



Real-Time Clustering for Large Sparse Online Visitor Data

Gromit Yeuk-Yin Chan¹, Fan Du², Ryan Rossi², Anup Rao², Eunyeek Koh², Claudio T. Silva¹, Juliana Freire¹

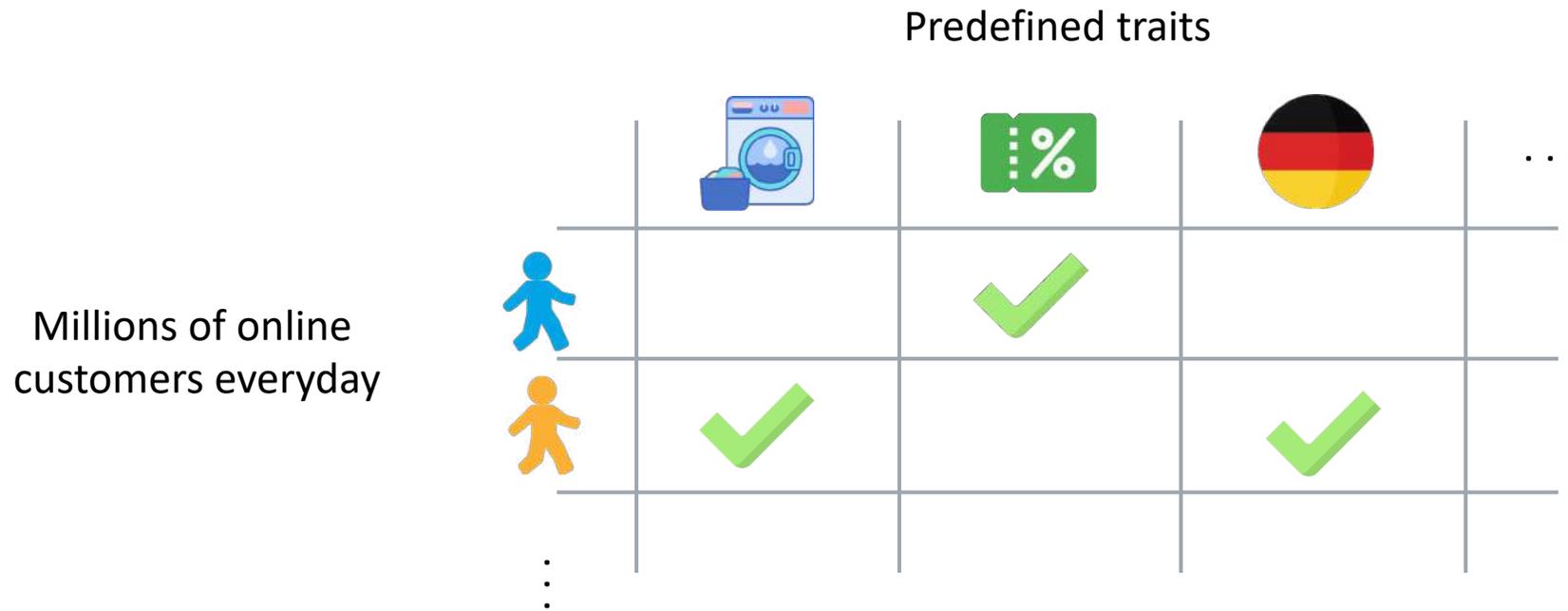
¹ New York University



² Adobe



Online Visitor Behavior are Large and Sparse



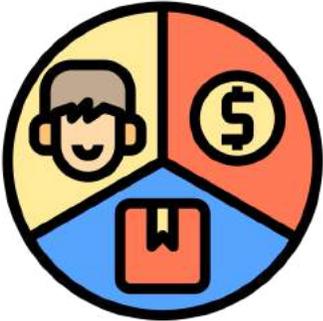
How to identify similar online behavior? → Cluster the similar customers!

Interactive Cluster Analysis Is Important

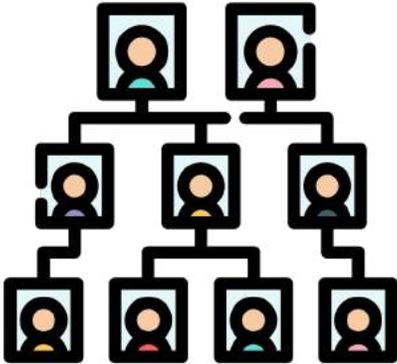
Marketing Analyst



Identify optimal clusters



Multi-level clusters

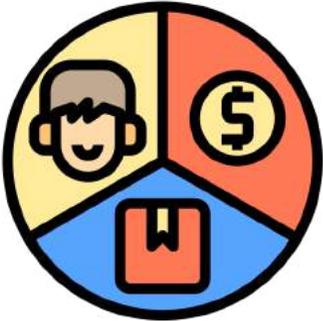


Interactive Analysis Is Important

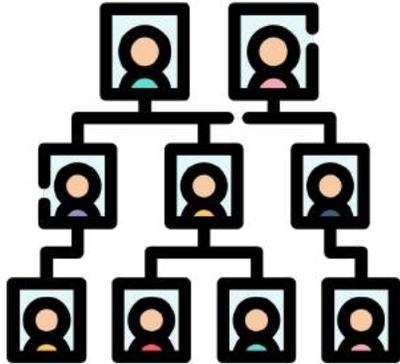
Marketing Analyst



Identify optimal clusters



Multi-level segments



Interactive Data Analysis
(playing with different parameters)

Challenges as a Data Platform Provider

- Lots of recalculations
 - To find the “best” clusters, users need to try different combinations of parameters.
- Resource Utilization
 - Each time a clustering is started from scratch, a job is submitted to the distributed system.

Challenges as a Data Platform Provider

- Lots of recalculations
 - To find the “best” clusters, users need to try different combinations of parameters.
- Resource Utilization
 - Each time a clustering is started from scratch, a job is submitted to the distributed system.

Solution 1

lightweight and fast end-to-end clustering
(e.g. K Means)

Solution 2

Pre-compute a hierarchy of cluster membership
(e.g. linkage clustering)

Challenges as a Data Platform Provider

- Lots of recalculations
 - To find the “best” clusters, users need to try different combinations of parameters.
- Resource Utilization
 - Each time a clustering is started from scratch, a job is submitted to the distributed system.

Solution 1

lightweight and fast end-to-end clustering
(e.g. K Means)

Solution 2



Pre-compute a hierarchy of cluster membership
(e.g. linkage clustering)

The quality of interactive analysis will be affected if the result does not arrive within 500ms (Liu & Heer TVCG 2014)

Motivation

- Linkage Clustering (main technique to construct a hierarchy)
 - Requires a pairwise distance matrix
 - Impossible in terms of memory and time ($O(n^2)$) for moderate size data
- Application to Distributed System
 - Parallel algorithm
 - Even data distribution among the computation nodes.

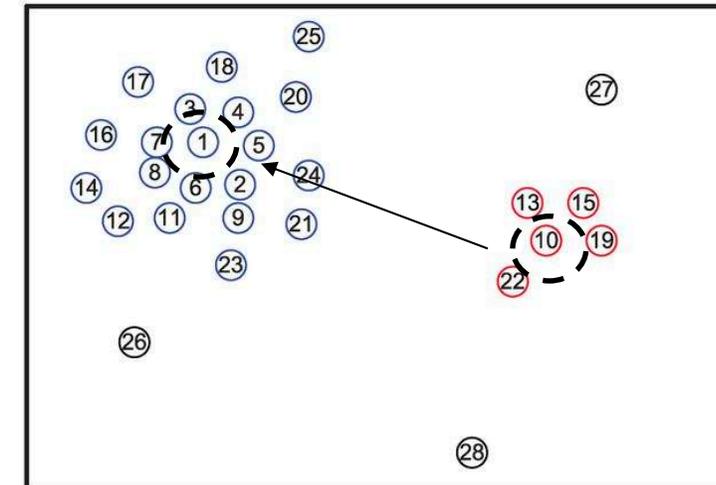
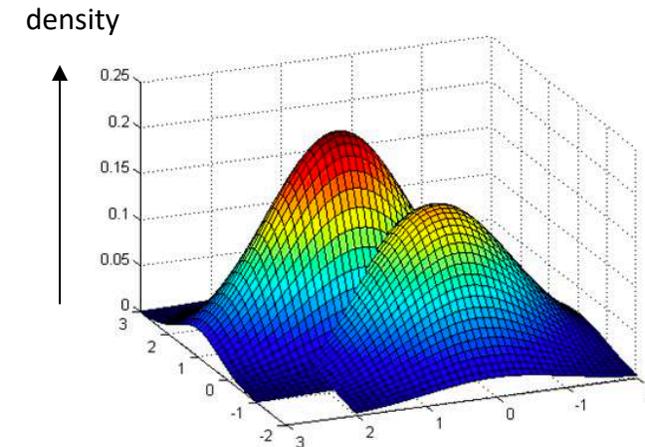
Motivation

- Linkage Clustering (main technique to construct a hierarchy)
 - Requires a pairwise distance matrix
 - Impossible in terms of memory and time ($O(n^2)$) for moderate size data
- Application to Distributed System
 - Parallel algorithm
 - Even data distribution among the computation nodes.

Can we create linkage without all pairwise similarity calculations in the distributed system?

Density Peaks Clustering

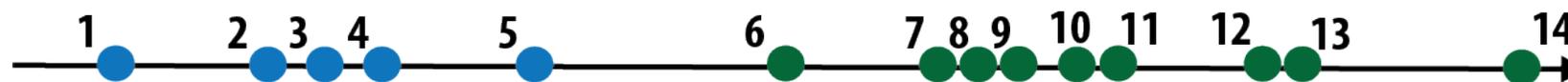
- Density Peaks (DP) Clustering
 - Density: number of neighbors around a point.
 - Assumption: if a point's **closest higher density point** is far away, the point is likely to be a cluster center.
 - Parameter: cutoff distance (d_{cutoff})



Rodriguez, Alex, and Alessandro Laio. "Clustering by fast search and find of density peaks." *Science* 344.6191 (2014): 1492-1496.

