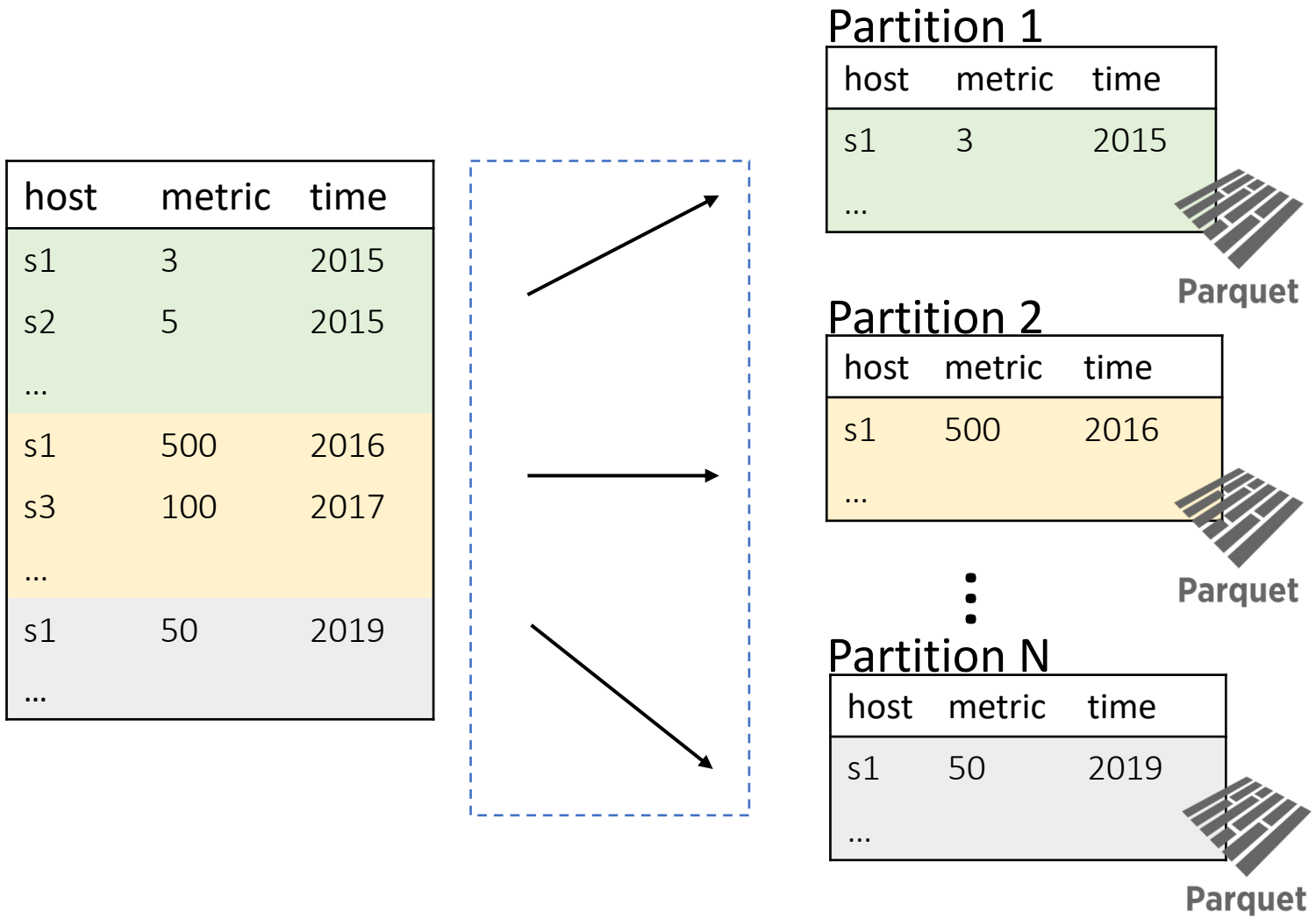
40th IEEE International Conference on
Data Engineering
Utrecht, Netherlands | 13th - 17th May
2024

Dynamic Data Layout Optimization with Worst-case Guarantees

Kexin Rong[1,3], Paul Liu[2,4], Sarah Ashok Sonje[1], Moses Charikar[2]

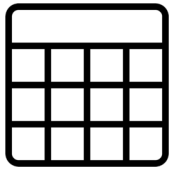
Georgia Tech[1], Stanford University[2], VMware Research[3], Vatic Labs[4]

Data partition as a basic unit for storage



Data layout: $f(row\_id) \longrightarrow partition\_id$

Data layouts affect query performance



Partition 1

host	metric	time
s1	3	2015
…		

Partition 2

host	metric	time
s1	500	2016
…		

Partition 3

host	metric	time
s1	50	2019
…		

read

read

read

read

Partition-level metadata

Part	min(time)	max(time)	min(host)	max(host)
1	2015	2015	server1	server5
2	2016	2019	server1	server5
3	2019	2020	server1	server5

