Ge Song, Frédéric Magoulès (Supervisor), Fabrice Huet (Co-Supervisor)
Lab MICS
CentraleSupélec
Université Paris-Saclay

Parallel and Continuous Join Processing for Data Stream

Thèse pour l'obtention du grade de Docteur

Table of Contents

Introduction

Big Data Everywhere

- Google: 24 PB / day
- Facebook: 10 millions photos + 3 billion “likes” / day
- Youtube: 800 million visitors / month
- Twitter: Doubling its size every year

Issues

- The most significant issue comes from the size of Big Data.
- The flip side of size is speed.
- The cost of network communication in transferring data.
- The dynamics of data.

Dynamic Data Stream

Persistent Static Relations $\Rightarrow$ Transient Dynamic Data Streams

Batch-oriented data processing $\Rightarrow$ Real-time stream processing

$\Downarrow$

Architecture Level: possible to add or remove computational nodes based on the current load

Application Level: able to withdraw old results and take new coming data into account

Objective: parallel and continuous processing for Join operation

Join: a popular and often used operation in the big data area.

- Data parallelism $\Rightarrow$ Data Driven Join $\Rightarrow$ kNN
- Task parallelism $\Rightarrow$ Query Driven Join $\Rightarrow$ Semantic Join on RDF data

Part I: Data Driven Stream Join (kNN)

Outline

- Introduction
- Parallel Workflow
- Theoretical Analysis
- Continuous kNN
- Experiment Result
- Conclusion

Outline

Introduction

Definition: kNN

Given a set of query points $\mathbb{R}$ and a set of reference points $\mathbb{S}$, a $\mathbb{k}$ **nearest neighbor join** is an operation which, for each point in $\mathbb{R}$, discovers the $\mathbb{k}$ nearest neighbors in $\mathbb{S}$.

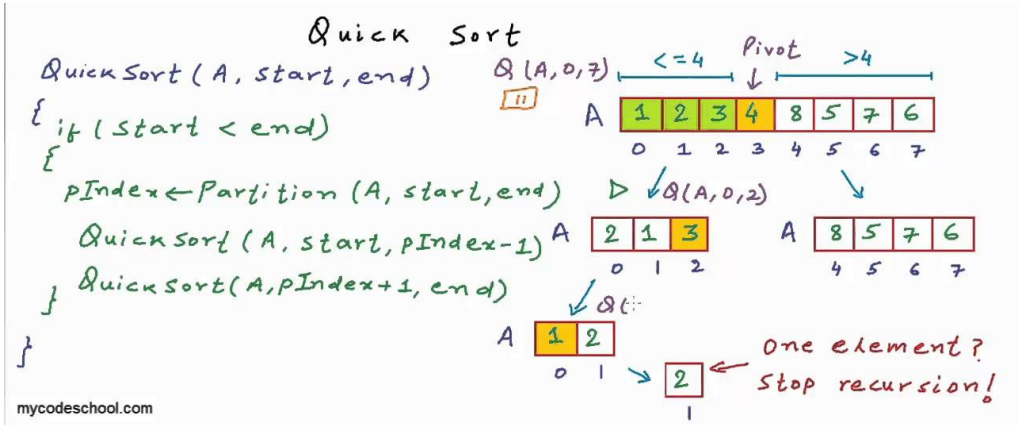
- Query never changes
- Data changes: GPS (2 Dimensions), Twitter (77 Dimensions), Images (128 Dimensions) etc.

Introduction: Basic Idea

- Nested Loop – Calculate the Distances (Complexity $O(n^2)$)

```
for(int r : R){
    for(int s : S){
        Distance(r, s);
    }
}
```

- Sort – Find the top k smallest distance (Complexity $n \cdot \log(n)$)



Outline

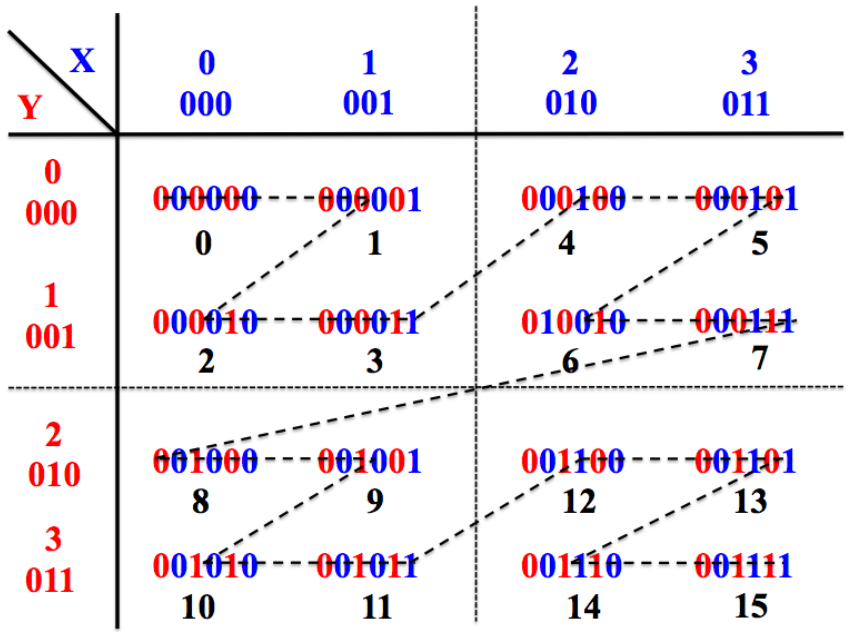
- Introduction
- Parallel Workflow
- Theoretical Analysis
- Continuous kNN
- Experiment Result
- Conclusion

