Instance-Optimized Data Layouts for Cloud Analytics Workloads

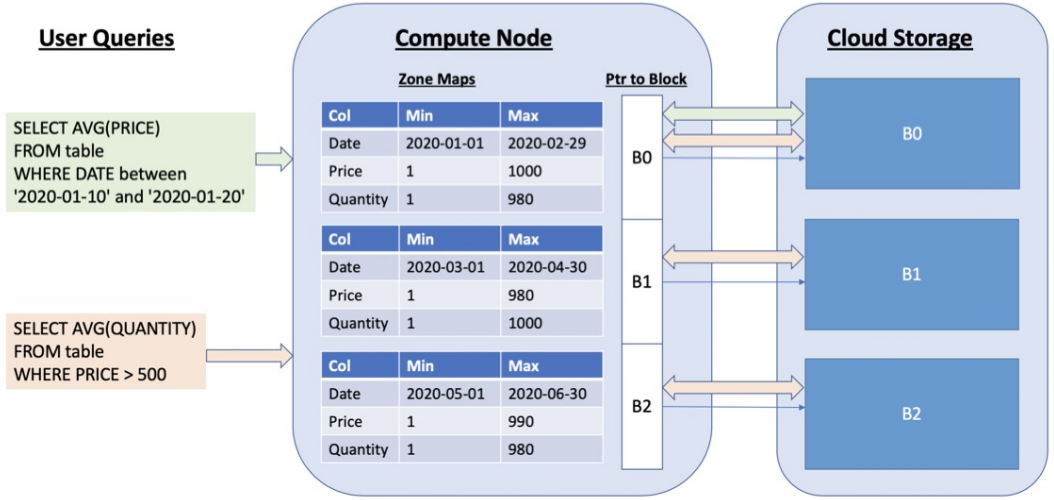
Authors: Jialin Ding, Umar Farooq Minhas, Badrish Chandramouli,
Chi Wang, Yinan Li, Ying Li, Donald Kossmann, Johannes Gehrke, Tim Kraska

Presented By:

Soumen Sahoo

INTRODUCTION

- Modern cloud servers rely on techniques to reduce data movement and access.
- Per-block metadata, which is often cached in memory, is used to avoid accessing blocks.



INTRODUCTION

- Zone maps, a form of per-block metadata, stores the minimum and maximum values.
- Zone maps have low maintenance costs and are also potentially useful.
- Z-order is a multi-dimensional sorting technique which uses the idea of sorting through multiple columns.
- Instance-optimized data layouts was defined as a measure to overcome the traditional data layout techniques that required to select columns manually.

INTRODUCTION

- Instance-optimized layouts learn a special blocking scheme that further improves the blocking speed.
- With the currently existing instance optimized layouts, it is only possible to optimize a single table.
- But analytics workloads contain many tables to deal with which makes the layouts inefficient.
- So, to be able to optimize multiple tables, MTO (Multi-Table Optimizer) was introduced which is a framework that can be used to optimize the whole datasets.

Current Blocking Scheme

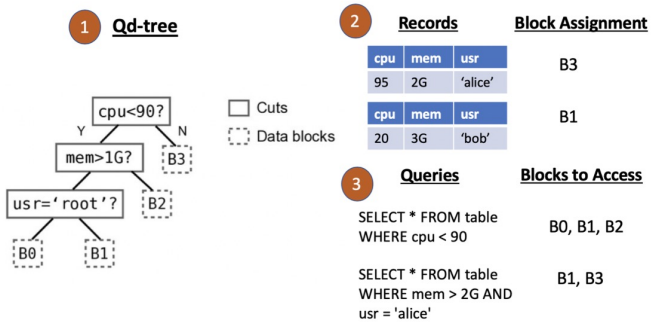


Figure 2: (1) Qd-tree defines blocks using cuts. (2) Qd-tree is used *offline* to route records to the blocks they are stored in and (3) is used *online* to determine which blocks need to be accessed during query execution.

Q-d Tree

- Existing Instance-Optimized data layout designed for single tables.
- Initially designed to modify the block assignment strategy for a given query workload.
- Q-d tree works as follows:

(i) I/P is a table and workload of queries which is used to construct a decision tree.

