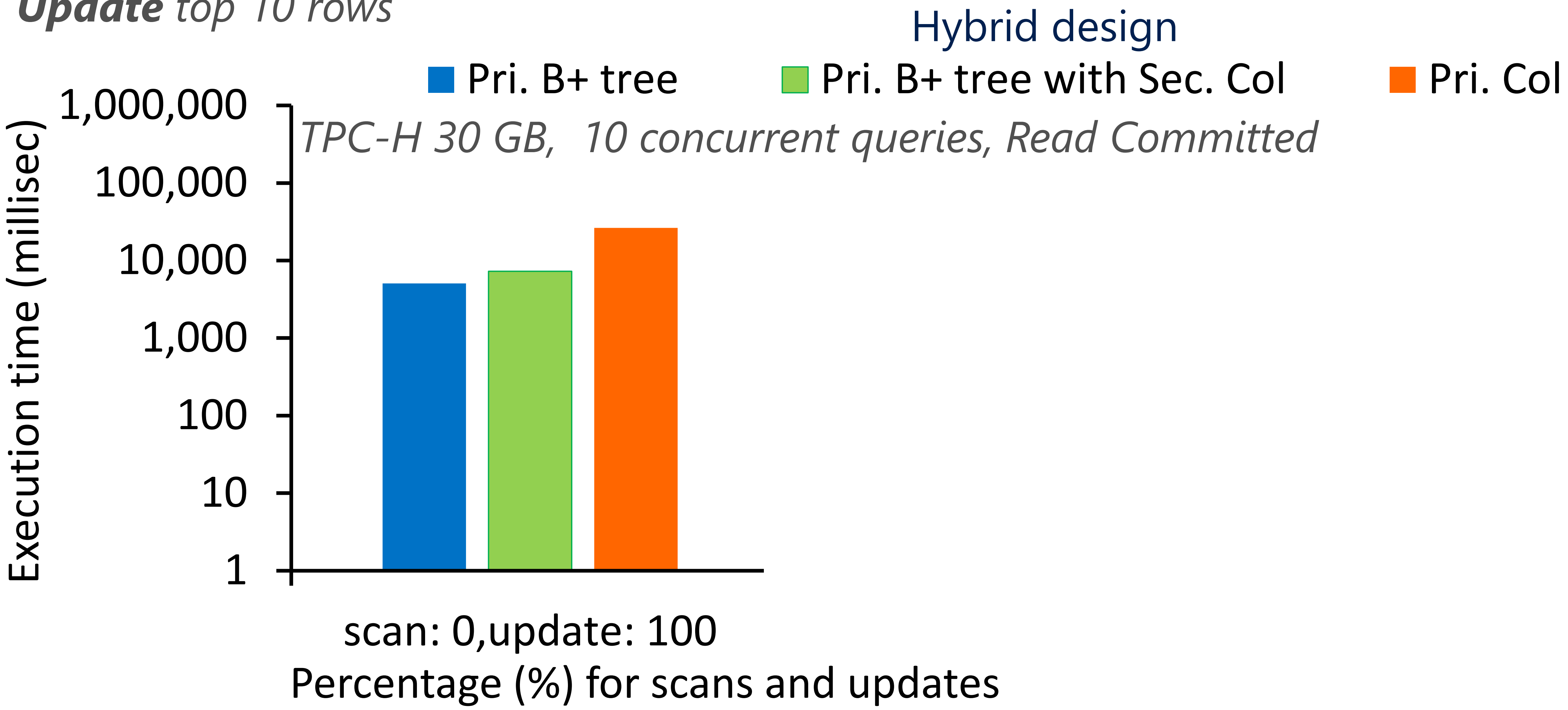
***Update** top 10 rows*

Hybrid design



Mixed workload: large scans & small updates

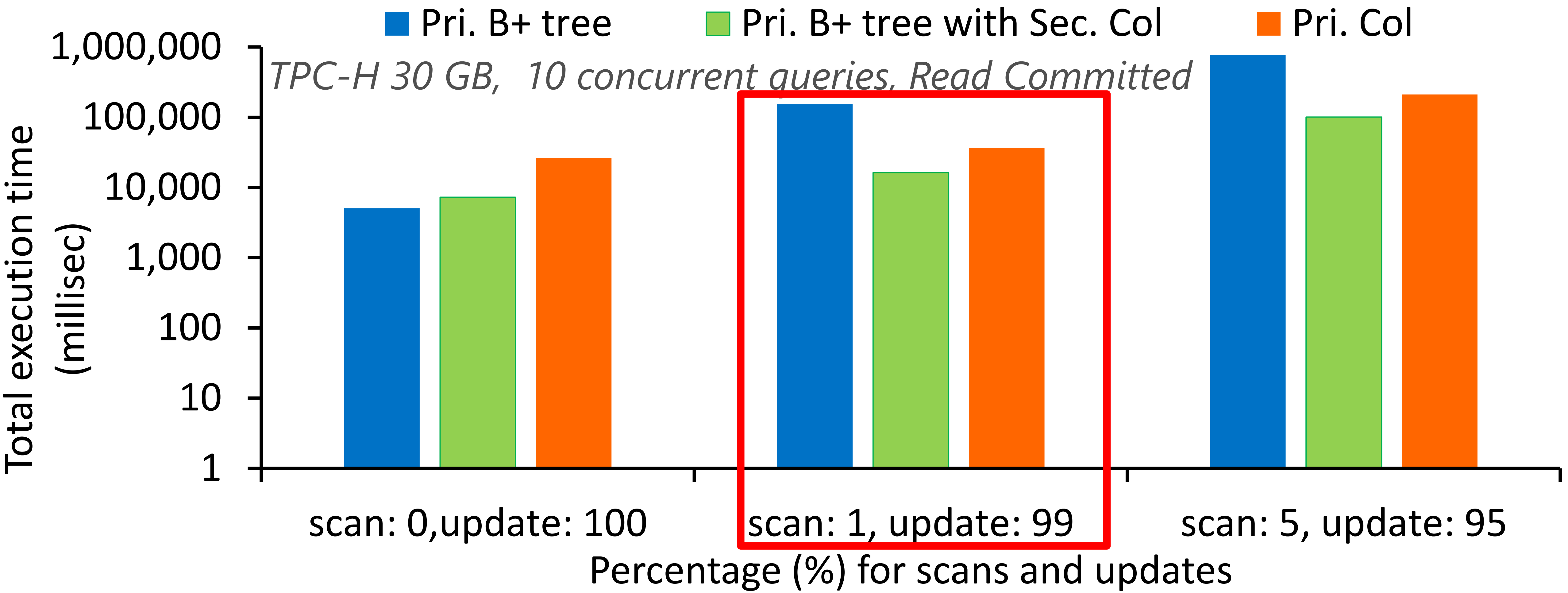
Update *top 10 rows*



B+ trees cheaper than Columnstores for pure updates

Mixed workload: large scans & small updates

Update *top 10 rows,* ***Select*** *sum of quantity & price for a single shipdate from lineitem table*

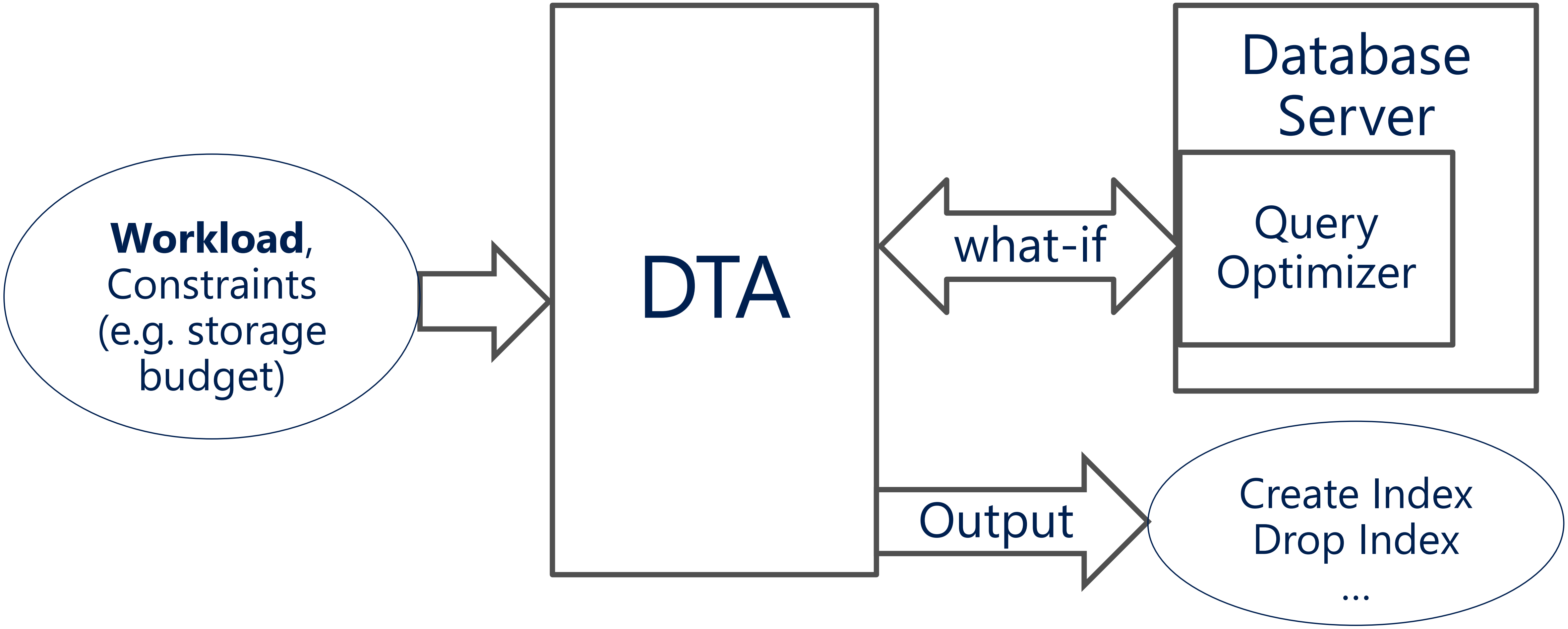


Secondary Columnstore: balance small updates & large scans

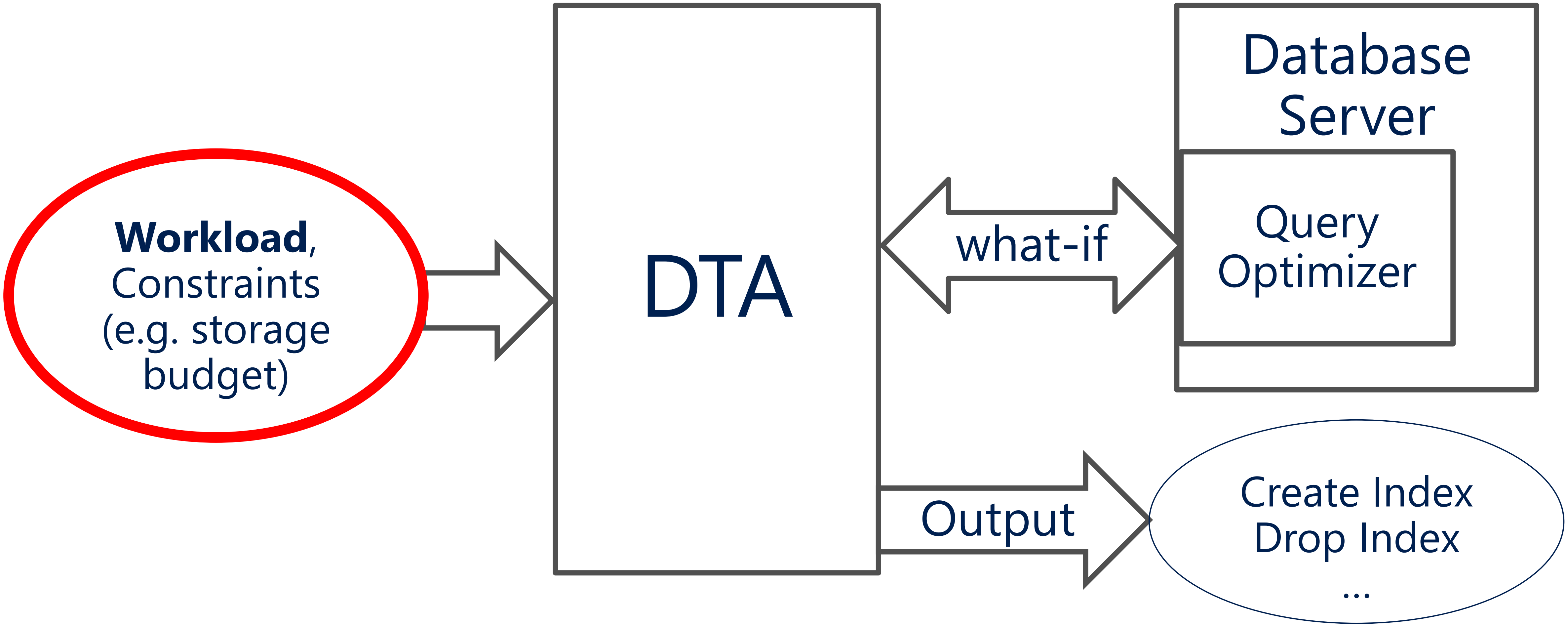
Hybrid design = B+ tree & Columnstore

1. Micro-benchmarks
2. Auto Recommend Hybrid Designs
3. End-to-end evaluation

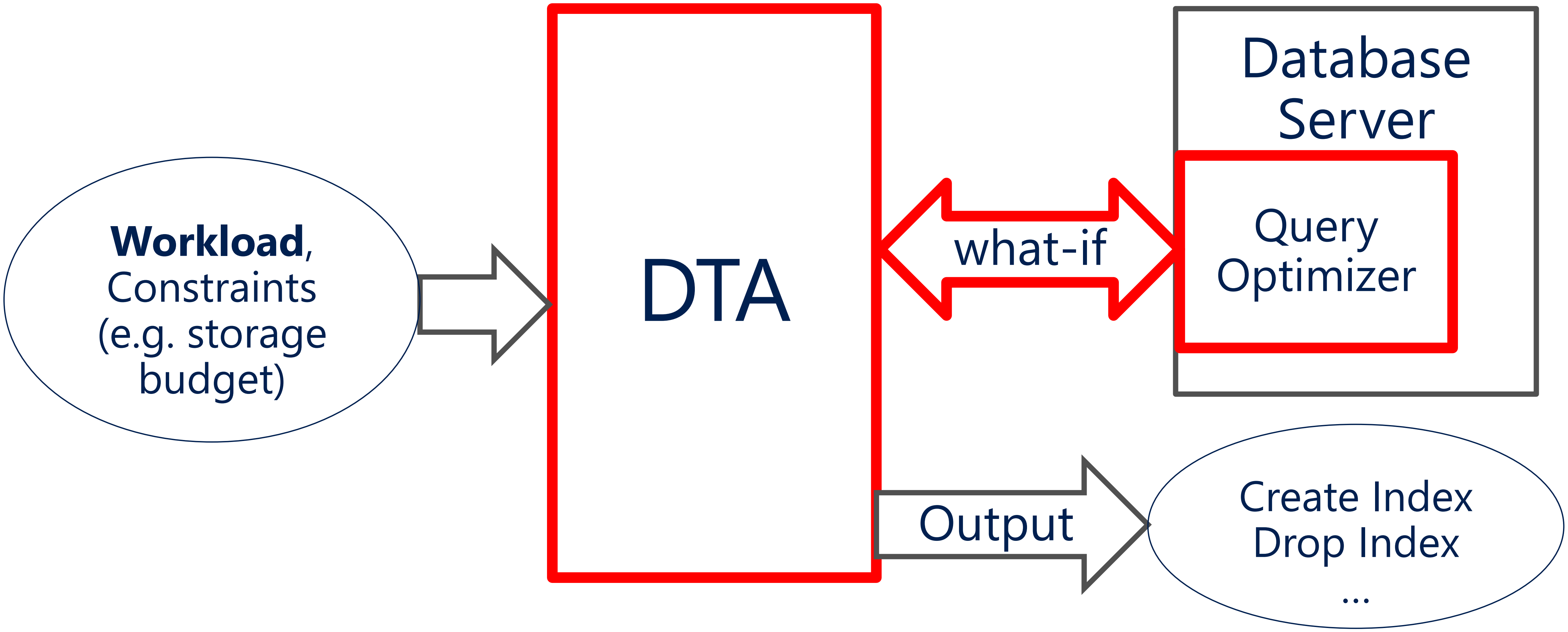
Database Engine Tuning Advisor (DTA)



