Spatial Data Management in Apache Spark

The **GeoSpark** Perspective and Beyond

Jia Yu

THIS TALK

GeoSpark

- GeoSpark overview
- Spatial RDD / DataFrame layer
- Spatial query processing layer
- Query optimizer
- GPU-based spatial database
- Experiments

THIS TALK

GeoSpark

- GeoSpark overview
- Spatial RDD / DataFrame layer
- Spatial query processing layer
- Query optimizer
- GPU-based spatial database
- Experiments

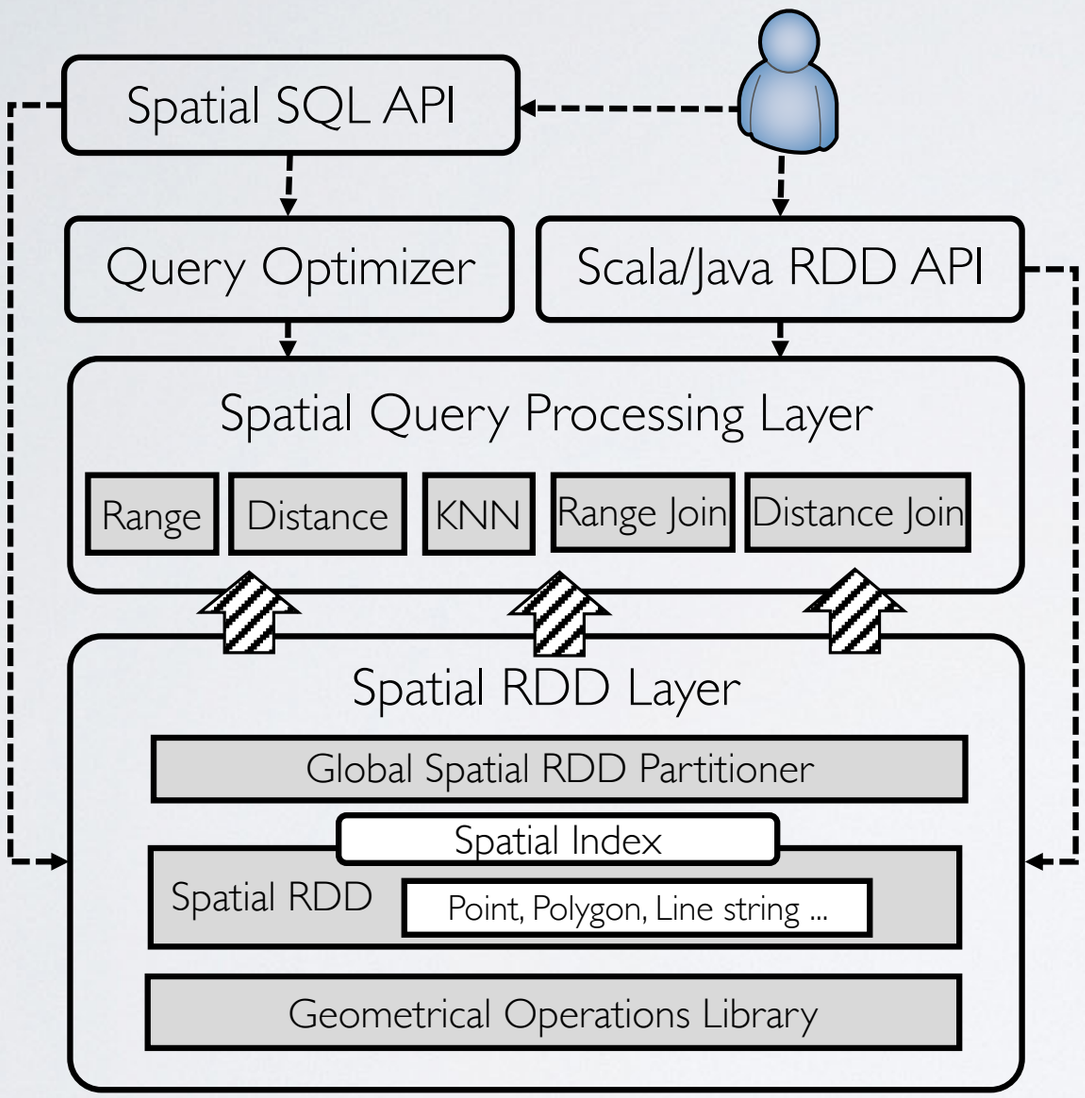
WHAT IS GEOSPARK

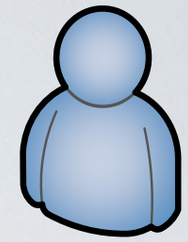
- A spatial data management system on top of Apache Spark **since 2015**
- Some statistics
  - **Monthly: downloads > 4k, visits > 8K; Overall: downloads > 40K, visits > 100K**
  - Was on listed as Infrastructure Project on Apache Spark official 3rd party project page
  - Users and contributors from Apple, Facebook, Uber, numerous startup companies
- Evaluation from a recent Very Large Data Bases (VLDB) 2018 research paper

"GeoSpark comes close to **a complete spatial analytics system** because of data types and queries supported and the control user has while writing applications. **It also exhibits the best performance in most cases.**"

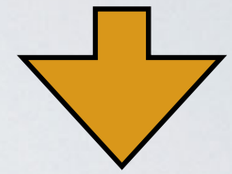
- How Good Are Modern Spatial Analytics Systems? PVLDB Vol11

GEOSPARK OVERVIEW

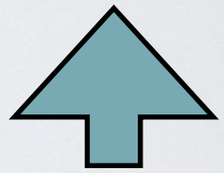




```
SELECT superhero.name
FROM city, superhero
WHERE ST_Contains(city.geom, superhero.geom)
AND city.name = 'Gotham';
```

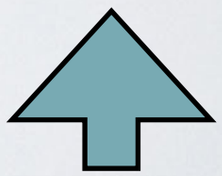


Query result



Query optimization

Spatial RDD / DataFrame



Spatial partitioning, Index

THIS TALK

GeoSpark

GeoSpark overview

Spatial RDD / DataFrame layer

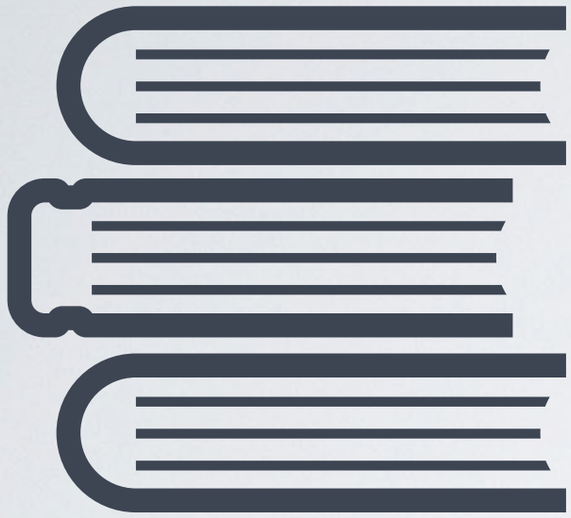
Spatial query processing layer

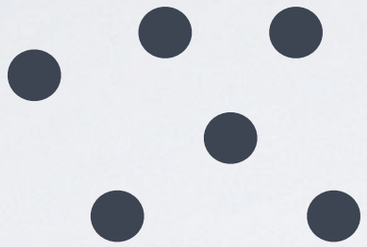
Query optimizer

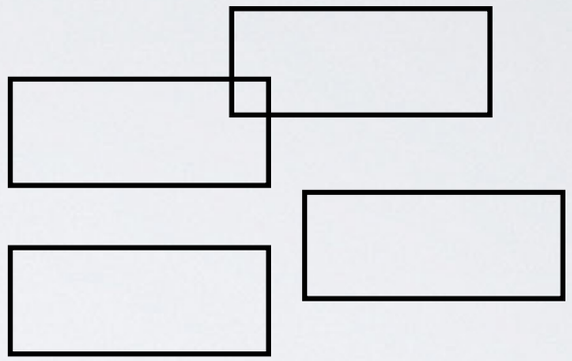
GPU-based spatial database

Experiments

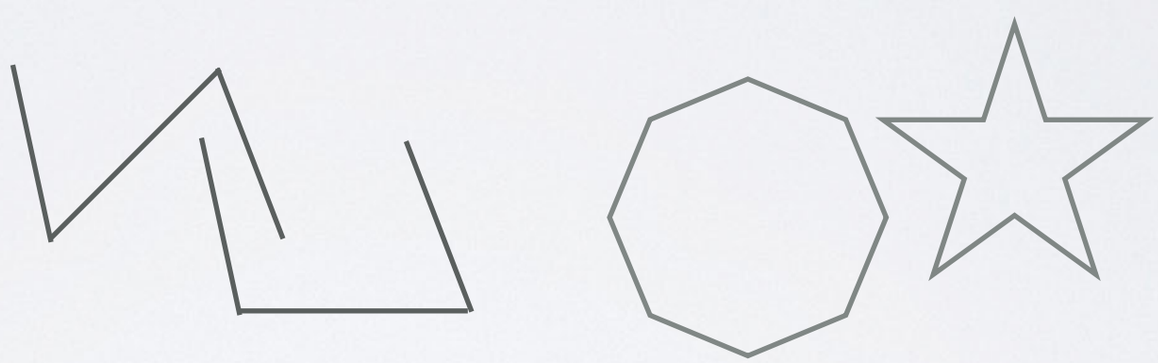
HETEROGENEOUS GEOMETRIES







How about this?

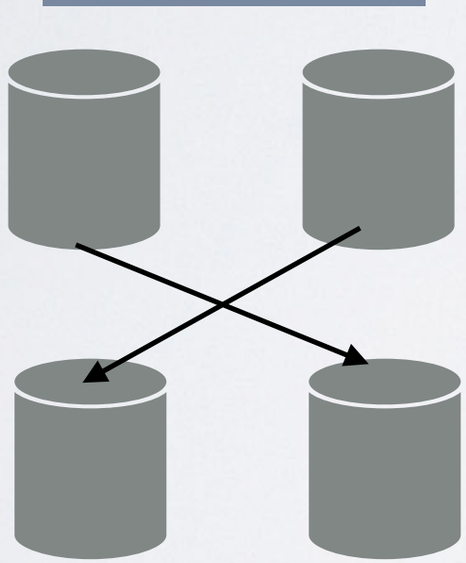


It can be even more complex
i.e., country boundaries

```
SELECT ST_GeomFromWKT ( TaxiTripRawTable.pickuppointString )
FROM TaxiTripRawTable
```

CUSTOM SERIALIZER

Byte array



Spark

Serializing . . . . . .

Byte array

De-serializing . . . . . .

