Big Data Algorithm Design and Application

SIGBPS Workshop on Business Processes and Services
Tutorial, Auckland, 2014

Kunpeng Zhang
Assistant Professor
Information and Decision Sciences
University of Illinois at Chicago
December, 2014

More about me…

- Past big data analytics teaching experience
  - Several semester-long big data courses
    - http://kzhang6.people.uic.edu/teaching/ids594_f14/
  - Tutorials on big data algorithm and application
    - http://kzhang6.people.uic.edu/tutorial/amcis2014.html
- Past big data analytics research experience
    - http://kzhang6.people.uic.edu/research.html

What we'll cover…

- Big data overview
- Hadoop overview
  - Framework
  - MapReduce programming
- Applications
  - Scalable clustering algorithm: K-Means
  - Iterative graph algorithm: single source shortest path
  - Scalable recommender system: Item-based collaborative filtering

Focus on MapReduce-based algorithm design

We can not cover all...

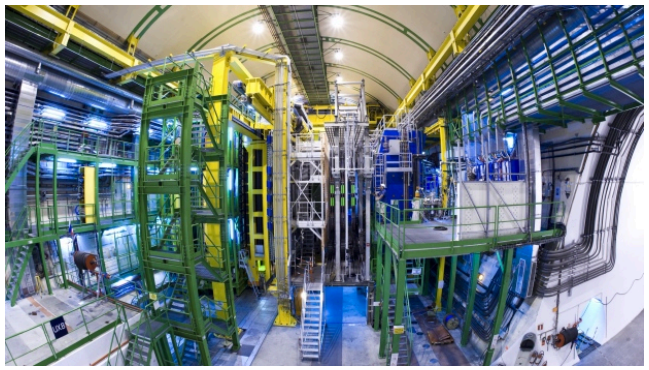
etc


Big data

Big data is everywhere...

processed about 24 petabytes of data per day in 2009.

transfers about 30 petabytes of data through its networks each day.



By 2012, LHC collision data was being produced at approximately 25 petabytes per year.

As of January 2013, Facebook users had uploaded over 240 billion photos, with 350 million new photos every day.

Twitter now sends and receives as many as 200 million "tweets" every day.

S3: 449B objects, peak 290k request/second (7/2011)
1T objects (6/2012)

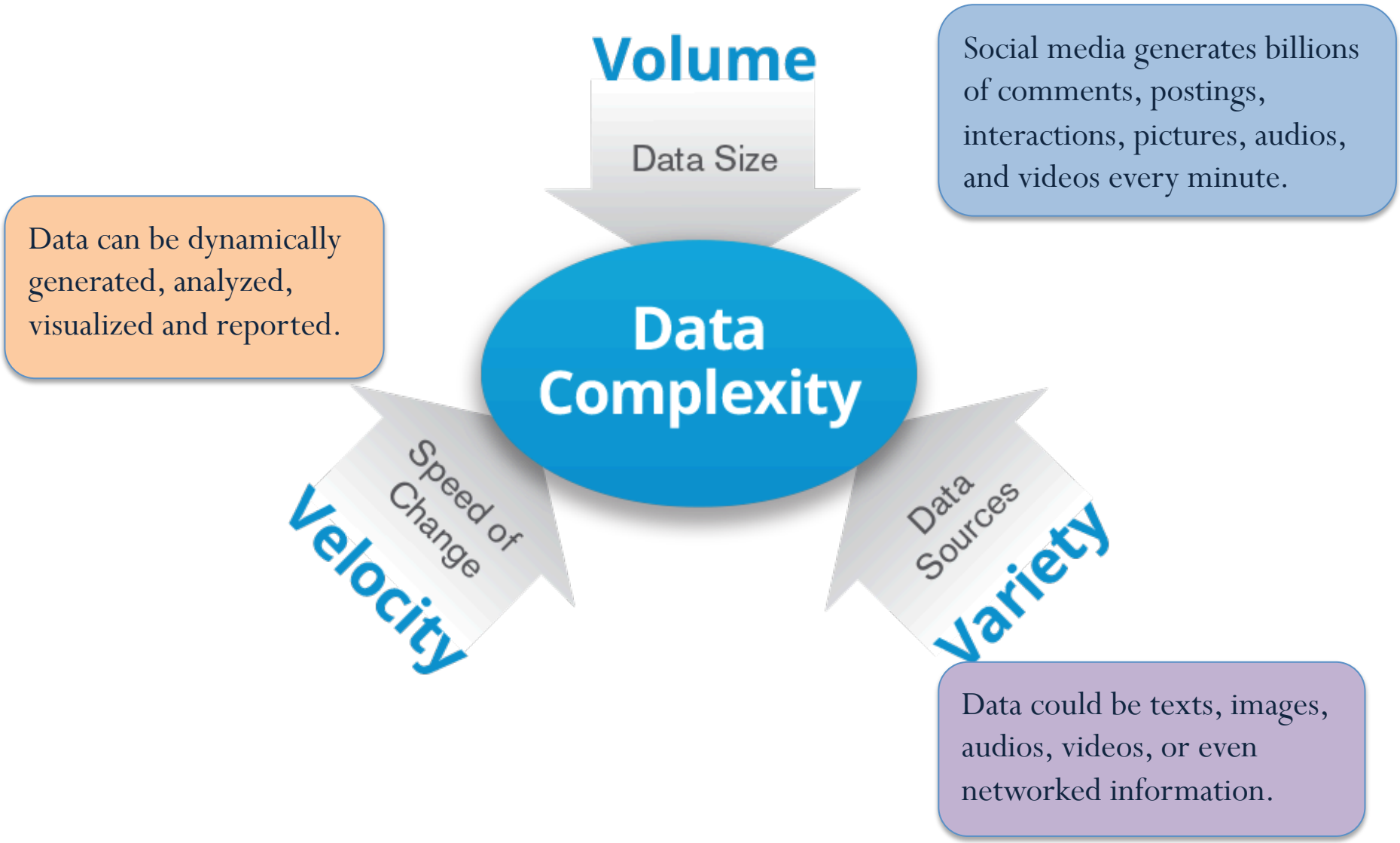
150 PB on 50k+ servers running 15k apps (6/2011

Why big data?

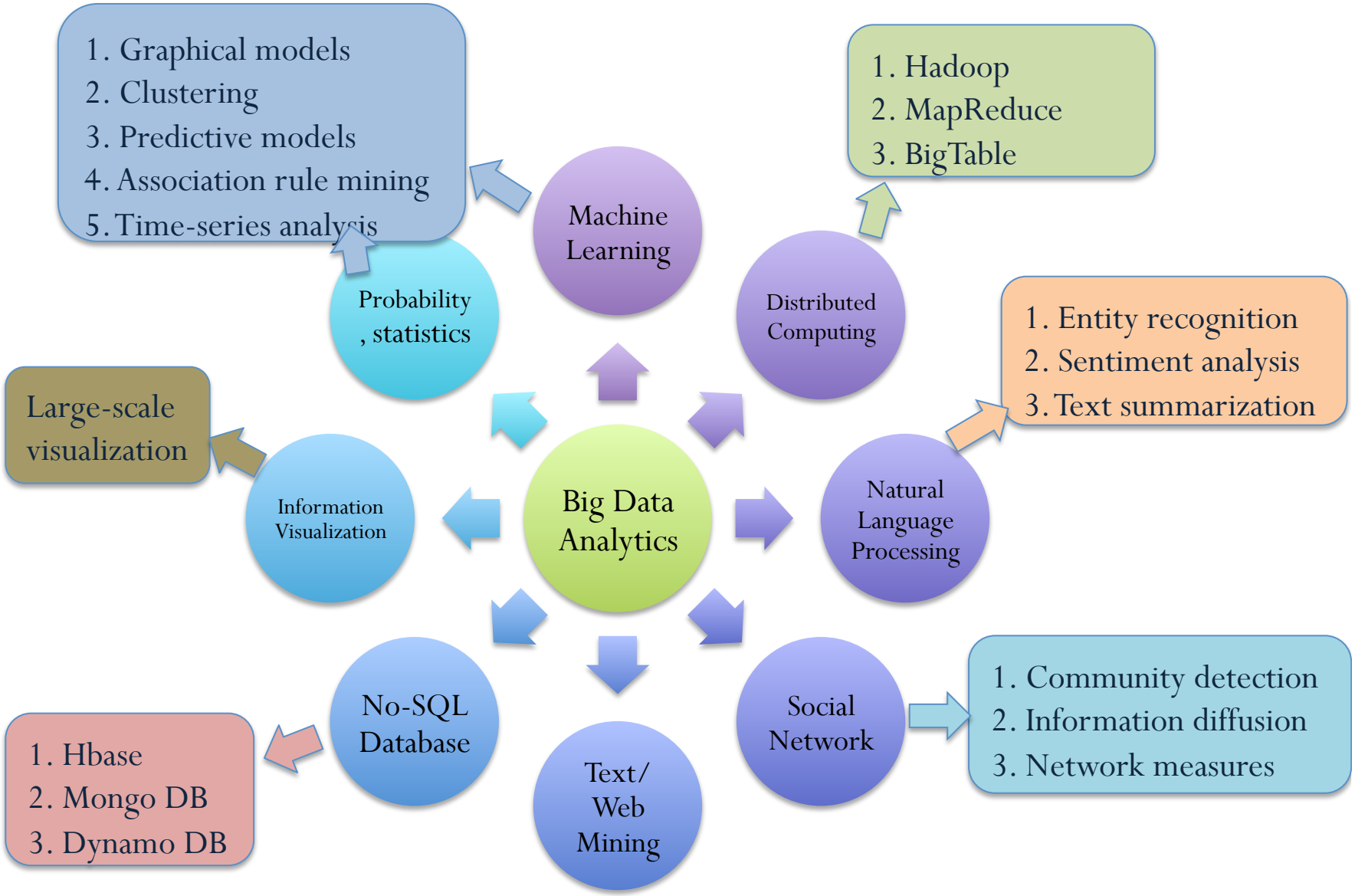
What is big data?

