10th International Conference on Model and Data Engineering (**MEDI’2021**)
21 - 23 June 2021, Tallinn, Estonia

Enhancing Sedona (formerly GeoSpark) with Efficient kNearest Neighbor Join Processing

Francisco García-García[1], Antonio Corral[1], Luis Iribarne[1], and Michael Vassilakopoulos[2]

[1]Applied Computing Group, Dept. of Informatics, University of Almería, Spain
[2]DaSE Lab, Dept. of Electrical and Computer Engineering, University of Thessaly, Volos, Greece

UNIVERSITY OF THESSALY

UNIVERSIDAD DE ALMERÍA

GOBIERNO DE ESPAÑA

MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

TIN2017-83964-R

“Study of a holistic approach for the interoperability and coexistence of dynamic systems: Implication in Smart Cities models”

***Speaker**

Outline

Problem and Motivation
- k Nearest Neighbor Join Query (kNNJQ)
- Distributed Spatial Analytics Systems

k Nearest Neighbor Join Query in Apache Sedona
- Sedona
- kNNJQ algorithm in Sedona

Experimentation
- Real World Datasets
- Spatial Partitioning Techniques
- Use of Local Indices
- Number of executors

Conclusions and Future Work

1 Problem and Motivation

UNIVERSIDAD DE ALMERÍA

What is a k Nearest Neighbor Join Query (kNNJQ)?

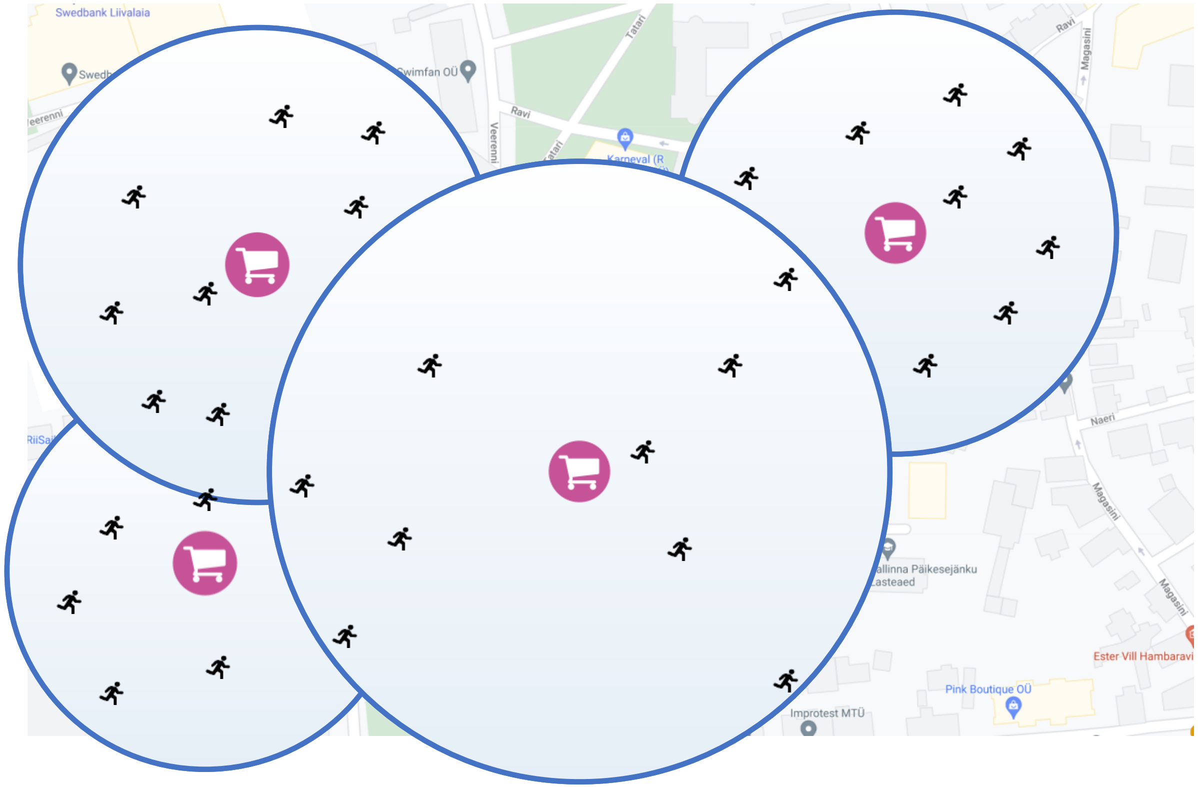
In **mobile location services**:

Locations of **shopping centers**

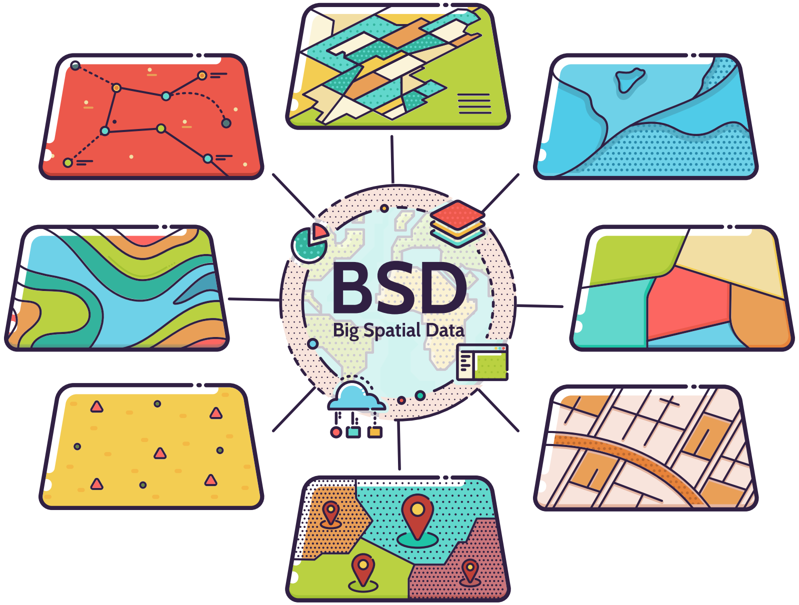
Positions of possible **customers** (smartphone with GPS)

*“find the **10** nearest possible customers to each shopping center for sending an advertising SMS about a fashion brand available there.”*

What is a k Nearest Neighbor Join Query (kNNJQ)?



What if we are talking about Big Spatial Data?



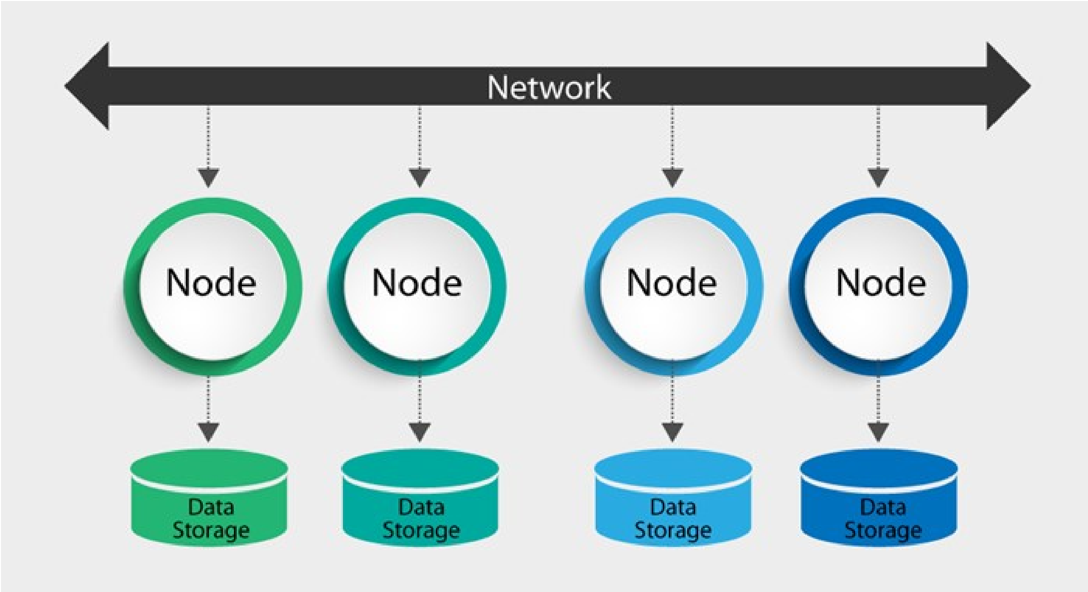
Problem and Motivation

There is a huge increase of the volume of available spatial data world-wide

How to deal with that?