(ii) Use the q-d tree to assign table to blocks.

(iii) During the time of query execution, use the q-d tree to determine which blocks are to be used by the query to access.

OVERVIEW OF MTO

- MTO (Multi Table Optimizer) creates instance optimized data layouts that can be used to optimize multi-table datasets.
- Goal of the optimizer is to learn the instance-optimized layouts that can help to maximize the block skipping.
-

MTO OVERVIEW – SIDEWAYS INFORMATION PASSING

- The currently existing single table layouts do not use SIP (Sideways Information Passing)
- As per the figure, table A will have elements already present in the block.

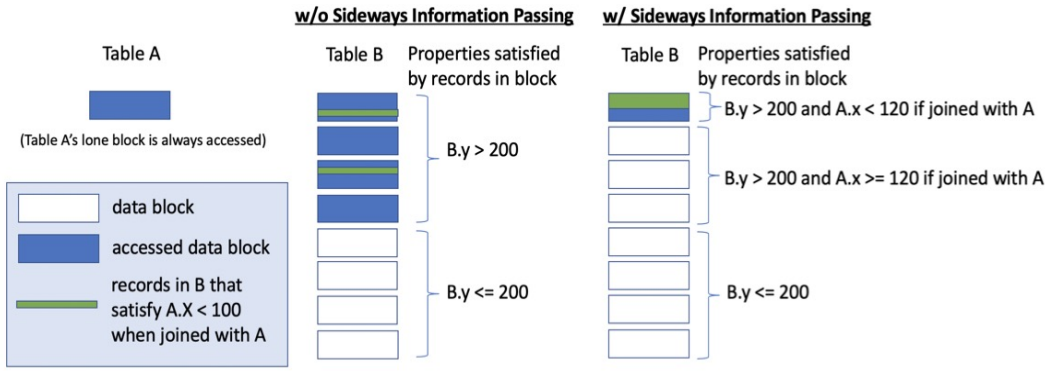


Figure 3: By taking advantage of sideways information passing to optimize the data layout, we can increase block skipping. Only blocks in the shaded regions are read.

MTO OVERVIEW – SIDEWAYS INFORMATION PASSING

- Table 1 shows the different terms that are used for the joint-induced predicate as shown.

Table 1: Join-induced predicate terminology example.

Term	Definition	Running Example
Simple predicate	Predicate over one table	`A.X < 100`
Join-induced predicate/cut	Predicate over columns in multiple tables, composed of nested semi-join subqueries	`A.BKEY IN (SELECT B.BKEY FROM B WHERE B.CKEY IN ( SELECT C.CKEY FROM C WHERE C.Z > 200))`
Literal cut	Result of evaluating subqueries in a join-induced cut	`A.BKEY IN (3, 14, 159)`
Source table	Table with the original predicate	`C`
Target table	Table whose predicate is induced	`A`
Source cut	Source table's predicate	`C.Z > 200`
Induction path	List of tables and join columns connecting source to target	$C \rightarrow_{CKEY} B \rightarrow_{BKEY} A$
Induction depth	Length of the induction path	2

MTO WORKFLOW

- The workflow, as shown in the figure, has 2 optimizations (Offline optimization and Online query execution)
- In the offline optimization, MTO takes multi-table dataset and query workload as input and uses it to generate one q-d tree per table.

