Parallel clustering Using Hadoop

Department of Computer Science & Engineering

Table of Content

Hadoop

- Apache **Hadoop** is an open source software project that enables distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with very high degree of fault tolerance.

- A **Hadoop cluster** is a special type of computational **cluster** designed specifically for storing and analyzing huge amounts of unstructured data in a distributed computing environment.

Hadoop Architecture

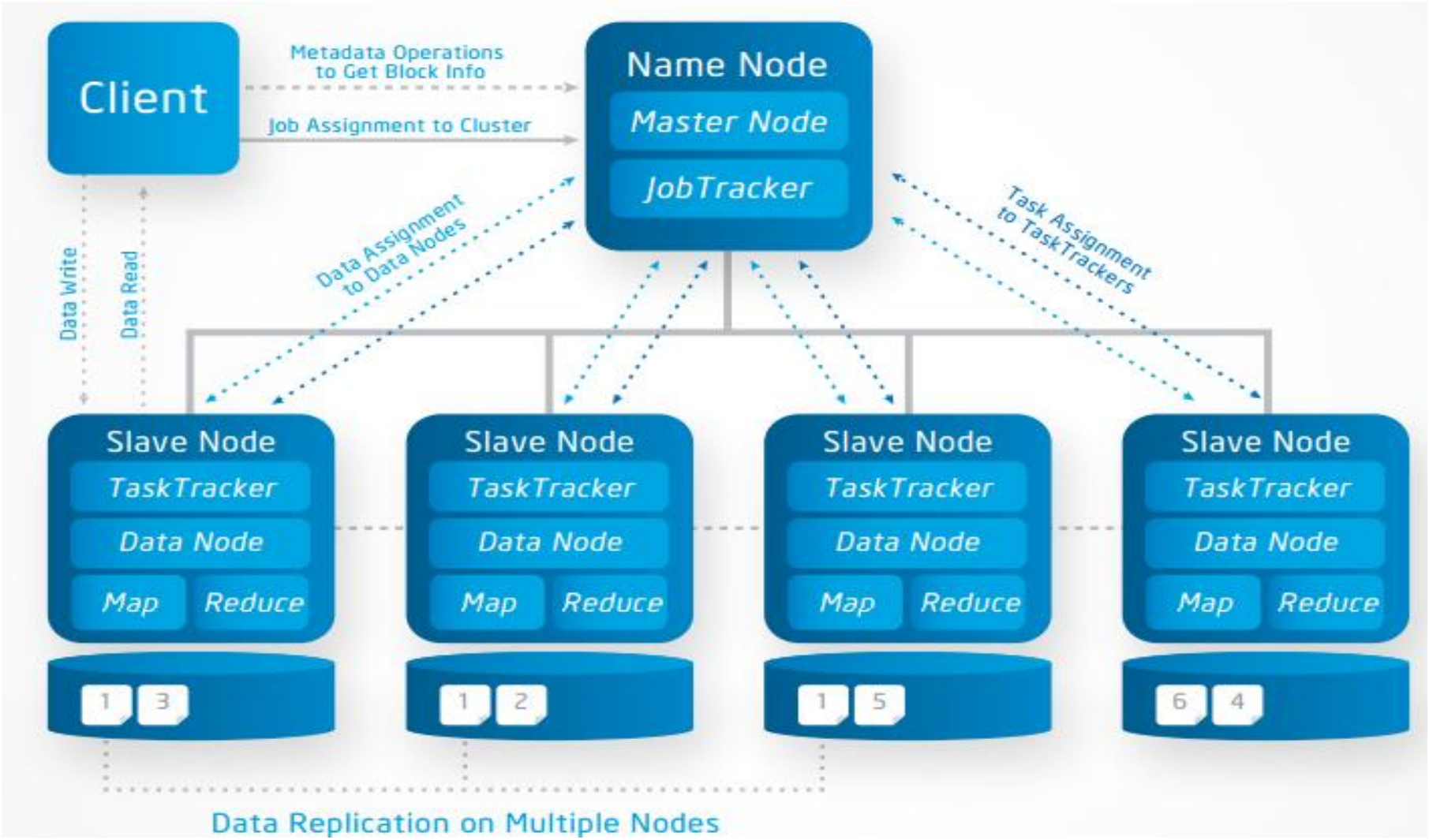


Fig 2: Hadoop Architecture [26]

The MapReduce Framework (pioneered by Google)