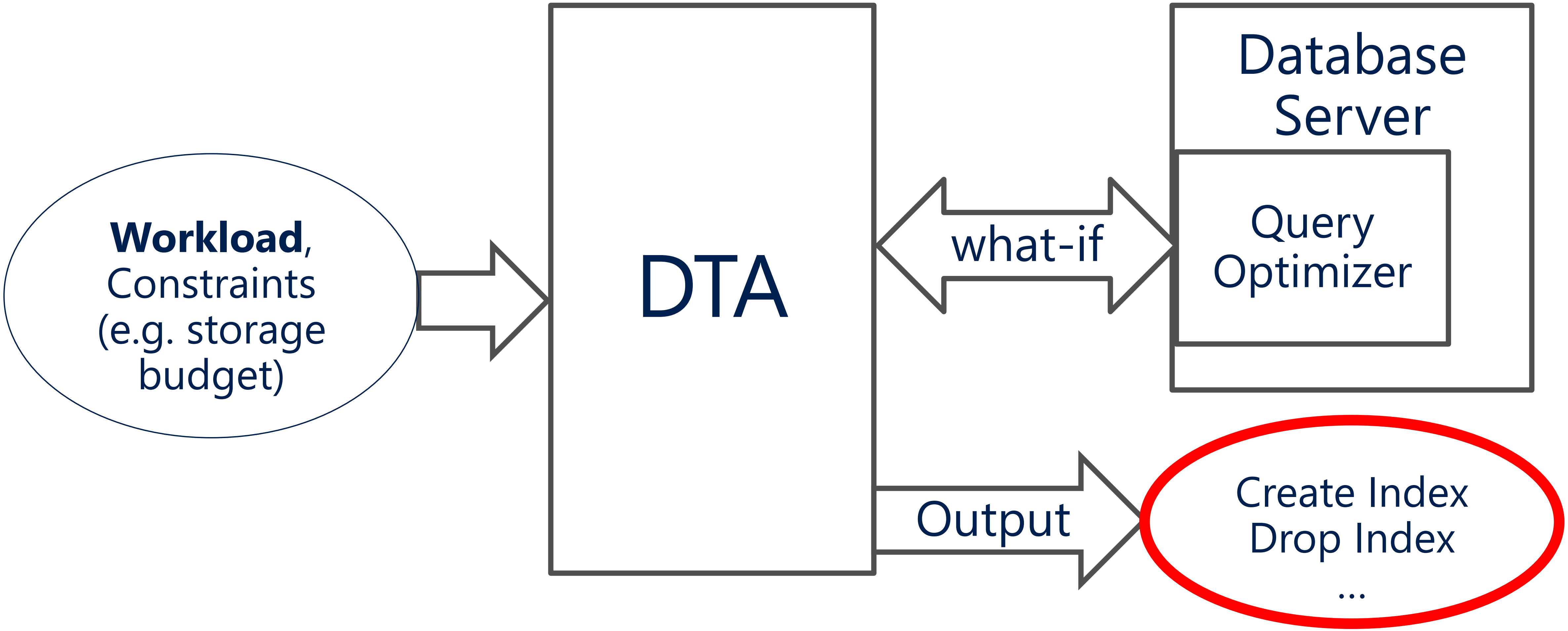
Database Engine Tuning Advisor (DTA)



Database Engine Tuning Advisor (DTA)



Database Engine Tuning Advisor (DTA)



Extensions to SQL Server Query Optimizer

Augment the "what-if" API for:

1. Costing queries on Hypothetical Columnstores

Extensions to SQL Server Query Optimizer

Augment the "what-if" API for:

1. Costing queries on Hypothetical Columnstores

2. Per-column size estimation
   - stay within storage budget
   - per-column access cost
   - hard problem

DTA extensions for hybrid designs

1. Optimal designs searched over:
   combined space of
   Columnstores & B+ trees

DTA extensions for Hybrid Designs

1. Optimal designs searched over:

   combined space of

   Columnstores & B+ trees

2. Released as part of

   SQL Server 2017 CTP

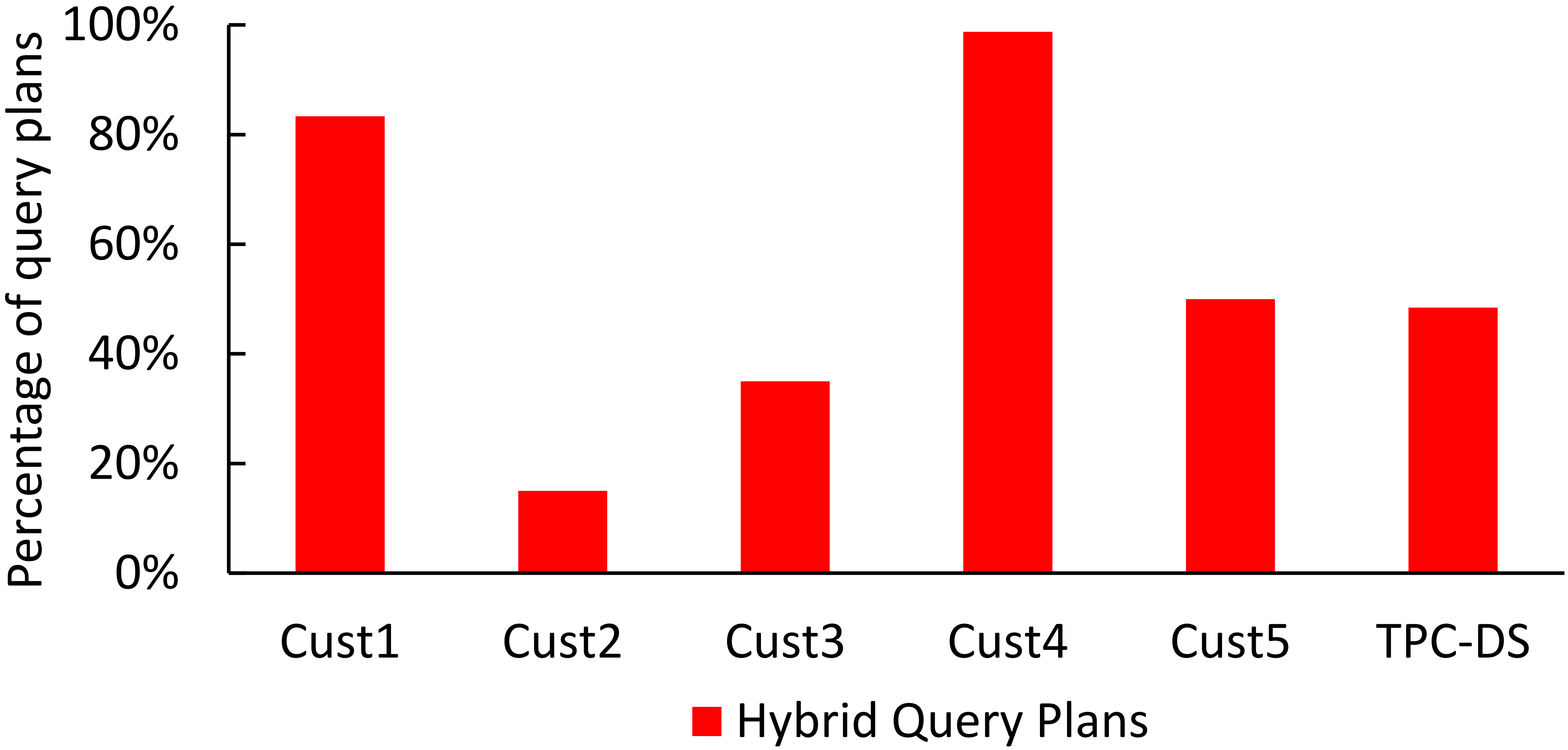
Hybrid design = B+ tree & Columnstore

1. Micro-benchmarks
2. Auto Recommend Hybrid Designs
3. End-to-end evaluation

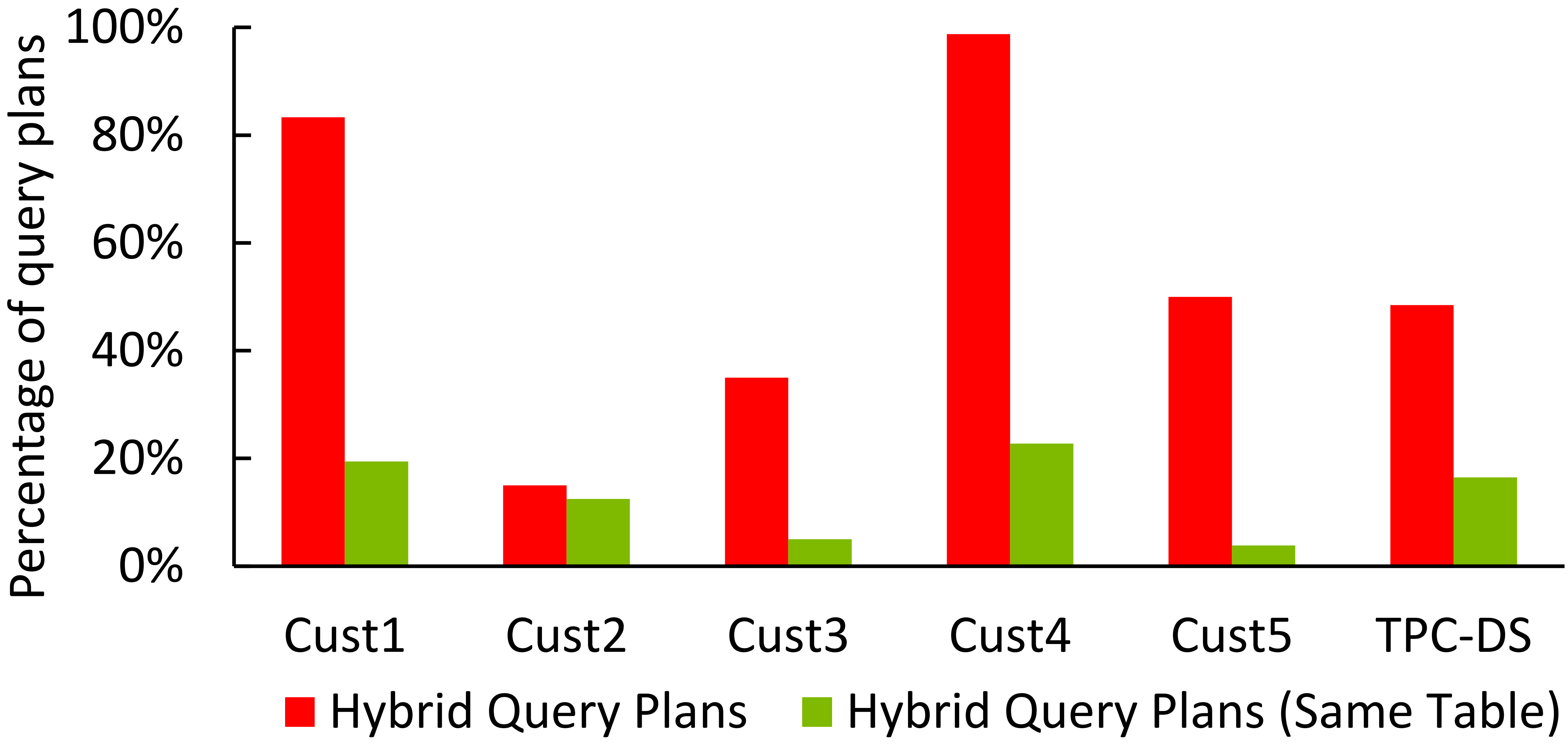
End-to-end evaluation of hybrid designs

- 5 customer workloads, TPC-DS, and CH benchmark
- Homogenous design
  - B+ tree only (for each query, tune with DTA's B+ tree index recommendations)
  - Columnstore only (secondary Columnstores on all tables for all possible columns)
- Hybrid design
  - B+ tree and Columnstore indexes recommended by **DTA** (Database Engine Tuning Advisor)
- Speedup = (time on Homogenous) / (time on Hybrid)
  - Metrics: median execution time & CPU time (per query)
  - Warm execution & working set in memory
- The same hardware & software as for micro-benchmarks

