Self-Organizing Data Containers

Thomas Glas

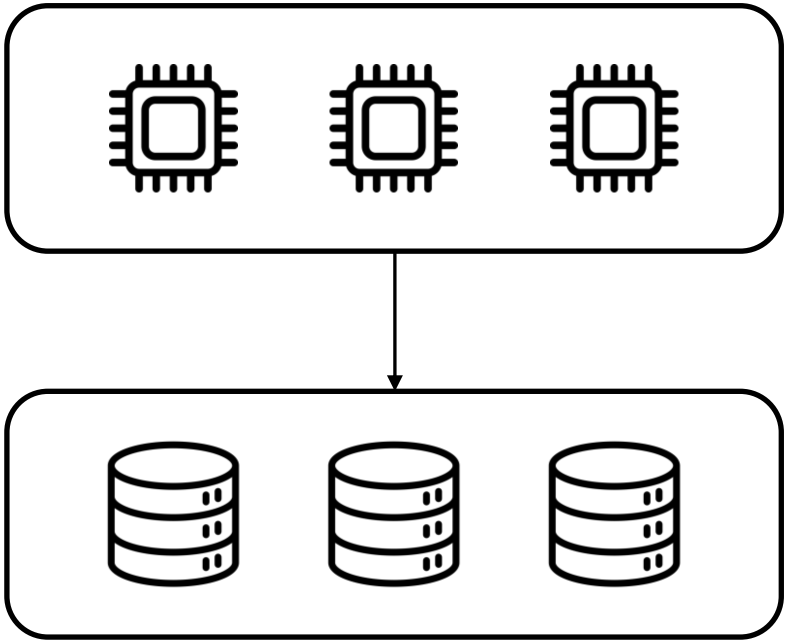
Technische Universität München

TUM School of Computation, Information and Technology

Lehrstuhl für Datenbanksystem

München, 14. März 2023

