A Main-Memory Database for Future Connected Mobility Workloads

Andreas Kipf

Technical University of Munich

HPTS 2017

High-Performance Geospatial Analytics in HyPerSpace

Varun Pandey
Tobias Mühlbauer

Andreas Kipf
Thomas Neumann

Dimitri Vorona
Alfons Kemper

Technische Universität München
{pandey, kipf, vorona, muehlbau, neumann, kemper}@in.tum.de

ABSTRACT

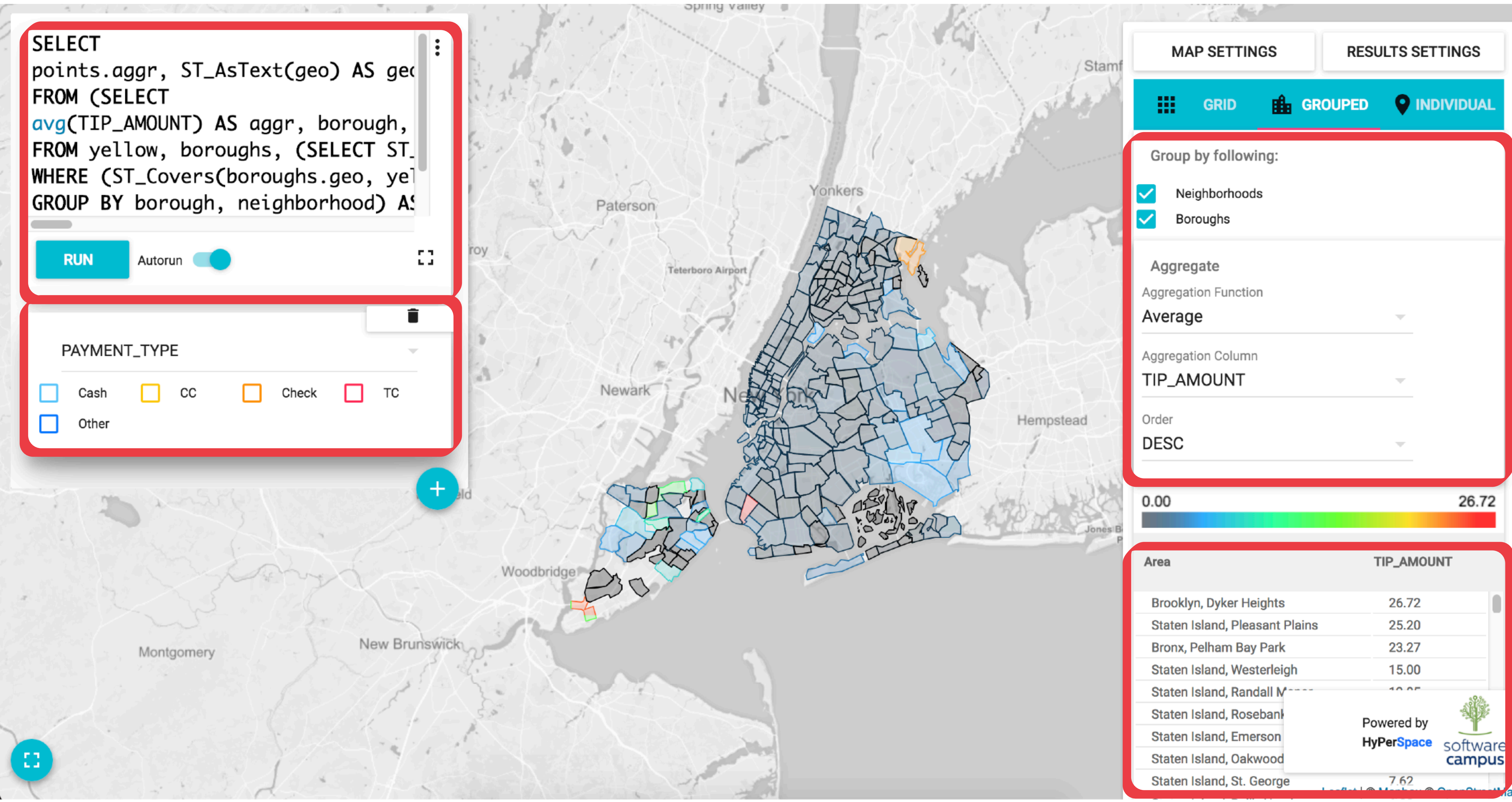
In the past few years, massive amounts of location-based data has been captured. Numerous datasets containing user location information are readily available to the public. Analyzing such datasets can lead to fascinating insights into the mobility patterns and behaviors of users. Moreover, in recent times a number of geospatial data-driven

