Self-Designing LSM Memory Buffer

Alexander Ott James Chen Arpita Saha Subhadeep Sarkar

Brandeis University

lab

LSM Design Space

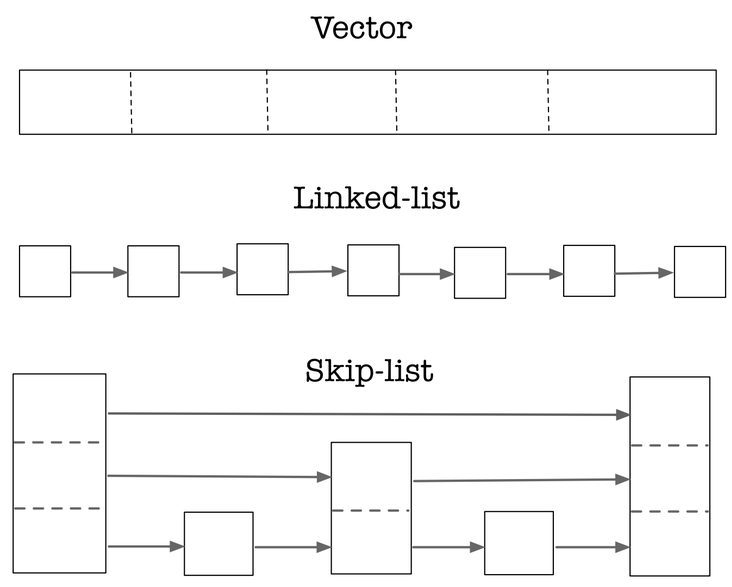
LSMs Everywhere

RocksDB, cassandra, influxdb, levelDB, SCYLLA, BigTable, mongoDB

Under changing workload, how do LSMs realize optimal memory buffer implementation ?

?

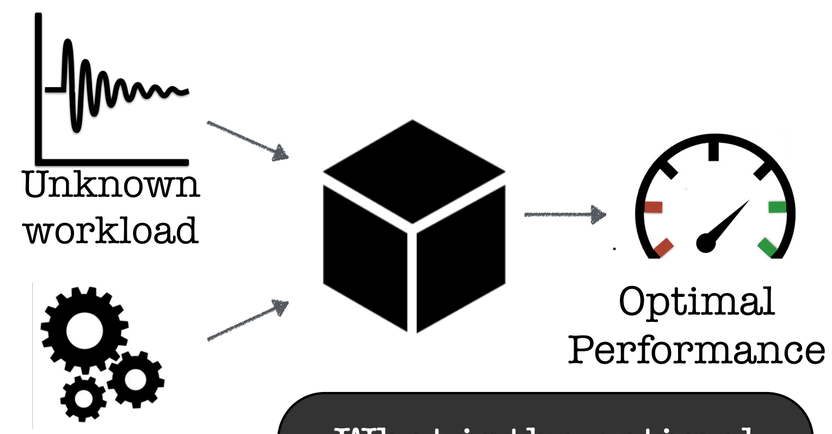
Memory Buffer Implementation



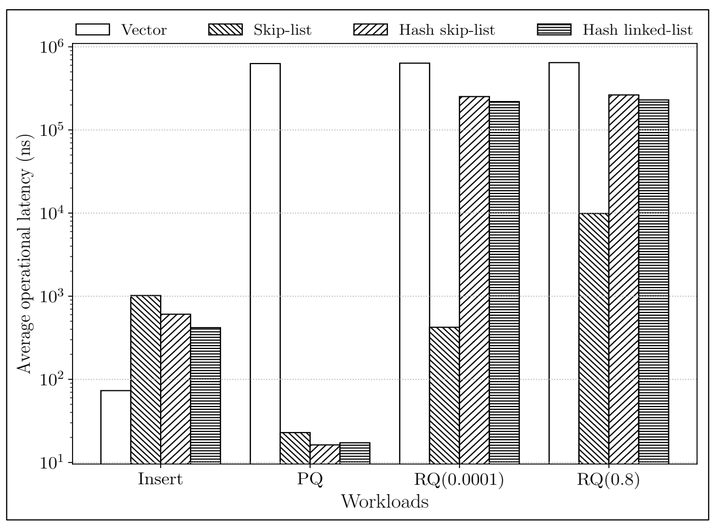
The choice of in memory buffer greatly affects the performance of LSMs.

Problem

Tuning LSMs



What is the optimal buffer configuration?