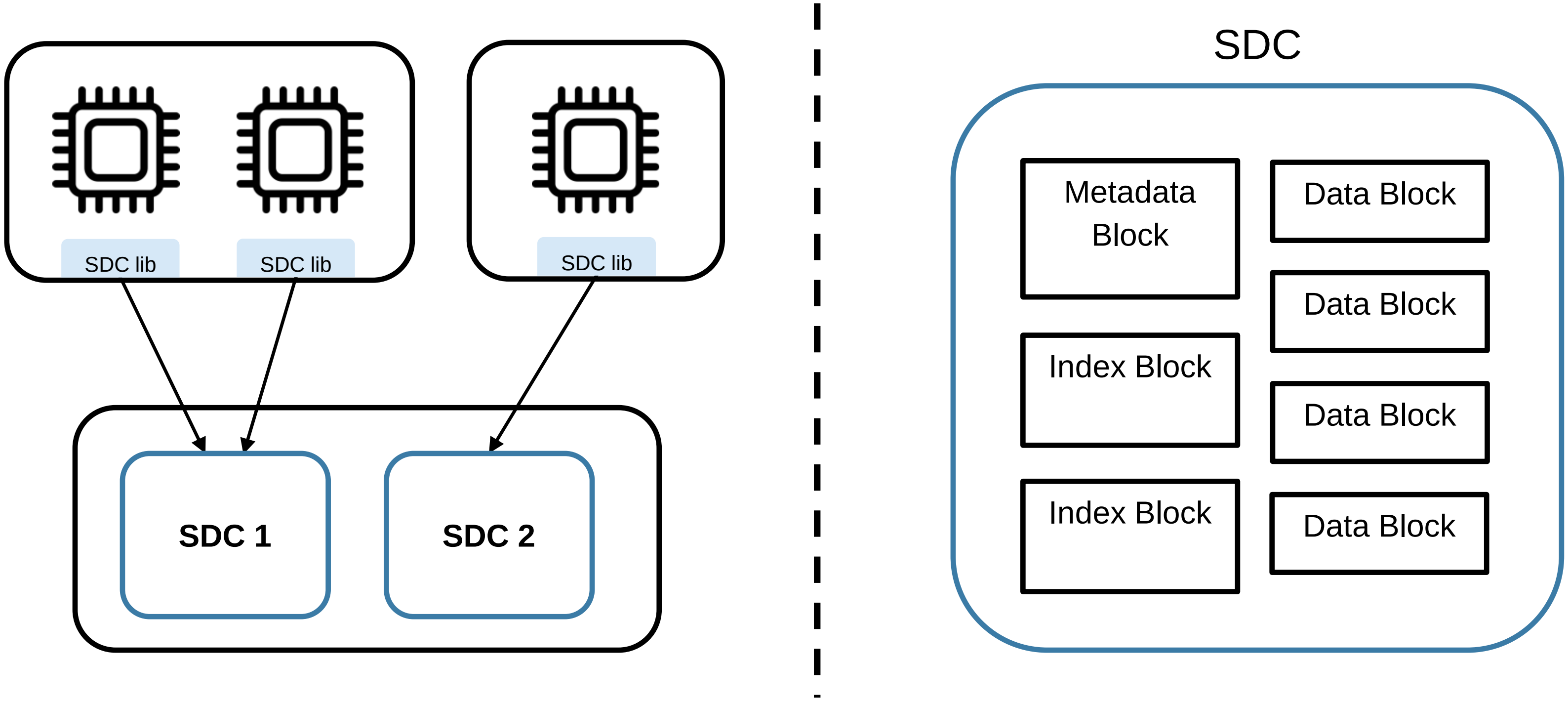
Motivation



How can we minimize data transfer?
➡ only retrieve data needed for query

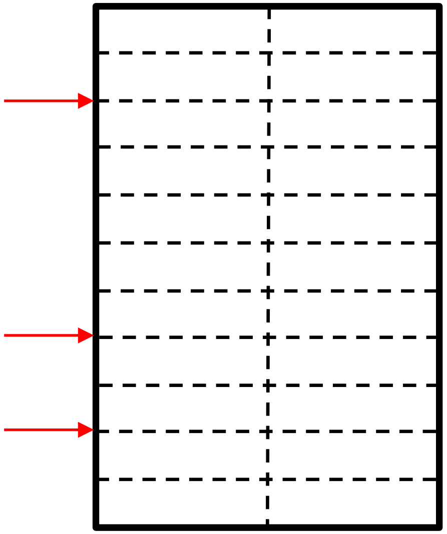
How can we create a storage layer with rich metadata to support this?