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,

SIGMOD 2016

hypermaps-db.in.tum.de


SELECT
points.aggr, ST_AsText(geo) AS geo
FROM (SELECT
avg(TIP_AMOUNT) AS aggr, borough,
FROM yellow, boroughs, (SELECT ST_
WHERE (ST_Covers(boroughs.geo, yel
GROUP BY borough, neighborhood) AS
RUN
Autorun
PAYMENT_TYPE
Cash
CC
Check
TC
Other
MAP SETTINGS
RESULTS SETTINGS
GRID
GROUPED
INDIVIDUAL
Group by following:
Neighborhoods
Boroughs
Aggregate
Aggregation Function
Average
Aggregation Column
TIP_AMOUNT
Order
DESC
0.00
26.72
Area
TIP_AMOUNT
Brooklyn, Dyker Heights 26.72
Staten Island, Pleasant Plains 25.20
Bronx, Pelham Bay Park 23.27
Staten Island, Westerleigh 15.00
Staten Island, Randall M
Staten Island, Rosebank
Staten Island, Emerson
Staten Island, Oakwood
Staten Island, St. George 7.62
Powered by
HyPerSpace
software campus
Paterson
Yonkers
Teterboro Airport
Newark
New York
Hempstead
Woodbridge
Montgomery
New Brunswick

Real-Time Maps Example (1)

Car-hailing companies require real-time maps

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

UBER

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

"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."

Source: https://medium.com/@orbital_insight/orbital-insight-correctly-predicts-retail-sales-miss-hit-rate-grows-to-78-aa65ea445423

