

# Web Mining – II

## Parallelizing K-means Clustering with MapReduce

CSE 590 DATA MINING  
Prof. Anita Wasilewska  
SUNY Stony Brook

Presented By:  
Tushar Deshpande (SB ID: 106676653)  
Tejas Vora (SB ID: 106879612)

# References

- **Christophe Bisciglia, Aaron Kimball and Sierra Michels-Slettvet, MapReduce Theory and Implementation, Presentation from Google Code University 2007**
  - <http://code.google.com/edu/submissions/mapreduce-minilecture/lec2-mapred.ppt>
- **Christophe Bisciglia, Aaron Kimball and Sierra Michels-Slettvet, Clustering Algorithms, Presentation from Google Code University 2007**
  - <http://code.google.com/edu/submissions/mapreduce-minilecture/lec4-clustering.ppt>
- **Gordon S. Linoff, MapReduce and K-Means Clustering, Blog Entry on Data Miners Blog, 2008**
  - <http://www.data-miners.com/blog/2008/02/mapreduce-and-k-means-clustering.html>
- **Tapsie Giridher, Bhuvan Mital, Cluster Analysis, CSE 590 Class Presentation, 2009**
  - <http://www.cs.sunysb.edu/~cse634/spring2009/cluster1.pdf>
- **Kardi Teknomo, K-Mean Clustering Tutorials, An Online Tutorial, 2004**
  - <http://people.revoledu.com/kardi/tutorial/kMean/index.html>
- **Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, OSDI-2004**
  - <http://labs.google.com/papers/mapreduce-osdi04.pdf>
-

# Content

- What is clustering ?
- K-means Clustering Algorithm
- Need to speed up K-means clustering
- MapReduce
- Parallel algorithm for K-means clustering
- Parallel K-means clustering using MapReduce

# What Is Clustering ?

- Clustering is an important part of Data Mining.
- A cluster is a collection of objects which are similar to one another within same cluster and dissimilar to the objects in another cluster.
- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- Let's see some applications of clustering.

# Google News

## [iPhone activation headaches still trouble users](#)

Computerworld - 1 hour ago

July 02, 2007 (Computerworld) -- It took Iain Gillott 47 hours to activate his iPhone after waiting in the Texas heat Friday afternoon to buy one.

Most iPhone users thrilled but a few are iRate Reuters

Apple iPhone Arrives in the US Techtree.com

Forbes - ZDNet - Ars Technica - Wired News

[all 562 news articles »](#)



Local6.com

- They didn't pick all 3,400,217 related articles by hand...
- Or Amazon.com
- Or Netflix...

## [HCL Leaptop Z39](#)

Ub News - 2 hours ago

HCL Infosystem has launched new notebook. HCL Leaptop Z39 has a core 2 duo processor, a 8GB RAM and 500 GB HDD. It has a DVD Dual layer super multi drive, 2MP video camera and a 14.1 in screen.

[HCL Leaptop Z39 looks like a super laptop](#) Khabrein.info

[HCL Infosystems launches Leaptop Z39](#) News Line 365

[TechGadgets.in](#) - [What is the Word](#) - [enterpriser.in](#) - [CRN](#)

[all 17 news articles »](#)



