Technology and Application of Big Data

Qing LIAO(廖清)

School of Computer Science and Technology

HIT

Course Details

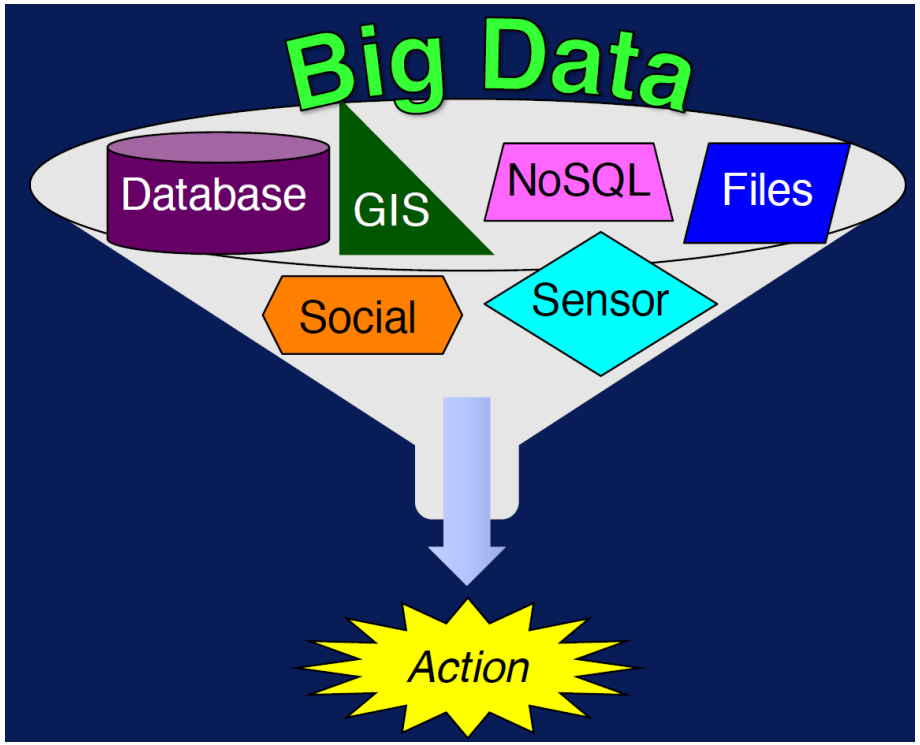
- Instructor:
  - Qing LIAO, liaoqing@hit.edu.cn
  - Rm. 303B, Building C
  - Office hours: by appointment
- Course web site:
  - liaoqing.me
- Reference books/materials:
  - Big data courses from University of California
  - Book: BIG DATA: A Revolution That Will Transform How We Live, Work, and Think
  - Papers
- Grading Scheme:
  - Paper Report 30%
  - Final Exam 70%
- Exam:
  - 21st July(Friday), 14:00-16:00, A502

What You Learnt: Overview

- Topics:
  1) Introduction of Big Data
  2) Characterizes of Big Data
  3) How to Get Value from Big Data
  4) Technologies of Big Data
  5) Applications of Big Data
- Prerequisites
  - Statistics and Probability would help
    - But not necessary
  - Machine Learning would help
    - But not necessary

Previous Section

- How to Get Value from Big Data



Previous Section

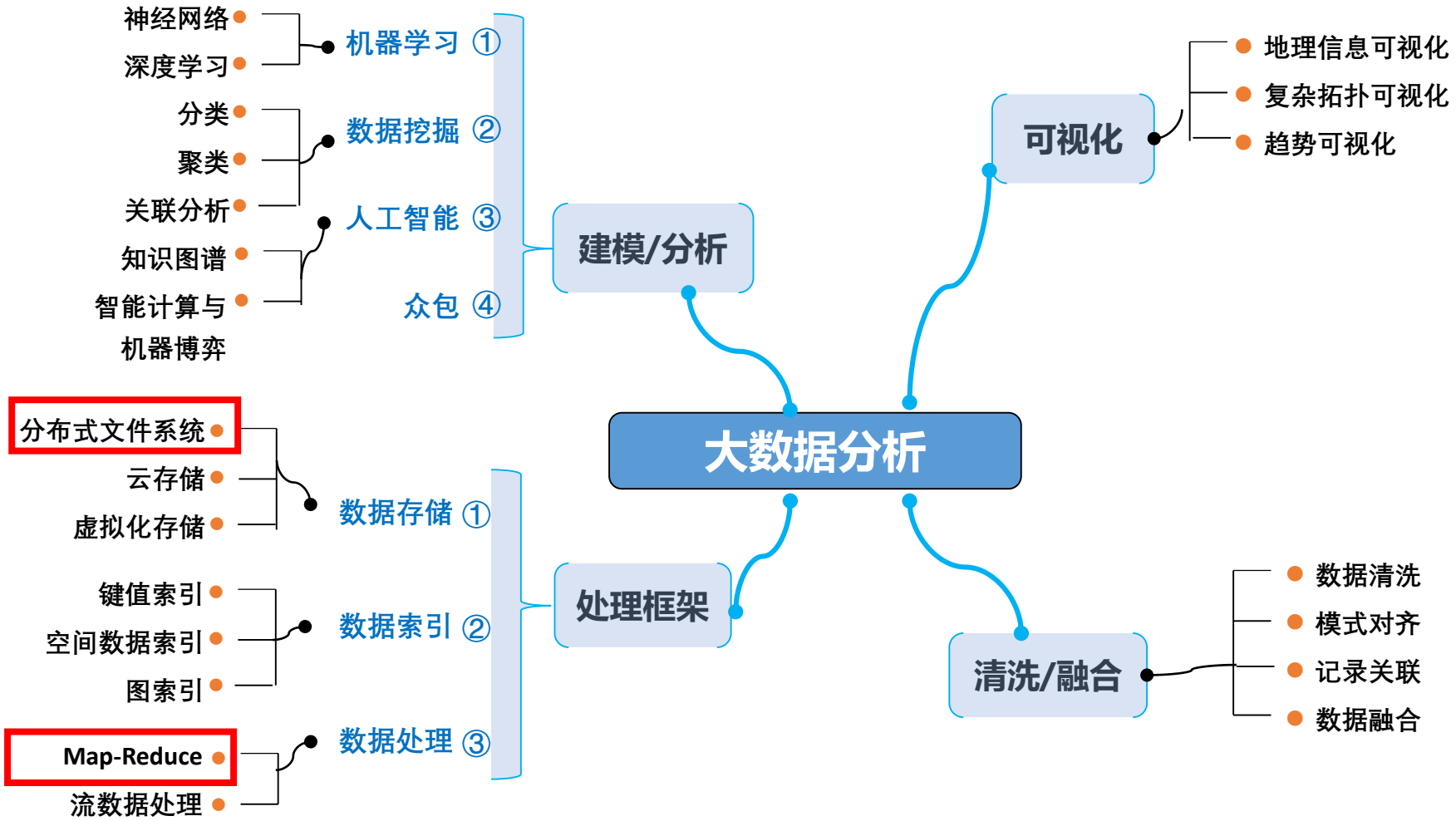
- Information Extraction results

(a). An example page segment

image 1	Cabinet Organizers by Copco	9-in.	Round Turntable: White	*****	$4.95
image 1	Cabinet Organizers by Copco	12-in.	Round Turntable: White	*****	$7.95
image 2	Cabinet Organizers	14.75x9	Cabinet Organizer (Non-skid): White	*****	$7.95
image 3	Cabinet Organizers	22x6	Cookware Lid Rack	****	$19.95

(b). Extraction results

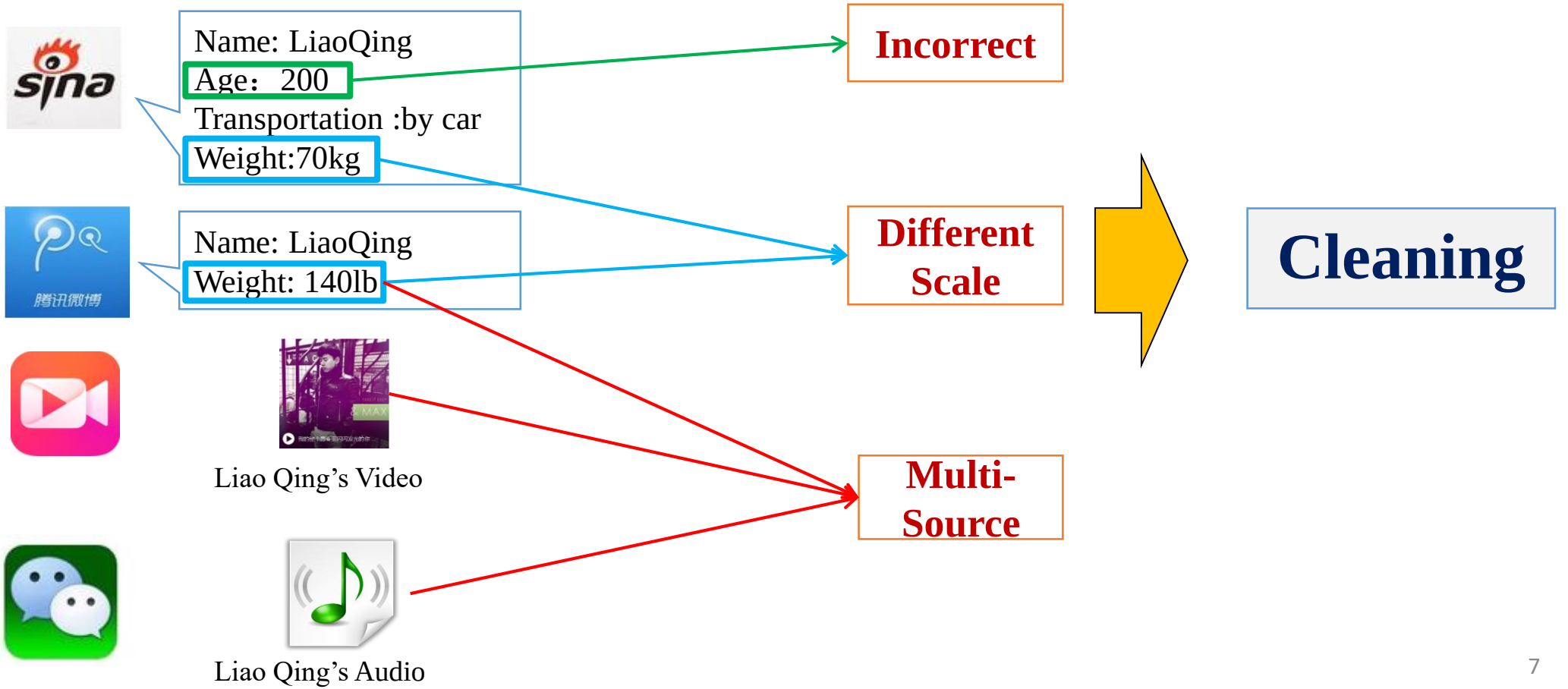
Technologies of Big Data



Thanks NUDT Provides Slides

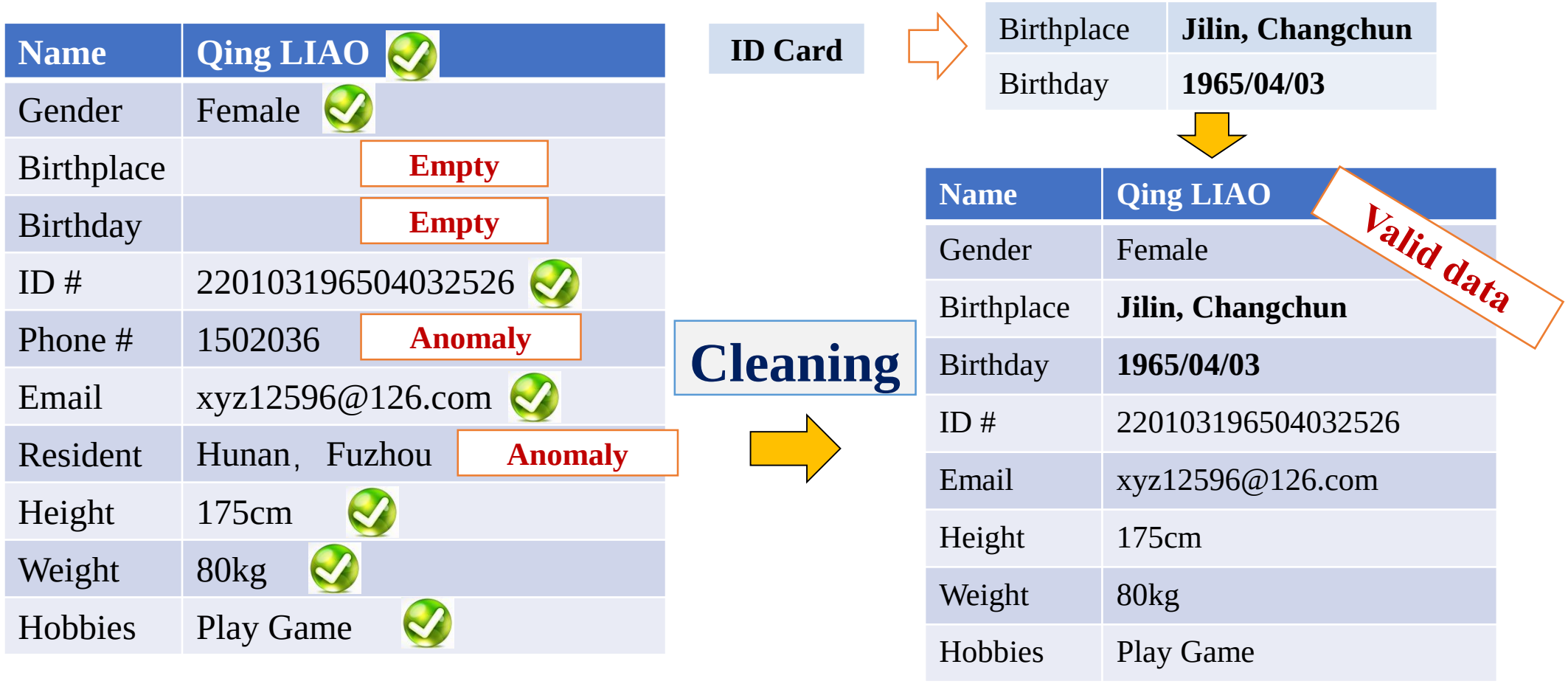
Previous Section

- Step 2: Prepare Data - Pre-process



Previous Section

- Step 2: Prepare Data - Pre-process



Previous Section

- Step 2: Prepare Data - Pre-process

Data Source 1	**(Name, family phone #, home address, office phone # , office address)**
Data Source 2	**(Family name, given name, nick name, phone #, address, QQ #)**
Data Source 3	**a: (id, name); b: (id, Private phone #, office phone #) no address**
Data Source 4	**(Family name, given name, Private phone #, , home address)**
Data Source 5	**(Nick name, name, zip code、 phone #, City ， street)**

