

Geospatial Analysis of Denver Crime Data around RTD Bus Stops

Performance Tuning / Coordinate Conversion

Info:

- Crime dataset has ~390,000 unique records
- Bus stop dataset has ~8,000 stops
- Calculating nearest bus stop for each crime with raw coordinate data is infeasible (would take days)

Our Solution:

- Use a spatially indexed nearest neighbor join for massive speedup
- PostGIS extension to convert coordinate data to geom object

```
1 -- Enable PostGIS extension and create new geometry columns
2 CREATE EXTENSION IF NOT EXISTS postgis;
3 ALTER TABLE crime_data ADD COLUMN geom geometry(Point, 4326);
4 UPDATE crime_data
5 SET geom = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);
6
7 ALTER TABLE bus_data ADD COLUMN geom geometry(Point, 4326);
8 UPDATE bus_data
9 SET geom = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);
10
11 -- Create indexes to speed up join immensely
12 CREATE INDEX crimes_gix ON crime_data USING GIST (geom);
13 CREATE INDEX bus_stops_gix ON bus_data USING GIST (geom);
```

QUERY PLAN

```
Sort  (cost=222282263.71..222282263.74 rows=1 width=28) (actual time=118616.018..118616.034 rows=539 loops=1)
  Sort Key: ((c.geom))
  Sort Method: quicksort Memory: 62kB
-> GroupAggregate (cost=222281779.66..222282263.72 rows=1 width=28) (actual time=118611.980..118615.952 rows=539 loops=1)
    Group Key: bus_data.bsid, bus_data.lng, bus_data.lat
    Filter: (count() > 20)
    Rows Removed by Filter: 1978
      -> Sort  (cost=222281779.66..222281876.47 rows=38724 width=20) (actual time=118611.971..118613.235 rows=39126 loops=1)
          Sort Key: bus_data.bsid, bus_data.lng, bus_data.lat
          Sort Method: quicksort Memory: 3563kB
            -> Nested Loop  (cost=5739.68..222278828.71 rows=38724 width=28) (actual time=5.325..118603.013 rows=39126 loops=1)
              -> Seq Scan on crime_data c  (cost=0.00..14589.10 rows=38724 width=32) (actual time=0.089..58.875 rows=39126 loops=1)
                  Filter: is_violent
                  Rows Removed by Filter: 352684
                -> Limit  (cost=5739.68..5739.68 rows=1 width=60) (actual time=3.029..3.029 rows=1 loops=39126)
                  -> Sort  (cost=5739.68..5766.65 rows=8387 width=60) (actual time=3.029..3.029 rows=1 loops=39126)
                      Sort Key: ((c.geom <> bus_data.geom))
                      Sort Method: top-N heapsort Memory: 25kB
                    -> Seq Scan on bus_data  (cost=0.00..5697.74 rows=8387 width=60) (actual time=0.020..2.243 rows=8387 loops=39126)

Planning Time: 0.111 ms
Execution Time: 118616.191 ms
(21 rows)
```

Performance Without Indices

QUERY PLAN

```
Sort  (cost=54141.01..54141.02 rows=1 width=28) (actual time=1052.823..1052.839 rows=540 loops=1)
  Sort Key: (count())
  Sort Method: quicksort Memory: 62kB
-> HashAggregate (cost=54140.99..54141.00 rows=1 width=28) (actual time=1052.597..1052.766 rows=540 loops=1)
  Group Key: bus_data.bsid, bus_data.lng, bus_data.lat
  Filter: (count() > 20)
  Batches: 1  Memory Usage: 649kB
  Rows Removed by Filter: 1978
    -> Nested Loop  (cost=0.15..53793.75 rows=38724 width=20) (actual time=0.115..1045.210 rows=39126 loops=1)
      -> Seq Scan on crime_data c  (cost=0.00..14589.10 rows=38724 width=32) (actual time=0.017..58.705 rows=39126 loops=1)
          Filter: is_violent
          Rows Removed by Filter: 352684
        -> Limit  (cost=0.15..0.99 rows=1 width=60) (actual time=0.025..0.025 rows=1 loops=39126)
          -> Index Scan using bus_stops_gix on bus_data  (cost=0.15..7056.80 rows=8387 width=60) (actual time=0.025..0.025 rows=1 loops=39126)
              Order By: (geom <> c.geom)

Planning Time: 0.167 ms
Execution Time: 1052.877 ms
(17 rows)
```