<u>Distributed Spatial Analytics Systems</u>

- Shared-nothing clusters
- **Apache Spark**
  - In memory processing
  - less disk writes and reads than Hadoop



hadoop

APACHE Spark

Problem and Motivation

Pandey, V., Kipf, A., Neumann, T., Kemper, A.: How good are modern spatial analytics systems? PVLDB **11**(11), 1661–1673 (2018).

Experimental evaluation

- SpatialSpark
- Simba
- LocationSpark (Winner for kNNJQ)
- GeoSpark (aka Sedona)

Apache Sedona

- Almost complete (data types and queries)
- Best performance
- Actively under development
- It does not support kNNJQ

Problem and Motivation

Simba

- **Block nested loop kNNJ**
  - BKJSpark-N
  - BKJSpark-R (R-tree)
- **VKJSpark** (Voronoi)
- **ZKJSpark** (Z-values)
- **RKJSpark** (R-tree)

LocationSpark

WINNER

- **Block nested loop kNNJ**
  - nestR-tree
  - nestQtree
- **sfcurve (Hilbert-curve)**
- **pgbjk**
- **spitfire**

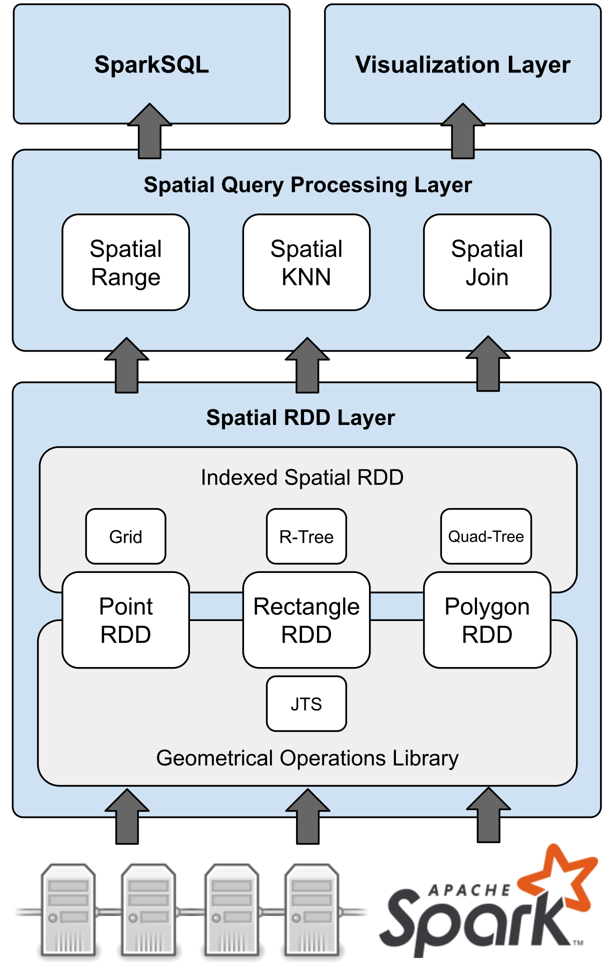
2 k Nearest Neighbor Join Query in Apache Sedona

UNIVERSIDAD DE ALMERÍA

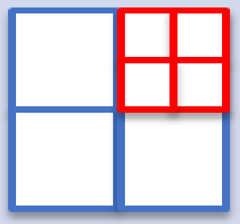
Apache Sedona

Extends Apache Spark and SparkSQL with **native support for spatial data.**

- **Spatial RDD Layer**
  - Spatial Data types (Spatial RDD)
  - Basic Geometric operations (JTS)
  - Global partitioning
    - Grid, R-tree, Quadtree, kDB-tree
  - Local Indices
- **Spatial Query Processing Layer**
  - Spatial operations
- **Visualization Layer**
  - GeoSparkViz
  - GeoSparkSim (urban traffic sim)