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

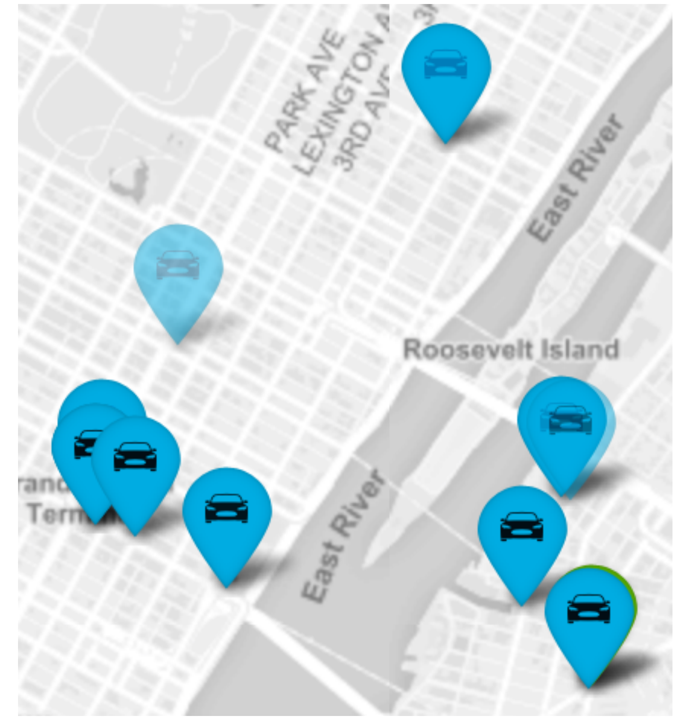
- E.g., GPS positions

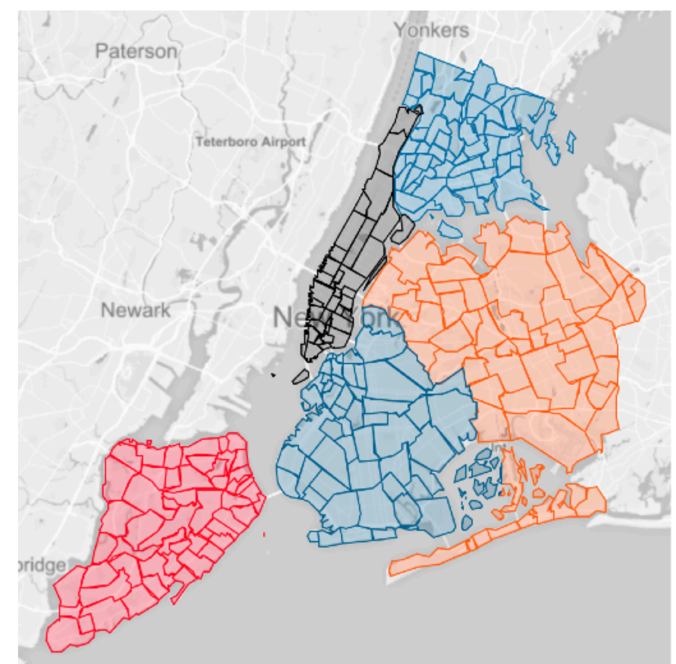
Polygons

- 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

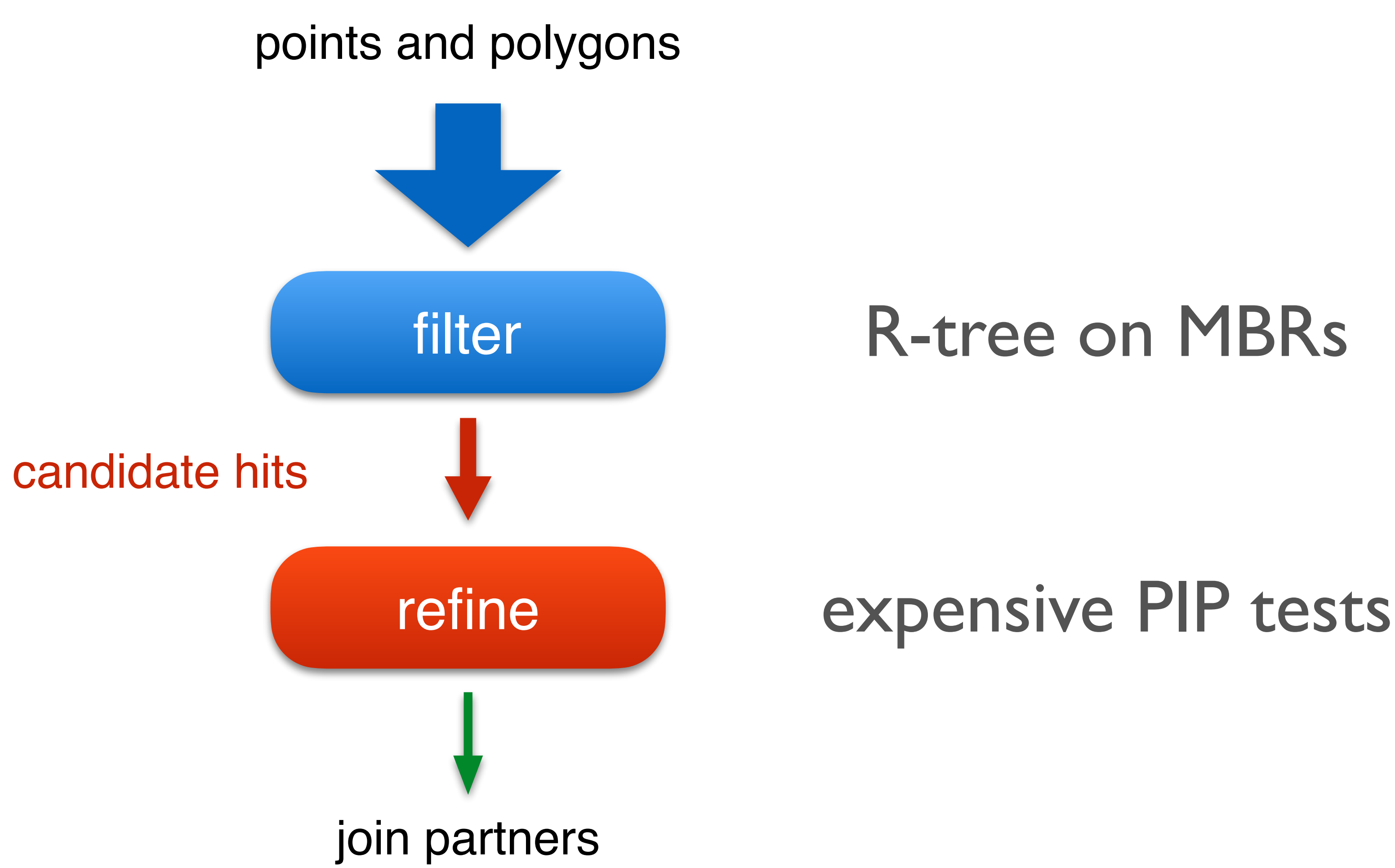




Area	FARE_AMOUNT
Staten Island	33.44
Queens	32.03
Bronx	13.62
Brooklyn	13.16
Manhattan	10.72