Techtree.com

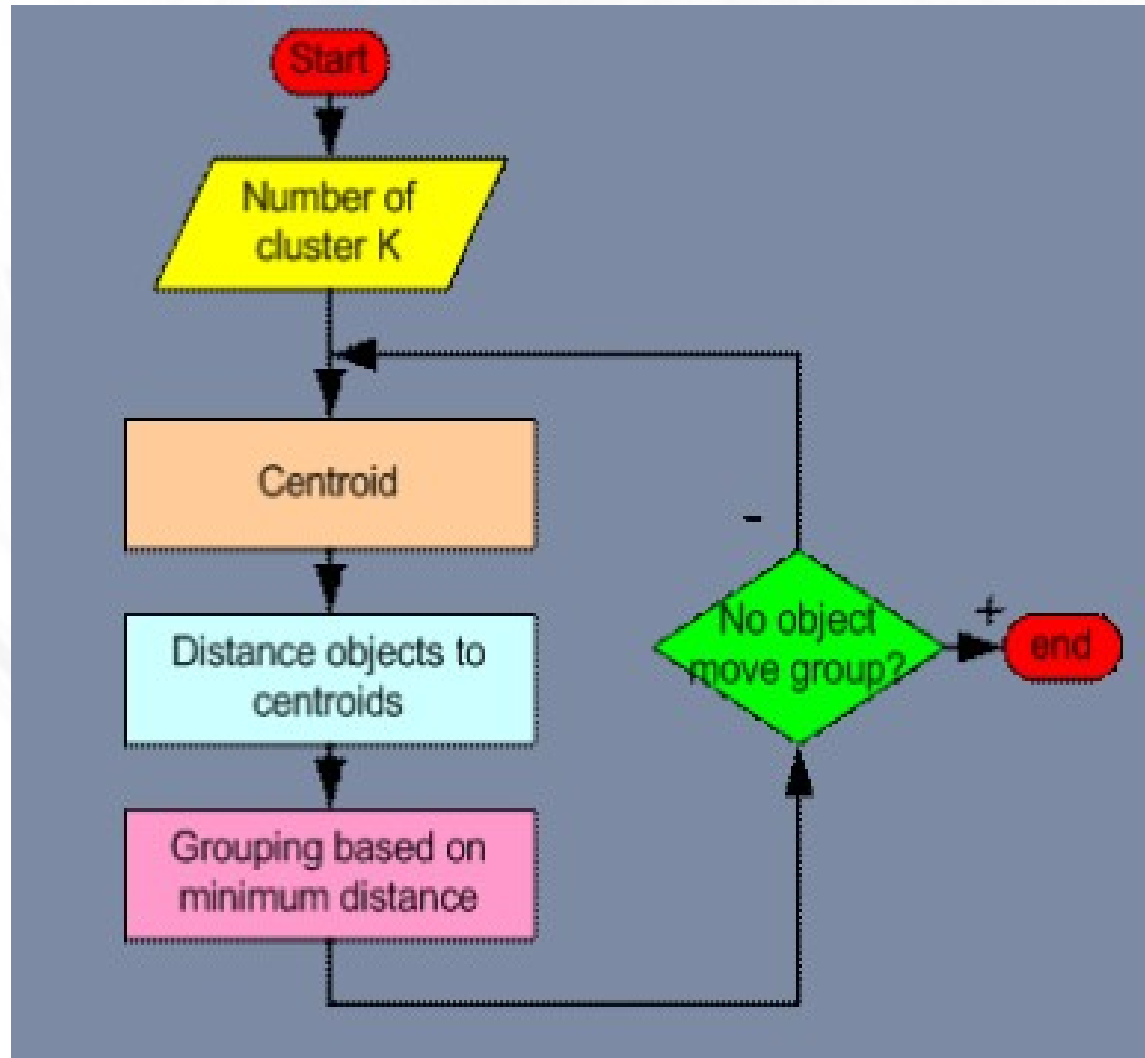
# Other Applications of Clustering

- Hospital Records
- Scientific Imaging
  - Related genes, related stars, related sequences
- Market Research
  - Segmenting markets, product positioning
- Social Network Analysis
- Data mining
- Image segmentation...

