Parallel K-Means

outline

- Introduction
- K-Means Algorithm
- Parallel K-Means Based on MapReduce
- Experimental Results
- K-Means on spark

Introduction

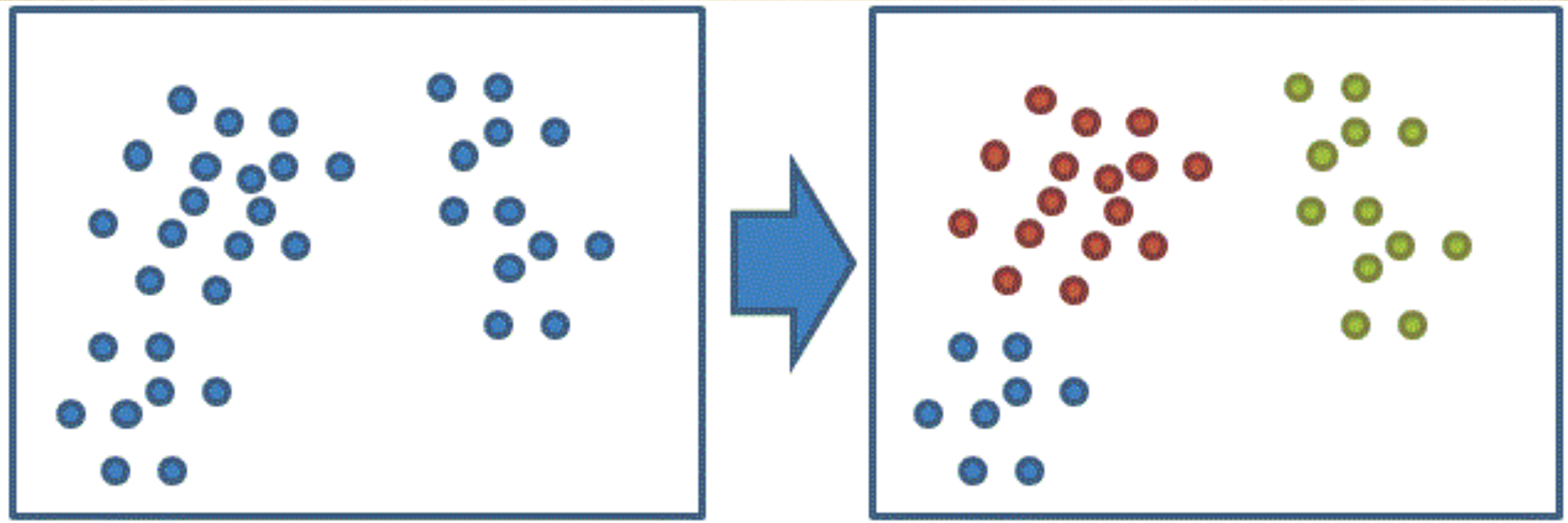
- They assume that all objects can reside in main memory at the same time.
- Their parallel systems have provided restricted programming models.

Introduction

- They assume that all objects can reside in main memory at the same time.
- Their parallel systems have provided restricted programming models.

