Feather: Hierarchical Querying for the Edge

Seyed Hossein Mortazavi, Mohammad Salehe, Moshe Gabel, Eyal de Lara
Data on the Edge

• Data is generated over a wide geographic area
  • Is stored near the edges
  • Pushed periodically upstream to a hierarchy of data centers

• Network properties:
  • Limited bandwidth
  • High latency
  • Failures
Querying data on the Edge

```
SELECT monitor_id, MAX(temperature) 
FROM Sensors 
GROUP BY monitor_id 
WHERE now() - timestamp < 600s
```
Querying Over a Distributed Hierarchical Database

Common approaches:

➢ Process on query on the Cloud
➢ Stream Processing (continuous query)
➢ Query edge data centers
Feather

➢ Hybrid Approach
  • Take benefit of data that exists on intermediate nodes
  • User specifies data freshness
    • System guarantees data freshness criteria
    • Improved query response time and total bandwidth
Querying

• Get max temperature for each sensor in the last 10 minutes

```
SELECT monitor_id, MAX(temperature) FROM Sensors GROUP BY monitor_id WHERE now() - timestamp < 600s
LAXITY = 60s
```
Contributions

• Global queries with control over staleness and query latency

• Fault tolerance with estimates about result completeness, coverage
Idea: Relax Freshness Requirement

• User provides minimum freshness requirement ("Laxity")
• System guarantees answer is at least as fresh ("Staleness")

Freshness guarantee is similar to formal treatments such as Δ-atomicity (Golab et al) [27] and t-freshness (Rahman et al.)
Example

Select Avg(K) from T1
Laxity = 2

Cloud

K1, 2
K3, 4
Regional Office E

K1, 2
K6, 7.5
Factory C

K3, 4
K7, 5.5
Factory B

K2, 3
K5, 6.5
Regional Office D

K2, 3
K5, 6.5
Factory A

K5, 6.5
Factory A push
1 2 3 4 5 6

K5, 6.5
Factory B push

K7, 5.5
Factory A push

9
Feather Features

• Supports: Filtering, aggregation, grouping, ordering, and limiting of the result set.

• Coverage estimation:
  • For each query return network and row coverage estimation

• Failures:
  • Best effort: Relax freshness guarantee and provide best results
    • (K1, K2, K3, K5)
  • Return partial results but up-to-date results
    • (K1, K2, K5)
Experimental Setup for Controlled Experiments

- **NYC Taxi Dataset**
  - 7 million taxi rides of December 2019
    - (sped up x30 times for more dense data)

- **Geo-distributed labelled data**
  - SELECT, GROUPBY, MIN queries
Feather Tradeoffs

- Flexible trade-off between latency, staleness while guaranteeing the freshness threshold.
Staleness vs latency
Coverage

Strong agreement between the real and the estimated row coverage
Real world Experiment

- Geo-tagged public tweets as the dataset
- 10 datacenters from three different cloud operators spread over three continents
- Scraped a total of 1 million tweets from 6 edge cities over a one-week period from December 2019.
- Real world latencies are not uniform!
Real world Experiment

Latency/staleness tradeoff for queries in the twitter experiment shows more clusters

Coverage estimation remains very accurate
More results

- MIN, GROUPBY, SELECT
- Rows vs. Laxity: 0, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500

- Latency vs. Laxity: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
- Topologies: deep, medium, wide
- $f = 30, 60$

- Avg. rows per edge vs. Laxity: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

University of Toronto
Summary

• Feather: a geo-distributed, hierarchical, eventually-consistent tabular data store that supports efficient global queries

• Feather provides a user-controlled tradeoff between latency, staleness, bandwidth, and load on edge nodes

• Feather provides completeness (coverage) estimate.

• Future work:
  • Improve the implementation for non-disjoint keys.
  • To investigate dynamic control policies for the latency/staleness tradeoff
Questions

mortazavi@cs.toronto.edu