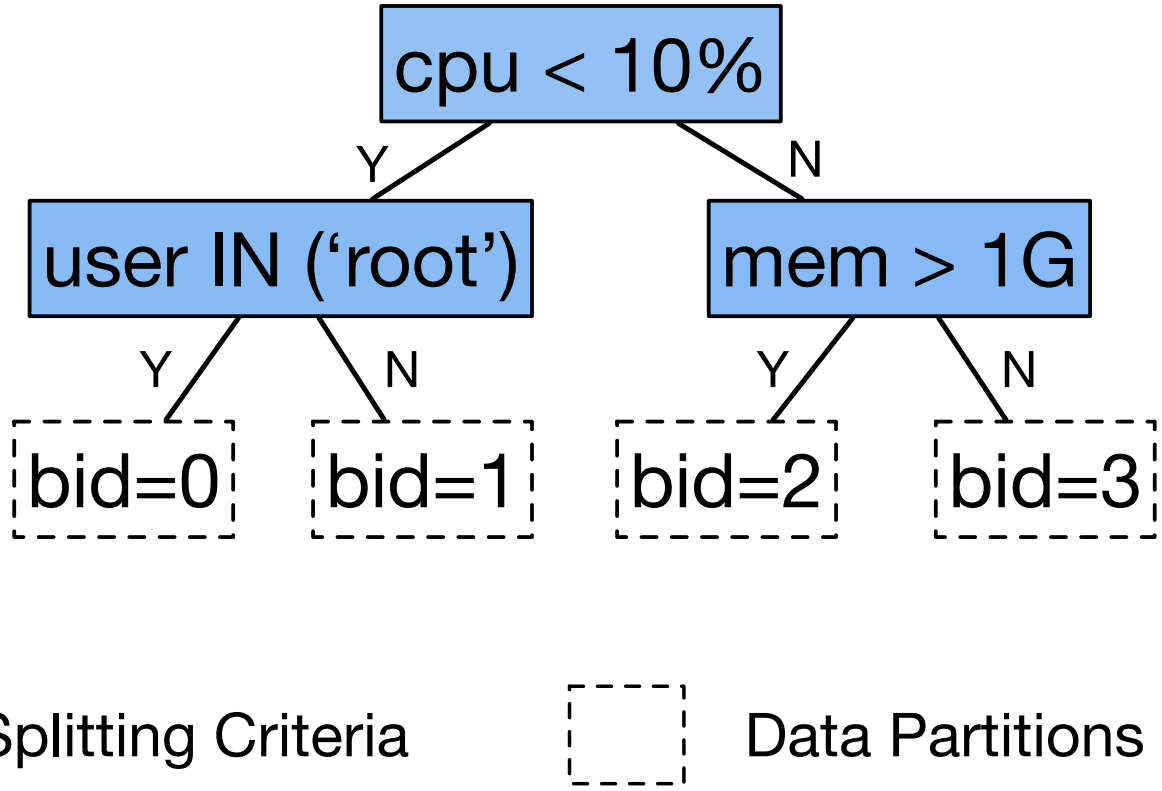
```
SELECT * FROM tbl
WHERE time = 2015
```

```
SELECT * FROM tbl
WHERE host = server2
```

Workload-aware layouts maximize data skipping

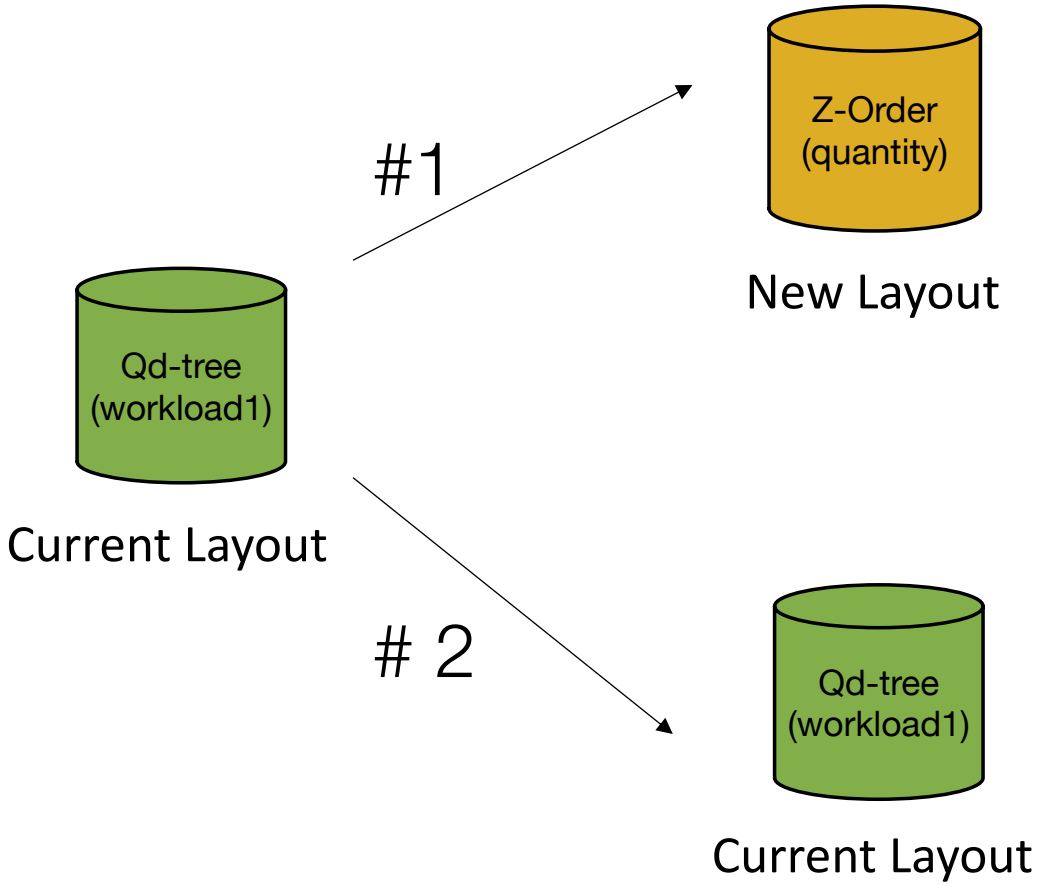
Example: Qd-tree [1]

- Use workload predicates to partition data
- Efficient for target query workloads
- Performance degrades when workload changes



[1] Z. Yang, et al. Qd-tree: Learning Data Layouts for Big Data Analytics. In SIGMOD 2020.

Our focus: dealing with workload changes



Option 1: Change layout

Reorganization cost +

Query cost -

Option 2: Do nothing

Reorganization cost

Query cost +

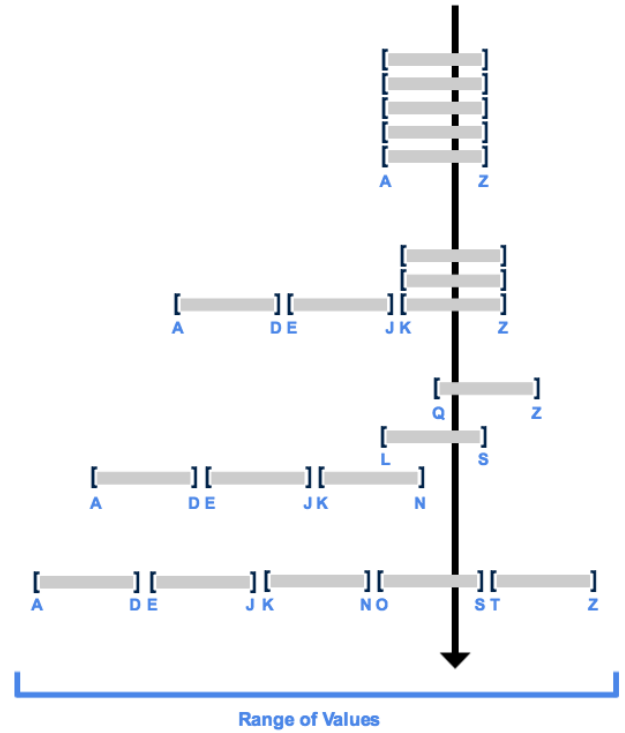
Current practices

Based on rules and heuristics

- Snowflake [1]: monitor “clustering depth” (#partitions that overlap at the same point)
- SAT [2]: monitor the ratio of actual query selectivity and data skipping rate
- No guarantee on performance

