Workload-Driven Horizontal Partitioning

Lina Qiu, Zichen Zhu

Overview

- Proposed a workload-driven horizontal partitioning scheme

- Implemented two baselines
  - Equal-size k-means
  - Skipping-oriented partitioning [1]

[1] L. Sun, M. J. Franklin, S. Krishnan, and R. S. Xin. Fine-grained partitioning for aggressive data skipping. In SIGMOD, pages 1115–1126, 2014.

Focus & Assumptions

- Workloads consist of range queries
- The training workload used to do partitioning is consistent with the test workload
- Not support data updates

Workload-Driven Partitioning

What we know in advance:

1. The qualification of every data tuple for every query in the training workload
2. The selectivity of every query
3. The number of occurrences of every distinct query (freq)

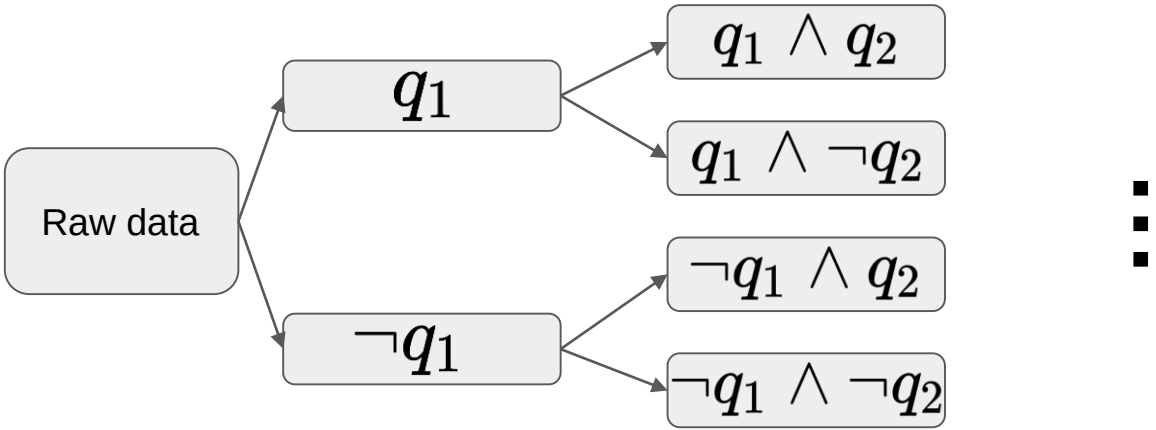
Workload-Driven Partitioning

Steps:

1. Sort queries by $f_q(1 - s_q)$

   We obtain a sequence of queries $q_1, q_2, \cdots, q_t, \cdots$

1. Do query-driven partition until the size of a partition is less or equal to a page



Equal-size k-Means

Desired cluster size: page_size

k = num_data / page_size

Difference:

Assign a data tuple to the closest cluster that hasn’t reached the max capacity

Break if max_iter is reached, or rss is below our specified threshold (set to 0)