Performance With Indices

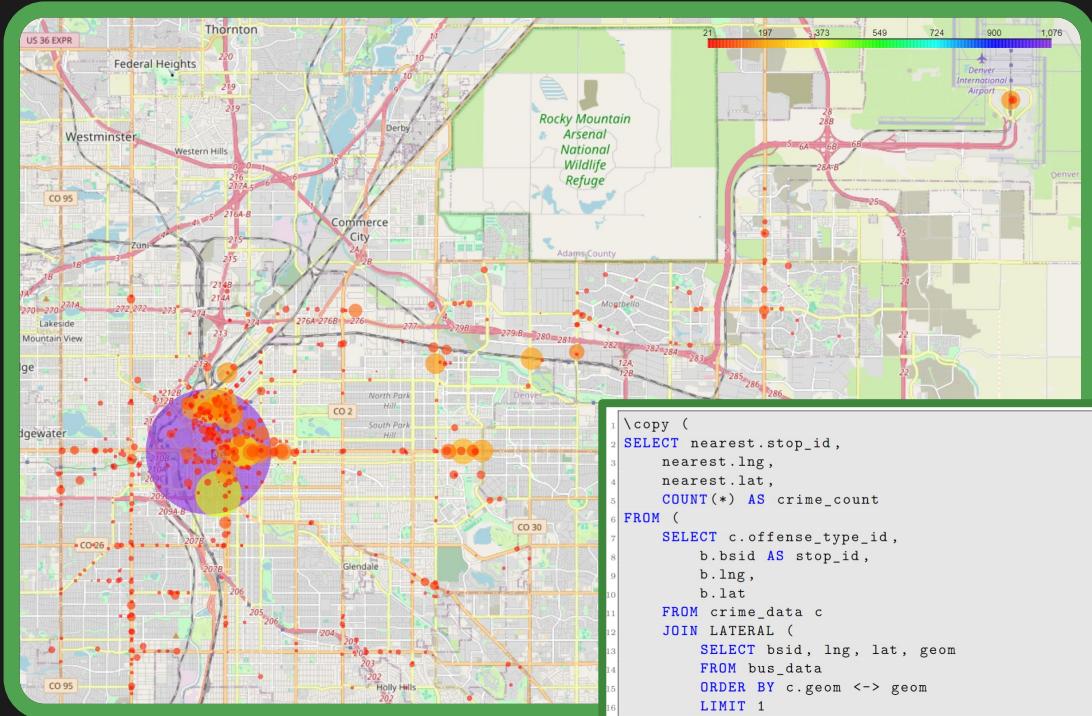
>100x Speed Up!

Geospatial Analysis of Denver Crime Data around RTD Bus Stops

Visualizing Violent Denver Crime by Stops

- Classify crimes as violent vs. non-violent.
- For each violent crime, find the nearest RTD bus stop using a nearest-neighbor geospatial search
- Group crimes by their closest stop and count them
- Keep only stops with > 20 violent crimes nearby
- Export results to CSV for Python + Folium to build a bubble-map visualization

[Demo Link](#)



```
\copy (
  SELECT nearest.stop_id,
         nearest.lng,
         nearest.lat,
         COUNT(*) AS crime_count
    FROM (
      SELECT c.offense_type_id,
             b.bsid AS stop_id,
             b.lng,
             b.lat
       FROM crime_data c
      JOIN LATERAL (
        SELECT bsid, lng, lat, geom
          FROM bus_data
        ORDER BY c.geom <-> geom
        LIMIT 1
      ) b ON TRUE
     WHERE is_violent = TRUE
    ) AS nearest
   GROUP BY nearest.stop_id, nearest.lng, nearest.lat
  HAVING COUNT(*) > 20
  ORDER BY crime_count DESC
) TO 'crime_bus_counts.csv' CSV HEADER;
```