Self-Organizing Data Containers

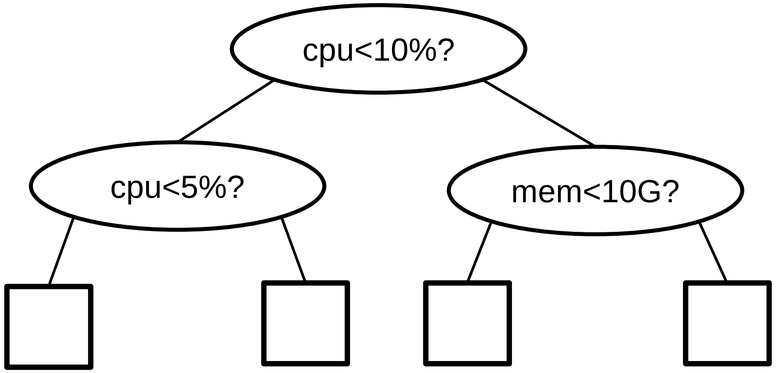


Partitioning Strategies

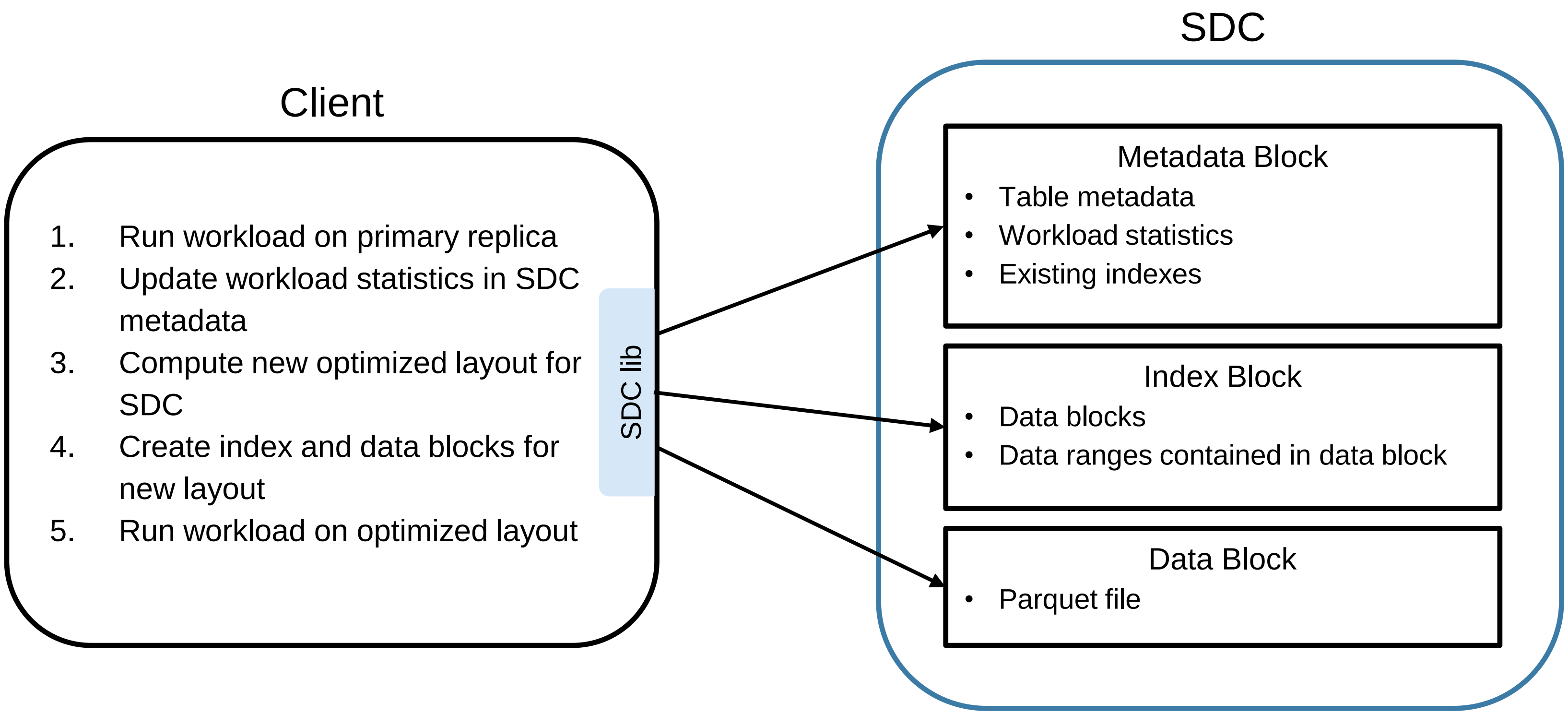
Column Range Partitioning



Qd-Tree



Self-Organizing Data Containers



Client

1. Run workload on primary replica
2. Update workload statistics in SDC metadata
3. Compute new optimized layout for SDC
4. Create index and data blocks for new layout
5. Run workload on optimized layout