- Big data is a blanket term for any types of data sets so large and complex that it becomes difficult to process using on-hand data management tools or traditional data processing applications. [from Wikipedia]

Big data characteristics (3 Vs)



Techniques in big data analytics



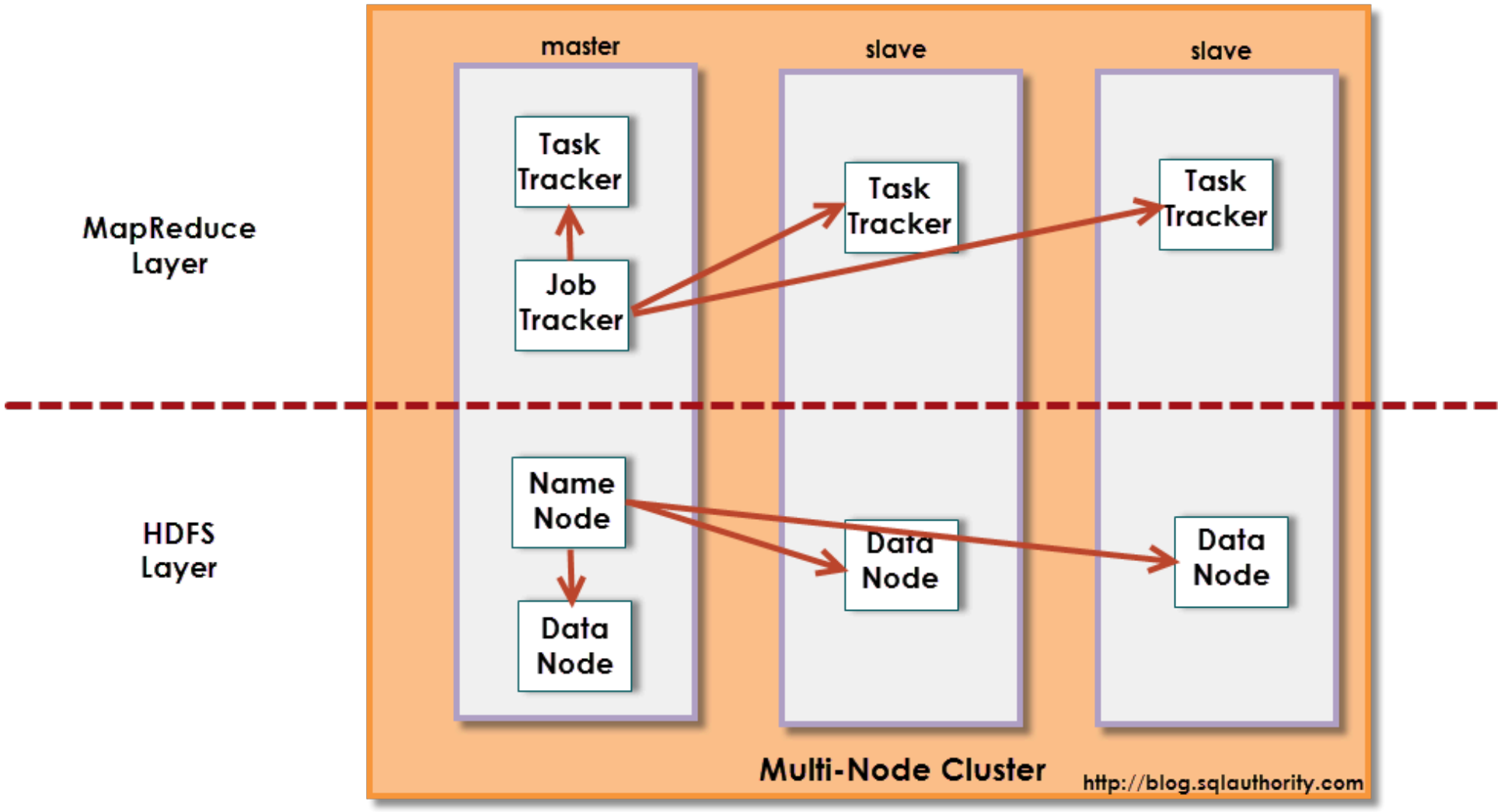
- Hadoop is a software framework for distributed processing of large datasets across large clusters of computers.
- It is based on a simple programming model called MapReduce.
- It has two main layers:
  - Distributed file system (HDFS)
  - Execution engine (MapReduce)

Usually, big data problems

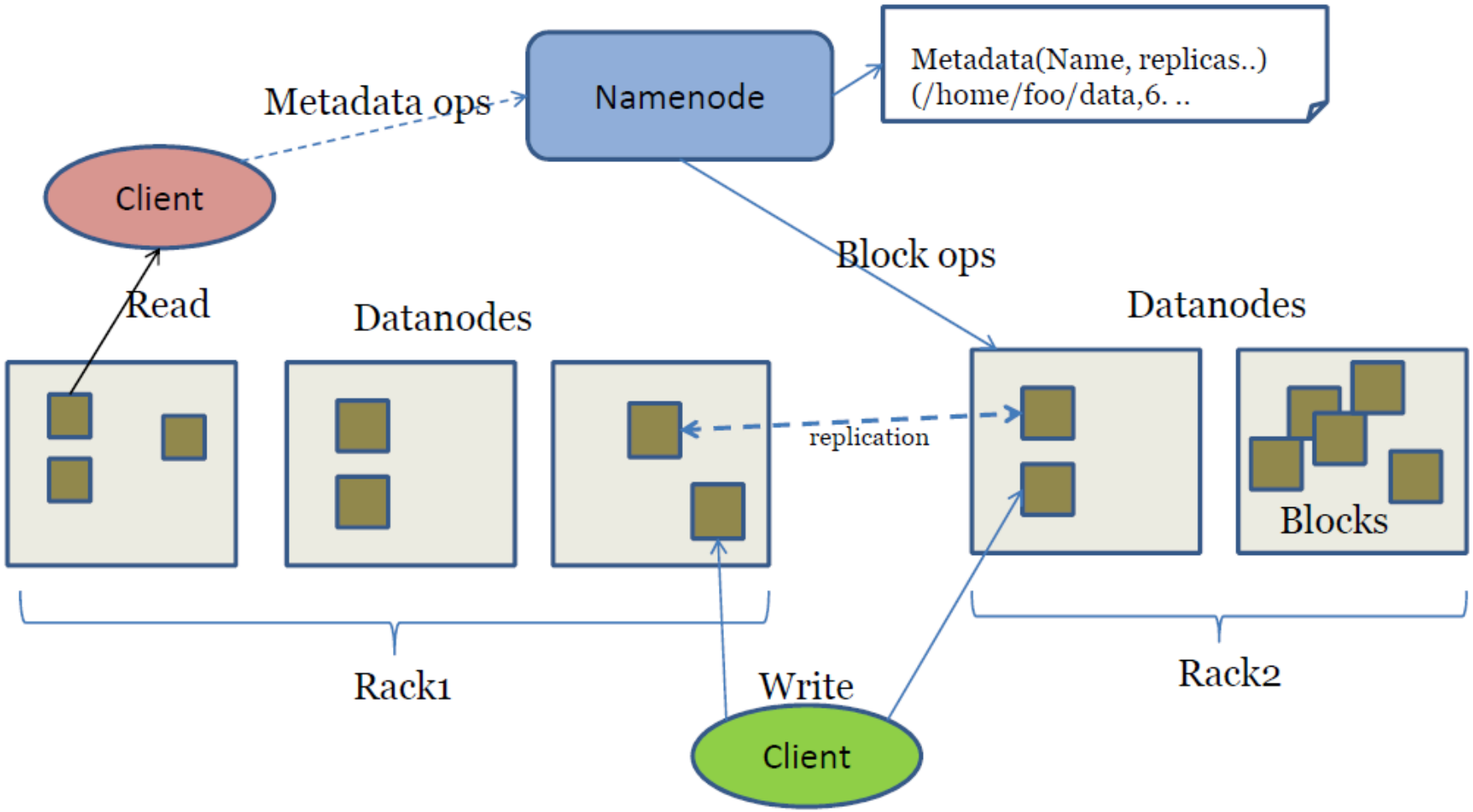
- Iterate over a large number of records
- Extract something of interest from each → **Map**
- Shuffle and sort intermediate results
- Aggregate intermediate results → **Reduce**
- Generate final output

Key idea: provide a functional abstraction for these two operations

(Dean and Ghemawat, OSDI 2004)

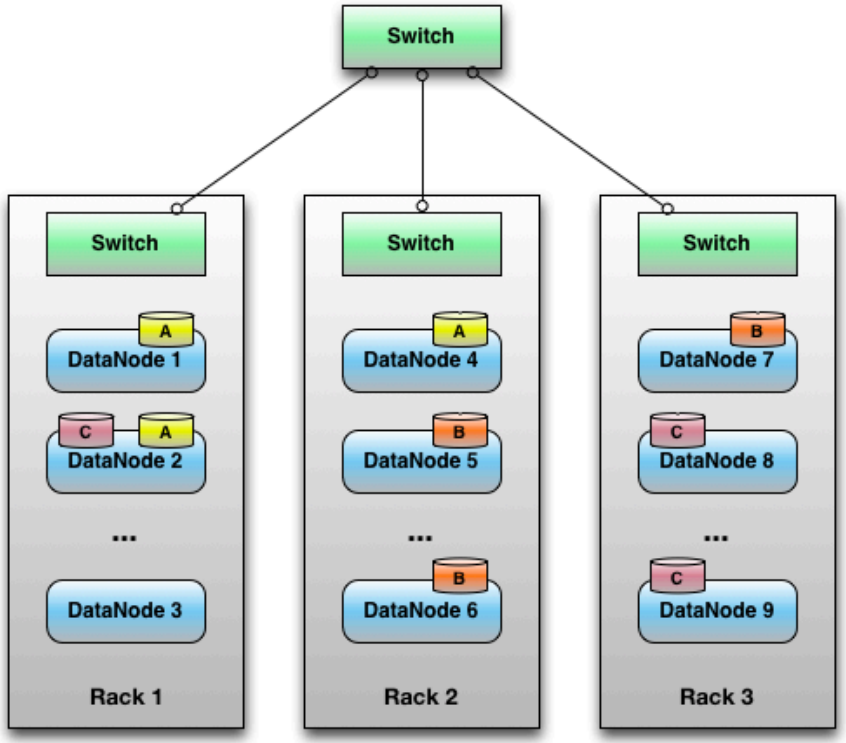

master
slave
slave
Task Tracker
Task Tracker
Task Tracker
Job Tracker
MapReduce Layer
HDFS Layer
Name Node
Data Node
Data Node
Data Node
Multi-Node Cluster
http://blog.sqlauthority.com

HDFS Architecture



Data replication

- Frist copy is written to the local node (write affinity).
- Second copy is written to a DataNode within the same rack.
- Third copy is written to a DataNode in a different rack.