GeoSpark

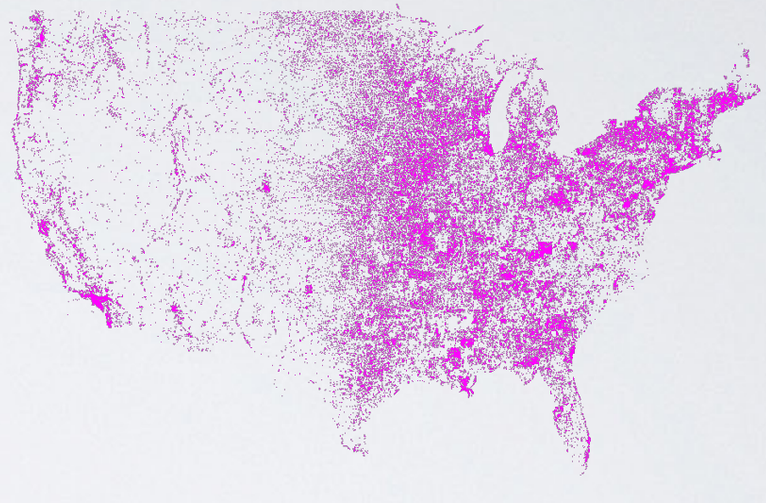
. . .

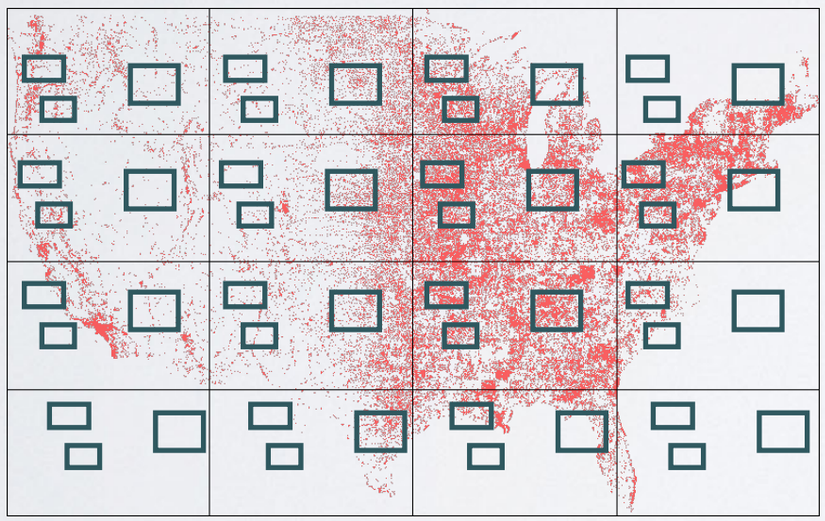
Byte array

Point, Polygon, LineString, …, Spatial index….