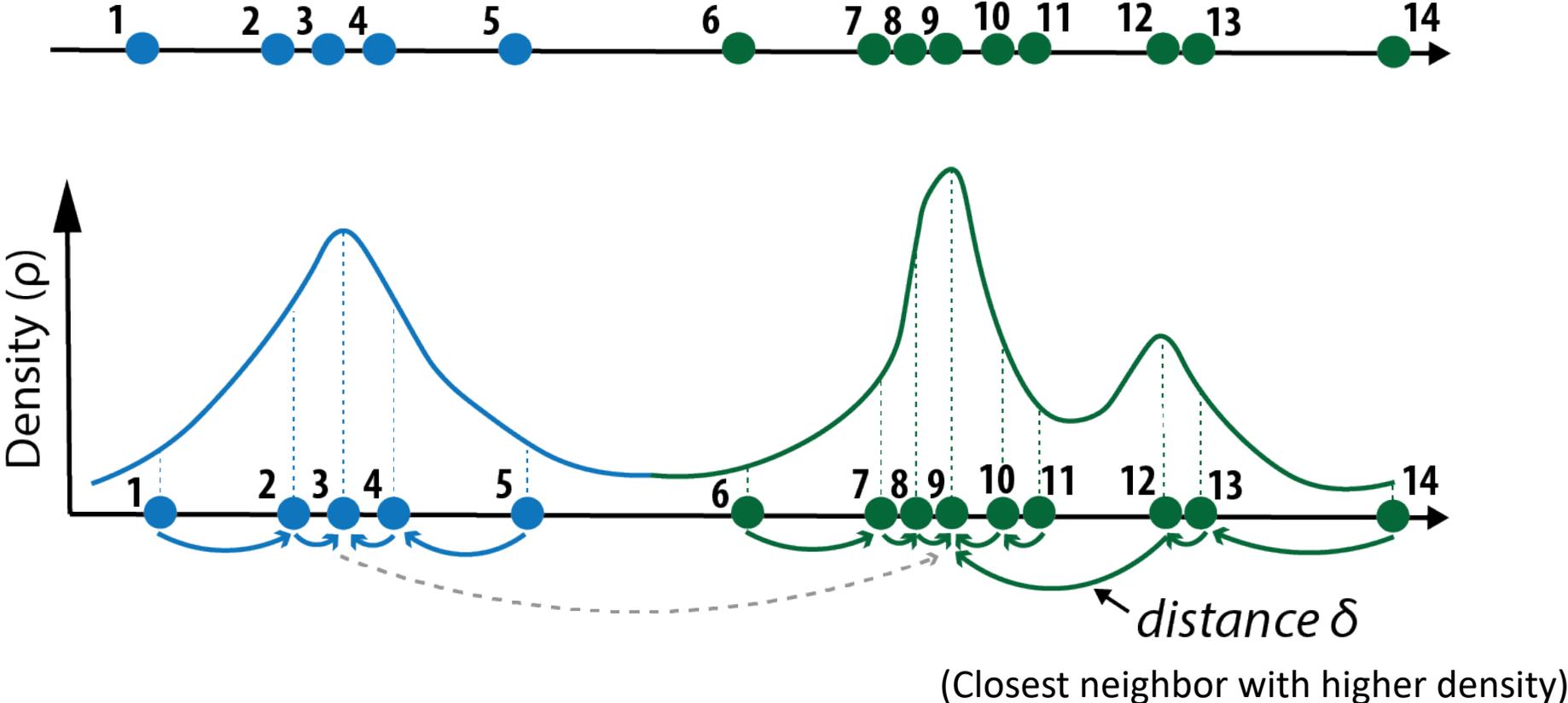
Density Peaks Clustering

Consider a 1-D example, we want to group the points with 2 clusters.



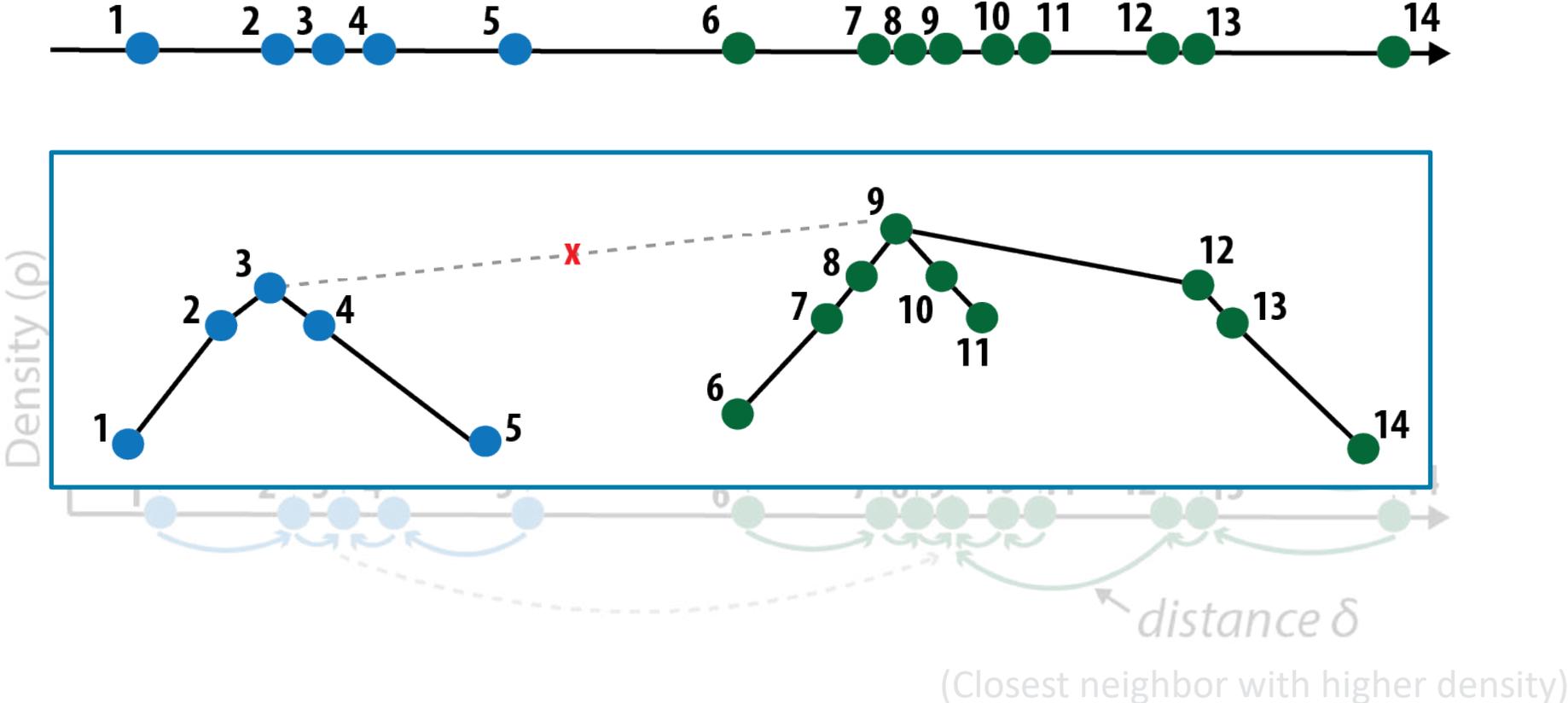
Density Peaks Clustering

Consider an 1-D example, we want to group the points with 2 clusters.



Density Peaks Clustering

Consider a 1-D example, we want to group the points with 2 clusters.



Density Peaks Clustering

- Density Peaks (DP) Clustering Algorithm
 1. Calculate density ρ
 2. Calculate shortest distance of higher density Points δ
 3. Sort the elements by density, then assign them to cluster same as the nearest neighbor of higher density.