Parallel Workflow

- Data Preprocessing
  - To reduce the dimension of data
  - To select central points of data clusters
- Data Partitioning
  - Distance Based Partitioning Strategy
  - Size Based Partitioning Strategy
- Computation
  - One Round Job
  - Two Rounds jobs

Data Preprocessing — To reduce the dimension of data

Space Filling Curve (Z-Value)

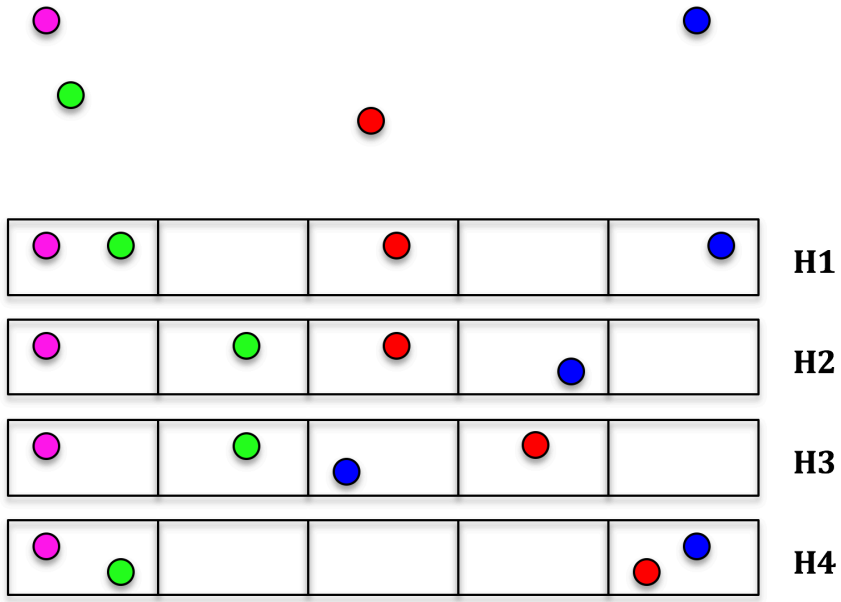


Locality Sensitive Hashing (LSH)

Data Preprocessing — To reduce the dimension of data

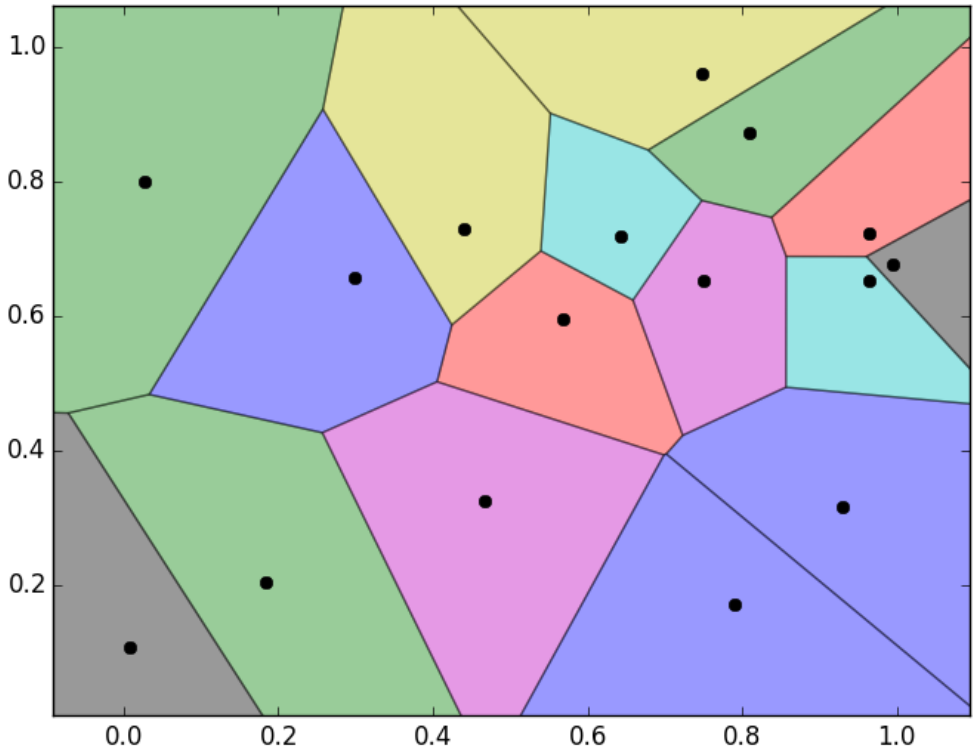
Space Filling Curve (Z-Value)