Different Format

Target Format **(Nick name, name, phone #, address, QQ #)**

Data Source 1	**(~~Nick name~~, name, phone #, address, ~~QQ #~~)**
Data Source 2	**(Nick name, name, phone #, address, QQ #)**
Data Source 3	**(~~Nick name~~, name, phone #, address, ~~QQ #~~)**
Data Source 4	**(~~Nick name~~, name, phone #, address, ~~QQ #~~)**
Data Source 5	**(Nick name, name, phone #, address, ~~QQ #~~)**

Pattern Alignment

Previous Section

- Step 2: Prepare Data - Pre-process

Web Page

Color

Brand Sony/索尼 **Model** DSC-H300 **Type** 数码长焦照相机 35倍变焦 大广角 2010万像素 **Resolution**

13人付款 20条评论

正品Sony/索尼DSC-H300数码小单反相机 35倍长焦大陆行货全国联保 **Feature**

48人付款 129条评论

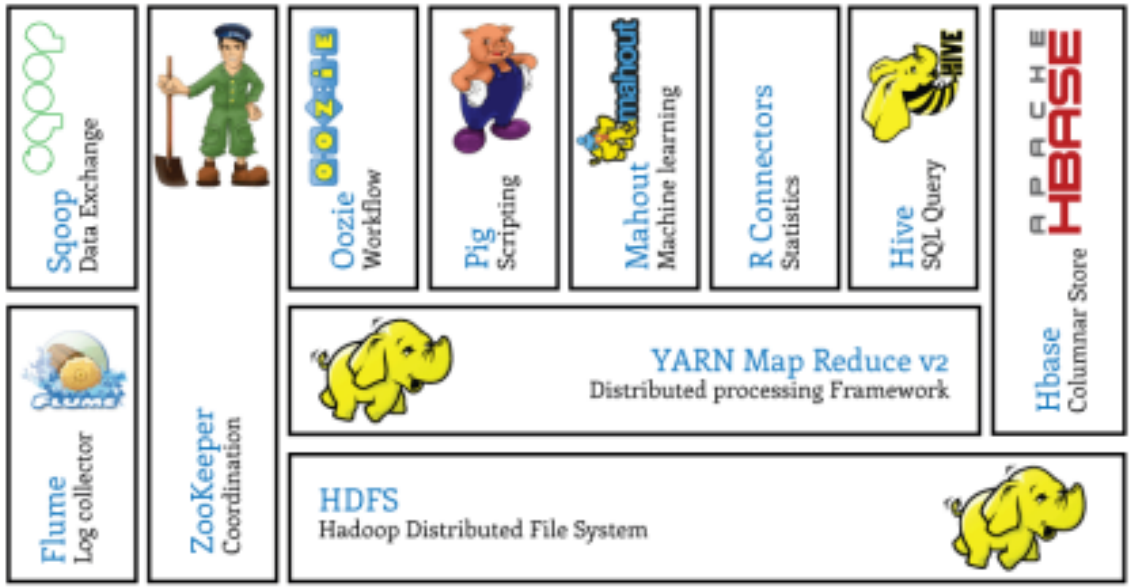
Record Association

Table Format

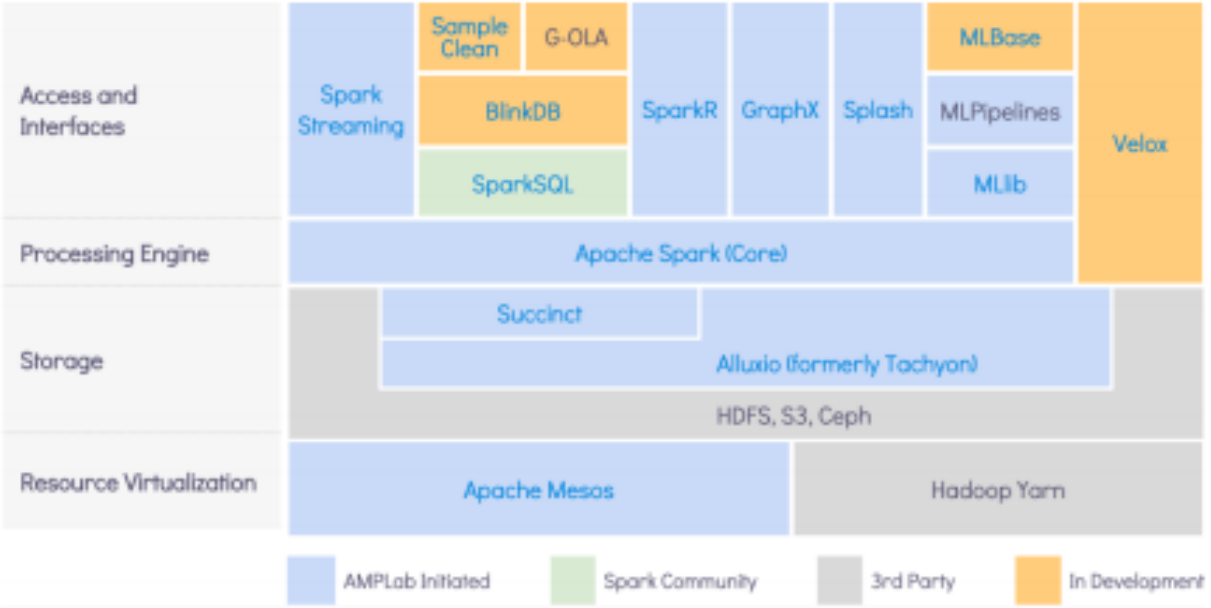
Category	Camera
Brand	Sony
Model	DSC-H300
Type	Telephoto Lens
Pixels	21 Million photo pixel
Color	Black
Feature	35X Telephoto Lens

Hadoop Eco-System

Hadoop Eco-System

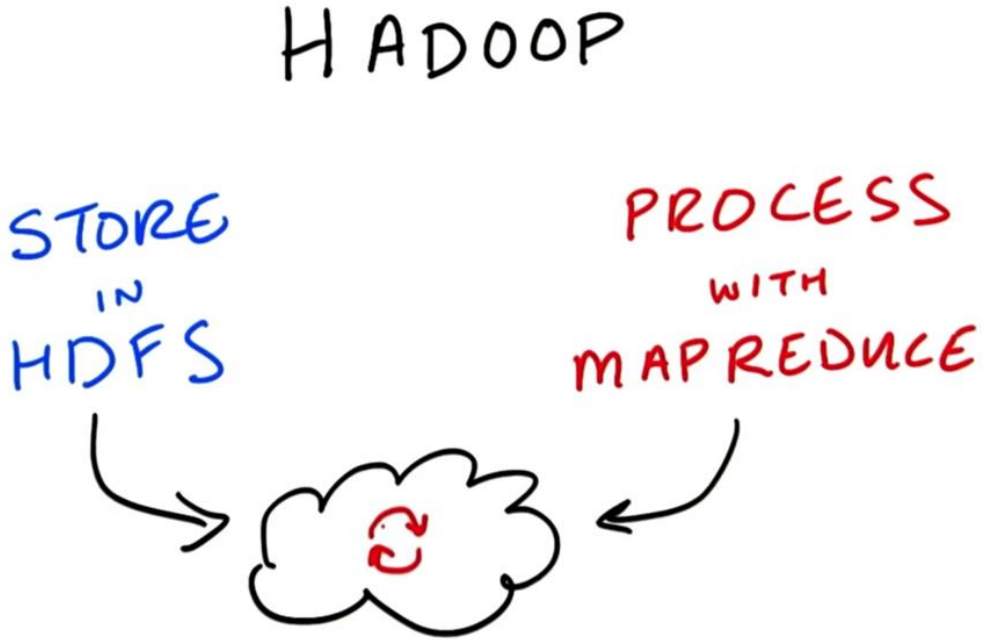


Spark Eco-System

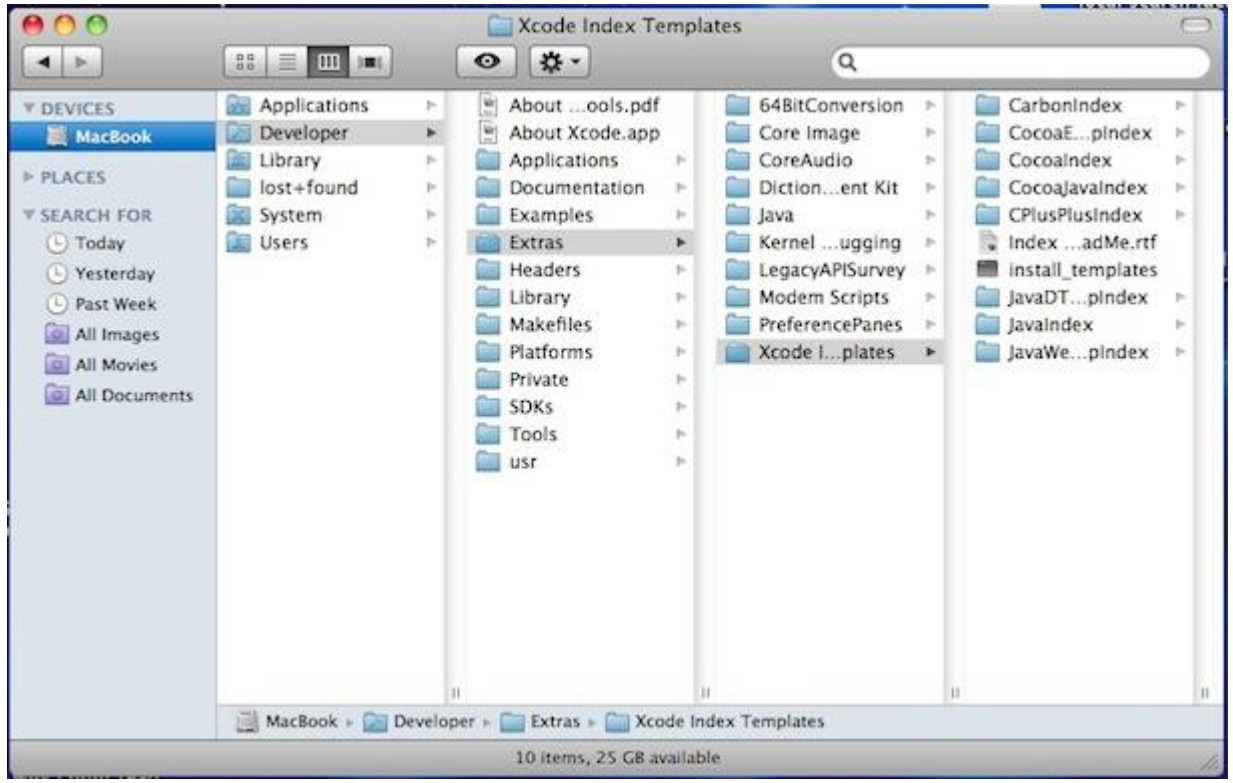


Hadoop HDFS

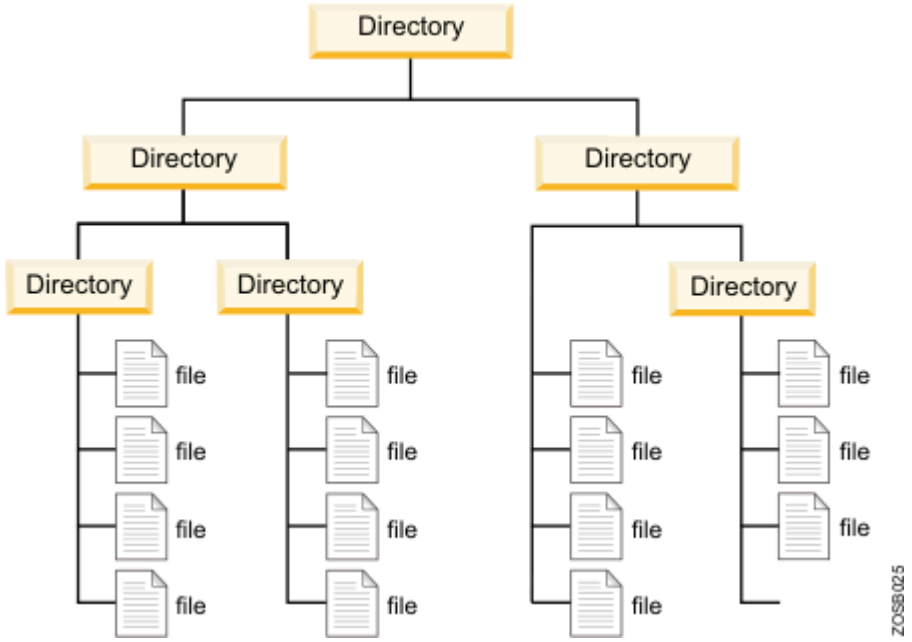
- Currently one of the best distributed file system
  - Based on GFS by Google in 2003
  - Doug Cutting first proposed by Yahoo in 2006