Objectives: **load balancing, fast access, fault tolerance.**

MapReduce

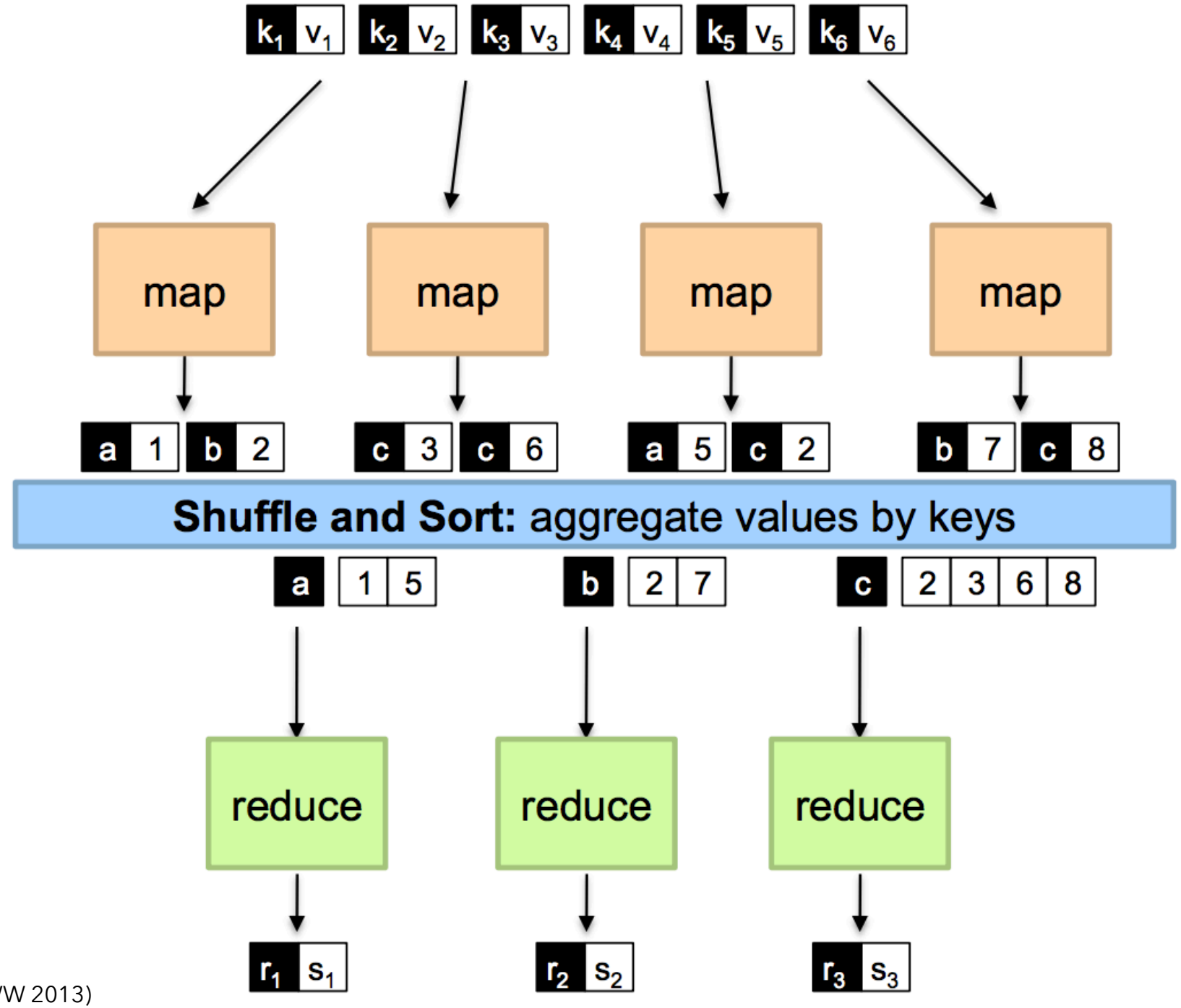
- Everything in MapReduce is <key, value> pair

- Programmers create two functions/methods:

  map(key1, value1) → [<key2, value2>]

  reduce(key2, [value2]) → [<key3, value3>]

- The execution framework in Hadoop handles everything else…


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

(Lin, WWW 2013)

MapReduce

- Programmers create two functions/methods:

  map(key1, value1) → [<key2, value2>]

  reduce(key2, [value2]) → [<key3, value3>]

- The execution framework in Hadoop handles everything else…

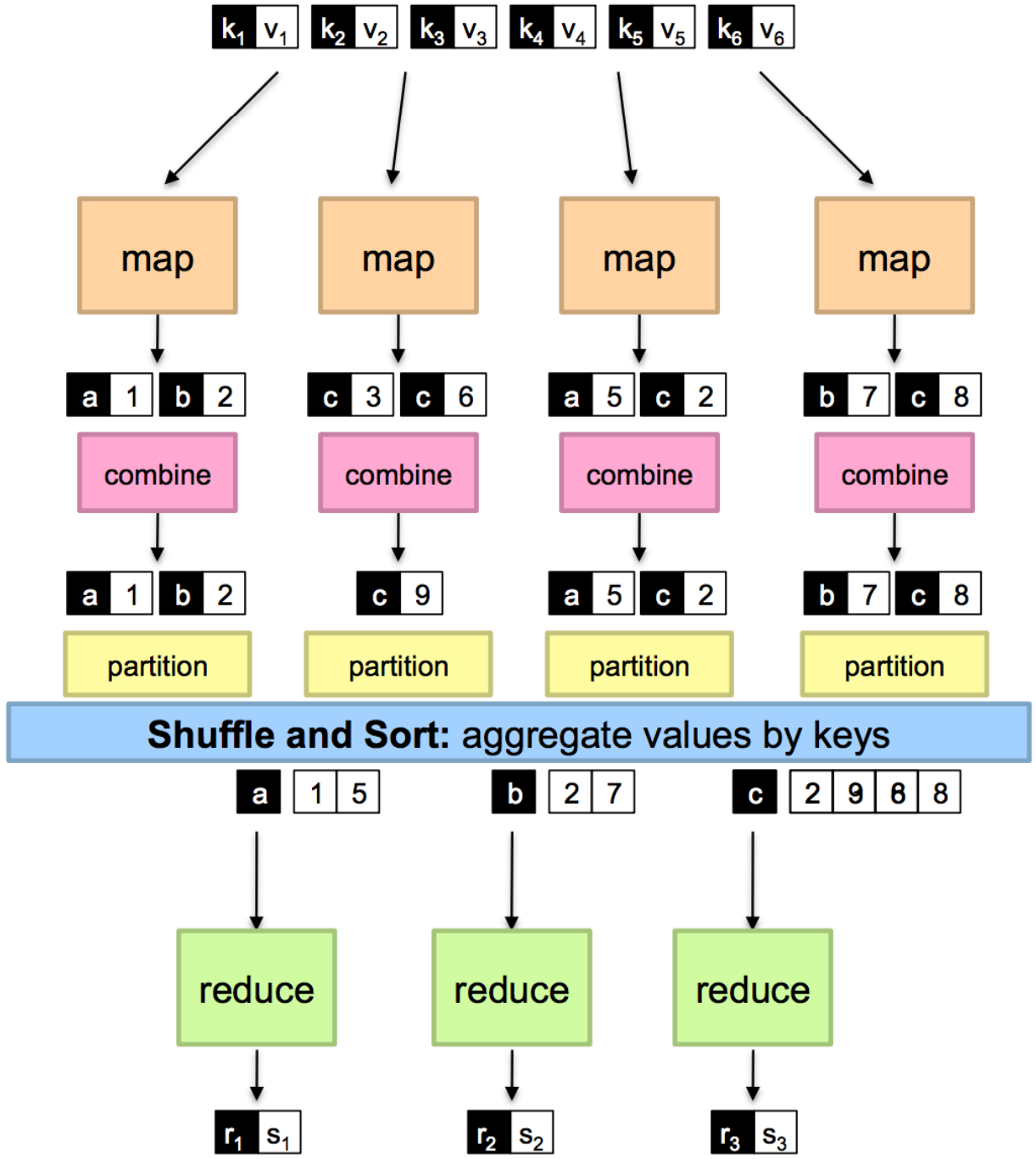
What's "everything else"?

MapReduce “Runtime”

- Handles scheduling
  - Assigns mappers and reducers to do tasks
- Handles data distribution
  - Moves processes to data
- Handles synchronization
  - Collects, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects failures and restarts
- Everything happens on top of a distributed file system

MapReduce

- Programmers create two functions/methods:
  map(key1, value1) → [<key2, value2>]
  reduce(key2, [value2]) → [<key3, value3>]
- The execution framework in Hadoop handles everything else…
- Usually, programmers also customize:
  partition (k′, number of partitions) → partition for k′
    Often a simple hash of the key, e.g., hash(k′) mod n
    Divides up key space for parallel reduce operations
  combine (k′, v′) → <k′, v′>*
    Mini-reducers that run in memory after the map phase
    Used as an optimization to reduce network traffic


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8 8
reduce
reduce
reduce
r1 s1
r2 s2
r3 s3