Based on future workload behaviors

- MTO [3]: can run q more queries from the same distribution before the next workload shift
- Assumptions on workload distribution



[1] https://docs.snowflake.com/en/user-guide/tables-clustering-micropartitions
[2] X. Xie, et al. “Sat: sampling acceleration tree for adaptive database repartition,” World Wide Web, vol. 26, no. 5, pp. 3503–3533, 2023.
[3] J. Ding, et al. “Instance-optimized data layouts for cloud analytics workloads,”, SIGMOD 2021.

Our idea: leveraging online algorithms

We adapt classic algorithms for *Metrical Task Systems (MTS)*[1]

- Does NOT rely on assumptions of future workload
- Complementary to partitioning and reorganization strategies
- Provide guarantees in the form of competitive ratio

$$\sup_{I} \frac{cost(online\ algorithm)}{cost(offline\ algorithm)}$$

i.e., the best sequence of layouts given the entire workload in advance

- Using this framework: >32% improvement over a static layout

[1] A. Borodin, N. Linial, and M. E. Saks, "An optimal on-line algorithm for metrical task system," J. ACM, vol. 39, no. 4, pp. 745–763, 1992.

Problem Setup

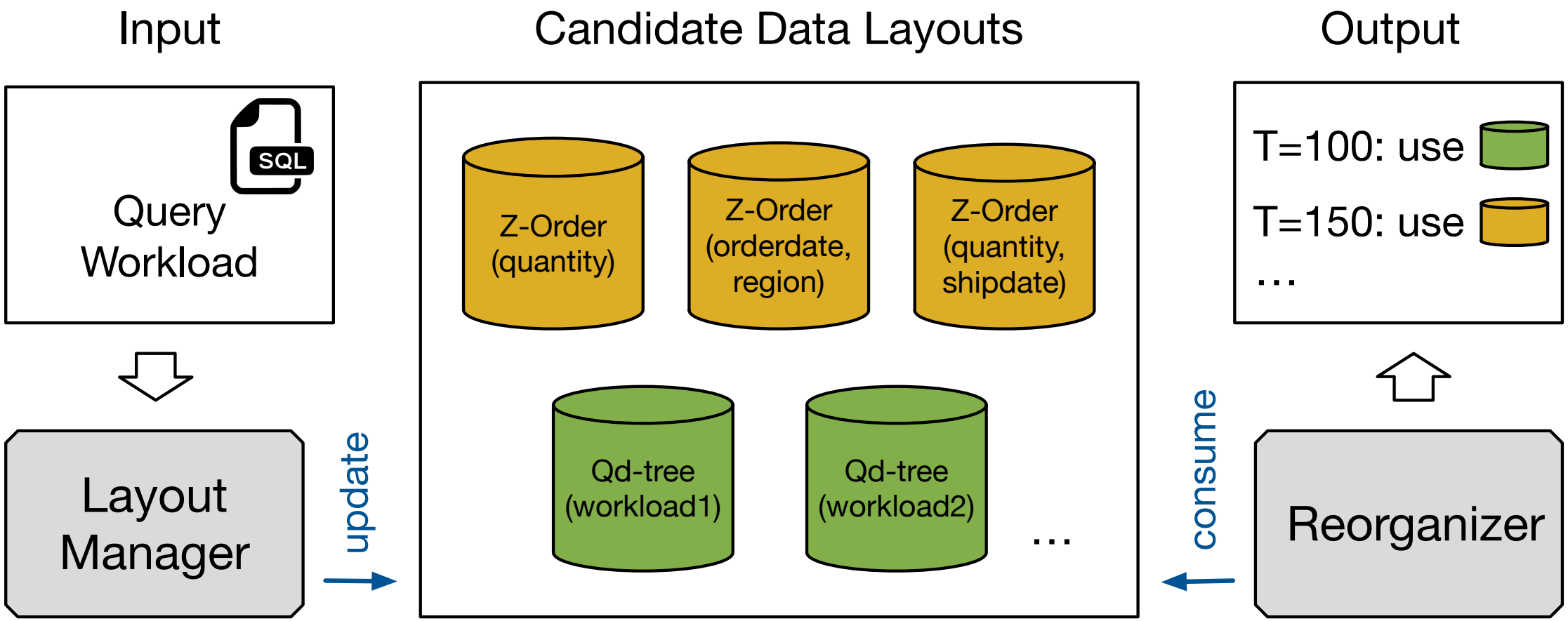
system dependent parameter;
constant for simplicity

Objective: minimize $\textbf{total cost} = \text{query} + \alpha \cdot \#\text{reorgs}$