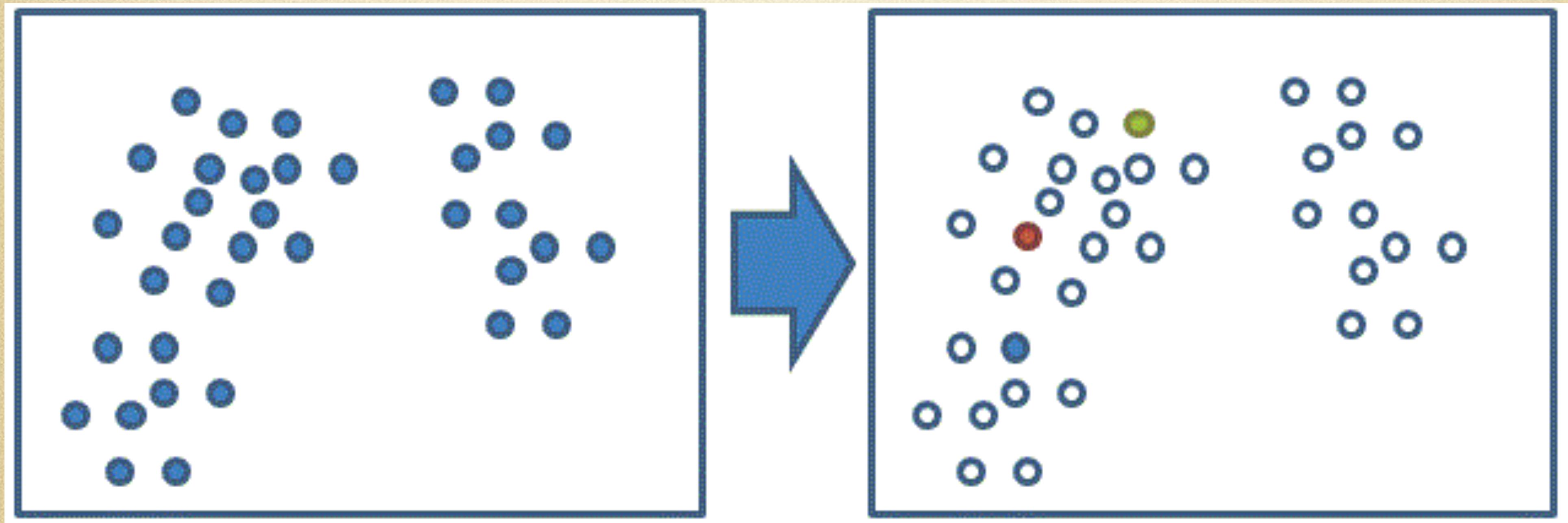
dataset oriented parallel clustering algorithms should be developed.

K-Means Algorithm



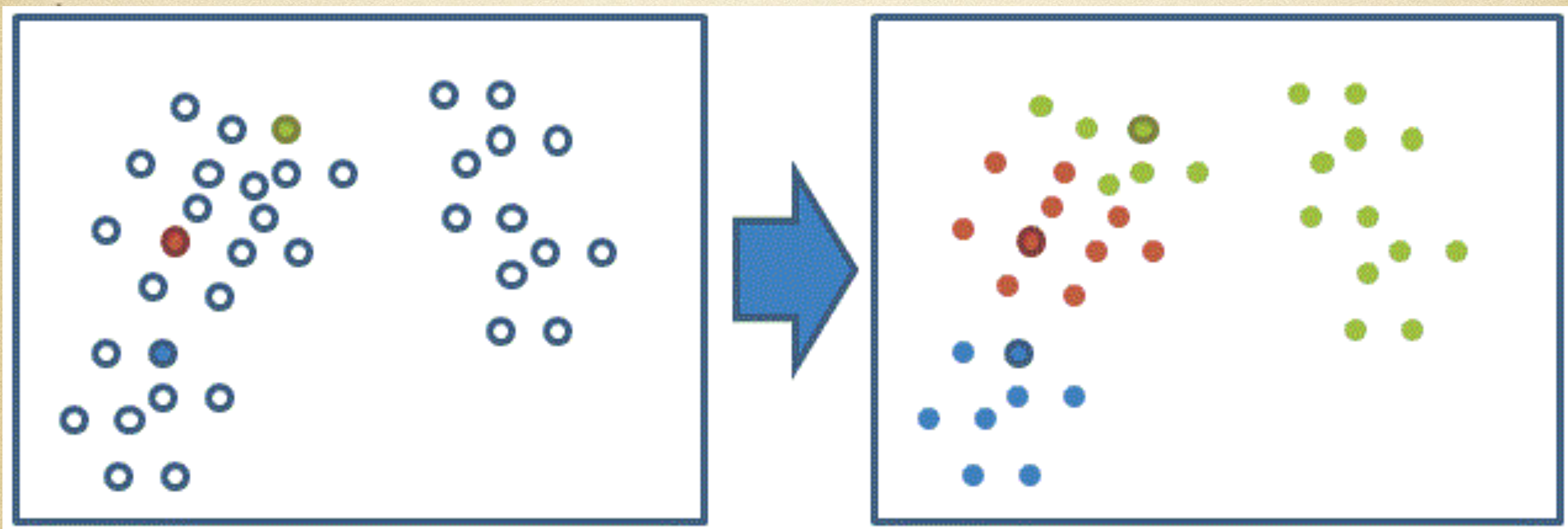
K-Means Algorithm

Firstly, it randomly selects k objects from the whole objects which represent initial cluster centers.



