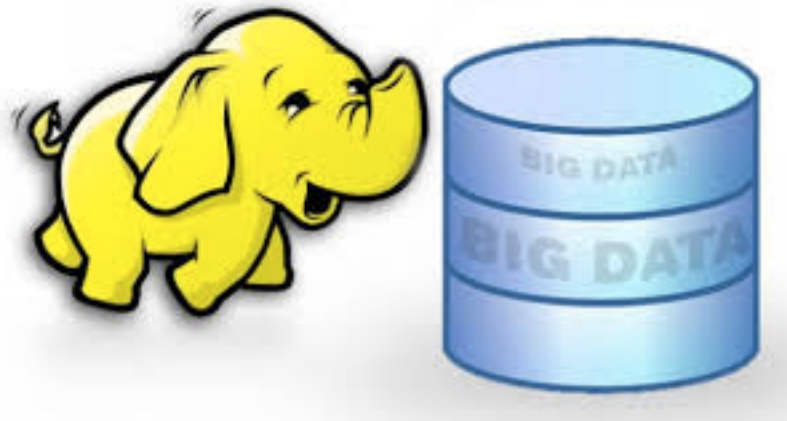


Big Data Processing



CSE 40437/60437-Spring 2015

Prof. Dong Wang

Key Questions to Answer

- Why Hadoop?
- What is Hadoop?
- How to Hadoop?
- Examples of Hadoop

- Why is Hadoop ?

What is Big Data

- A bunch of data?
- A technical term?
- An industry?
- A trend?
- A set of skills?
- ...

What is your interpretation of BIG DATA?

What is Big Data

- Wikipedia big data
 - An all-encompassing term for any collection of data sets so **large** and **complex** that it becomes **difficult** to process using on-hand data management tools or traditional data processing applications.

How big is big?

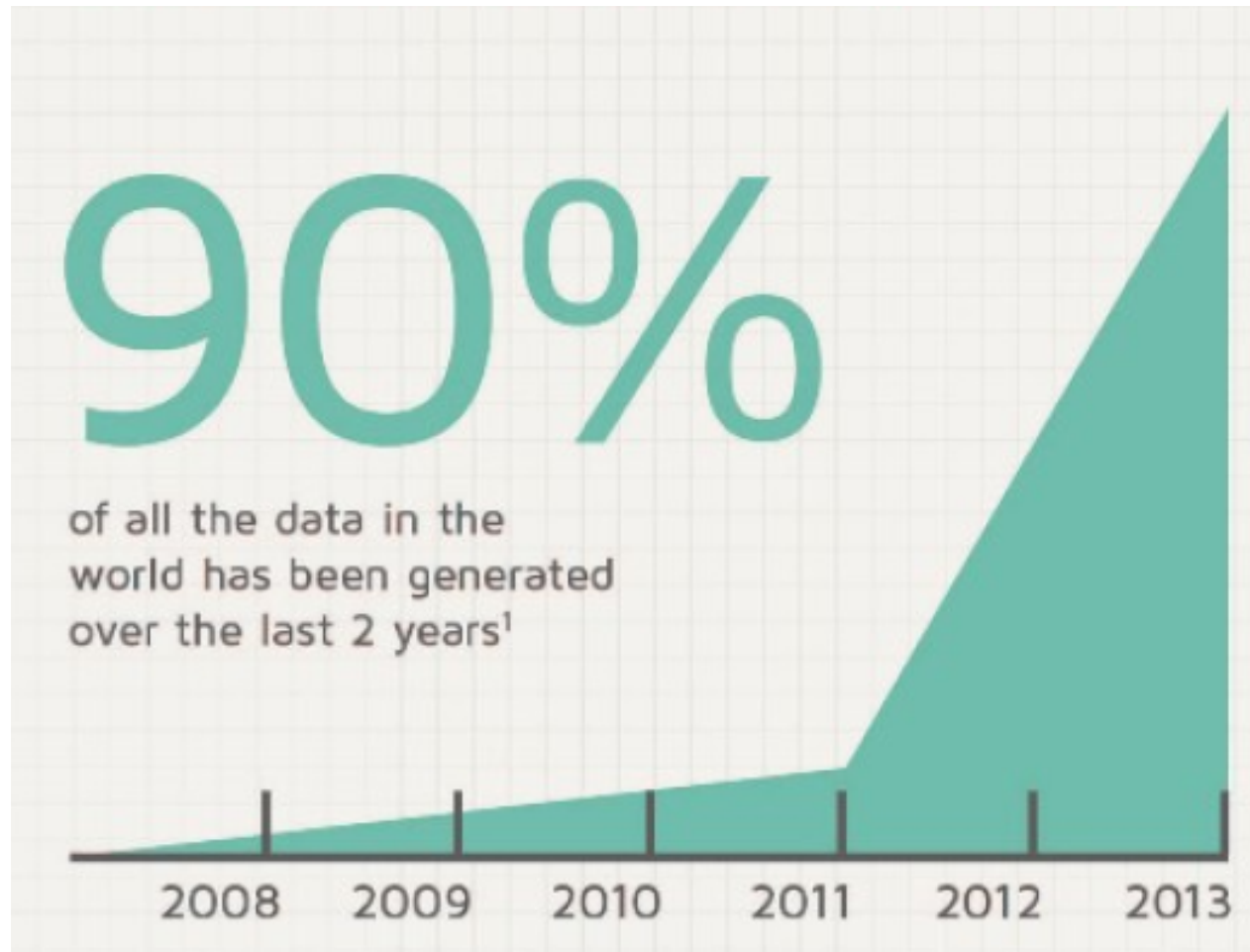
- 2008: Google processes 20 PB a day
- 2009: Facebook has 2.5 PB user data + 15 TB / day
- 2011: Yahoo! has 180-200 PB of data
- 2012: Facebook ingests 500 TB/day
- 2013: YouTube 1000 PB video storage; 4 billion views/day