File System

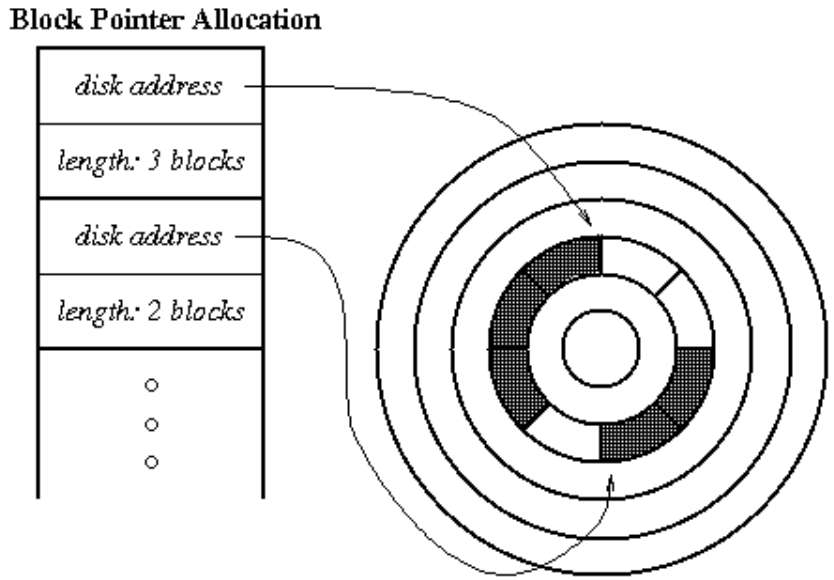


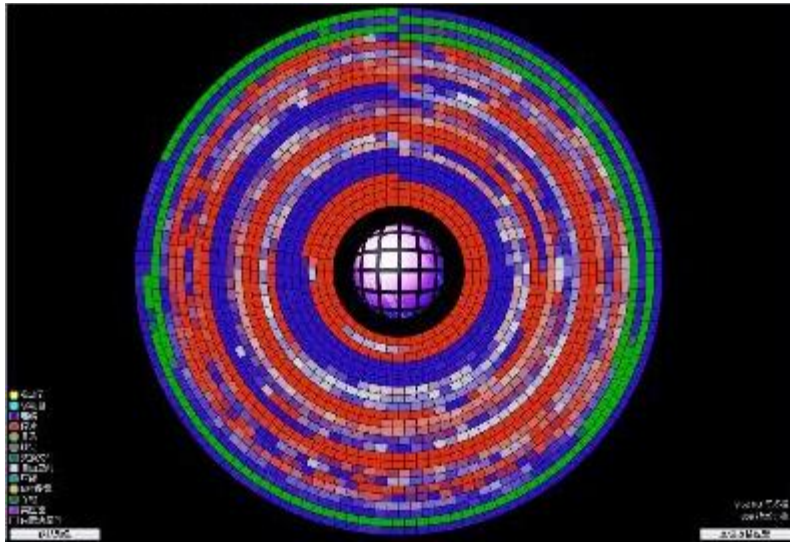
Structured, tree-like file storage



Local File System

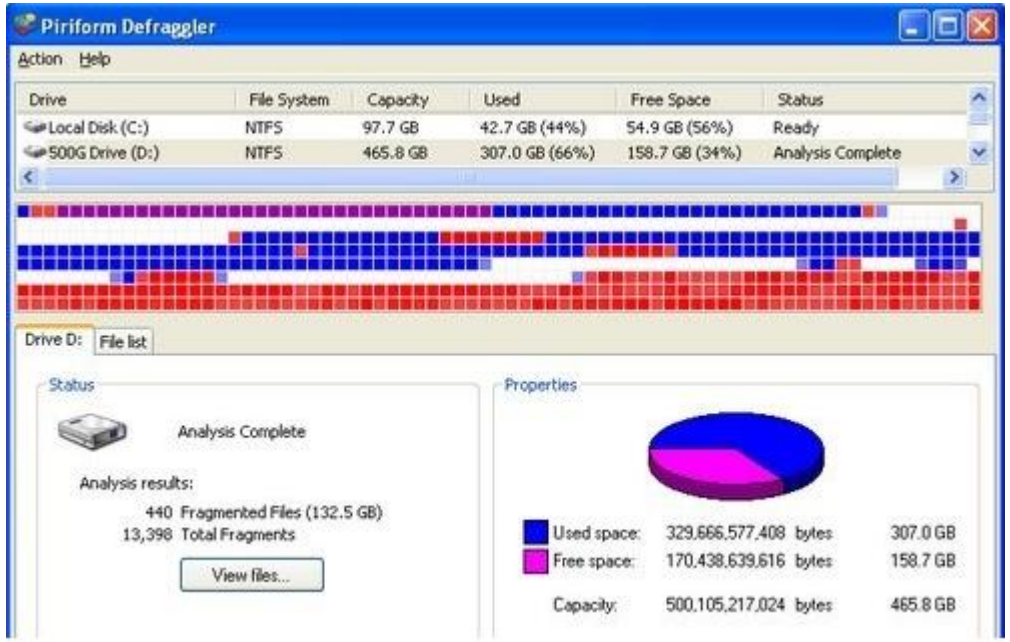
- Files are stored locally (in one hard disk)
- How files are really stored:
  - File system is split into two parts:
    - 1) Header / file pointer (inode)
    - 2) File data (in blocks)
    - Formally, file system format
    - E.g., FAT32, NTFS, ext4, HFS, APFS (since iPhone 7)
  - Files are not stored continuously
    - Formally, random access





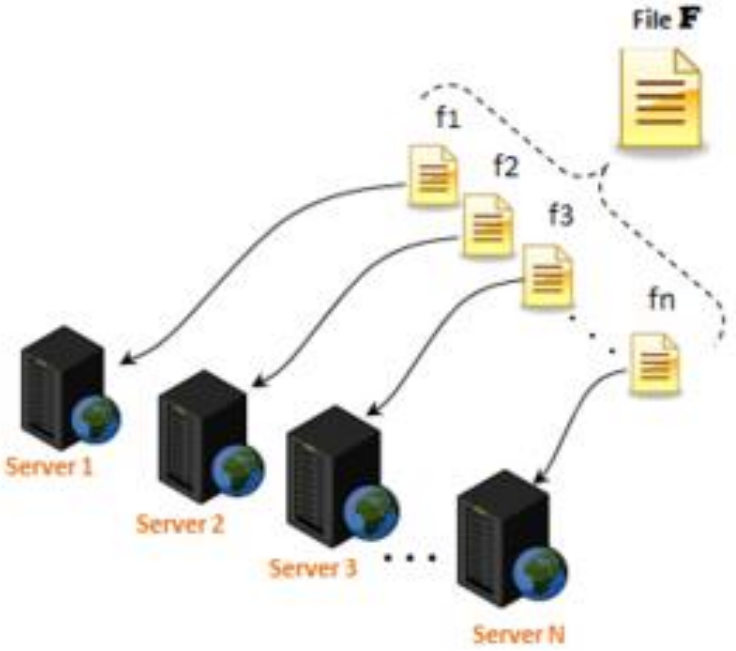
Local File System

- In some old, bad-implemented file systems, defragmentation is sometimes required.
- Most of the modern file system design has eliminated such problem.

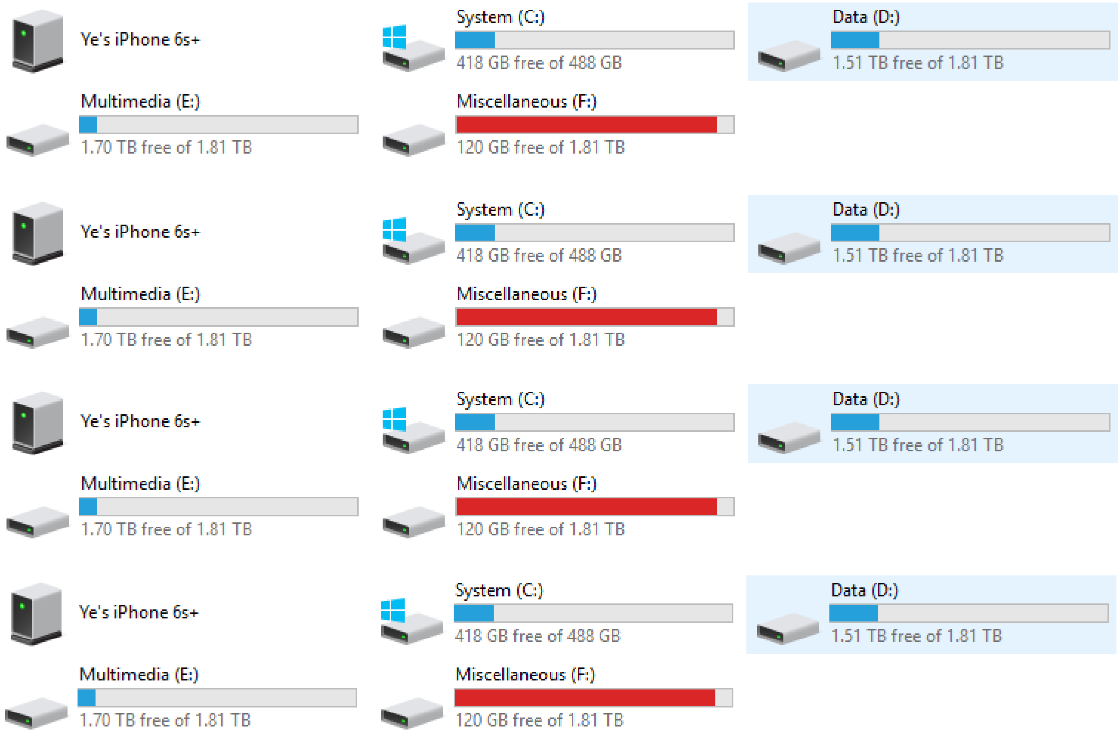


Distributed File System

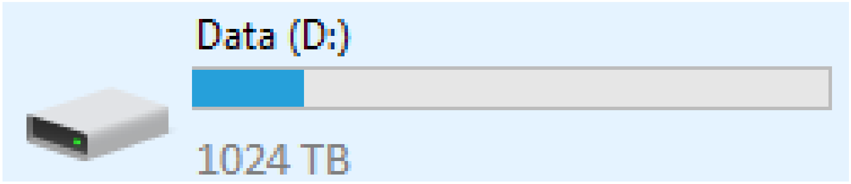
- Local file system: files are stored locally (in one hard disk)
- Distributed file system:
  - Files are stored in multiple hard disks
  - Sometimes even distributed across the Internet
- Pros:
  - Store up to peta level data
    - 1 SSD: up to 4 TB / 1 HDD: up to 16 TB
    - 1 PB = 1,024 TB / 64 HDDs
  - Content safe: replica / fault tolerance
  - Load balance benefits