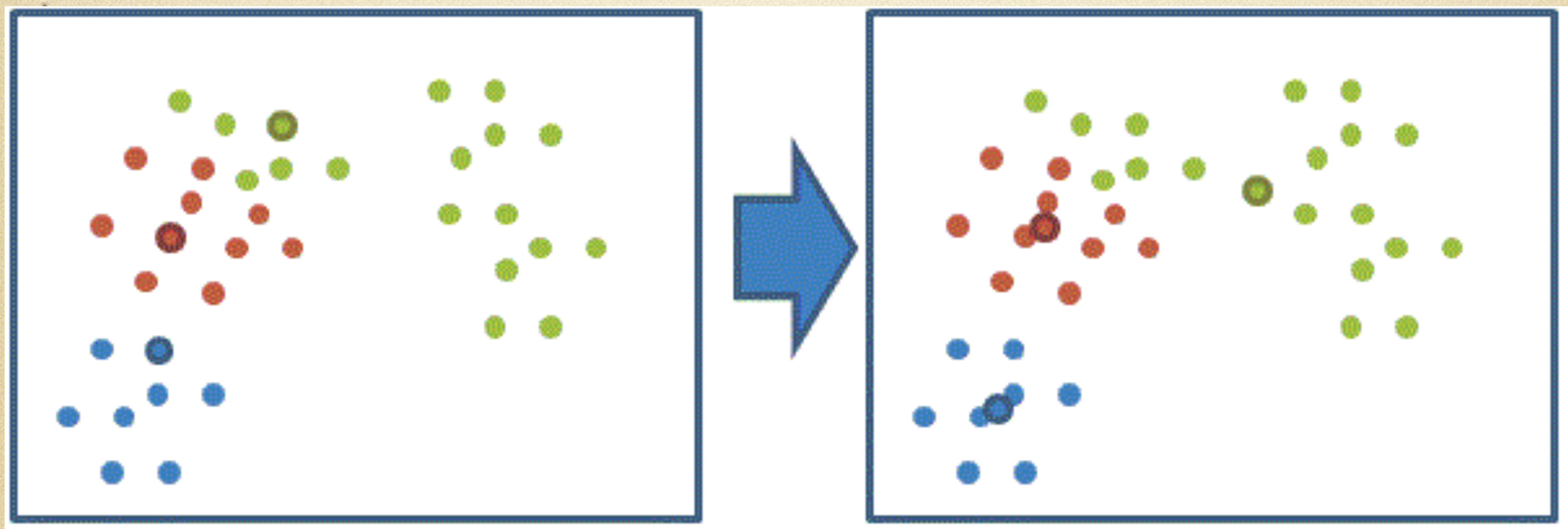
K-Means Algorithm

Each remaining object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster center.