SDC lib

SDC

Metadata Block

- Table metadata
- Workload statistics
- Existing indexes

Index Block

- Data blocks
- Data ranges contained in data block

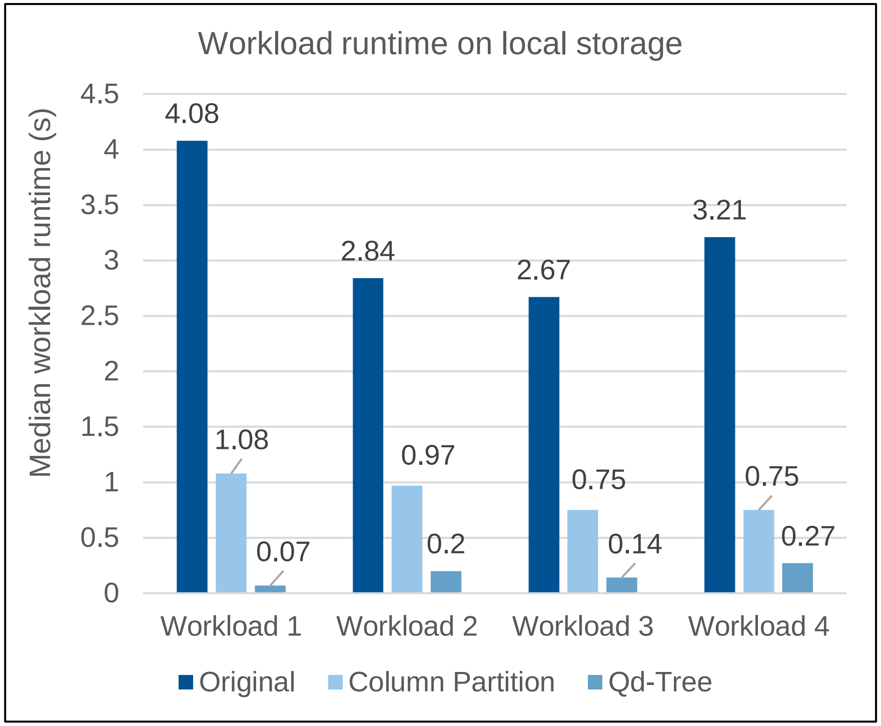
Data Block

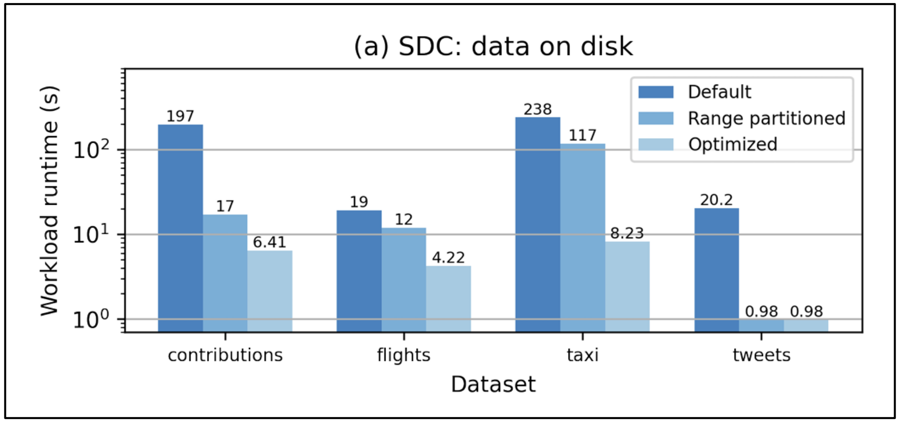
- Parquet file

Demo of my SDC prototype

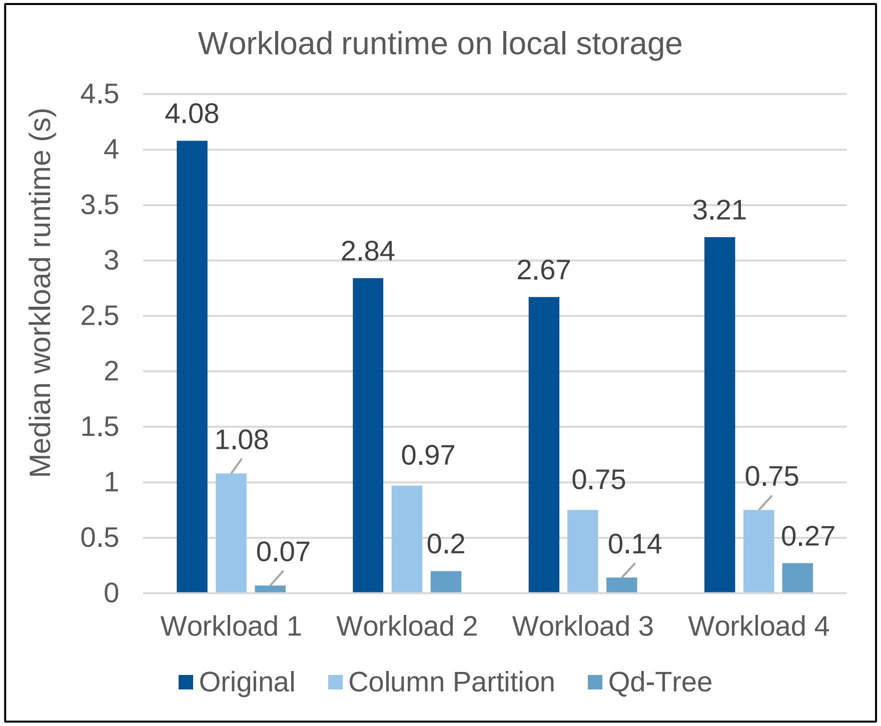
- SDC Library API: Projections and filters on tables
- Indexes: Primary, column range partitioning, Qd-Tree
- Dataset: NYC TLC Trip Record Data
- Workloads: Single-table range queries
- Storage layer: Local disk vs cloud storage (simulated)
- Data blocks: Apache Parquet files
- Metadata blocks: JSON files
- Clients: Single client