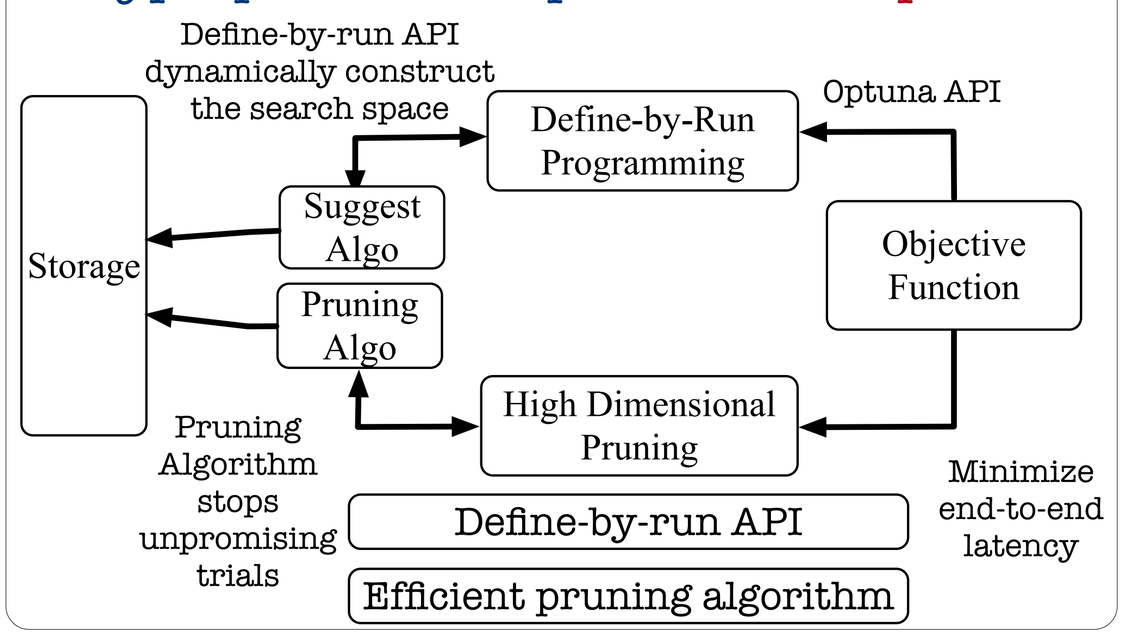
Performance varies widely with changing workloads and different buffer implementations

Design Challenges

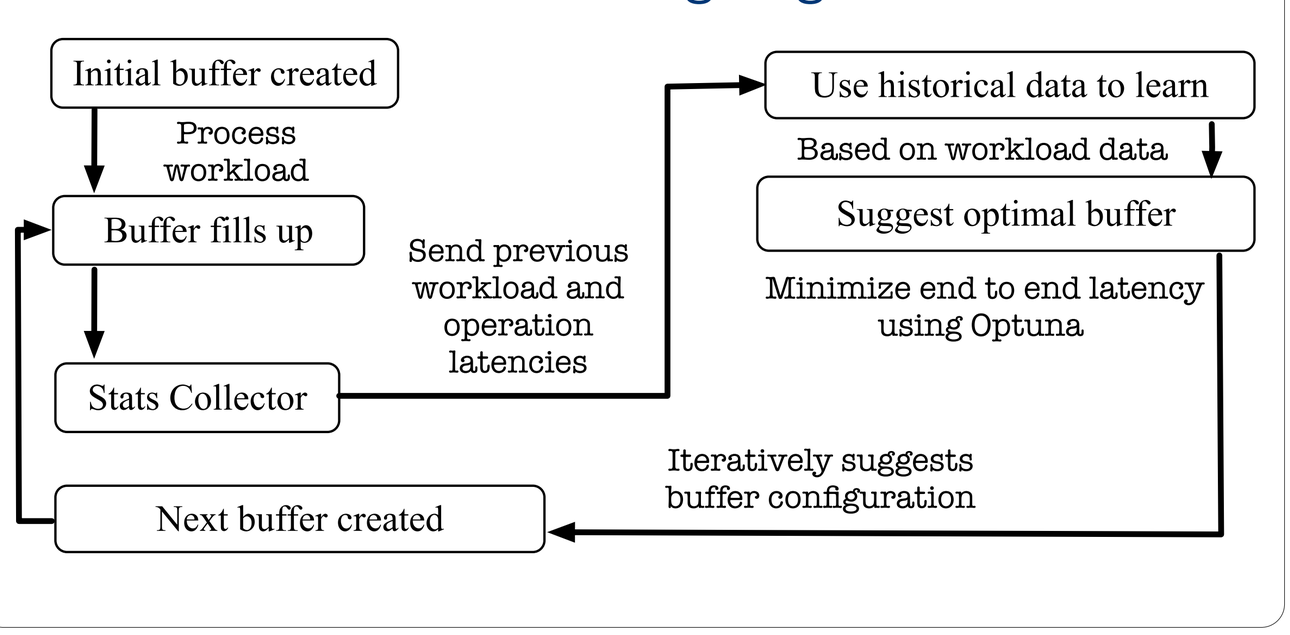
- Hand tuning not feasible
- Vast design space
- Hard to model interactions
- Unknown workloads
- Lack of benchmark dataset

