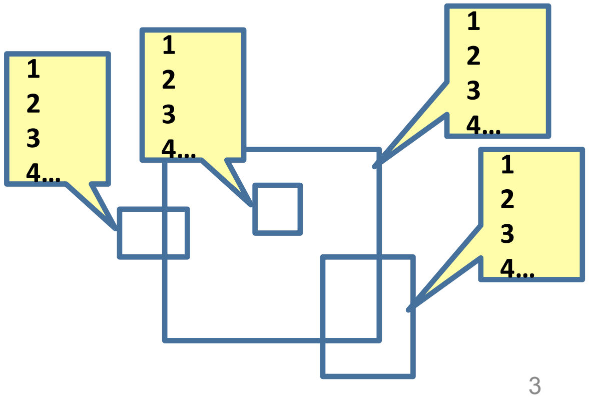
KTV Tree: Interactive Top-K Aggregation on Large Dataset in Cloud

Yuzhe Tang, Ling Liu, Junichi Tatemura, Hakan Hacigumus

Introduction on Top-K Aggregation

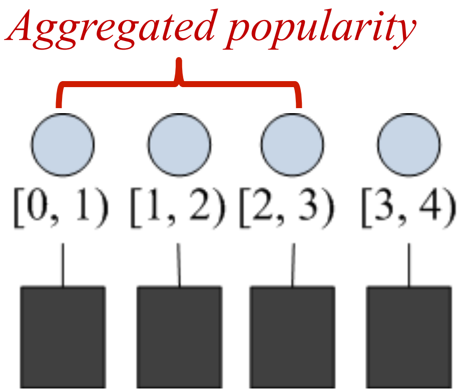
- Query: Top-K aggregation with range selections
- App: What are tweeted now in a particular area?
  - Schema: Recent Tweets (tag, *long*, *lat*, …)
  - Top-k popular tags in a given geographic area.

```
SELECT tag, count(*) as c FROM tweets
WHERE long BETWEEN l1 AND l2
      lat BETWEEN l3 AND l4
GROUP BY tag
ORDER BY c LIMIT k
```



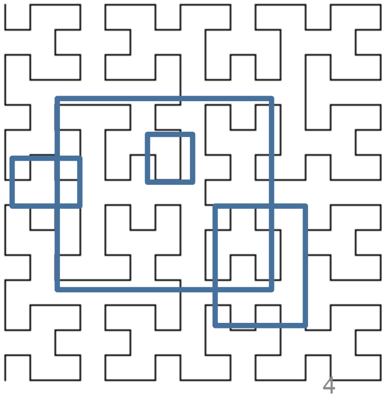
Problem Statement

- For schema <a, b>, given queried range [b1, b2], find top-*k* popular records within limited latency.



1D range ← 2D range

Hilbert space-filling curve



Related Work (on Top-K)

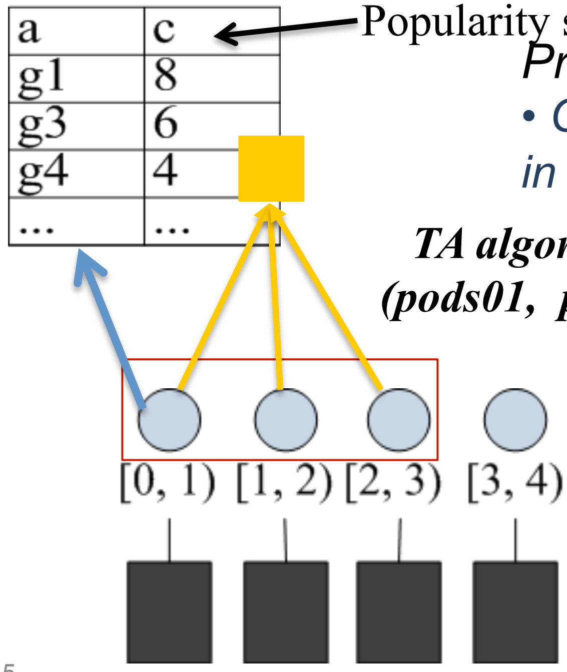
- Top-k query processing algorithms
  - FA (pods96), TA (pods01, jccs03), TPUT (podc04), KLEE (VLDB05)
- Materialized View for Top-k Query
  - Yi Ke (icde03), LPTA (vldb06), Top-k monitoring (sigmod03)
- Few prior work addresses "**interactive** top-k aggregation processing with **dynamic range predicates**."

Baselines

- Partitioning: Given schema <*a, b*>, we range-partition data on attribute *b*.