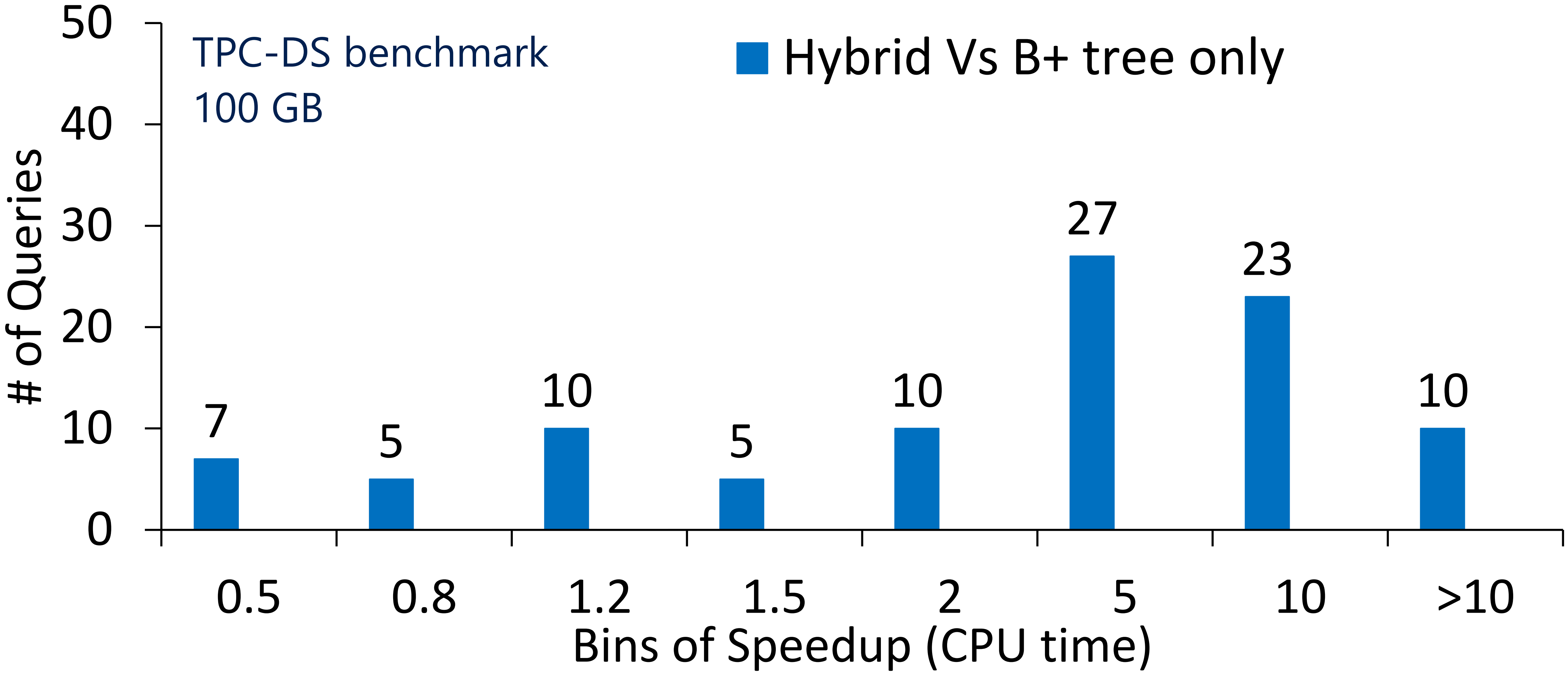
Hybrid Query Plans are common



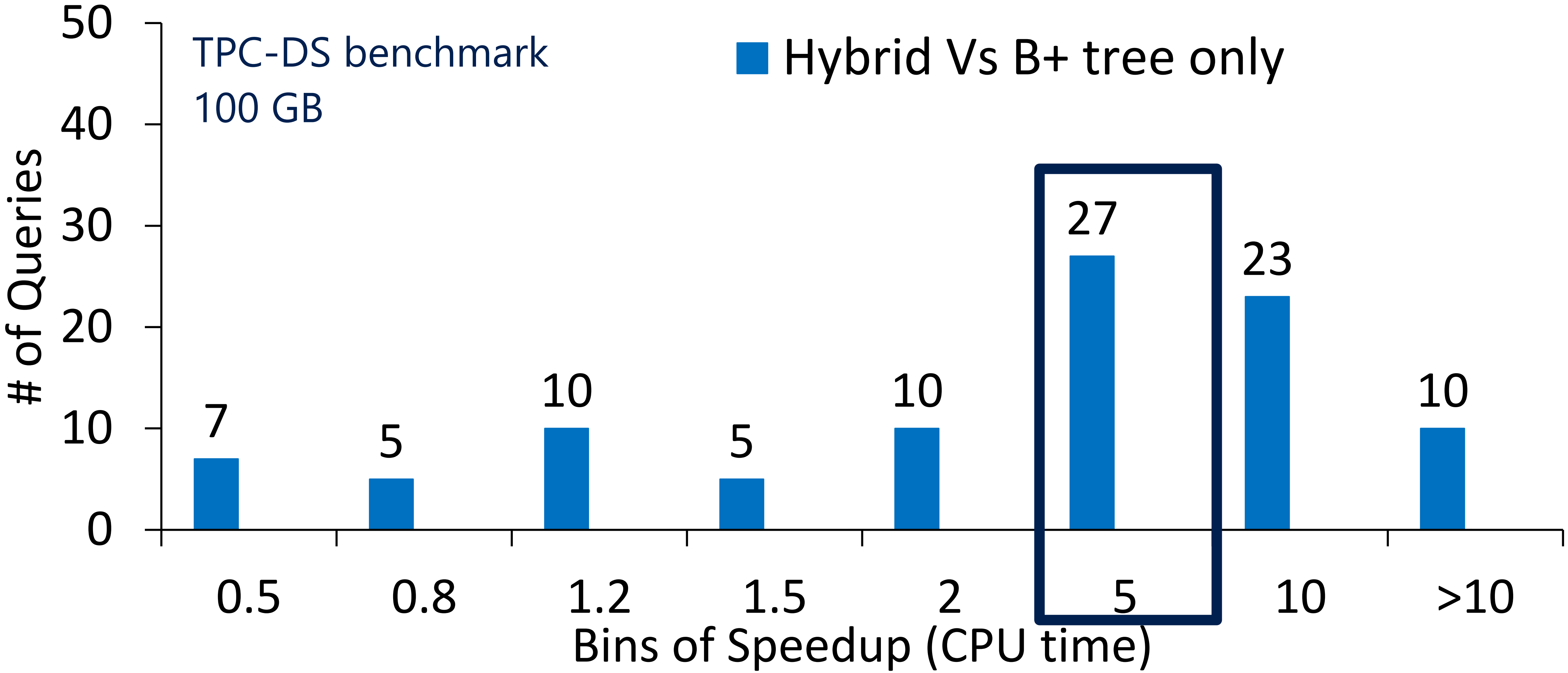
Hybrid Query Plans are common



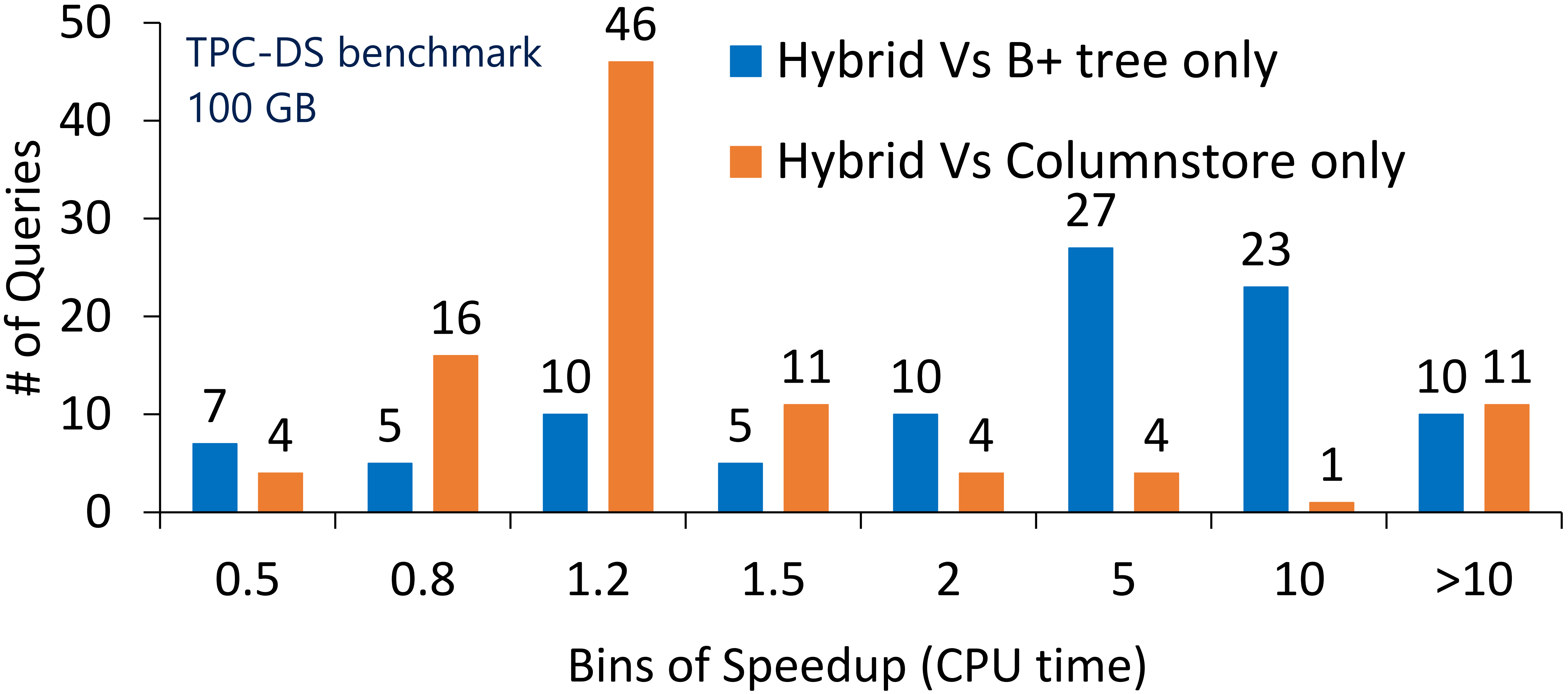
Columnstores & B+ trees selected by DTA



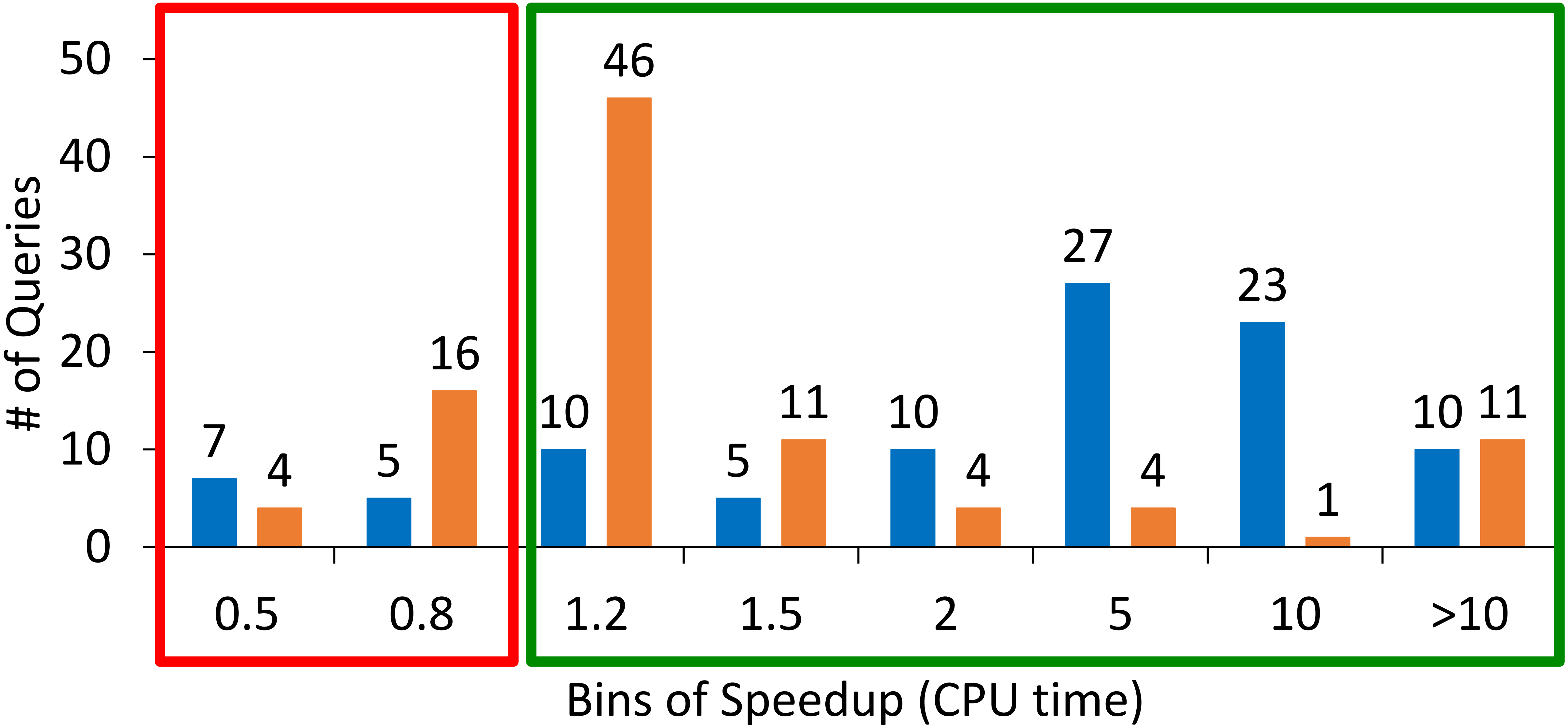
Columnstores & B+ trees selected by DTA



Columnstores & B+ trees selected by DTA



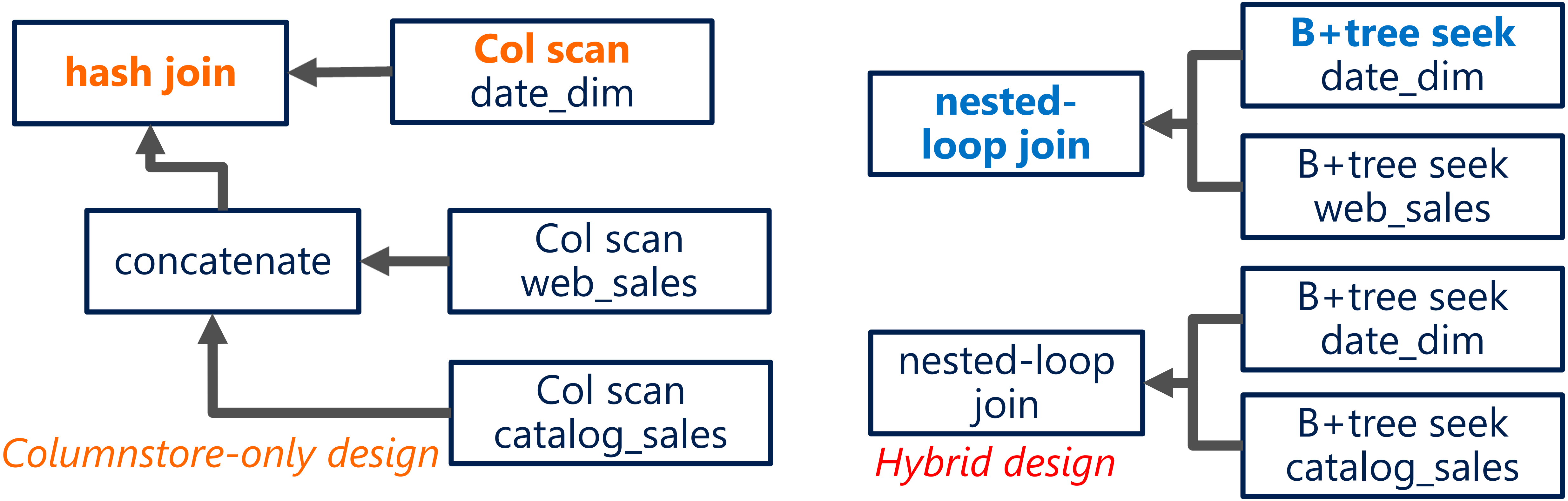
Columnstores & B+ trees selected by DTA



Hybrid designs significantly improve decision support workload

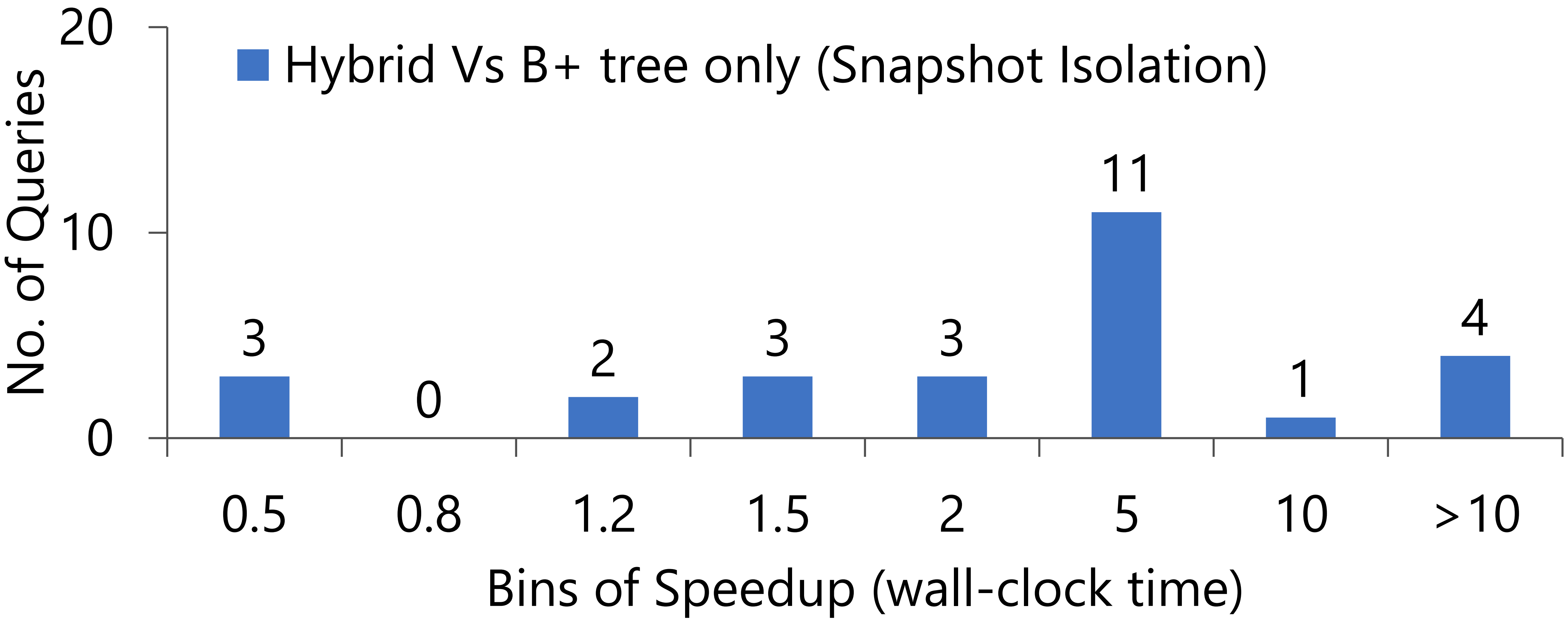
Example: Hybrid Design for TPC-DS Q54

- Q54: several selective predicates on dimension table: date_dim
- DTA: B+ tree index on date_dim as well as fact tables catalog_sales and web_sales
- **20X** reduction in execution cost of **leaf nodes** (8,000 ms to 400 ms),
- **10X** reduction in query execution cost (from 26,000 ms to 2,600 ms)



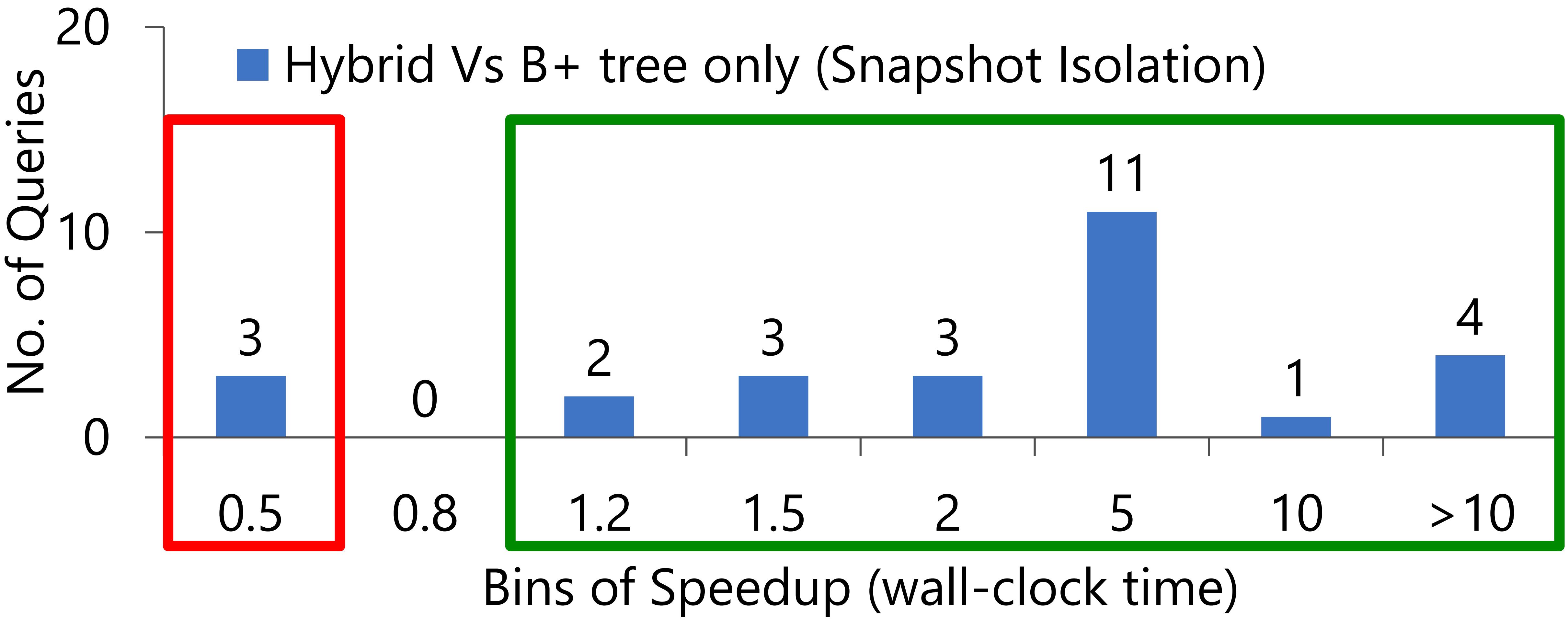
CH Benchmark: OLTP + Analytics

15 cores for analysis queries, 5 cores for transactions, workload runs 6 hours, 1K warehouses