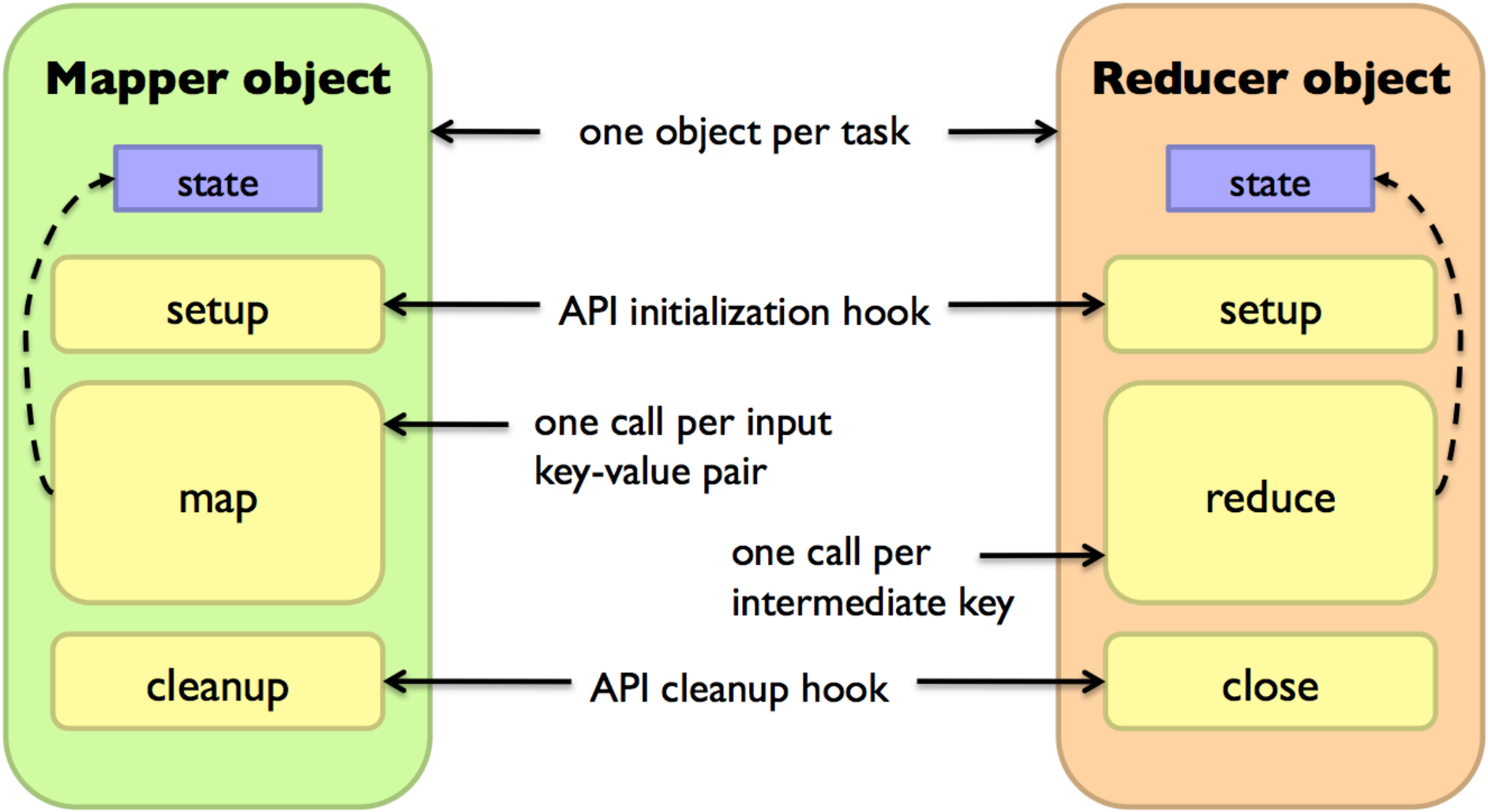
(Lin, WWW 2013)

More about MapReduce…

- The same key must be sent to the same reducer for reduce tasks
- Key arrived at each reducer in a sorted order
  - No enforced ordering across reducers
- The number of partitions is based on the number of reducers

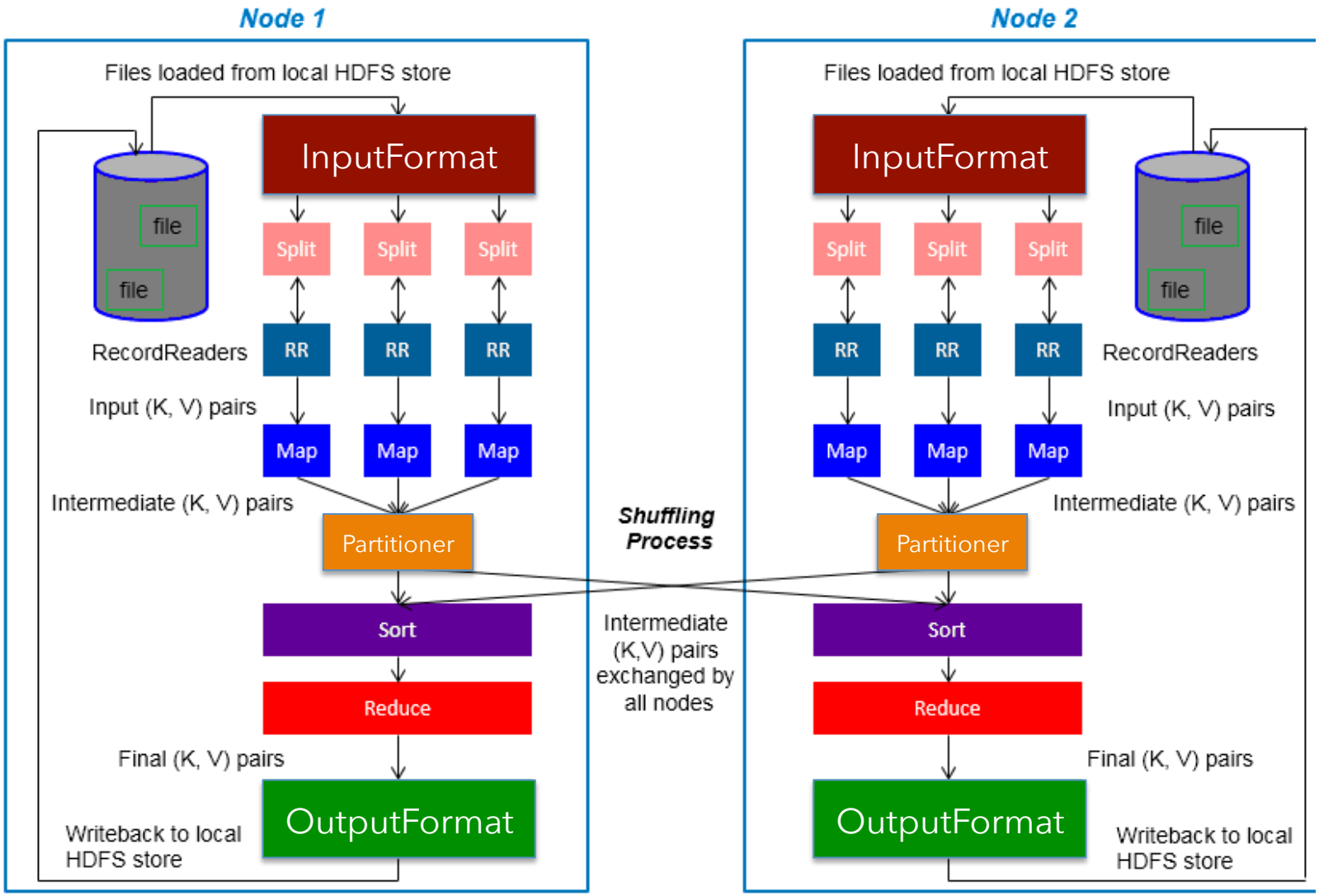
- Limitations
  - Limited control over data and execution flow
    - All algorithms must expressed in map, reduce, combine, or partition
  - You can not control:
    - Where mappers and reducers run
    - When a mapper or reducer begins or finishes
    - Which input a particular mapper is processing
    - Which intermediate key a particular reducer is processing

Mapper and Reducer objects

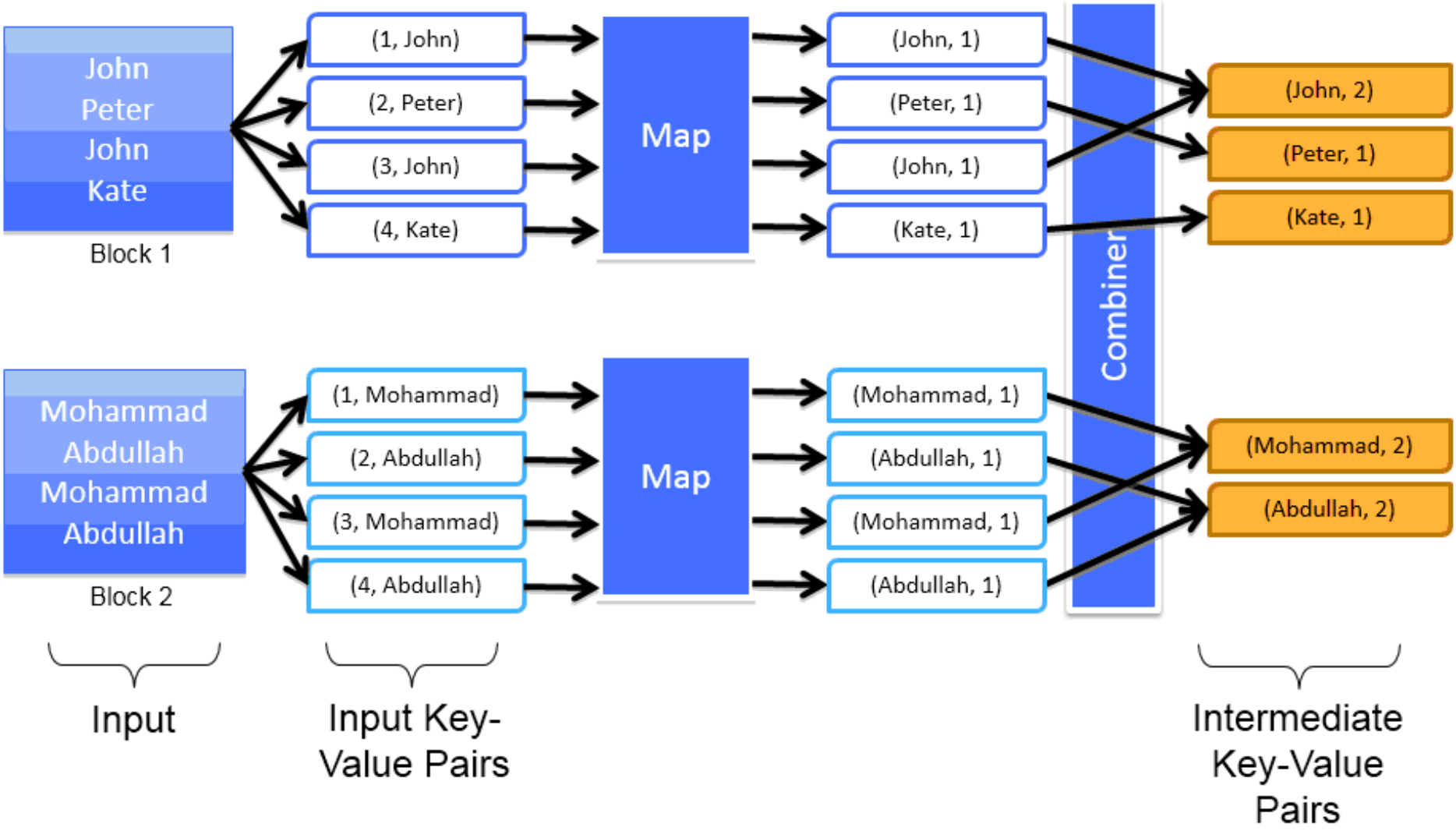


(Lin, WWW 2013)