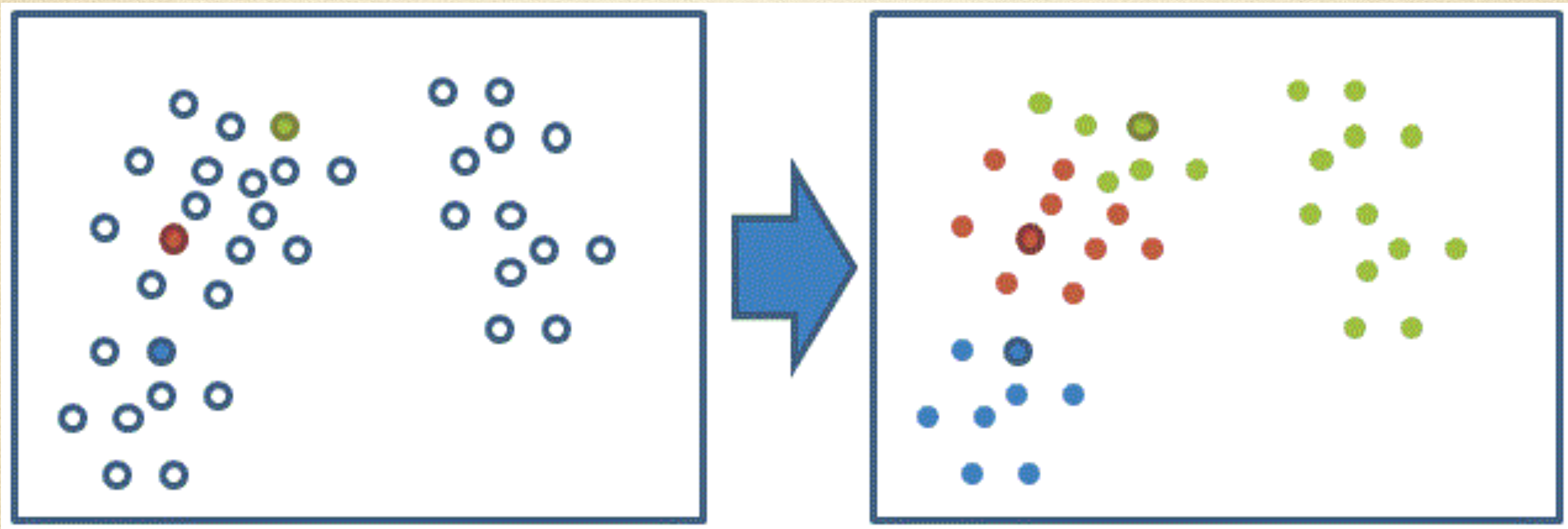
K-Means Algorithm

The new mean for each cluster is then calculated. This process iterates until the criterion function converges.



Parallel K-Means Based on MapReduce

most intensive calculation to occur is the calculation of distances.



each iteration require nk distance

Parallel K-Means Based on MapReduce

the distance computations between one object with the centers is irrelevant to the distance computations between other objects with the corresponding centers.

distance computations between different objects with centers can be parallel executed.

Parallel K-Means Based on MapReduce

data

1,1
2,2
3,3
11,11
12,12
13,13

target

1 class

1,1
2,2
3,3

2 class

11,11
12,12
13,13

Parallel K-Means Based on MapReduce

1,1
2,2
3,3
11,11
12,12
13,13

random two centroid

c1:(1,1)
c2:(11,11)

Parallel K-Means Based on MapReduce

c1:(1,1)
c2:(11,11)

1,1
2,2
3,3
11,11
12,12
13,13

store two nodes

Parallel K-Means Based on MapReduce

c1:(1,1)

c2:(11,11)