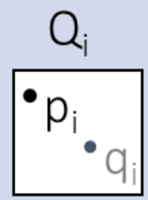


kNNJQ algorithm in Apache Sedona. P x Q where |P| < |Q|



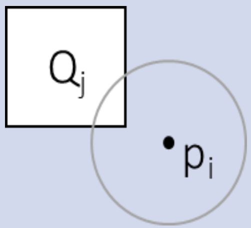
1. Information Distribution

- **It partitions Q using:**
  - Grid, R-tree, Quadtree, kDB-tree.
  - Optional **R-tree local index** over each partition.
- **Partitions P over the partitions of Q.**



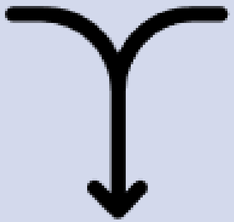
2. Bin kNNJ

- **Bin Spatial-Join of PxQ with kNNQ as join operand.**
- **Local kNNQ**
  - No-Index: Plane-sweep
  - With-Index: R-tree
- **Completness check**
  - Final kNN lists
  - Non final kNN lists



3. kNNJ on Overlapping Partitions

- **Spatial range join** using the **distance** of each non-final point of P to its **k-th nearest neighbor.**
- It can result in **multiple kNN lists per point.**



4. Merge Results

- It **AGGREGATES** and merges non final kNN lists from steps 2 and 3.
- It does an **UNION** between final kNN list from step 2 and previous results.

kNNJQ algorithm in Apache Sedona.