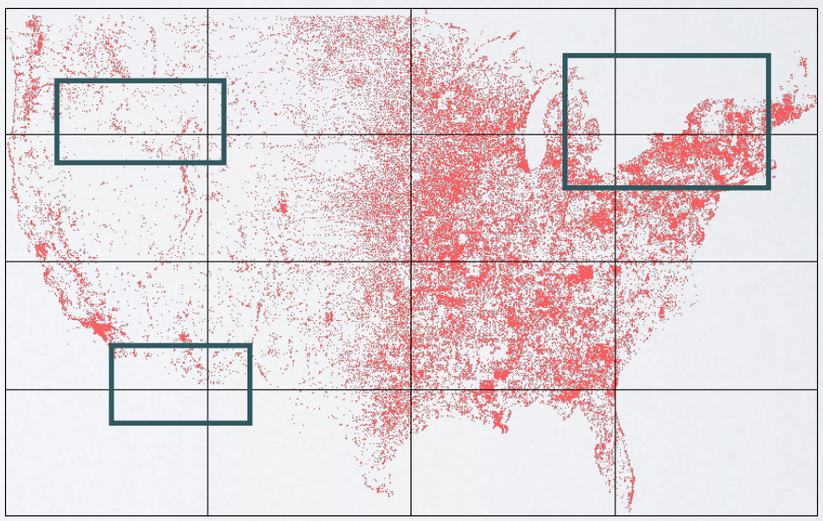
SPATIAL PARTITIONING

Range query, Join query



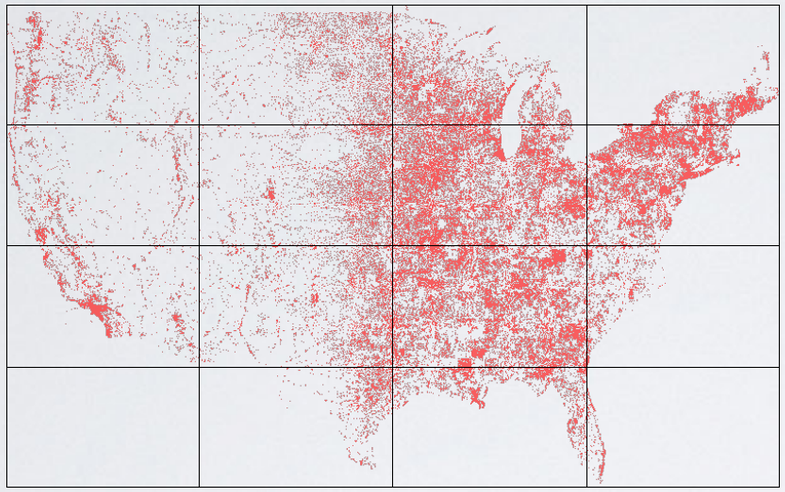


Not scalable

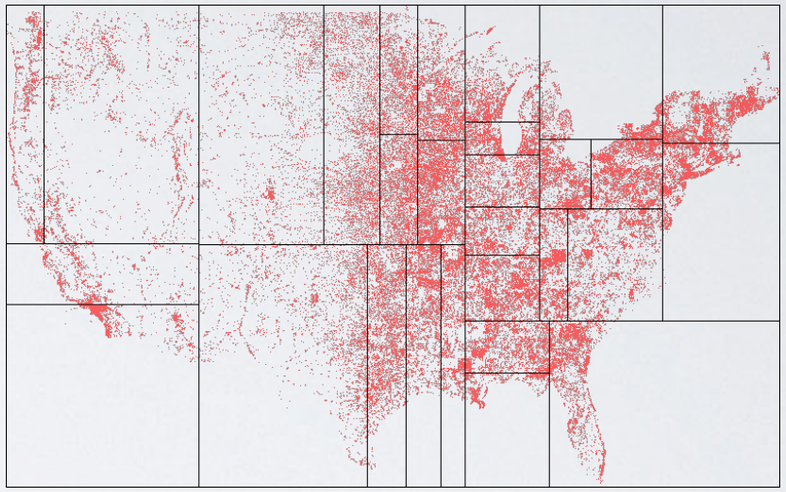


Scalable and fast

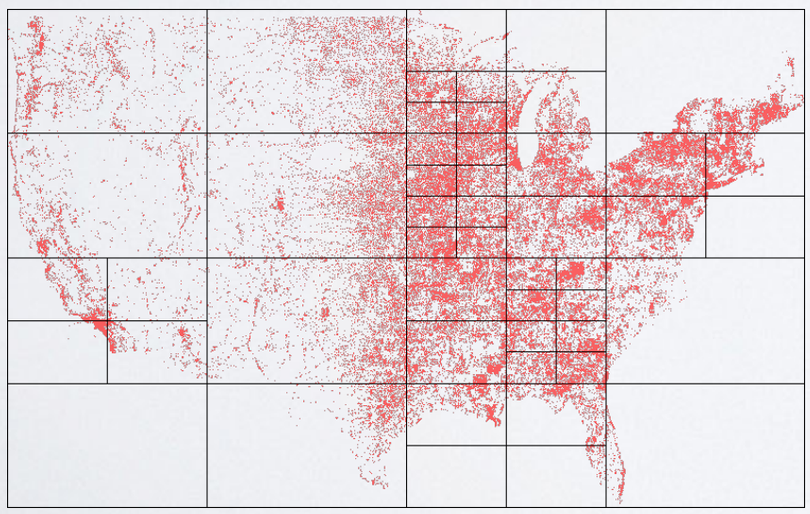
SPATIAL PARTITIONING



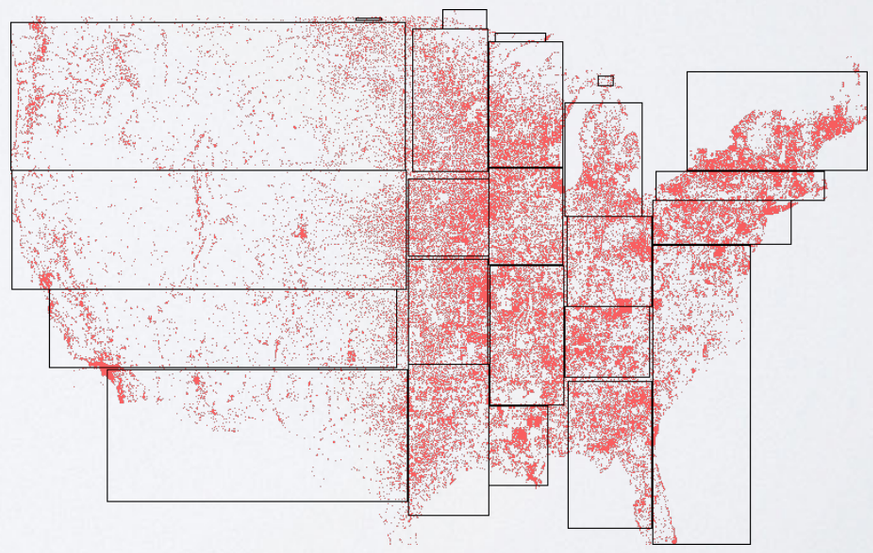
Uniform grids



KDB-Tree

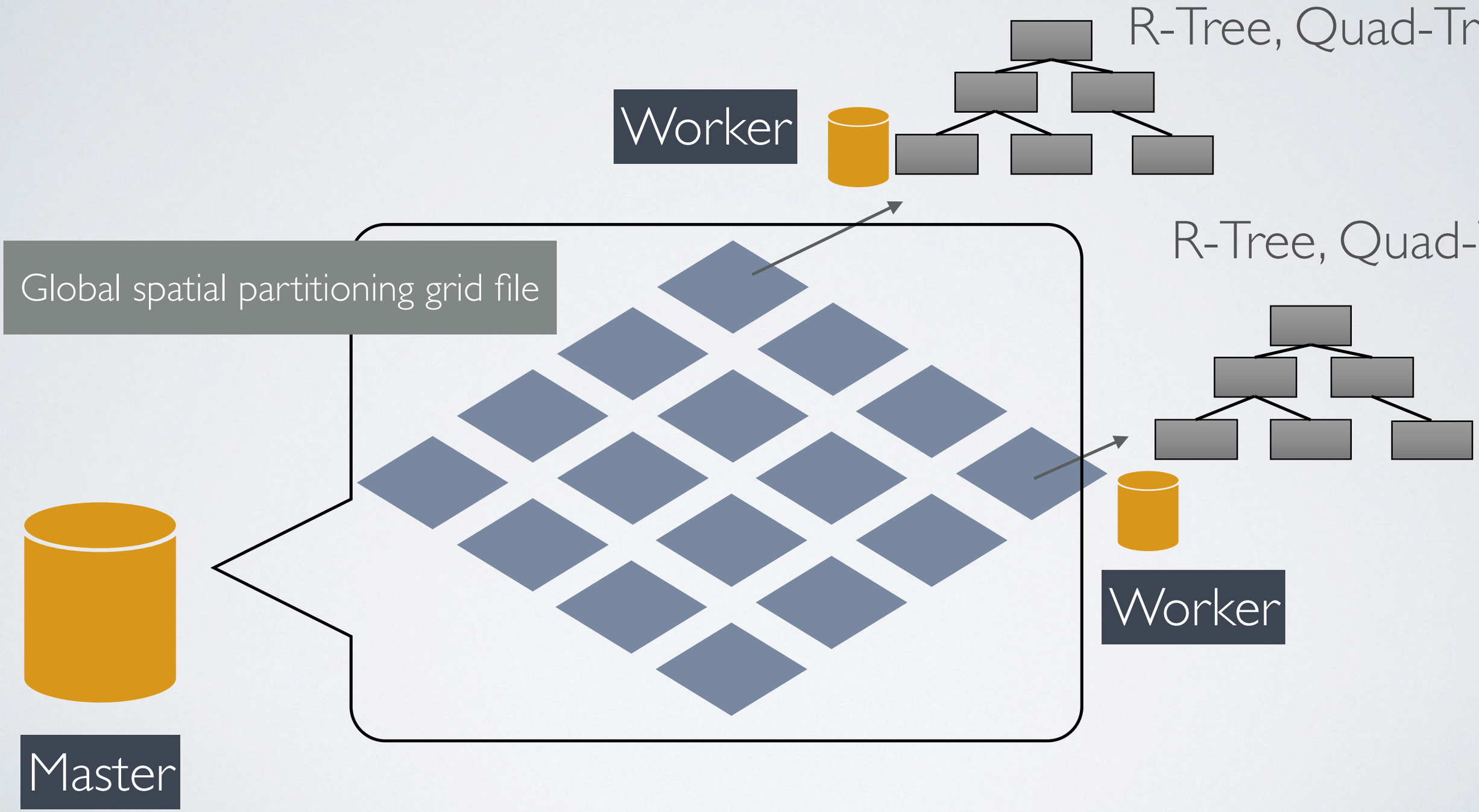


Quad-Tree



R-Tree

SPATIAL INDEXING



THIS TALK

GeoSpark

GeoSpark overview

Spatial RDD / DataFrame layer

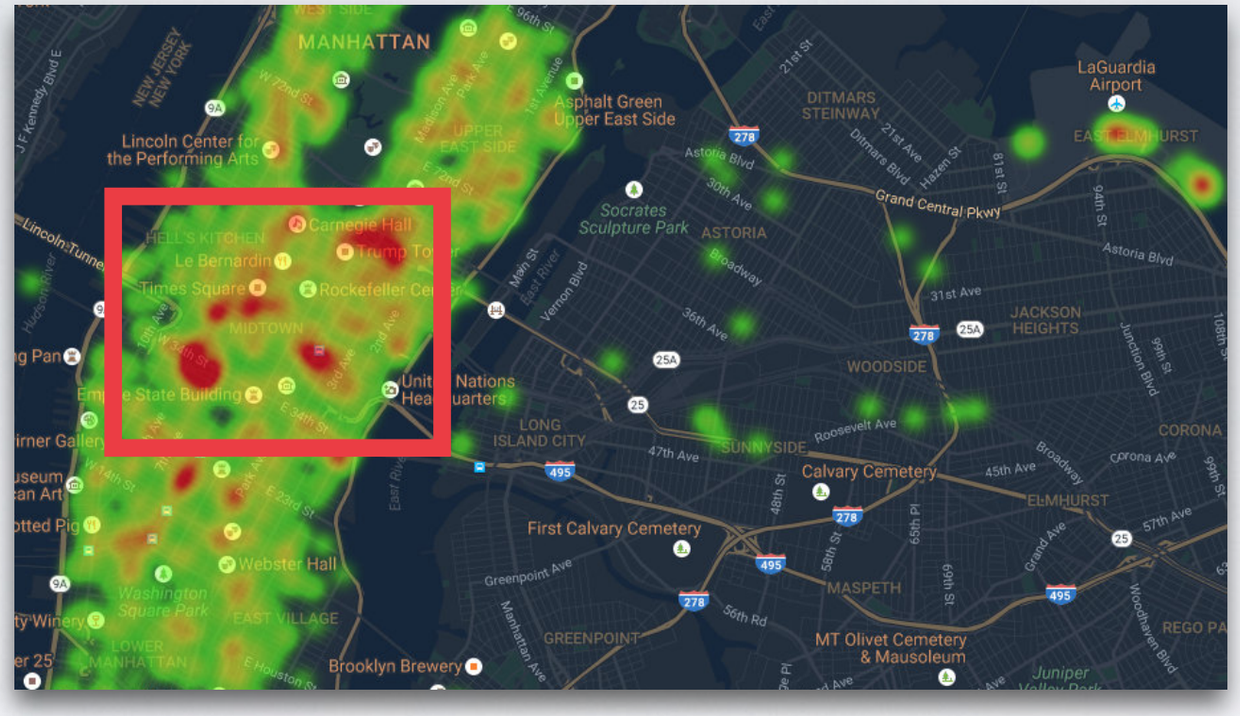
Spatial query processing layer

Query optimizer

GPU-based spatial database

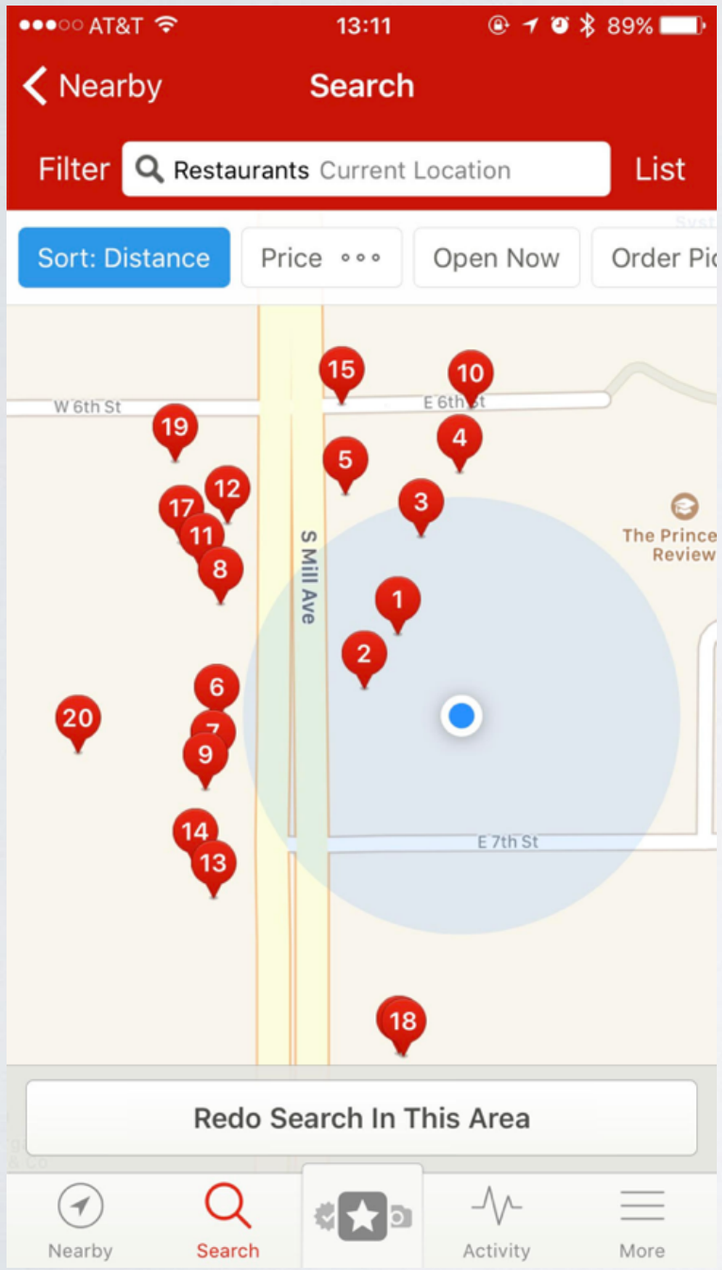
Experiments

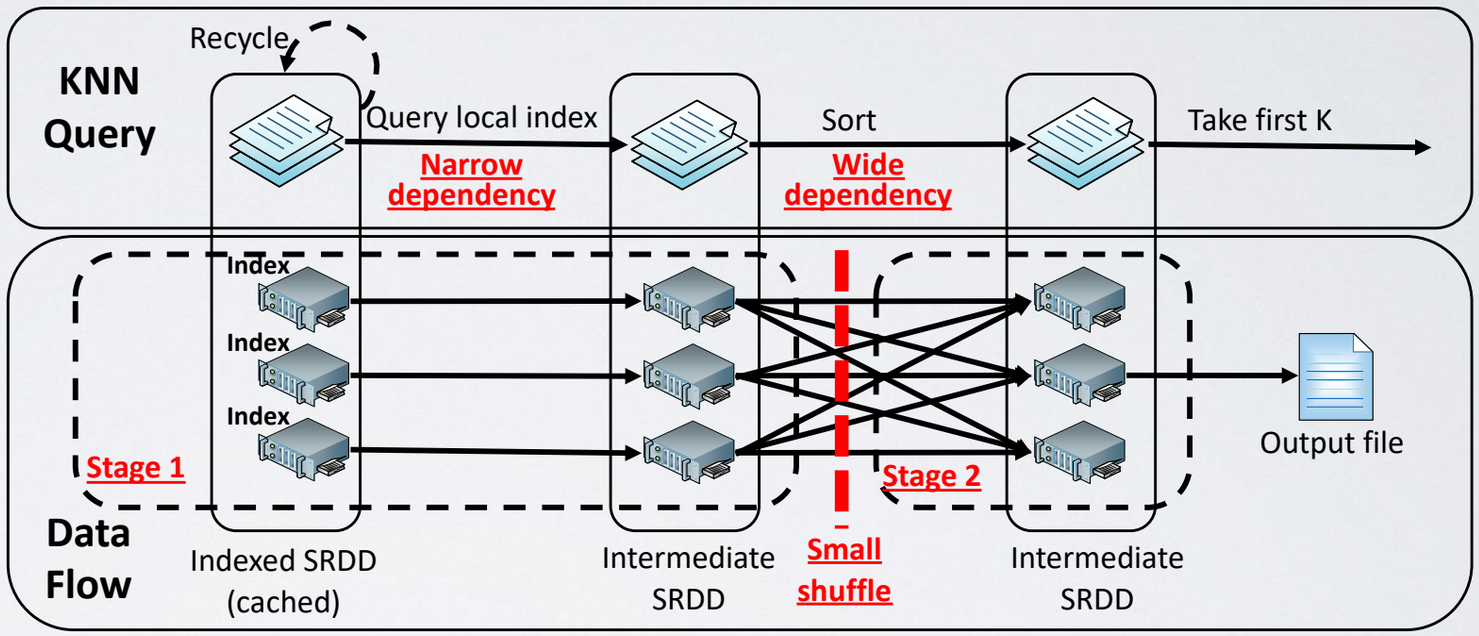
SPATIAL RANGE QUERY



```
SELECT *
FROM TaxiTripTable
WHERE ST_Contains(Manhattan, TaxiTripTable.pickuppoint)
```