Distributed File System



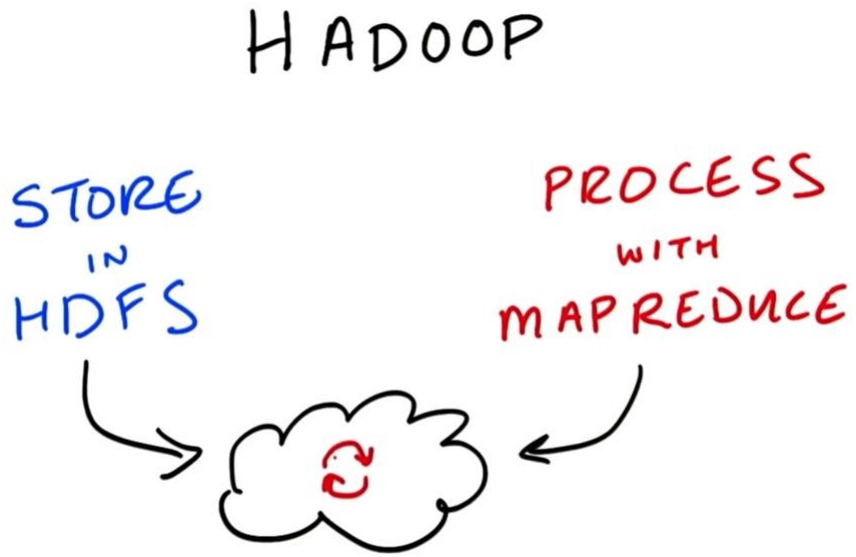
Without DFS

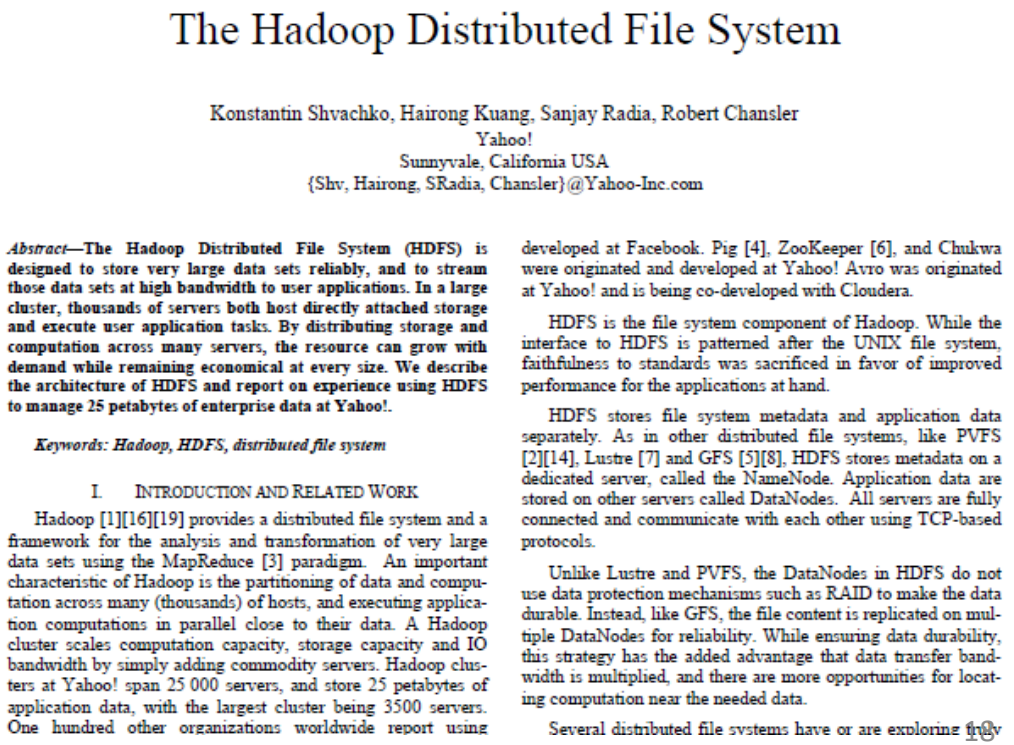


With DFS

Hadoop Distributed File System

- HDFS = Hadoop Distributed File System
- One of the implementation / standard of distributed file system
  - Based on GFS by Google in 2003
  - First proposed by Yahoo in 2006





The Hadoop Distributed File System

Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler
Yahoo!
Sunnyvale, California USA
{Shv, Hairong, SRadia, Chansler}@Yahoo-Inc.com

***Abstract*—The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 25 petabytes of enterprise data at Yahoo!.**

Keywords: Hadoop, HDFS, distributed file system

I. INTRODUCTION AND RELATED WORK

Hadoop [1][16][19] provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce [3] paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many (thousands) of hosts, and executing application computations in parallel close to their data. A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers. Hadoop clusters at Yahoo! span 25 000 servers, and store 25 petabytes of application data, with the largest cluster being 3500 servers. One hundred other organizations worldwide report using

developed at Facebook. Pig [4], ZooKeeper [6], and Chukwa were originated and developed at Yahoo! Avro was originated at Yahoo! and is being co-developed with Cloudera.

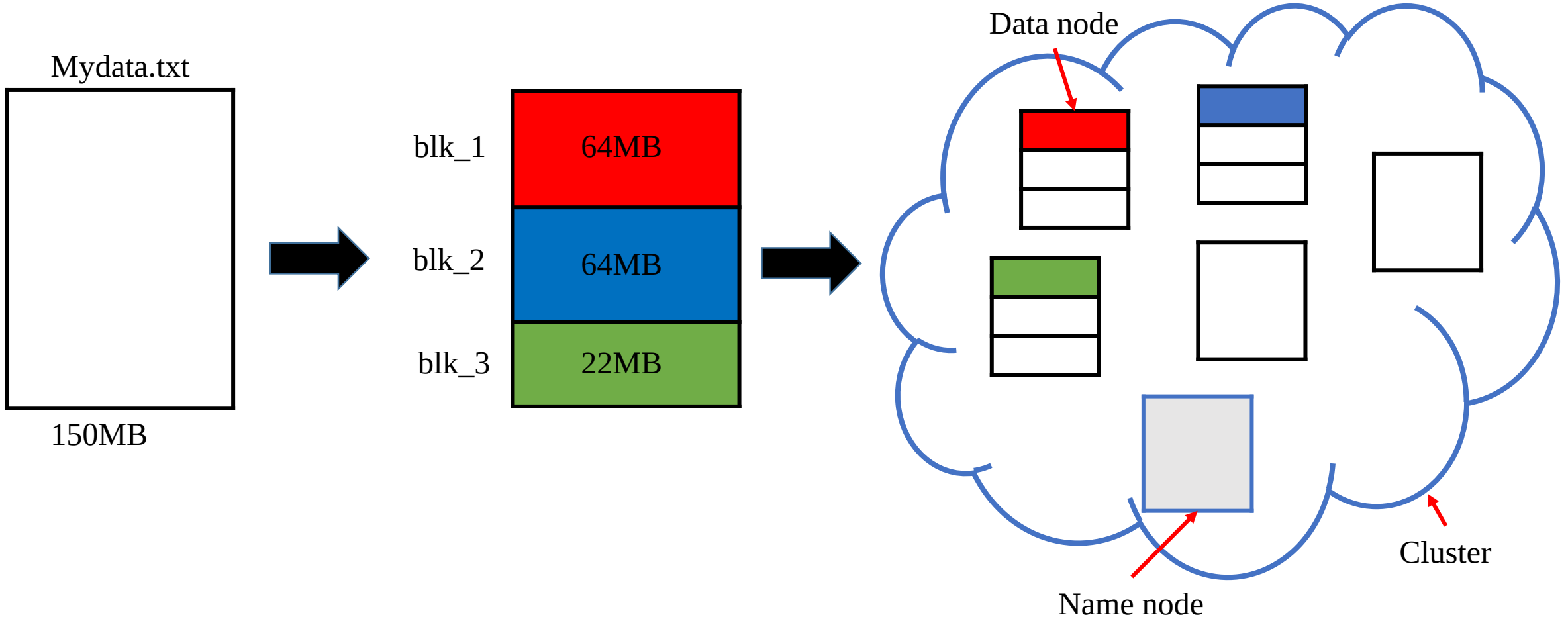
HDFS is the file system component of Hadoop. While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favor of improved performance for the applications at hand.

HDFS stores file system metadata and application data separately. As in other distributed file systems, like PVFS [2][14], Lustre [7] and GFS [5][8], HDFS stores metadata on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes. All servers are fully connected and communicate with each other using TCP-based protocols.

Unlike Lustre and PVFS, the DataNodes in HDFS do not use data protection mechanisms such as RAID to make the data durable. Instead, like GFS, the file content is replicated on multiple DataNodes for reliability. While ensuring data durability, this strategy has the added advantage that data transfer bandwidth is multiplied, and there are more opportunities for locating computation near the needed data.

Several distributed file systems have or are exploring truly

Hadoop Distributed File System



Hadoop Distributed File System

Is there a problem?

a) Network failure
b) Disk failure on data node
c) Not all data node used
d) Block size differ
e) Disk failure on name node

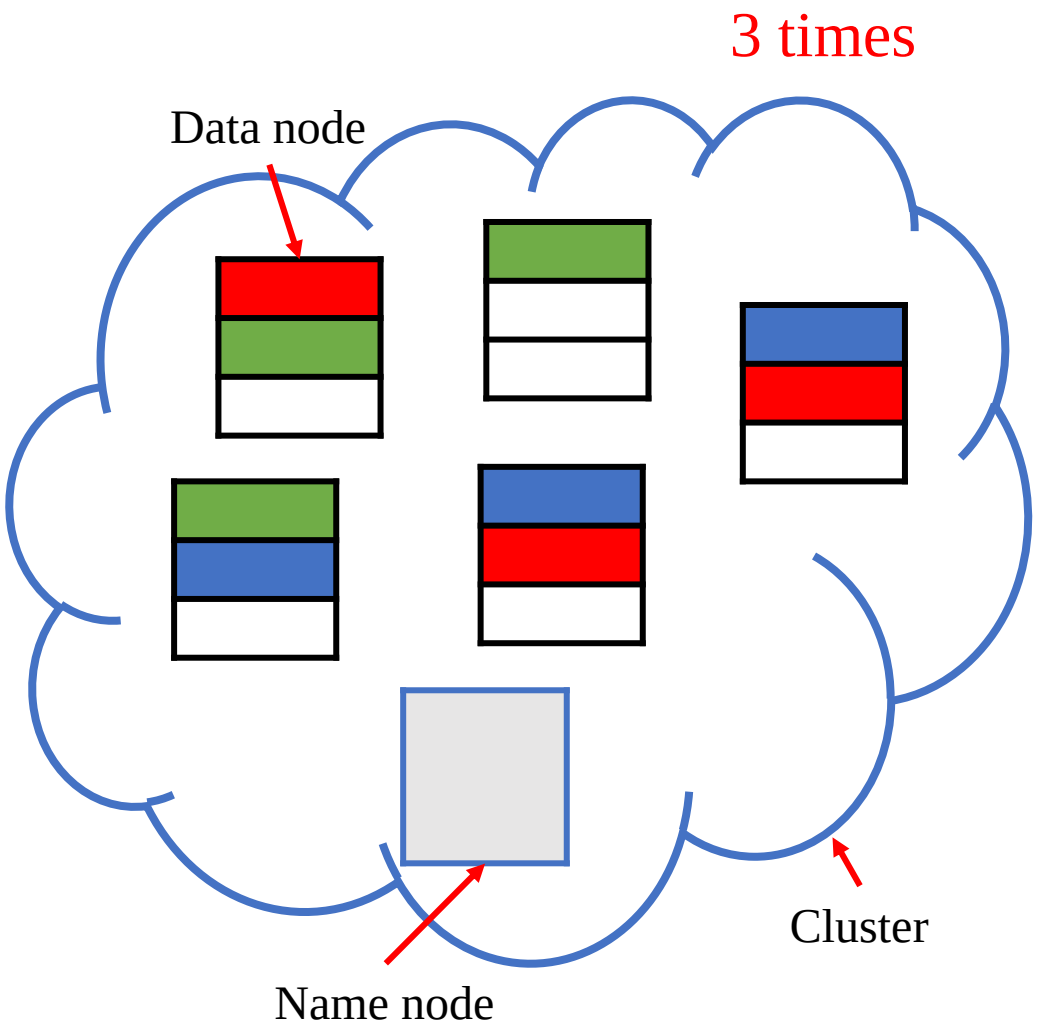
Hadoop Distributed File System

If one of our data nodes fails?

If this data node goes away?

Data Redundancy

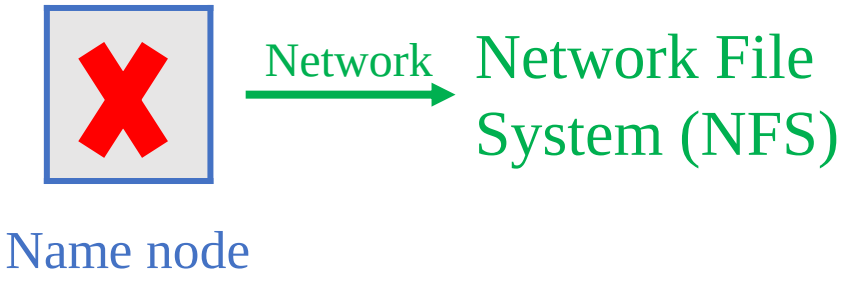
- Hadoop replicates each block three times as it's stored in HDFS
- NameNode is smart enough that if it sees that any of the blocks are under replicated, it will arrange to have those blocks re-replicated on the cluster so we're back to having three copies of them again



Hadoop Distributed File System

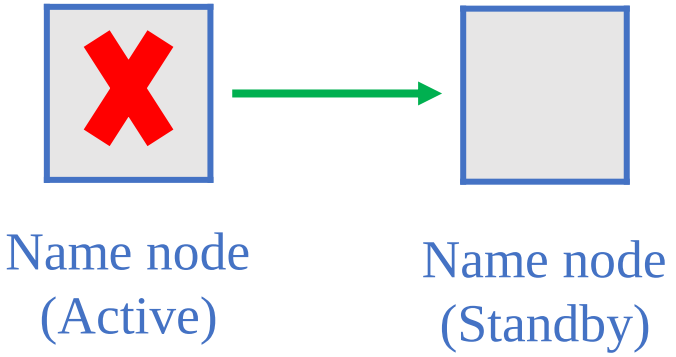
If the disk of the single NameNode would fail ?