Density Peaks Clustering

- Density Peaks (DP) Clustering Algorithm

1. Calculate density ρ $O(n^2)$

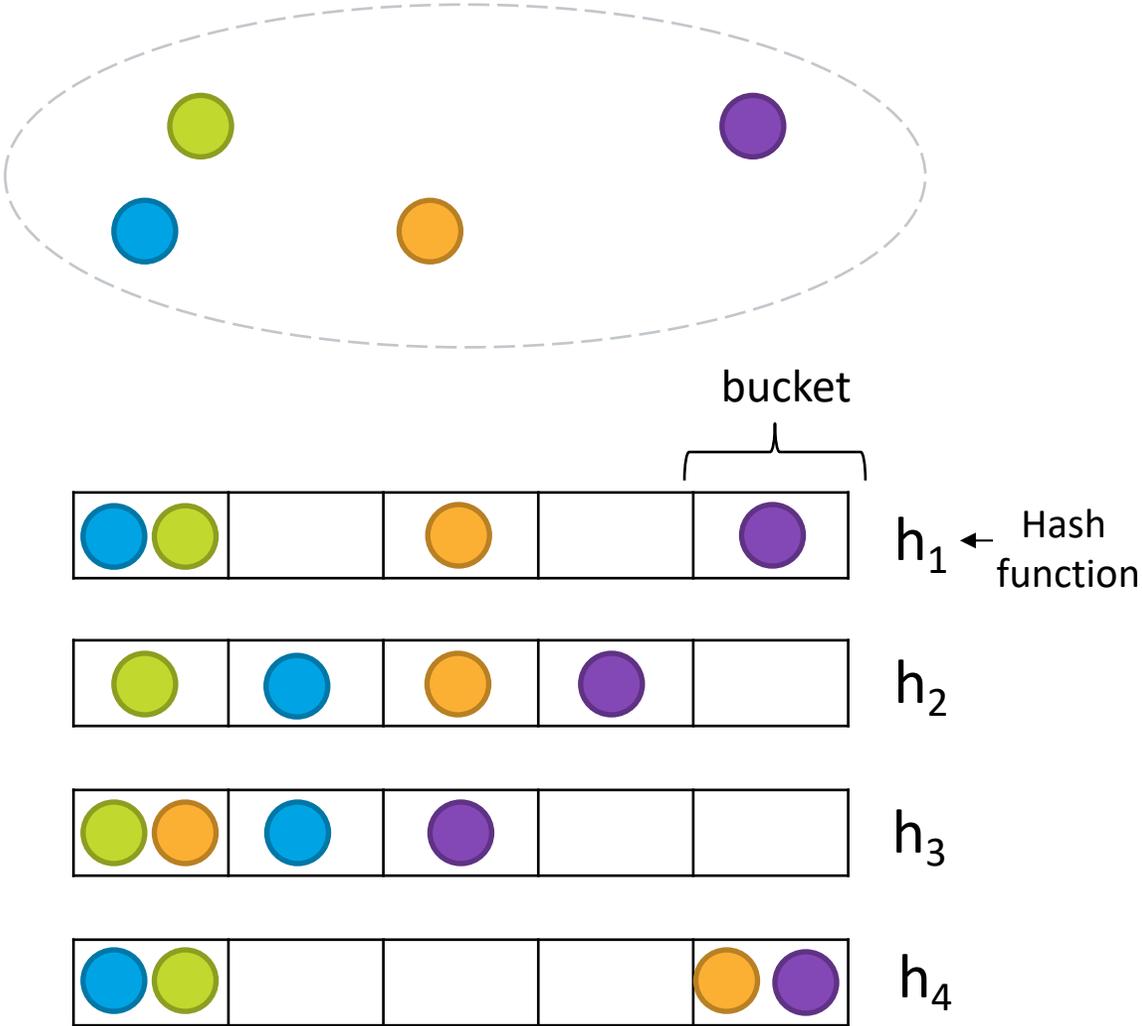
2. Calculate shortest distance of higher density Points δ $O(n^2)$

} Slow

3. Sort the elements by density, then assign them to cluster same as the nearest neighbor of higher density. $O(n)$

**Now the question:
How to make the preprocessing faster?**

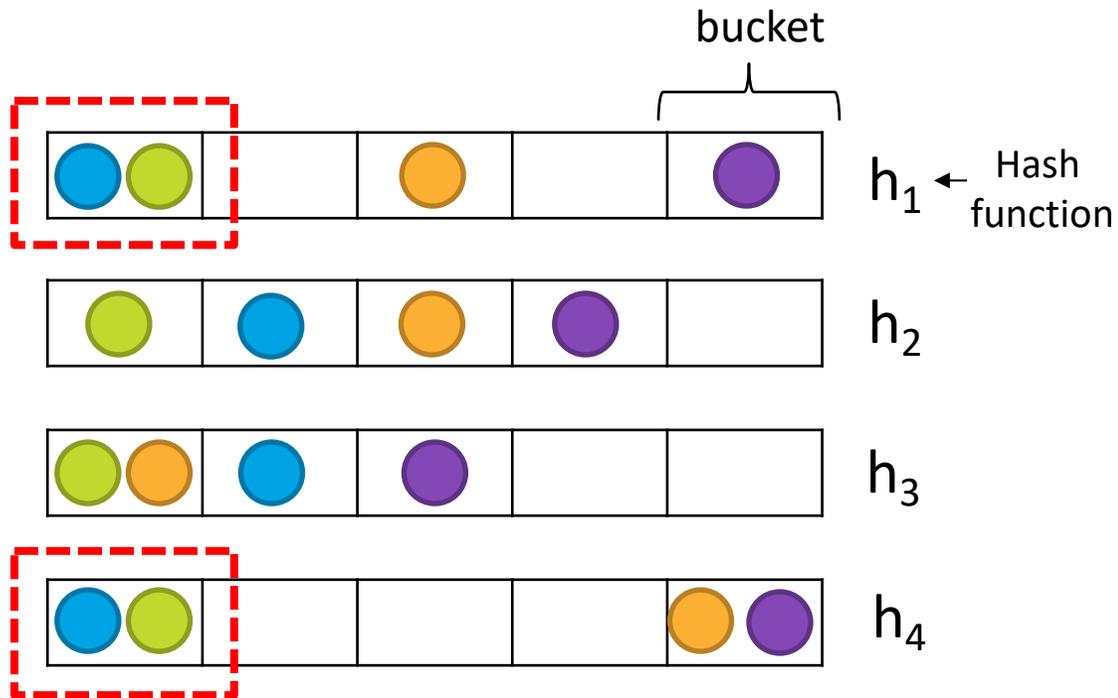
Locality Sensitive Hashing for Neighbor Querying



Locality-sensitive hashing (LSH) hashes similar items to the same hash “bucket” with high probability.

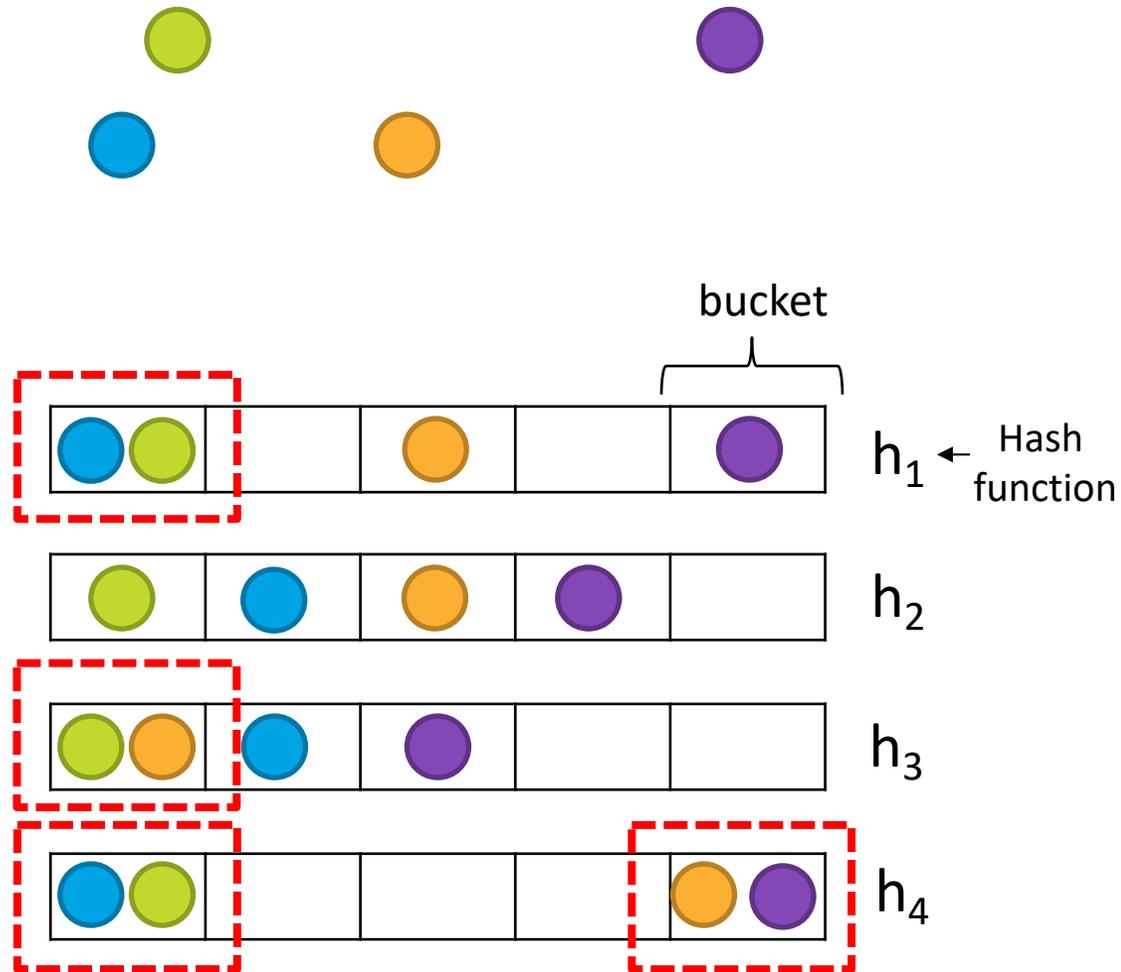
The greater number of times two items are hashed in the same bucket (*collision*), the higher the similarity they are.

Applying LSH to Density Peaks Calculation



To calculate the density ρ of each point, we query the points with high number of collisions.