# K-Means Clustering

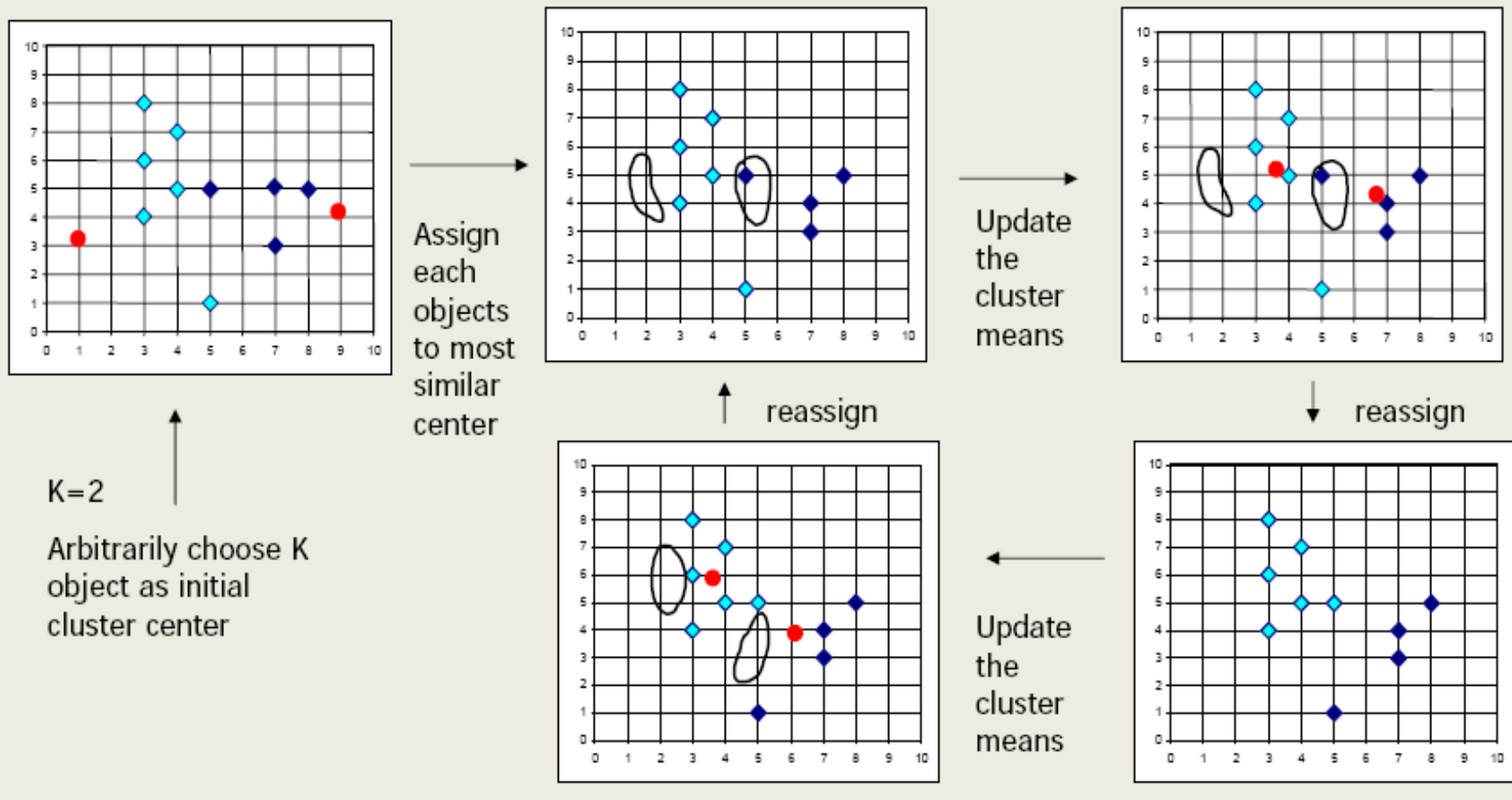
- K-means clustering algorithm (A quick recap...)
  1. Start with  $k$  cluster centers (chosen randomly or according to some specific procedure).
  2. Assign each record in the data to its nearest cluster center.
  3. Re-calculate the cluster centers as the "average" of the records in step 2
  4. Repeat, until the cluster centers no longer change.