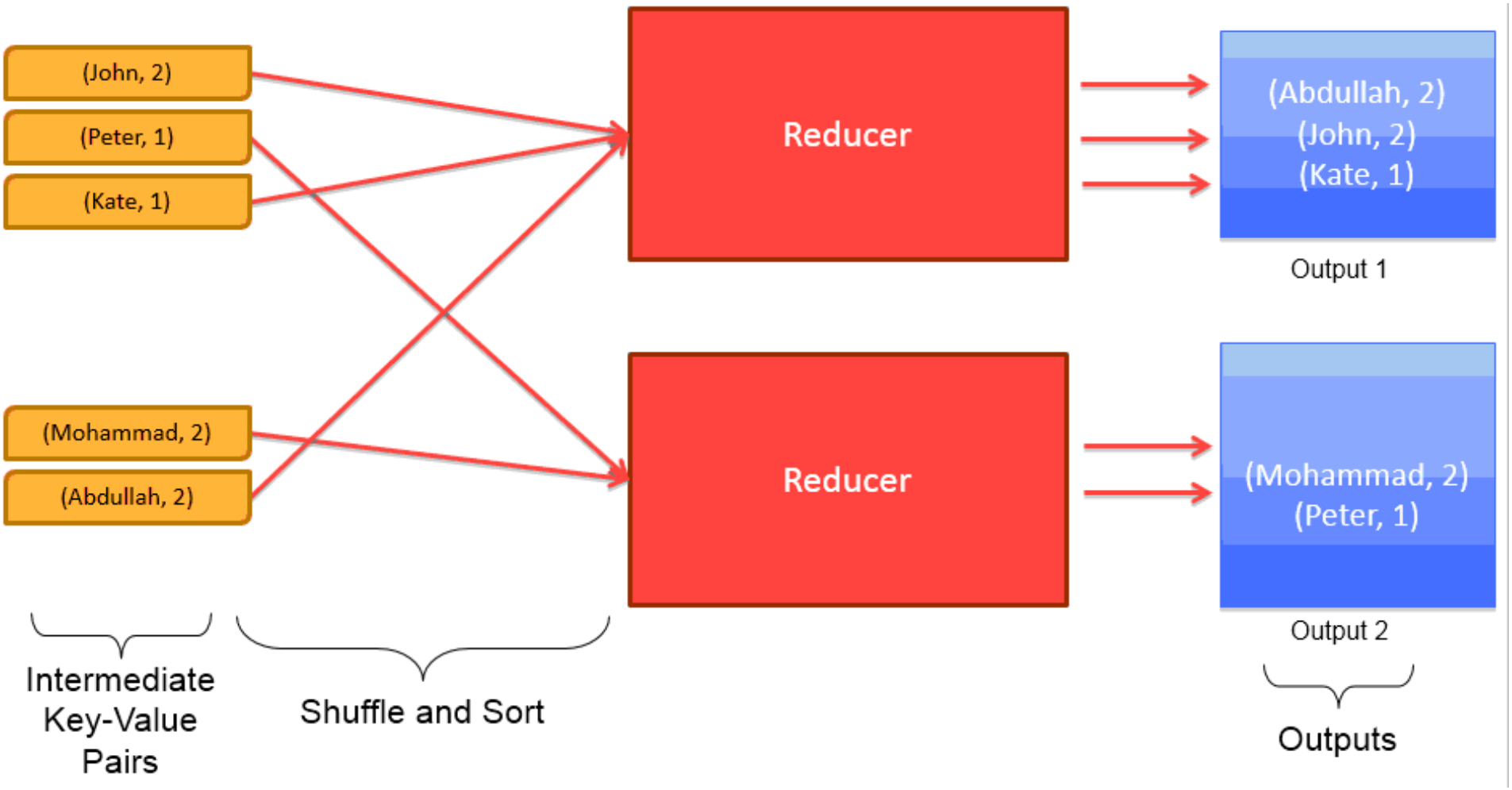
MapReduce flow



Wordcount example (1)



Wordcount example (2)



More details…

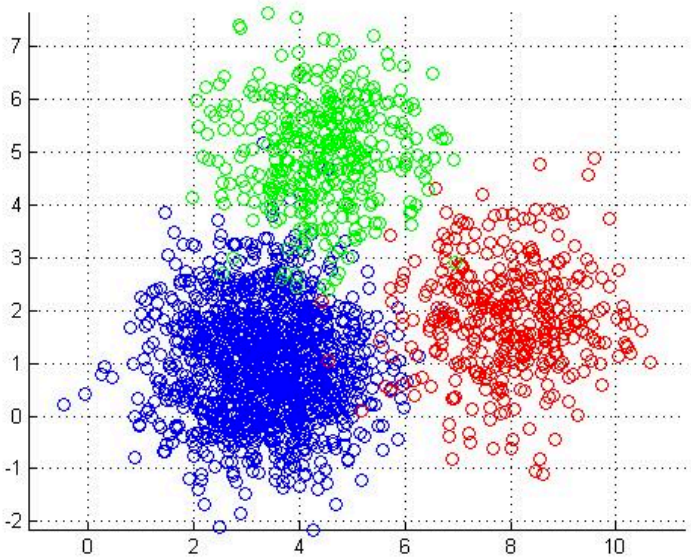
- Driver in Hadoop (job configuration)
  - Specify where to load input and where to put output
  - Specify input format and output format
  - Specify mapper, combiner, partitioner, and reducer
  - Specify number of mappers, reducers, etc.
  - …

- Chain multiple jobs:
  - Using dependencies among jobs
  - Using internal counter: iterative jobs


K-Means clustering algorithm (MapReduce)

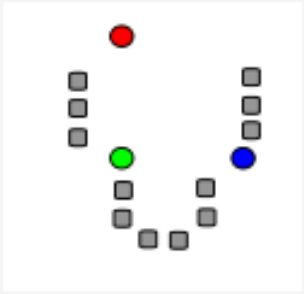
K-Means

- Unsupervised learning algorithm
- Partition a given data set into a certain number of K clusters (K is predefined)

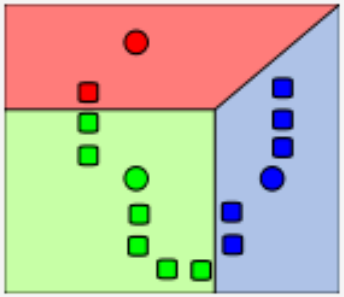


Algorithm

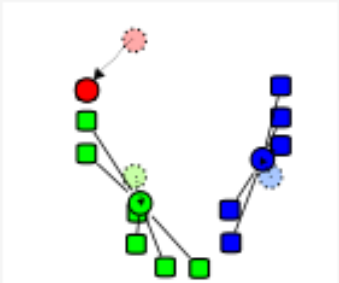
Demonstration of the standard algorithm



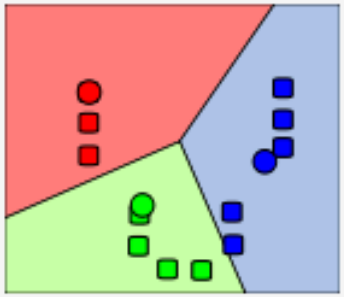
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

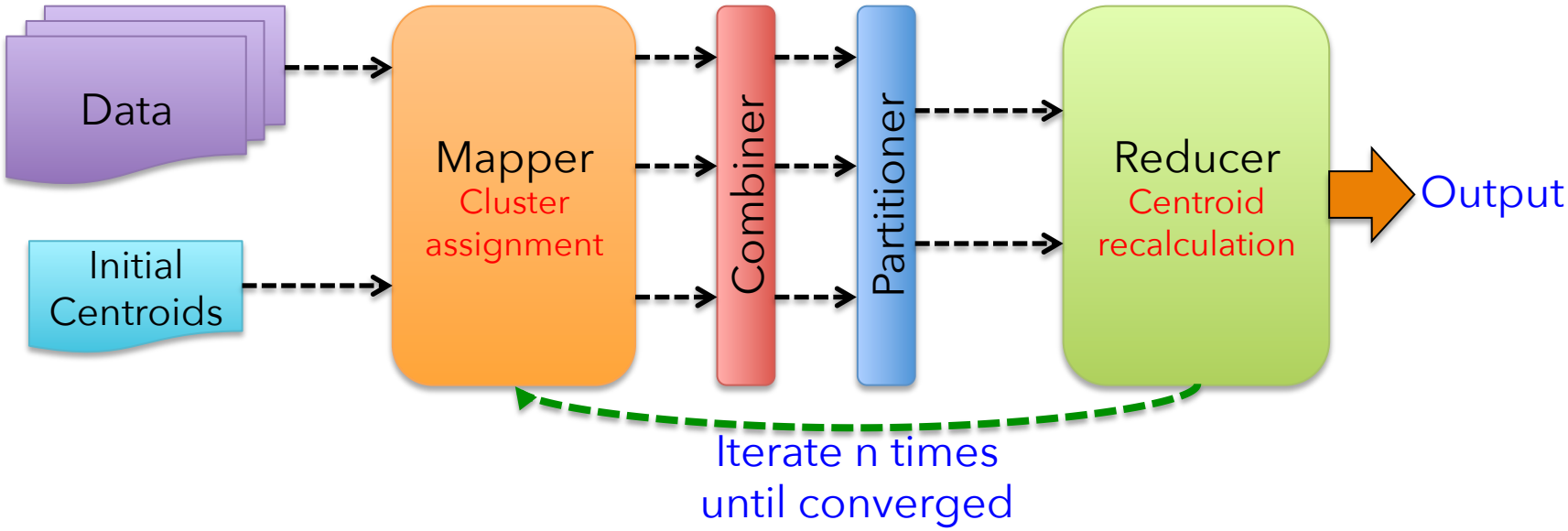


3) The centroid of each of the *k* clusters becomes the new means.