Locality Sensitive Hashing (LSH)



Data Preprocessing — To select central points (Pivots) of data clusters

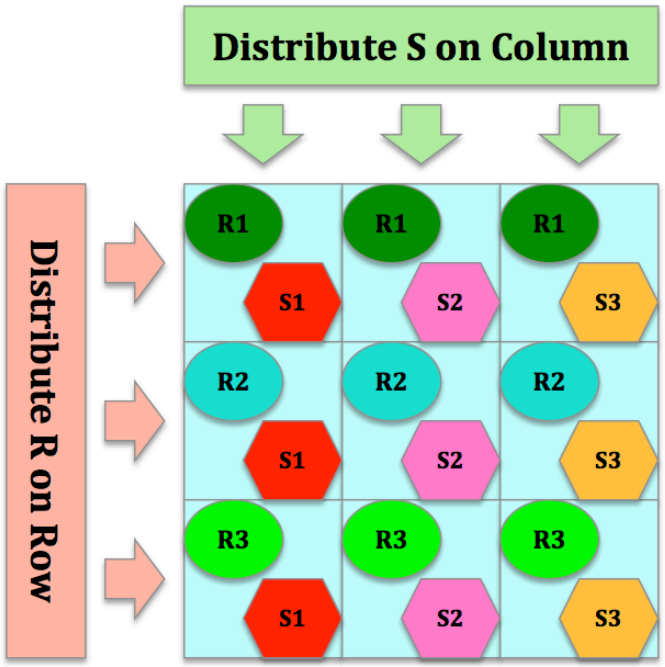
Voronoi Diagram:



Data Preprocessing — To select central points (Pivots) of data clusters

- **Random Selection:** generates a set of samples, calculates the pairwise distance of the points in the sample, and the sample with the biggest sum of distances is chosen as the set of pivots
- **Furthest Selection:** randomly chooses the rest pivot, and calculates the furthest point to this chosen pivot as the second pivot, and so on until having the desired number of pivots.
- **K-Means Selection:** applies the traditional k-means method on a data sample to update the centroid of a cluster as the new pivot each step, until the set of pivots stabilizes.

Data Partitioning – Basic Idea



Problem: n^2 tasks for calculating pairwise distances; wastes a lot of hardware resources, and ultimately leads to low efficiency.

Data Partitioning – Motivation

The key to improve the performance is to preserve spatial locality of objects when decomposing data for tasks.

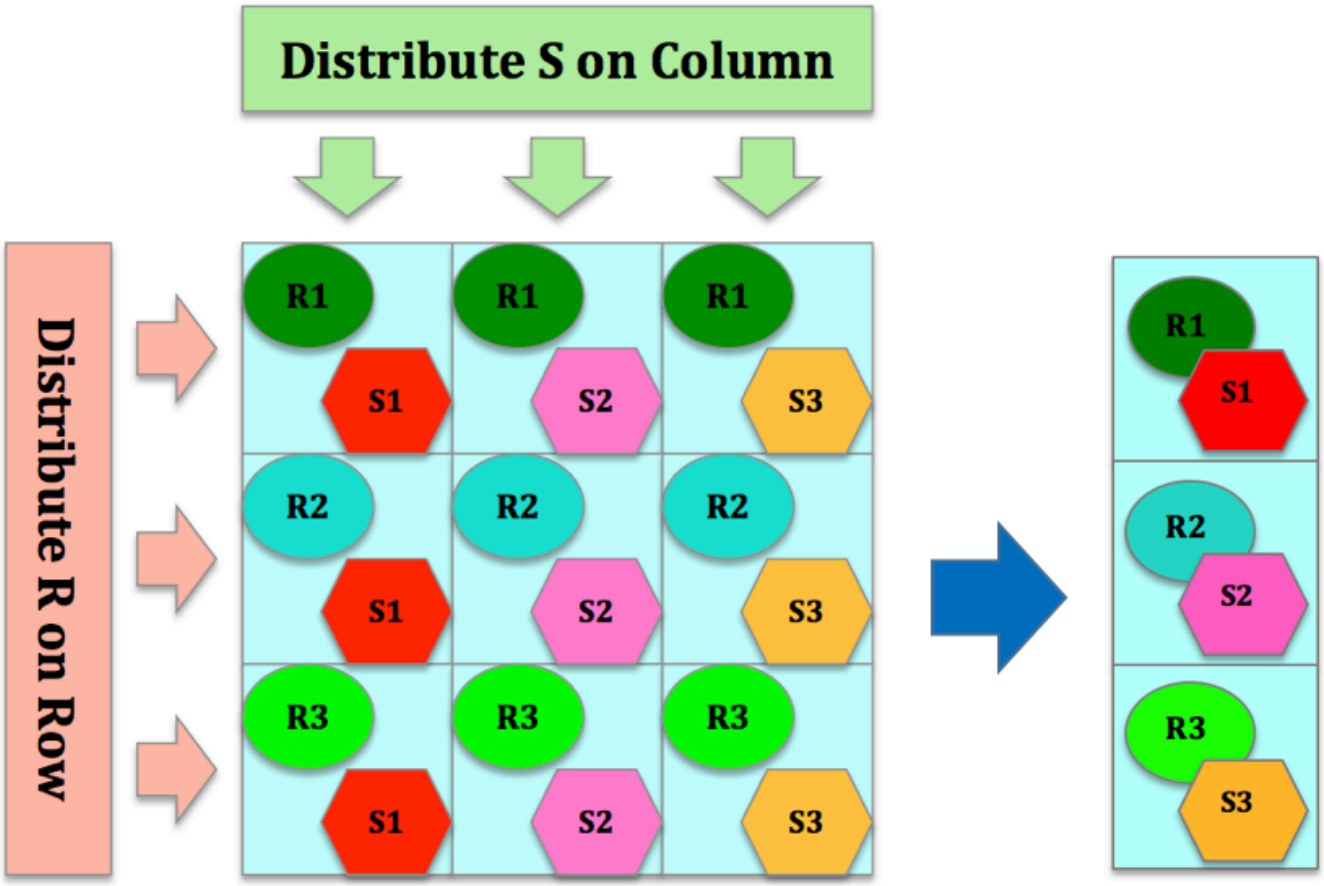
More precisely, what we want is: for every partition R_i ($\cup_i R_i = R$), find a corresponding partition S_j ($\cup_j S_j = S$), where:

$$kNN(R_i \ltimes S) = kNN(R_i \ltimes S_j)$$

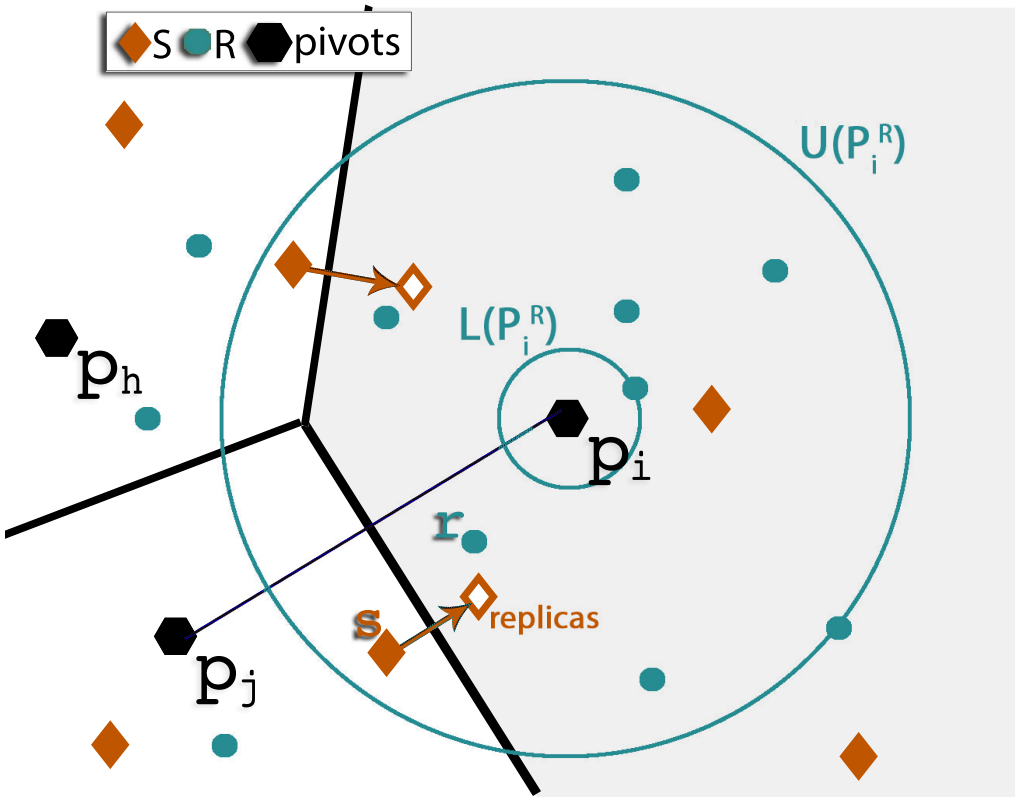
And,

$$kNN(R \ltimes S) = \bigcup kNN(R_i \ltimes S_j)$$