# K-Means Clustering (Contd..)



# K-means Clustering Example

Example (<http://www-faculty.cs.uiuc.edu/~hanj/bk2/07.ppt> (page 31))



# Speeding Up K-means Clustering

- Complexity of k-means clustering algorithm =

$k * n * O(\text{distance metric}) * \text{num}(\text{iterations})$  [Christophe Bisciglia, Aaron Kimball and Sierra Michels-Slettvet, Clustering Algorithms, Presentation from Google Code University 2007]

- We observe that k-means algorithm runs slower as the number of records (n) increase.
- The speed up can be achieved by parallelizing k-means clustering.
- Multiple processors can process different records in parallel, thus reducing the overall execution time.
- We can use MapReduce technique to parallelize the k-means clustering.

# MapReduce

# MapReduce

- What is MapReduce ?
  - A distributed programming model by Google.
  
- Motivation:
  - Large scale processing of data.
  - To process lots of data ( > 1 TB)
  - To parallelize across hundreds/thousands of CPUs

*Christophe Bisciglia, Aaron Kimball and Sierra Michels-Slettvet, MapReduce Theory and Implementation, Presentation from Google Code University 2007*

# Programming Model

- Borrows from functional programming
- Users implement interface of two functions:
  - `map (in_key, in_value) -> (out_key, intermediate_value) list`
  - `reduce (out_key, intermediate_value list) -> out_value list`

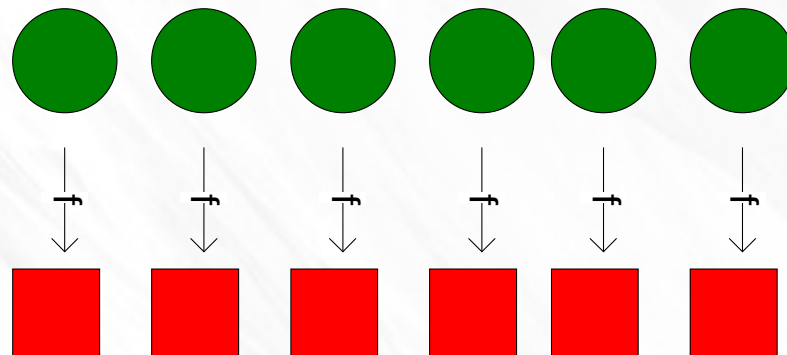
# map

- Records from the data source (lines out of files, rows of a database, etc) are fed into the map function as key\*value pairs: e.g., (filename, line).
- map() produces one or more *intermediate* values along with an output key from the input.

# Map

`map f lst: ('a->'b) -> ('a list) -> ('b list)`

Creates a new list by applying `f` to each element of the input list; returns output in order.



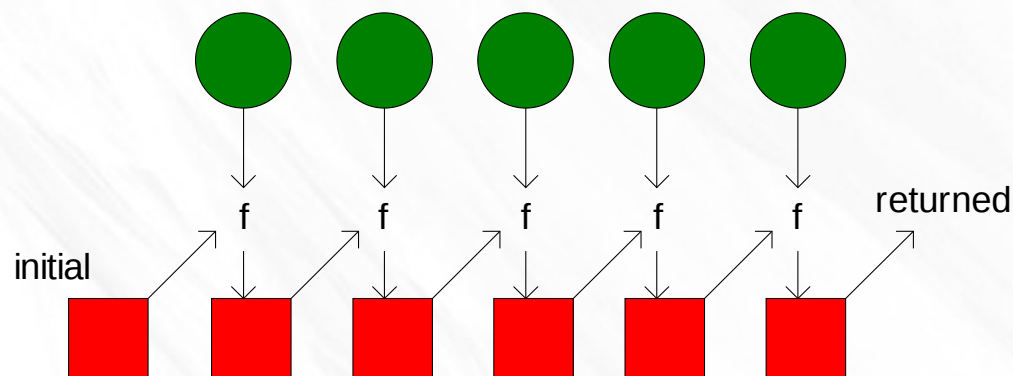
# reduce

- After the map phase is over, all the intermediate values for a given output key are combined together into a list
- `reduce()` combines those intermediate values into one or more *final values* for that same output key
- (in practice, usually only one final value per key)