Applying LSH to Density Peaks Calculation

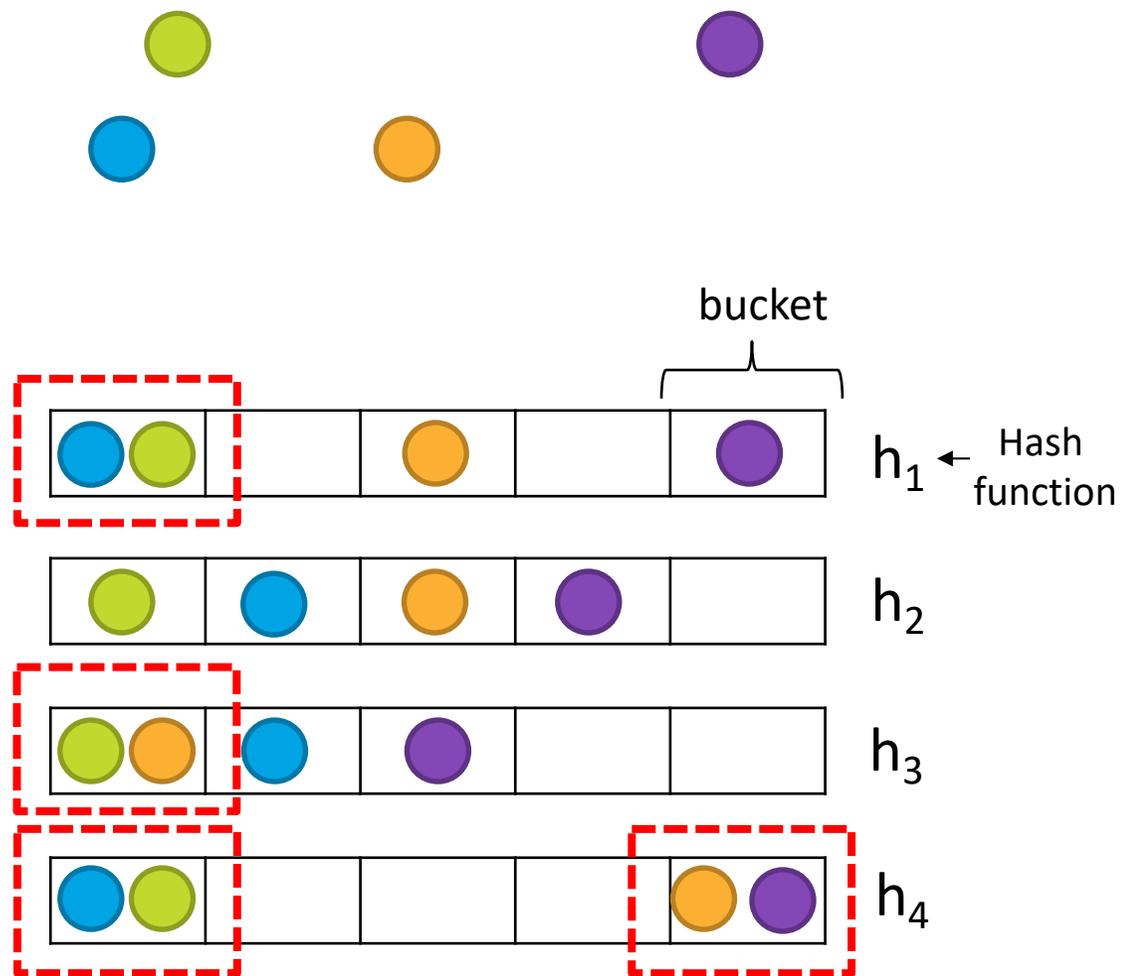


To calculate the density ρ of each point, we query the points with high number of collisions.

To retrieve distance δ of the nearest neighbor, we query the points with at least one collisions.

If a point does not have any queried result, we directly compare it with points with highest density.

Applying LSH to Density Peaks Calculation



After the query, we also have the following information:

Exact similarity between the points

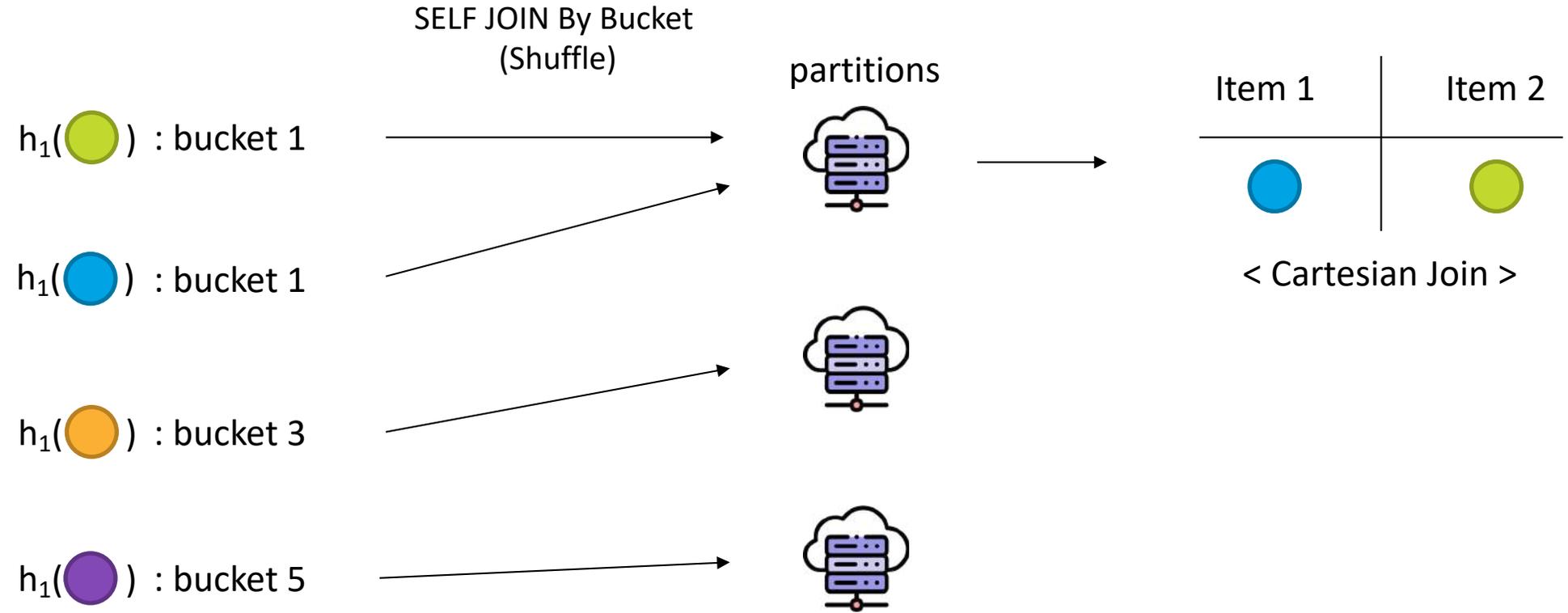


Probability of collision between the points

Thus, it is possible to calculate the *join size estimation* to refine the *density* and the *accuracy* of *nearest neighbor* query. (Zhang et. al. TKDE 2016)

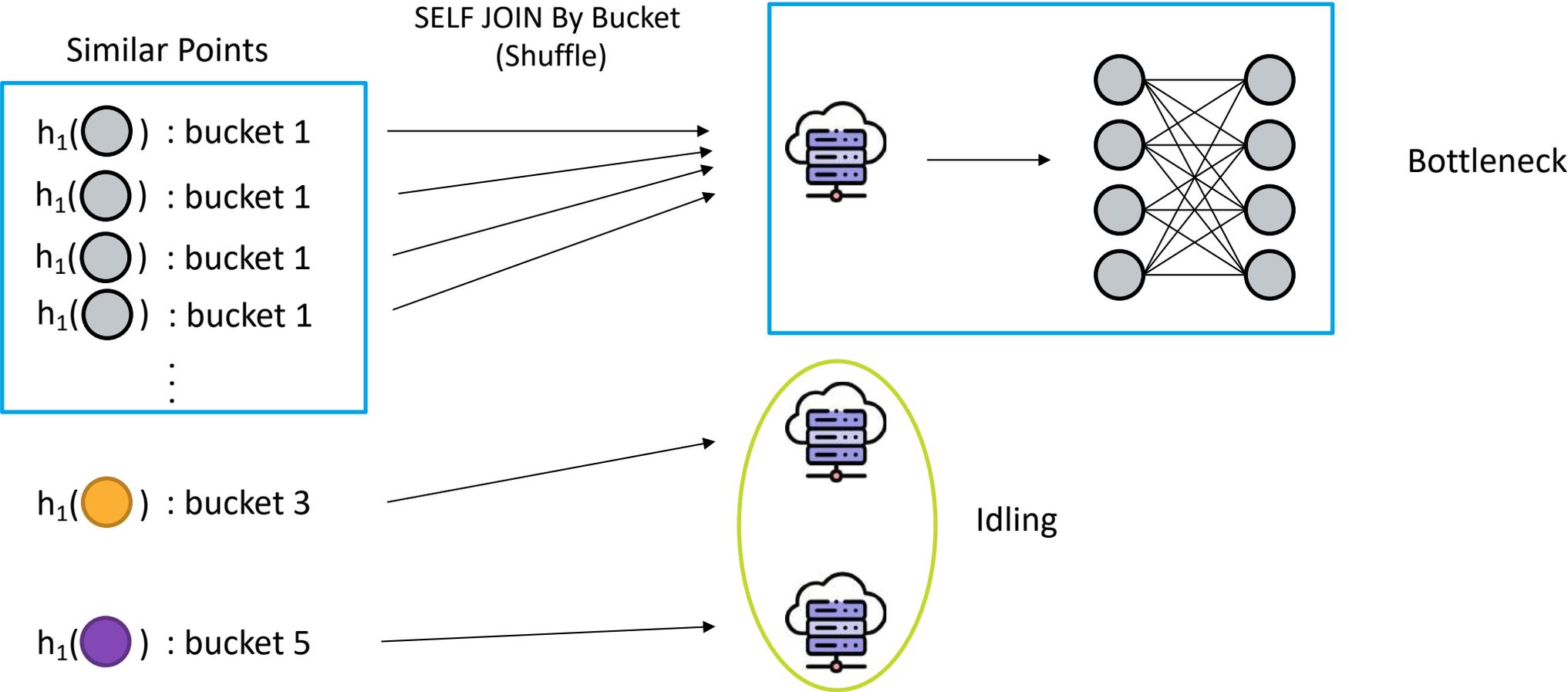
Parallelizing LSH Query in Spark

- Finding the items that collide with each other is a parallel process.