1,1
2,2
3,3
11,11
12,12
13,13

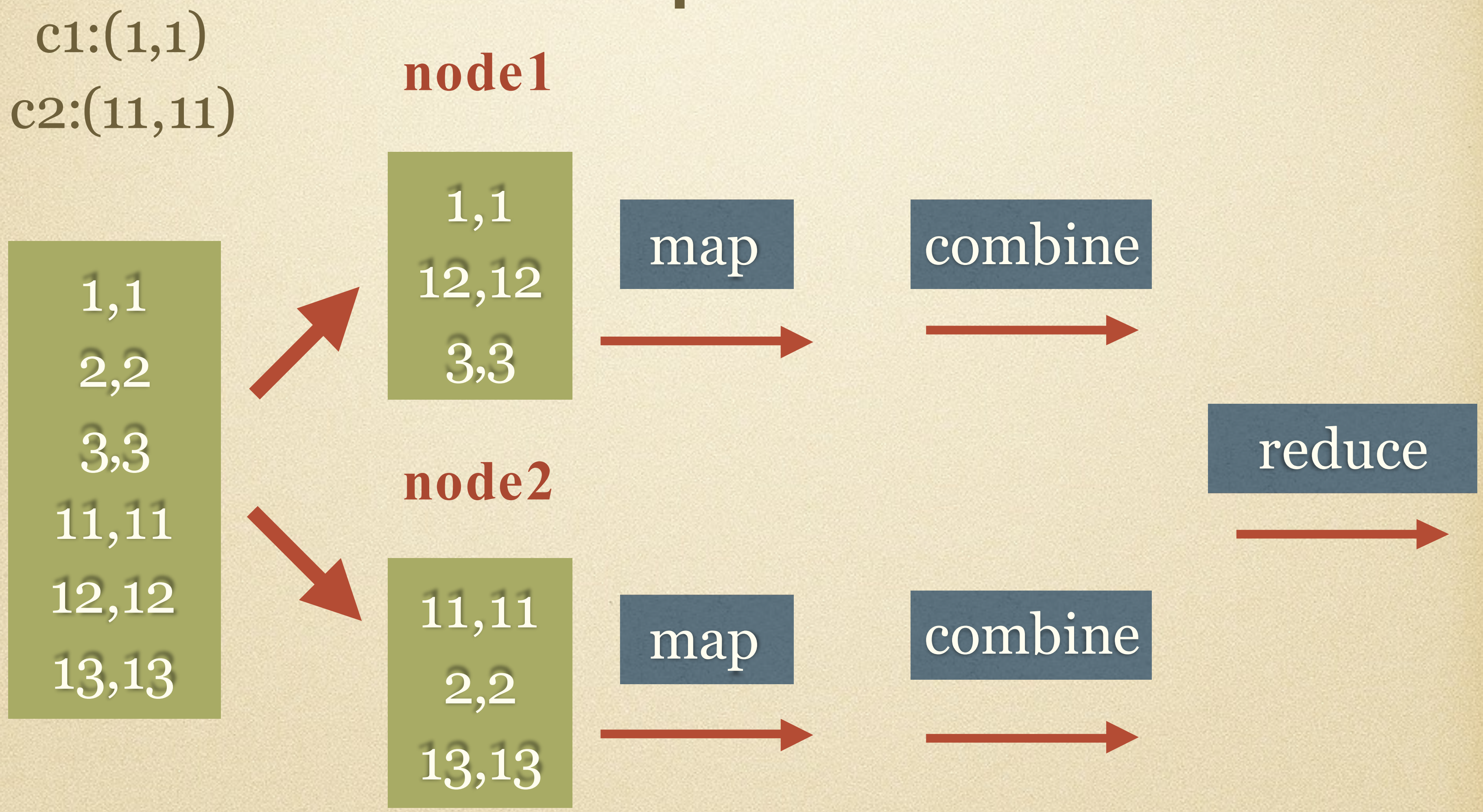
node1

1,1
12,12
3,3

node2

11,11
2,2
13,13

Parallel K-Means Based on MapReduce



Parallel K-Means Based on MapReduce

node1

1,1
12,12
3,3

map

3,3

c1:(1,1)
c2:(11,11)

assign to c1(1,1)

output<key,value>

key	**value**
(1,1) ,	{(3,3),(3,3)}

Parallel K-Means Based on MapReduce

key **value**

output<key,value>

(1,1) , {(3,3),(3,3)}

(1,1) **centroid**

{(3,3),(3,3)}

temporary to calculate new centroid, the object

Parallel K-Means Based on MapReduce

c1:(1,1)

c2:(11,11)

node1

1,1
12,12
3,3

map →

key	**value**
(1,1)	{(1,1),(1,1)}
(11,11)	{(12,12),(12,12)}
(1,1)	{(3,3),(3,3)}

Parallel K-Means Based on MapReduce

node1

1,1
12,12
3,3

c1:(1,1)
c2:(11,11)

map →

key	value
(1,1)	{(1,1),(1,1)}
(11,11)	{(12,12),(12,12)}
(1,1)	{(3,3),(3,3)}

node2

11,11
2,2
13,13

map →

key	value
(11,11)	{(11,11),(11,11)}
(1,1)	{(2,2),(2,2)}
(11,11)	{(13,13),(13,13)}