- Data inaccessible
- Data on HDFS would be lost permanently



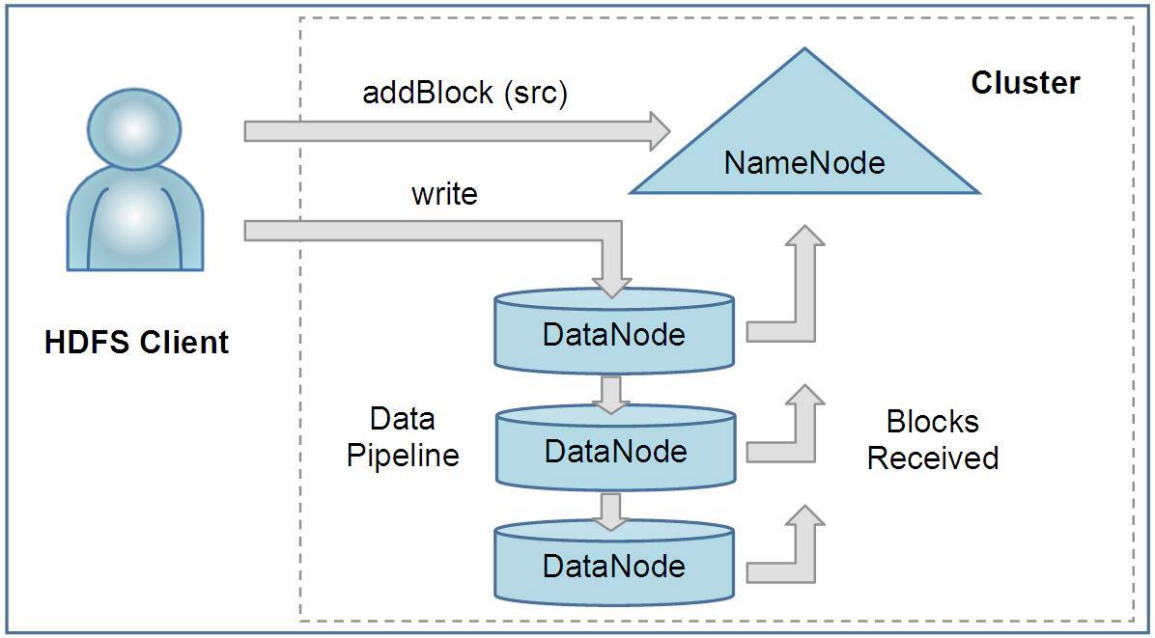
Two plans of Name Node

- NFS
- Standby Name node



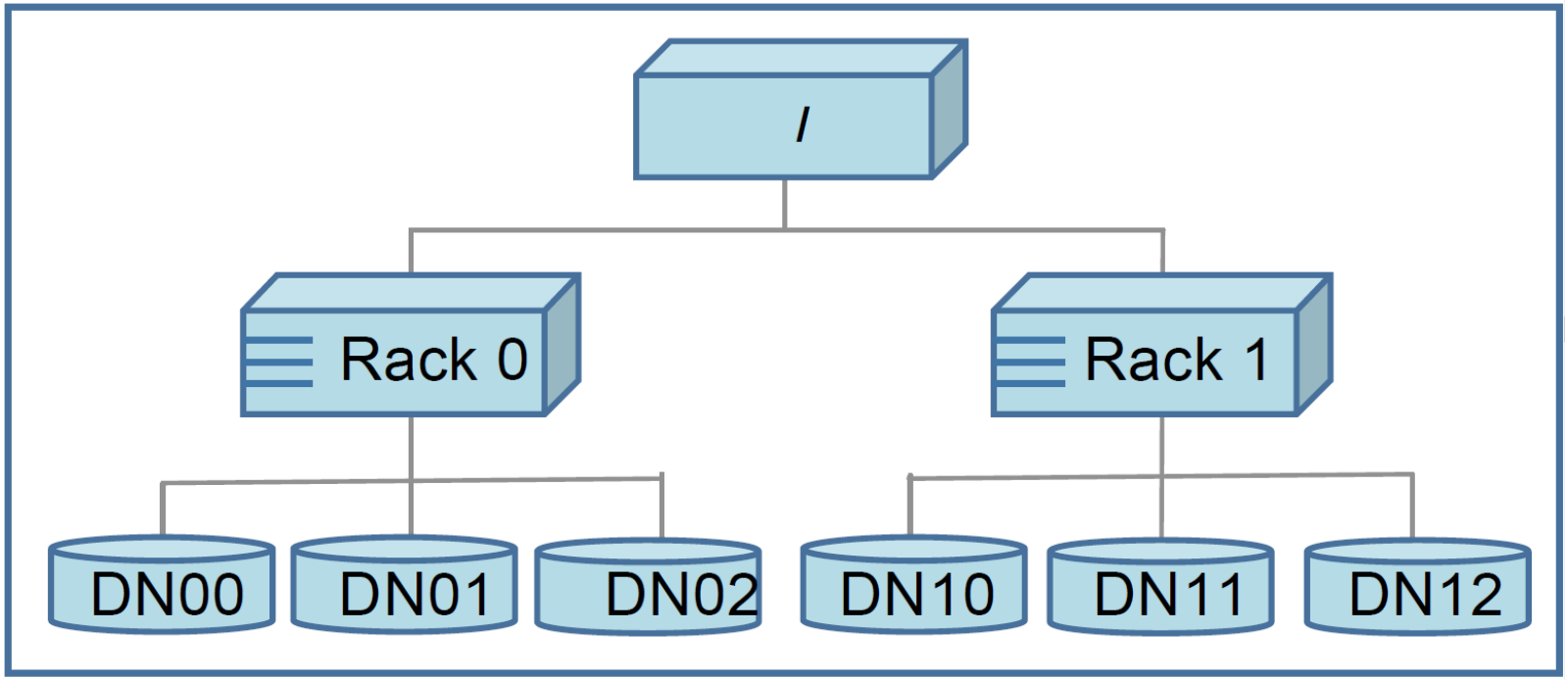
Hadoop Distributed File System

- Composed by NameNodes and DataNodes
- NameNode = header / file pointer in FS
- DataNode = file data in FS
- Write a file:
  - In FS: 1) header -> data blocks
  - In HDFS: 1) record in NameNode -> data blocks in DataNode



Hadoop Distributed File System

- File System: data blocks in one single disk
- HDFS: data blocks in multiple DataNodes contained in server racks across the network



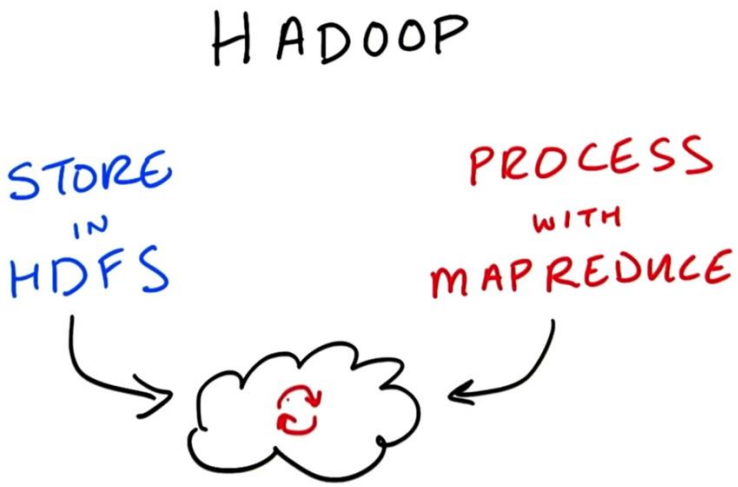
Hadoop Distributed File System

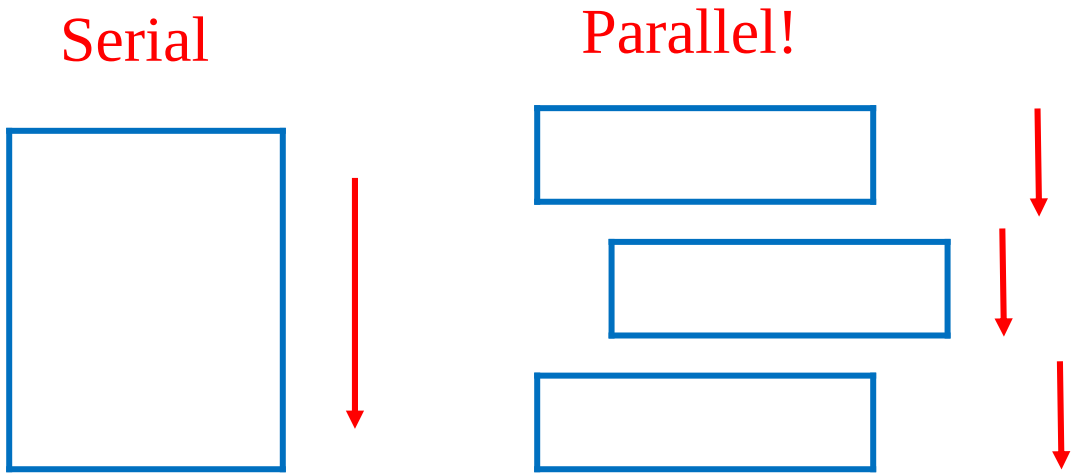
Summary

- One form of distributed file system
- Belongs to Hadoop eco system
- Competitors:
  - GFS
  - Ceph
  - GlusterFS
- Pros and cons: difficult to compare through standards
  - Is Chinese law good or Western law good?

Mapreduce

Mapreduce is designed to by a very parallelized way of managing data, meaning that you input data is split into many pieces, and each piece is processed simultaneously.





Mapreduce

Ledger:

2012-01-01 London Clothes 25.99

2012-01-01 Miami Music 12.15

2012-01-02 NYC Toys 3.10

2012-01-02 Miami Clothes 50.00

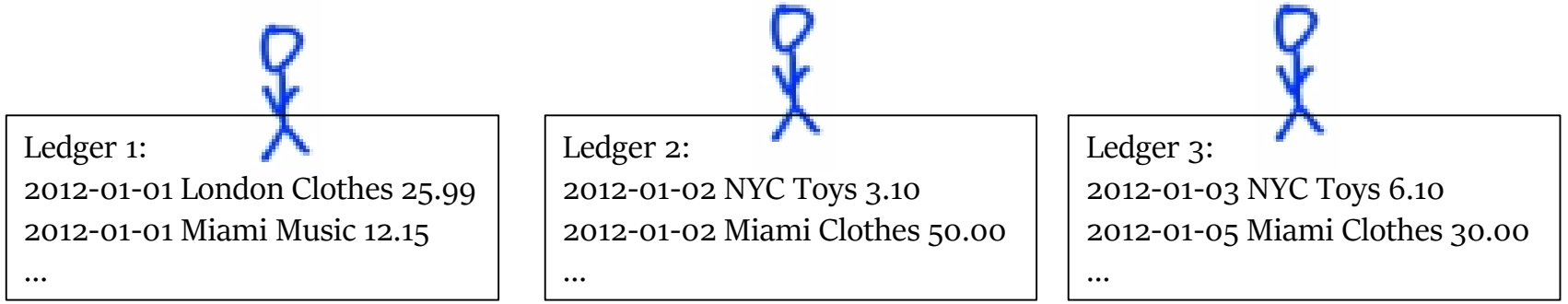
.....

London 25.99

Miami 62.15

NYC 3.10

.....



Key: City Name
Value: Money

Mapreduce

Ledger:

2012-01-01 London Clothes 25.99

2012-01-01 Miami Music 12.15

2012-01-02 NYC Toys 3.10

2012-01-02 Miami Clothes 50.00

.....

London 25.99

Miami 62.15

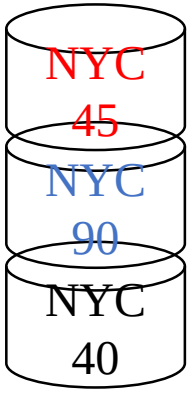
NYC 3.10

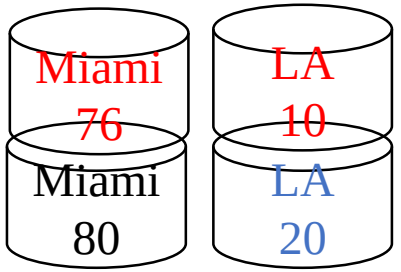
.....

Mappers

Reducers

Reducers

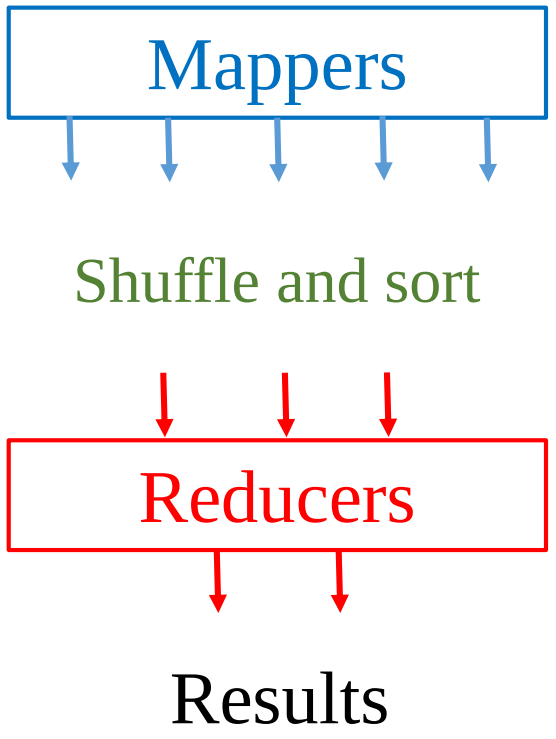




Key: City Name
Value: Money

Mapreduce

- Shuffle is the movement of the intermediate data from the Mappers to the Reducers and the combination of all the small sets of records together.
- Sort is the fact that the Reducers will organize the sets of records
  - For example, the piles of city cards into order
  - Reduce works on one set of records



Intermediate records
(Key, Value)