Parallelizing LSH Query in Spark

- A straightforward JOIN can lead to uneven data distributions.



Salting Strategies to Increase Parallelization

- Create Random JOIN keys to "scatter" the data

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1



Salting Strategies to Increase Parallelization

- Create Random JOIN keys to "scatter" the data

Random Keys

$h_1(\text{orange circle})$: bucket 1

$h_1(\text{purple circle})$: bucket 1

$h_1(\text{orange circle})$: bucket 1



Salting Strategies to Increase Parallelization

- Create Random JOIN keys to "scatter" the data

Random Keys

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1



Replicate 1

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1



Replicate 2

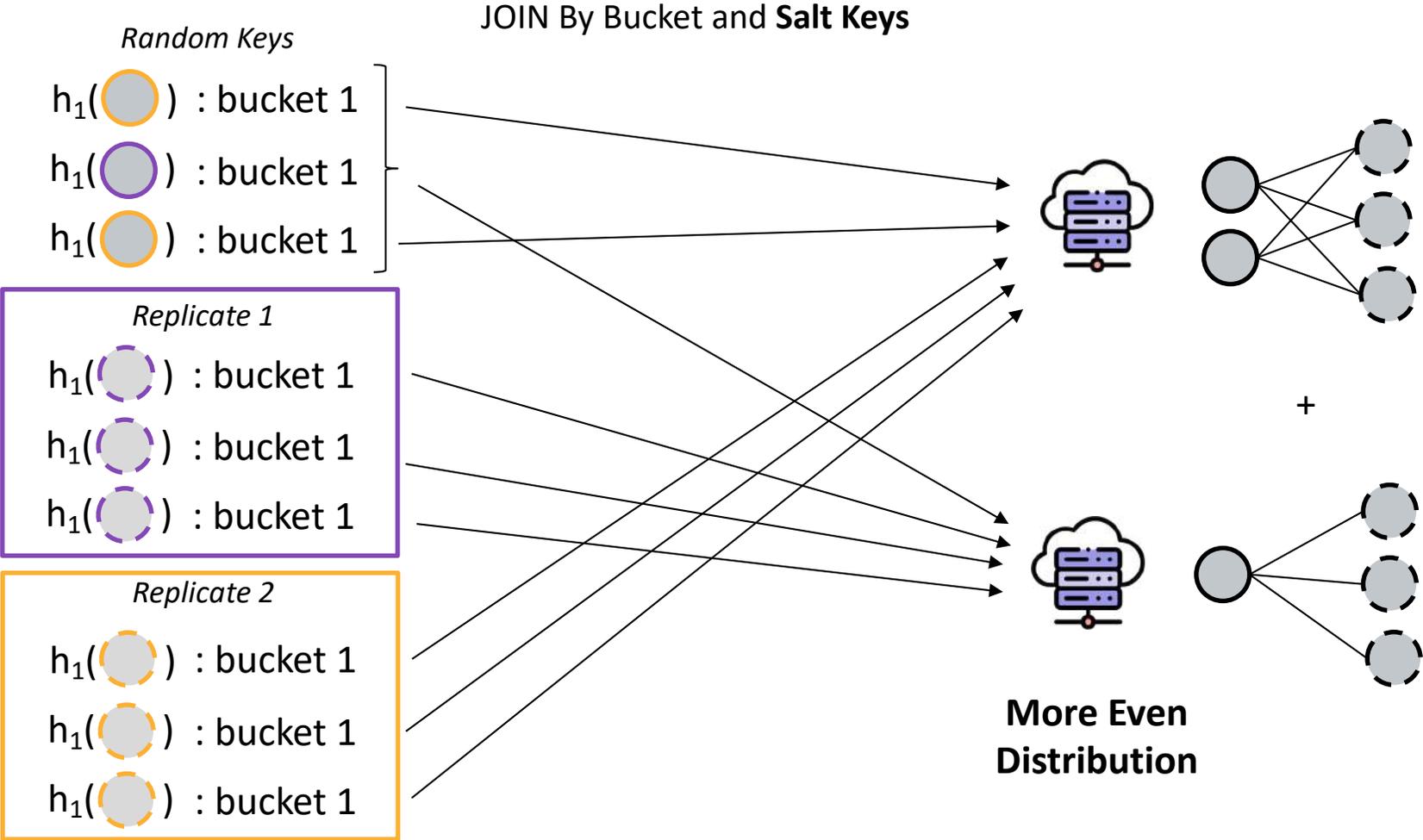
$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1

$h_1(\text{○})$: bucket 1

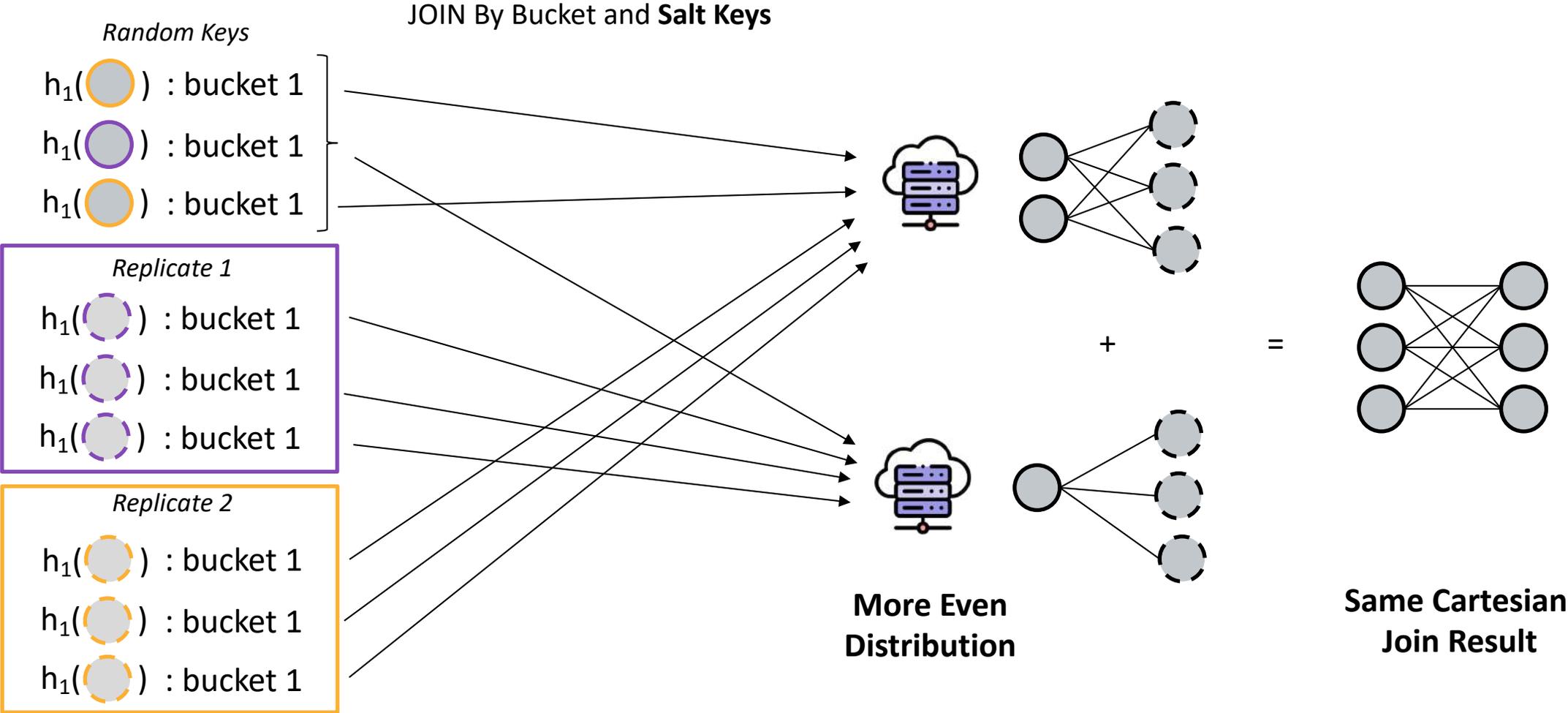
Salting Strategies to Increase Parallelization

- Create Random JOIN keys to "scatter" the data



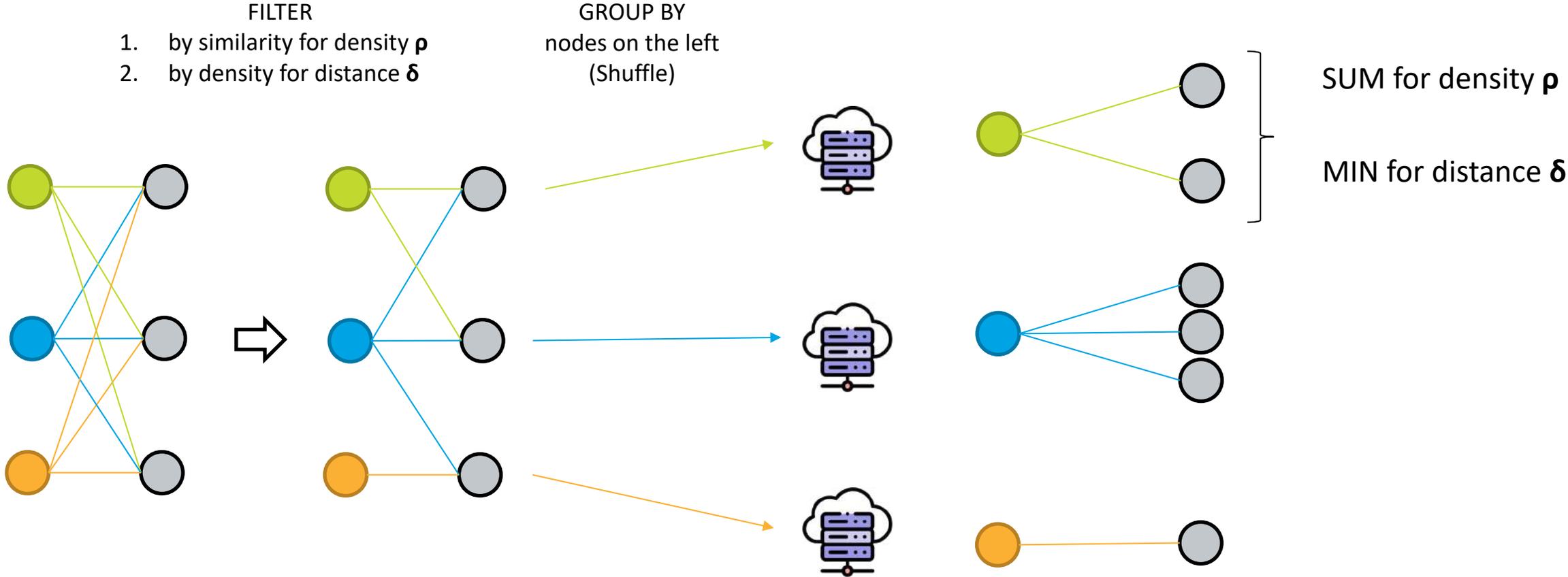
Salting Strategies to Increase Parallelization