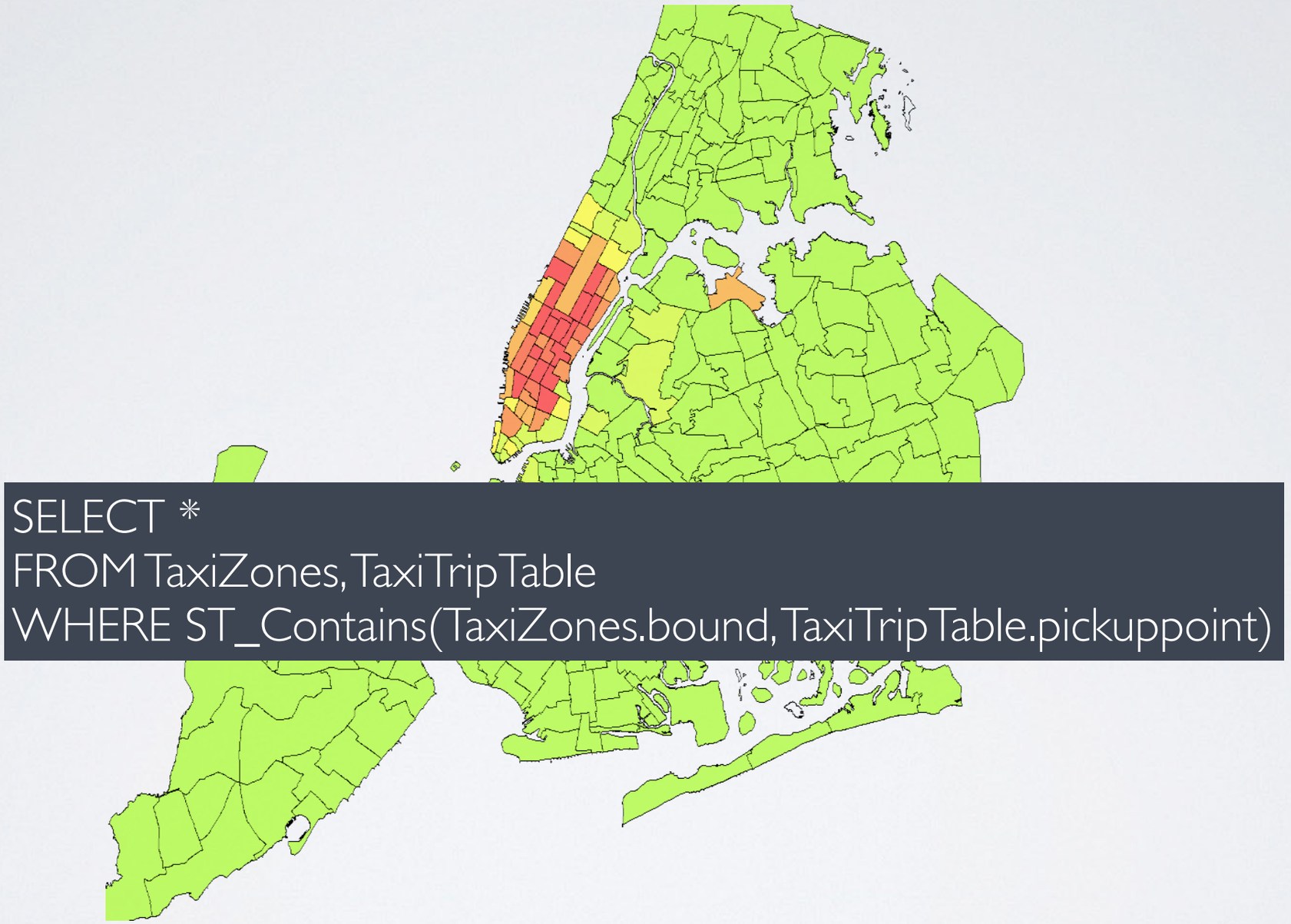
SPATIAL KNN QUERY





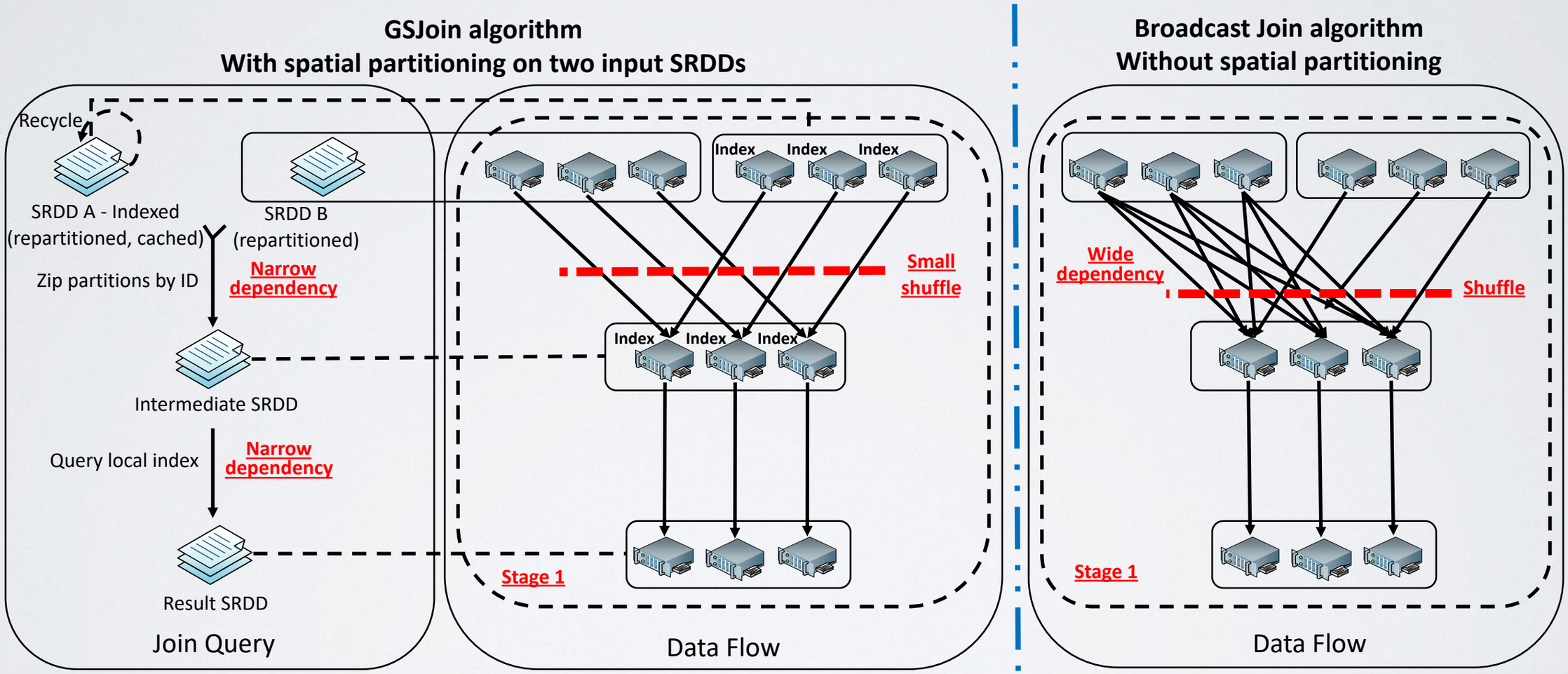

```
SELECT ST_Neighbors(MyLocation Restaurants.Locations, 20)
FROM Restaurants
```

SPATIAL JOIN QUERY



```
SELECT *
FROM TaxiZones, TaxiTripTable
WHERE ST_Contains(TaxiZones.bound, TaxiTripTable.pickuppoint)
```

SPATIAL JOIN QUERY



THIS TALK

GeoSpark

GeoSpark overview

Spatial RDD / DataFrame layer

Spatial query processing layer

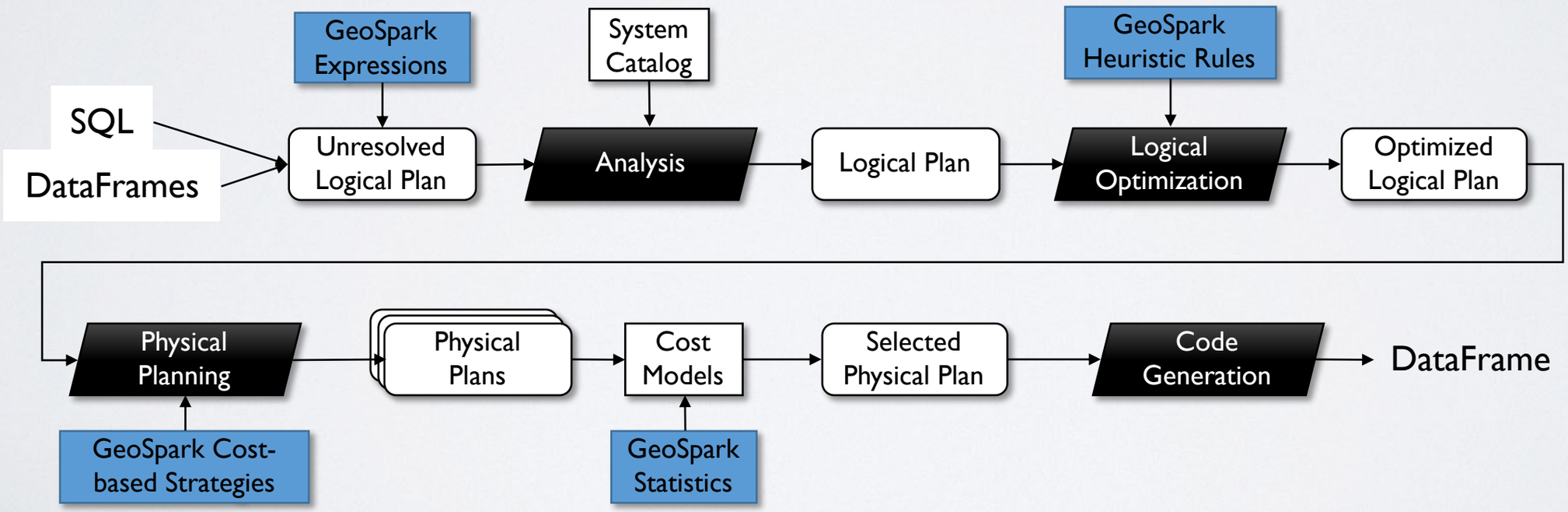
Query optimizer

GPU-based spatial database

Experiments

QUERY OPTIMIZER (V1.2.0)