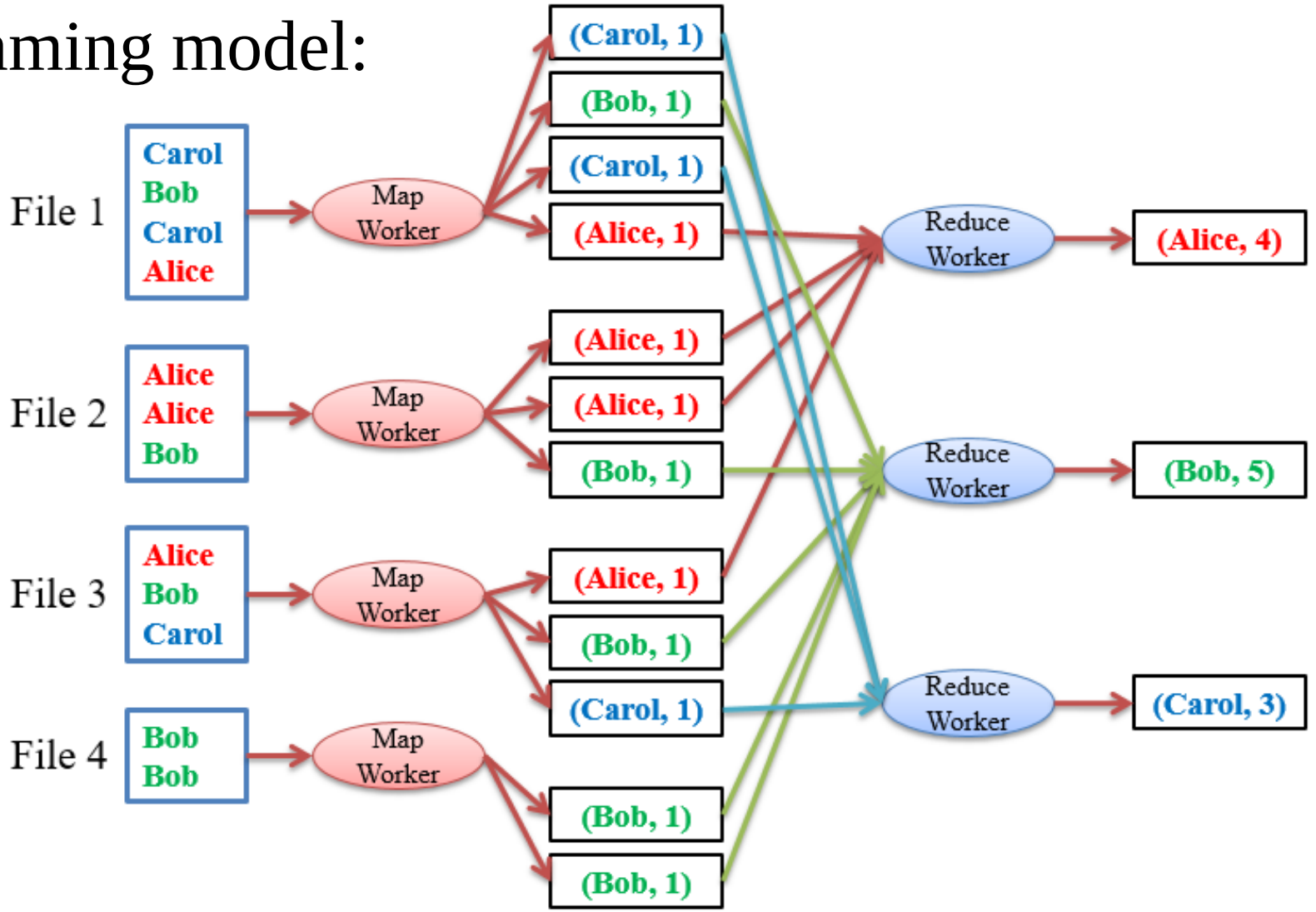
Mapreduce

The programming model:

Word count: read text files and count how often words occur.

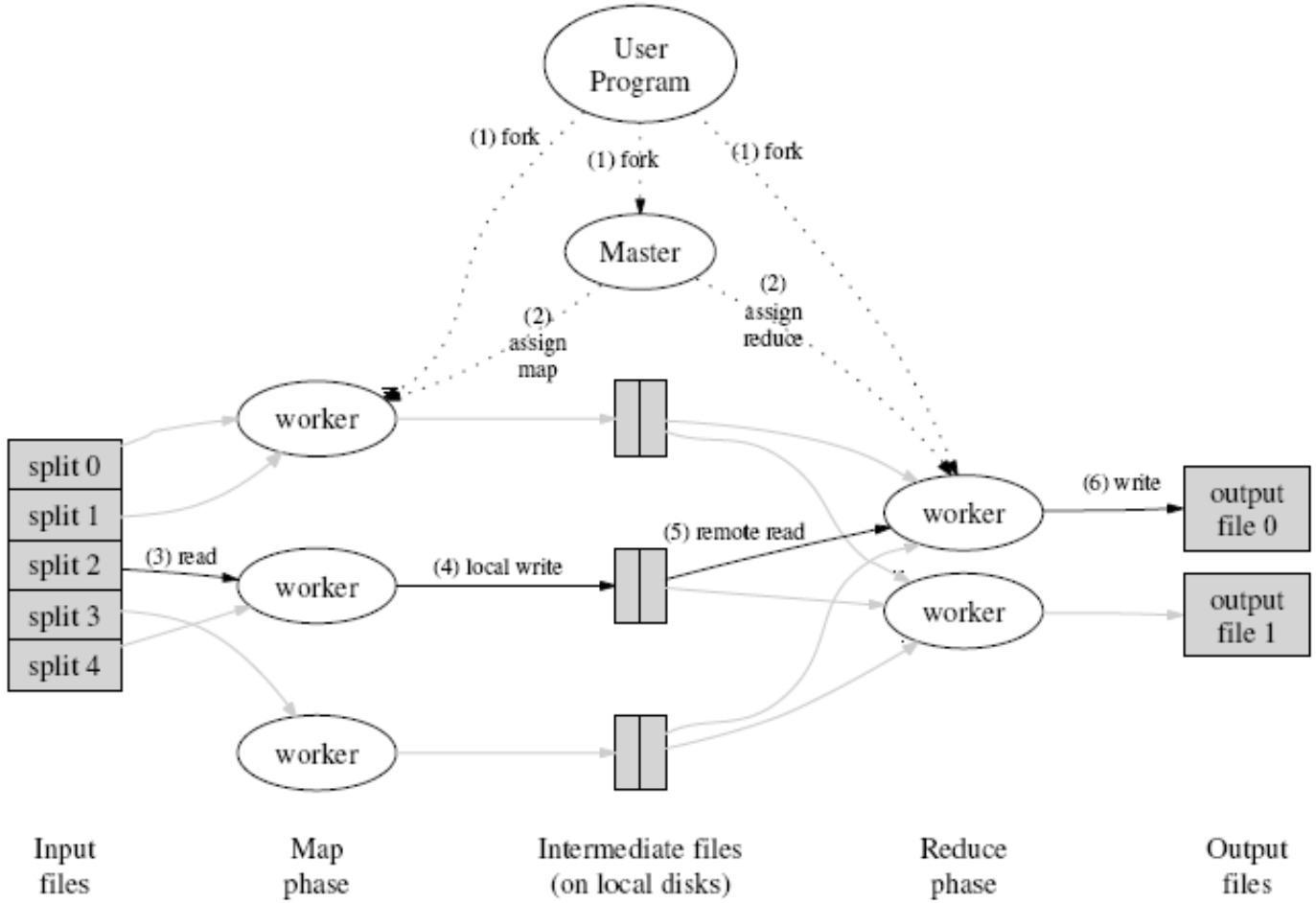
Key?

Value?



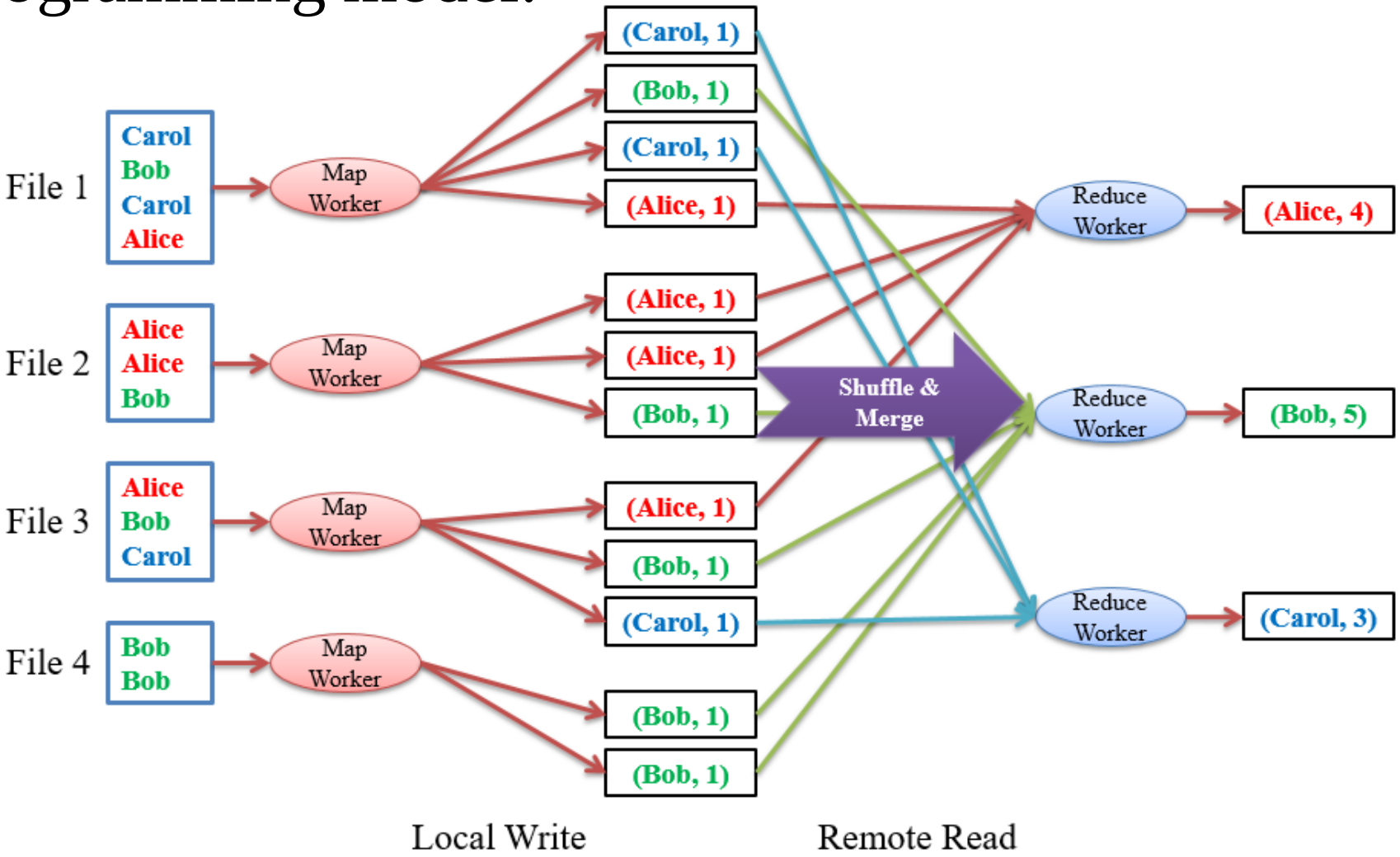
Mapreduce

The programming model:



Mapreduce

The programming model:

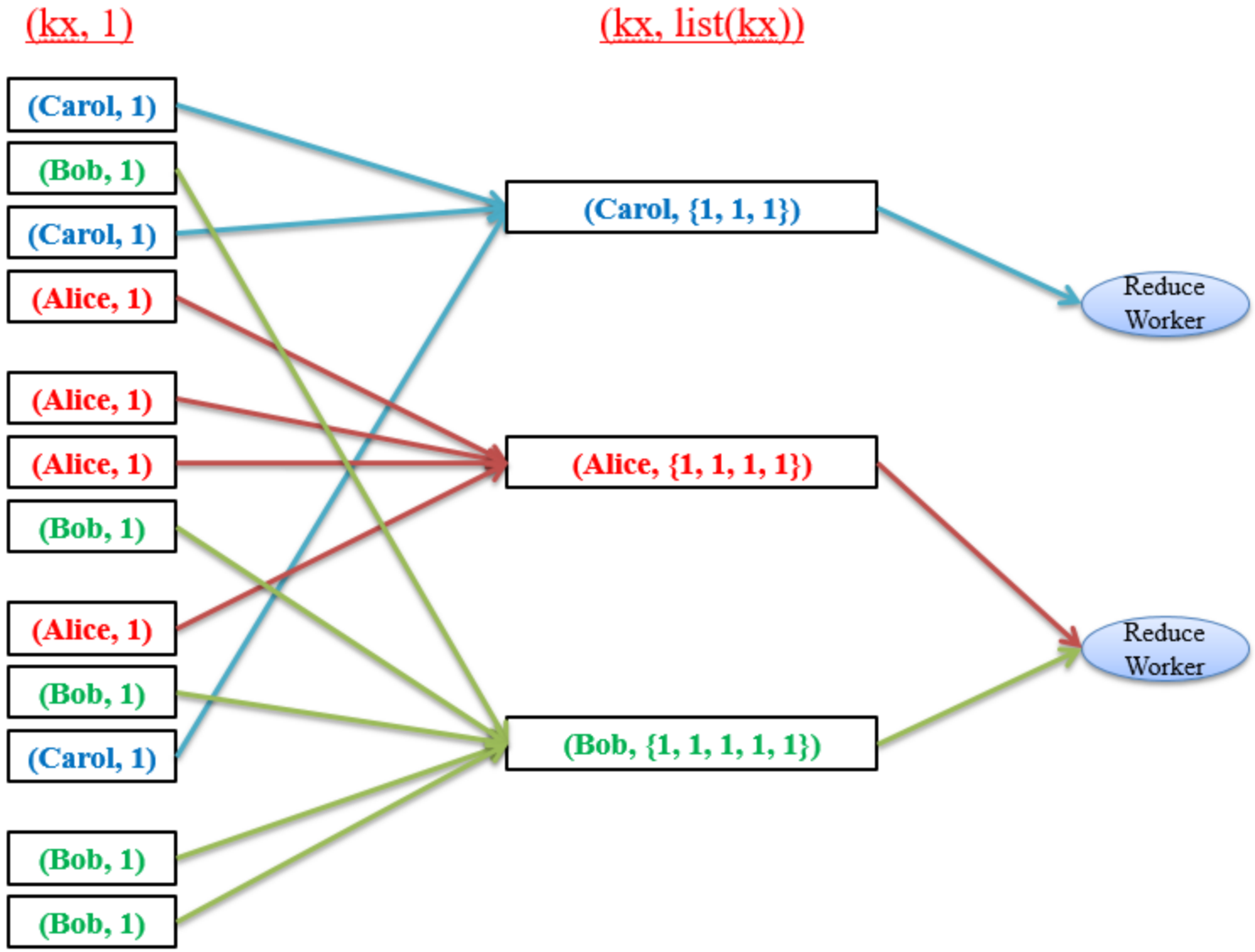


Mapreduce

- Map:
  - (k1, 1) ---> list(k1)--->{1,1,….}
- Reduce:
  - (k1, list(k1)) ---> (k1, n)
- For word count example:
  - (k1, 1): (*word*, 1)
  - list(k1): {(*the 1st occurrence of word in file*, 1),
    (*the 2nd occurrence of word in file*, 1),
    (*the 3rd occurrence of word in file*, 1),…}
  - (k1, list(k1)): (*word*, {1, 1, 1, 1,…})
  - (k1, n): (*word*, sum{1, 1, 1, 1,…})