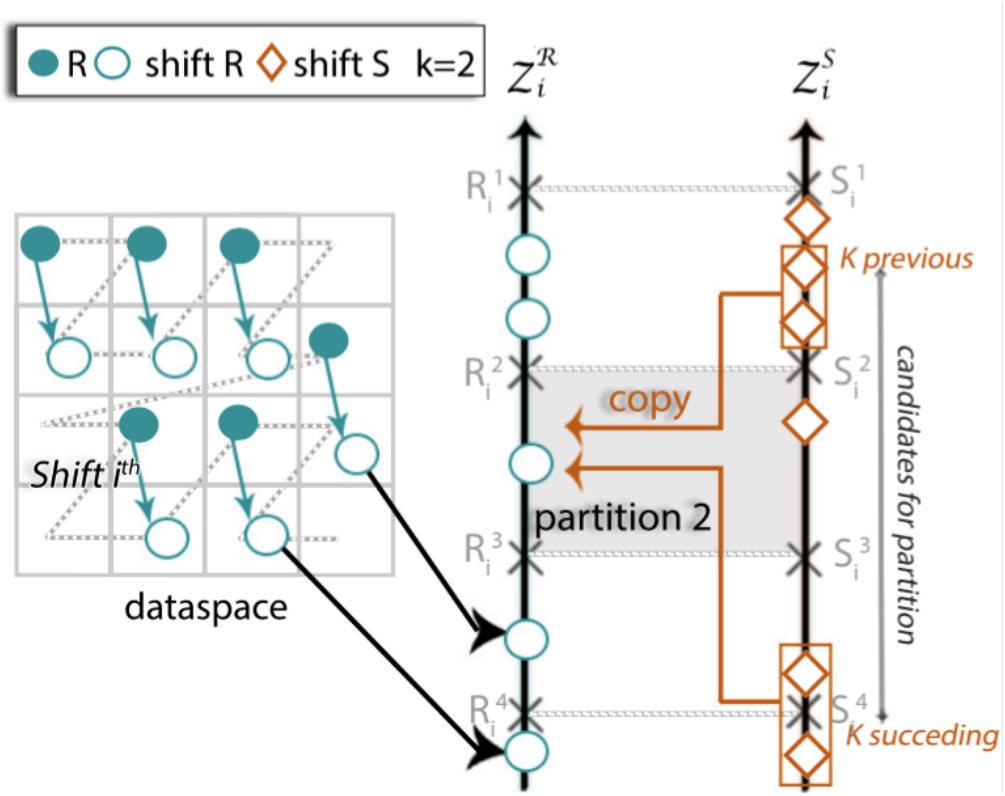
Data Partitioning – Motivation



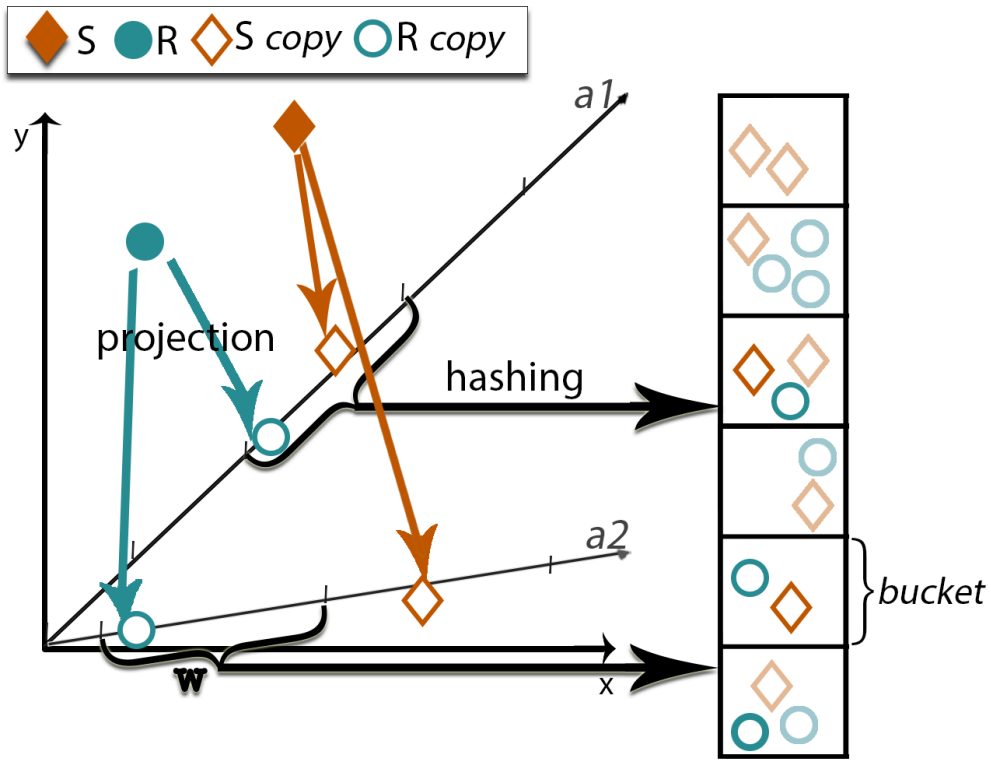
Data Partitioning — Distance Based Partitioning Strategy