Input: unknown sequence of queries

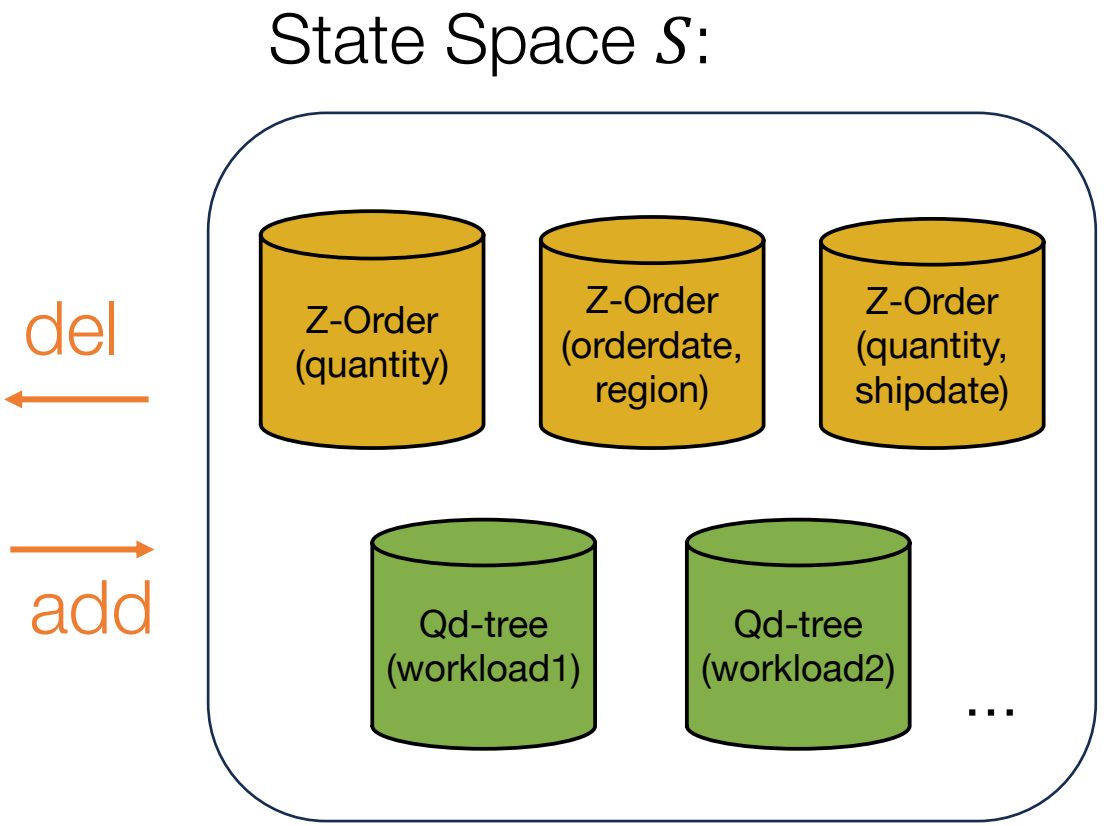
Output: *when* and *how* to reorganize

OREO: Online Reorg. Optimizer



* Reorganization happens in a separate background process and does not block queries

Key Challenge: working with a dynamic state space



MTS:

- Assume a fixed set S of candidate data layouts
- Competitive ratio $\sim \log(|S|)$

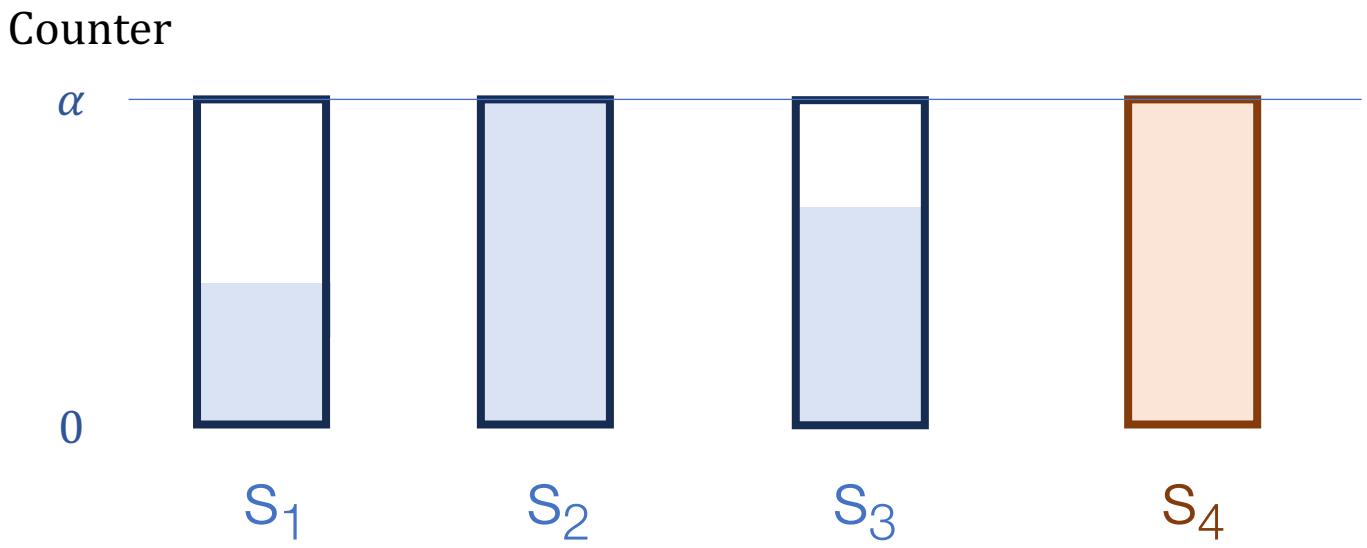
In practice:

- As new queries are processed, new layouts are generated on the fly

Our contribution: Introduce a dynamic variant of MTS (D-UMTS), and an algorithm that solves it with a tight competitive ratio $\sim \log(|S_{max}|)$

Reorganizer: how to switch?

The classic algorithm of Borodin, Linial, and Saks[1]