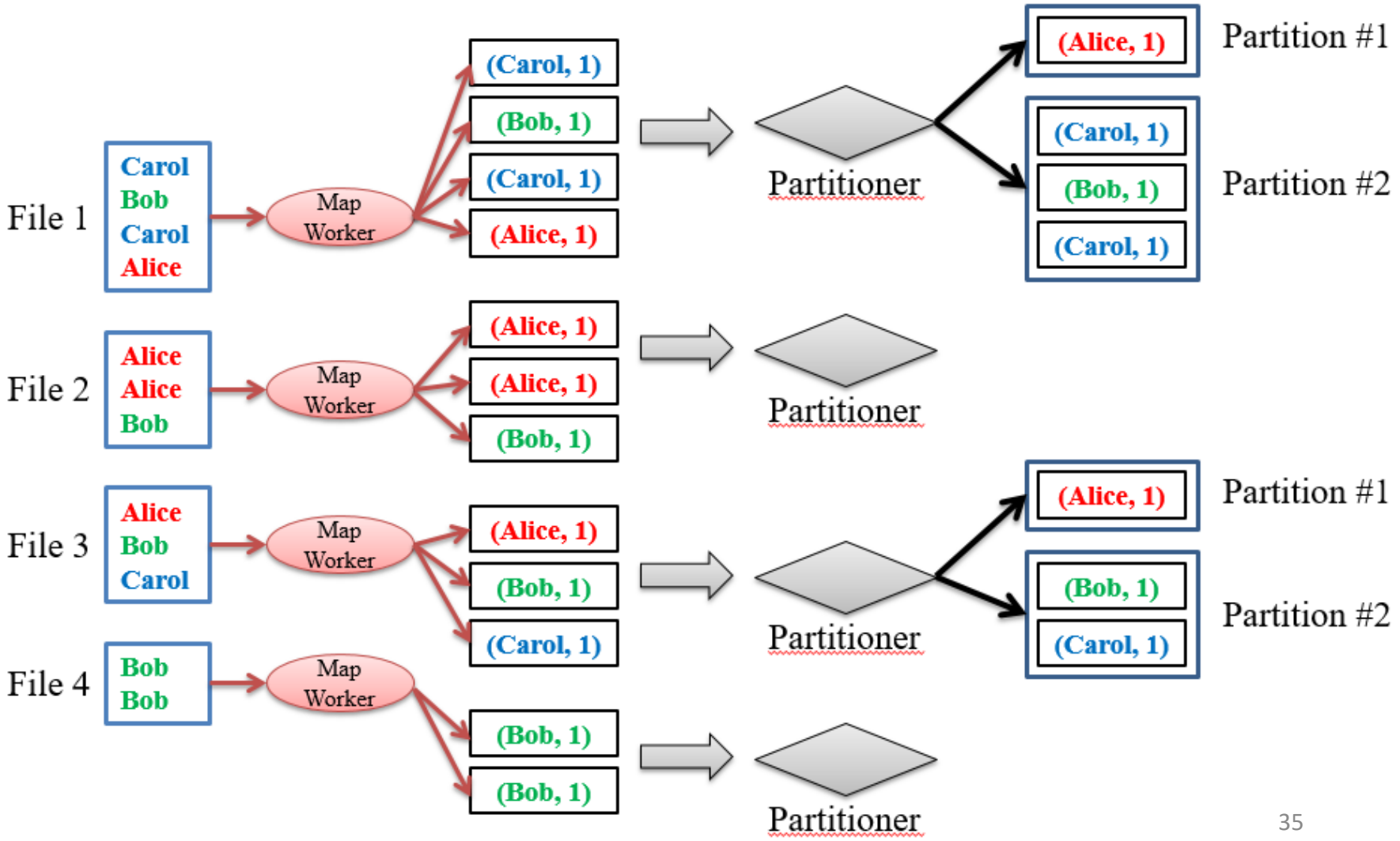
Mapreduce

- Rule No. 1: The same key goes to the same reduce worker



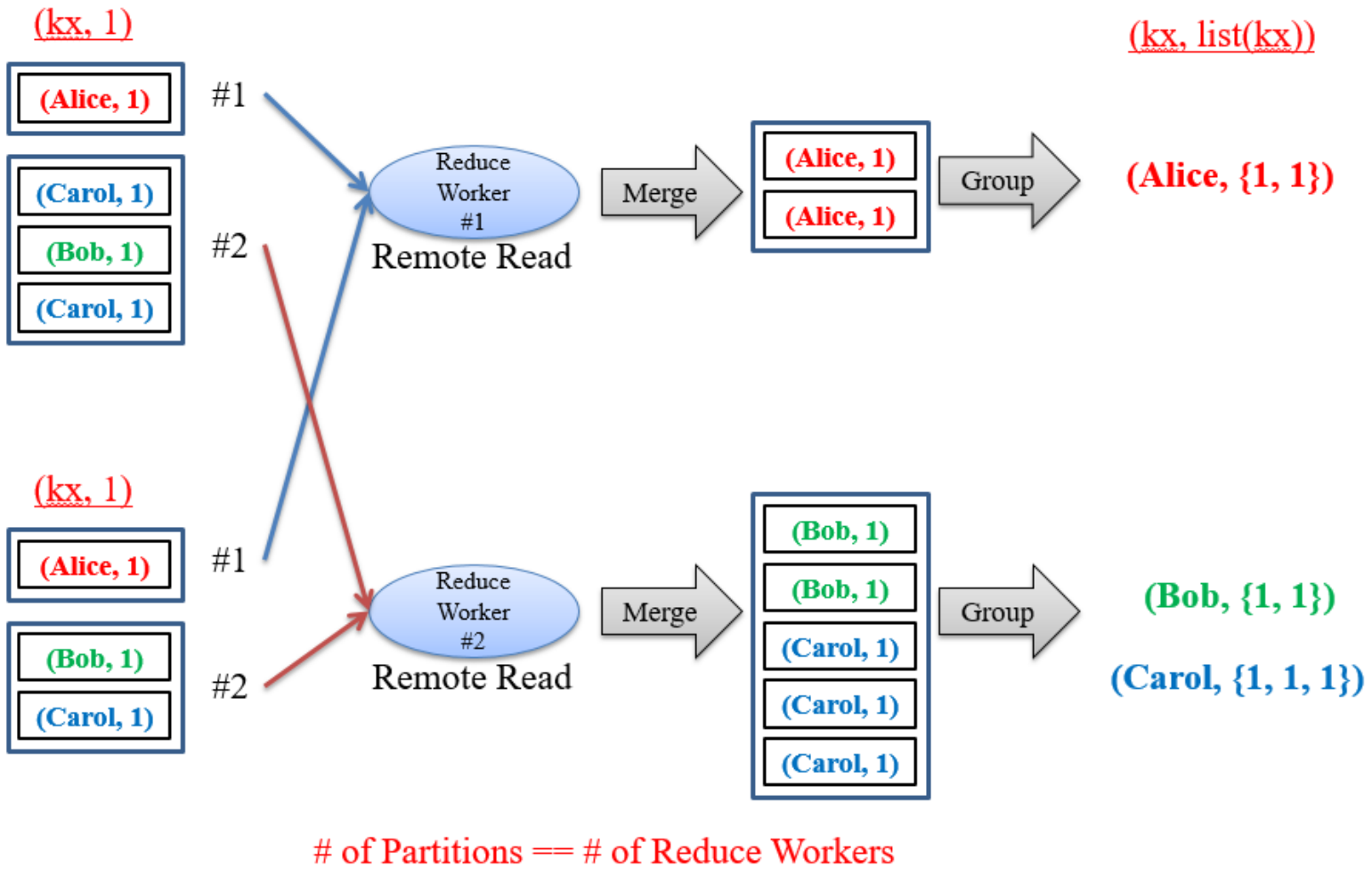
Mapreduce

- How Rule No. 1 is done (1)



Mapreduce

- How Rule No. 1 is done (2)

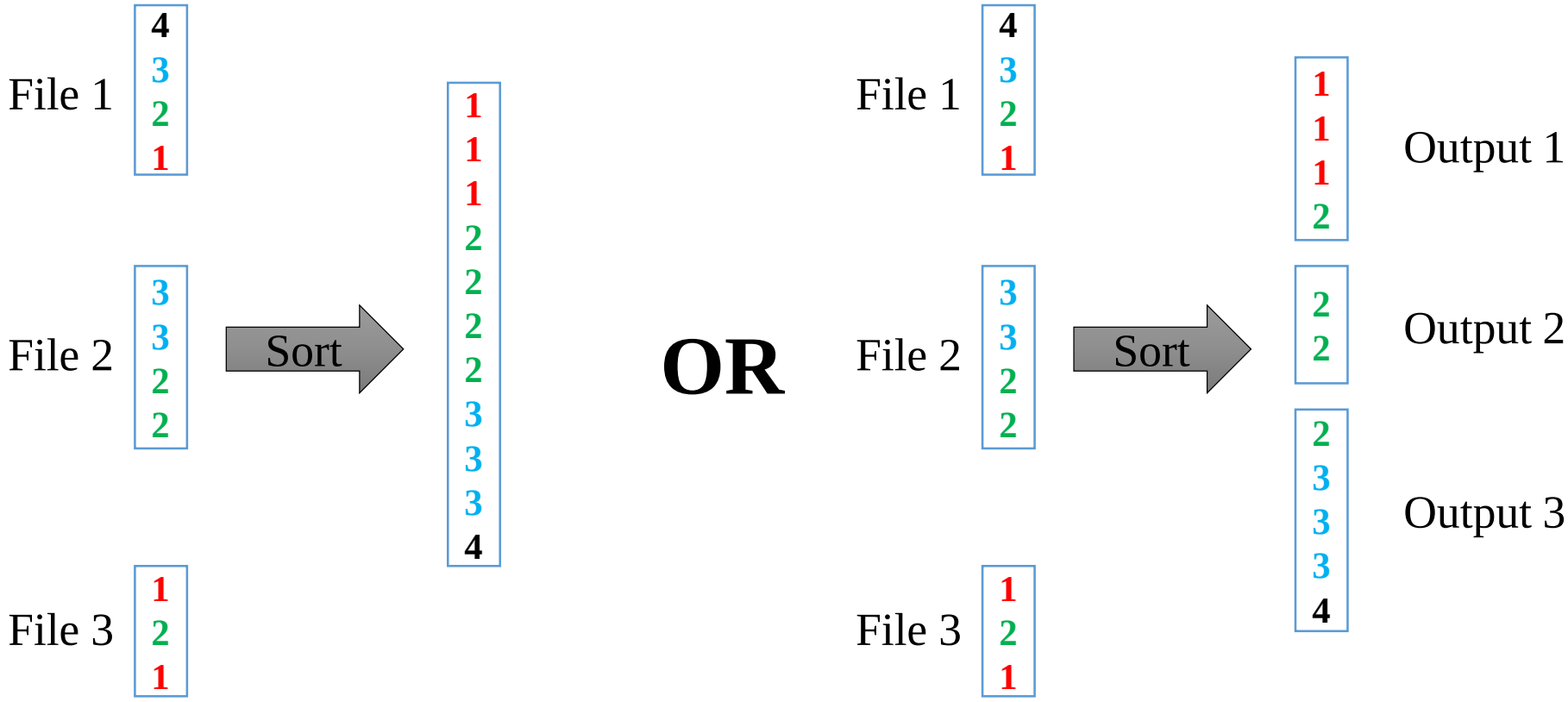


Mapreduce

- Rule No. 2: Keys are sorted on each reduce worker
  - A side effect of the Shuffle and Merge phase
  - Important to advanced tasks such as Sort and Join