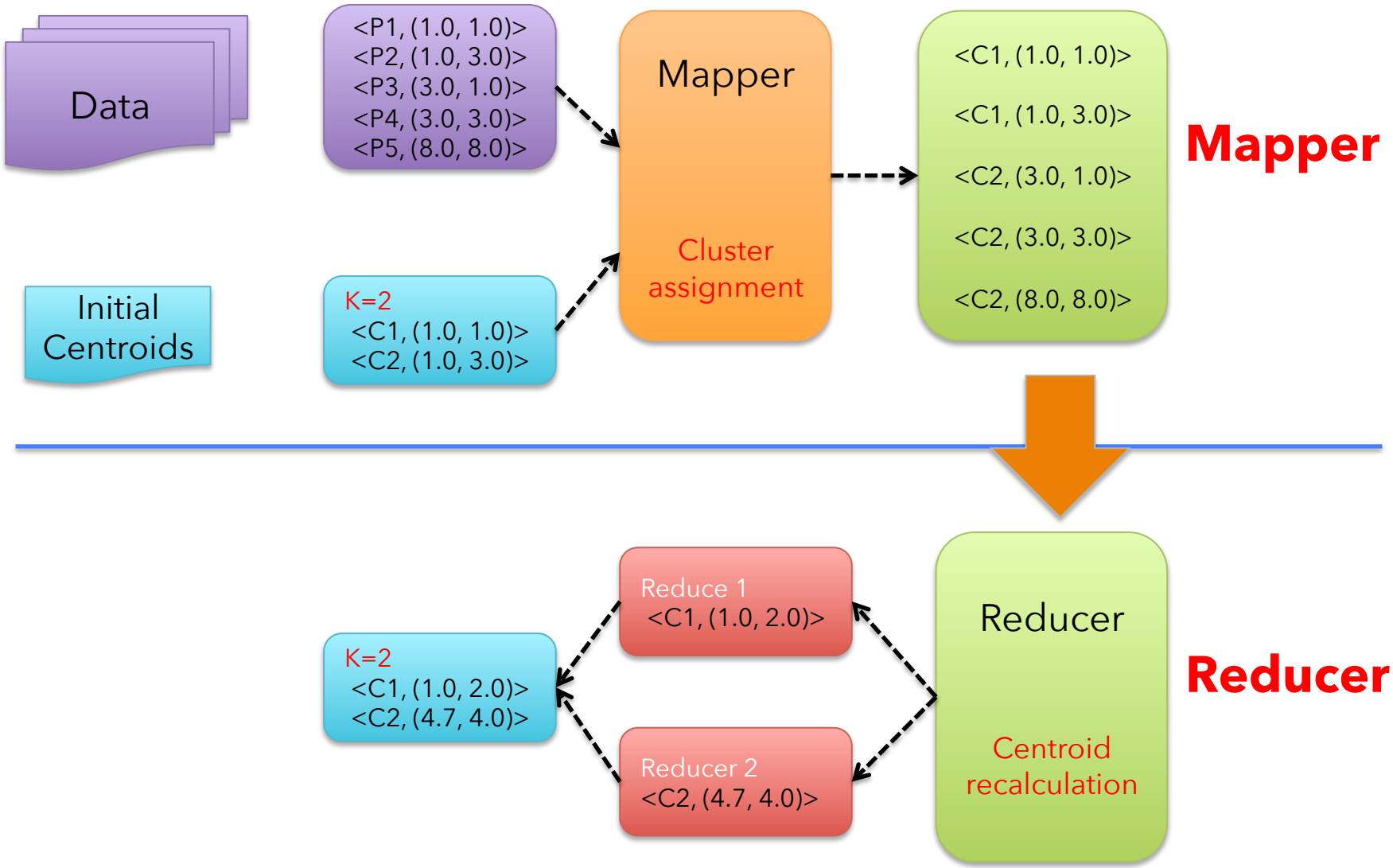
4) Steps 2 and 3 are repeated until convergence has been reached.

K-Means under MapReduce



Data never changes in the MapReduce process

Mapper and Reducer



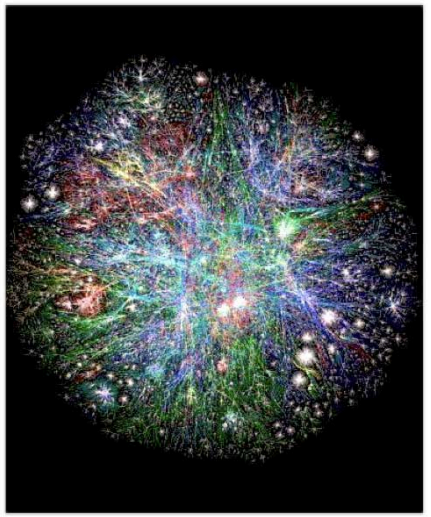
Remarks

- The way to initialize the means was not specified. One popular way to start is to randomly choose K of the samples.
- The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points.
- The results depend on the metric used to measure distance between data and centroids.
- The results depend on the value of K.


Iterative graph algorithm: single source shortest path

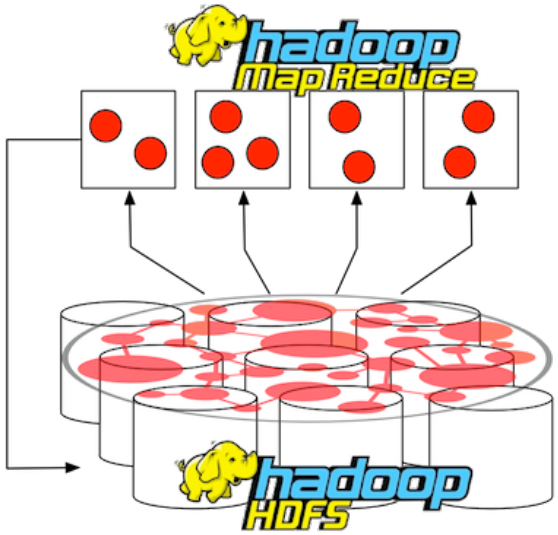
What is a graph?

- G = (V, E), where
  - V represents the set of nodes
  - E represents the set of links
  - Links and nodes may contain weights
- Different types of graphs
  - Directed vs. undirected
  - Weighted vs. unweighted
  - Acyclic vs. cyclic
- Networks/graphs are everywhere:
  - Social networks
  - Traffic networks
  - Web link networks



Graph and MapReduce

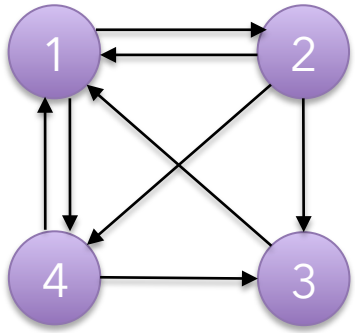
- Graph algorithms typically involves:
  - Computing something about nodes and links
  - Propagating information to the neighbors
- Key issues:
  - Graph representation under MapReduce framework
  - Information propagation under MapReduce framework



Graph representation

- Adjacency matrix

	1	**2**	**3**	**4**
1	0	1	0	1
2	1	0	1	1
3	1	0	0	0
4	1	0	1	0



- Adjacency list

1: 2, 4

2: 1, 3, 4

3: 1

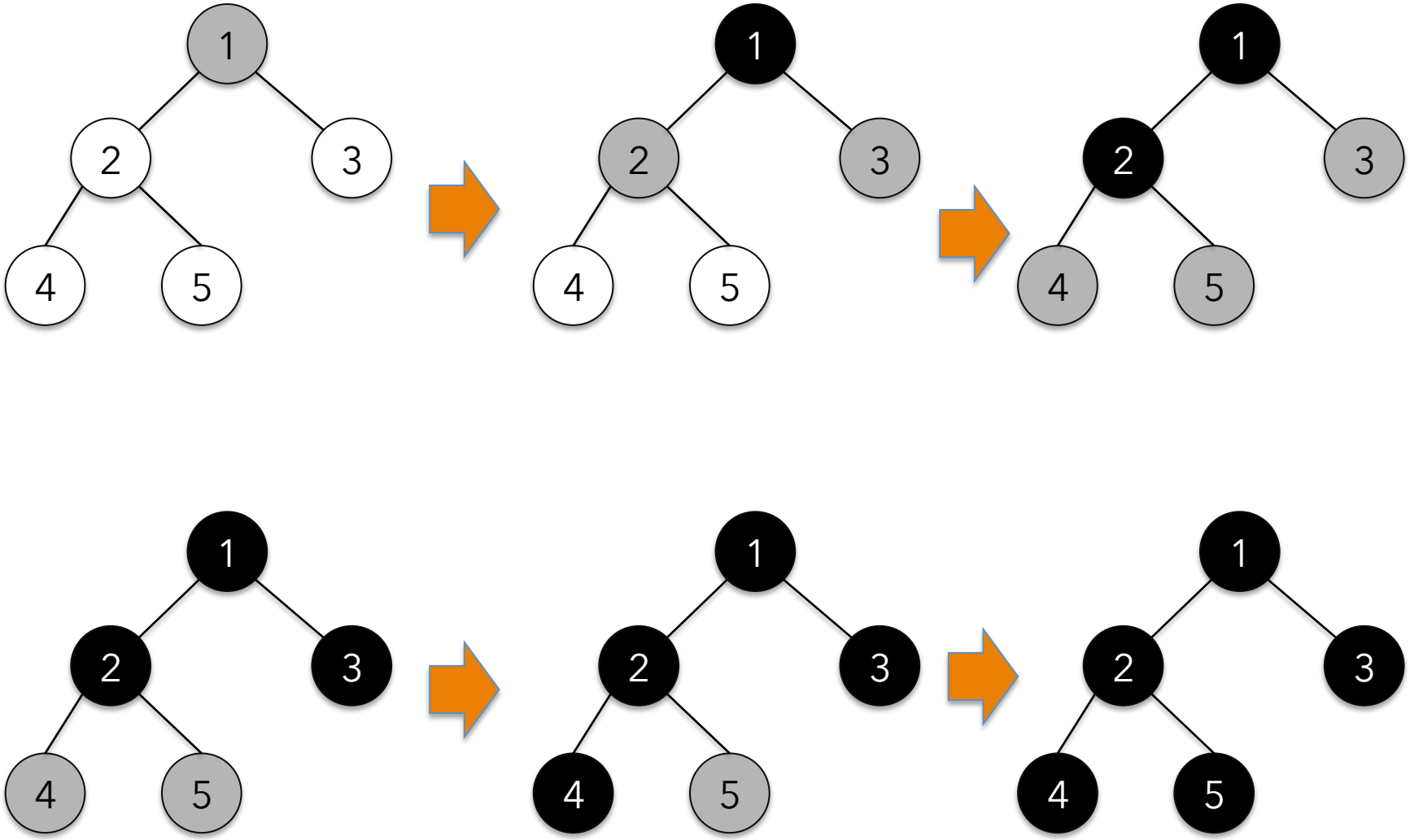
4: 1, 3

Single source shortest path

- **Problem**: find shortest path from a source node to one or more target nodes
- Single processor: Dijkstra's algorithm
- MapReduce: parallel breadth-first search (BFS)