Data Partitioning — Size Based Partitioning Strategy — Z-Value



Data Partitioning – Size Based Partitioning Strategy – LSH



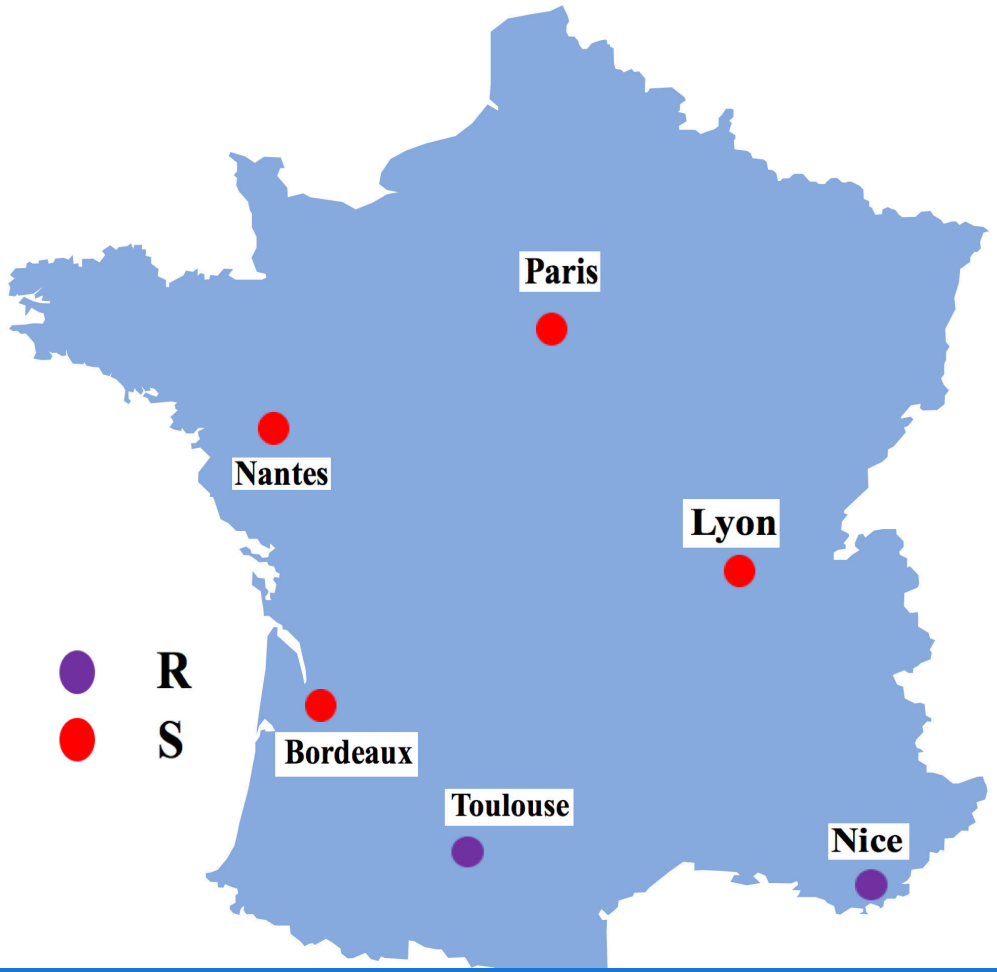
Computation

- One job — Direct give the global results
- Two consecutive jobs — First give the local results, then merge the local results into the global results

Purpose: using multiple rounds of jobs in order to reduce the number of elements to be sorted.

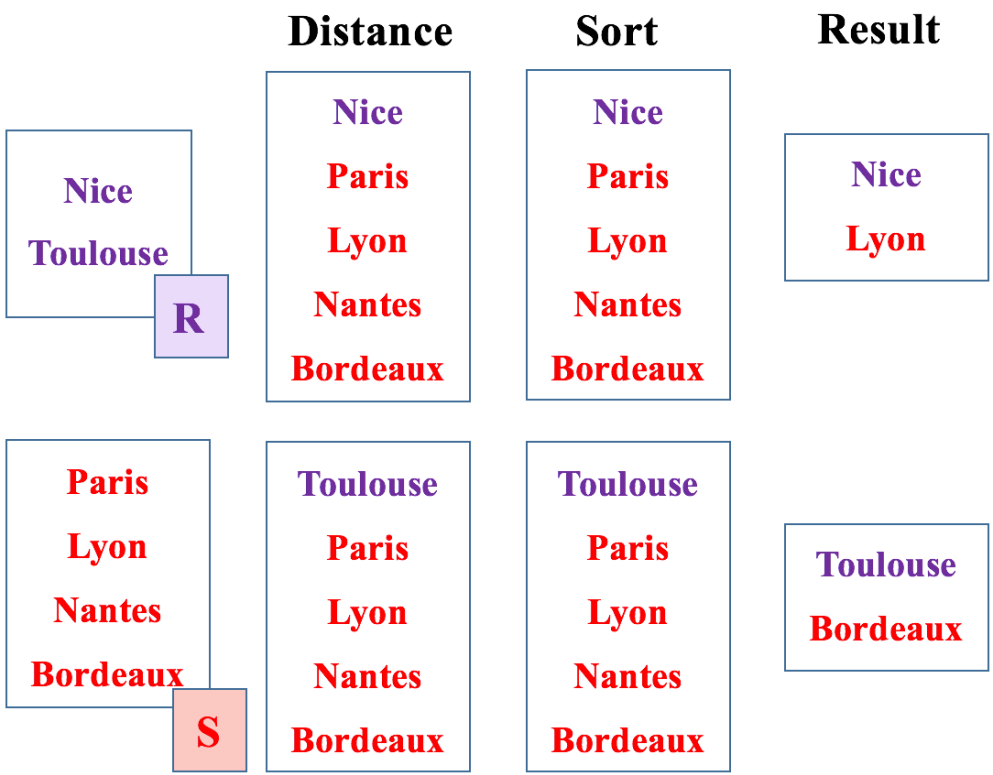
Computation – Example

For each city in R, find the nearest city in S.