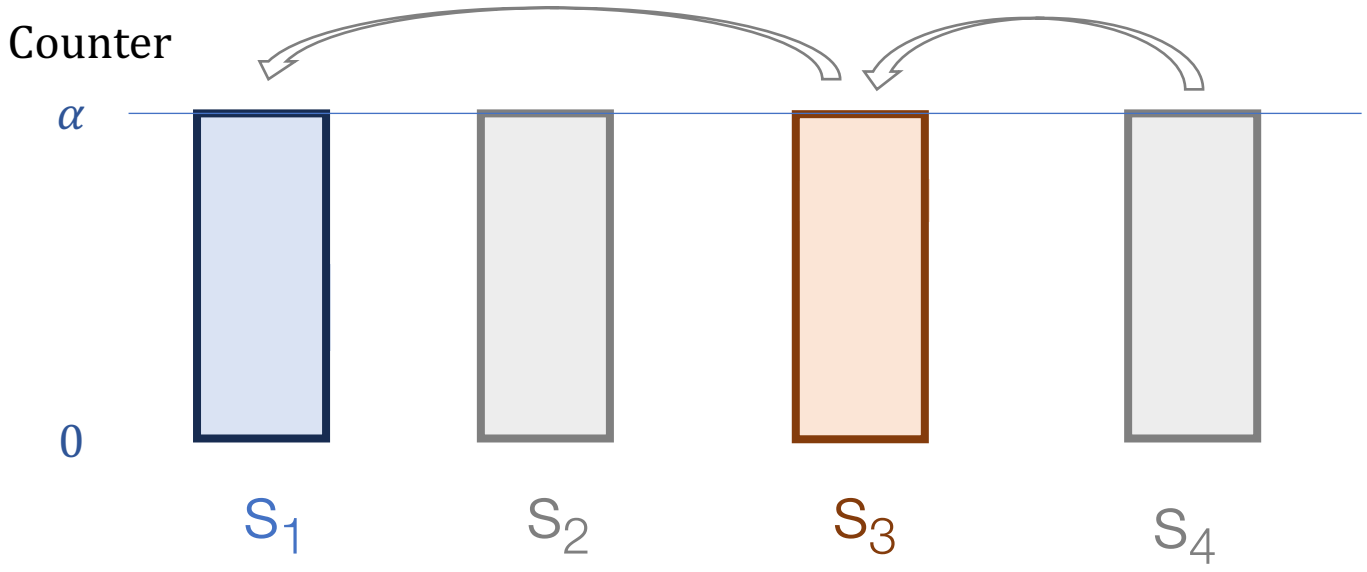


- For each query q: increase counter for S_i by $c(q, s_i)$
  - $c(q, s_i)$: % tables read
- When a counter is full ($>= \alpha$), *randomly* switch to a state whose counter is not full

[1] A. Borodin, N. Linial, and M. E. Saks, "An optimal on-line algorithm for metrical task system," J. ACM, vol. 39, no. 4, pp. 745–763, 1992.

Reorganizer: how to switch?

The classic algorithm of Borodin, Linial, and Saks[1]



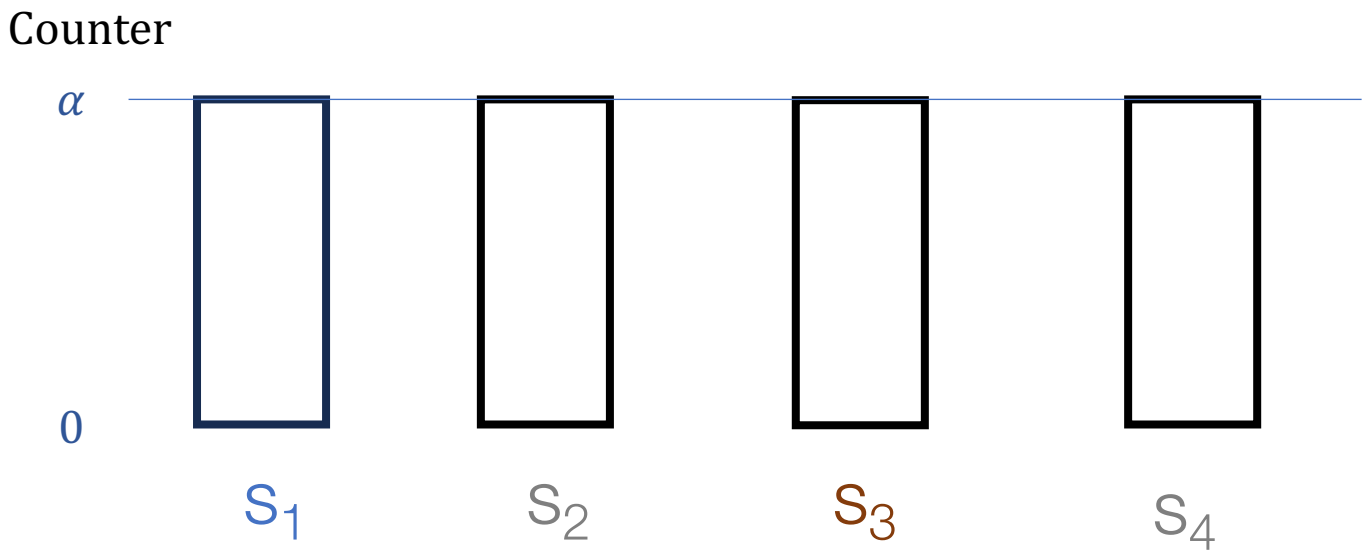
- For each query q: increase counter for S_i by $c(q, s_i)$
  - $c(q, s_i)$: % tables read
- When a counter is full ($>= \alpha$), *randomly* switch to a state whose counter is not full

A phase ends when all counters are full.

[1] A. Borodin, N. Linial, and M. E. Saks, "An optimal on-line algorithm for metrical task system," J. ACM, vol. 39, no. 4, pp. 745–763, 1992.

Reorganizer: how to switch?

The classic algorithm of Borodin, Linial, and Saks[1]



- For each query q: increase counter for S_i by $c(q, s_i)$
  - $c(q, s_i)$: % tables read
- When a counter is full ($>= \alpha$), *randomly* switch to a state whose counter is not full

A phase ends when all counters are full. Reset all counters to 0

[1] A. Borodin, N. Linial, and M. E. Saks, "An optimal on-line algorithm for metrical task system," J. ACM, vol. 39, no. 4, pp. 745–763, 1992.