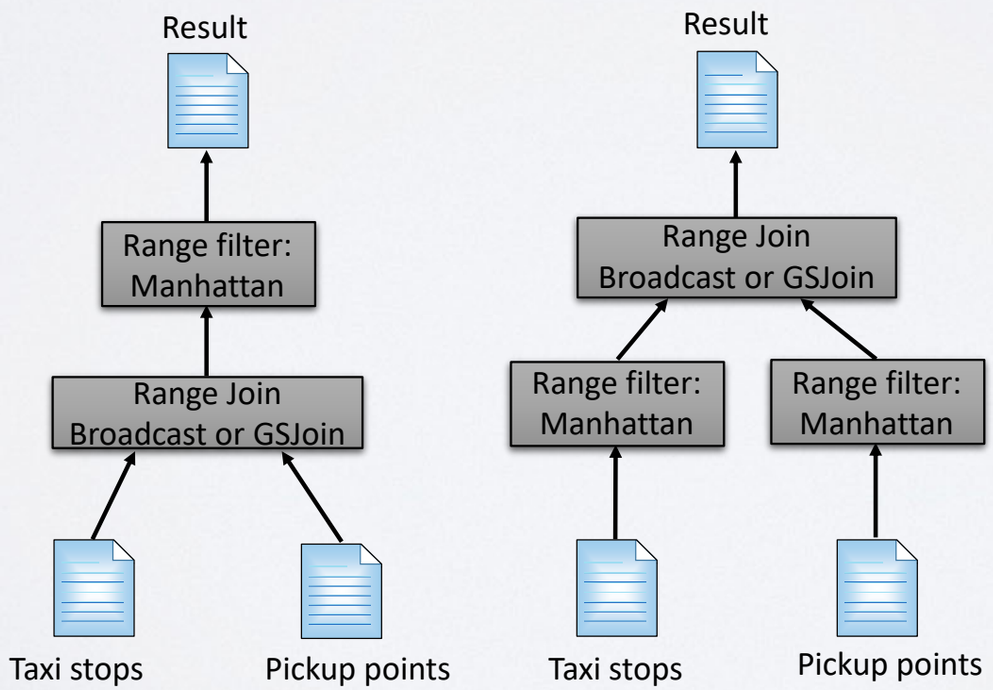
- Heuristics-Based Optimization
- Cost-Base Optimization



HEURISTICS BASED OPTIMIZATION

- Predicate pushdown

```
SELECT *
FROM TaxiStopStations, TaxiTripTable
WHERE ST_Contains(TaxiStopStations.bound, TaxiTripTable.pickuppoint)
AND ST_Contains(Manhattan, TaxiStopStations.bound)
```



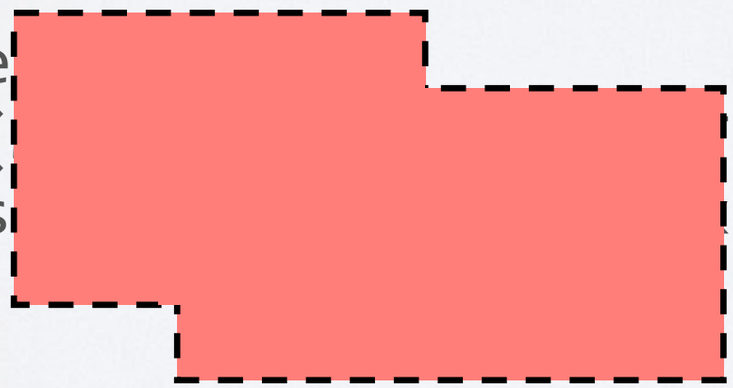
HEURISTICS BASED OPTIMIZATION

- Predicate merging

```
SELECT *
FROM TaxiTripTable
WHERE ST_Contains(Manh          TaxiTripTable.pickuppoint) AND
ST_Contains(Queens, TaxiTripTable.pickuppoint)
```

(a) AND, take the intersection

```
SELECT *
FROM TaxiTripTable
WHERE ST_Contains(                able.pickuppoint) OR
ST_Contains(Queens                uppoint)
```



(b) OR, take the union

HEURISTICS BASED OPTIMIZATION

- Intersection query rewrite

```
SELECT ST_Intersection(Lions.habitat, Zebras.habitat)
FROM Lions, Zebras
```

Cross join, slow

```
SELECT ST_Intersection(Lions.habitat, Zebras.habitat)
FROM Lions, Zebras
WHERE ST_Intersects(Lions.habitat, Zebras.habitat);
```

Optimized GeoSpark inner join, fast

COST BASED OPTIMIZATION

- Cost: based on GeoSpark statistics, MBR, count
- Index scan selection: Index scan VS DataFrame scan, based on query selectivity
- Spatial join algorithm selection: partition-wise GeoSpark join VS broadcast join

THIS TALK

GeoSpark

GeoSpark overview

Spatial RDD / DataFrame layer

Spatial query processing layer

Query optimizer

GPU-based spatial database

Experiments

	MapD	Kinetica	GeoSpark
Distributed	Yes, late 2017	Yes	Yes
SpatialSQL	Yes	Yes, limited	Yes
Compact in-mem geometry, index	No	No	Yes
Distributed spatial index	No, nested loop	No	Yes, dist. Quad-Tree, R-Tree
Distributed spatial data partitioning	No, still hash or round-robin	No	Yes, 4 spatial partition methods
Opt. distributed spatial join	No	No	Yes
Spatial query optimizer	No	No	Yes, HBO, CBO
Fault tolerance	No, fail right away	Yes	Yes, RDD lineage
SQL CodeGen	Yes	No	Yes
Streaming	Yes	Yes	Yes
Storage system	Yes	Yes	No, but + MapD, +Kinetica