Traditional Approach

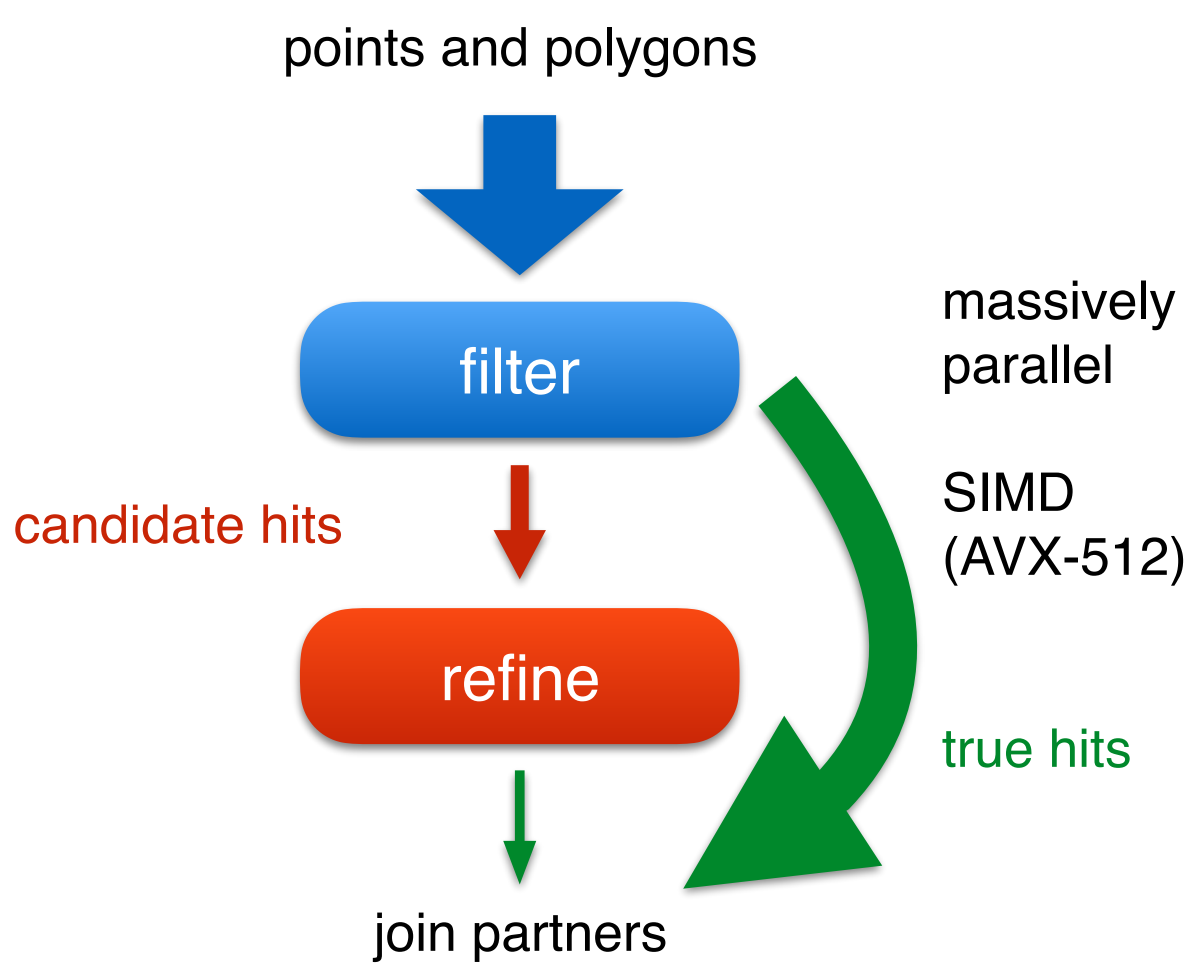
1. **Construct an R-tree index on the polygons' MBRs**
2. **Perform an index nested loop join**