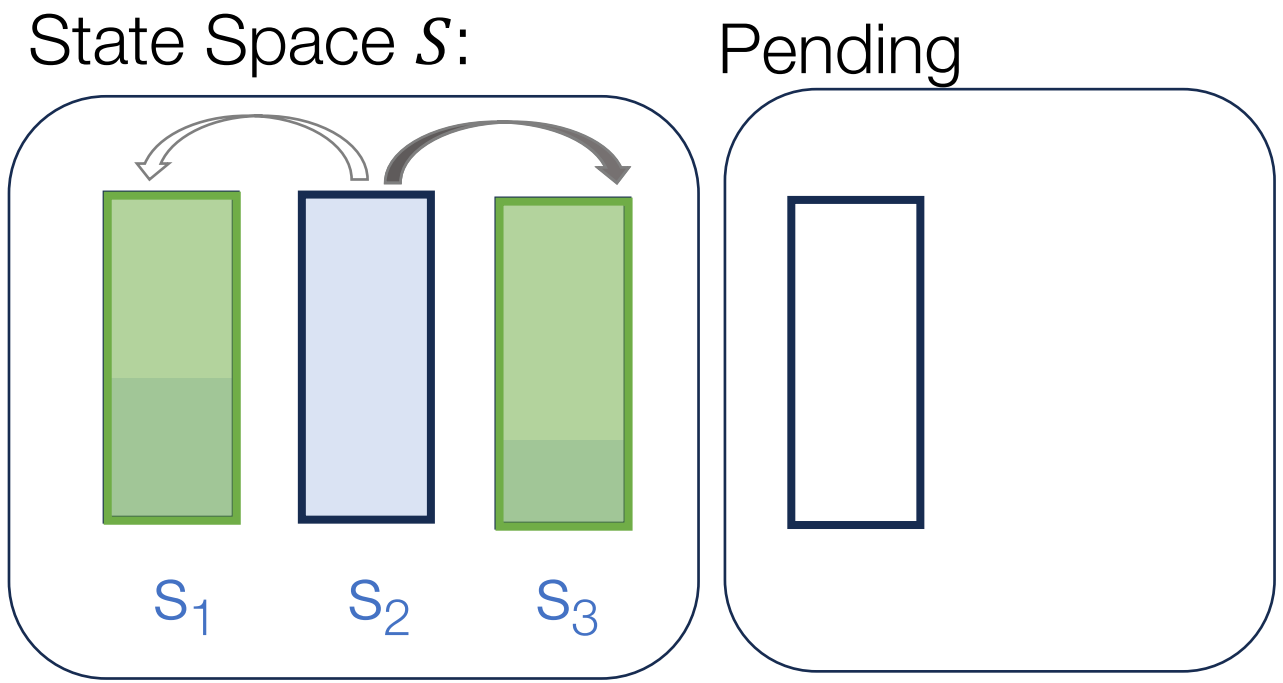
Reorganizer: adding dynamic state

Handling dynamic state space:

- Insert: delay until next phase
- Delete: set counter to full

Other enhancements

- Data-informed transition distribution
- Additional storage budget for hosting multiple data partitions



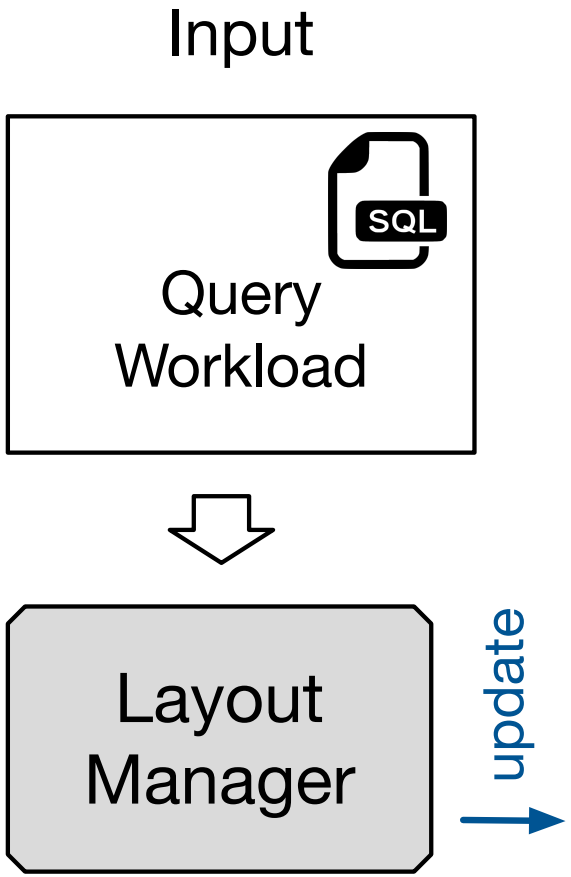
Layout Manager: which layouts to use?

generate_layout(D, Q, k):

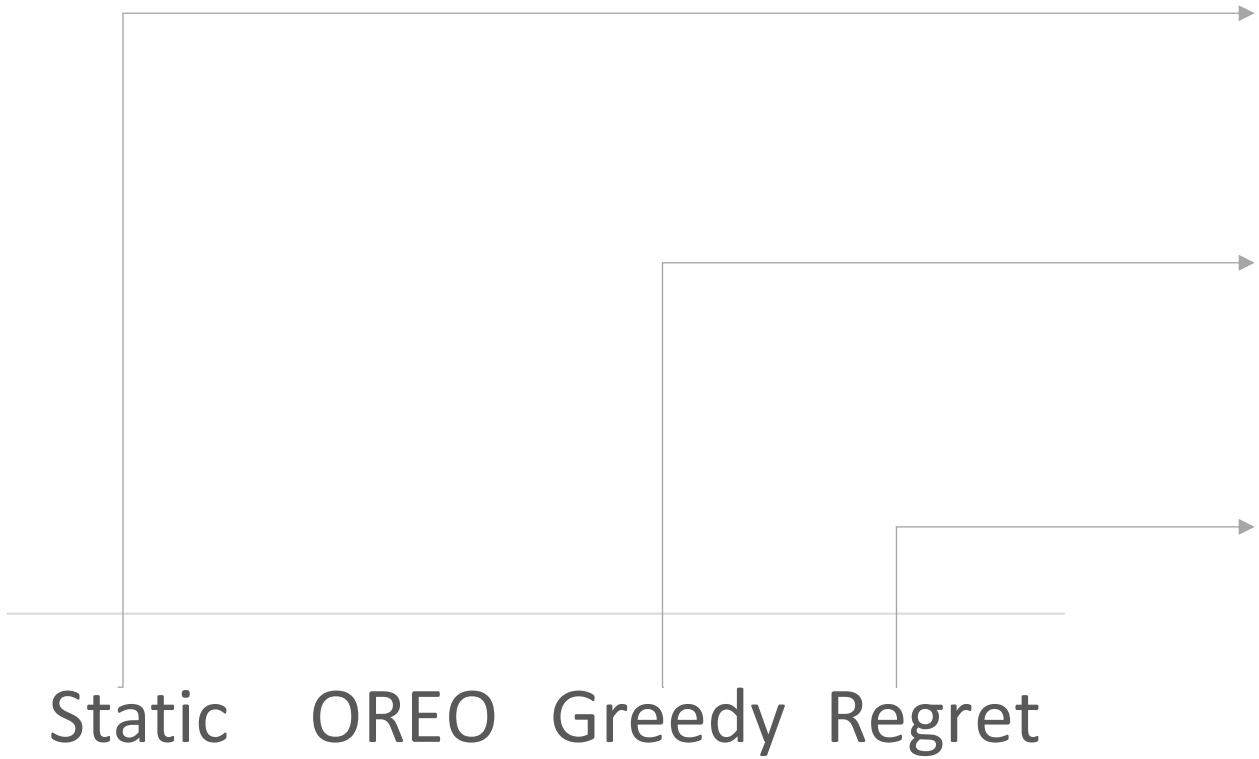
- D: a *sample* of datasets (~1%)
- Q: a sliding window of recent queries
- k: # total partitions