Important!!
Avoid wide dependencies

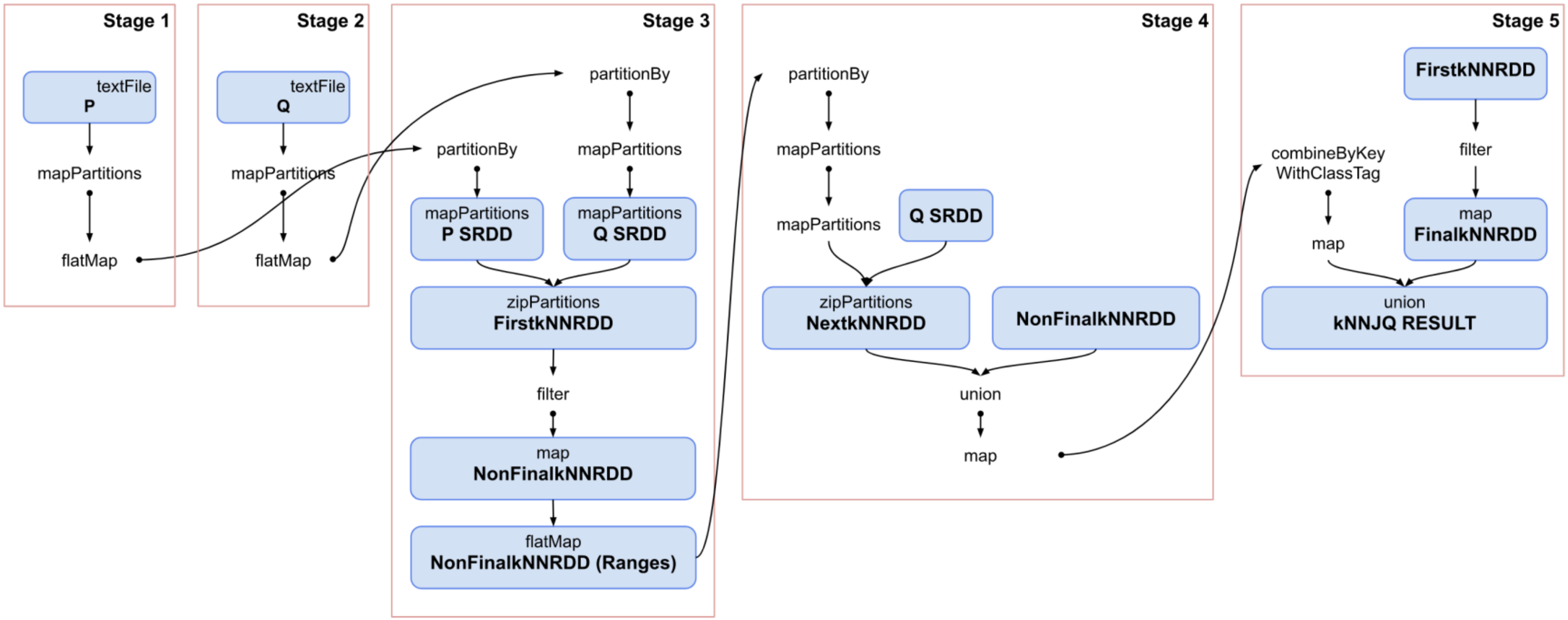


Fig. 1. Spark DAG for *k*NNJQ Algorithm in *Sedona*.

3 Experimentation

Experimentation

- **Performance metrics,**
  - Total Execution Time,
  - Total Shuffled Data,
    - Data redistributed across partitions that may or may not move across processes, executors or nodes
  - Peak Execution Memory
    - Highest execution memory of all the tasks of a specific job
- **Configuration parameters,**

Parameter	Values (default)
k for kNNJQ	(25), 50, 75, 100
Partitioning technique	Grid – G, R-tree – R, Quadtree – Q, (kDB-tree – KD)
Local Index	No Index, R-tree
number of executors (η)	2, 4, 6, 8, 10, (12)

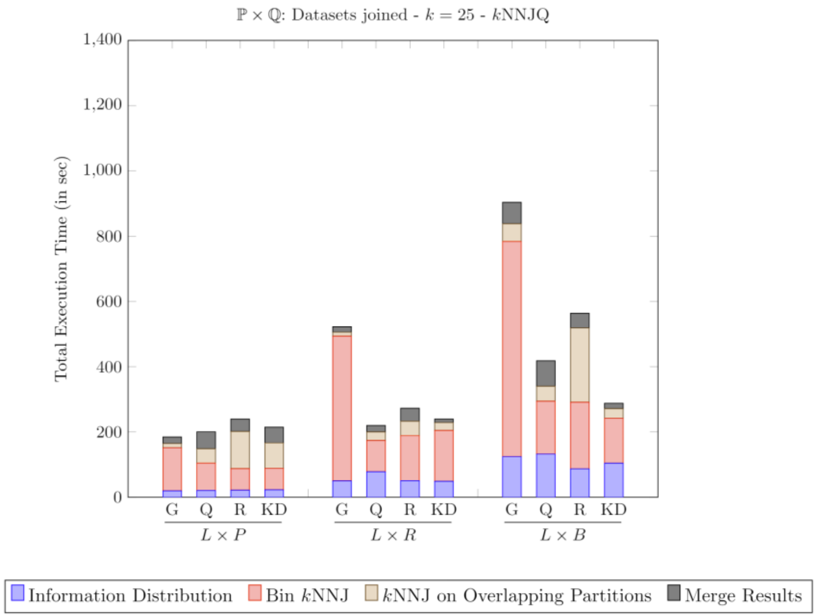
Experimentation

- **Real datasets** from **OpenStreetMap**:
  - **BUILDINGS (B)** which contains **115M** records of buildings,
  - **ROADS (R)** which contains **72M** records of roads,
  - **PARKS (P)** which contains **10M** records of parks and green areas,
  - and **LAKES (L)** which contains **8.4M** points of water areas,

Experimentation

- **Setup**
  - Cluster of **7 nodes** on an **OpenStack** environment.
    - 1 Master Node
    - 6 Slave Nodes
  - Each node has **8 vCPU** with **64GB** of main memory running **Linux** operating systems with:
    - **Hadoop 2.7.1.2.3**
    - **Spark 2.4.7**