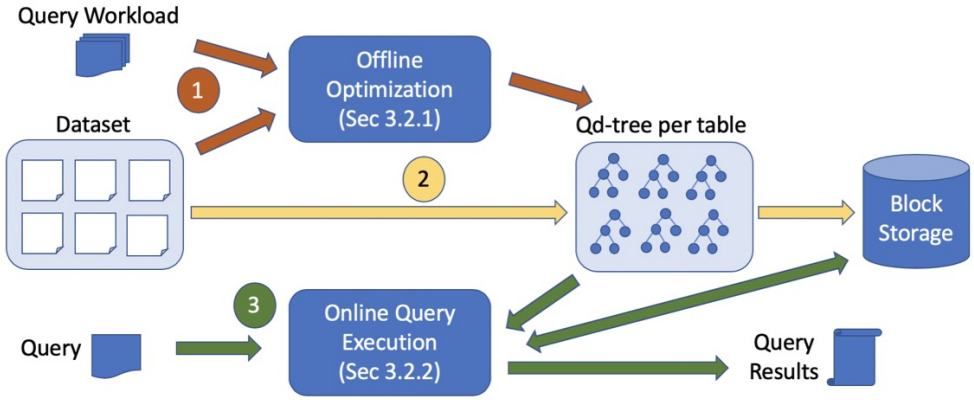


Figure 4: (1) In offline optimization, MTO produces a layout (a qd-tree per table) given a dataset and query workload. (2) MTO assigns records to blocks and stores them. (3) In online query execution, MTO skips blocks based on the layout.

MTO WORKFLOW – OFFLINE OPTIMIZATION

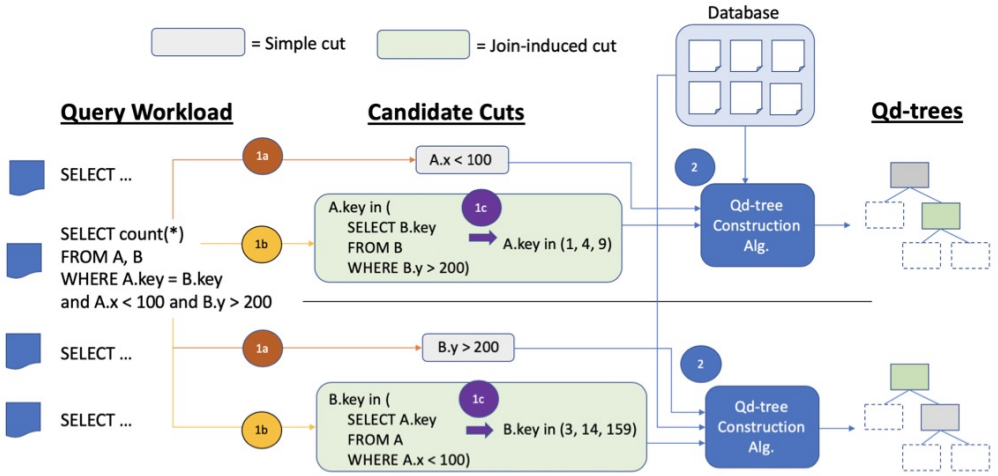


Figure 5: MTO optimization uses the query workload and dataset to create one qd-tree per table.

MTO WORKFLOW – ONLINE QUERY EXECUTION

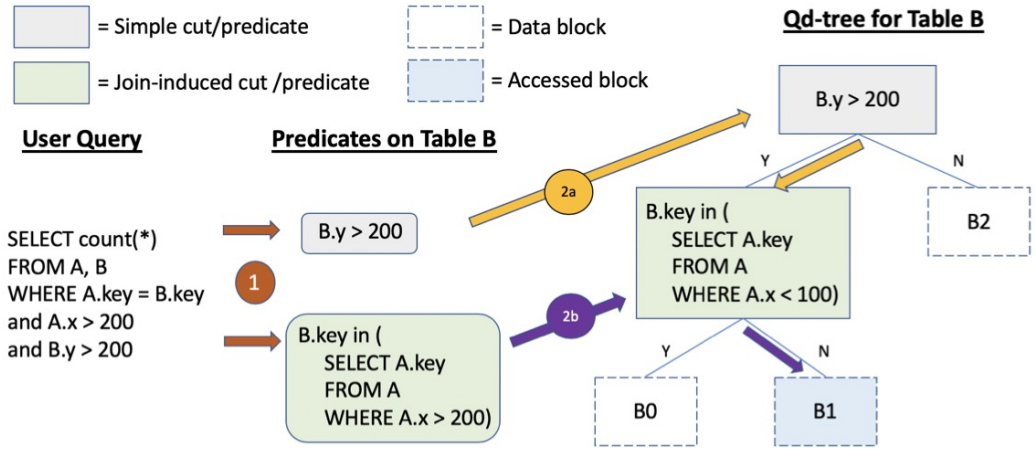


Figure 6: At query time, MTO uses the per-table qd-trees to determine which blocks to access from each table. This query only needs to read block 1 from Table B.