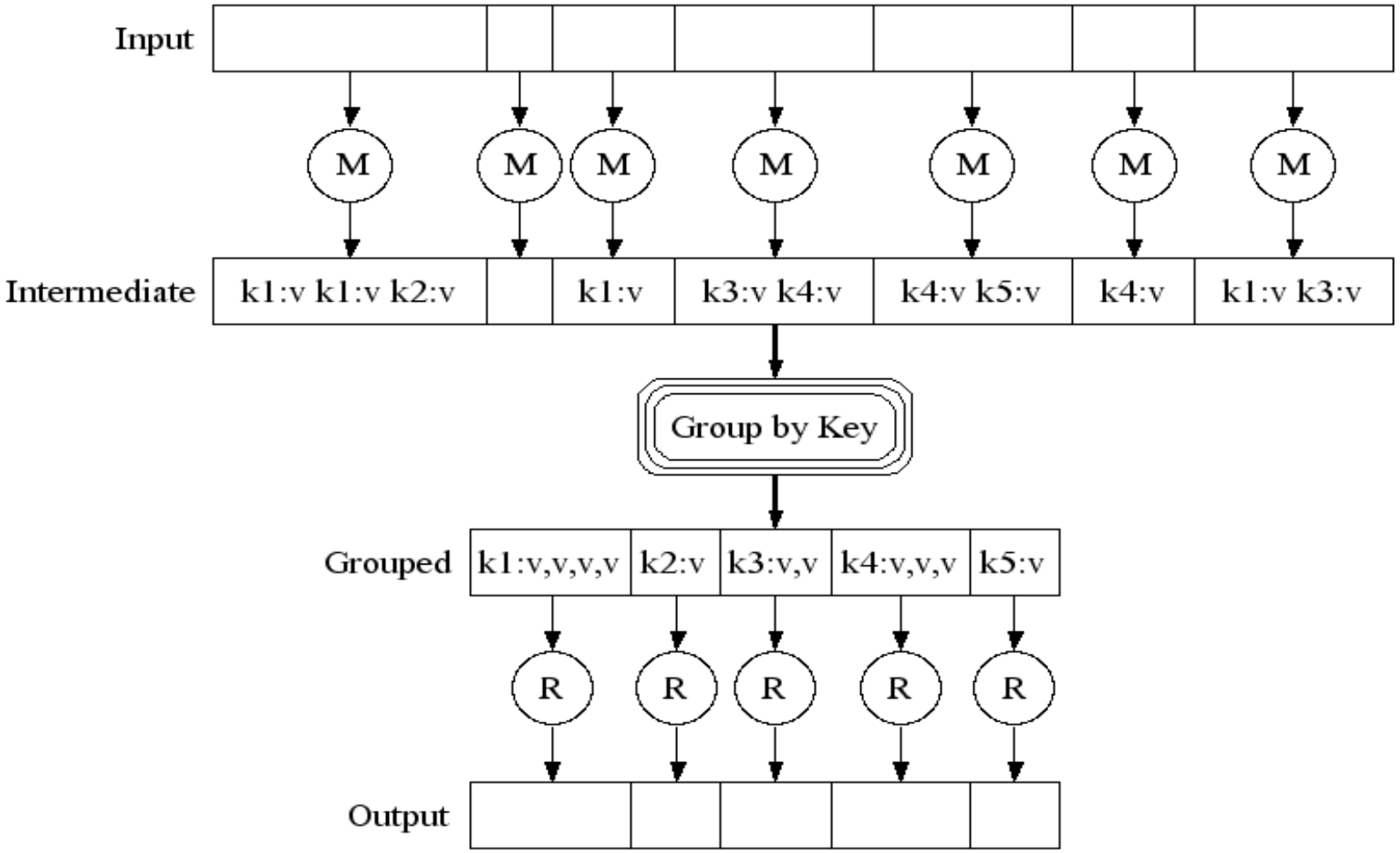


Fig 4:MRFramework[26]

File Formats

- TextInputFormat
- KeyValueTextInputFormat
- SequenceFileInputFormat
- SequenceFileAsTextInputFormat
  By defaullt it takes TextInput Format