Parallel K-Means Based on MapReduce

node1

1,1
12,12
3,3

c1:(1,1)
c2:(11,11)

map

key **value**

(1,1) , {(1,1),(1,1)}

(11,11) , {(12,12),(12,12)}

(1,1) , {(3,3),(3,3)}

combine

Parallel K-Means Based on MapReduce

key **value**

(1,1) , {(1,1),(1,1)}

(11,11) , {(12,12),(12,12)}

(1,1) , {(3,3),(3,3)}

same key combine

combine

key **value**

(1,1) , {(4,4),{(1,1),(3,3),2}

(11,11) , {(12,12),(12,12),1}

Parallel K-Means Based on MapReduce

key **value**

output<key,value>

(1,1) , {(4,4),{(1,1),(3,3)},2}

(11,11) , {(12,12),(12,12),1}

(1,1) **centroid**

{(4,4),{(1,1),(3,3)},2}

temporary to calculate new centroid, the objects ,number of objects

Parallel K-Means Based on MapReduce

key **value**

(1,1) , {(1,1),(1,1)}

(11,11) , {(12,12),(12,12)}

(1,1) , {(3,3),(3,3)}

combine →

key **value**

(1,1) , {(4,4),{(1,1),(3,3)},2}

(11,11) , {(12,12),(12,12),1}

key **value**

(11,11) , {(11,11),(11,11)}

(1,1) , {(2,2),(2,2)}

(11,11) , {(13,13),(13,13)}

combine →

key **value**

(1,1) , {(2,2),(2,2),1}

(11,11) , {(24,24),{(11,11),(13,13)},2}

Parallel K-Means Based on MapReduce

key **value**

(1,1) , {(4,4),{(1,1),(3,3)},2}

(11,11) , {(12,12),(12,12),1}