- Create Random JOIN keys to "scatter" the data



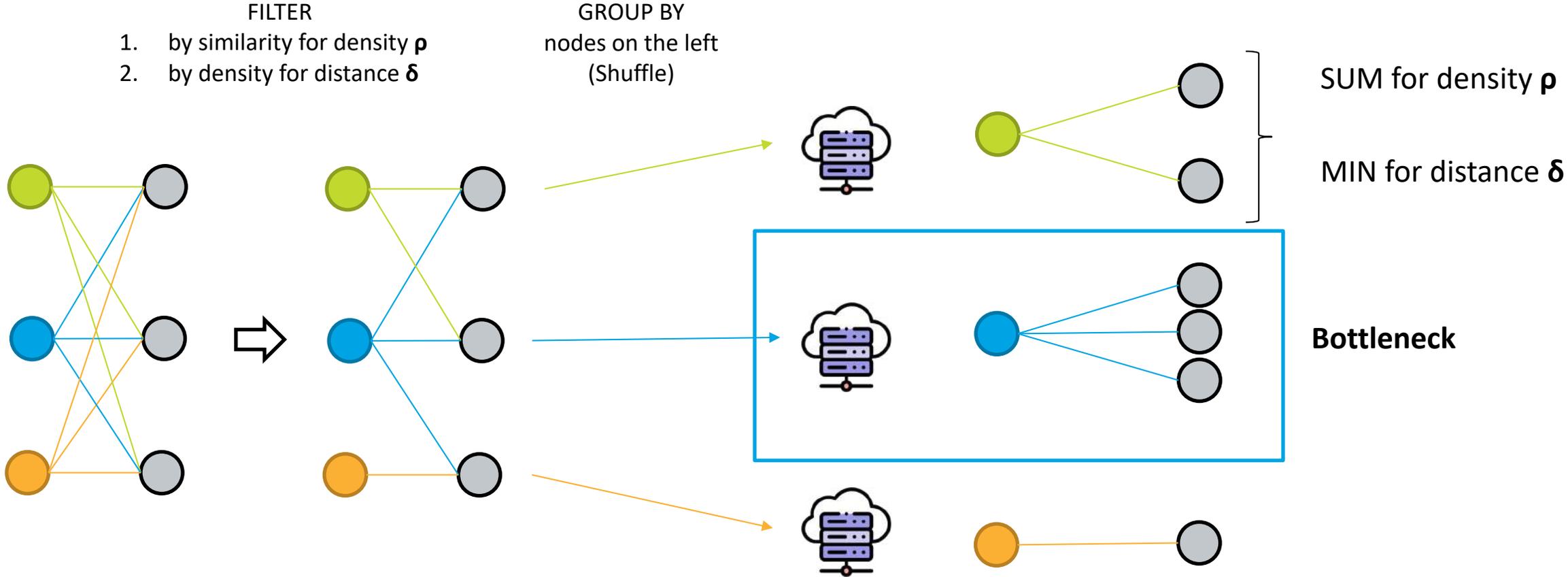
Group By Operations for Density and Distance

- The second stage to compute the Density Peaks can also be parallelized.



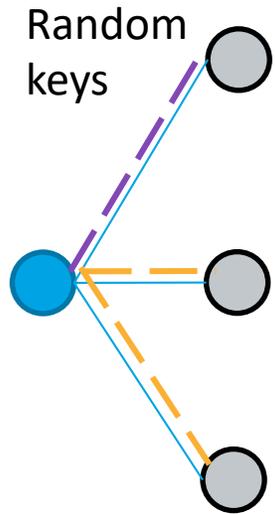
Group By Operations for Density and Distance

- The second stage to compute the Density Peaks can also be parallelized.



Salting Strategies for Group By Operations

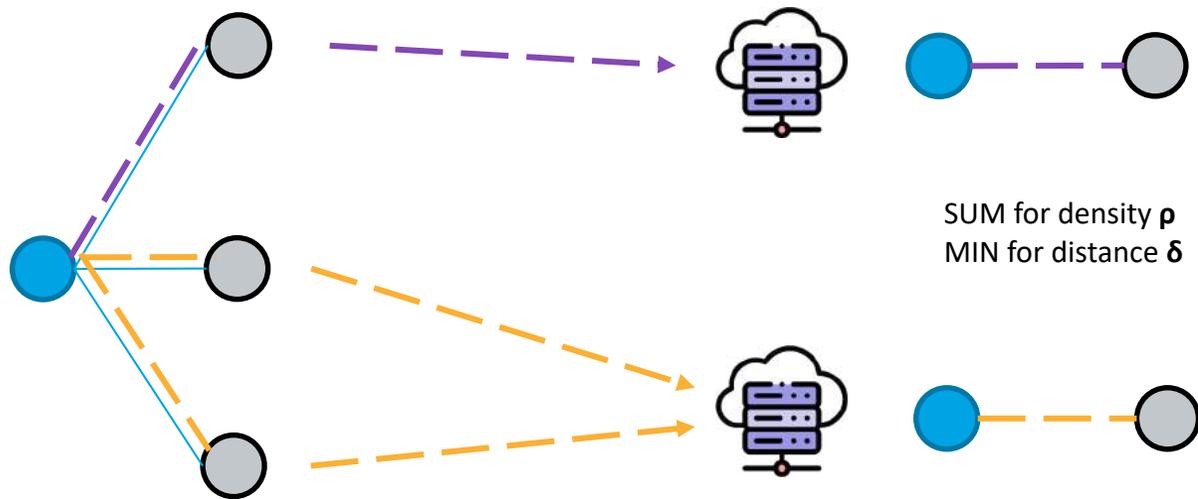
- The second stage to compute the Density Peaks can also be parallelized.



Salting Strategies for Group By Operations

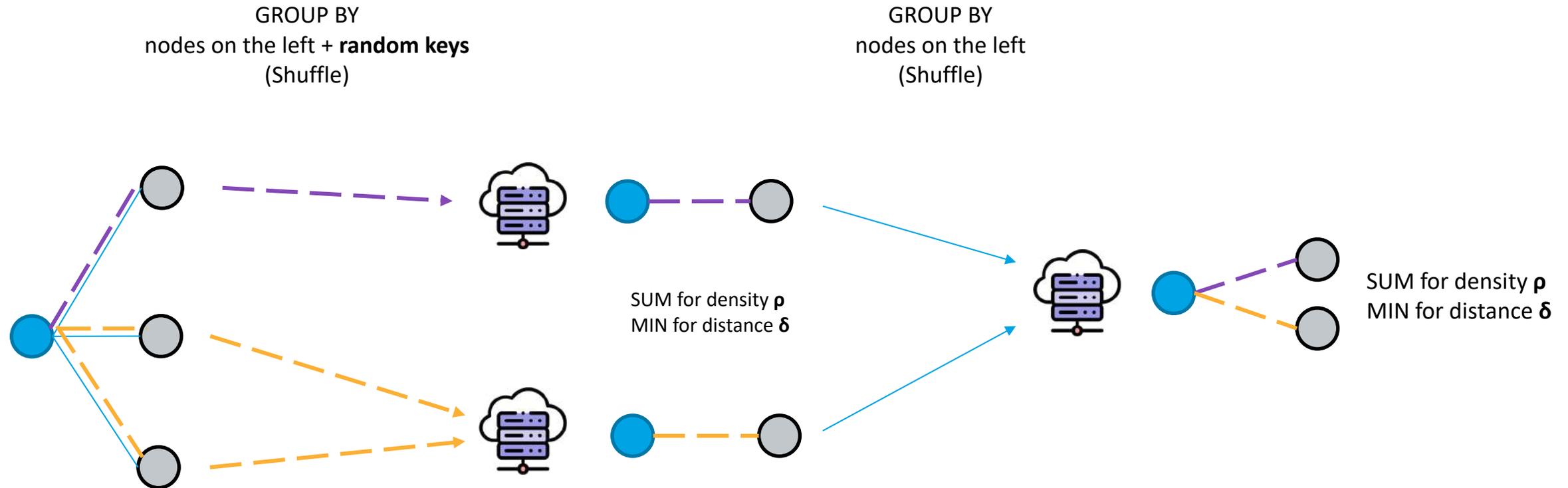
- The second stage to compute the Density Peaks can also be parallelized.