MapReduce in Hadoop

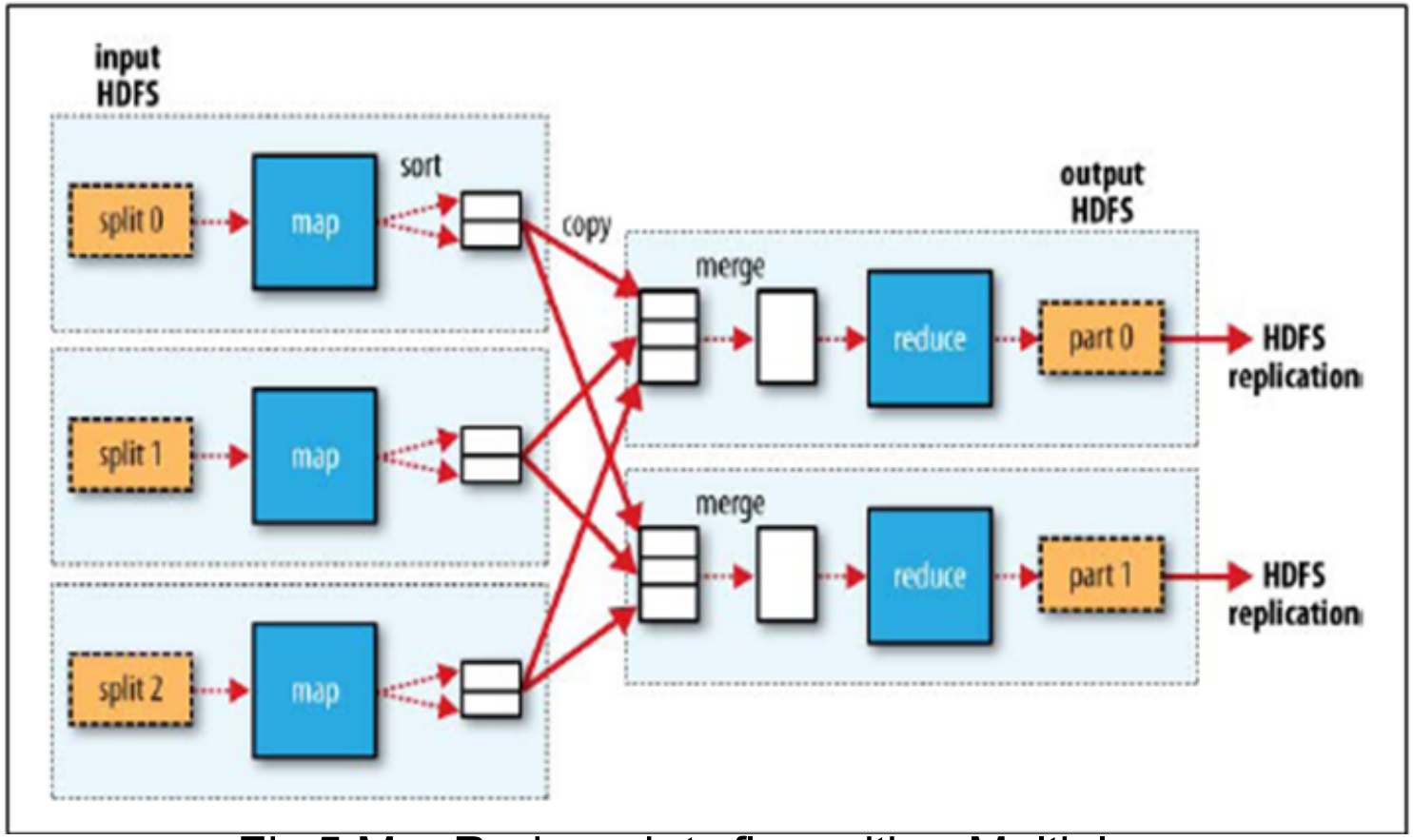
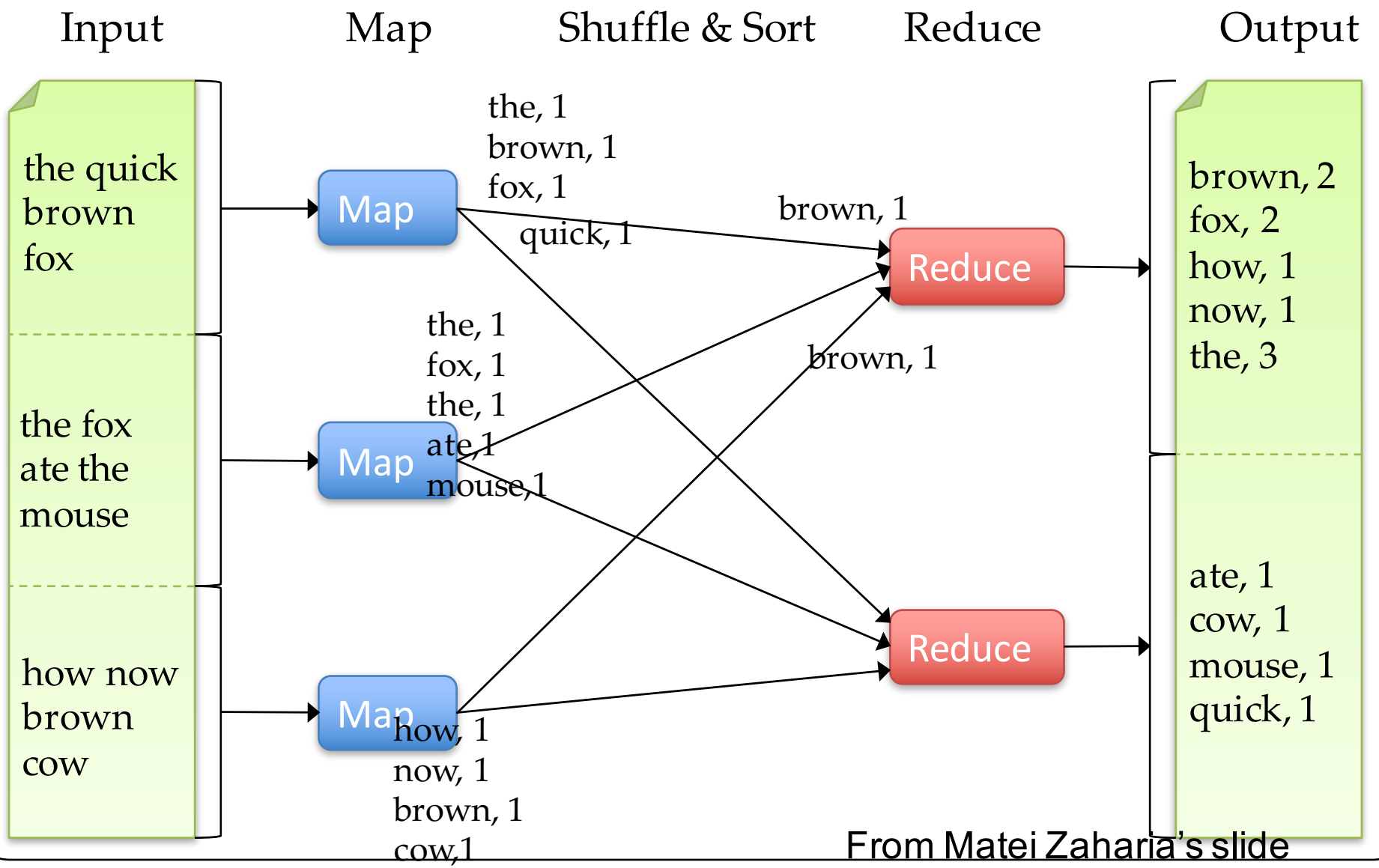