Algorithm design

- One way of performing the BFS is by coloring the nodes and traversing according to the color of the nodes.
  - WHITE(unvisited), GRAY(visited) and BLACK(finished)
- At the beginning, all nodes are colored white.
- The source node is colored gray.
- The gray node indicates that it is visited and its neighbors should be processed.
- All the nodes adjacent to a gray node that are white are changed to be gray colored.
- The original gray node is colored.
- The process continues until there are no more gray nodes to process in the graph.


1
2
3
4
5
1
2
3
4
5
1
2
3
4
5
1
2
3
4
5
1
2
3
4
5
1
2
3
4
5

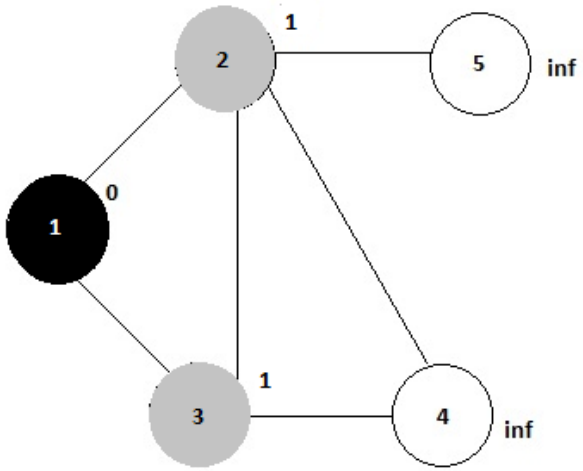
Algorithm design

- Data preparation (input format)
  - source<tab>adjacency list | distance from source | color | parent node
- Mapper
  - The nodes colored WHITE or BLACK are emitted as such. For each node that is colored GRAY, a new node is emitted with the distance incremented by one and the color set to GRAY. The original GRAY colored node is set to BLACK color and it is also emitted.
- Reducer
  - Make a new node which combines all information for this single node id that is for each key. The new node should have the full list of edges, the minimum distance, the darkest Color, and the parent/predecessor node

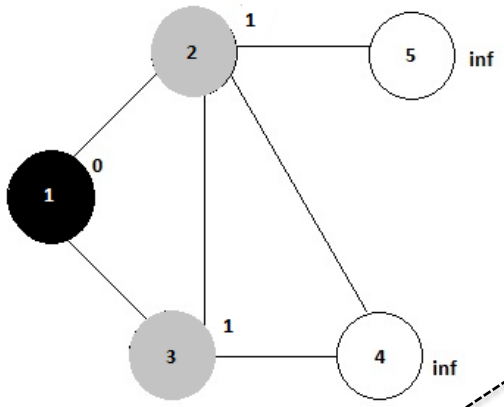
Example

- Input:

  1<tab>2,3,|0|BLACK|source
  2<tab>1,3,4,5,|1|GRAY|1
  3<tab>1,4,2,|1|GRAY|1
  4<tab>2,3,|Integer.MAX_VALUE|WHITE|null
  5<tab>2,|Integer.MAX_VALUE|WHITE|null



Mapper

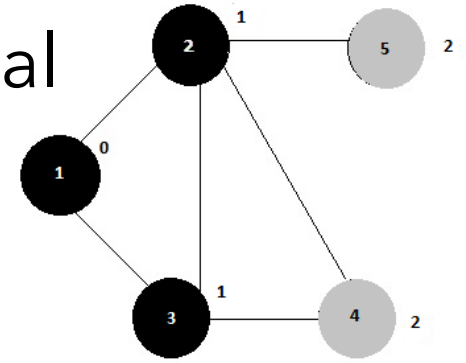


1<tab>2,3,|0|BLACK|source
2<tab>1,3,4,5,|1|GRAY|1
3<tab>1,4,2,|1|GRAY|1
4<tab>2,3,|Integer.MAX_VALUE|WHITE|null
5<tab>2,|Integer.MAX_VALUE|WHITE|null

1<tab>2,3,|0|BLACK|source
1<tab>2,3,|2|GRAY|source
3<tab>1,4,2,|2|GRAY|source
4<tab>2,3,|2|GRAY|source
5<tab>2,|2|GRAY|source
2<tab>1,3,4,5,|1|BLACK|1
1<tab>2,3,|2|GRAY|source
4<tab>2,3,|2|GRAY|source
2<tab>1,3,4,5,|2|GRAY|source
3<tab>1,4,2,|1|BLACK|1
4<tab>2,3,|Integer.MAX_VALUE|WHITE|null
5<tab>2,|Integer.MAX_VALUE|WHITE|null

Reducer

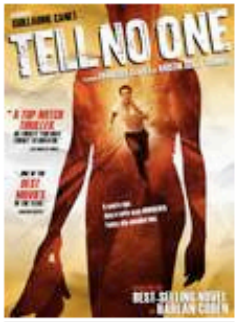
- For the same key, choose the darkest and the minimum distance as the final value



1<tab>2,3,|0|BLACK|source
1<tab>2,3,|2|GRAY|source
3<tab>1,4,2,|2|GRAY|source
4<tab>2,3,|2|GRAY|source
5<tab>2,|2|GRAY|source
2<tab>1,3,4,5,|1|BLACK|1
1<tab>2,3,|2|GRAY|source
4<tab>2,3,|2|GRAY|source
2<tab>1,3,4,5,|2|GRAY|source
3<tab>1,4,2,|1|BLACK|1
4<tab>2,3,|Integer.MAX_VALUE|WHITE|null
5<tab>2,|Integer.MAX_VALUE|WHITE|null

1<tab>2,3,|0|BLACK|source
4<tab>2,3,|2|GRAY|source
5<tab>2,|2|GRAY|source
2<tab>1,3,4,5,|1|BLACK|1
4<tab>2,3,|2|GRAY|source
3<tab>1,4,2,|1|BLACK|1

FOREIGN SUGGESTIONS (about 104) See all >



Tell No One

Because you enjoyed:
Memento
Syriana
Children of Men

Add

Not Interested

Let the Right One In

Because you enjoyed:
Seven Samurai
This Is Spinal Tap
The Big Lebowski

Add

Not Interested

I've Loved You So Long

Because you enjoyed:
The Queen
Syriana
Good Night, and Good Luck

Add

Not Interested

Downfall

Because you enjoyed:
Das Boot
The Killing Fields
Seven Samurai

Add

Not Interested

DRAMA SUGGESTIONS (about 82) See all >

The Wrestler

Because you enjoyed:
Sin City
Reservoir Dogs
The Big Lebowski

Add

Not Interested

The Visitor

Because you enjoyed:
Gandhi
The Motorcycle Diaries
The Queen

Add

Not Interested

Brick

Because you enjoyed:
The Big Lebowski
Rushmore
Fight Club

Add

Not Interested

The Pianist

Because you enjoyed:
Amadeus
The Killing Fields
Empire of the Sun

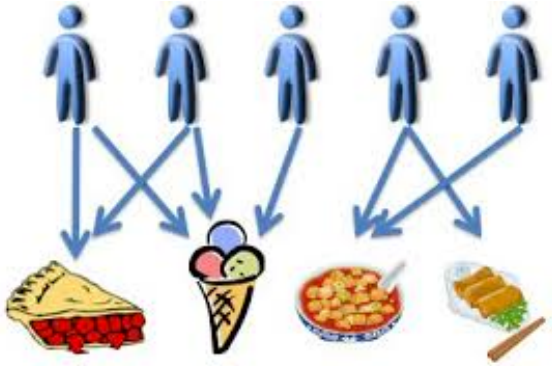
Add

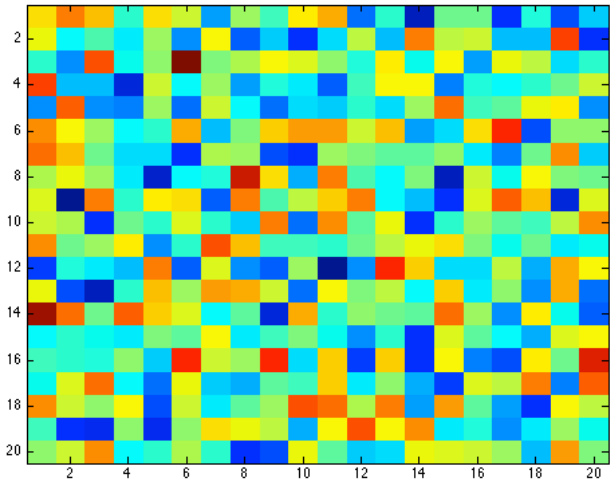
Not Interested

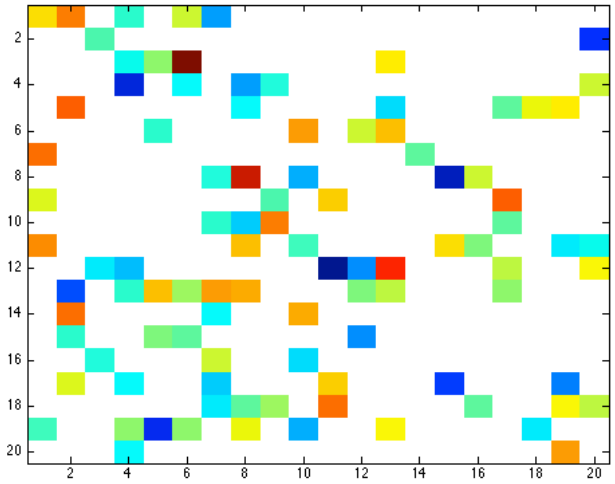
Recommender system

Recommender systems

- Collaborative filtering (CF) algorithm
  - Item-based
  - User-based
  - Content-based
- Matrix completion algorithm