REFERENCE

- MapD RoadMap: https://github.com/mapd/mapd-core/blob/master/ROADMAP.md

- Kinetica: https://www.kinetica.com/product/faq/

THIS TALK

GeoSpark

GeoSpark overview

Spatial RDD / DataFrame layer

Spatial query processing layer

Query optimizer

GPU-based spatial database

Experiments

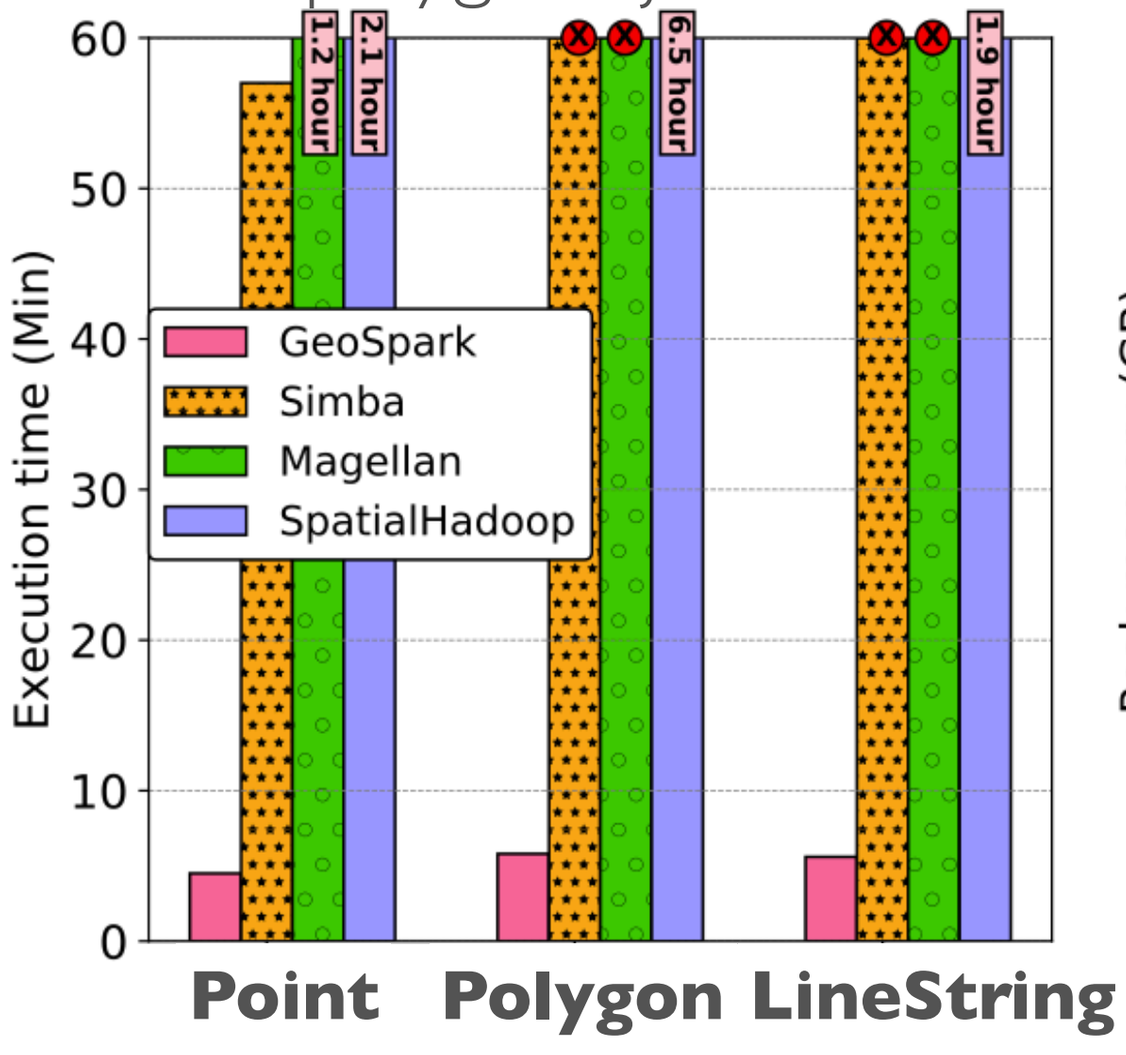
JOIN QUERY

1.3 billion points join 171 thousand polygons

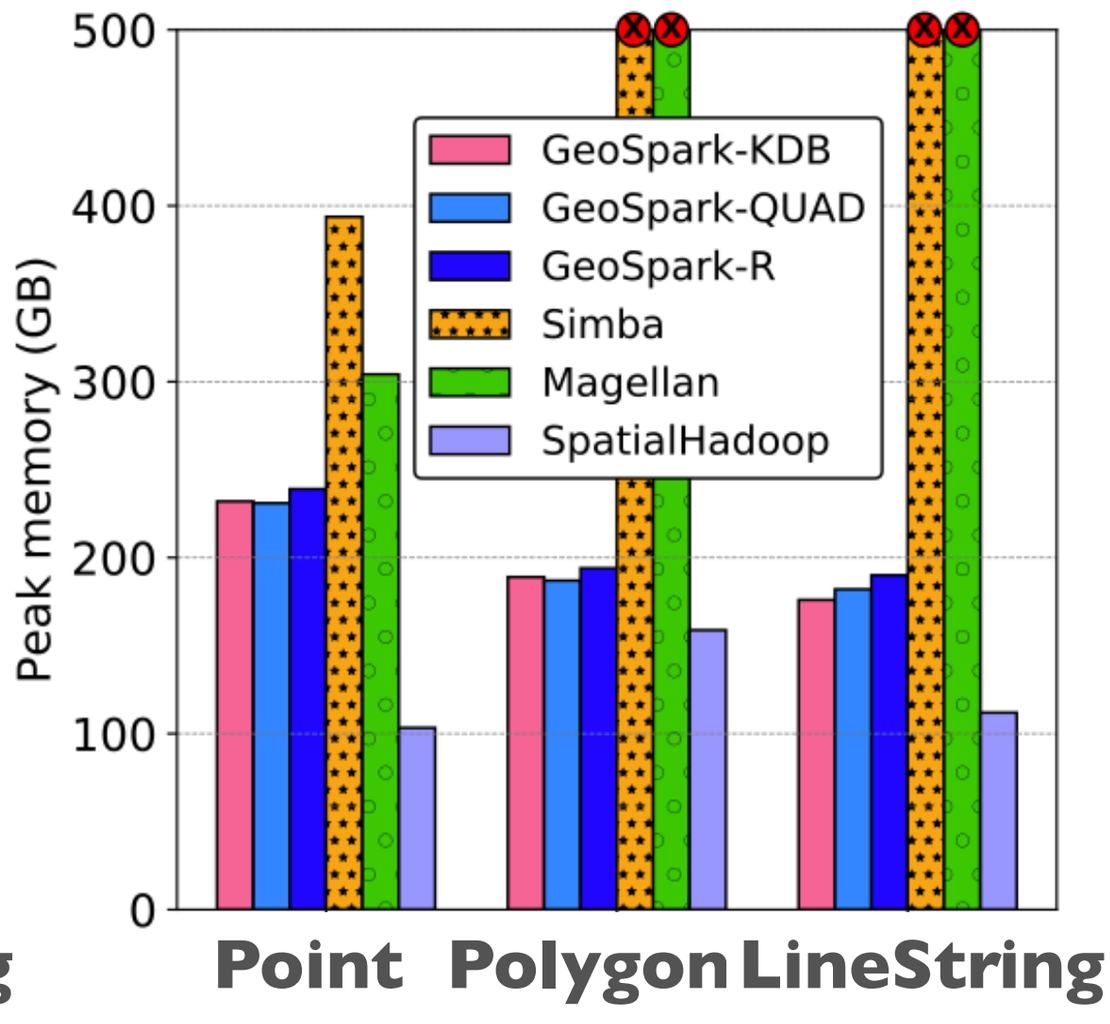
72.7 million line strings join 171 thousand polygons

263 million polygons join 171 thousand polygons

4 machines



(a) Execution time



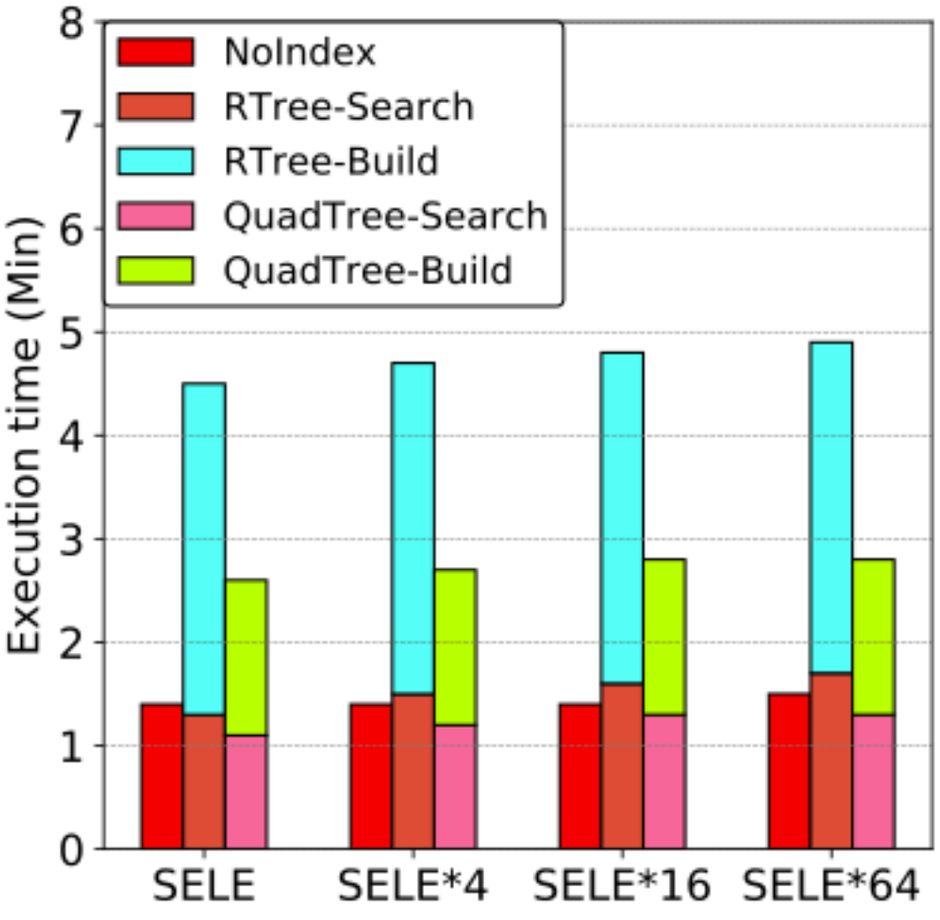
(b) Peak memory utilization

CONCLUSION

- GeoSpark is the fastest approach compared to other systems
- For join query, GeoSpark has the least memory because it can make Spark quickly serialize/ deserialize data without having too much intermediate data be sticking in memory

QUESTIONS?

THE IMPACT OF INDEX



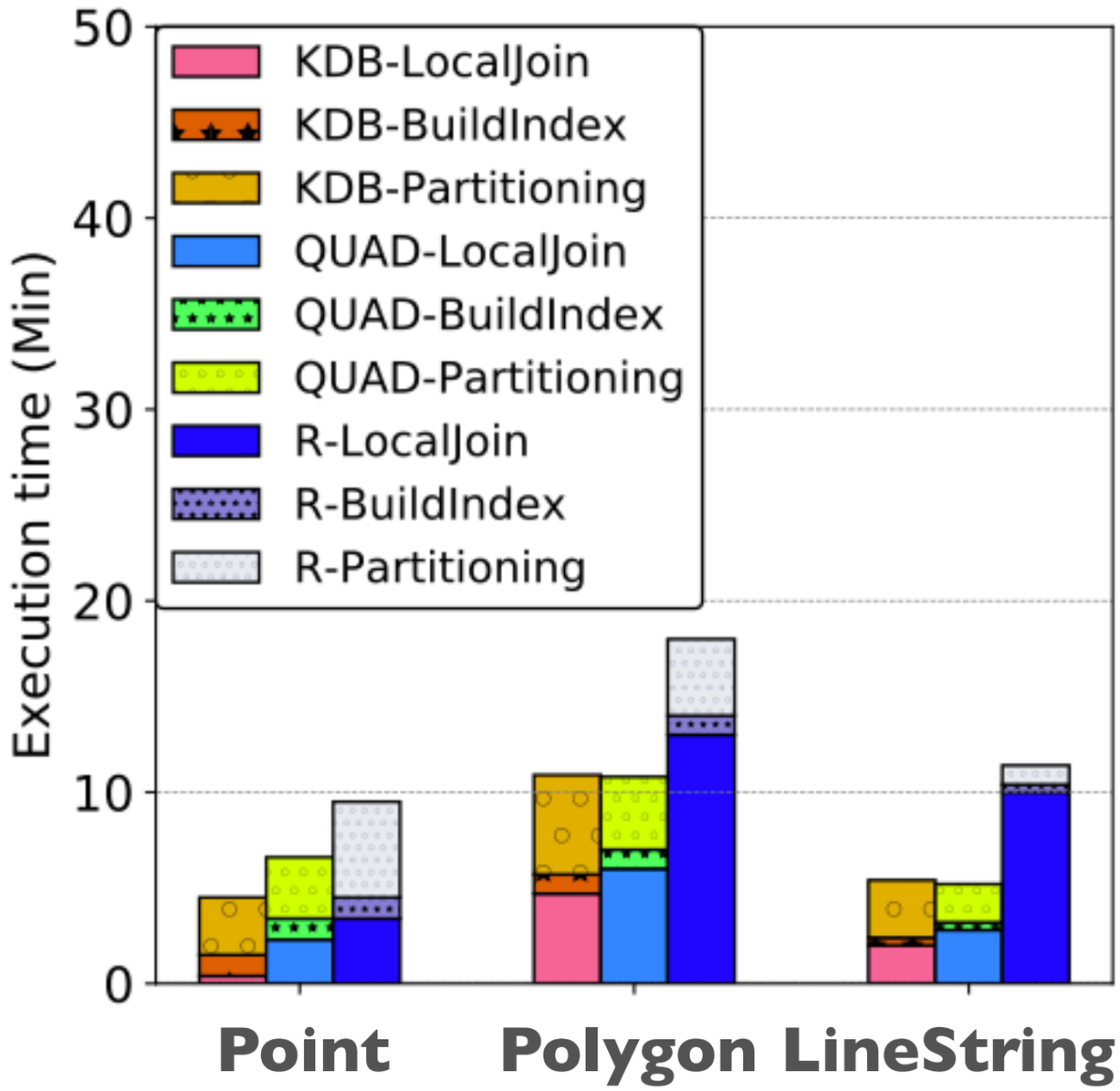
Point data

Polygon data

CONCLUSION 2

- Spatial index is only helpful when prune complex shapes because of filter and refine model

