K-Means in Clustering Flights Delays

Group 6:
JL, LQ,
ZW, JW

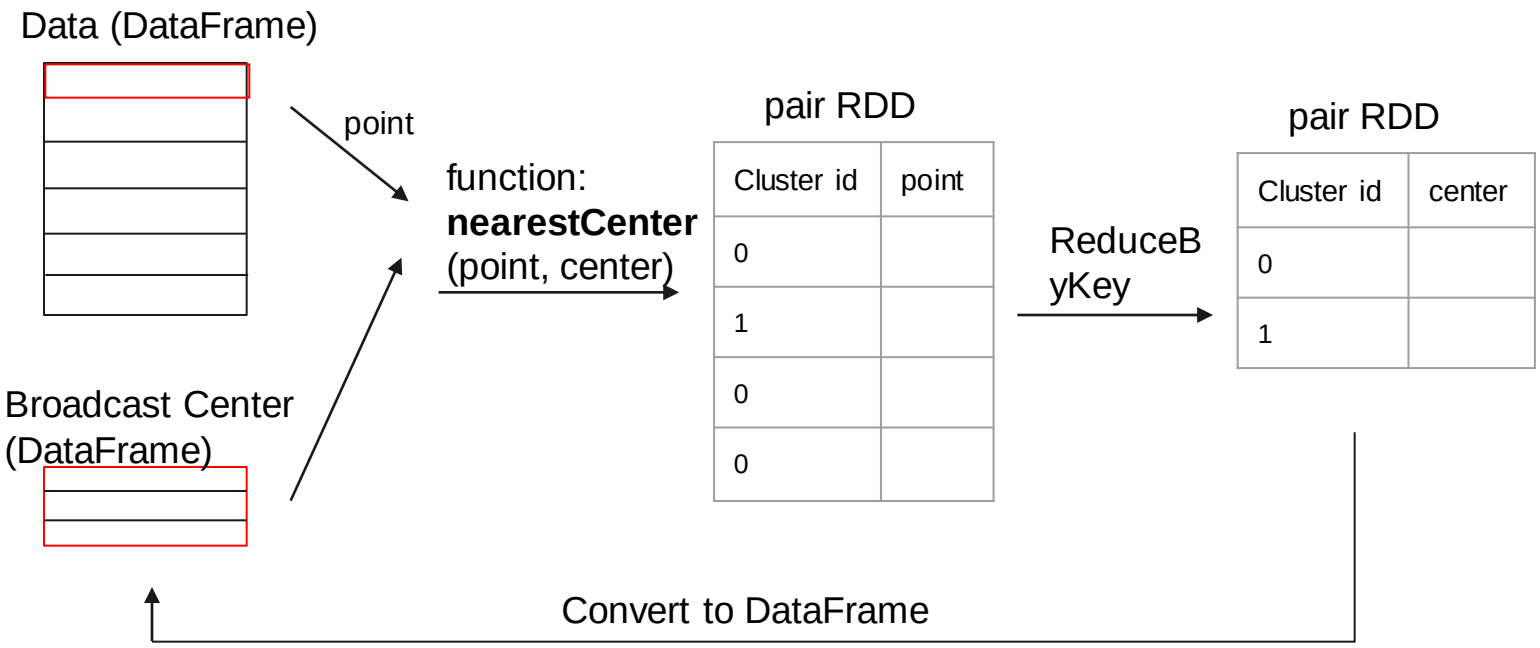
Data & Tasks

Data: 2 million records, each with 13 attributes,
collected from U.S. Department of Transportation

Task 1:
the distributed K-means clustering algorithm that uses multiple tasks for each iteration

Task 2:
the local K-means algorithm that computes an entire clustering for a single K in a single task.