CH Benchmark: OLTP + Analytics

15 cores for analysis queries, 5 cores for transactions, workload runs 6 hours, 1K warehouses



2X slowdown of transactions & 10X speed-up of analytics

Summary

Hybrid Designs: crucial for mixed workloads

- Hybrid physical designs should not be ignored!
  - Effective even for read-only queries
- Small slowdown for OLTP transactions and 10X or more speed-up of complex analytical queries
- DTA can recommend combination of Columnstores and B+ trees
- Several open challenges
  - Columnstore size estimation
  - Modeling concurrency effects

Thank you

Backup

Microsoft

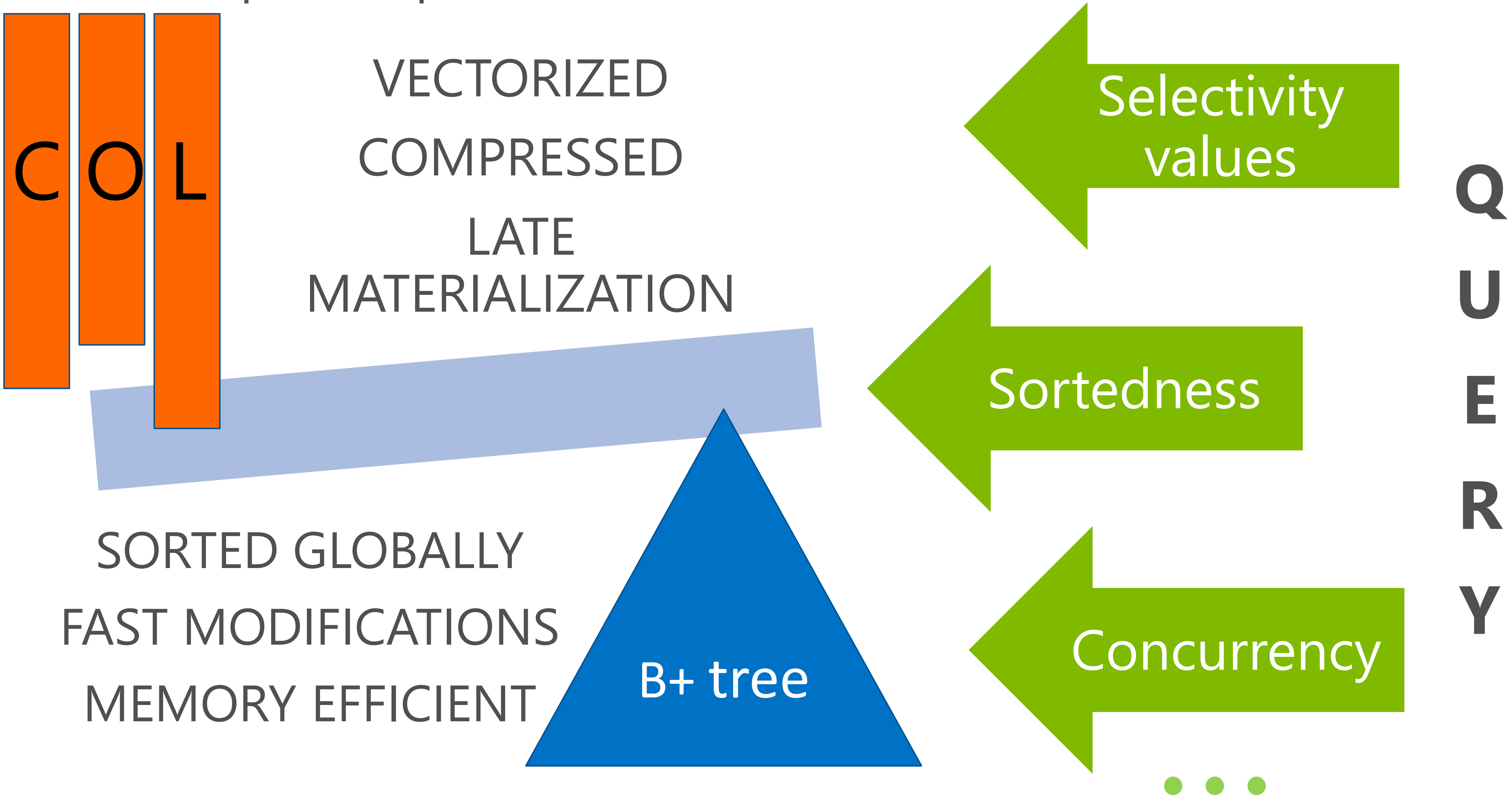
THE UNIVERSITY OF CHICAGO

UNIVERSITY of WASHINGTON

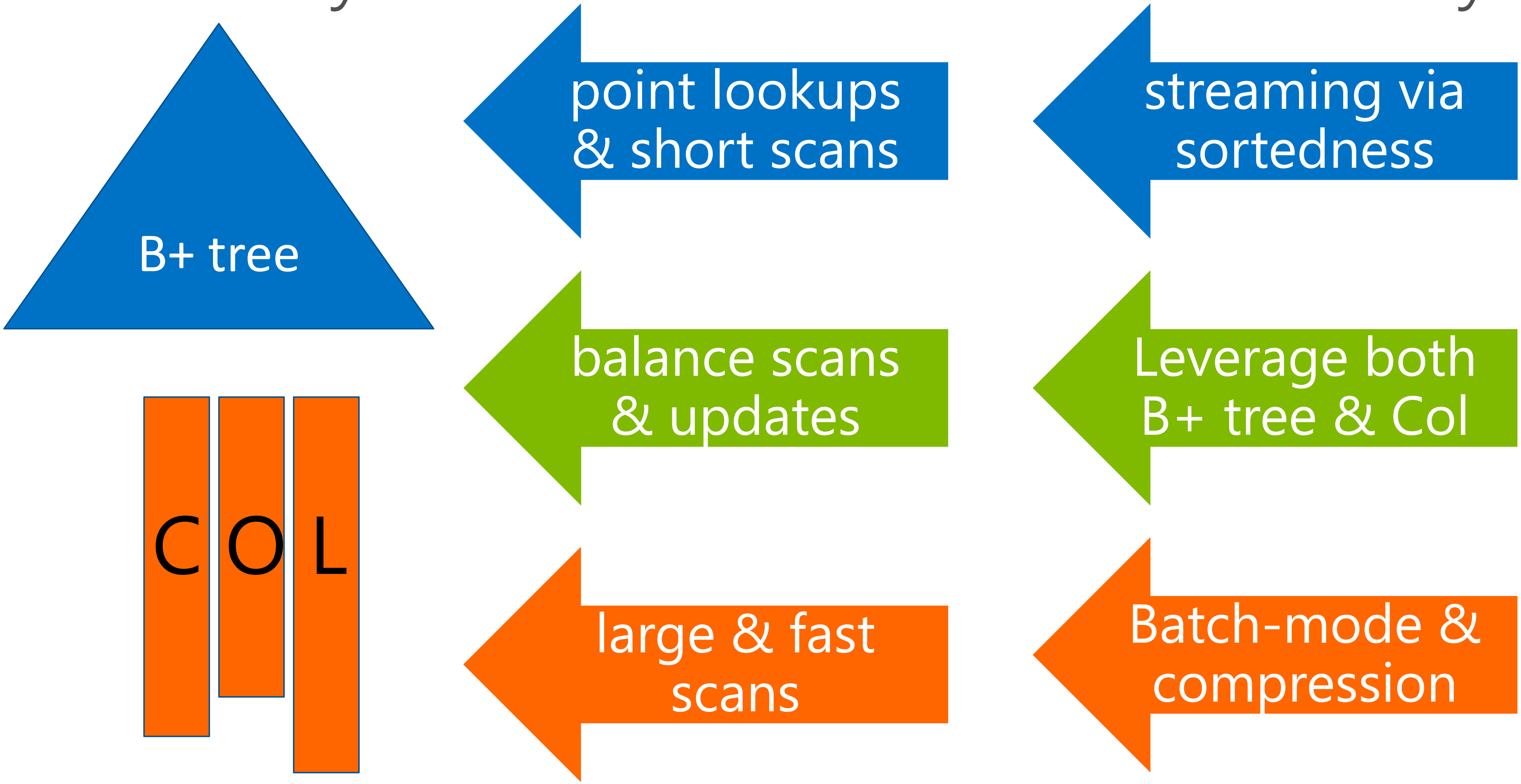
Notes

- Primary index – means that this is the main storage of the data
- Mixed workload insight: as soon as we have one big scan, the Hybrid Design provides much better performance
- Primary Columnstores not efficient for mixed workloads
- DTA – optimization tool + access to the query optimizer via the what-if API
- SQL Server Columnstore not globally ordered
- Explore when to use B+ tree or a Columnstore
- SQL supports hybrid designs (not new, already there)

Tease apart performance characteristics



Takeaways from micro-benchmark analysis



Key takeaways from end-to-end evaluation

- Hybrid designs can result in 10X to 100X improvement in execution costs compared to B+ tree or Columnstore only designs
- DTA – automated recommendation of hybrid physical designs is cost-based and workload-dependent
- Open challenges for hybrid designs in query optimization, concurrency, and locking

Compression with RLE & GEE estimator

Initial sample

A	B
30	0.0
30	0.1
0	0.0
10	0.0
30	0.1
30	0.1

Encoded

A	B
3	0
3	1
0	0
1	0
3	1
3	1

Sorted by B,A

A	B
0	0
1	0
3	0
3	1
3	1
3	1

Compressed

A
0, 1
1, 1
3, 4

B
0, 3
1, 3

GEE estimator groups that occur once in the sample are scaled by *total_size / sample_size* (e.g. [0,1] and [1,1]), other groups are counted once in total

Columnstore size estimation

- Not scalable to scan and apply encodings on the whole dataset
- Estimate per column/segment sizes using sampling
- For each column segment: 1) encode values
2) determine optimal row ordering 3) compress.
- Variety of encodings, row ordering optimization and compression techniques make size estimation hard

Customer workloads and benchmarks

Work-load	DB Size (GB)	# of tables	Max table size (GB)	Avg. # of cols per table	# of queries	Avg. # of joins	Avg. # of ops per plan
Cust1	**172**	23	63.8	14.1	36	7.2	29.1
Cust2	44.6	614	44.6	23.5	40	8.1	28.3
Cust3	138.4	**3394**	79.8	**26.3**	40	8.7	24.1
Cust4	93	22	54.8	20.3	24	6.9	24.4
Cust5	9.83	474	1.52	5.5	47	**23.1**	**57.7**
TPC-DS	87.7	24	34.9	17.2	**97**	7.9	28.2
TPC-CH		11			27		

Motivation

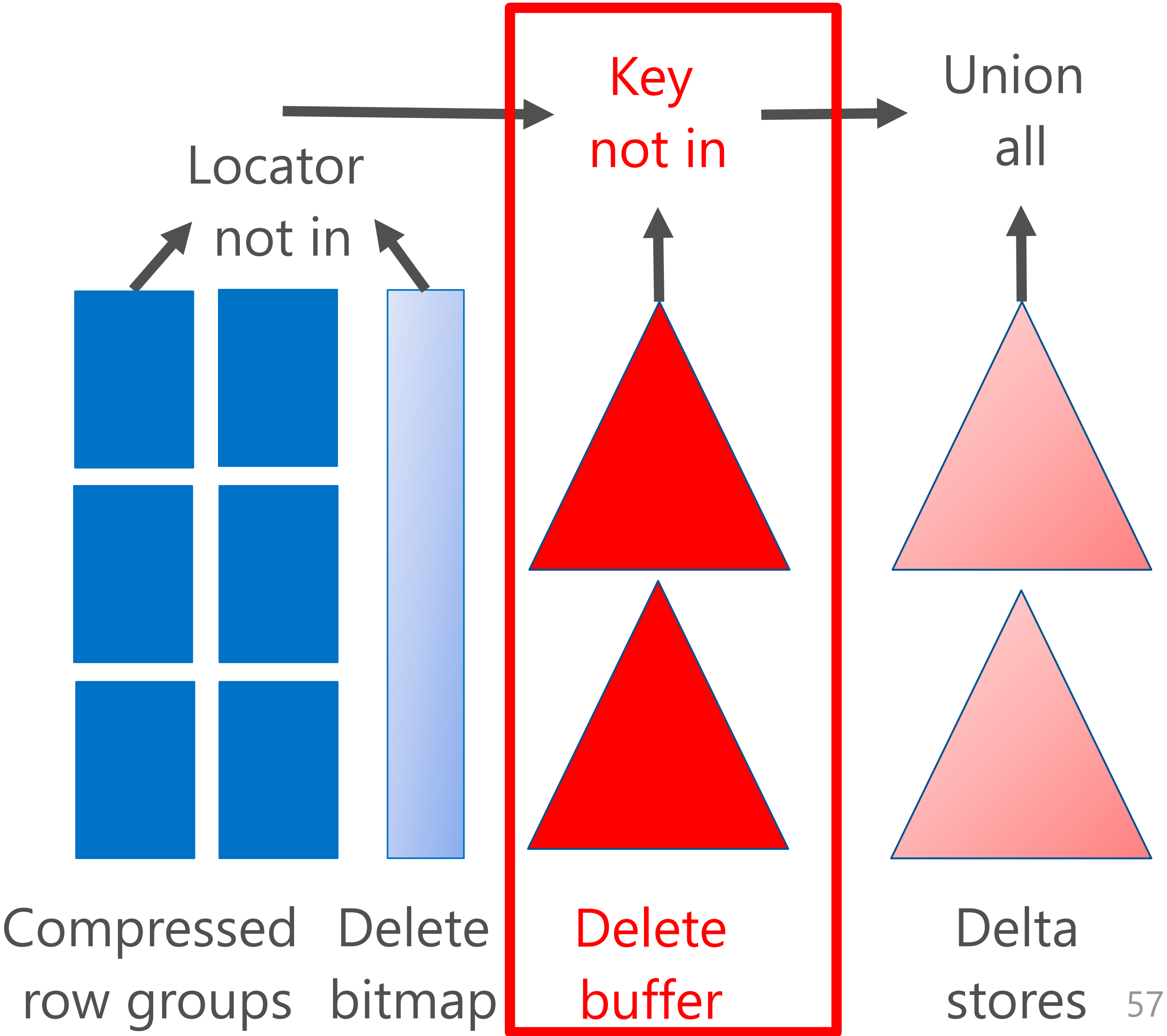
- *Wide variety of workloads* with different usage characteristics: OTLP, OLAP, Mixed workloads (e.g. real-time analytics)
- Mixed workloads imply need for Hybrid Physical Designs to achieve good performance
- SQL Server supports a wide variety of Physical Design options
- What are the ramifications of Hybrid Designs for auto-indexing in SQL Server?