Fig 5:MapReduce data flow with a Multiple reduce task[27]

Word Count Execution



From Matei Zaharia's slide

MapReduce WordCount.java

(Mapper Code)

```
public static class Wordcount extends MapReduceBase implements Mapper <LongWritable, Text,
    Text, IntWritable>{
 public void map(Long Writable key, Text value, OutputCollector<Text ,IntWritable>output
   ,Reporter   r) throws IOException{
  String s =value.toString());
  for(String word:s.split(" ")){
        if(word.lenghth.>0)                    //regular expression
        {  output.collect(new Text(word), new IntWritable(1))   //collect is a method of output
   collector interface
  }
 }
```

Reduce code in WordCount.java

```
public class WordReducer  extends MapReduceBase Impliments
      Reducer<Text,IntWritable,Text,IntWritable> {
   public void reduce(Text key, Iterable<IntWritable> values,
                  OutputCollector<Text,Intwritable> output Reporter  ) throws IOException{
   int count =0;
  while(value.hasNext())   // this method checks that whether there is any value
{
      IntWritable i=value.Next();    //  this method get that value
      count +=i.get();                    // get method converts it to int(object type to primitive type.
              }

     output.collect(key,new   Intwritable(count));

  }
}
```

Reduce code in WordCount.java

```
Public class WordCounter {
Public static void main(String [] args) throws IOEXception, InterruptedException ,
      ClassNotFoundException{
 Job job =new Job();
 job.setJobName (“wordcounter ”);
 job.setJarByClass(WordCounter.class”);
job.setMapperClass(Wordcount.class)
job.setReduceClass(Wordreduce.class)
Job.setOutputKeyClass(Text.class);
Job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath (Job, new Path(“/sample/word.txt”));
FileOutputFormat.setOutputPath(job ,new Path (“/sample/wordcount”));
System.exit(job.waitForCompletion (true)? 90:1);
```

Algorithm *k-means*

Input-n Data points

Output-k cluster centroids.

1. Randomly choose K data items from X as initial centroids.
2. Repeat
   - Assign each data point to the cluster which has the closest centroid.
   - Calculate new cluster centroids.