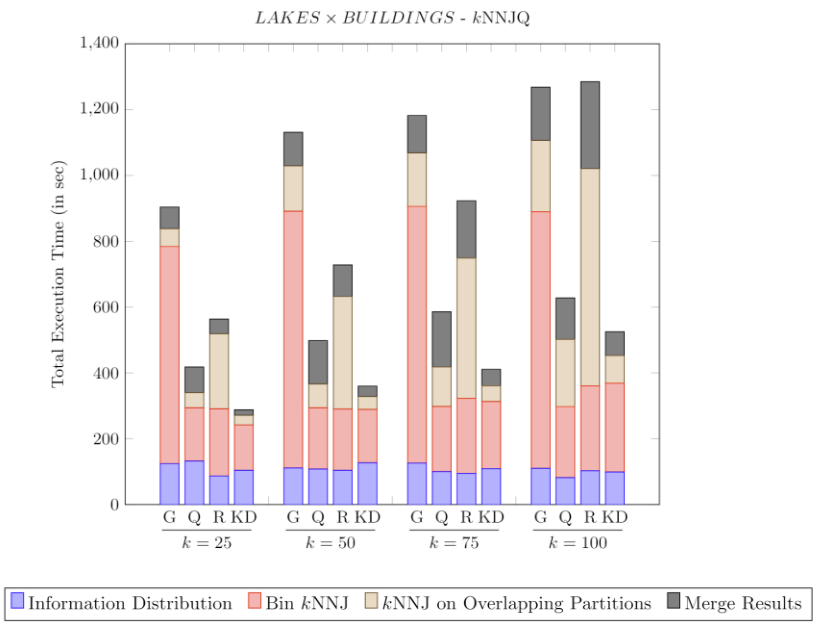
Experimentation
Different Spatial Partitioning Techniques (Dataset sizes)



Execution time grows as dataset size increases

- **Grid**
  - **Highest execution times**
    - Uniform -> Real data
    - Skew problems (*Bin kNNJ*)
- **R-tree**
  - **Higher time values** (*kNNJ on Overlapping Partitions*)
    - Non-regular partitions
- **Quadtree and kDB-tree**
  - **Winners**, especially **kDB-tree.**
  - **kDB-tree has more balanced partitions**
    - Quadtree -> spatial properties
    - kDB-tree -> number of points

Experimentation
Different Spatial Partitioning Techniques (k values)

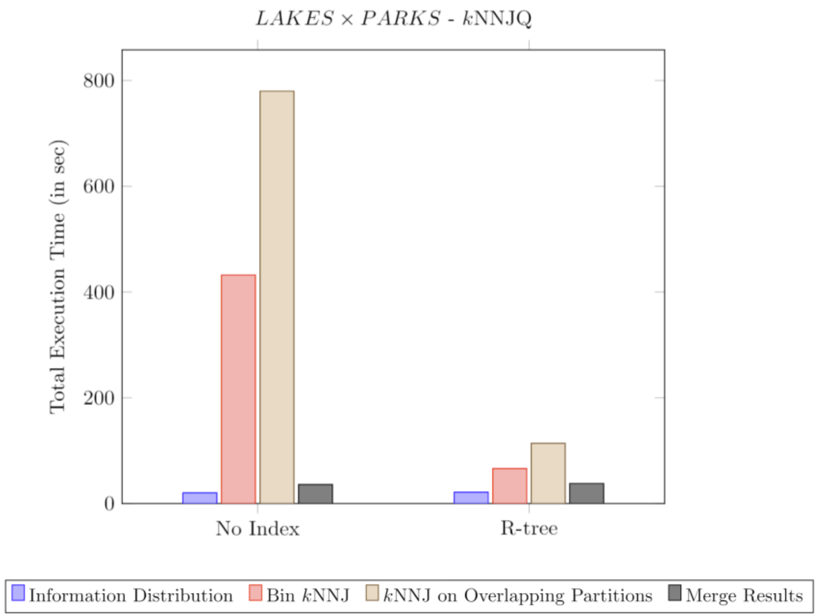


The larger the k value, the larger the execution time

- **Grid**
  - **Stable growth of the execution time**
    - Step in k = 50 -> uniform
- **R-tree**
  - **Higher increase** (*kNNJ on Overlapping Partitions* vs *Bin kNNJ*)
    - Number of overlaps grows significantly -> more kNN lists
- **Quadtree and kDB-tree**
  - **Winners**, especially **kDB-tree.**
  - **kDB-tree has better data distribution**
    - Reduced number of overlapping partitions -> small increase in Step 3.
  - **Quadtree has multiple overlapping partitions per point**
    - Increase in *Merge Results*