Skipping-oriented Partitioning

1. Select top k most frequently queried filters

$$W = (f_{q_1}, f_{q_2}, .., f_{q_k})$$

1. Build a k-dimension bit vector for each tuple (0: unqualified, 1:qualified)

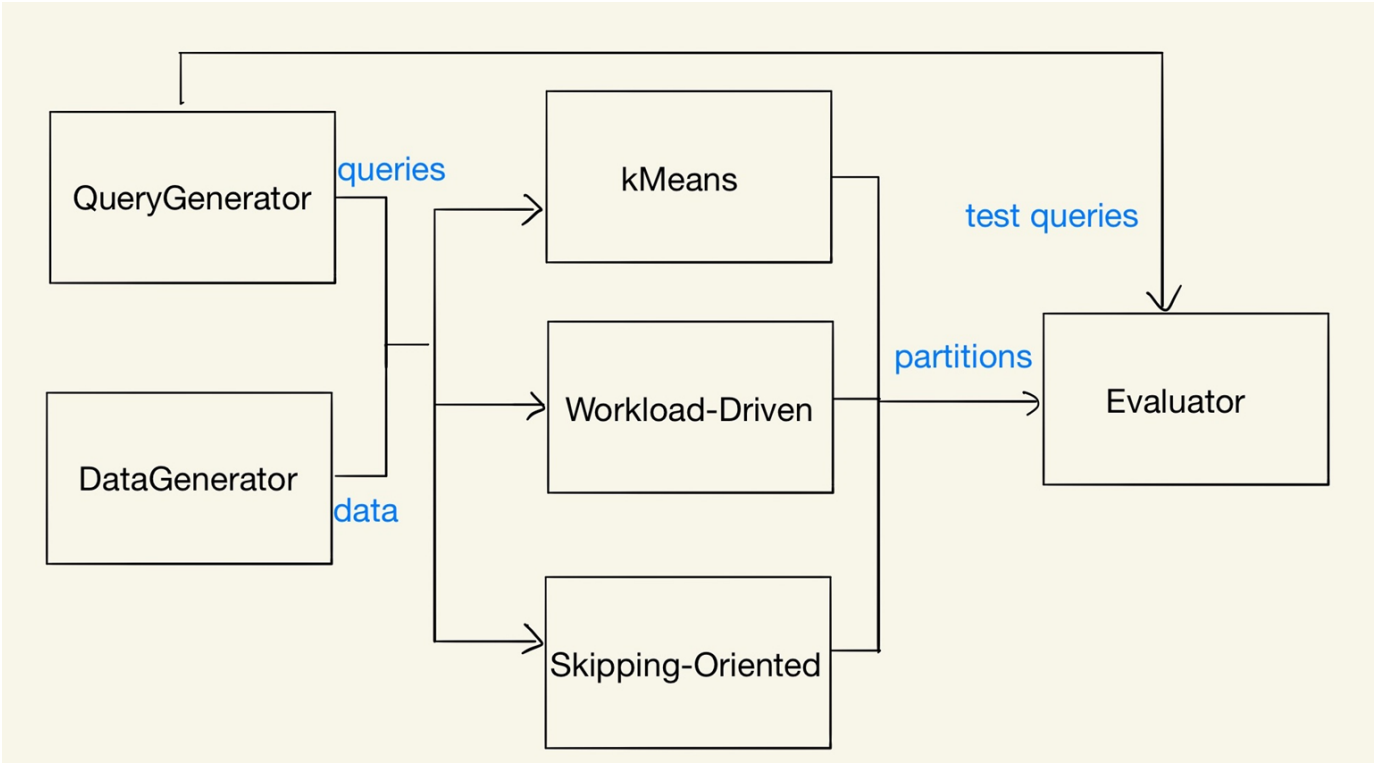
$$v_t = (0, 1, ..0)$$

1. Solve the optimization problem (approximate solution with Ward's method)

$$\arg\max_{\mathcal{P}} \sum_{P_i \in \mathcal{P}} |P_i| W (I - \bigcup_{t \in P_i} v_t)^T$$

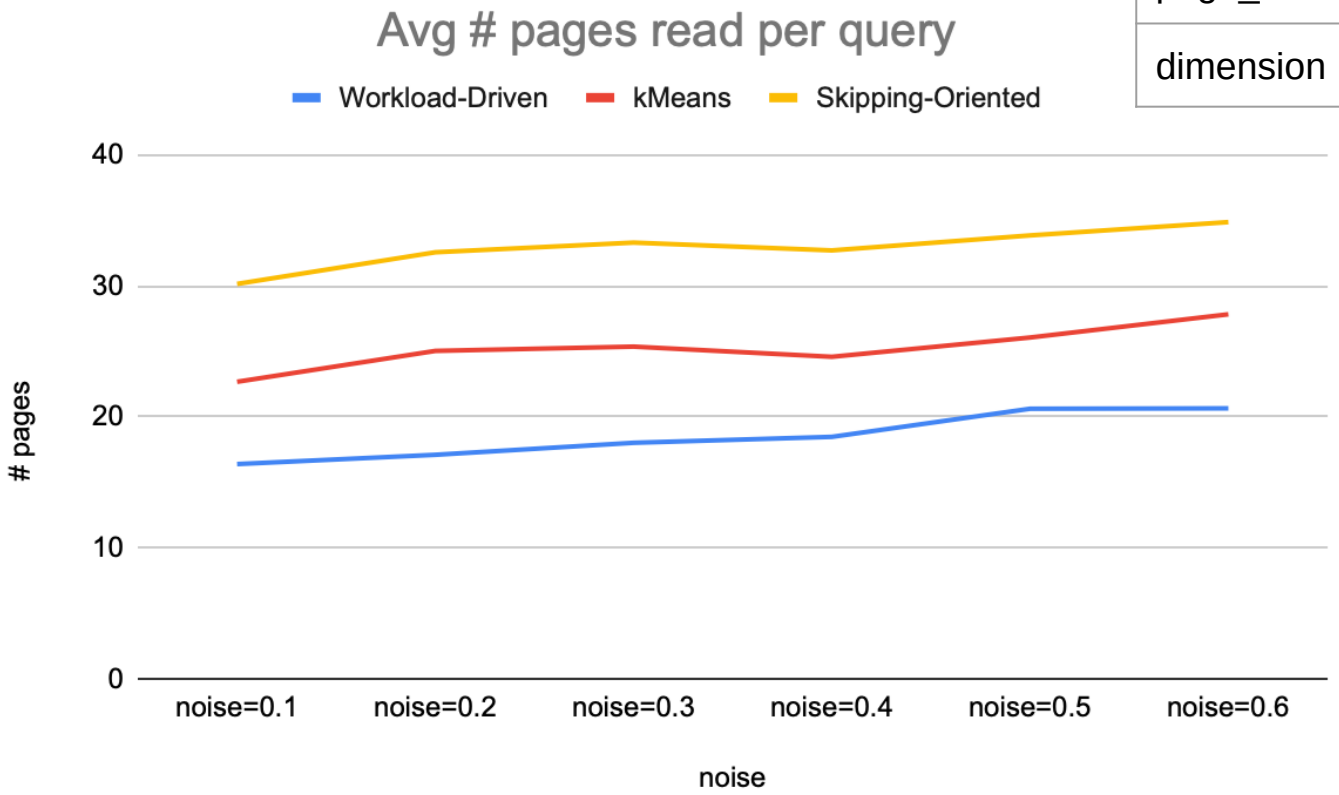
$$subject\ to\ |P_i| = p \ \ \forall P_i \in \mathcal{P}$$