GROUP BY
nodes on the left + **random keys**
(Shuffle)



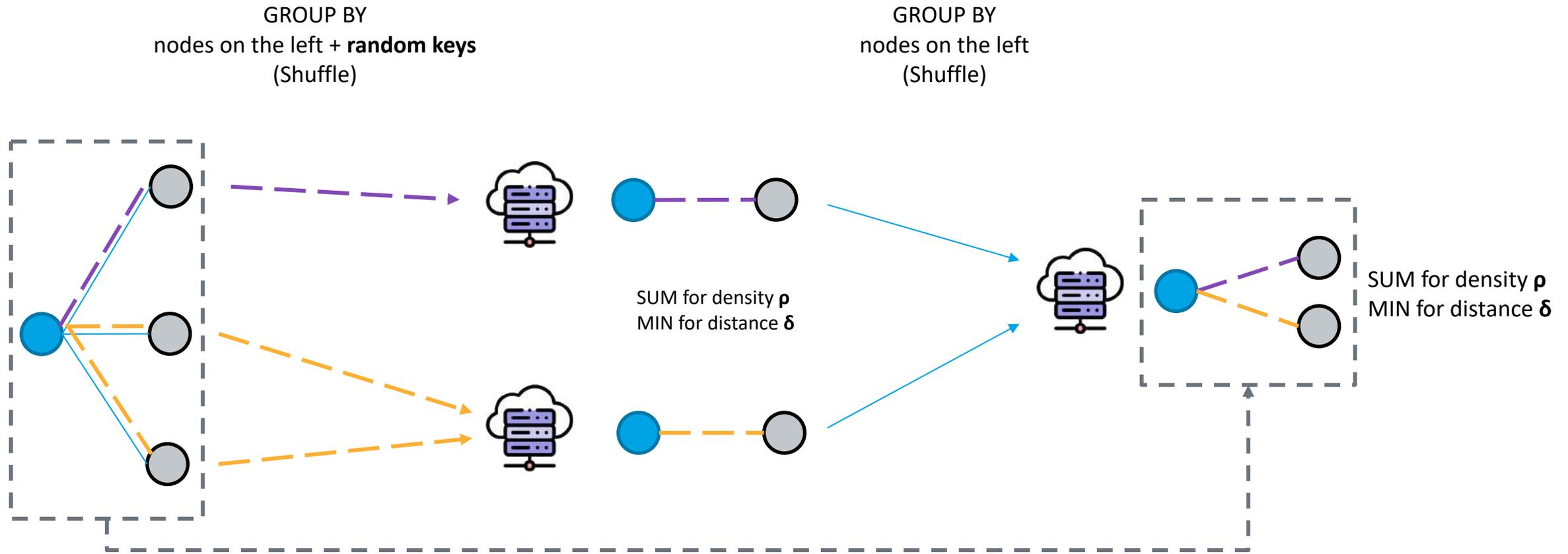
Salting Strategies for Group By Operations

- The second stage to compute the Density Peaks can also be parallelized.



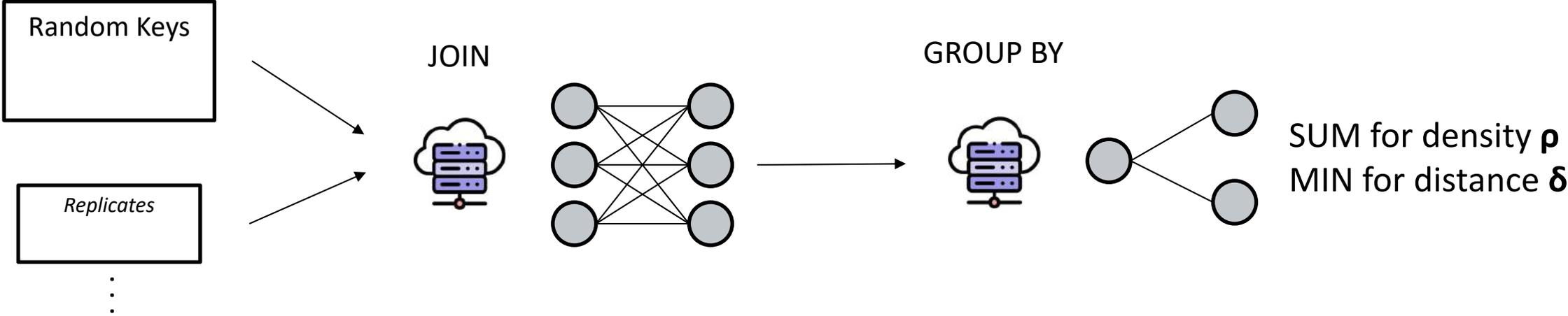
Salting Strategies for Group By Operations

- The second stage to compute the Density Peaks can also be parallelized.



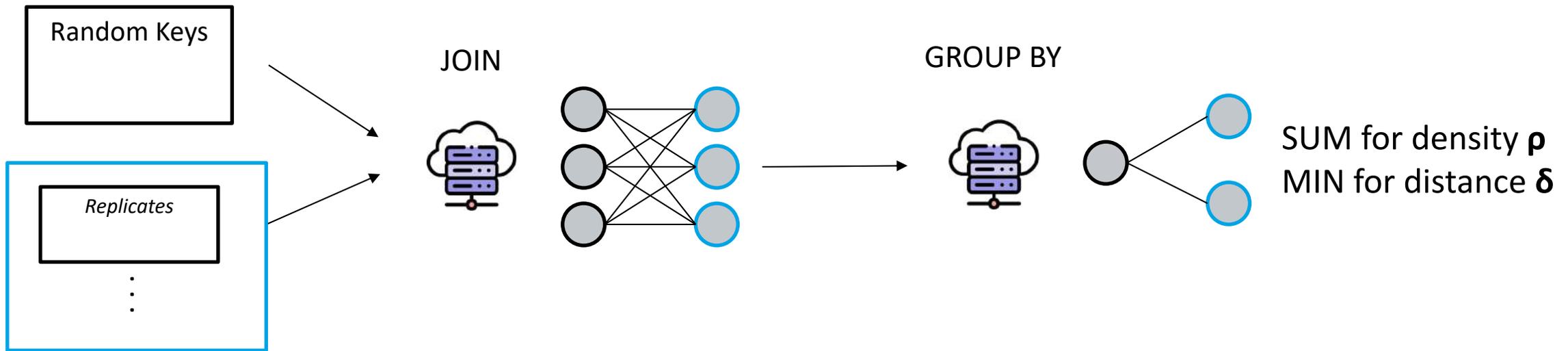
Batch Processing in the Pipeline

- The whole pipeline allows taking batches one by one.



Batch Processing in the Pipeline

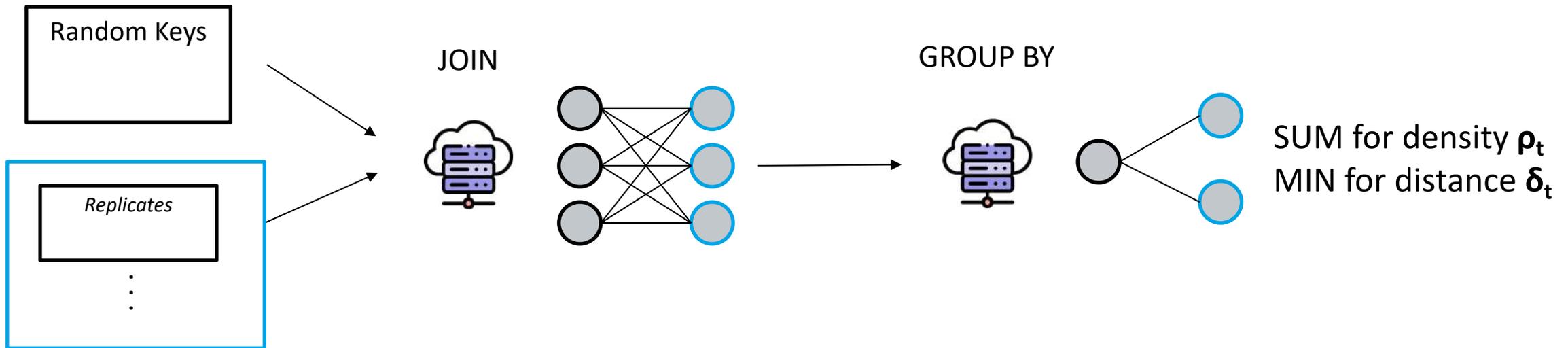
- The whole pipeline allows taking batches one by one.



Instead of taking all data, we replace it with replicate for a small batch.

Batch Processing in the Pipeline

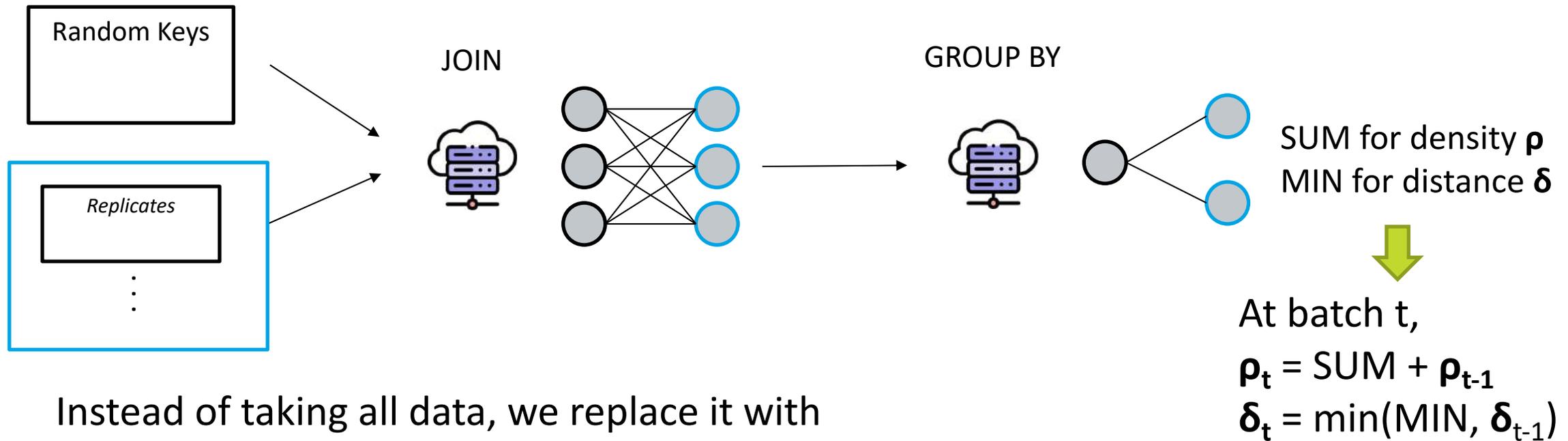
- The whole pipeline allows taking batches one by one.