Our Approach

Skip the expensive refinement phase

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

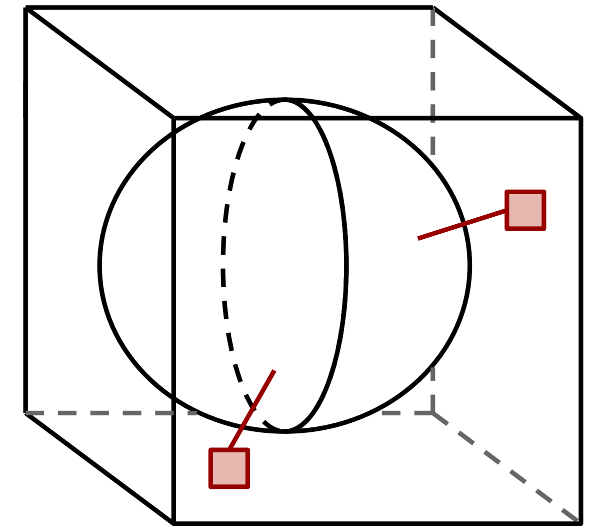


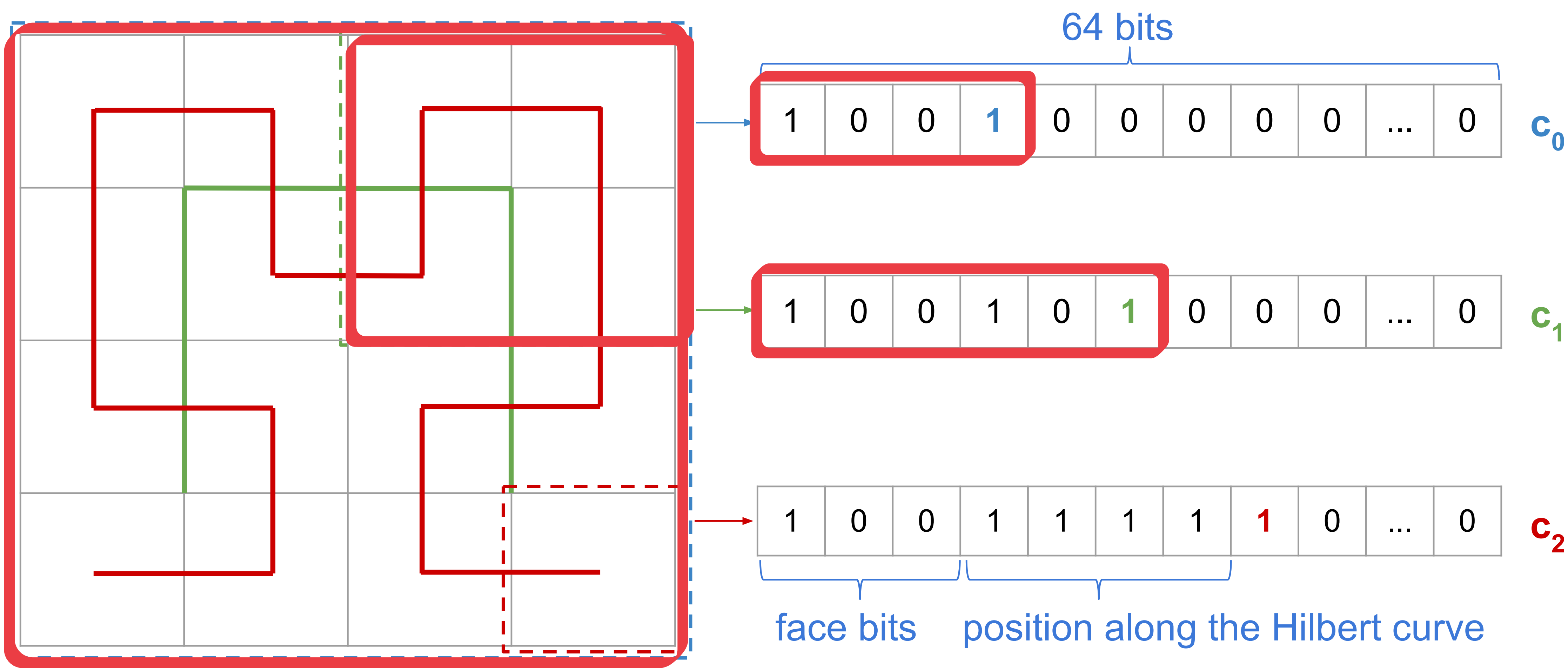
Google S2

Open sourced by Google in 2011

Maps every point on earth onto a cube