1 PB = 10^3 TB = 10^6 GB = 10^{15} B

1 Exabyte = 1000 PB

Zettabyte, Yottabyte ...

The percentage of all data in the world that has been generated in **last 2 years**?



Who is generating Big Data?

Who is generating Big Data?

Social



User Tracking & Engagement



Homeland Security



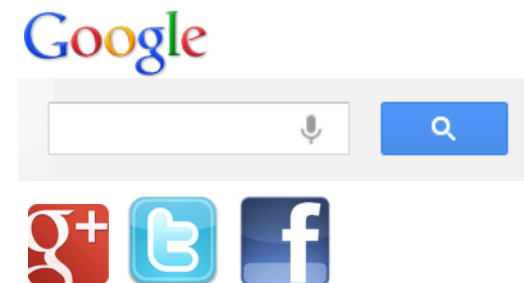
eCommerce



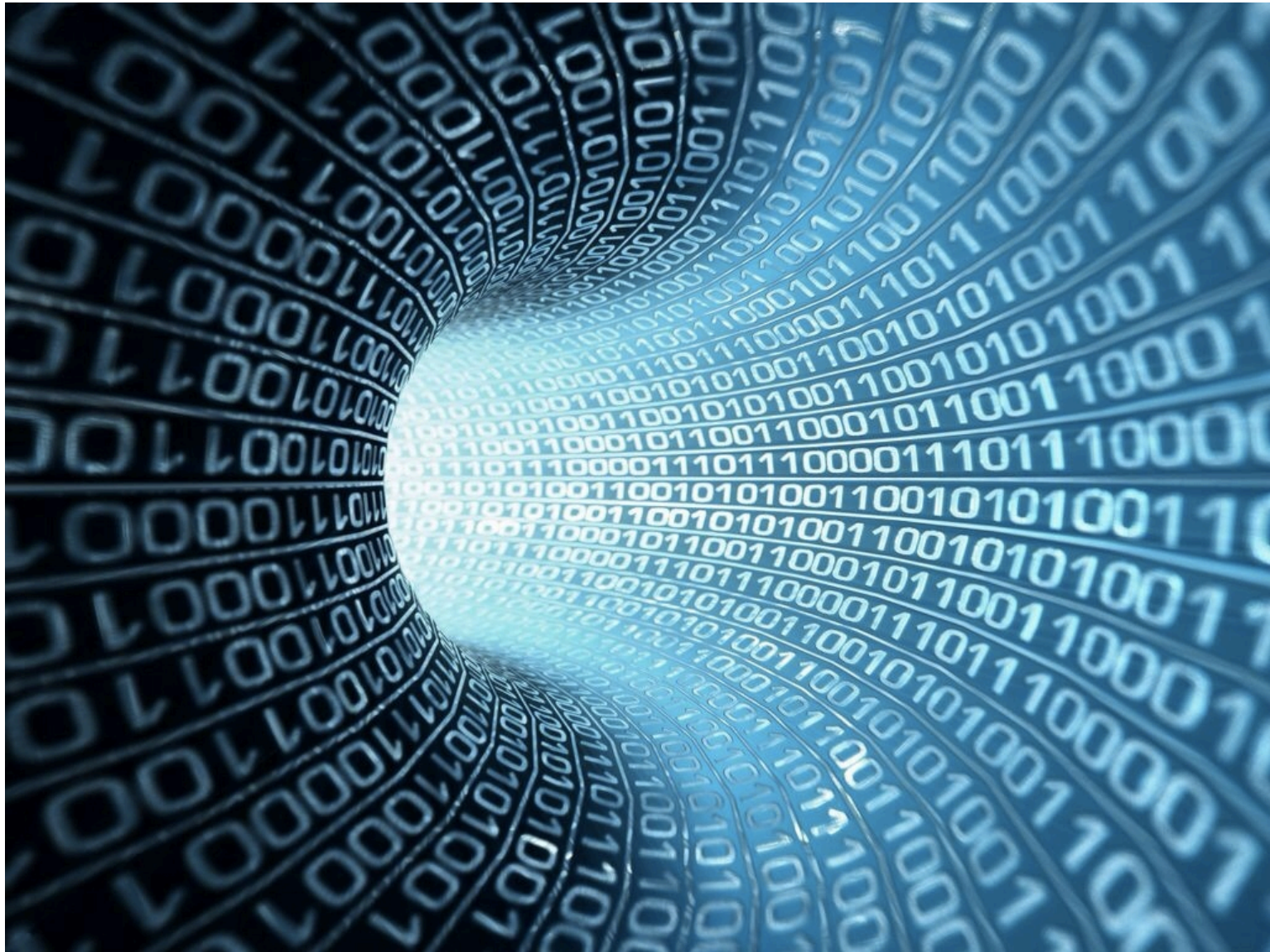
Financial Services



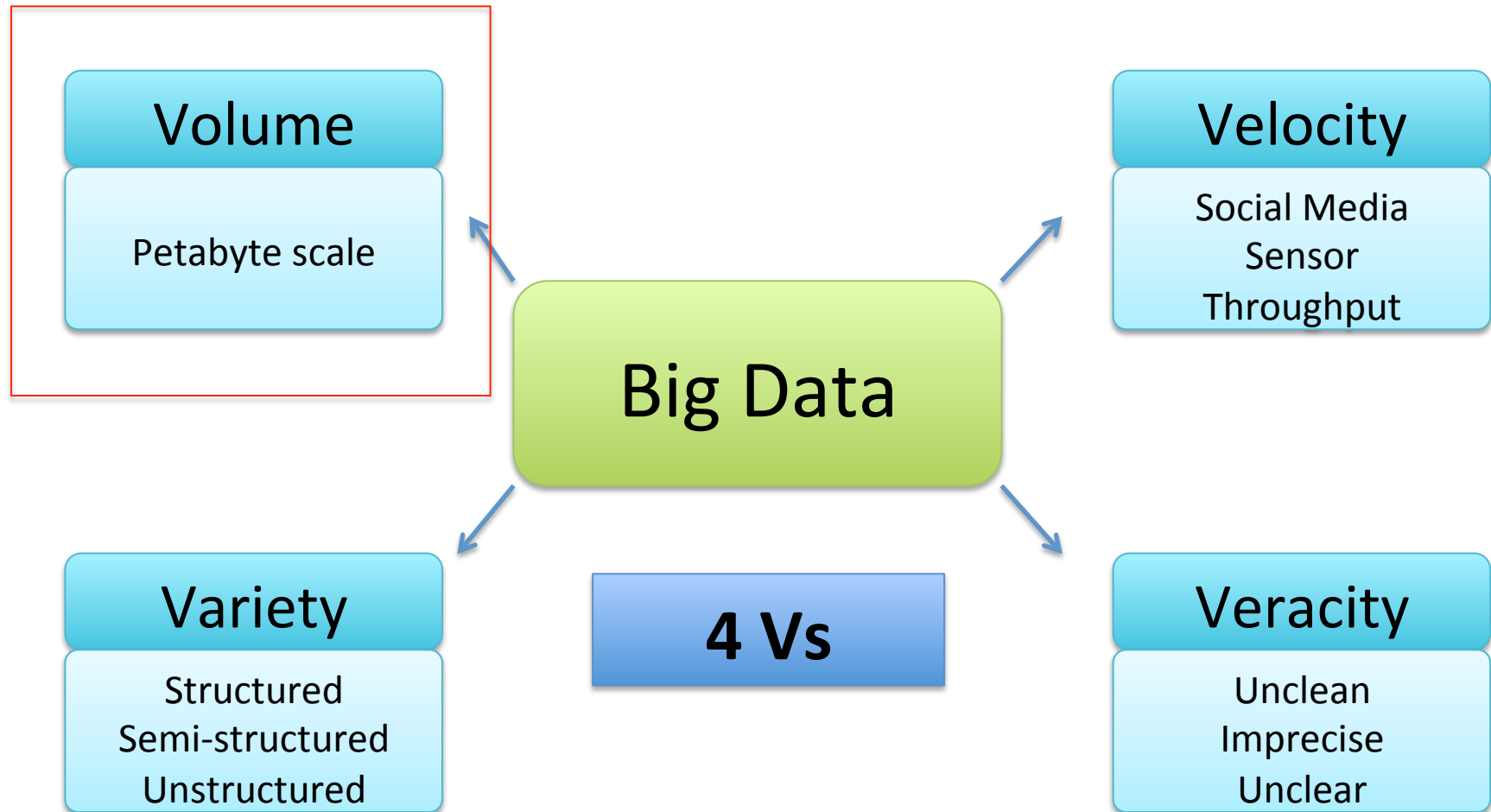
Real Time Search



That is a lot of data ...