Instead of taking all data, we replace it with replicate for a small batch.

Batch Processing in the Pipeline

- The whole pipeline allows taking batches one by one.



Instead of taking all data, we replace it with replicate for a small batch.

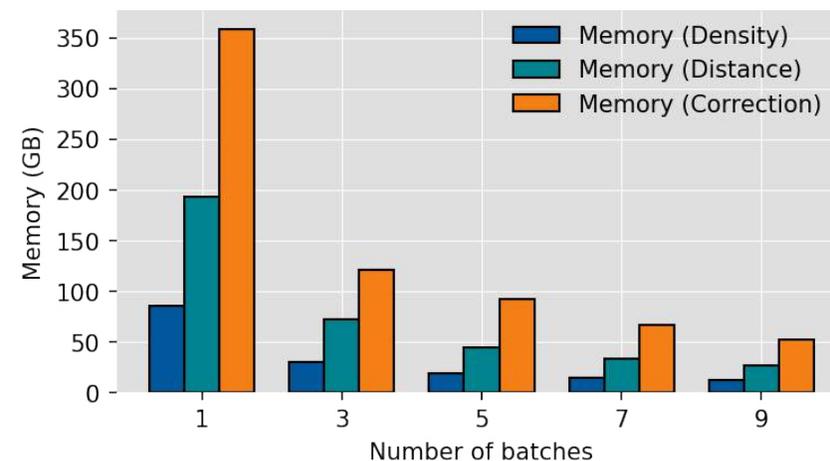
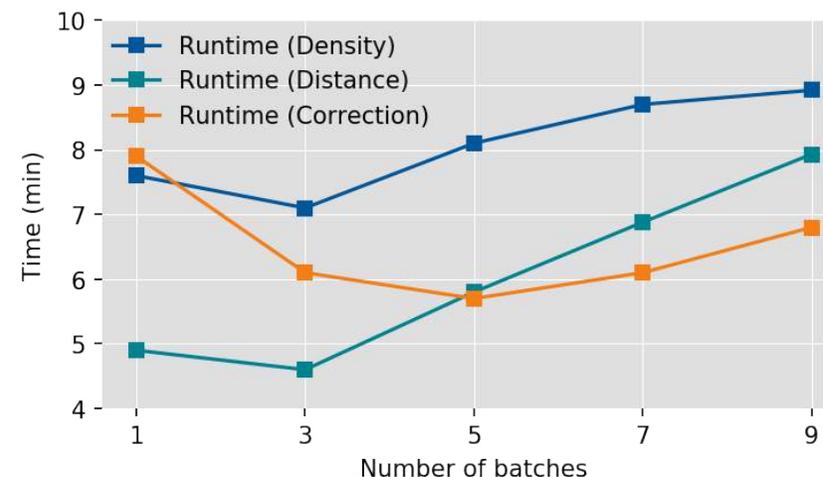
Tradeoff between Number of Batches and Shuffles

The greater number of batches, the more shuffle stages are needed.

The greater number of batches, the fewer chances to have long bottlenecks and large shuffle data.

∴ While increasing number of batches can reduce the memory needed, there is an equilibrium in runtime.

The equilibrium depends on hardware specifications.



User Interface for Interactive Visitor Clustering

The screenshot displays the 'Audience Clustering' interface. At the top, there are navigation icons and a 'Create Segment' button. Below the title, there are tabs for 'Overview' and 'Browse'. The main workspace is divided into two panels, each with 'Options', 'SPLIT', 'MERGE', 'CLEAR', and 'RUN' buttons. The left panel shows a single cluster of 29,378,469 (100%) with split criteria of 'Health Fitness & Beauty | Brand: Fitbit | View, Jonesboro AR, Little Rock-Pine Bluff AR ...' and a 'split by' value of 10. The right panel shows five clusters, each with its own split criteria and a 'split by' field. Dotted lines connect the left cluster to the first cluster in the right panel.

Cluster Size (Count)	Cluster Size (Percentage)	Split Criteria	Split By
29,378,469	100%	Health Fitness & Beauty Brand: Fitbit View, Jonesboro AR, Little Rock-Pine Bluff AR ...	10
14,851,503	50%	Media Exposure Activity, Starcom Activity, DCM Activity ...	
7,209,817	24%	Media Exposure Classification Digital Insert, Media Exposure Multi-Category, Media Exposure Funding Source MDF ...	
6,295,655	21%	Media Exposure Major Appliances, Classification NONE, Classification NONE ...	
604,214	2%	Media Exposure Home Theater, Classification NONE, None Activity ...	
358,853	1%	Email Opens, Email Opens, Email Opens PartyID Sync CID ...	

User Interface for Interactive Visitor Clustering

The screenshot displays the 'Audience Clustering' interface. On the left, a large segment is defined by 'split by 10' and contains 29,378,469 visitors (100%). This segment is split into five smaller segments on the right, each with its own 'split by' field and a 'select' checkbox. A bracket on the right side of the interface groups these five segments together, with a purple text annotation: 'Interactively segment the visitors into 3 groups'.

Segment	Count	Percentage	Attributes
Initial Segment	29,378,469	100%	Health Fitness & Beauty Brand: Fitbit View, Jonesboro AR, Little Rock-Pine Bluff AR ...
Segment 1	14,851,503	50%	Media Exposure Activity, Starcom Activity, DCM Activity ...
Segment 2	7,209,817	24%	Media Exposure Classification Digital Insert, Media Exposure Multi-Category, Media Exposure Funding Source MDF ...
Segment 3	6,295,655	21%	Media Exposure Major Appliances, Classification NONE, Classification NONE ...
Segment 4	604,214	2%	Media Exposure Home Theater, Classification NONE, None Activity ...
Segment 5	358,853	1%	Email Opens, Email Opens, Email Opens PartyID Sync CID ...

User Interface for Interactive Visitor Clustering

Audience Clustering

Overview Browse

Options SPLIT MERGE CLEAR RUN

- 29,378,469 (100%) split by 10 select
Health Fitness & Beauty | Brand: Fitbit | View, Jonesboro AR, Little Rock-Pine Bluff AR ...

Options SPLIT MERGE CLEAR RUN

- 14,851,503 (50%) split by select
Media Exposure Activity, Starcom Activity, DCM Activity ...
- 7,209,817 (24%) split by select
Media Exposure | Classification | Digital Insert, Media Exposure | Multi-Category, Media Exposure | Funding Source | MDF ...
- 6,295,655 (21%) split by select
Media Exposure | Major Appliances, Classification | NONE, Classification | NONE ...
- 604,214 (2%) split by select
Media Exposure | Home Theater, Classification | NONE, None Activity ...
- 358,853 (1%) split by 5 select
Email | Opens, Email Opens, Email Opens | PartyID Sync CID ...

Options SPLIT MERGE CLEAR RUN

- 276,588 (77%) split by select
Los Angeles CA, Chicago IL, San Francisco-Oakland-San Jose CA ...
- 3,379 (0%) split by select
Des Moines-Ames IA, View Type | Apps, Local Store Pages ...
- 78,886 (21%) split by select
New York NY, Boston MA-Manchester NH, St. Louis MO ...

Further split a selected segment

User Interface for Interactive Visitor Clustering

The screenshot displays the Audience Clustering interface. On the left, a list of segments is shown with their sizes and percentages. On the right, a table lists attributes with their popularity and influence scores. A purple callout points to the 'New York NY' segment in the table.

Segments:

- 14,851,503 (50%) - Media Exposure Activity, Starcom Activity, DCM Activity ...
- 7,209,817 (24%) - Media Exposure | Classification | Digital Insert, Media Exposure | Multi-Category, Media Exposure | Funding Source | MDF ...
- 6,295,655 (21%) - Media Exposure | Major Appliances, Classification | NONE, Classification | NONE ...
- 604,214 (2%) - Media Exposure | Home Theater, Classification | NONE, None Activity ...
- 358,853 (1%) - Email | Opens, Email Opens, Email Opens | PartyID Sync CID ...

Attributes Table:

Attribute Name	Popularity	Influence
New York NY	45%	1.0
Boston MA-Manchester NH	34%	0.76
St. Louis MO	14%	0.31
Ft. Myers-Naples FL	6%	0.13
ALF Imp	0%	0.0012
Impression Test	0%	0.0012
DCM Activity	0%	0.0012
Starcom Activity	0%	0.0012
Washington DC (Hagerstown MD)	0%	0.0012
Media Exposure Activity	0%	0.0012

Inspect the useful traits inside a destined segment

Conclusions and Future Work

- Applying LSH and DP clustering to enable interactive clustering on sparse online visitor data.
- Designing speed up strategies for clustering pipeline in a distributed environment.
- Future Work:
 - Further reduce the uneven data distribution in LSH Join
 - The increasing number of hash tables in LSH worsen the data distribution easily.



Thank You

Twitter: @GromitC

Email: gromit.chan@nyu.edu