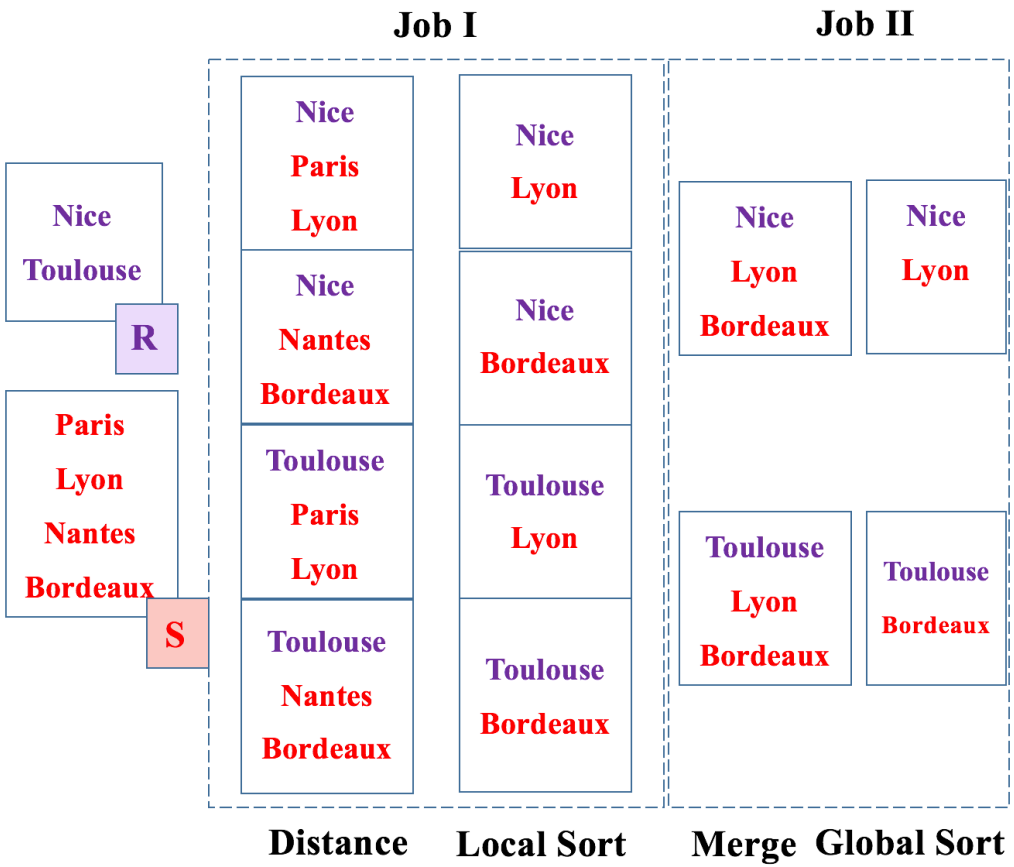
Computation – Example

One Job:



Computation – Example

Two Jobs:



Outline

- Introduction
- Parallel Workflow
- Theoretical Analysis
- Continuous kNN
- Experiment Result
- Conclusion

Theoretical Analysis

- Load Balance
- Accuracy
- Complexity

Load Balance

What is Load Balance?

$$|R_i| \times |S_i| = |R_j| \times |S_j|$$

Sub-Optimal Option

$$\forall\, i \neq j,$$
$$\text{if } |R_i| = |R_j| \text{ or } |S_i| = |S_j|,$$
$$\text{then } |R_i| \times |S_i| \approx |R_j| \times |S_j|$$

Load Balance

if $|R_i| = |R_j|$, the Worst Case Complexity is:

$$\mathcal{O}\left(|R_i| \times \log|S_i|\right) = \mathcal{O}\left(\frac{|R|}{n} \times \log|S|\right)$$

if $|S_i| = |S_j|$, the Worst Case Complexity is:

$$\mathcal{O}\left(|R_i| \times \log|S_i|\right) = \mathcal{O}\left(|R| \times \log\frac{|S|}{n}\right)$$

$n \ll |S| \Rightarrow |R_i| = |R_j|$ is better.

All advanced partitioning strategies first partition R into equal sized partitions, then find the corresponding S for each R.

Accuracy

The lack of accuracy is the direct consequence of techniques to reduce the dimensionality with techniques of as z-values and LSH.