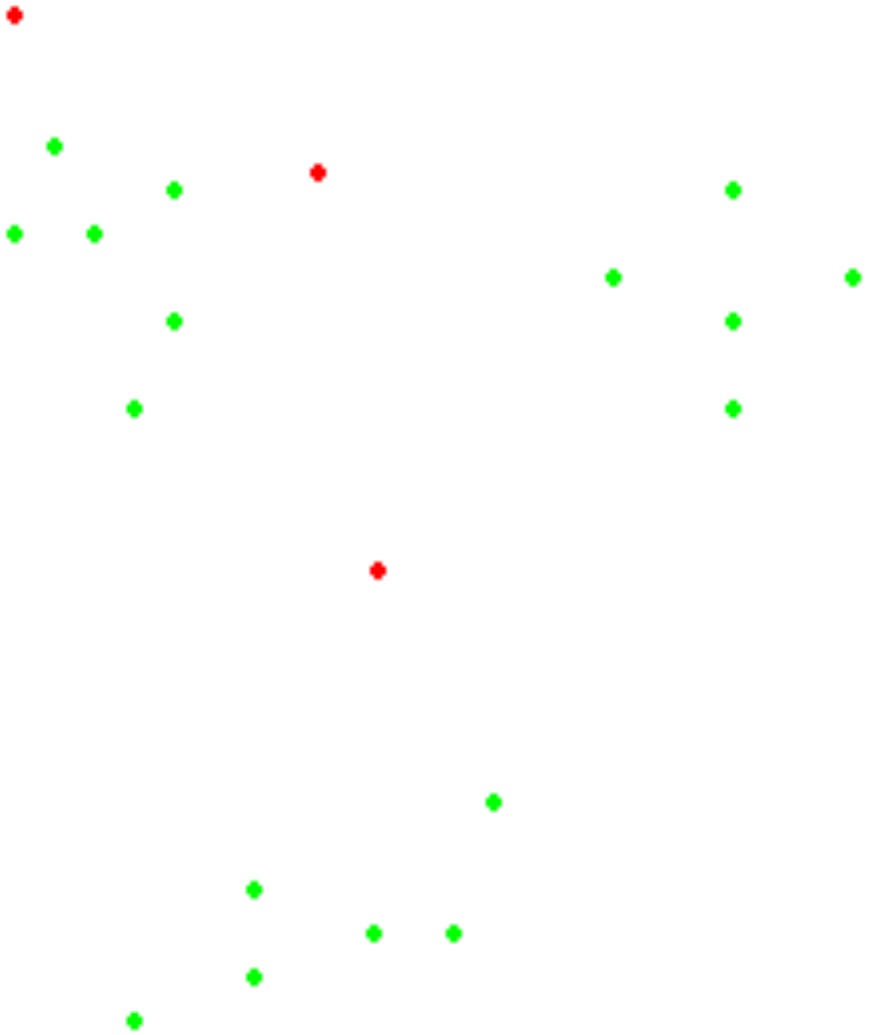
   (New centroid will be the mean of the data points which belongs to that cluster)

   Until the convergence criteria is met.