Selectively admit new layouts

- competitive ratio $\sim \log(|S_{max}|)$
- Only admit a new layout to the state space if it's different enough from all existing ones



Evaluation: End-to-end Time in Spark



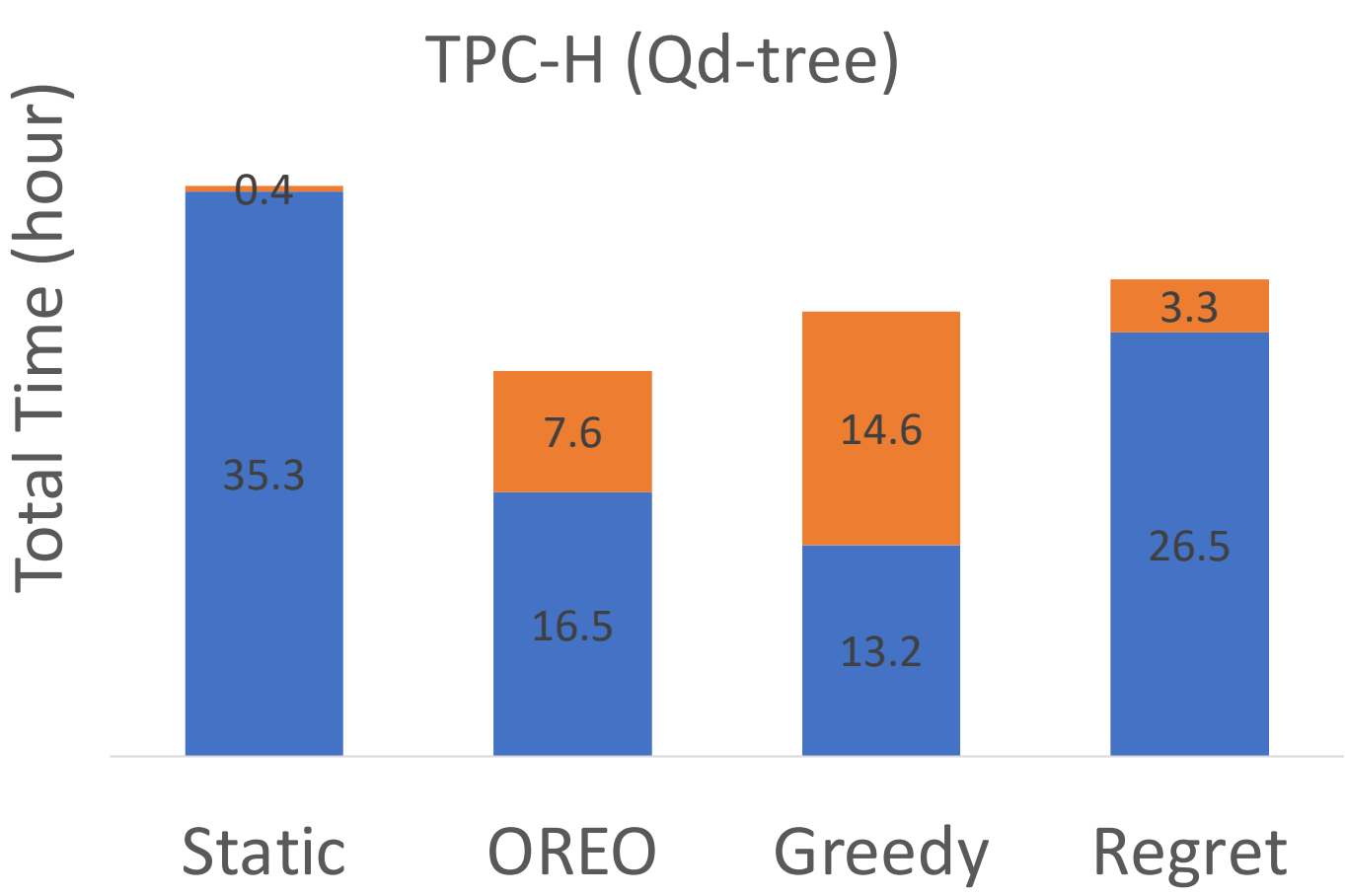
Best offline layout for the entire query workload
(large query cost, zero reorg cost)

Switch to new layout if better
(small query cost, large reorg cost)

Switch to new layout if cumulative regret > reorganization cost [1]

[1] M. Daum et al., “Tasm: A tile-based storage manager for video analytics, ICDE 2021