Major Questions

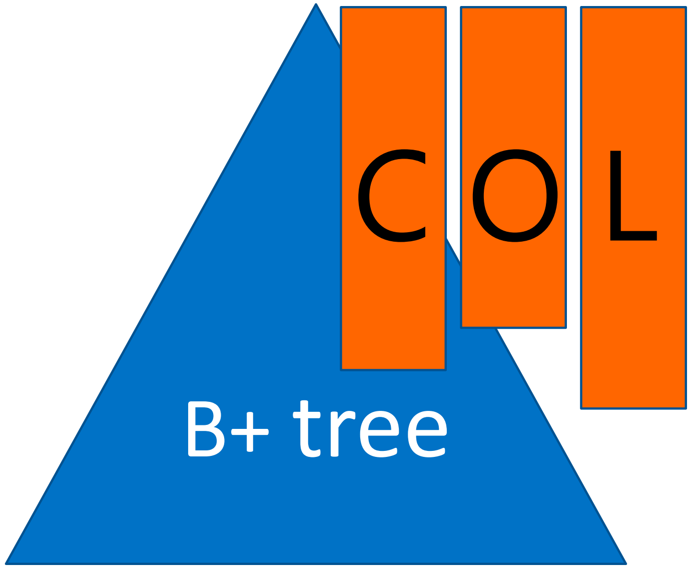
- Efficient data skipping and sort order of B+ tree vs. efficient vectorized processing of Columnstores
  - What are the trade-offs?
    - **Data skipping**
    - Concurrency
    - **Memory constraints**
- Impact of updates on B+ trees & Columnstores
  - Primary (clustered) and secondary (non-clustered) have different designs
- Performance of Hybrid Designs for Mixed Workloads - short updates and reporting queries scanning data

Two types of Columnstore Indexes (CSIs)

- CSI updatable via auxiliary structures
- Primary CSI designed to optimize scan performance in DW
  - No delete buffer in primary CSI, making small updates expensive



Characteristics: B+ tree, Col & hybrid

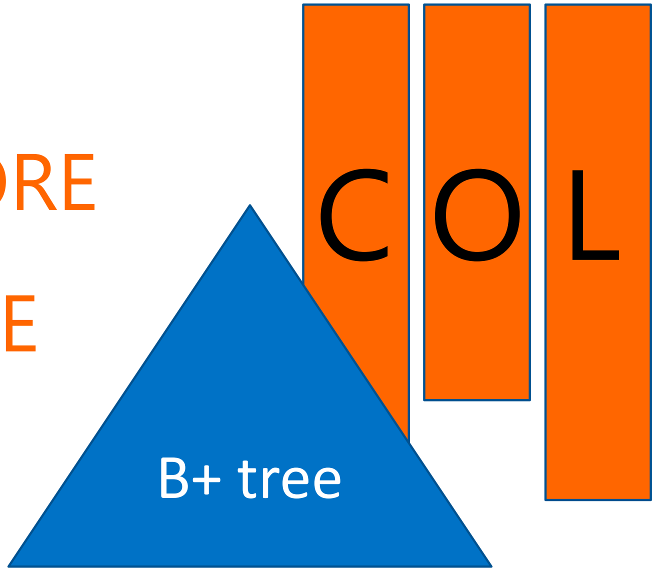


SORTED B+ TREE

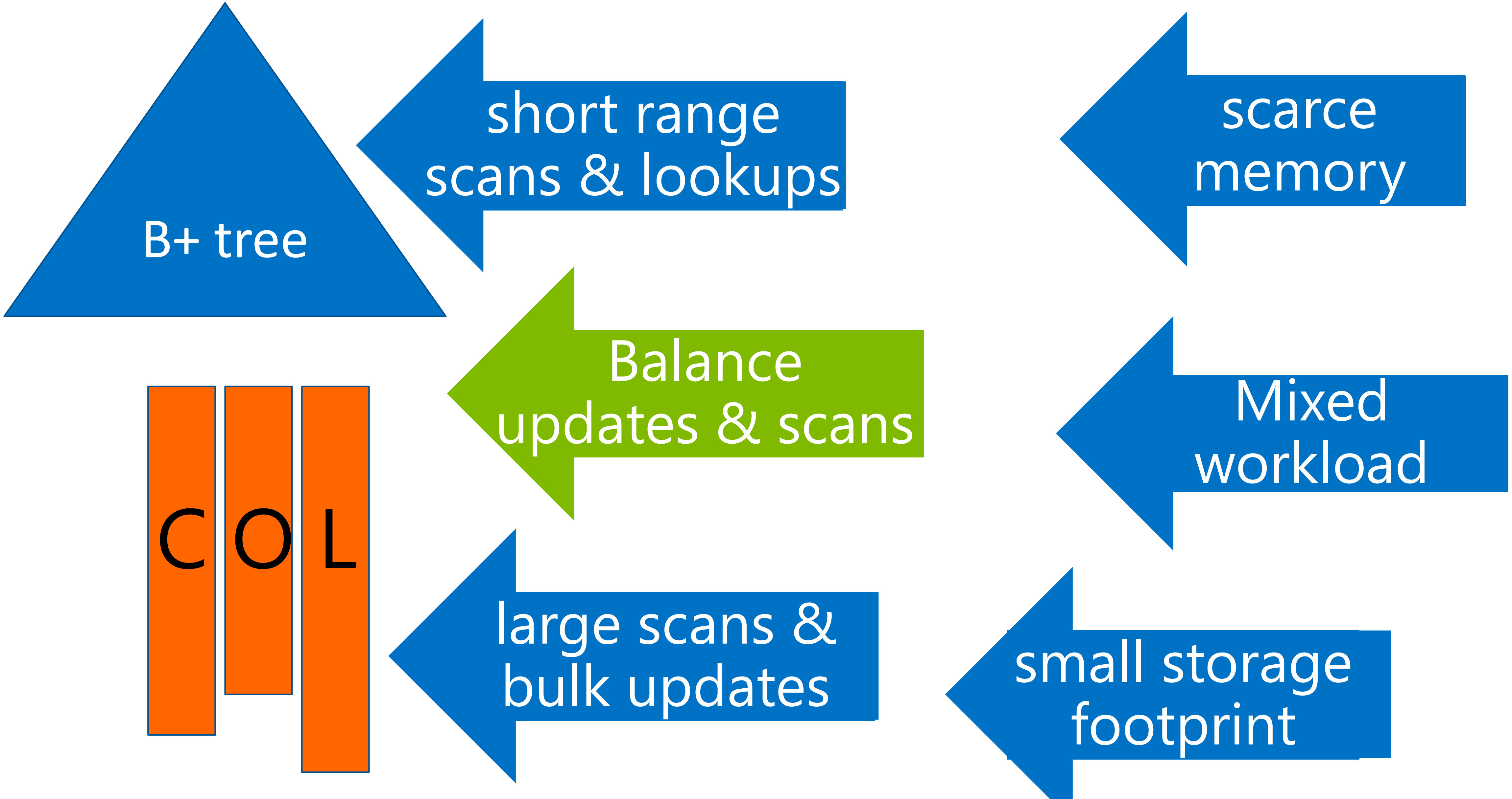
PRIMARY B+ TREE
&
SECONDARY COLUMNSTORE

NOT SORTED GLOBALLY COLUMNSTORE

COLUMNSTORE PRIMARY STORAGE
&
SECONDARY B+ TREE

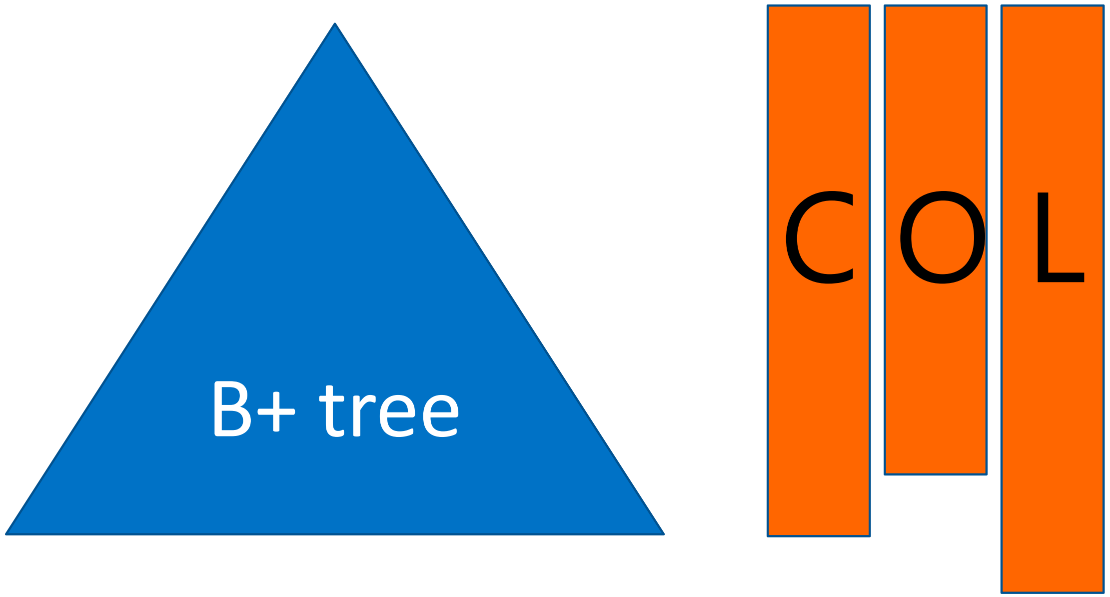


Takeaways from micro-benchmark analysis



Takeaways from Microbenchmark analysis

- B+ trees: suitable for point lookups, short range scans, update-only
- Sortedness of B+ trees only help when memory is insufficient
- Secondary CSIs strike a right balance between scans and updates
- Hybrid physical designs are essential for many mixed workloads where updates often have selective predicates

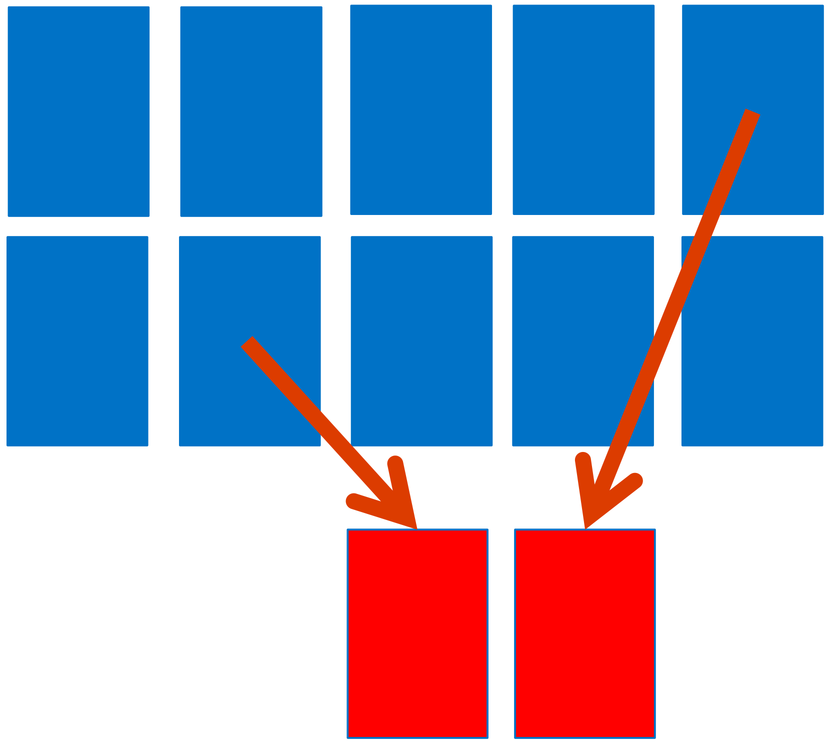


Mixed Workload: CH BenCHmark

- A join between TPC-C benchmark with the analytical H component from TPC-H
- **C component**: unmodified TPC-C schema and 5 transactions
- **H component:** 3 additional tables and adapted 22 queries which mimic TPC-H queries
- Concurrent queries compete for resources & locks
- Affinitize component C & H to different cores

Why is Columnstore size estimation hard?

Large data & design space



Sample based techniques

Complex storage

A
0.03
0.02
0.02
0.03
0.02

Input

A
3
2
2
3
2

Encode

A
2
2
2
3
3

Sort

A
2, 3
3, 2

Compress

1) Build index on samples
2) Model full index

Hypothetical Columnstore size estimation

Build index on samples

+ Simple
+ No changes required when index storage modified
- Low accuracy
- High overhead (run algorithms and store index)

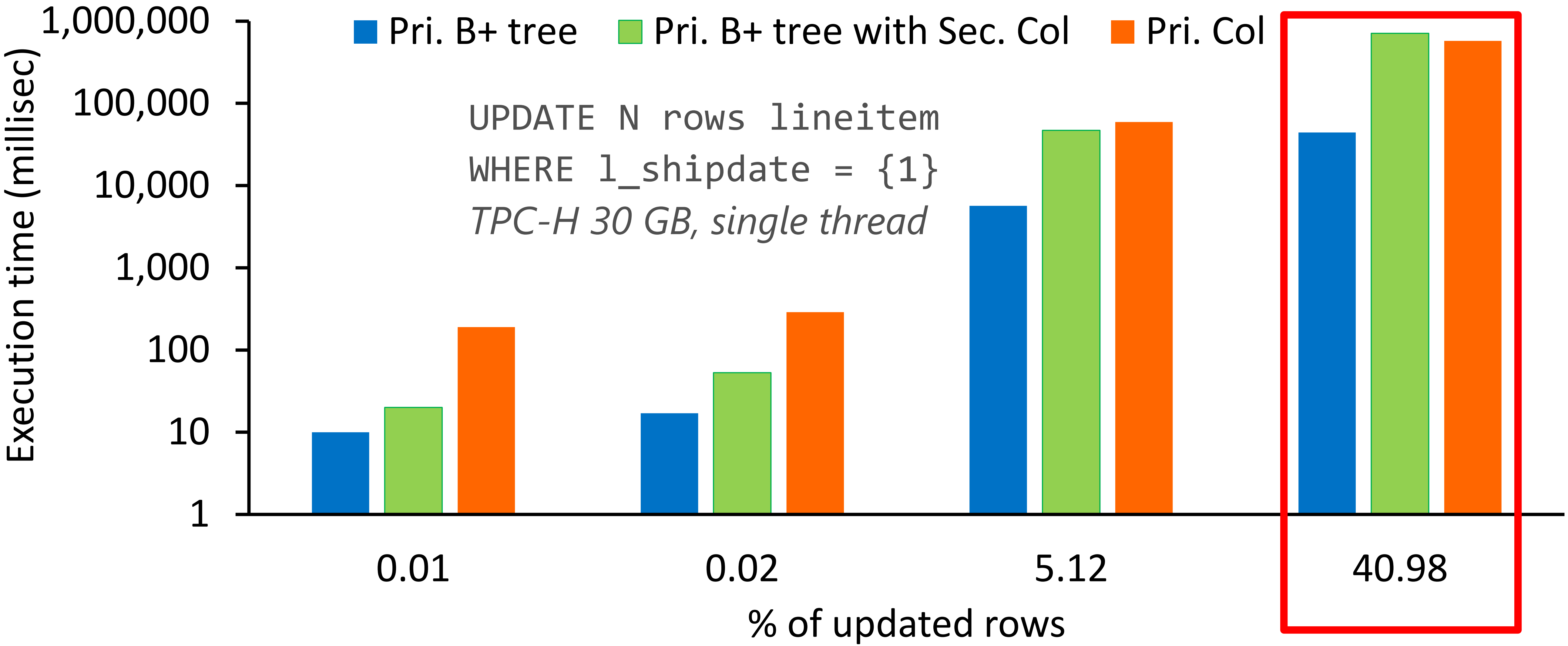
Model full index

+ More accurate
+ No overhead of sorting or writing index to disk
- High maintenance cost
- Complex (uses GEE estimator)

Hypothetical Columnstore size estimation

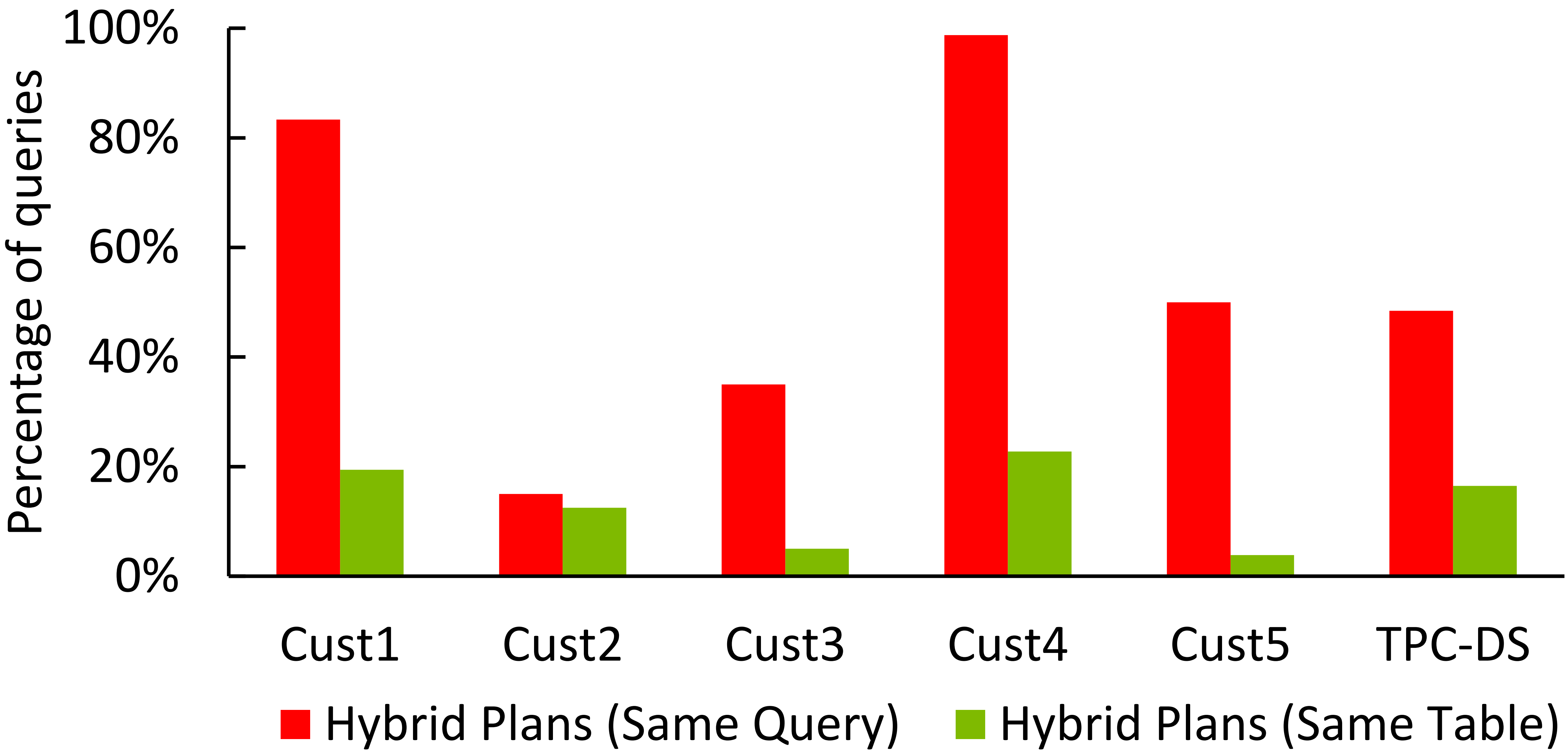
- Correctness: do not go over storage budget
- Efficiency: cannot afford to build the whole index
- Accuracy:
  - estimate the size on samples of data
  - model complex storage involving encoding, sorting and compression