MTO ALGORITHMS – JOIN INDUCED PREDICATES

```
SELECT AVG(A.Z) FROM A WHERE A.X < 100 AND A.Y < (
    SELECT COUNT(*) FROM B WHERE A.KEY = B.KEY AND B.Z > 200)
```

MTO ALGORITHMS – JOIN INDUCED PREDICATES

Some rules to be followed to decide which predicates can be induced:

- Predicates can be induced in both the directions to induce by using inner joins.
- In order to perform self join, MTO creates a copy of the table which makes two different sets and then uses it to
- As per the mentioned query, the predicate A.X<100 induces into the remainder of the query by using the inner join A.KEY = B.KEY.
- Like simple cuts which are present in q-d tree, join induced cuts use the idea of navigating the records and queries in the tree.

SCALABILITY

- MTO optimizes by using uniform samples of the given dataset rather than relying on the dataset itself in order to reduce the time required for optimizing the layouts.
- In order to optimize, MTO will create a sample from fraction s(sampling rate) of the available records for each table at random in a uniform way.
- MTO attaches a value called Cardinality Adjustment (CA) to join induced cuts defined by pow(s,d).

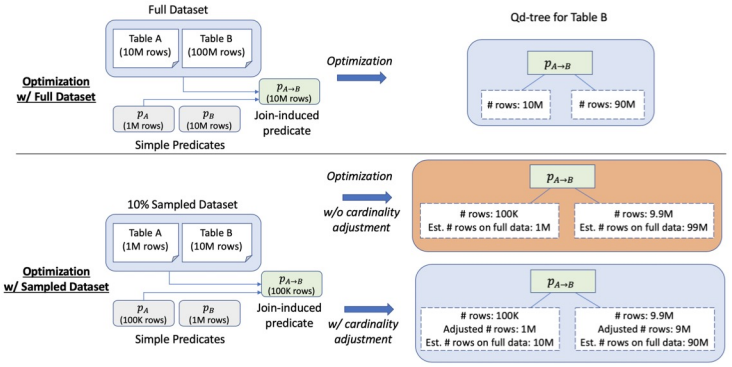


Figure 7: Cardinality adjustment allows MTO to achieve accurate block size estimates when optimizing based on a dataset sample, which improves the quality of the resulting layout.

Dynamic Workloads

	Frac. Data Reorganized	Re-opt. Time (min)	Frac. Subtrees Considered in Re-opt.
$q = 100$	0	0	0
$q = 200$	0.370	9.81	0.031
$q = 500$	0.841	25.0	0.163
$q = 1000$	0.893	17.3	0.084
$q = \infty$	1.0	2.48	N/A

Table 5: MTO behavior after workload shift.

EVALUATION

- On commercial-cloud based analytics services, MTO achieves an accuracy of 93% in reducing the accessing of blocks and there is also 75% reduction in query time.
- MTO is successful in achieving lower optimized time which results in improved end-to-end performance.
- Evaluation was done on 3 datasets, Star Schema Benchmark(SSB), TPC-H, TPC-DS with 60GB for SSB and 100 GB for TPC-H and TPC-DS.
-