Recursively subdivides the cube

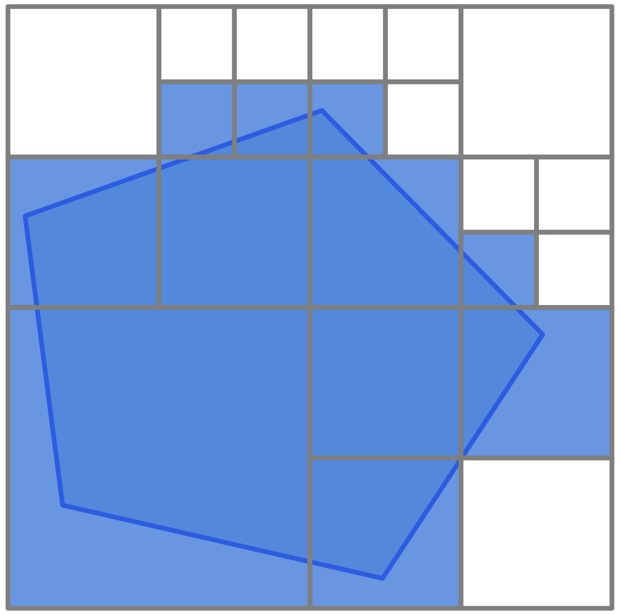




Polygon Approximations

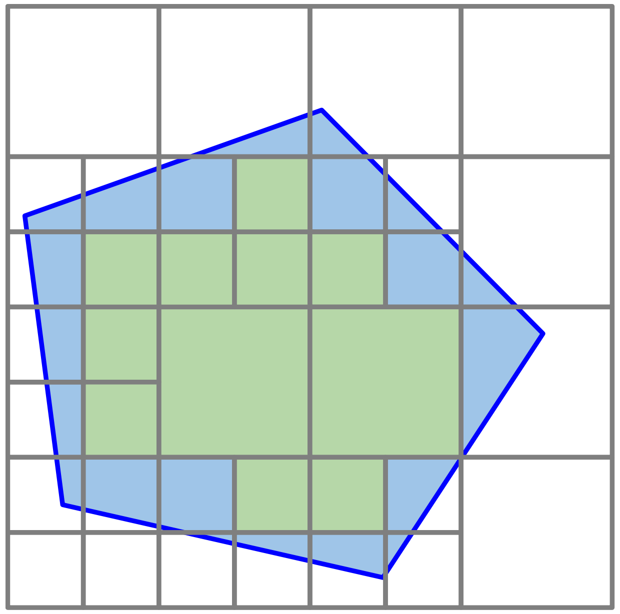
Covering

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



Interior covering