same key reduce

reduce

key **value**

(1,1) , {(2,2),(2,2),1}

(11,11) , {(24,24),{(11,11),(13,13)},2}

Parallel K-Means Based on MapReduce

(1,1) , {(4,4),{(1,1),(3,3)},2}

same key reduce

(1,1) , {(2,2),(2,2),1}

reduce

(1,1) , {(2,2),{(1,1),(2,2),(3,3)}

Parallel K-Means Based on MapReduce

(1,1) , {(4,4),{(1,1),(3,3)},2}

(1,1) , {(2,2),(2,2),1}

(4+2)/(2+1) ,(4+2)/(2+1) = 2,2

2,2 = new centroid

(1,1) , {(2,2),{(1,1),(2,2),(3,3)}

1,1
2,2
3,3

centroid is 2,2

Parallel K-Means Based on MapReduce

(1,1) , {(4,4),{(1,1),(3,3)},2}

(1,1) , {(2,2),(2,2),1}

centroid

(1,1) , {(2,2),{(1,1),(2,2),(3,3)}

(1,1) , {(2,2),{(1,1),(2,2),(3,3)}

new centroid, the objects ,new cluster

Parallel K-Means Based on MapReduce

(1,1) , {(2,2),{(1,1),(2,2),(3,3)}

reduce

(11,11) , {(12,12),{(11,11),(12,12),(13,13)}

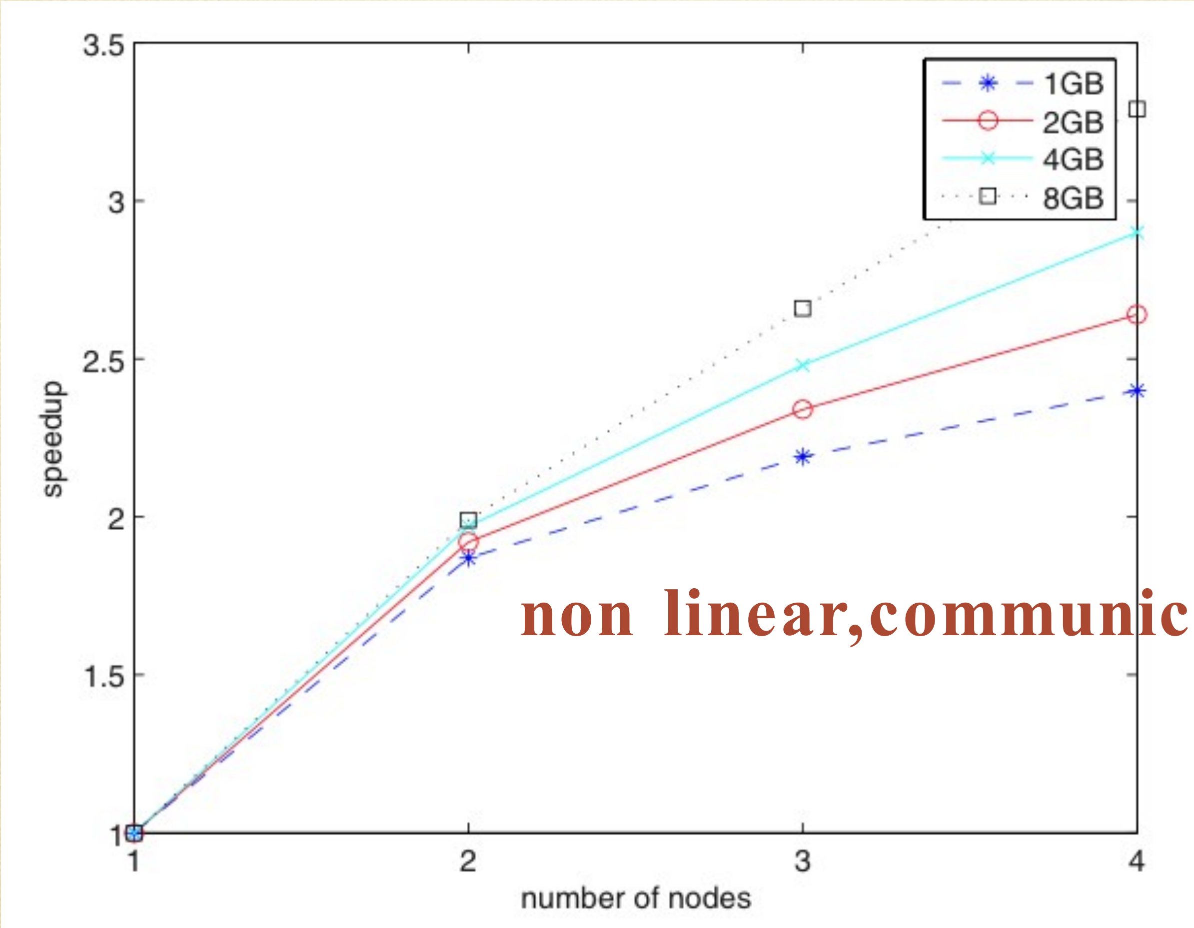
update new centroid and next iteration

until converge or arrive to iteration number

Experimental Results

two 2.8 GHz cores and 4GB of memory

Experimental Results

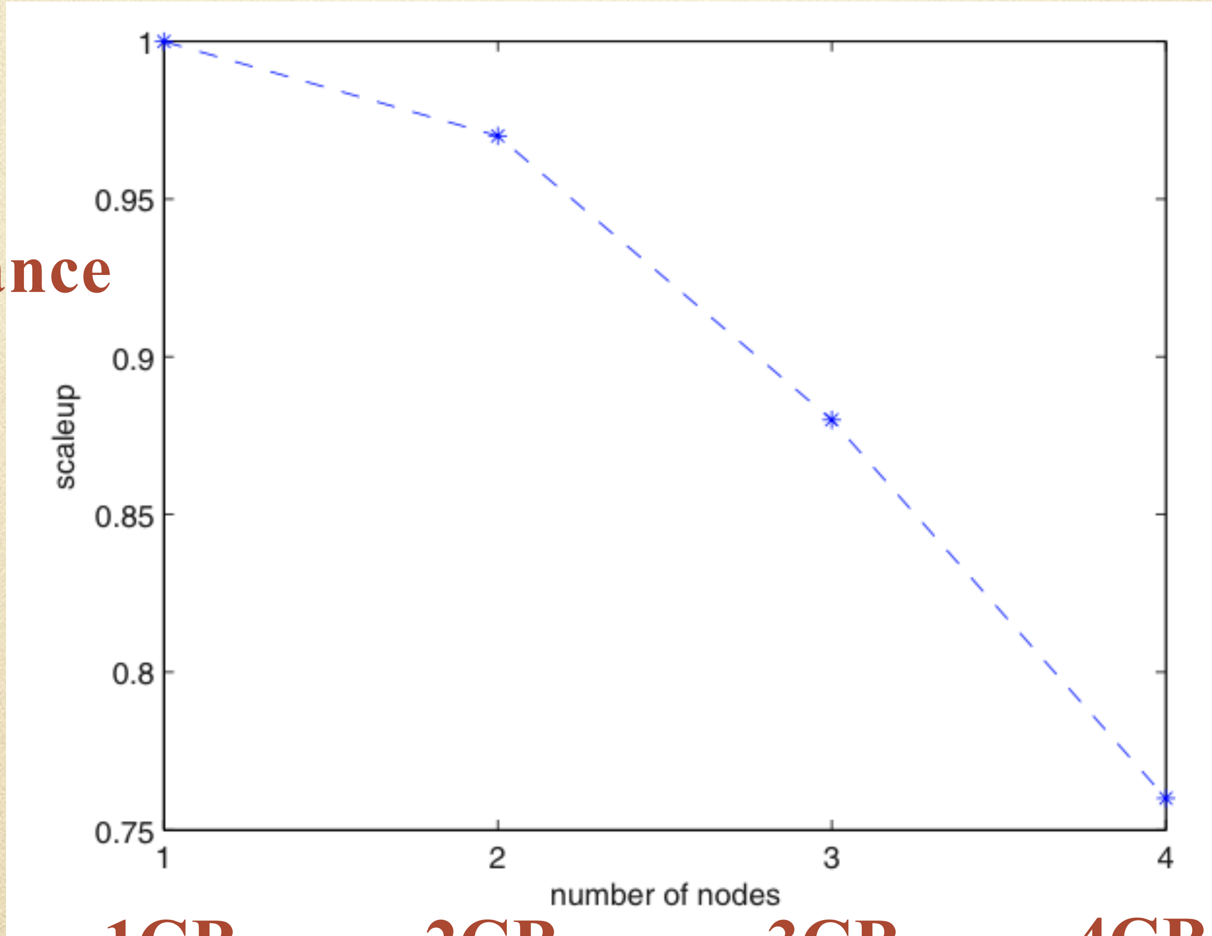


non linear,communication cost

Speedup

Experimental Results

performance



datasets 1GB 2GB 3GB 4GB

Scale up

K-Means on spark

Spark 1.1.0 Overview Programming Guides API Docs Deploying More

- *epsilon* determines the distance threshold within which we consider k-means to have converged.

Examples

Scala **Java** **Python**

The following code snippets can be executed in `spark-shell`.

In the following example after loading and parsing data, we use the `KMeans` object to cluster the data into two clusters. The number of desired clusters is passed to the algorithm. We then compute Within Set Sum of Squared Error (WSSSE). You can reduce this error measure by increasing *k*. In fact the optimal *k* is usually one where there is an "elbow" in the WSSSE graph.

```scala
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.linalg.Vectors

// Load and parse the data
val data = sc.textFile("data/mllib/kmeans_data.txt")
val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble)))

// Cluster the data into two classes using KMeans
val numClusters = 2
val numIterations = 20
val clusters = KMeans.train(parsedData, numClusters, numIterations)
```

Reference

K means algorithm

Parallel K-Means Clustering Based on MapReduce
Weizhong Zhao1,2, Huifang Ma1,2, and Qing He1
2009