Experimentation

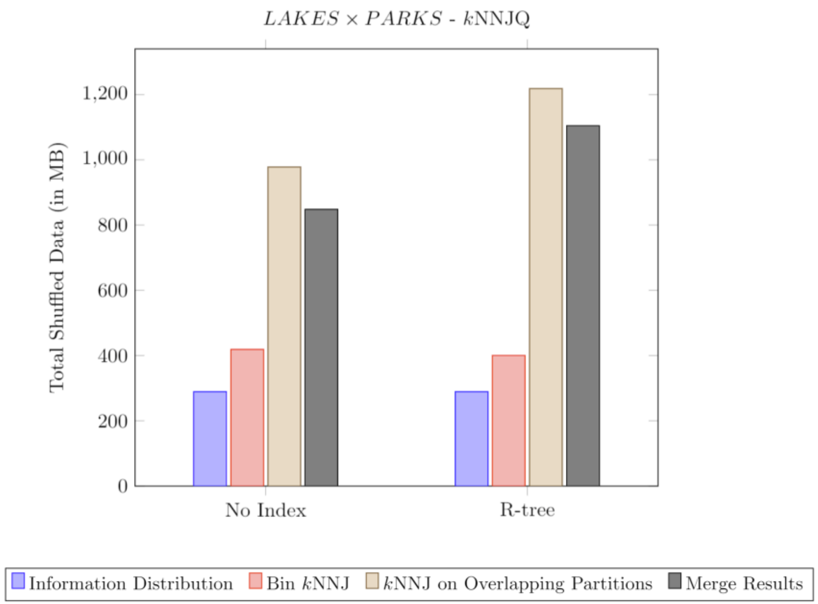
Indices over best spatial partitioning technique kDB-tree (execution time)



- **Similar execution time** of the ***Information Distribution*** and ***Merge Results*** steps
- **Proportional execution** time between **Bin kNNJ** and **kNNJ on Overlapping Partitions**
- **R-tree** index is **6x faster** than **No Index**
- **No Index (Plane sweep) has better performance than brute force.**

Experimentation

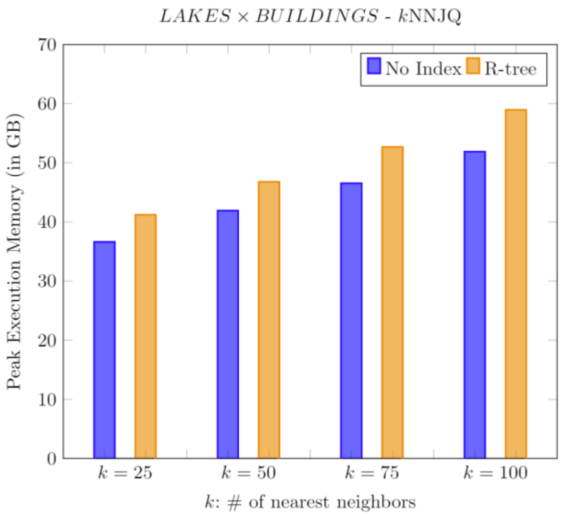
Indices over best spatial partitioning technique kDB-tree (Total Shuffled Data)



- **Similar values for each step of the algorithm**
- **Same values** for the **Information Distribution** and **Bin kNNJ**
  - Same partitioning and number of generated knn lists
- **Higher values for R-tree** for the **kNNJ on Overlapping Partitions** and **Merge Results**
  - **Optimization in the plane-sweep** algorithm reduces number of kNN lists
  - **R-tree** uses **built-in spatial range join**
  - No significant compared to performance.