EVALUATION

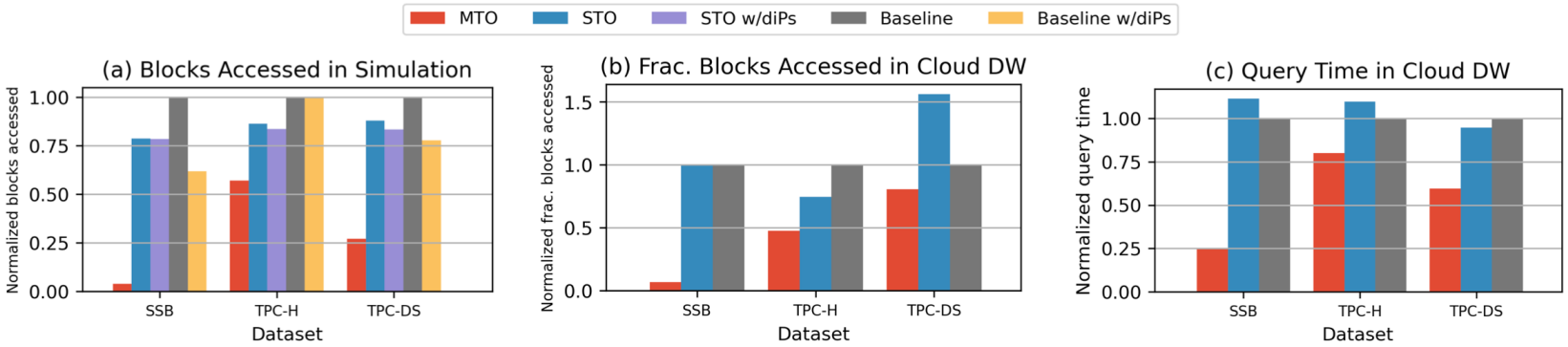


Figure 10: MTO achieves better overall workload performance than alternatives across datasets and metrics. Note that the y-axes are normalized to the metric achieved by Baseline.

EVALUATION

- MTO was compared with two approaches, (i) Baseline, that sorts the tables based on columns that are user tuned, (ii) STO, which is an instance optimized layout approach that uses MTO algorithms but, without join-induced predicates.
- MTO was evaluated based on certain metrics which are:

i. number of blocks which were accessed (approx. 500K records)

ii. fraction of the blocks that were accessed.

iii. End-to-end query runtime on Cloud DW.

EVALUATION

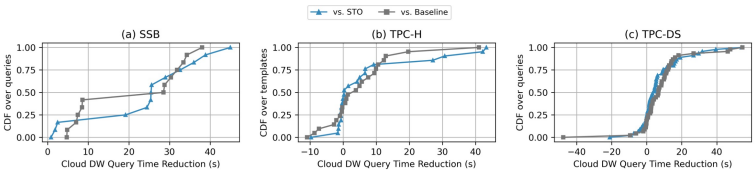


Figure 11: Reduction in query runtimes achieved by MTO, relative to STO and Baseline. Different queries achieve different performance gains.

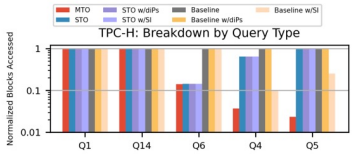


Figure 12: MTO has the most performance advantage over STO and Baseline on queries with selective filters over joined tables, like Q4 and Q5.

	SSB	TPC-H	TPC-DS
MTO			
Optimization time (min)	0.195	3.67	0.619
Data sample rate used for opt.	0.01	0.03	0.01
Routing time (min)	1.54	5.80	3.50
Total offline time (min)	1.73	9.47	4.12
STO			
Optimization time (min)	0.0213	0.697	0.0611
Data sample rate used for opt.	0.003	0.0003	0.01
Routing time (min)	0.360	0.978	0.771
Total offline time (min)	0.381	1.68	0.832

Table 3: Offline optimization times for Fig. 10.

CONCLUSION

- The dominating costs for query processing in cloud-based data analytics is the I/O that is required for accessing large data blocks.
- Per-block technique, as mentioned in the previous slides, is the common technique for reducing I/O by block skipping but, their effectiveness is dependent on how records are assigned.

QUESTIONS

Q-1: Can you describe the cardinality adjustment (CA) that MTO use to get an accurate estimate of block size for the entire dataset? (Atharva)

Q-2: When dealing with increased workloads and rising data volumes, how does MTO compare to other approaches such as STO and Baseline? (Atharva)

Q-3: Is this system used in the real world? (Tanusri)

QUESTIONS

- Q-4: How did the author evaluate the performance of this system and what metrics did they use? (Shreya)
- Q-5: Do you think MTO could have utilized these different information passing types to improve disk I/O?(Uzochi)
- Q-6: Could MTO take advantage of machine learning in any aspect? If so, where do you think a machine learning model would apply, and why? (Uzochi)