- **Z-Value**
  - Depends on k
  - Increase the number of shifts of data will decrease the error rate
- **LSH**
  - Depends on parameter tuning
  - Increase the number of hash functions will decrease the error rate

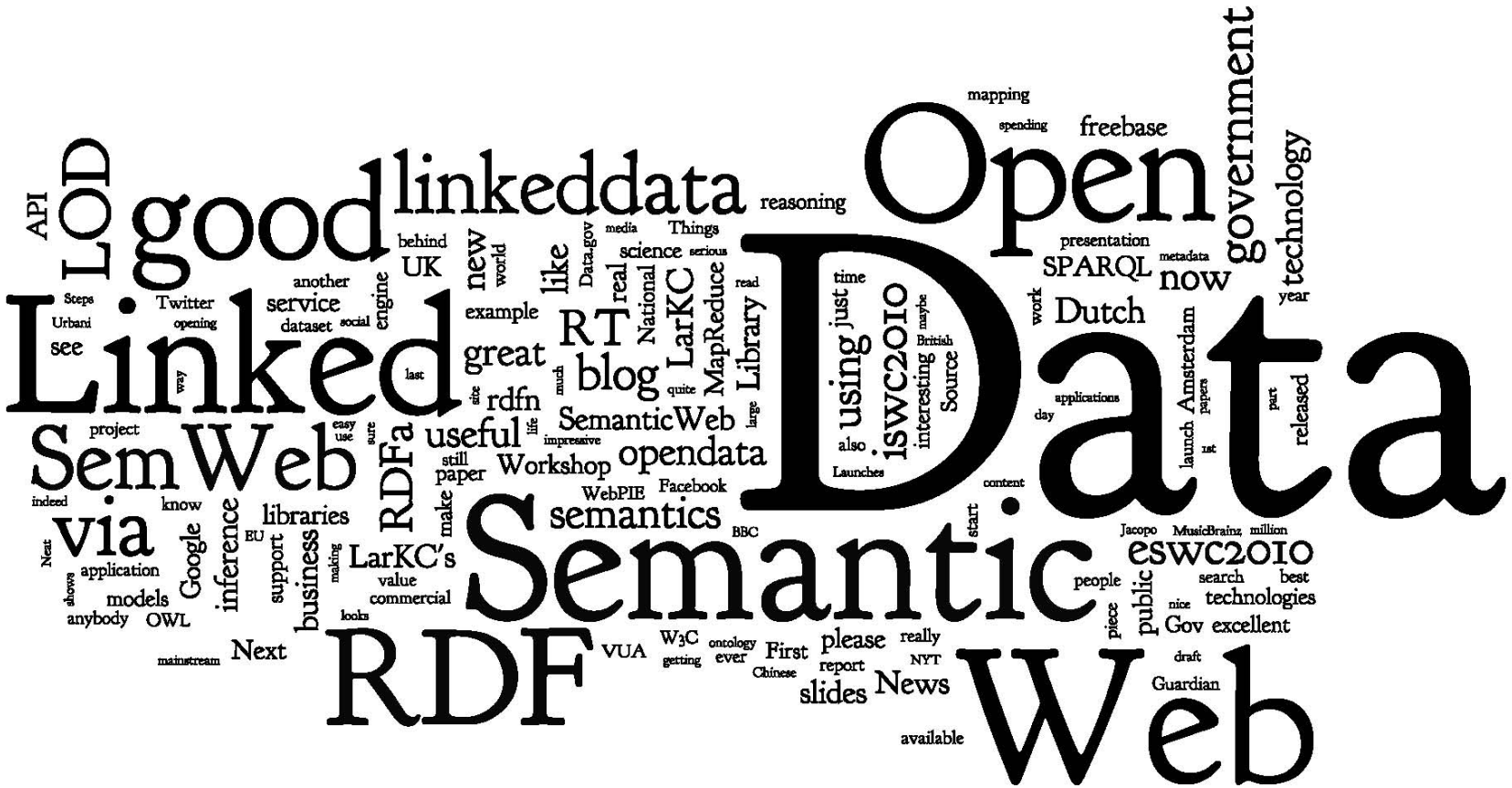
Complexity

- **The number of MapReduce jobs:** starting a job requires some initial steps.
- **The number of Map tasks and Reduce tasks used to calculate $\mathrm{kNN}(\mathrm{R_i} \ltimes \mathrm{S})$:** the larger this number is, the more information is exchanged through the network.
- **The number of final candidates for each object $\mathrm{r_i}$:** We have seen that advanced algorithms use pre-processing and partitioning techniques to reduce this number as much as possible.

Main Overhead

- Communication overhead:
  - the amount of data transmitted over the network
- Computation overhead:
  - computing the distances
  - finding the k smallest distances

Part II: Query Driven Stream Join (RDF)



Outline

- Related Work
- Query Decomposition and Distribution
- Data Partition and Assignment
- Parallel and Distributed Query Plan
- Continuous Join
- Analysis
- Implementation
- Experiment Result
- Conclusion

Outline

- Related Work
- Query Decomposition and Distribution
- Data Partition and Assignment
- Parallel and Distributed Query Planner
- Continuous Join
- Analysis
- Implementation
- Experiment Result
- Conclusion

Thank You!