Update performance

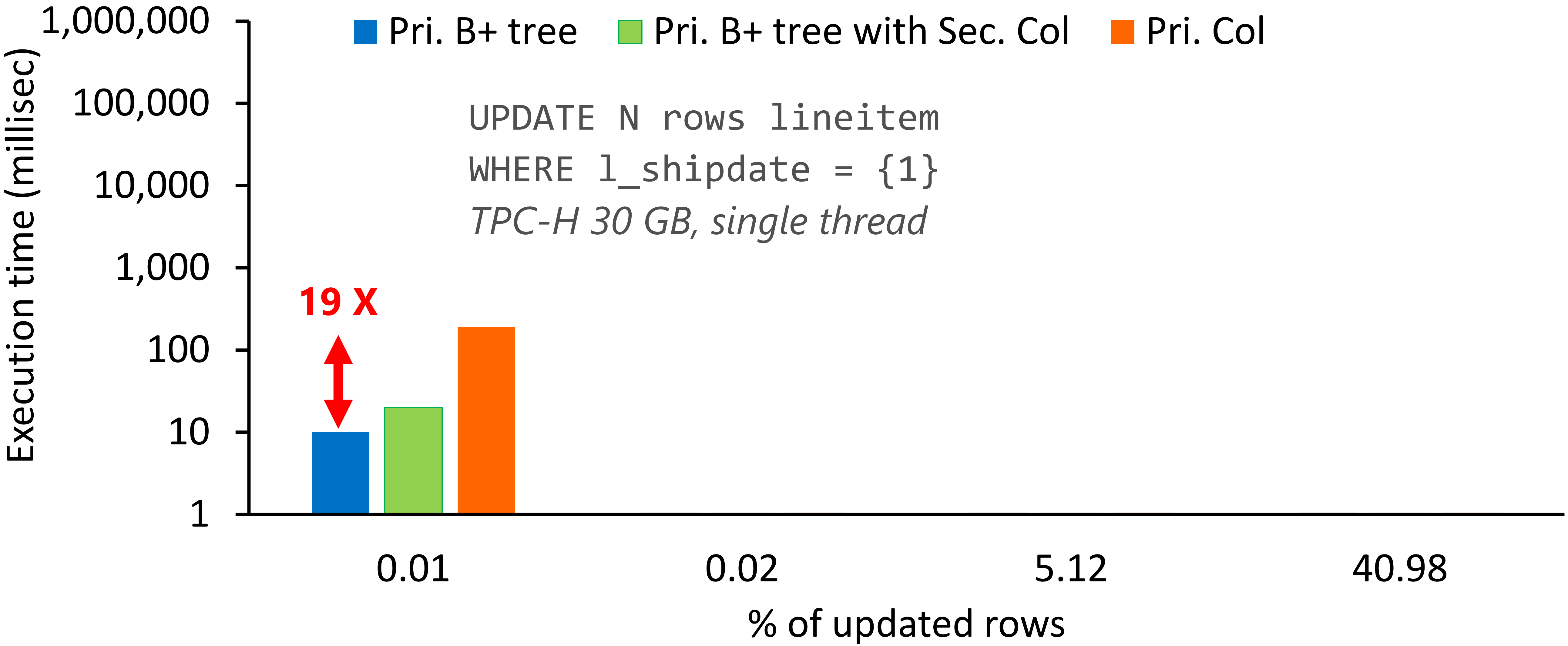


Primary & Secondary Columnstores comparable for large updates

Hybrid Query Plans are common

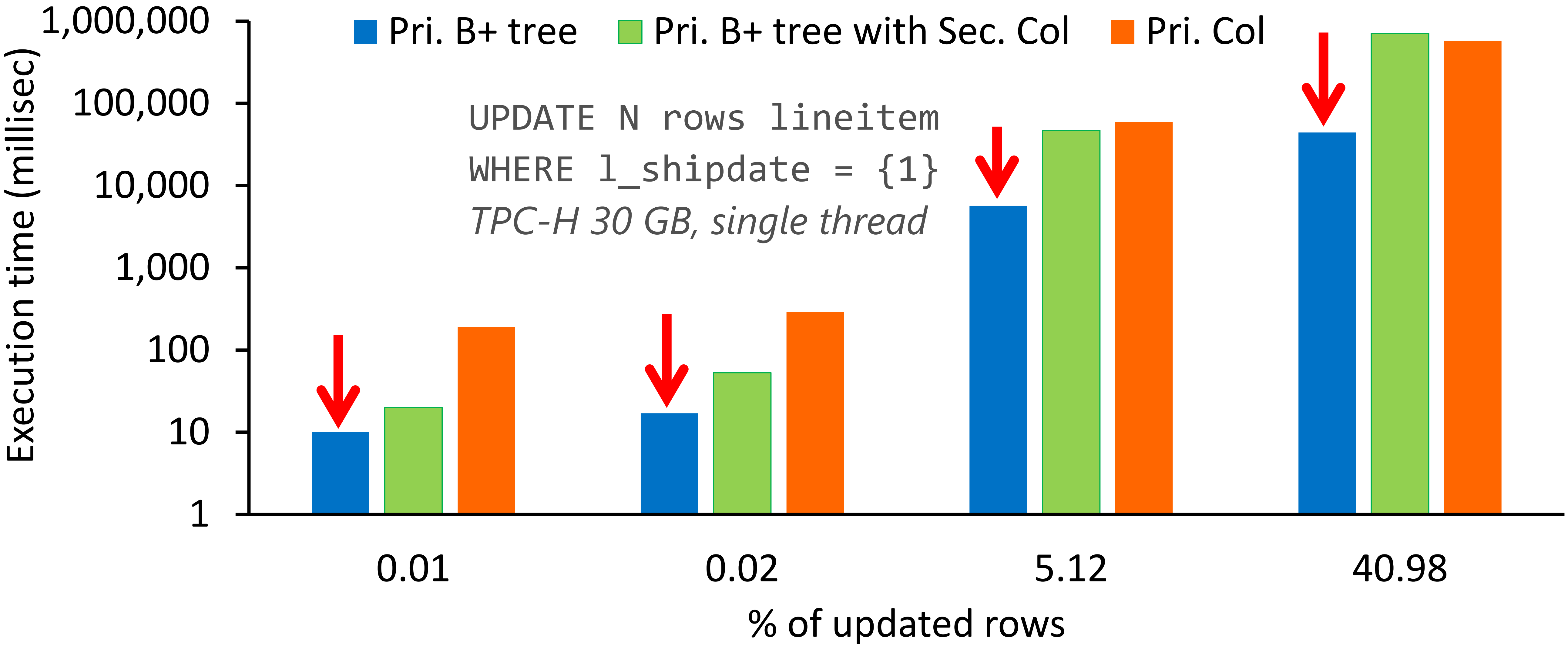


Update performance



Primary Columnstores incur high cost for small updates

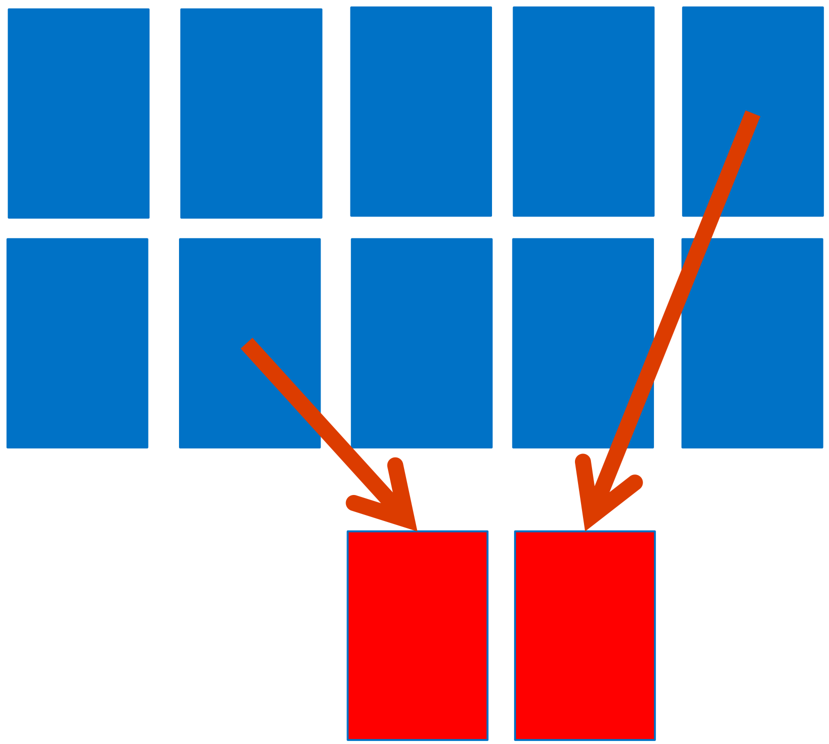
Update performance



Cheaper updates for B+ trees than for Columnstores

Why is Columnstore size estimation hard?

Large data & design space



Sample based techniques

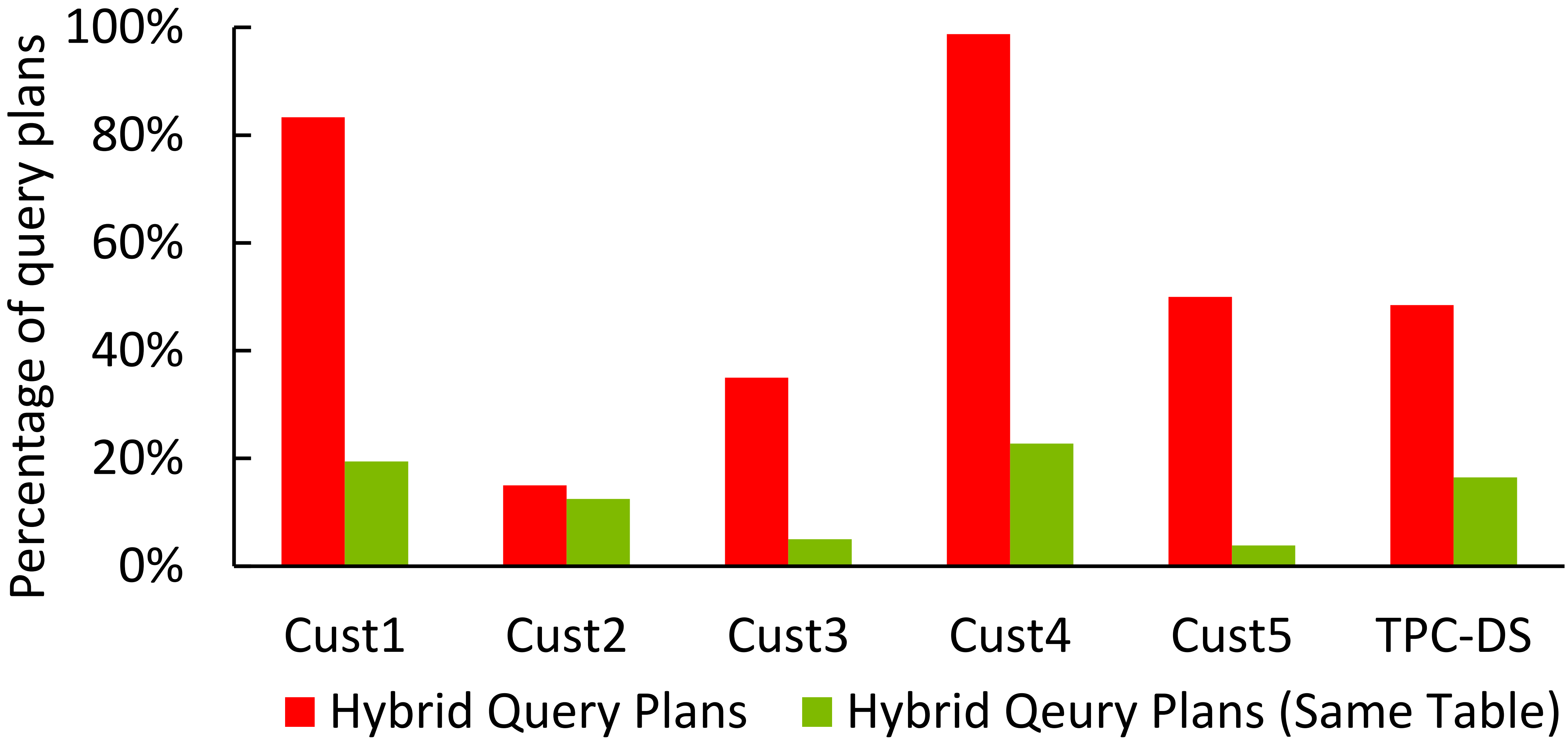
Complex Columnar storage

A	A	A
0.03	3	2
0.02	2	2
0.02	2	2
0.03	3	3
0.02	2	3
Input	**Encode**	**Sort**

A
2, 3
3, 2
Compress

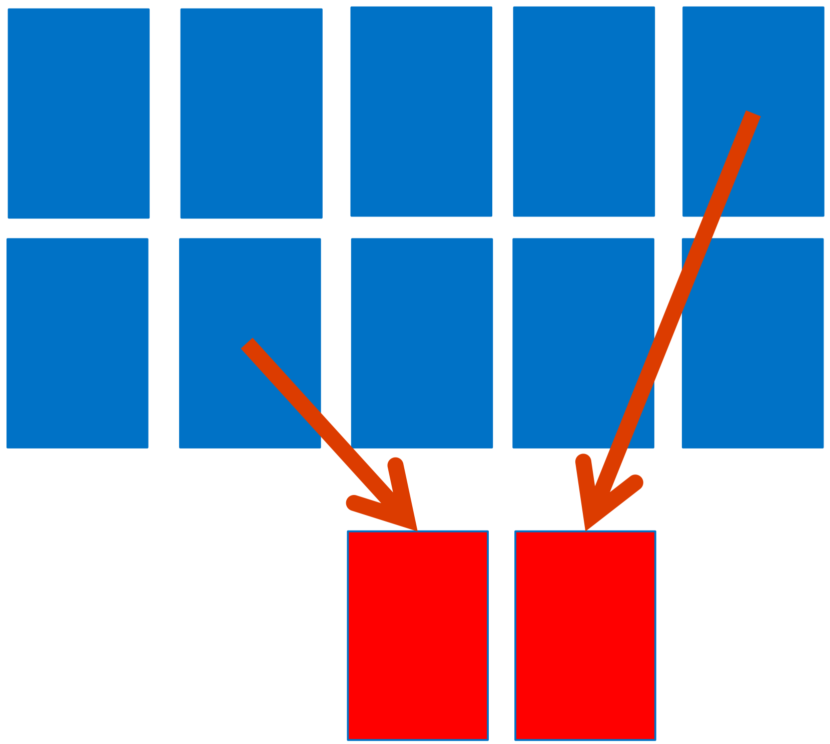
At least as hard as distinct value estimation

Hybrid Query Plans are common



Why is Columnstore size estimation hard?