# Reduce - In Functional Programming

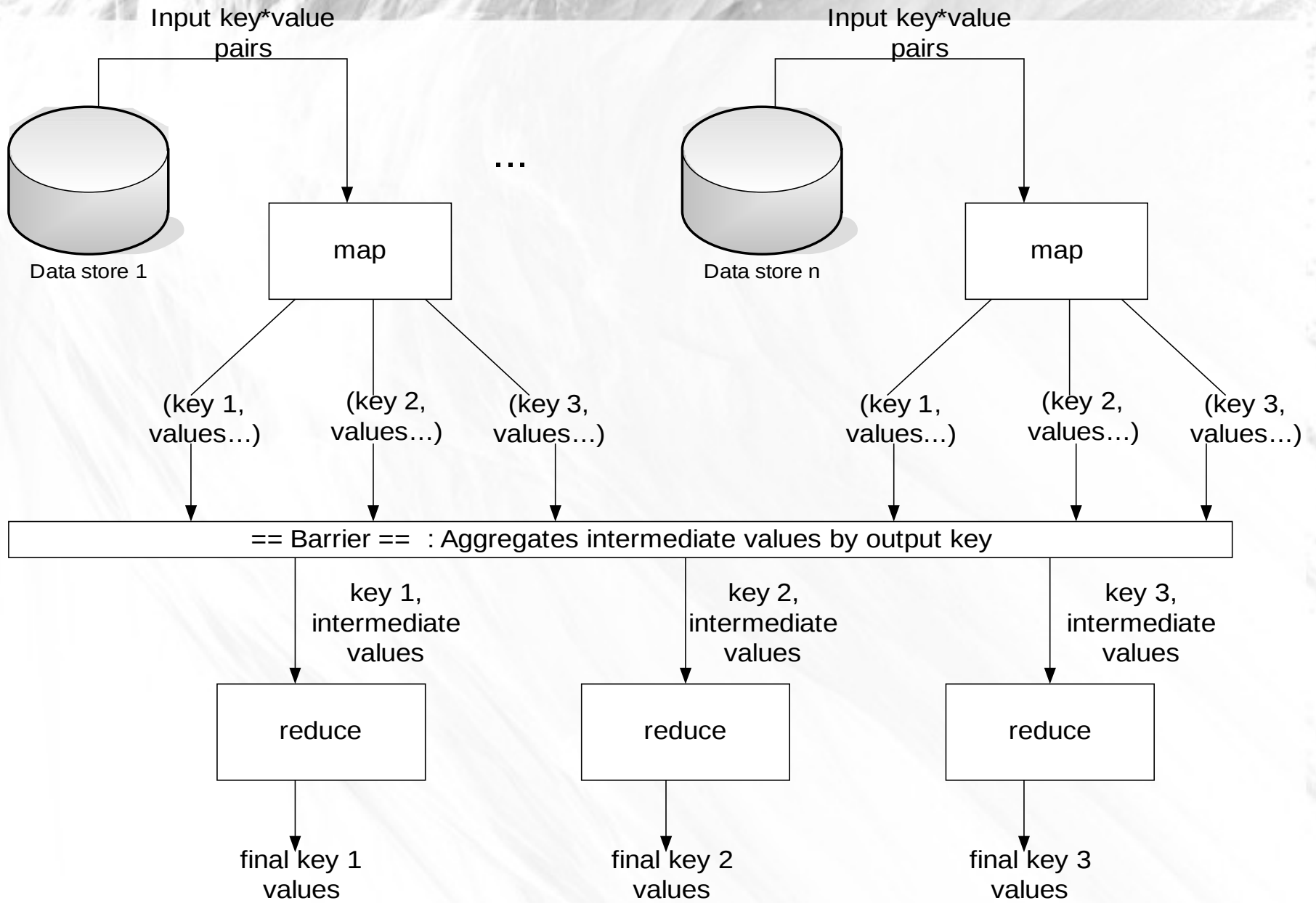
fold  $f$   $x_0$  lst: ('a\*'b->'b)->'b->('a list)->'b

Moves across a list, applying  $f$  to each element plus an *accumulator*.  $f$  returns the next accumulator value, which is combined with the next element of the list



# Parallelism

- map() functions run in parallel, creating different intermediate values from different input data sets
- reduce() functions also run in parallel, each working on a different output key
- All values are processed *independently*
- Bottleneck: reduce phase can't start until map phase is completely finished.



