Pyro: A Spatial-Temporal Big-Data Storage System

Shen Li
Shaohan Hu
Raghu Ganti
Mudhakar Srivatsa
Tarek Abdelzaher
Applications

• A huge amount of geo-tagged events are generated and stored in real-time.
  – Tweets, Photos
  – Taxi locations
  – Smartphone User Traces

• Query ask for events within a given time range and geographic area: geometry query.

Challenges

• Efficiently store and retrieve Spatial-temporal data

• Achieve Scalability

• Handle dynamic workload hotspot
Prior Approaches

• Make Geographic Information Systems (GIS) scalable
• Make Big-Data storage system understand spatial-temporal workload

Contributions

• Pyro is the first holistic solution specifically designed for Spatial-Temporal Applications.
  – Internally understands Spatial-Temporal data and query
  – Aggregatively optimizes IO
  – Manages data replicas to mitigate workload hotspots
Background

- **HBase**
  - The table is horizontally divided into HRegions.
  - Each HRegion is vertically divided into stores, one store per column family.
  - Data is first cached in the MemStore, and then flushed into a StoreFile when the size threshold is reached.

- **HDFS**
  - The Name Node manages file system namespaces.
  - Data Nodes store data chunks
  - DFS Client exposes APIs.
Pyro Architecture

- **Geometry Translator**
  - Encoding spatial-temporal information into row keys, and translating geometry queries into range scans

- **Multi-Scan Optimizer**
  - Aggregatively optimizing all range scans of the same geometry query

- **Group-Based Replica Placement**
  - Improves data locality during workload dynamics.
Pyro Architecture

- **Geometry Translator**
  - Encoding spatial-temporal information into row keys, and translating geometry queries into range scans

- **Multi-Scan Optimizer**
  - Aggregatively optimizing all range scans of the same geometry query

- **Group-Based Replica Placement**
  - Improves data locality during workload dynamics.

- Geometry Translator
  - Encoding spatial-temporal information into row keys, and translating geometry queries into range scans

- Multi-Scan Optimizer
  - Aggregatively optimizing all range scans of the same geometry query

- Group-Based Replica Placement
  - Improves data locality during workload dynamics.
• The space is recursively divided into tiles using a quad-tree

• Using a space filling curve (Z, Moore, Hilbert, etc.) to encode tiles

• Use the same quad-tree to calculate the tiles that intersect with the geometry

• Tiles then turns into range scans.
Pyro Architecture

• Geometry Translator
  – Encoding spatial-temporal information into row keys, and translating geometry queries into range scans

• Multi-Scan Optimizer
  – Aggregatively optimizing all range scans of the same geometry query

• Group-Based Replica Placement
  – Improves data locality during workload dynamics.
Multi-Scan Optimizer: Read Amplification

- A Geometry query may translate into a large number of range scans.
- These range scans usually force the underlying system to fetch more data or repeatedly go through the same data structure.
Multi-Scan Optimizer: Use Small Tile and HBlocks

- Keep tile size and block size small, and aggregatively optimize range scans.
- Profile P-Read delay vs size.
- Use Dynamic Programming to determine which blocks to read

Adaptive Aggregation Algorithm:

$$S[i] = \min\{S[j - 1] + E(j, i) | 1 \leq j \leq i\}$$
Pyro Architecture

- **Geometry Translator**
  - Encoding spatial-temporal information into row keys, and translating geometry queries into range scans

- **Multi-Scan Optimizer**
  - Aggregatively optimizing all range scans of the same geometry query

- **Group-Based Replica Placement**
  - Improves data locality during workload dynamics.
**Group-Based Replica Placement**

- Each HRegion handles a range of row keys, that corresponds to a subarea in the space.

- Spatial-temporal applications naturally create dynamic workload hotspots within small areas that may overwhelm corresponding HRegion servers.
Group-Based Replica Placement Policy

- A HRegion can split to input multiple daughter HRegions, and these daughter HRegions can be moved into other machines to mitigate workload hotspot.

- HRegions usually co-locate with HDFS datanodes that allows read/write data locality. Splitting may destroy data locality.

- Pyro employs group-based replica placement to achieve data locality.
Group-Based Replica Placement

Asymmetry

- The asymmetry in replica groups caters HFile format: meta data locates at the end of the Hfile.

- Meta blocks: minimize the probability of losing any DFS block

- Data blocks: minimize the expectation of the number of unavailable DFS blocks.

n: # of servers,  \( f \): # of failed servers,  
g: # of groups,  \( b \): # of DFS blocks in the file

![Graph showing the probability of failure vs. number of grouped replications.](Image)

- Pr[Meta Failure] vs. Number Grouped Replications
- Different lines correspond to different values of \( f/n \) and \( b \).

14
• Open data: ~700,000,000 NYC taxi trips from 2010 to 2013.
  – https://publish.illinois.edu/dbwork/open-data/

• Experimenting on an 80-server cluster:
  – 1 PyroDFS namenode, 30 datanodes
  – 1 PyroDB master, 3 ZooKeeper nodes, 30 co-located HRegion servers.
  – Remaining nodes generate workload and log latency.

• Compare with Md-HBase
  – Md-HBase adds an translation layer above Hbase, and uses Z-order encoding.
Evaluation

- Manually splitting a Pyro region vs Manually splitting a Md-HBase region.
  - To make the evaluation fair, this evaluation submits range scans rather than geometry query into two systems. In this case, both geometry translator and multi-scan optimizer in Pyro are disabled.
  - Both systems use Z-order encoding algorithm
Evaluation

- Throughput measurement of 100m X 100m rectangle geometry.
  - PyroM: Pyro using Moore encoding
  - PyroZ: Pyro using Zorder encoding
  - PyroM - A3: PyroM, disabled adaptive aggregation algorithm
  - PyroZ - A3: PyroZ, disabled adaptive aggregation algorithm
Thank you  Q&A