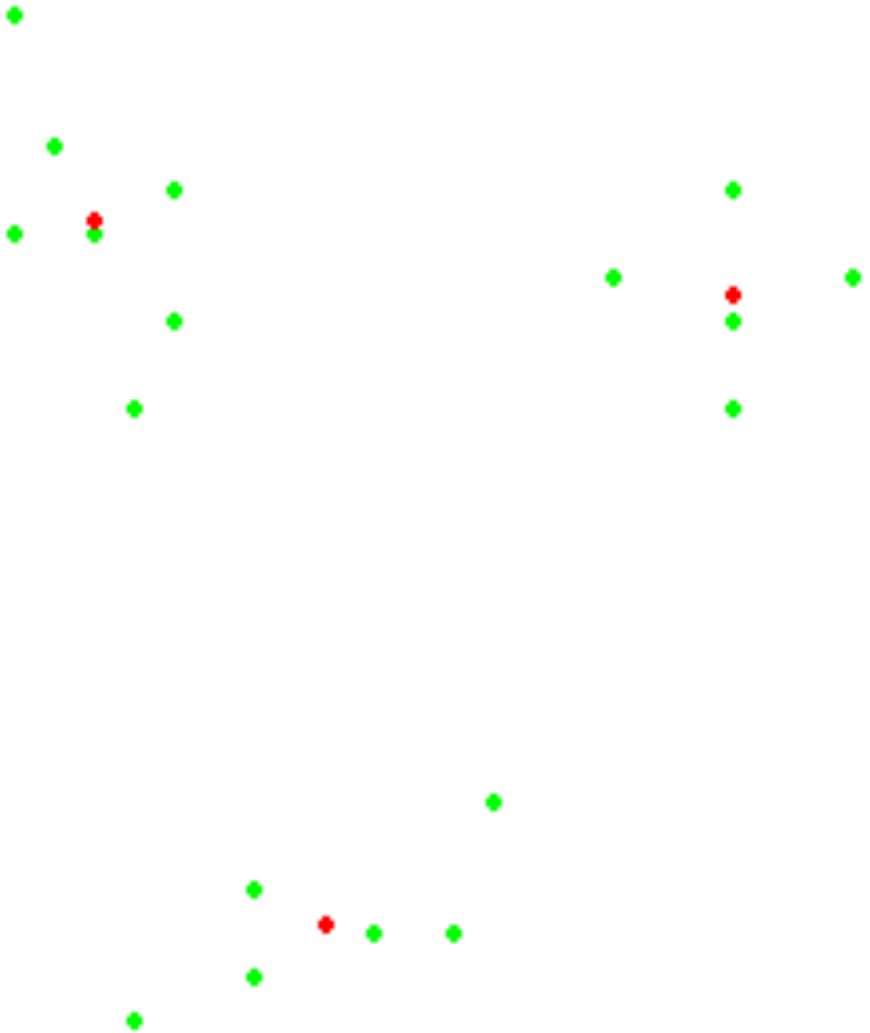
The data points

Initialization

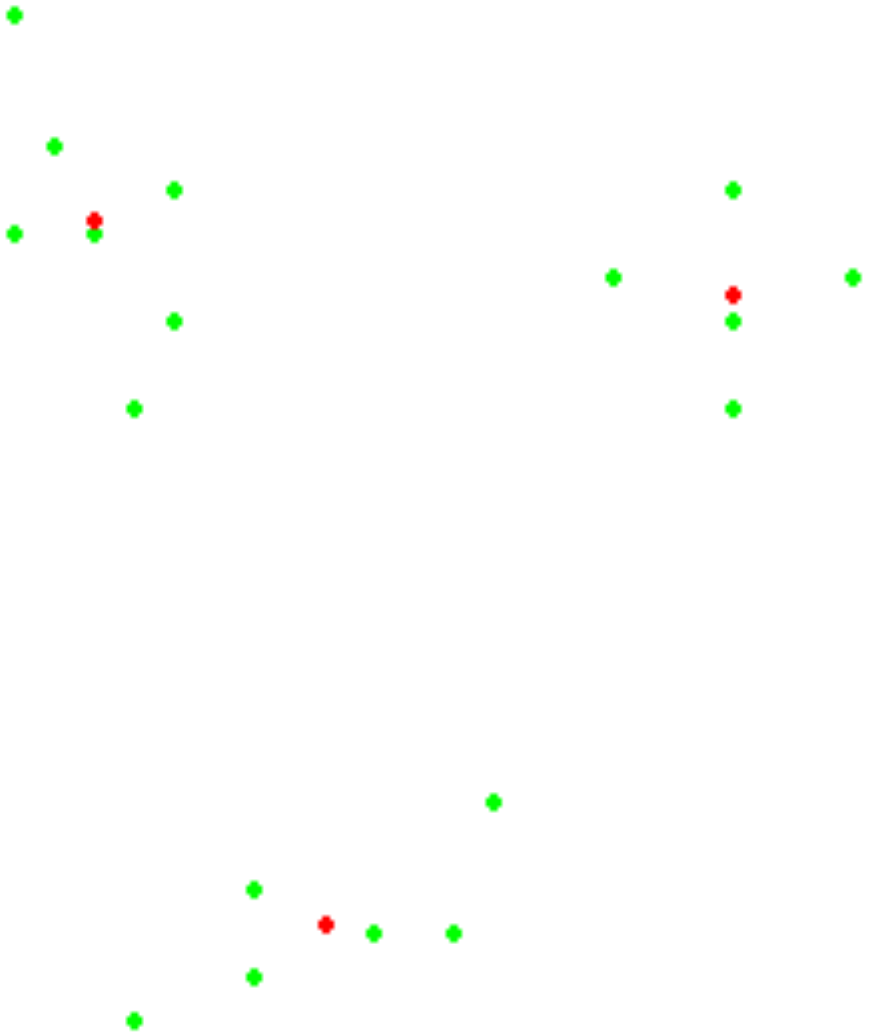
#Runs = 1



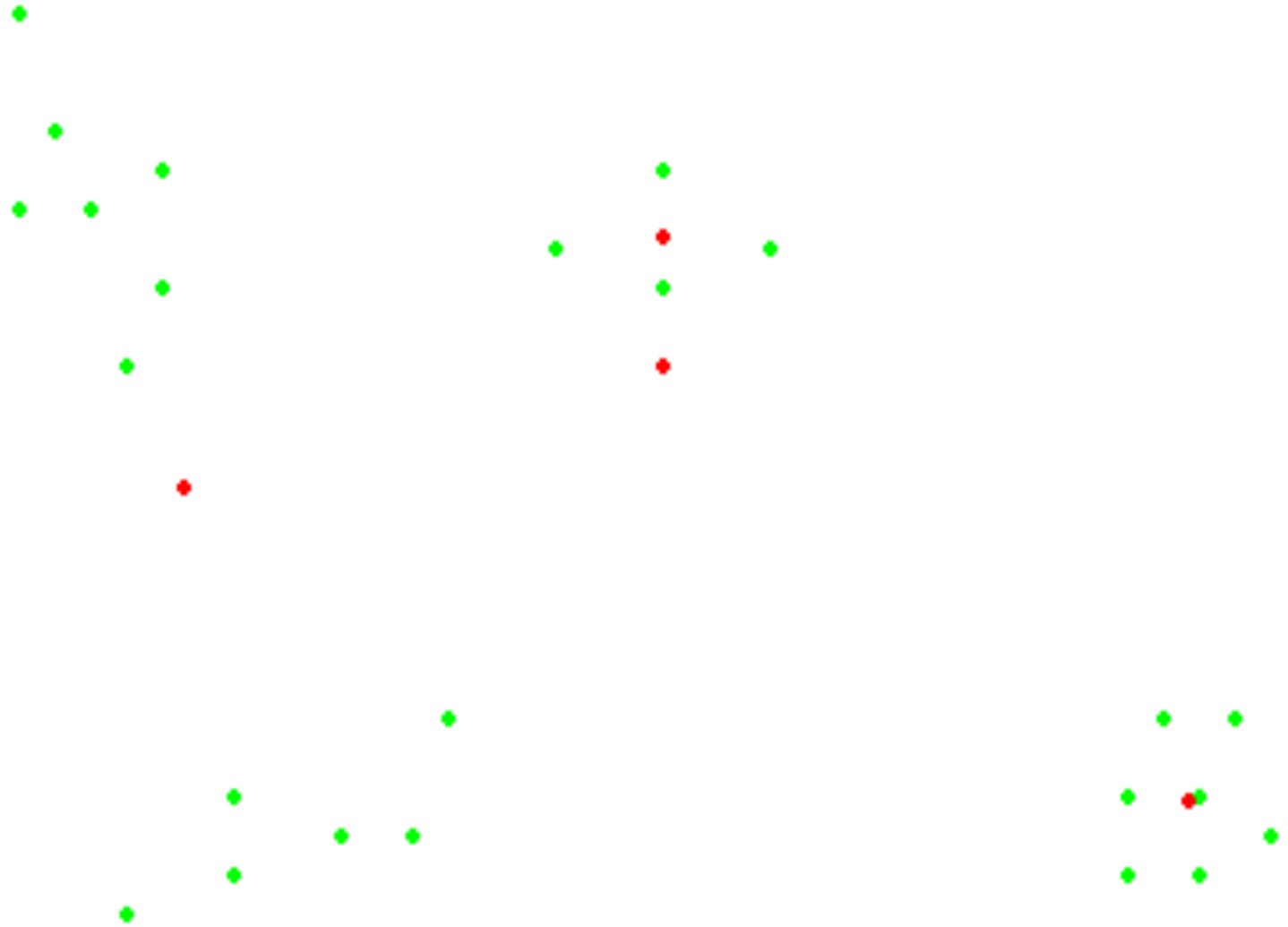
#Runs = 2



#Runs = 3



#Runs = 4



Parallel K-means using MapReduce

Algorithm 1. map $(key, value)$

Input: Global variable *centers*, the offset *key*, the sample *value*
Output: <*key*', *value*'> pair, where the *key*' is the index of the closest center point and *value*' is a string comprise of sample information

```
1. Construct the sample instance from value;
2. minDis = Double.MAX_VALUE;
3. index = -1;
4. For i=0 to centers.length do
        dis= ComputeDist(instance, centers[i]);
        If dis < minDis {
            minDis = dis;
            index = i;
        }
5. End For
6. Take index as key';
7. Construct value' as a string comprise of the values of different dimensions;
8. output < key', value' > pair;
9. End
```


Department of CSE/IT
Jaypee Institute of Information Technology, Noida

Combine Phase

Algorithm 2. combine (key, V)