- A collection of non-uniform cells **lying fully within** a polygon



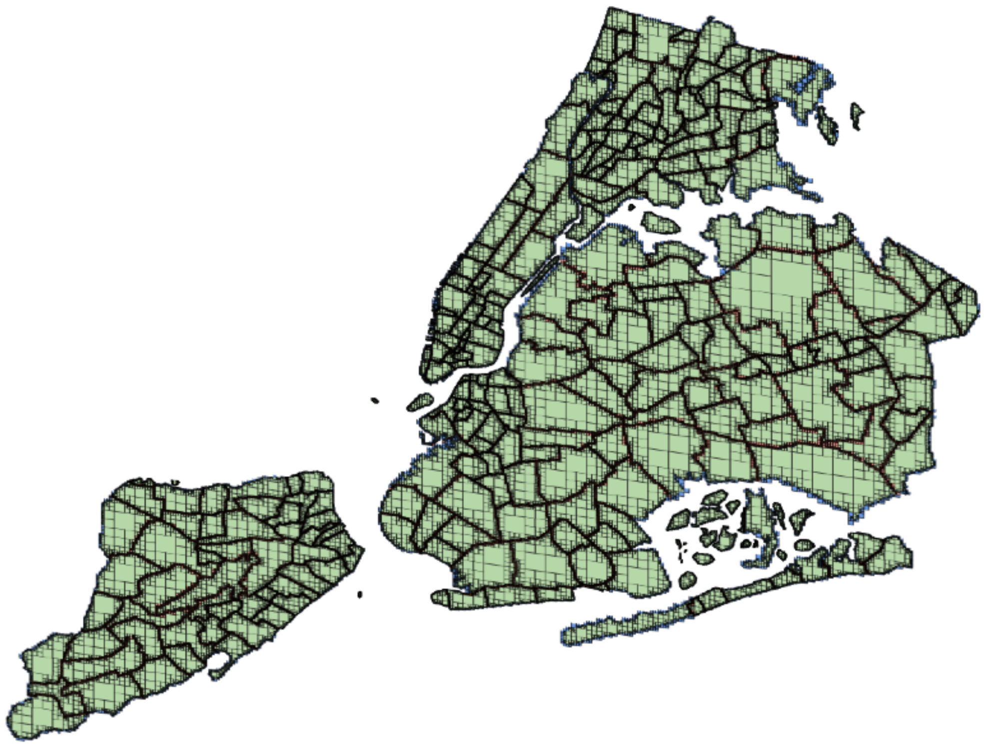
Polygon Approximations

Super covering

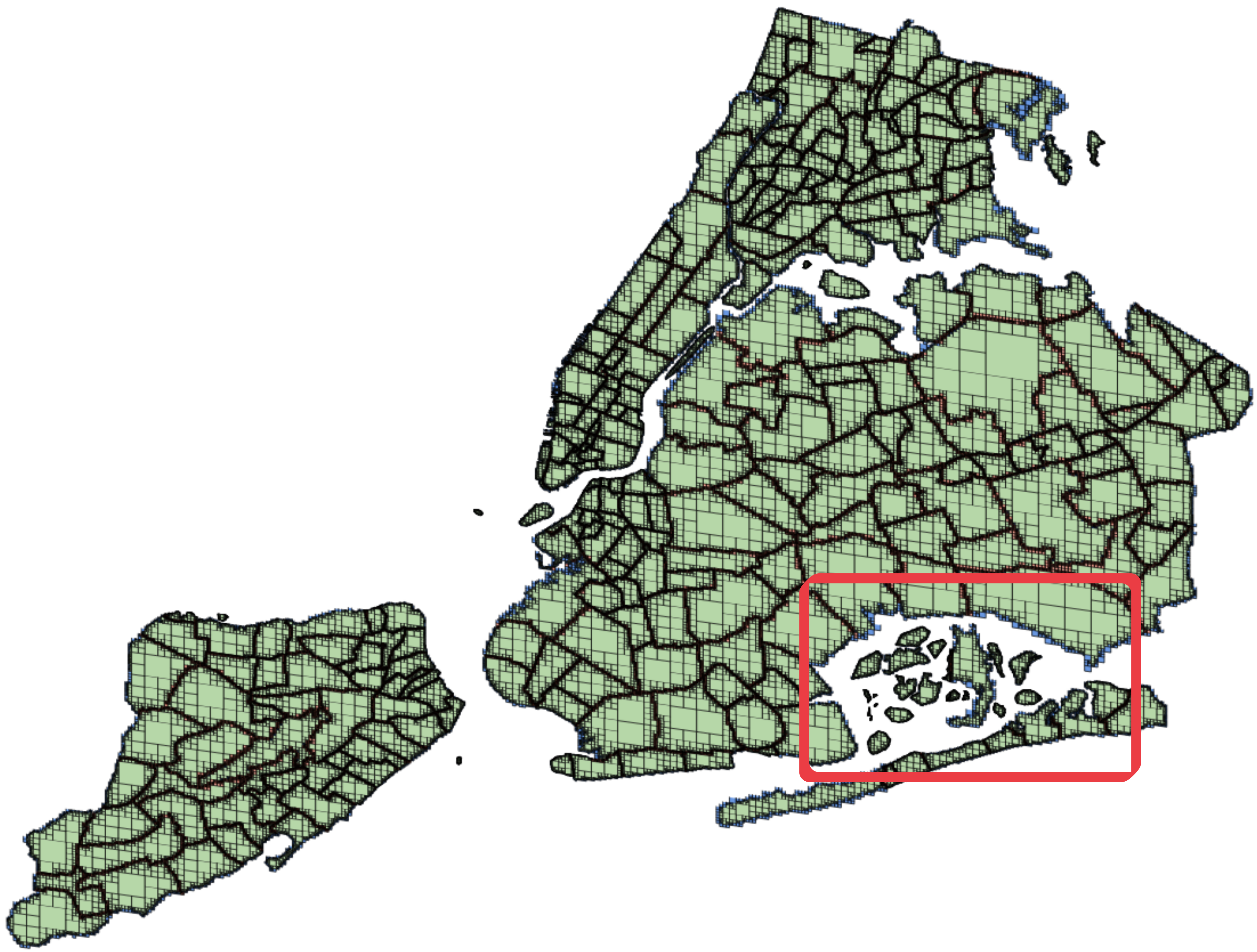
- A combination of multiple coverings and interior coverings with each cell mapping to one or many polygons