Benchmarks

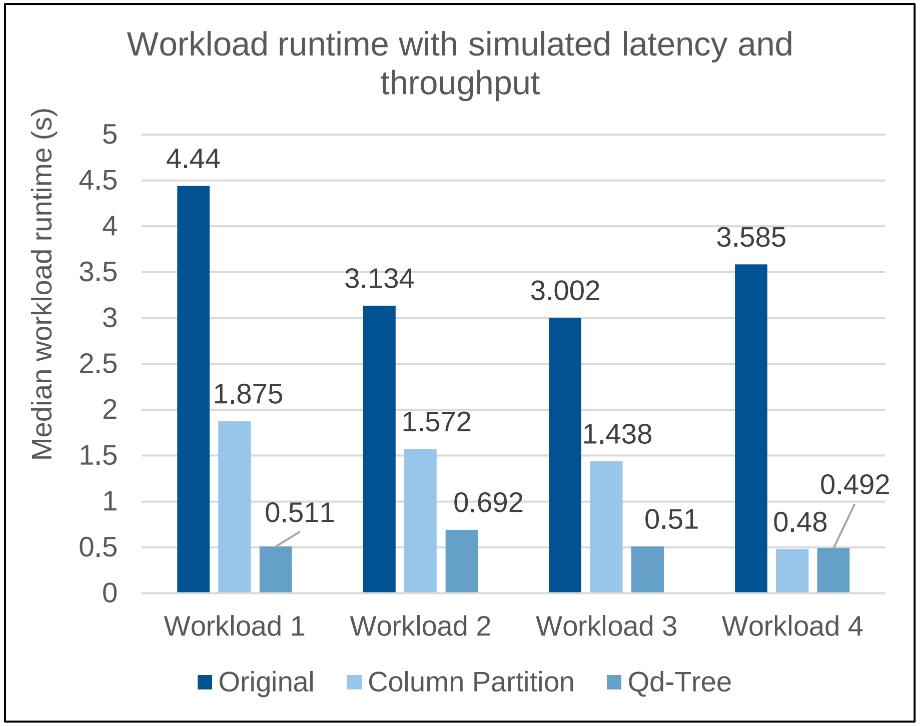

Workload runtime on local storage
Median workload runtime (s)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
4.08
1.08
0.07
2.84
0.97
0.2
2.67
0.75
0.14
3.21
0.75
0.27
Workload 1
Workload 2
Workload 3
Workload 4
■ Original
■ Column Partition
■ Qd-Tree


(a) SDC: data on disk
Workload runtime (s)
10^2
10^1
10^0
197
17
6.41
19
12
4.22
238
117
8.23
20.2
0.98
0.98
Default
Range partitioned
Optimized
contributions
flights
taxi
tweets
Dataset

Benchmarks



Workload runtime on local storage
Median workload runtime (s)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
4.08
1.08
0.07
2.84
0.97
0.2
2.67
0.75
0.14
3.21
0.75
0.27
Workload 1
Workload 2
Workload 3
Workload 4
■ Original
■ Column Partition
■ Qd-Tree



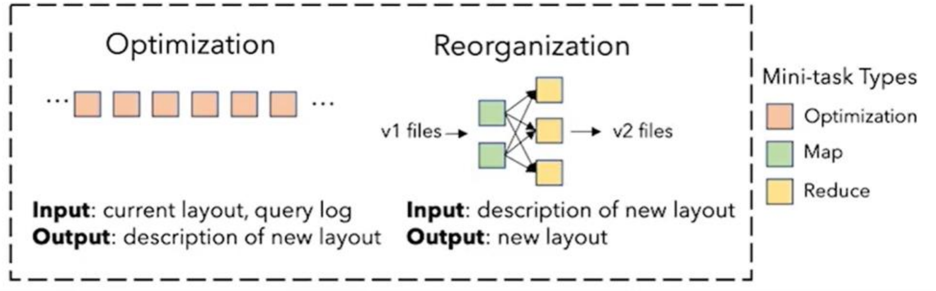
Workload runtime with simulated latency and throughput
Median workload runtime (s)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
4.44
1.875
0.511
3.134
1.572
0.692
3.002
1.438
0.51
3.585
0.48
0.492
Workload 1
Workload 2
Workload 3
Workload 4
■ Original
■ Column Partition
■ Qd-Tree