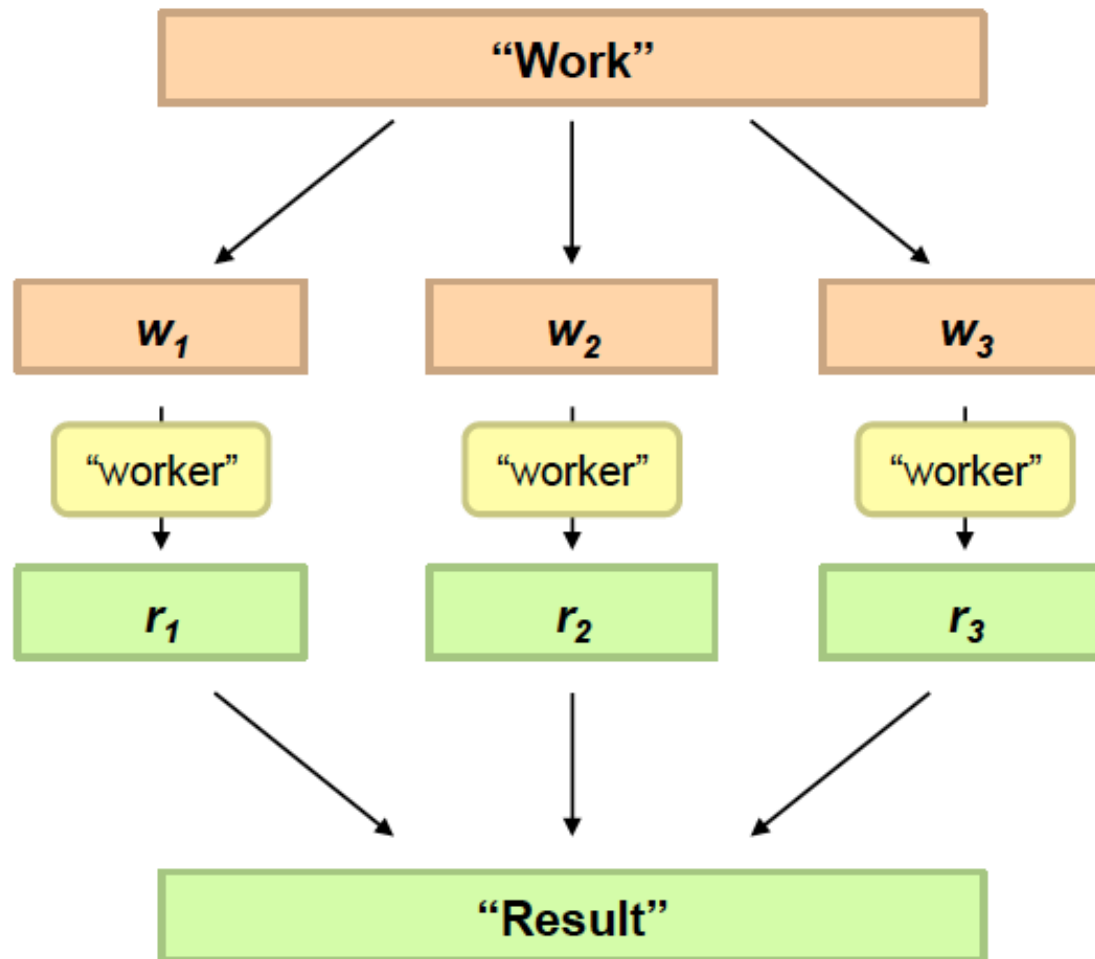


What are Key Features of Big Data?



- Philosophy to Scale for Big Data?

Divide and Conquer



Divide Work



Combine Results

Distributed processing is non-trivial

- How to assign tasks to different workers in an efficient way?
- What happens if tasks fail?
- How do workers exchange results?
- How to synchronize distributed tasks allocated to different workers?



Image courtesy of Master isolated images at FreeDigitalPhotos.net

Big data storage is challenging

- Data Volumes are massive
- Reliability of Storing PBs of data is challenging
- All kinds of failures: Disk/Hardware/Network Failures
- Probability of failures simply increase with the number of machines ...



One popular solution: Hadoop



Hadoop Cluster at Yahoo! (Credit: Yahoo)

Hadoop offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination



Hadoop offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination



Programmers

*No longer need to
worry about*



**Q: Where file is
located?**

**Q: How to handle
failures & data lost?**

**Q: How to divide
computation?**