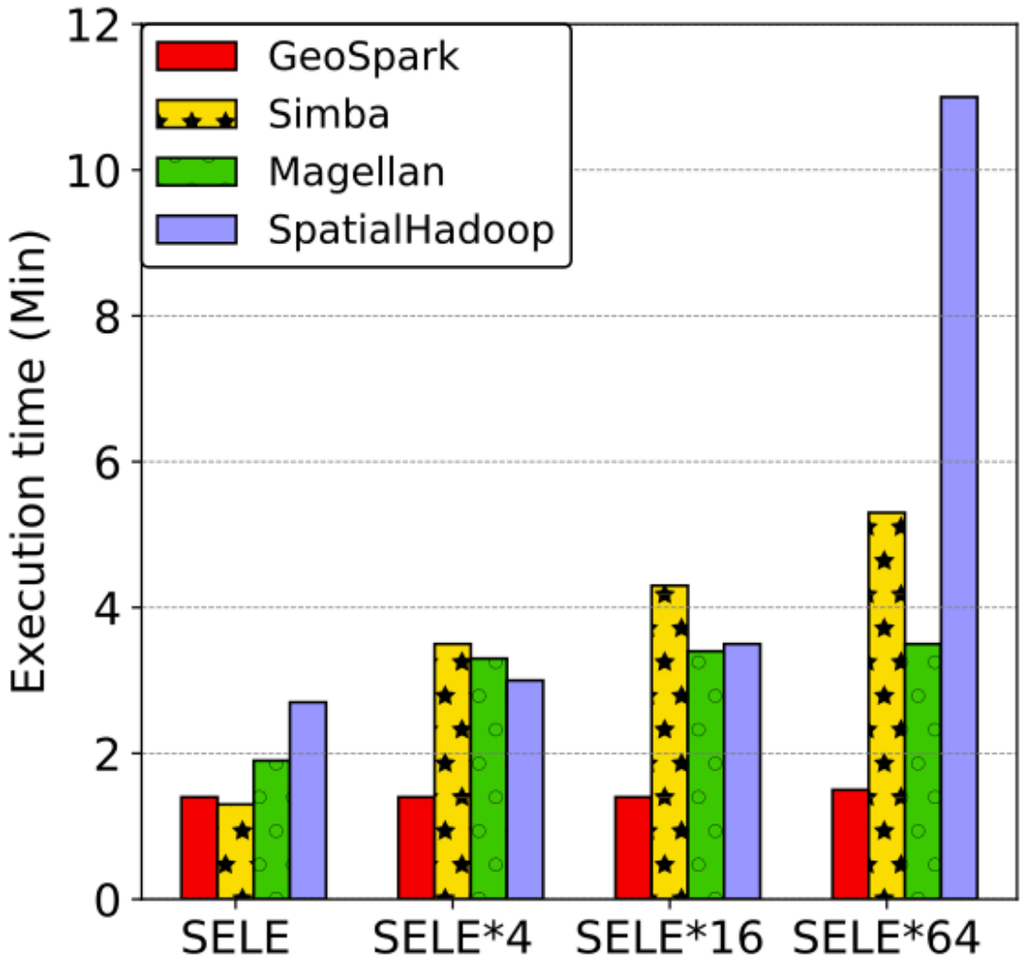
THE IMPACT OF SPATIAL PARTITIONING



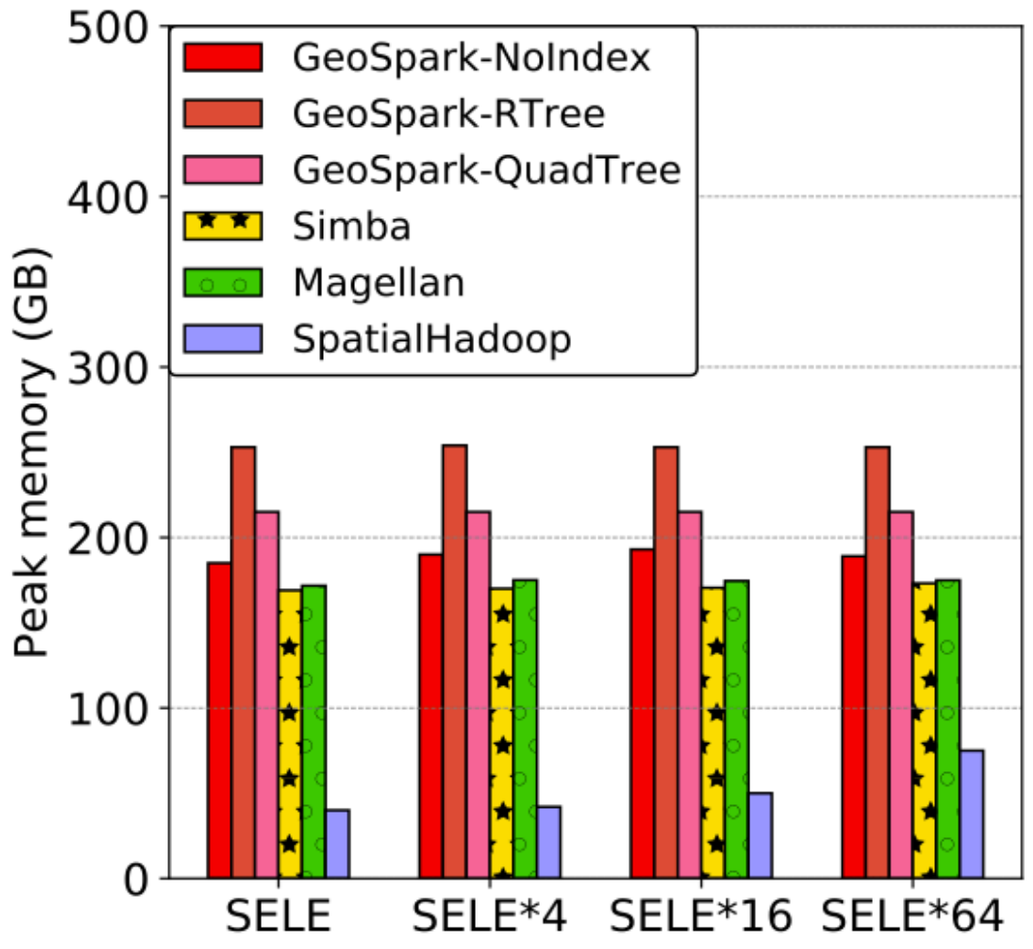
CONCLUSION 3

- KDB-Tree partition is the most load-balanced
- Quad-Tree is better
- R-Tree is the worst but better than uniform grids