Large data & design space



Sample based techniques

Complex Columnar storage

Input: A	Encode: A	Sort: A
0.03	3	2
0.02	2	2
0.02	2	2
0.03	3	3
0.02	2	3

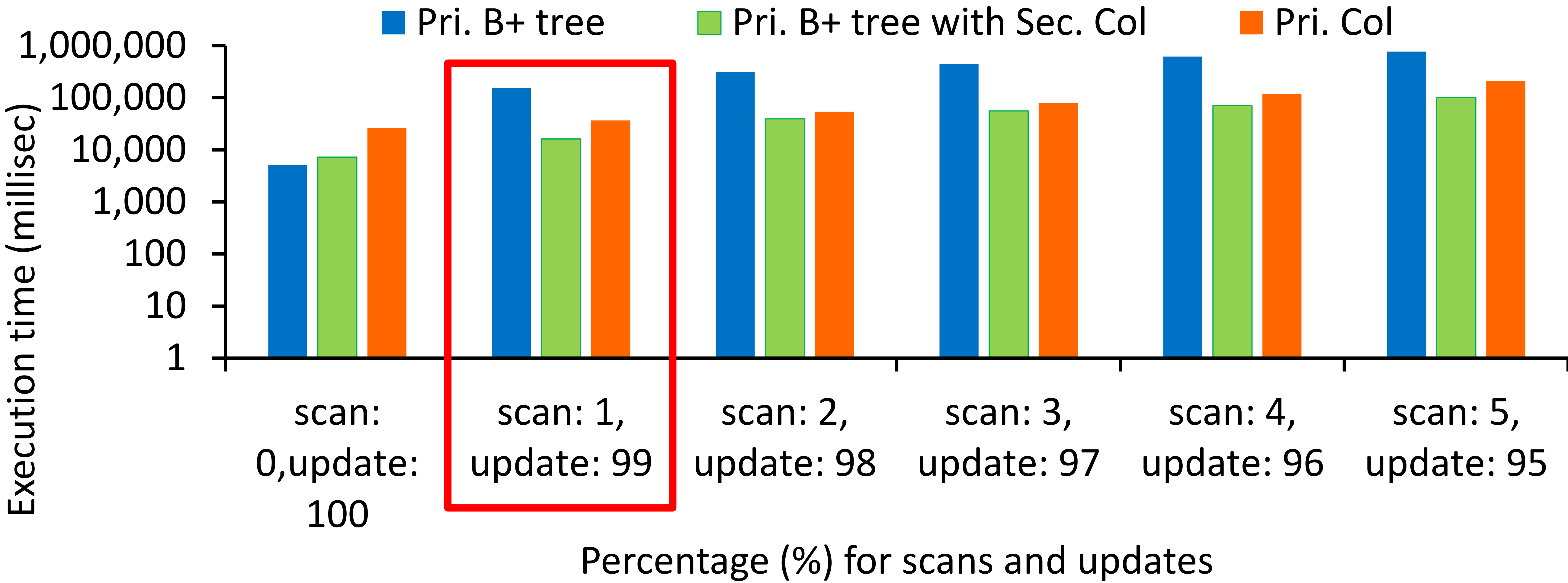
Compress: A
2, 3
3, 2

Input **Encode** **Sort** **Compress**

At least as hard as distinct value estimation

Mixed workload: large scans & small updates

Update *top 10 rows,* ***Select*** *sum of quantity & price for a single shipdate from lineitem table*



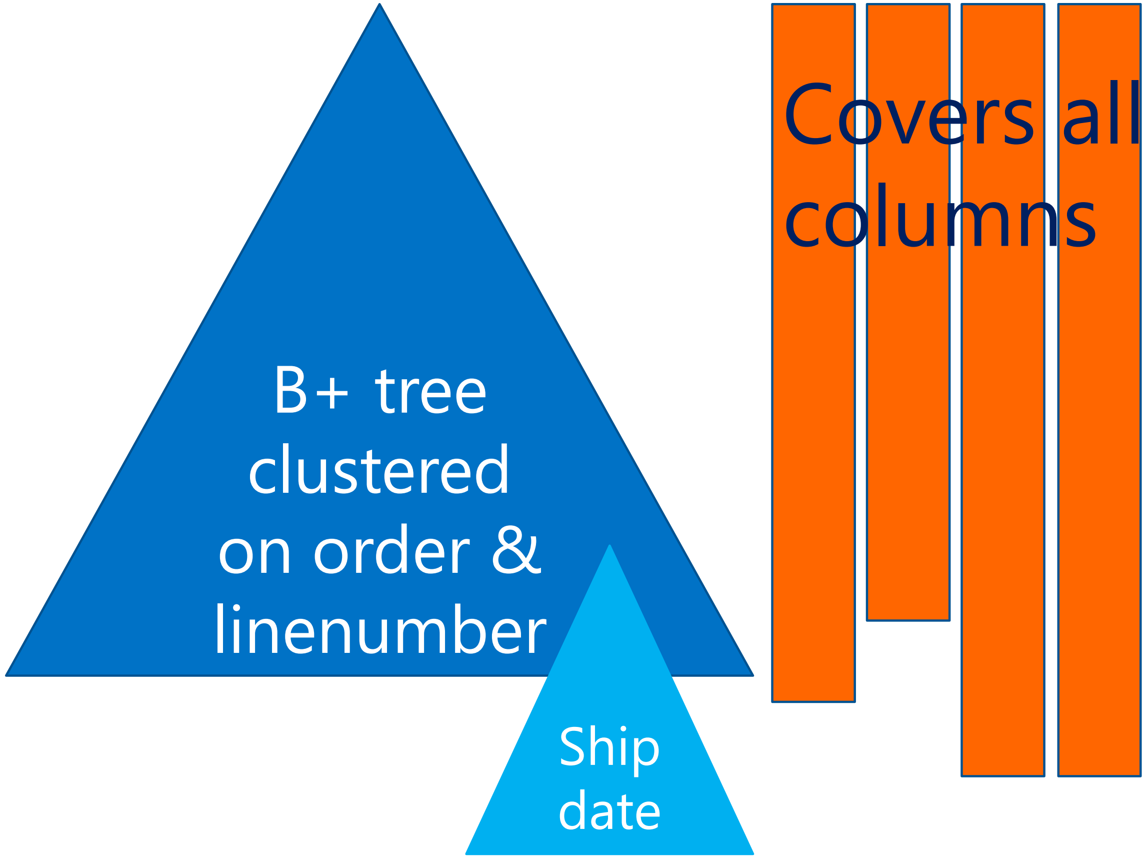
Secondary Columnstore: balance small updates & large scans

Differences to Kester et al. (sections 2.4 & 4)

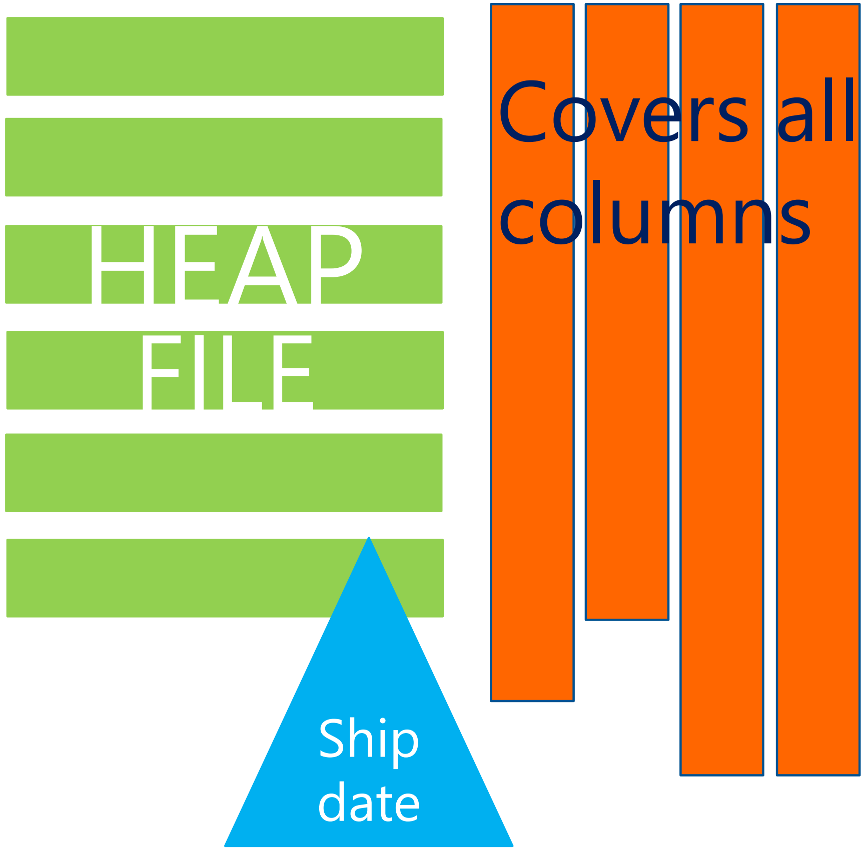
Concept	Access Path Selection	Hybrid designs
B+ tree	Main memory optimized	General (disk-based)
Scans	Shared	Non-shared
DB engine	Prototype	SQL Server
Main focus	Model Concurrency	DTA and many workloads
Physical designs	Columnstore and Secondary B+tree	Hybrid Physical Designs

Hybrid Designs

Primary B+ tree (base table)
Secondary Columnstore
Secondary B+tree on ship date

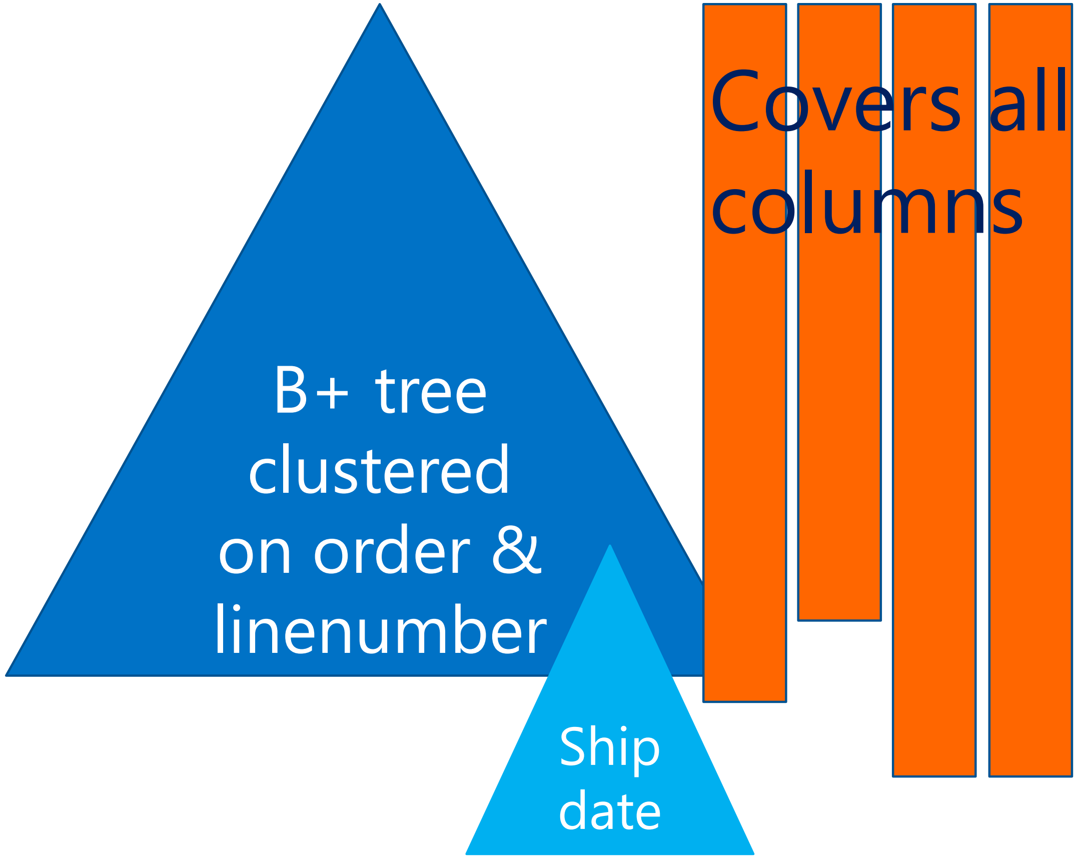


Heap file base table
Secondary Columnstore
Secondary B+tree on ship date

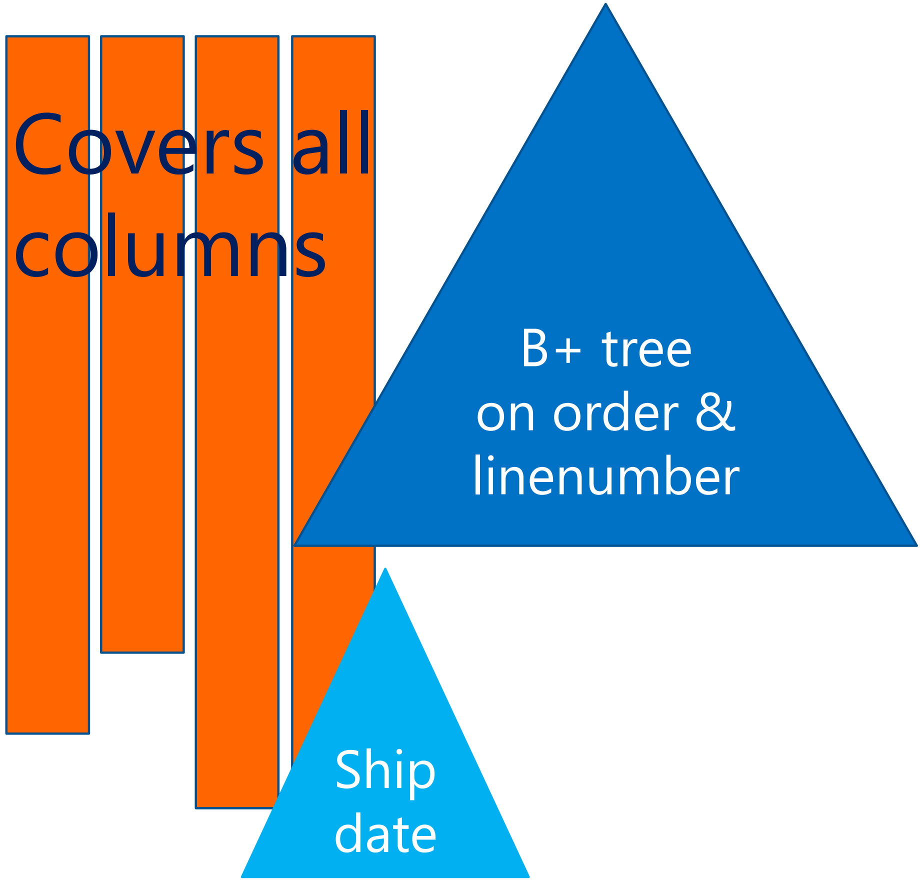


Hybrid Designs

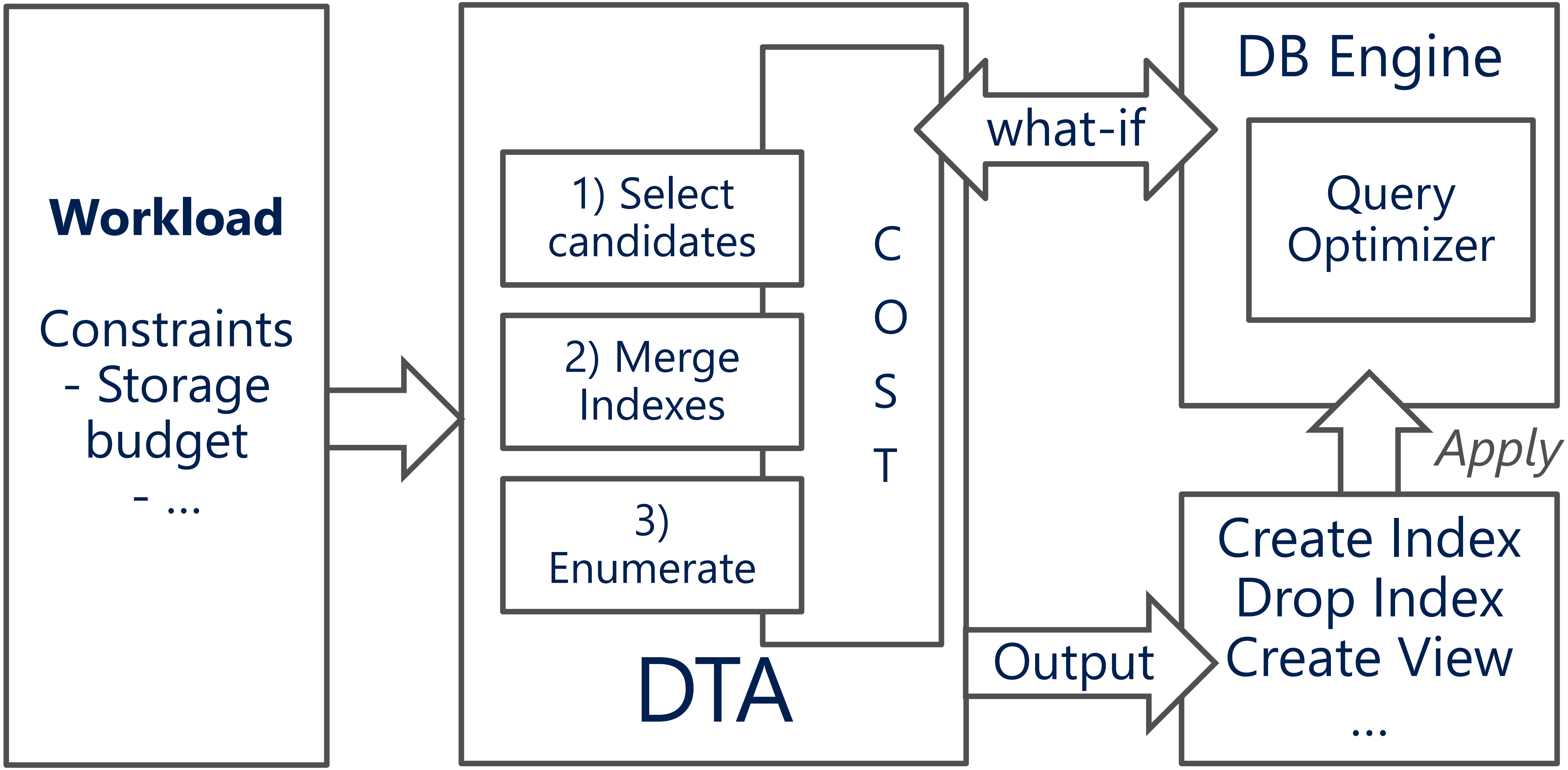
Primary B+ tree (base table)
Secondary Columnstore
Secondary B+ tree on ship date