Experimentation

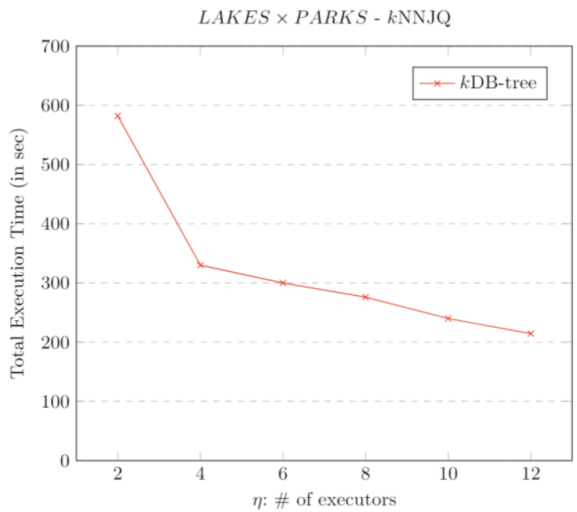
Indices over best spatial partitioning technique kDB-tree (Peak Execution Memory)



- **Memory requirement increases linearly** with the **increase of k**
  - More kNN lists
- **No Index** consumes **less memory** than **R-tree**
  - **R-tree** needs the **tree nodes information in-memory** (13 %).

Experimentation

Number of executors (k = 25)



- **Better performance if more executors are used**
- **Higher values** (η > 4) have **less performance gain**
  - Data **skew issues**
    - Most Expensive task = **155 s**
    - Median = **11.4 s**
- **Solution**:
  - **Improve spatial partitioning methods**
    - Bulk-loading methods
  - Use spatial **repartitioning** techniques

4 Conclusions and Future Work

UNIVERSIDAD DE ALMERÍA

Conclusions

- A set of **experiments** over the proposed **kNNJQ algorithm in Sedona** have **demonstrated**
  - the **efficiency** (in terms of total execution time)
  - the **scalability** (in terms of k values, sizes of datasets and number of executors (η)).

- **kDB-tree** partitioning technique **shows the best performance** thanks to
  - its **regular data-based subdivision**
  - its **more balanced partitions**.

- The use of **R-tree** as an **in-memory local index** significantly **increases the performance**
  - **compared to other non-indexed methods such as a plane-sweep algorithm**.

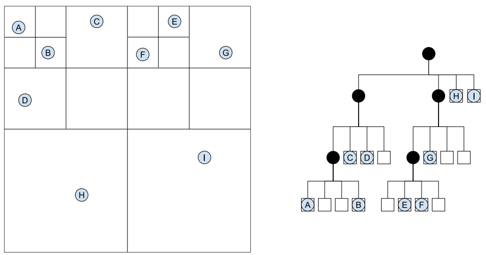
Conclusions

- **Memory requirements** (in terms of Peak Execution Memory and Shuffled Data) **increase linearly with the value of k**
  - allowing the use of higher k values without consuming many cluster resources.

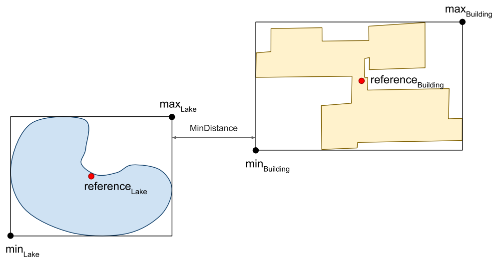
- The **performance** of the kNNJQ algorithm **improves as the number of executors (η)) increases**,
  - although **<u>there are skew problems</u>** that prevent further improvements

Future Work

- Implement kNNJQ using **Quadtree as local index**



- Extend the algorithm to **other spatial data types**, like point-rectangle, rectangle-polygon, etc.



- **Comparison** with other **DSAS like LocationSpark.**

Thank you for your attention

Questions?

UNIVERSIDAD DE ALMERÍA

Thank you for your attention

Enhancing Sedona (formerly GeoSpark) with Efficient kNearest Neighbor Join Processing

Francisco García-García[1], Antonio Corral[1], Luis Iribarne[1], and Michael Vassilakopoulos[2]

[1]Applied Computing Group, Dept. of Informatics, University of Almeria, Spain
[2]DaSE Lab, Dept. of Electrical and Computer Engineering, University of Thessaly, Volos, Greece

paco.garcia@ual.es

http://acg.ual.es

GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD TIN2017-83964-R

UNIVERSIDAD DE ALMERÍA