POINT RANGE

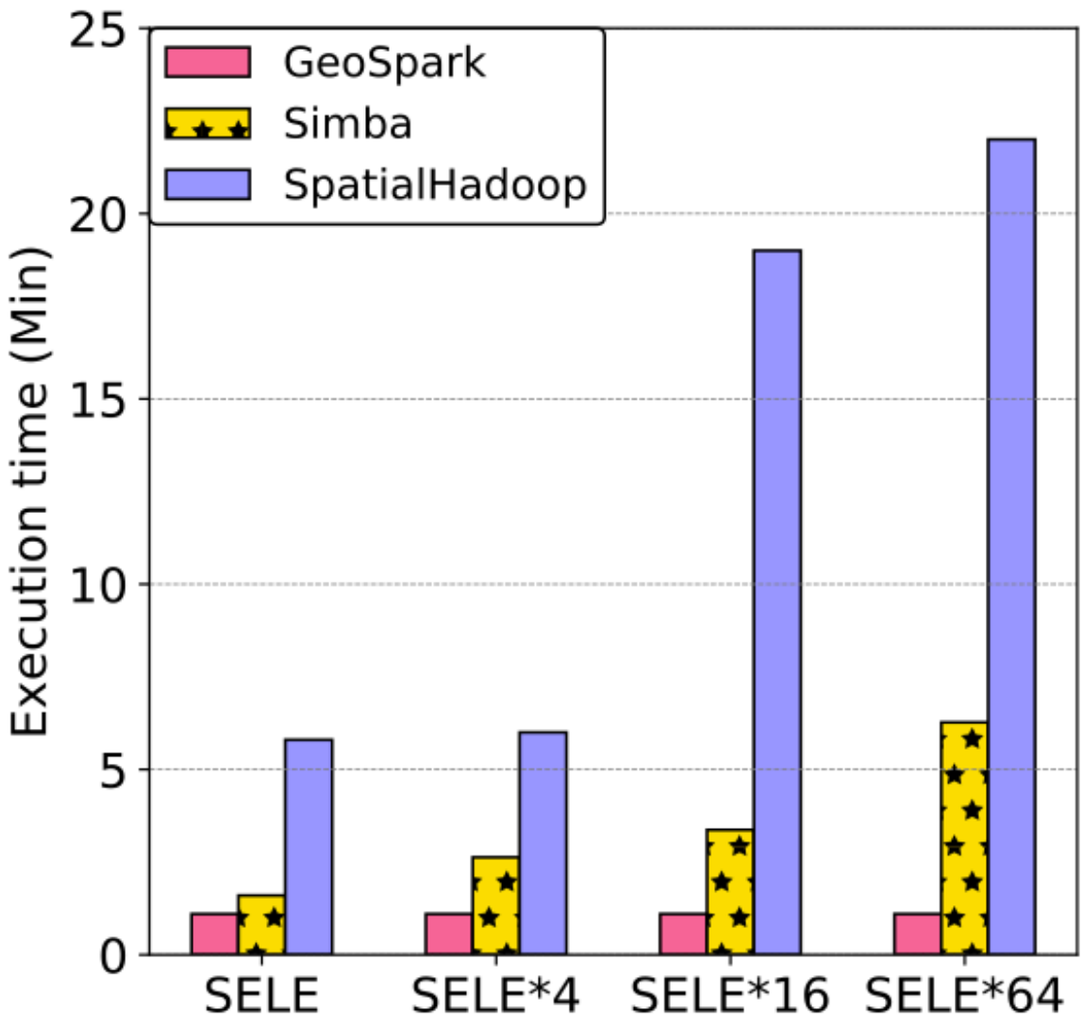


(a) Execution time

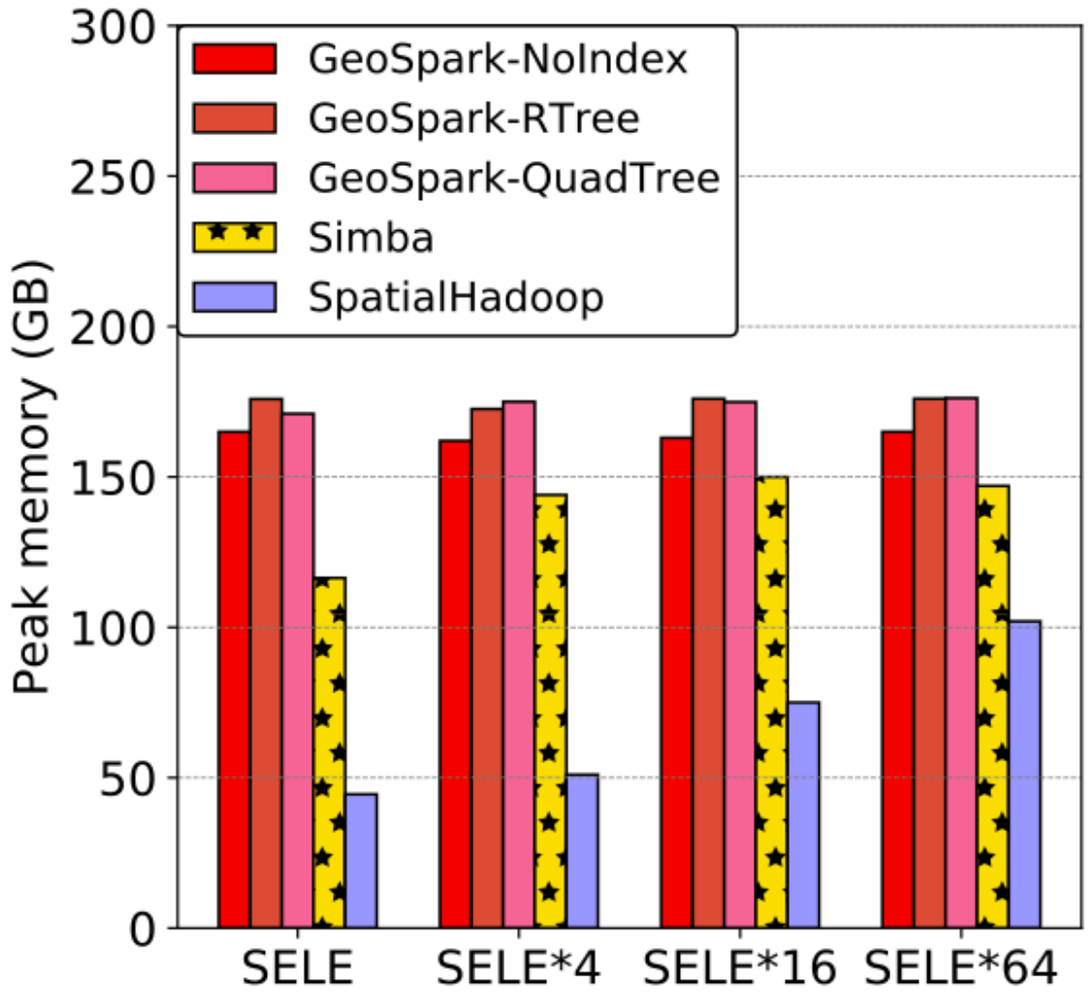


(b) Peak memory utilization

POLYGON RANGE

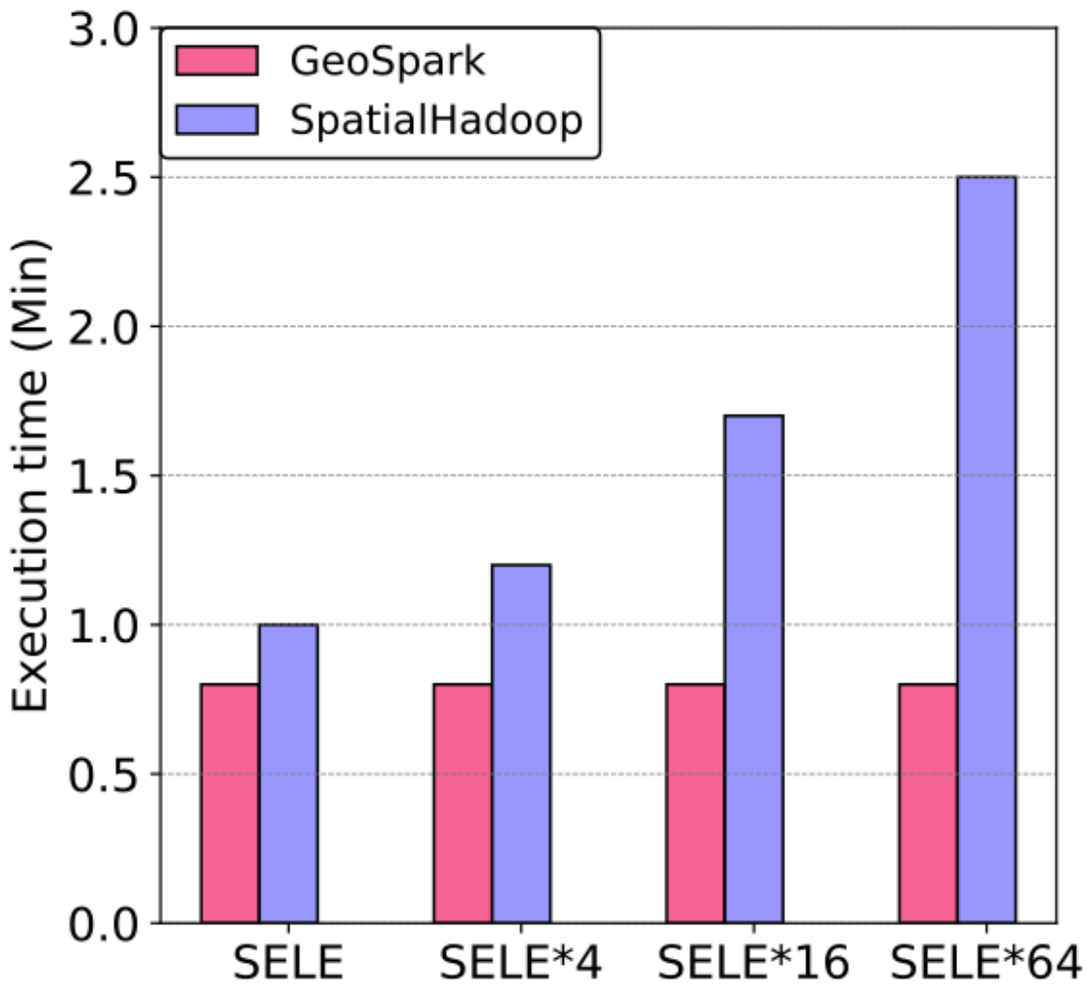


(a) Execution time

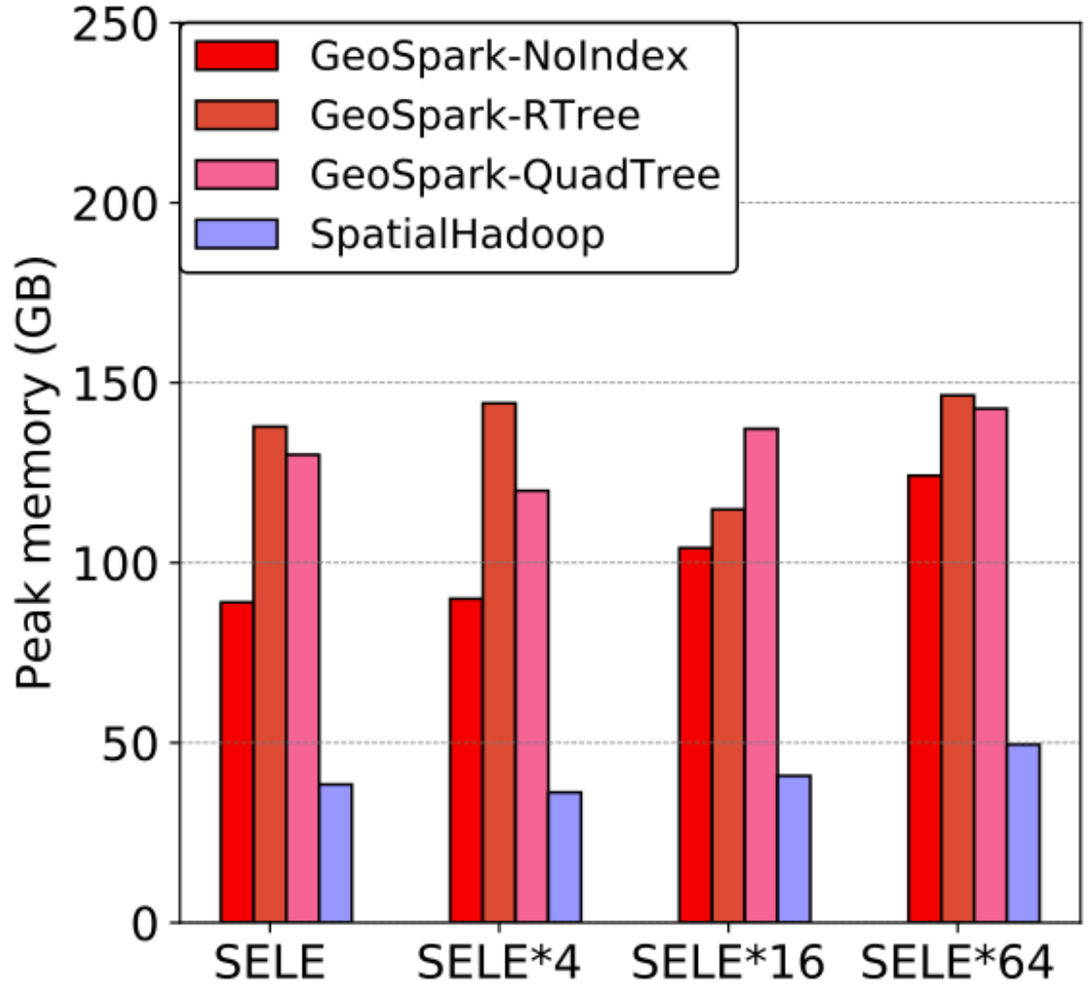


(b) Peak memory utilization

LINE STRING RANGE



(a) Execution time



(b) Peak memory utilization