A Main-Memory Database for Future Connected Mobility Workloads

Andreas Kipf

Technical University of Munich

HPTS 2017
ABSTRACT
In the past few years, massive amounts of location-based data has been captured. Numerous datasets containing user location information are readily available to the public. Analyzing such datasets can lead to fascinating insights into the mobility patterns and behaviors of users. Moreover, in recent times a number of geospatial data-driven applications that are useful for knowledge discovery into the database that are useful for knowledge discovery into the database kernel. The goal is to have a full-fledged general-purpose database that allows big data analysis along with conventional transaction processing.

At the same time, there has been an emergence of data-driven applications. Companies like Uber, Lyft, and Foursquare have a need to create real-time applications,
SELECT points.aggr, ST_AsText(geo) AS geo
FROM (SELECT avg(TIP_AMOUNT) AS aggr, borough, FROM yellow, boroughs, (SELECT ST_Covers(boroughs.geo, yellow) AS yellow) AS yellow
WHERE ST_Covers(boroughs.geo, yellow)) AS boroughs
GROUP BY borough, neighborhood) AS aggrs

Group by following:
- Neighborhoods
- Boroughs

Aggregate
Aggregation Function
- Average

Aggregation Column
- TIP_AMOUNT

Order
- DESC

Area

<table>
<thead>
<tr>
<th>Area</th>
<th>TIP_AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brooklyn, Dyker Heights</td>
<td>25.72</td>
</tr>
<tr>
<td>Staten Island, Pleasant Plains</td>
<td>25.20</td>
</tr>
<tr>
<td>Bronx, Pelham Bay Park</td>
<td>23.27</td>
</tr>
<tr>
<td>Staten Island, Westerleigh</td>
<td>15.00</td>
</tr>
<tr>
<td>Staten Island, Randall</td>
<td>12.98</td>
</tr>
<tr>
<td>Staten Island, Rosebank</td>
<td></td>
</tr>
<tr>
<td>Staten Island, Emerson</td>
<td></td>
</tr>
<tr>
<td>Staten Island, Oakwood</td>
<td></td>
</tr>
<tr>
<td>Staten Island, St. George</td>
<td>7.69</td>
</tr>
</tbody>
</table>

Powered by HyperSpace
Real-Time Maps Example (1)

**Car-hailing companies require real-time maps**

- High-velocity updates
- (Geospatial) queries on up-to-date state

**Geofencing use cases**

- Given a user’s lat/lng coordinates
  - Find the geofence (polygon) that the coordinates lie in
  - Show user products that are available at the given location
- Dynamic pricing per neighborhood

Source: https://eng.uber.com/go-geofence/
Real-Time Maps Example (2)

**Satellite image processing companies provide a virtual representation of the real world**

- They extract features (e.g., cars) from satellite images and repeatedly join these features with existing datasets (e.g., US parking lots)
- Show that they can forecast the stock price of US retail chains

“Orbital Insight uses deep learning algorithms to accurately identify cars from satellite images at 55,000+ parking lots of major retail chains across the U.S.”

System Design Challenges

**Fast data and query ingestion**
- High-velocity updates and key lookups along with high-performance networking

**Fast geospatial joins**
- High-performance geospatial joins that adapt to changing workloads

**Fast processing of continuous queries**
- Re-optimization of query plans based on observed cardinalities

**Fast analysis of historical spatio-temporal data**
- Efficient storage layouts and dynamic pre-aggregation

Today, there’s no single system that holistically addresses all of these challenges
Geospatial Join Problem

Points
• E.g., GPS positions

Polygons
• Typically disjoint political boundaries such as neighborhoods
• Or Voronoi cells (for NN queries)

Point/polygon join
• Which polygon does a given point lie in?
• Summary statistics for all points that lie in a certain polygon
Traditional Approach

1. Construct an R-tree index on the polygons’ MBRs
2. Perform an index nested loop join
Our Approach

**Skip the expensive refinement phase**

- Referred to as true hit filtering
- Invented in the 90s
- Only a few systems have used this idea in the last two decades
Google S2

Open sourced by Google in 2011
Maps every point on earth onto a cube
Recursively subdivides the cube

face bits
position along the Hilbert curve

1 0 0 1 0 1 0 0 0 0 0 0 ... 0

1 0 0 1 1 1 1 1 1 1 1 1 0 ... 0

1 0 0 0 0 0 0 0 0 ... 0

64 bits
Polygon Approximations

**Covering**
- A collection of non-uniform cells *covering* a polygon

**Interior covering**
- A collection of non-uniform cells *lying fully within* a polygon
Polygon Approximations

**Super covering**

- A combination of multiple coverings and interior coverings with each cell mapping to one or many polygons

**Cell types**

- Blue cells are covering cells of single polygons
- Red cells are covering cells of multiple polygons
- Green cells are interior cells of single polygons
Polygon Approximations
Polygon Approximations
Evaluation

**Evaluation system**

- 2x Intel(R) Xeon(R) CPU E5-2680 v4 CPU (2.40 GHz, 3.30 GHz turbo)
- 256 GB DDR3 RAM
- Ubuntu 16.04

**Points**

- NYC taxi rides (1B)

**Polygons**

- NYC boroughs (5)
- NYC neighborhoods (290)
- NYC census blocks (40k)
## Evaluation

### Throughput in M points/s

<table>
<thead>
<tr>
<th></th>
<th>boroughs</th>
<th>neighborhoods</th>
<th>census blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostGIS</td>
<td>0.39</td>
<td>1.09</td>
<td>0.69</td>
</tr>
<tr>
<td>Spark Magellan</td>
<td>0.88</td>
<td>4.57</td>
<td>2.24</td>
</tr>
<tr>
<td>R-tree</td>
<td>3.88</td>
<td>61.2</td>
<td>28.9</td>
</tr>
<tr>
<td>exact</td>
<td>3735</td>
<td>1459</td>
<td>431</td>
</tr>
<tr>
<td>approx.</td>
<td>4532</td>
<td>2280</td>
<td>874</td>
</tr>
</tbody>
</table>
Conclusions

Modern geospatial workloads are challenging

Existing work on geospatial data processing needs to be adapted for the use of modern hardware

Ongoing and future work

- High-performance networking (mTCP)
- A distributed version of our geospatial join that uses RDMA
- Re-optimization of query plans based on observed cardinalities
- Indexing of historical spatio-temporal data

Ultimately, we want to build a highly efficient database system for spatio-temporal data...

stay tuned!