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.

```sql
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 \([b_1, b_2]\), find top-\(k\) popular records within limited latency.

Diagram:

- Aggregated popularity
- Hilbert space-filling curve

6/29/15
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

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>8</td>
</tr>
<tr>
<td>g3</td>
<td>6</td>
</tr>
<tr>
<td>g4</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Problem of Local Views
- Can’t scale to large # of partitions in query latency and costs.

TA algorithms (pods01, podc04)

Popularity scores
Baseline 2: Segment-tree View

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>28</td>
</tr>
<tr>
<td>g3</td>
<td>16</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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.
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
Update the thresholds

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

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

- Threshold-based Tree
- Baseline (no-threshold)
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?

Thank you

Contact:
Yuzhe Tang
Syracuse University
Email: y tang100@syr.edu
Web: ecs.syr.edu/faculty/yuzhe