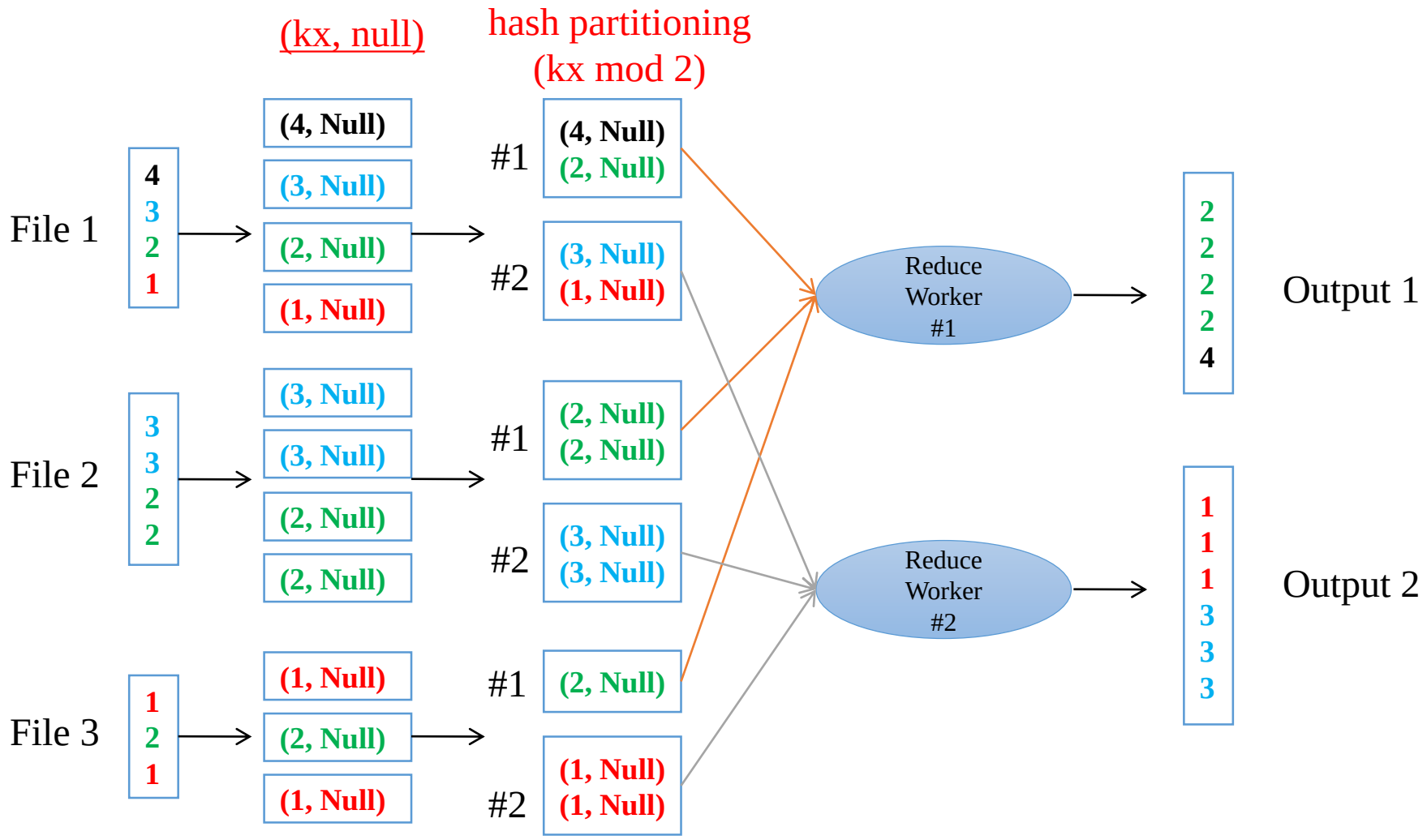
Mapreduce

- Rule No. 2: Keys are sorted on each reduce worker



Mapreduce

- Keys are already sort of sorted (partially sorted)



Mapreduce

- However, to obtain a good balance partitioner is not trivial
- Load balance is crucial since the job execution time is determined by the slowest reduce worker.
- Range split points depend on data distribution.
  - {1, 2, 3, 4, 5, 5, 5, 5, 7, 7, 7, 7}: [1, 5), [5, 7), [7, +Infinity)
  - {1, 1, 1, 2, 3, 4, 5, 7, 8, 8, 9, 9}: [1, 3), [3, 8), [8, +Infinity)
  - ……
- It is feasible to use sampling to estimate the data distribution.

Mapreduce

- Join

Name	Color
Alice	Red
Alice	Green
Bob	Blue

Name	Device
Alice	iPhone
Bob	iPhone
Bob	BlackBerry

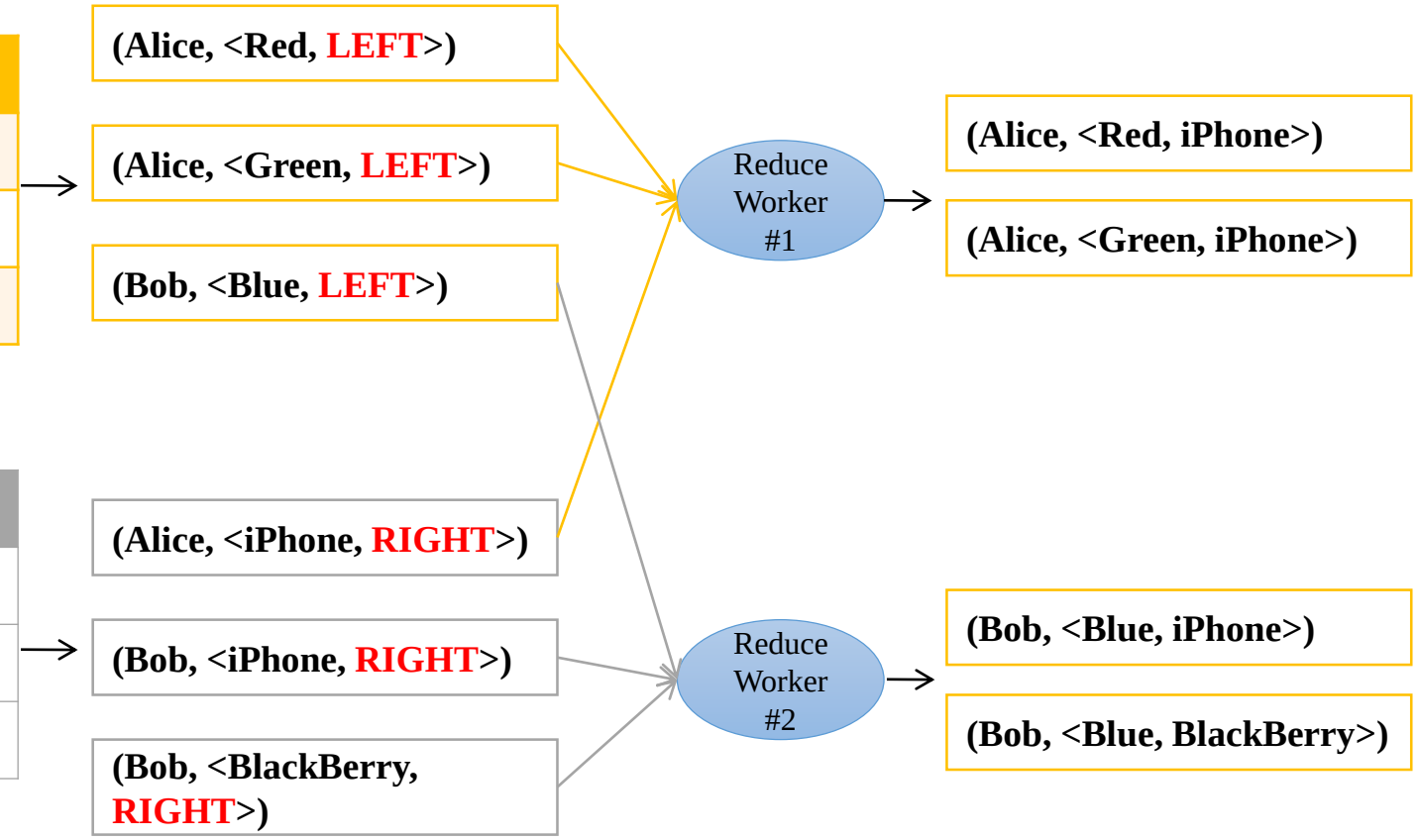
Name	Color	Device
Alice	Red	iPhone
Alice	Green	iPhone
Bob	Blue	iPhone
Bob	Blue	BlackBerry

Mapreduce

- Join

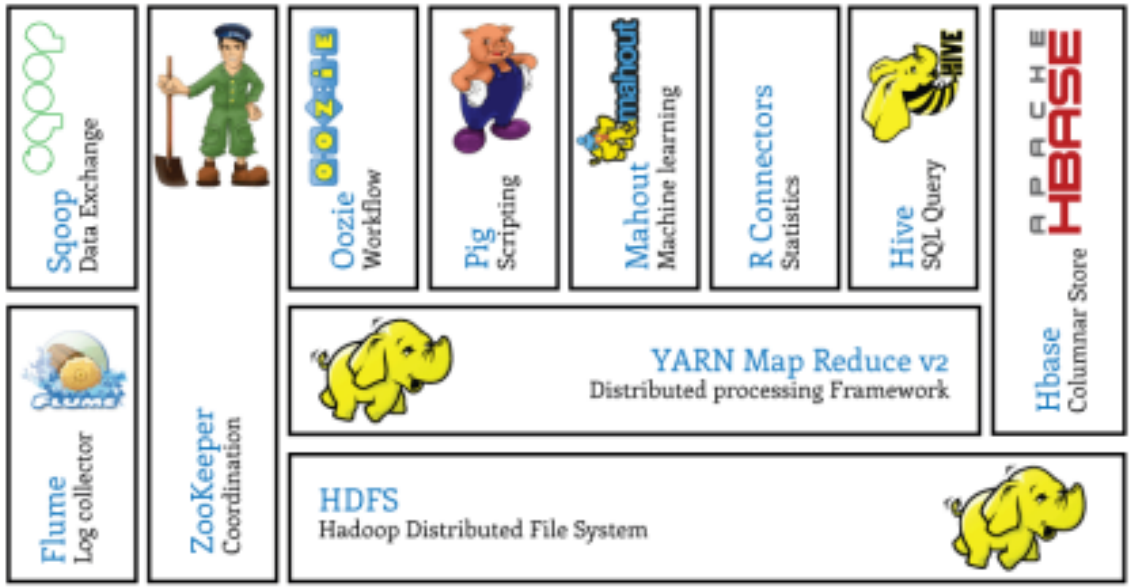
Name	Color
Alice	Red
Alice	Green
Bob	Blue

Name	Device
Alice	iPhone
Bob	iPhone
Bob	BlackBerry

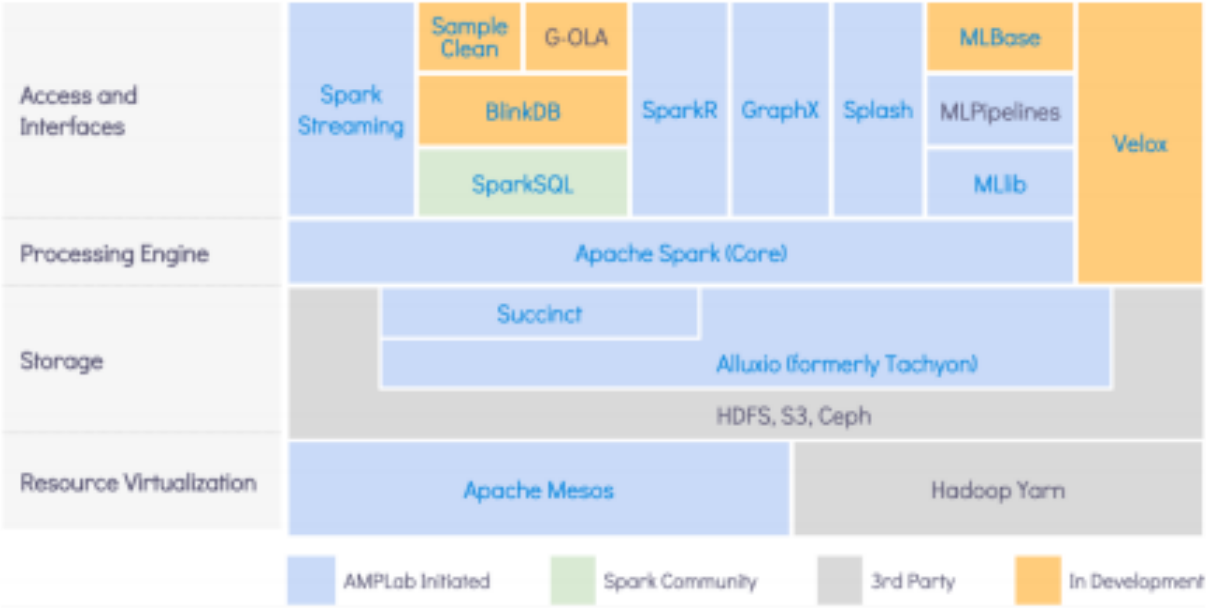


Hadoop Eco-System

Hadoop Eco-System



Spark Eco-System



Hadoop Eco-System

- Hbase:
  - Column Store
  - Column vs row store
- Hive
  - SQL Query
- R Connectors
  - R Programming Language
- Mahout
  - Machine Learning Library
  - Mahout is great / is bad?
- Pig
  - Pig scripting