Task 1: Workflow



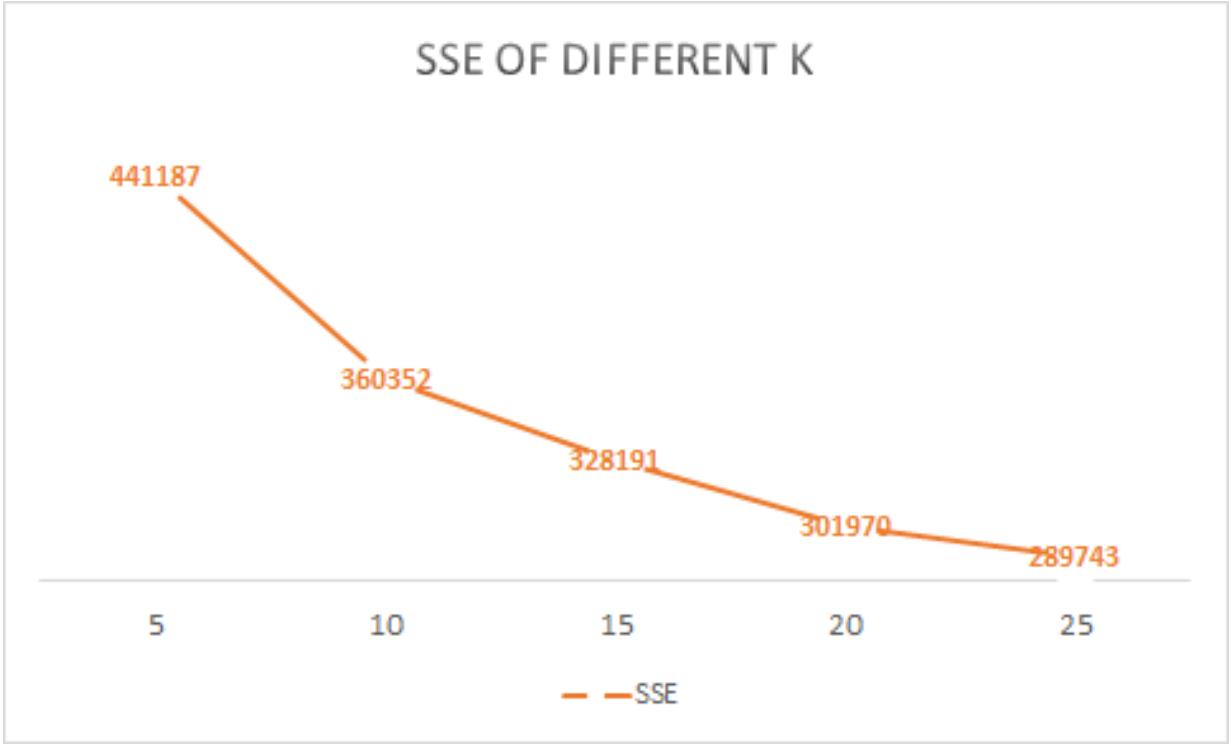
Task 1: Challenges

Problem: When we try to change the variable inside foreach loop, this change will not be applied after we end the loop (in re-assignment part)

Solution: Use map to calculate new cluster id

Task 1: the best k

1 million data
with 5 machines



Task 1

Speedup:

Number of workers(1 million)	**Running time**	**Number of workers(2 million)**	**Running time**
5	303s	5	960s
10	194s	10	384s

Scaleup:

Input size	**Running time change**
100,000	+ 37s (97s)
1,000,000	+ 206s (303s)
2,000,000	+ 657s (960s)

Data & Tasks

Data: 2 million records, each with 13 attributes,
collected from U.S. Department of Transportation

Task 1:
the distributed K-means clustering algorithm that uses multiple tasks for each iteration

Task 2:
the local K-means algorithm that computes an entire clustering for a single K in a single task.