Conclusion & further research

- Simple yet effective self-learned storage optimization
- Easy to add indexes
- Easy to integrate into any type of applications (not just DBMS)
- Use of data replication: trading off storage cost for query performance

Further research

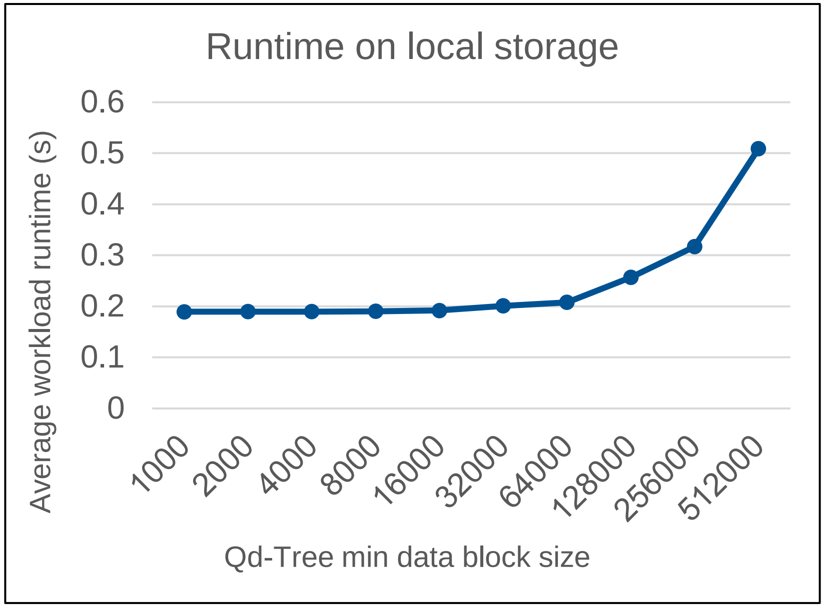
- Distributing optimization work among clients
- Find query clusters in workload for effective indexes

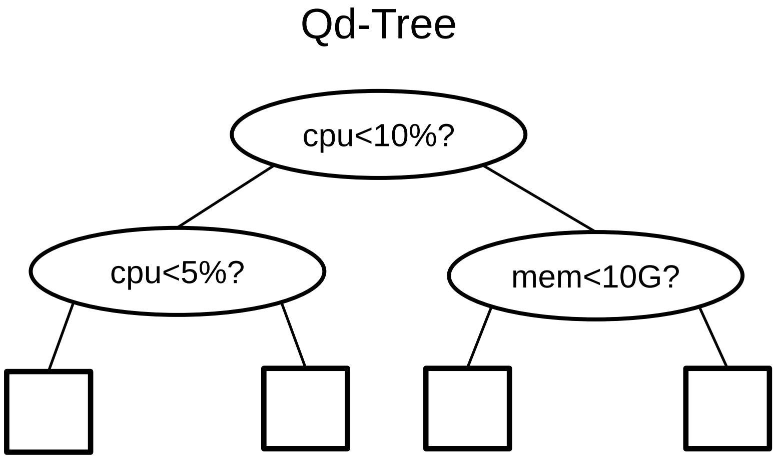

Optimization
Reorganization
v1 files
v2 files
Input: current layout, query log
Output: description of new layout
Input: description of new layout
Output: new layout
Mini-task Types
Optimization
Map
Reduce

Code on Github:

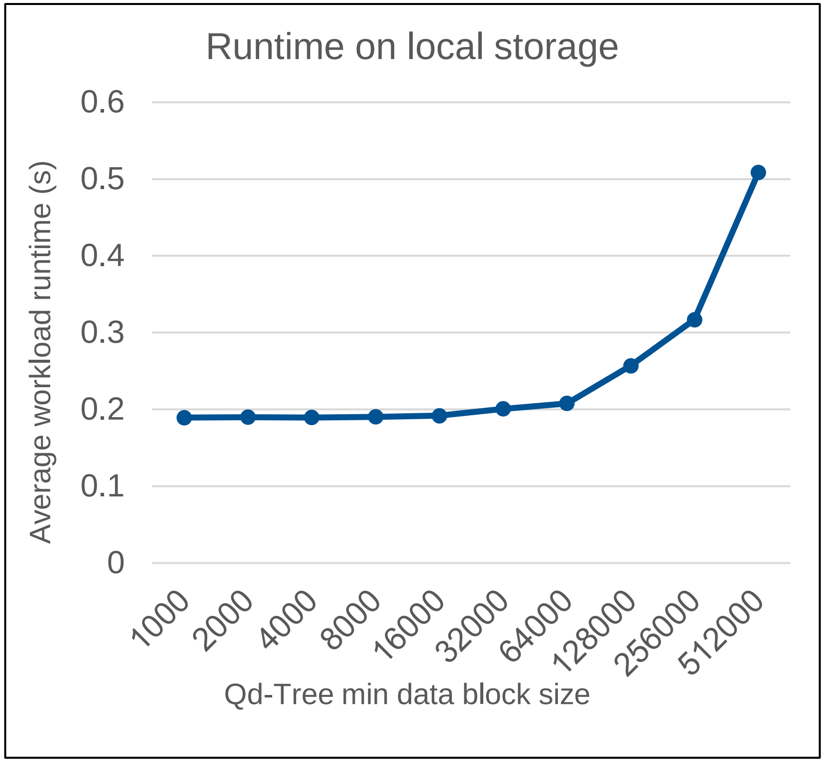
https://github.com/thomasglas/SDCs

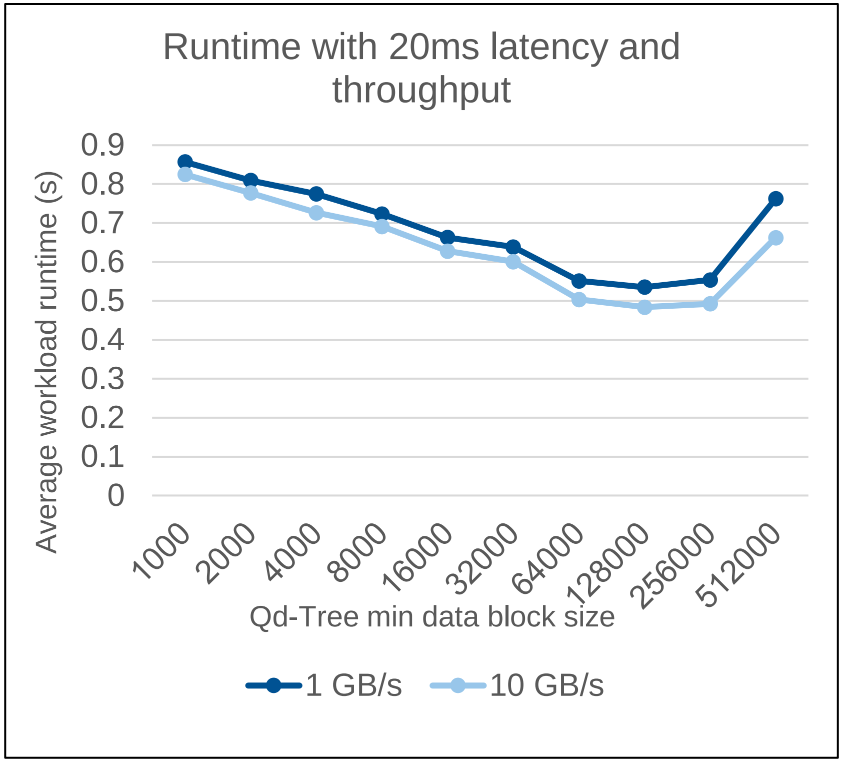
Benchmarking Qd-Tree min block sizes





Benchmarking Qd-Tree min block sizes


Runtime on local storage
Average workload runtime (s)
Qd-Tree min data block size


Runtime with 20ms latency and throughput
Average workload runtime (s)
Qd-Tree min data block size
1 GB/s
10 GB/s