Cell types

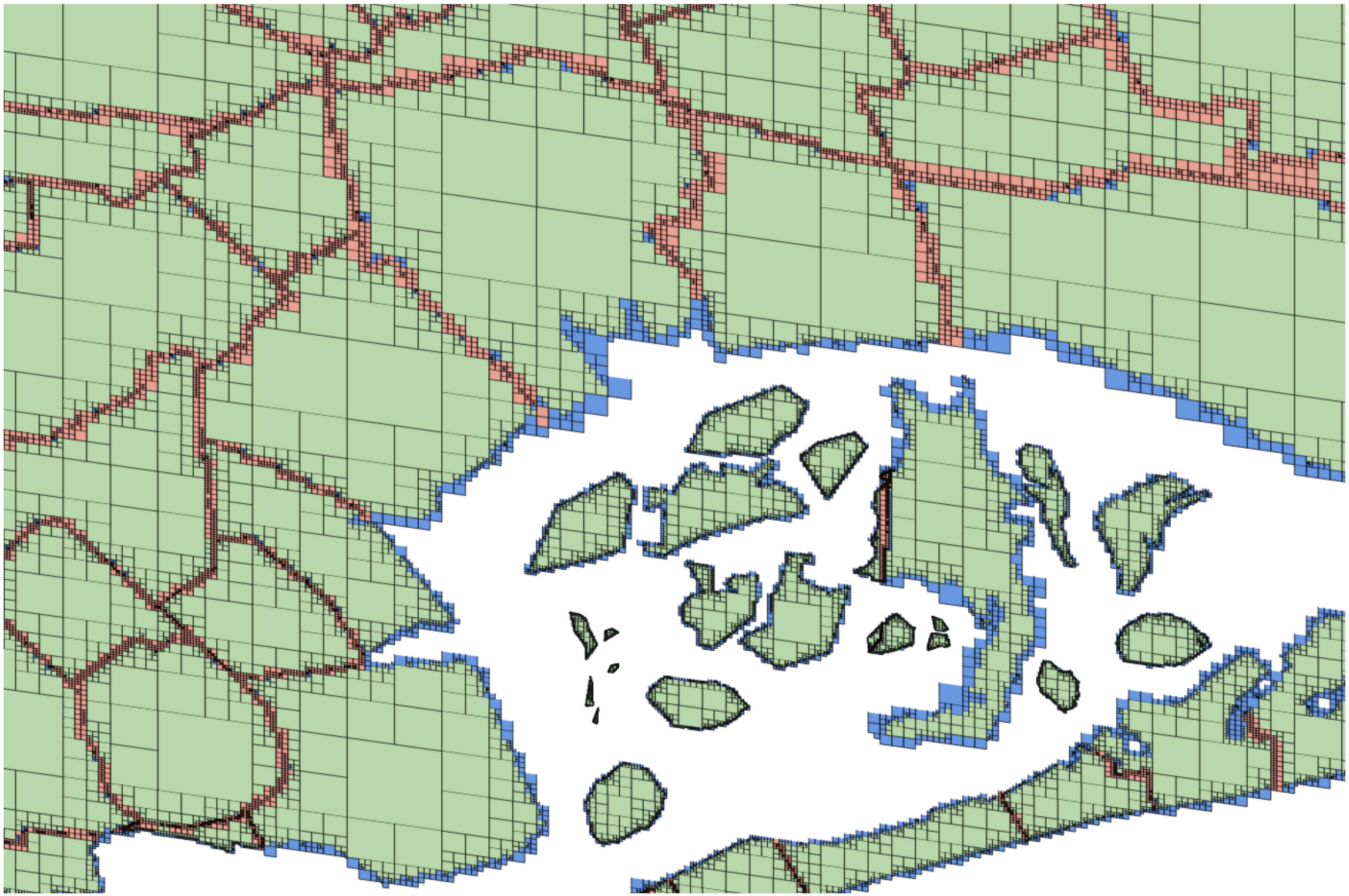
- Blue cells are covering cells of single polygons
- Red cells are covering cells of multiple polygons
- Green cells are interior cells of single polygons



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

	boroughs	neighborhoods	census blocks
PostGIS	0.39	1.09	0.69
Spark Magellan	0.88	4.57	2.24
R-tree	3.88	61.2	28.9
exact	3735	1459	431
approx.	4532	2280	874

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!