## Example: Count word occurrences

```
map(String input_key, String input_value):
```

```
  // input_key: document name
```

```
  // input_value: document contents
```

```
  for each word w in input_value:
```

```
    EmitIntermediate(w, "1");
```

```
reduce(String output_key, Iterator  
intermediate_values):
```

```
  // output_key: a word
```

```
  // output_values: a list of counts
```

```
  int result = 0;
```

```
  for each v in intermediate_values:
```

```
    result += ParseInt(v);
```

```
  Emit(AsString(result));
```

# Fault Tolerance

- Master detects worker failures
  - Re-executes completed & in-progress map() tasks
  - Re-executes in-progress reduce() tasks
- Master notices particular input key/values cause crashes in map(), and skips those values on re-execution.
  - Effect: Can work around bugs in third-party libraries!

# MapReduce Conclusions

- MapReduce has proven to be a useful abstraction  
MapReduce is used for the generation of data for Google's production web search service, for sorting, for data mining, for machine learning [Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI-2004]
- Greatly simplifies large-scale computations at Google
- Functional programming paradigm can be applied to large-scale applications
- MapRduce offers built-in fault-tolerance.

# Parallelizing k-means Clustering with MapReduce

# Parallel Version of K-means Clustering

- We partition records among multiple processors.
- Each processor can read the previous iteration's cluster centers and assign the records on the processor to clusters.
- Each processor then calculates new centers for its share of records.
- Each actual cluster center (for the records across all processors) is then the weighted average of the centers on each processor.

# Additional Shared Information

- We need to handle the cluster center information.
- We create a shared file that has the centroids as calculated for each processor. This file contains:
  - The iteration number.
  - The cluster id.
  - The cluster coordinates.
  - The number of records assigned to the cluster.
- This is the centroid file. An iteration through the algorithm is going to add another set of records to this file.
- This information is the only information that needs to be communicated globally

# K-means Clustering Using map, combine and reduce

- Step 1:
- We create a file which contains the information as described in previous slide.
- This file contains the cluster centers for each iteration.
- This file is shared among all processors.

# K-means Clustering Using map, combine and reduce (Contd..)

- Step 2:
- The Map function reads this file to get the centers from the last finished iteration.
- It then reads the input records (the data) and calculates the distance (Euclidean Distance) to each center.
- For each record, it produces an output pair with:
  - key -- cluster id
  - value -- coordinates of records.

# K-means Clustering Using map, combine and reduce (Contd..)

- Step 3: Combine
- The 'map' step produces a lot of data.
- So we use a Combine function to reduce the size before sending it to Reduce.
- The Combine function calculates the average of the coordinates for each cluster id, along with the number of records.
- This is simple, and it produces one record of output for each cluster:
  - key is cluster
  - value is number of records and average values of the coordinates.
- The amount of data now is the number of clusters times the number of processors times the size of the information needed to define each cluster.
- This is small relative to the data size.

# K-means Clustering Using map, combine and reduce (Contd..)

- Step 4
- The Reduce function calculates the weighted average of its input.
- Its output is written to the cluster file, and contain:
  - the iteration number;
  - the cluster id;
  - the cluster center coordinates;
  - the size of the cluster.
- The iteration process can than continue.

# Example

Consider a sample data. [Data taken from Kardi Teknomo, K-Mean Clustering Tutorials, An Online Tutorial, 2004]

Object	Attribute 1 (x)	Attribute 2(y)
A	1	1
B	2	1
C	4	3
D	5	4

- Let  $k=2 \Rightarrow$  We've 2 groups cluster 0 and cluster 1
- Let (1,1) be centroid for cluster 0 and (4,3) be centroid for cluster 2
- Assume that we've 2 processors.
- Each processor processes 2 records.