Evaluation: End-to-end Time in Spark



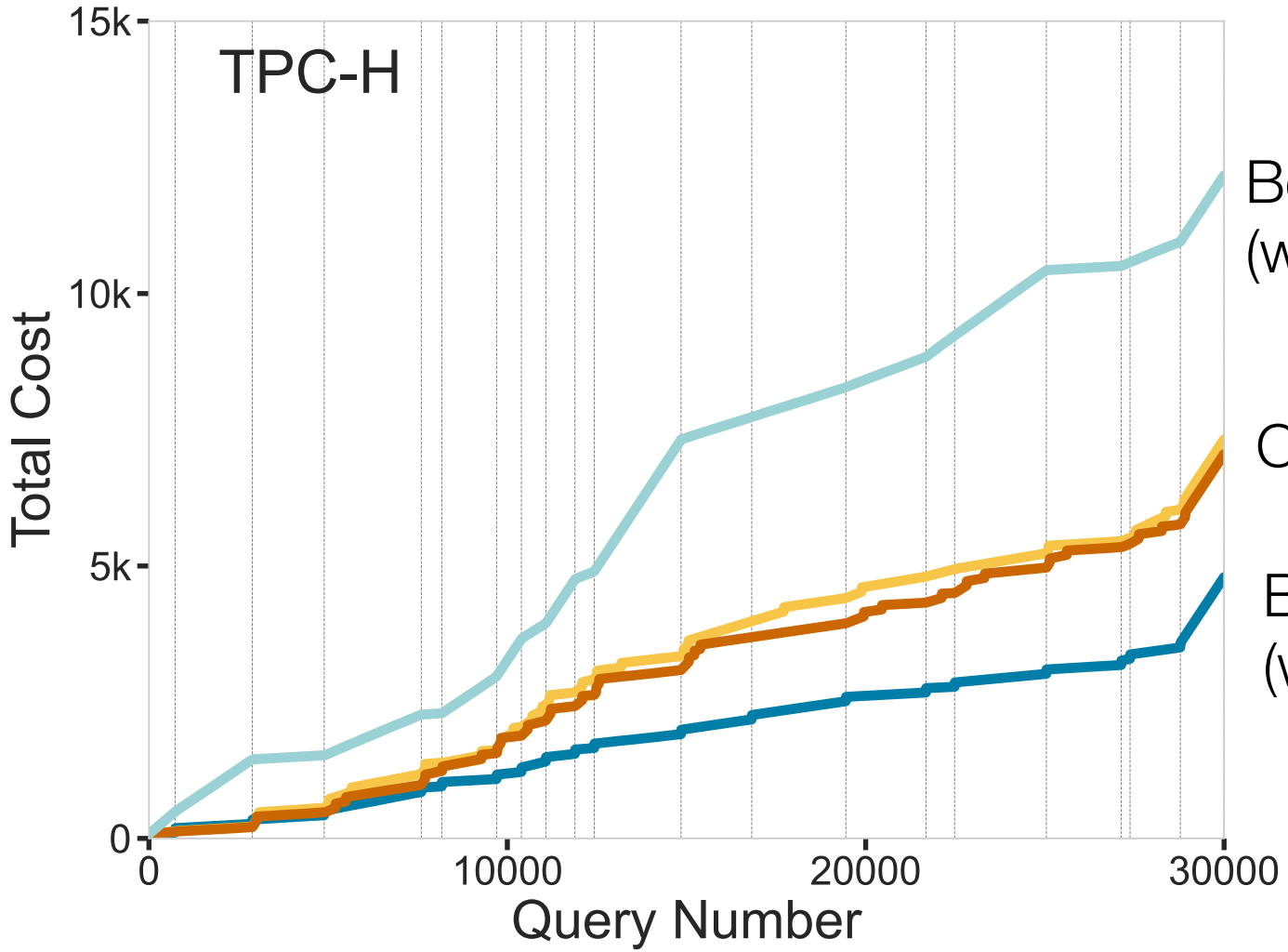
Dataset
- TPC-H* (sf=100)

Workload
- 30k queries
- 20 templates

Metric:
- query + reorganization time

Up to 32% improvement over a static data layout optimized for the entire workload

Evaluation: Gap to Offline Optimal



Best Static Layout
(with knowledge of entire workload)

OREO / OREO + optimal state space

42%

Best Dynamic Layout
(with knowledge of entire workload)

Up to 42% to be improved upon in future work.

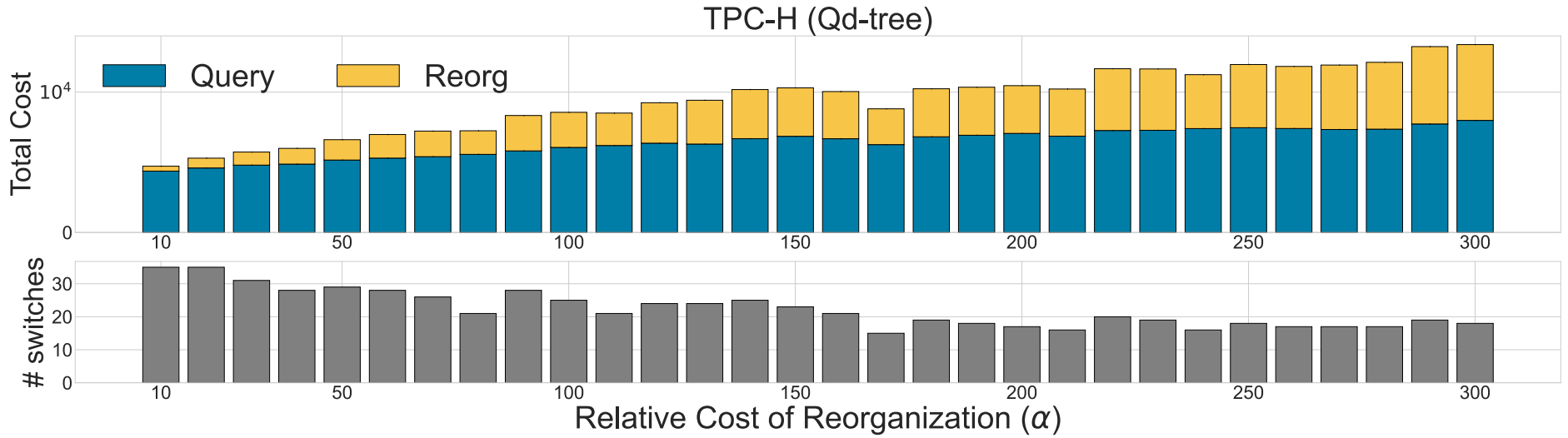
More experiments in the paper

Additional workloads:

- TPC-DS
- Production workload from VMWare
- Telemetry

Detailed analysis

- Effect of reorganization cost α
- Sliding window vs reservoir sampling



Conclusion and Future Works

OREO

- MTS with dynamic state space
- Up to 32% improvement in combined query and reorganization time compared to using a single, optimized data layout for the entire workload

Assumptions and limitations

- Static dataset
- Uniform switching cost – analysis of our algorithm assumes state switching is constant cost (Uniform MTS)

Future work

- Non-uniform MTS

Code: https://github.com/d2i-lab/oreo

Thank you! Questions?

Dynamic Data Layout Optimization with Worst-case Guarantees

Kexin Rong[1,3], Paul Liu[2,4], Sarah Ashok Sonje[1], Moses Charikar[2]

Georgia Tech[1], Stanford University[2], VMware Research[3], Vatic Labs[4]

Code: https://github.com/d2i-lab/oreo