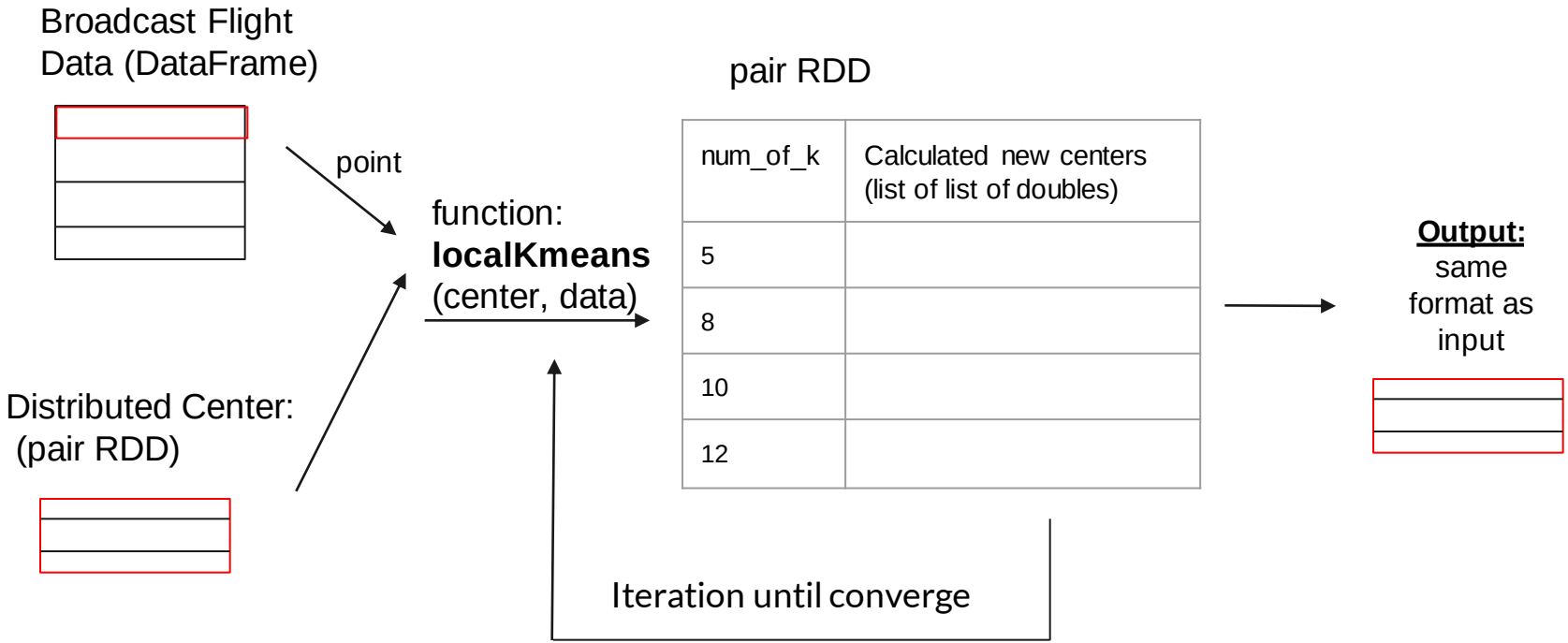
Task 2: Input

- Two Input
  - Pre-processed flight data
  - File with various k and their randomly selected initial centers. (format: num_of_k@center_1~center_2~...)

Example:

2@0.9397260273972603,0.16666666666666666,0.8542538354253835,0.9067501739735561,0.8895066018068102,0.946490618485059,0.0743801652892562,0.08454106280193237,0.037656903765690 38,0.07432432432432433,0.047018115204559334,0.04761904761904761 6,0.07027027027027027~0.021917808219178082,0.33333333333333337,0.6311018131101812,0.7362560890744607,0.6532314107018763,0.7505211952744961,0.2975206611570248,0.05475040257648953,0.02092050209205021,0.2922297297297297,0.2468959902300020 5,0.07142857142857142,0.035135135135135 14

Task 2: Workflow



Task 2: Challenges

- Change the partition size of the algorithm.
- Increase the heap size, so aws could handle larger amount of input data.
  - 1g (default) -> up to 250,000 rows
  - 3g -> 1,000,000 rows
- Increase executors' memory as the broadcast variable becomes larger.

Task 2

Speedup:

- Default Partition Size: 2 when input is 100,000
- We increase the partition size to 10 and 26.

Input size	Partition Size	Running time change
1,000,000, k = 2- 27	10 (5 machine)	1 hour 21 min
1,000,000, k = 2- 27	20 (5 machine)	57min (24 min less)
1,000,000, k = 2- 27	20 (10 machine)	57min (same with 5 machines)
1,000,000, k = 2- 27	26 (7 machine)	45min (36min less)

Task 2

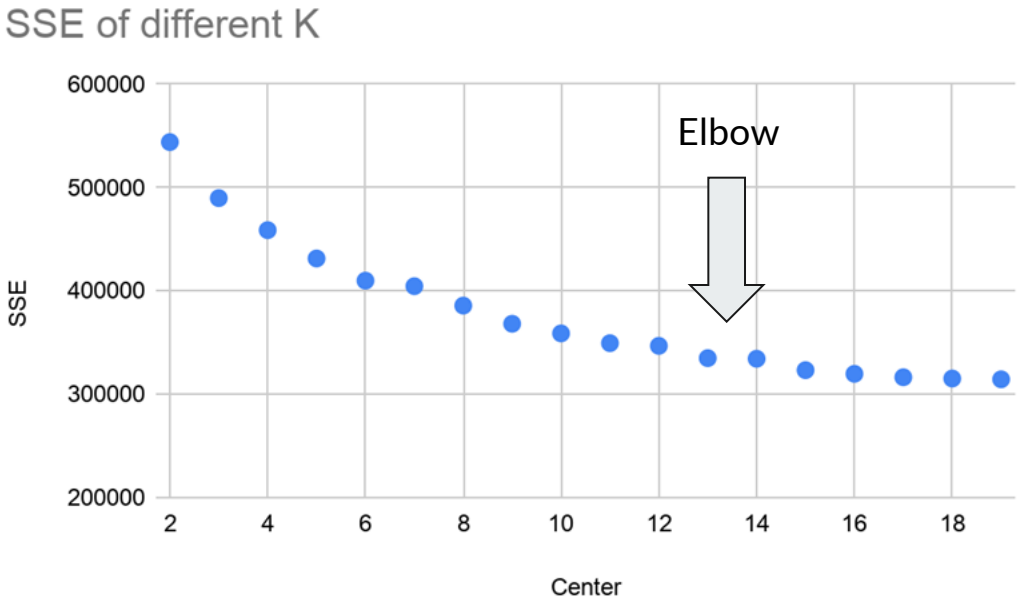
Scaleup:

- With 27 different k value
- 5 Machine with 20 partition

Input size	Memory Size	Running time change
100,000	Default (1g driver memory)	5min (cluster < 23, as when k > 23 there are null clusters appear)
500,000	3g driver memory	26min
1,000,000	3g driver memory	57min
2,000,000	6g driver memory 3/6g executor memory	Have out of memory exception with regular m5.xlarge machines

Task 2: Elbow Method Finding Best K

- 1 Million row of data
- Experimented k range from 2 to 27
- The elbow is found around k = 12 ~ 15
- Only need one run



Conclusion

- Both programs show good speedup and scaleup
- Local Kmeans is good at running multiple results within one run. Easier and faster for finding the most suitable k value.
- Distributed Kmeans is better at handling more data for one run than local Kmeans.