Input: key is the index of the cluster, V is the list of the samples assigned to the same cluster
Output: $< key', value' >$ pair, where the key' is the index of the cluster, $value$' is a string comprised of sum of the samples in the same cluster and the sample number

1. Initialize one array to record the sum of value of each dimensions of the samples contained in the same cluster, i.e. the samples in the list V;
2. Initialize a counter num as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext()){
   Construct the sample $instance$ from V.next();
   Add the values of different dimensions of $instance$ to the array
   num++;
4. }
5. Take key as key';
6. Construct $value$' as a string comprised of the sum values of different dimensions and num;
7. output $< key', value' >$ pair;
8. End

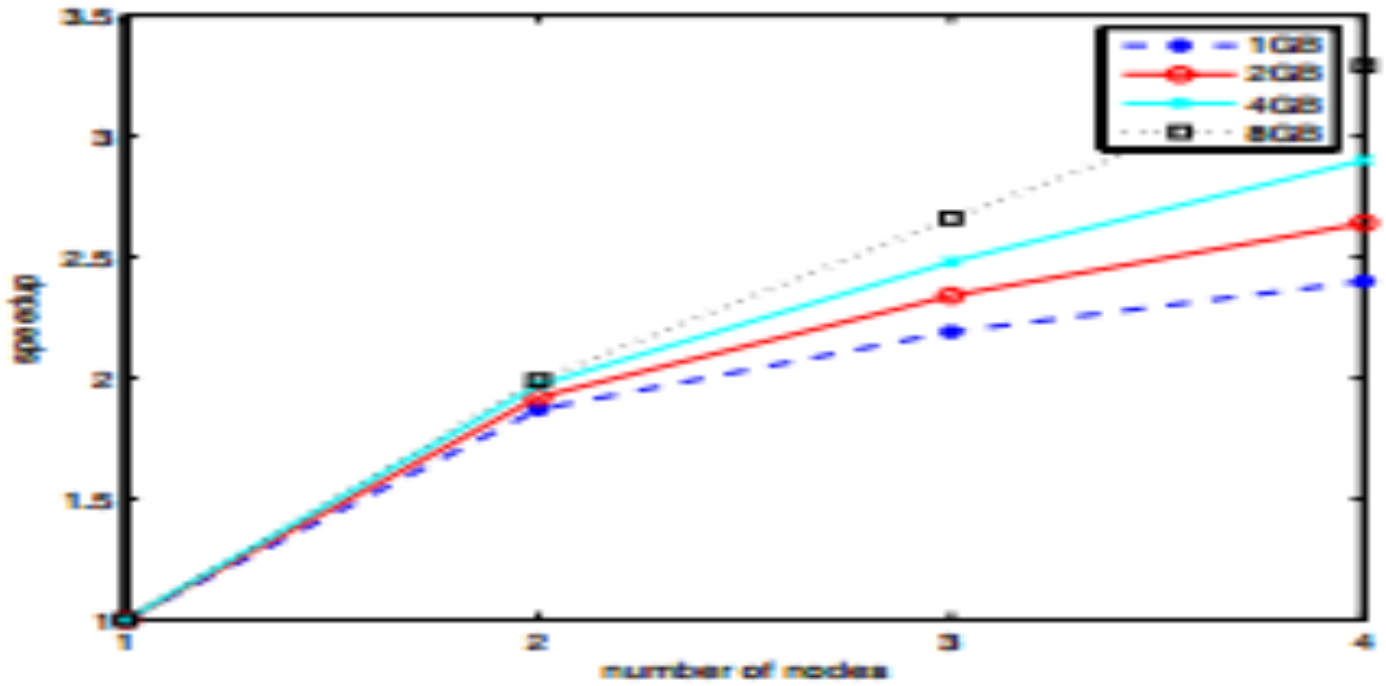
Reduce Phase

Algorithm 3. reduce $(key,\ V)$

Input: key is the index of the cluster, V is the list of the partial sums from different host
Output: $< key', value' >$ pair, where the key' is the index of the cluster, $value$' is a string representing the new center

1. Initialize one array record the sum of value of each dimensions of the samples contained in the same cluster, e.g. the samples in the list V;
2. Initialize a counter NUM as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext()){
   Construct the sample $instance$ from V.next();
   Add the values of different dimensions of $instance$ to the array
   NUM += num;
4. }
5. Divide the entries of the array by NUM to get the new center's coordinates;
6. Take key as key';
7. Construct $value$' as a string comprise of the $center$'s coordinates;
8. output $< key', value' >$ pair;
9. End

Speedup



(a) Speedup


Thank you !