# Example (Contd..)

- Processor 1 processes records A and C.
- Processor 2 processes records B and D.
- Initially, the cluster file looks like.

Iteration No	0	
Cluster Id	Cluster Coordinates	No. of Records Assigned
0	(1,1)	0
1	(2,1)	0

# Example (Contd..)

- Iteration 1: Output of map stage
- Processor 1's Output

Cluster Id	Coordinates of record
0	A(1,1)
1	C(4,3)

- Processor 2's Output

Cluster Id	Coordinates of record
1	B(2,1)
1	D(5,4)

# Example (Contd..)

- Iteration 1: Output of combine stage
- Processor 1's Output

Cluster Id	No. of records	Average Value of Coordinates
0	1	(1,1)
1	1	(4,3)

- Processor 2's Output

Cluster Id	No. of records	Average Value of Coordinates
1	2	(3.5,2.5)

# Example (Contd..)

## Computing Weighted average

- From the data from previous slide we know that there are following 2 entries related to cluster 1

Processor Number	Cluster Id	Number of Records (Weight)	Average Value of Coordinates
1	1	1	(4,3)
2	1	2	(3.5,2.67)

$$\text{Weighted avg of cluster centers for cluster 1} = \left( \frac{1*4 + 2*3.5}{1+2}, \frac{1*3 + 2*2.5}{1+2} \right)$$

$$\text{Weighted avg of cluster centers for cluster 1} = (3.67, 2.67)$$

# Example (Contd..)

- Iteration 1: Output of reduce stage

Iteration No	1	
Cluster Id	Cluster Coordinates (Weighted average of output of combine stage)	No. of records Assigned
0	(1,1)	1
1	(3.67,2.67)	3

# Example (Contd..)

- Iteration 2: Output of map stage
- Processor 1's Output

Cluster Id	Coordinates of records
0	A(1,1)
1	C(4,3)

- Processor 2's Output

Cluster Id	Coordinates of records
0	B(2,1)
1	D(5,4)

# Example (Contd..)

- Iteration 2: Output of combine stage
- Processor 1's Output

Cluster Id	No. of records	Average Value of Coordinates
0	1	(1,1)
1	1	(4,3)

- Processor 2's Output

Cluster Id	No. of records	Average Value of Coordinates
0	1	(2,1)
1	1	(5,4)

# Example (Contd..)

- Iteration 2: Output of reduce stage

Iteration No	2	
Cluster Id	Cluster Coordinates (Weighted average of output of combine stage)	No. of Records Assigned
0	(1.5,1)	2
1	(4.5,3.5)	2

# Example (Contd..)

- Iteration 3: Output of map stage
- Processor 1's Output

Cluster Id	Coordinates of records
0	A(1,1)
1	C(4,3)

- Processor 2's Output

Cluster Id	Coordinates of records
0	B(2,1)
1	D(5,4)

# Example (Contd..)

- Iteration 3: Output of combine stage
- Processor 1's Output

Cluster Id	No. of records	Average Value of Coordinates
0	1	(1,1)
1	1	(4,3)

- Processor 2's Output

Cluster Id	No. of records	Average Value of Coordinates
0	1	(2,1)
1	1	(5,4)

# Example (Contd..)

- Iteration 3: Output of reduce stage

Iteration No	3	
Cluster Id	Cluster Coordinates (Weighted average of output of combine stage)	No. of records Assigned
0	(1.5,1)	2
1	(4.5,3.5)	2

- Note that this output exactly matches output of iteration 2.
- So, we stop k-means algorithms. Records are assigned to following clusters.

Object	Attribute 1 (x)	Attribute 2(y)	Cluster Id
A	1	1	0
B	2	1	0
C	4	3	1
D	5	4	1

# Conclusion

- We can use MapReduce to successfully parallelize the K-means clustering algorithm.
- Each processor processes a group of records in parallel.
- This significantly improves the performance.

**Thank You !**