Solution Approach

Hyperparameter Optimization: Optuna



Workflow of self-designing LSM buffer

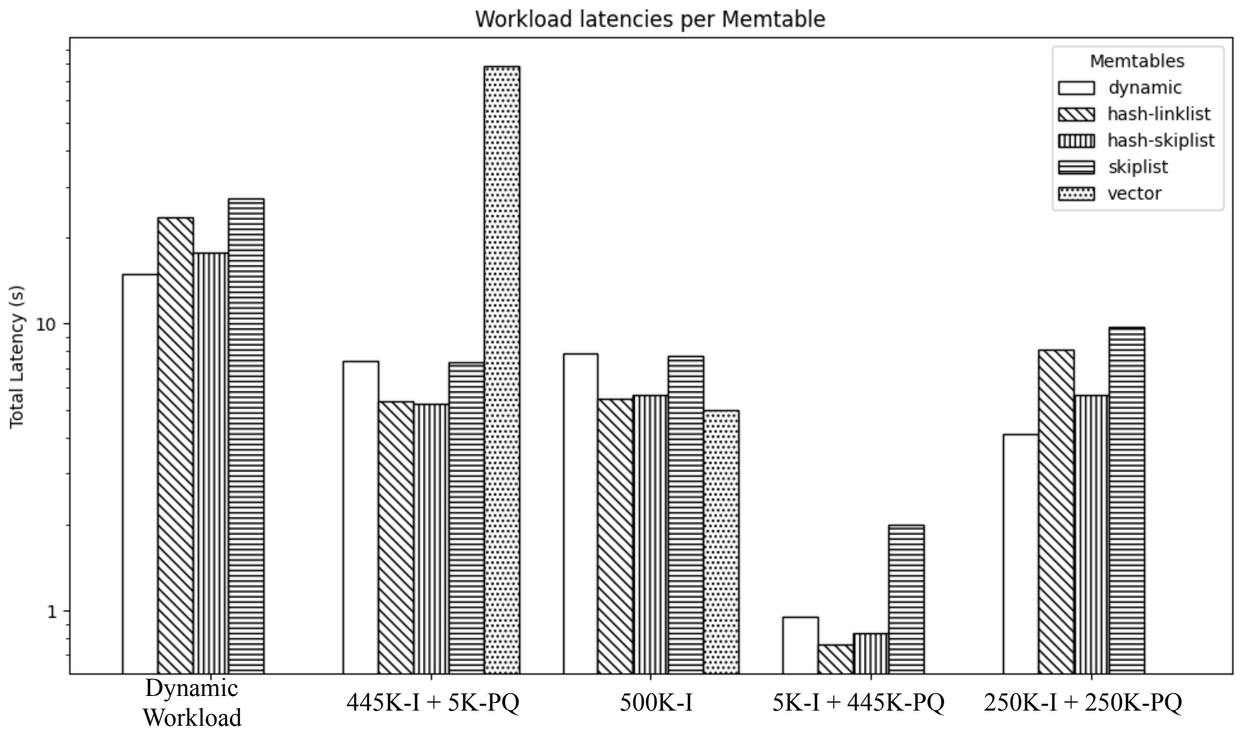


Evaluation

Workload Change

Reads: 5%
Writes: 90%
Updates: 5%

Point Reads: 90%
Writes: 5%
Updates: 5%



Buffer size: 16 GB
Entry size: 32 Byte uniformly random
Size ratio: 10

Dynamic workload composition:
5k inserts 445k point queries interleaved,
250k inserts 250k point queries interleaved,
445k inserts 5k point queries interleaved,
500k inserts.

For dynamic workloads, dynamic buffer performs optimally despite communication overhead

Brandeis University
Michtom School of Computer Science
lab