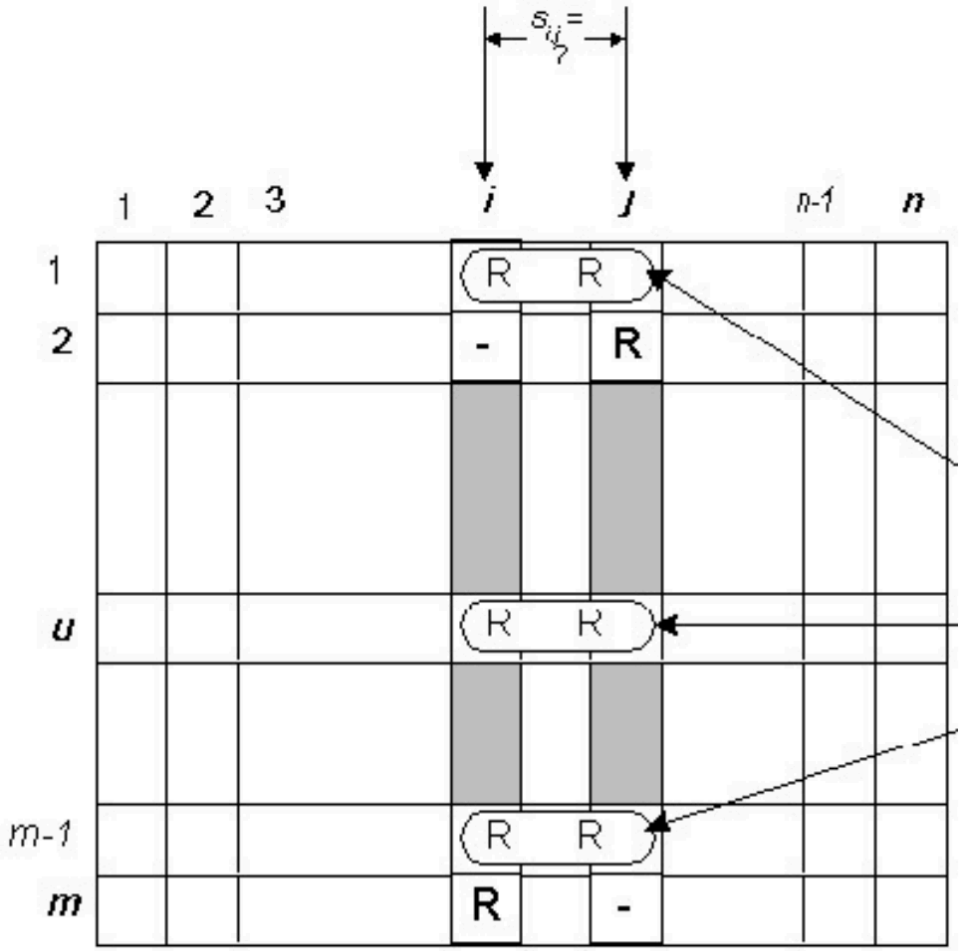






Item-based CF

- Key idea: calculate item similarities



Computed by looking into co-rated items only. These co-rated pairs are obtained from different users.

Item similarity

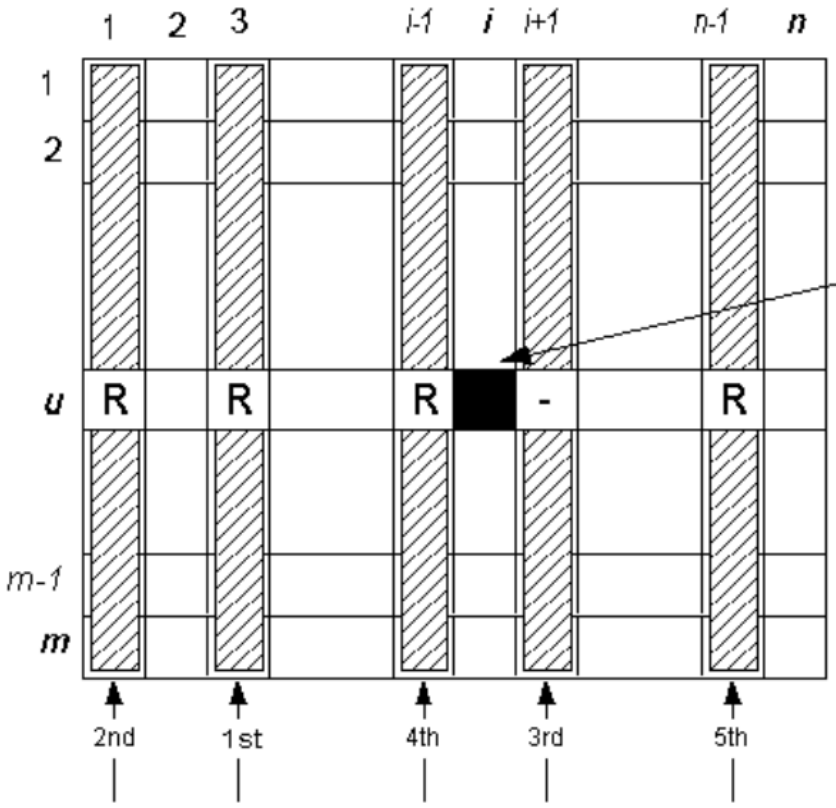
- Similarity of The Matrix and Inception
  - rating vector of The Matrix: (5, -, 4)
  - rating vector of Inception: (4, 5, 2)
  - isolate all co-occurred ratings (all cases where a user rated both items)
  - Pick a similarity measure to compute a similarity value between -1 and 1. e.g. Pearson correlation, cosine similarity, etc.

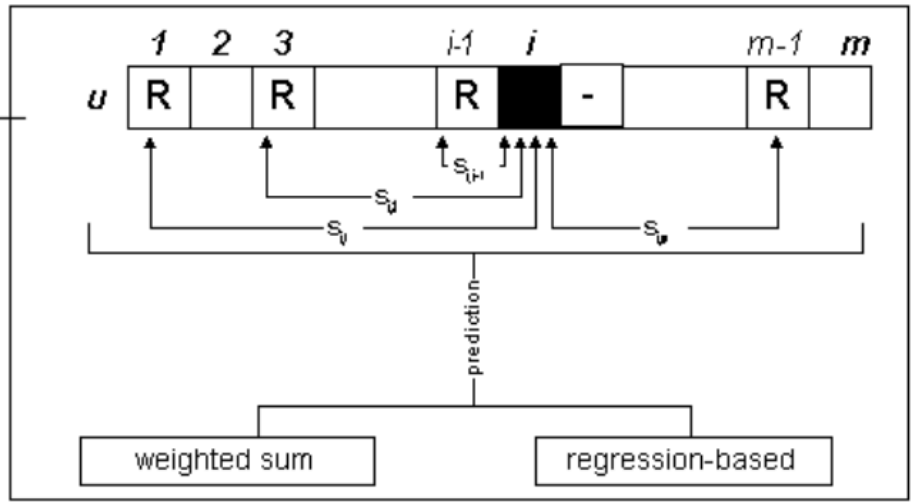
5	4
-	5
4	2

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}$$

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Prediction





$$P_{u,i} = \frac{\sum_{\text{all similar items, N}} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}} (|s_{i,N}|)}$$

Algorithm in MapReduce – job 1

- **Map** – make user the key

(Alice,Matrix,5)	→	Alice (Matrix,5)
(Alice,Alien,1)	→	Alice (Alien,1)
(Alice,Inception,4)	→	Alice (Inception,4)
(Bob,Alien,2)	→	Bob (Alien,2)
(Bob,Inception,5)	→	Bob (Inception,2)
(Peter,Matrix,4)	→	Peter (Matrix,4)
(Peter,Alien,3)	→	Peter (Alien,3)
(Peter,Inception,2)	→	Peter (Inception,2)

- **Reduce** – create inverted index

Alice (Matrix,5) Alice (Alien,1) Alice (Inception,4)	→	Alice (Matrix,5)(Alien,1)(Inception,4)
Bob (Alien,2) Bob (Inception,5)	→	Bob (Alien,2)(Inception,5)
Peter (Matrix,4) Peter (Alien,3) Peter (Inception,2)	→	Peter (Matrix,4)(Alien,3)(Inception,2)

Algorithm in MapReduce – job 2

- **Map** – emit all co-occurred ratings

Alice (Matrix,5)(Alien,1) (Inception,4)	→	Matrix,Alien (5,1) Matrix,Inception (5,4) Alien,Inception (1,4)
Bob (Alien,2)(Inception,5)	→	Alien,Inception (2,5)
Peter (Matrix,4)(Alien,3) (Inception,2)	→	Matrix,Alien (4,3) Matrix,Inception (4,2) Alien,Inception(3,2)

- **Reduce** – compute similarities

Matrix,Alien (5,1) Matrix,Alien (4,3)	→	Matrix,Alien (-0.47)
Matrix,Inception (5,4) Matrix,Inception (4,2)	→	Matrix,Inception (0.47)
Alien,Inception (1,4) Alien,Inception (2,5) Alien,Inception (3,2)	→	Alien,Inception (-0.63)

Some good tools

- Spark
  - http://spark.apache.org
- Mahout
  - https://mahout.apache.org
- D3.js
  - http://d3js.org
- Others

- E-Mail: kzhang6@uic.edu
- Website: http://kzhang6.people.uic.edu