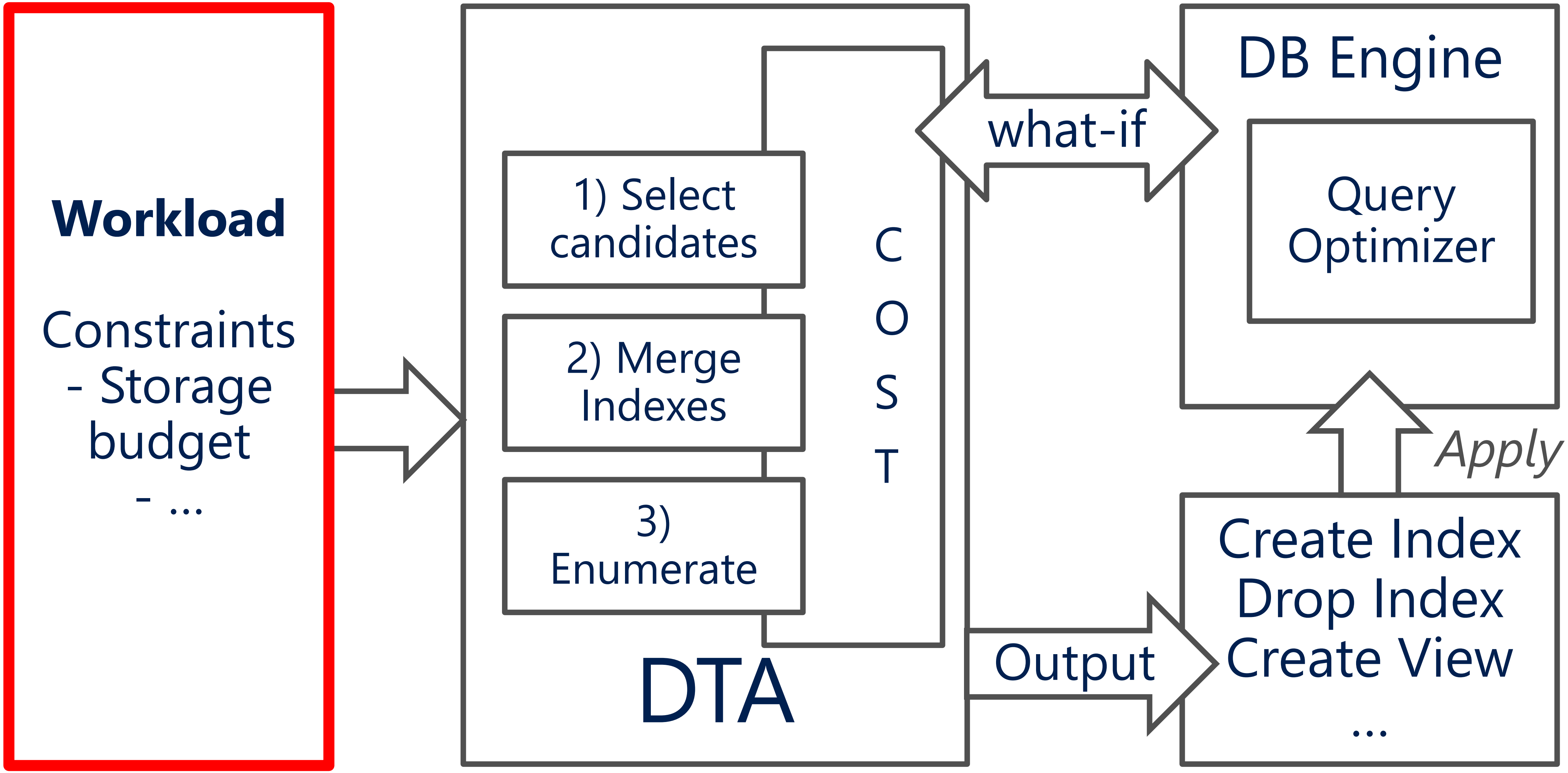
Primary Columnstore
Secondary B+ tree on order/line
Secondary B+ tree on ship date



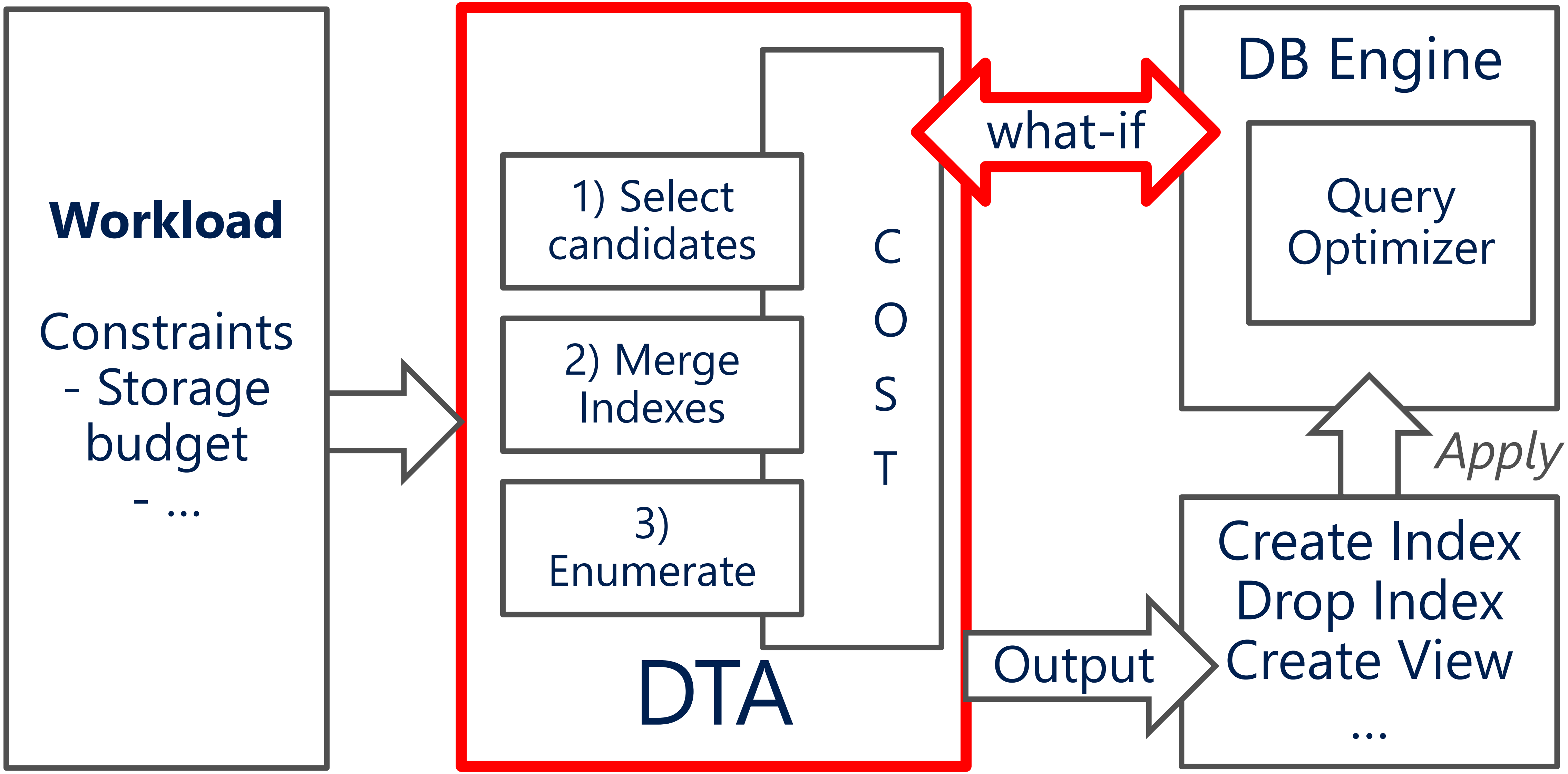
Database Engine Tuning Advisor (DTA)



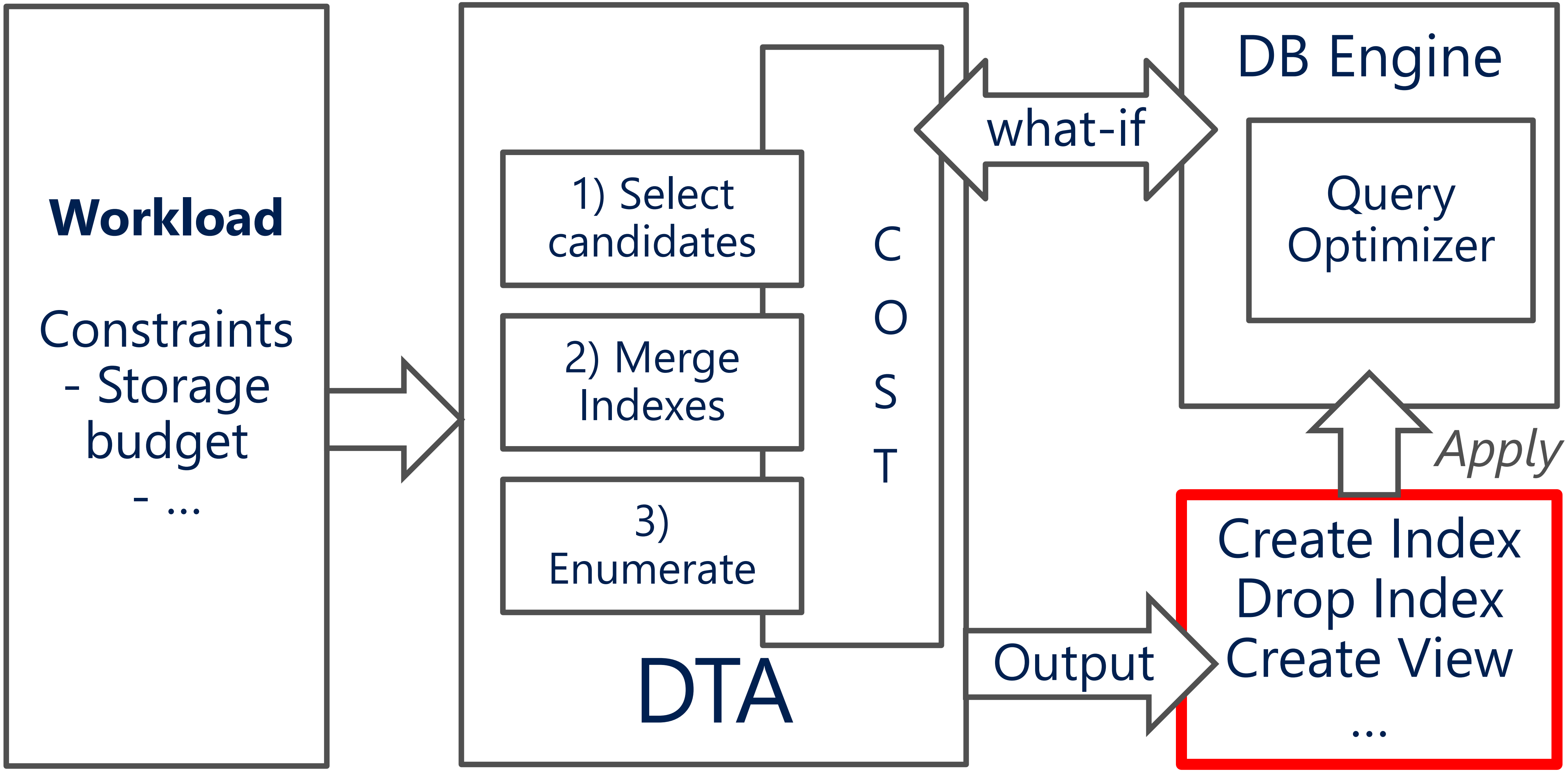
Database Engine Tuning Advisor (DTA)

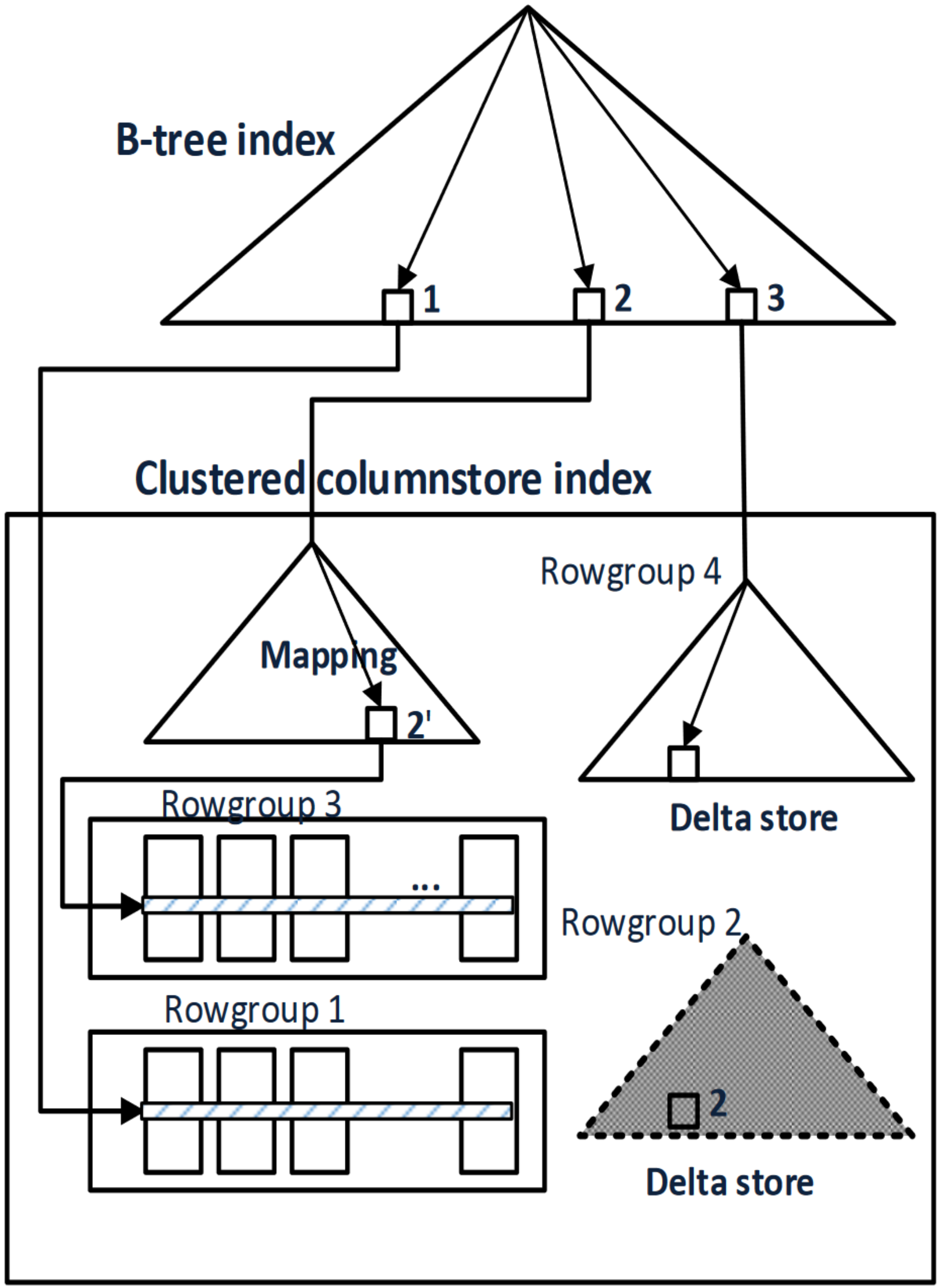


Database Engine Tuning Advisor (DTA)



Database Engine Tuning Advisor (DTA)





Primary Columnstore & Secondary B+ tree index:

- Efficient point-lookups and small range scans \w B+ tree
- enforce constraints efficiently (e.g. Primary Key)

<u>Source: </u>Real-Time Analytical Processing with SQL Server *– Paul Larson, Adrian Birka, Eric Hanson, Weiyun Huang, Michal Novakiewicz, Vassilis Papadimos (Microsoft)*