- Two baselines for top-k processing
  - Local view with threshold algorithm (TA)
  - Segment tree-based view

Baseline 1: Local View



a	c
g1	8
g3	6
g4	4
…	…

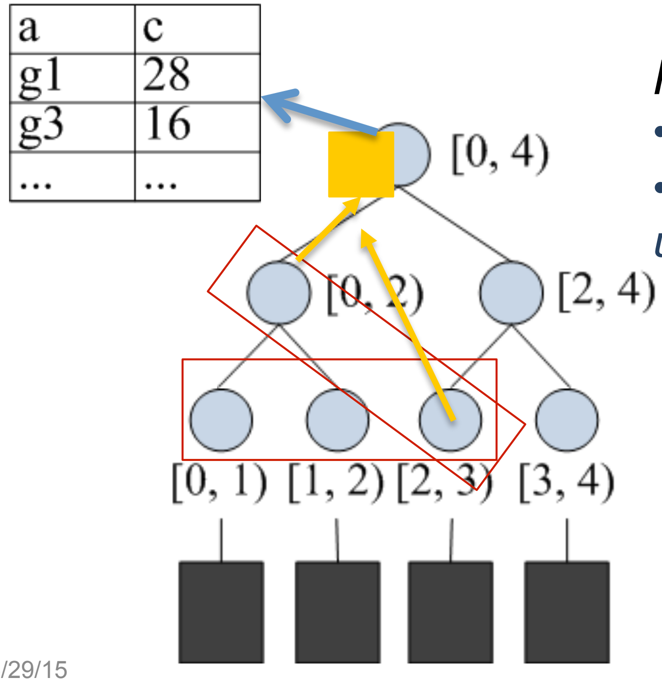
Popularity scores

TA algorithms
(pods01, podc04)

Problem of Local Views

- *Can't scale to large # of partitions in query latency and costs.*

Baseline 2: Segment-tree View



a	c
g1	28
g3	16
…	…

Problem of Tree Views

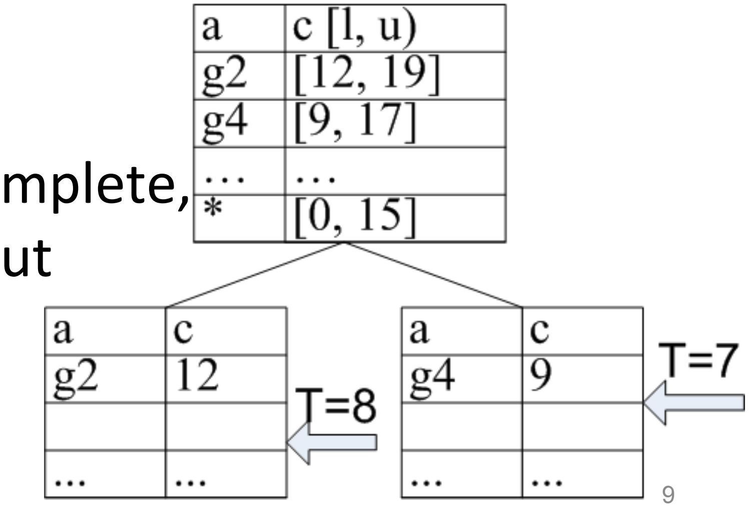
- *Extra maintenance overhead.*
- *High-level views handle global updates, leading to bottleneck.*

Benefits of Segment Tree Views

- *For query spanning r leaves, only **log(r)** internal nodes are required for query answering*

KTV-Tree: Threshold-based Incomplete View Tree

- Basic idea:
  - Threshold on each node to filter out updates on small values.
- The maintained view is incomplete, due to the threshold filter out certain updates.



a	c [l, u)
g2	[12, 19]
g4	[9, 17]
…	…
*	[0, 15]

a	c
g2	12
…	…

T=8

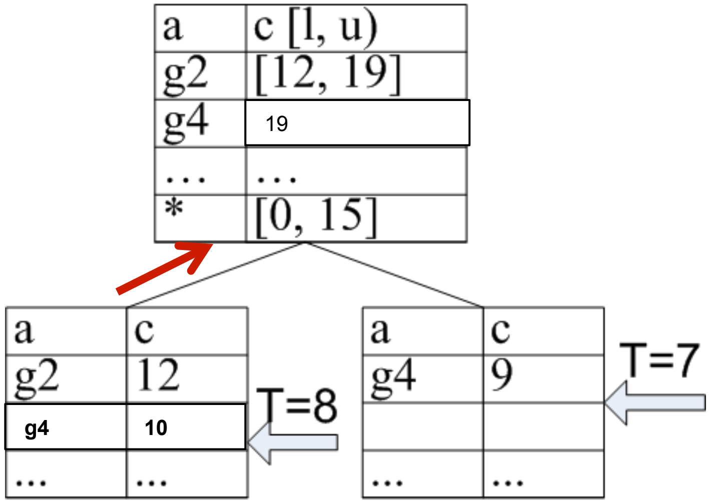
a	c
g4	9
…	…

T=7

Incremental tree maintenance

- Update the views (given fixed threshold).
  - – Given updates from child, decide whether to report to parents.
    - Based on threshold

- Update the thresholds.
  - – Triggered periodically



a	c [l, u)
g2	[12, 19]
g4	19
…	…
*	[0, 15]