Workflow


QueryGenerator
queries
kMeans
test queries
DataGenerator
data
Workload-Driven
partitions
Evaluator
Skipping-Oriented

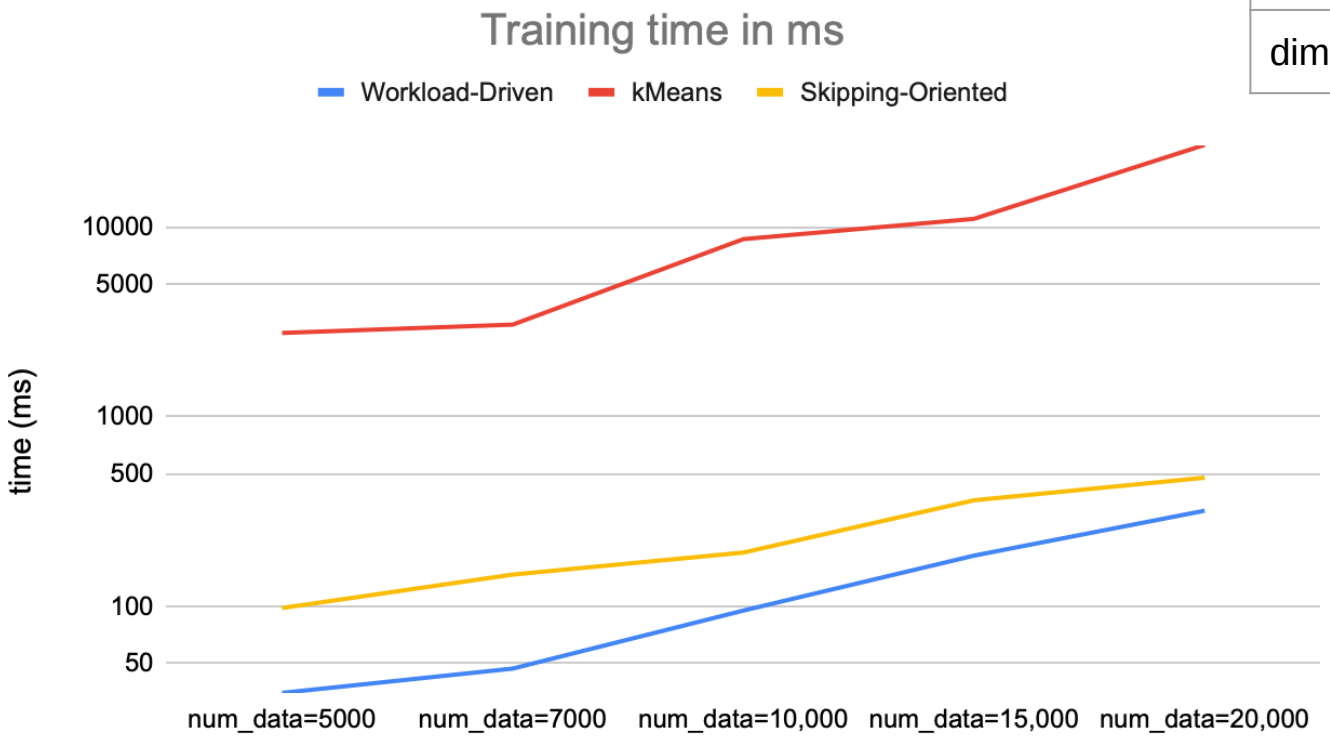
Evaluation

#data	10,000
#query	400
page_size	250
dimension	2 (uni, uni)



Evaluation

#data	vary
#query	400
page_size	250
dimension	2 (uni, uni)



Evaluation

#data	10,000
#query	vary
page_size	250
dimension	2 (uni, uni)

Future Work

- Take raw data similarity into consideration
- Build index for filling data to reduce extra memory footprint
- Tile-based partition
- How to support update/When should we do repartition