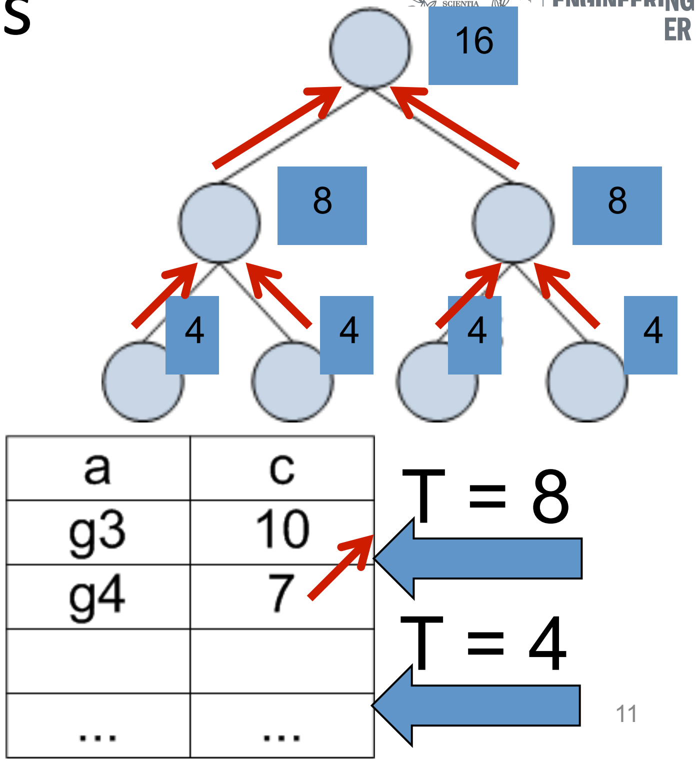
a	c
g2	12
g4	10
…	…

a	c
g4	9
…	…

Update the thresholds

- Step 1: a top-down process for updating threshold
  - Initiated by root node,
  - Propagate down to leaf, such that $T_{parent} = \sum T_{child}$.

- Step 2: a bottom-up process to refill view entries.

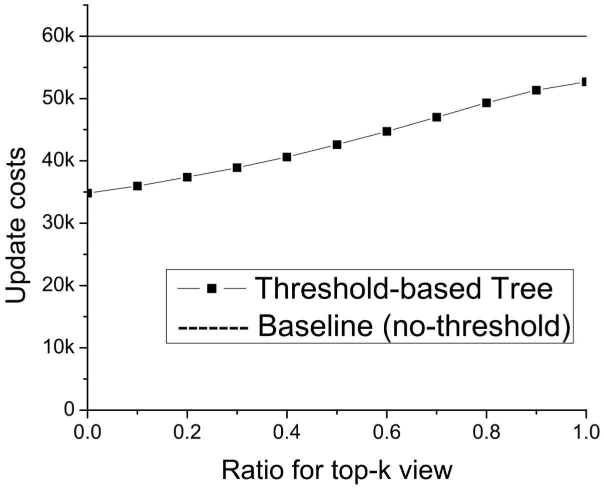


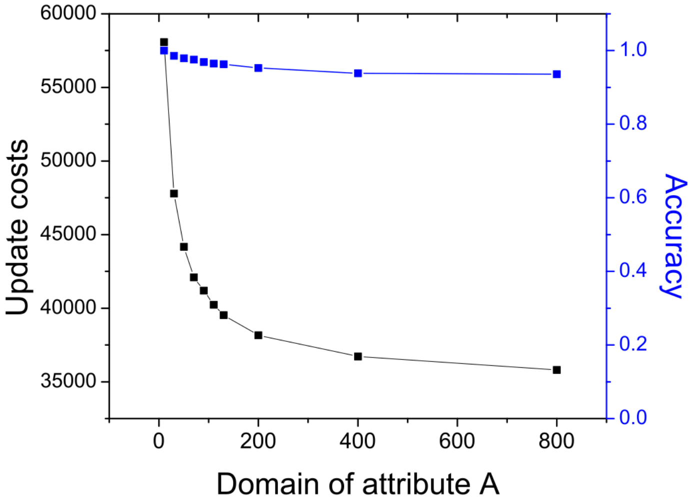
a	c
g3	10
g4	7
…	…

Experiment setups

- Synthetic dataset: triplet < *A*,*B*,*C* >
  - A, is randomly picked from 50 distinct tags.
  - B is numeric, randomly distributed in [0,32].
  - C is bounded by 50, following uniform distribution and Zipf distribution.
- Two data batches:
  - Loading: populating the data store and initializing thresholds
  - Performance evaluation
- Platform setup:
  - Software: Implemented on top of HBase
  - Hardware: Twenty commodity machines.

Preliminary results: Update costs





Summary

- We study the problem of interactive top-*k* aggregation query over dynamic data.
- We propose KTV-TREE, which combines the threshold based mechanism with materialized views
- KTV tree achieves the fast top-*k* aggregation processing with reasonably degraded accuracy.
- Future work includes more mature prototyping of KTV-TREE (e.g. on Spark) and experimentation.

Questions?

SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

Thank you

Contact:

Yuzhe Tang
Syracuse University
Email:ytang100@syr.edu
Web: ecs.syr.edu/faculty/yuzhe