**Q: How to program
for scaling?**

A real world example of New York Times

- **Goal:** Make entire archive of articles available online: 11 million, from 1851
- **Task:** Translate 4 TB TIFF images to PDF files
- **Solution:** Used Amazon Elastic Compute Cloud (EC2) and Simple Storage System (S3)

- **Time: ?**



- **Costs: ?**



The
New York
Times

A real world example of New York Times

- **Goal:** Make entire archive of articles available online: 11 million, from 1851
- **Task:** Translate 4 TB TIFF images to PDF files
- **Solution:** Used Amazon Elastic Compute Cloud (EC2) and Simple Storage System (S3)
- **Time:** < 24 hours
- **Costs:** \$240

The
New York
Times

So we are



A little history on Hadoop

- Hadoop is an open-source implementation based on **Google File System** (GFS) and **MapReduce** from Google
- Hadoop was created by **Doug Cutting** and **Mike Cafarella** in 2005
- Hadoop was donated to **Apache** in 2006

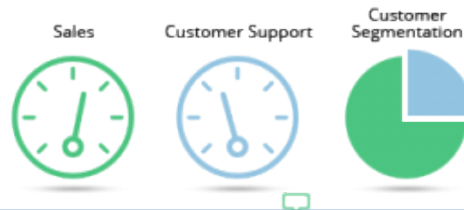


Who are using Hadoop?

Social



User Tracking & Engagement



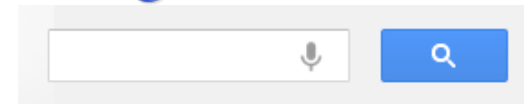
Homeland Security



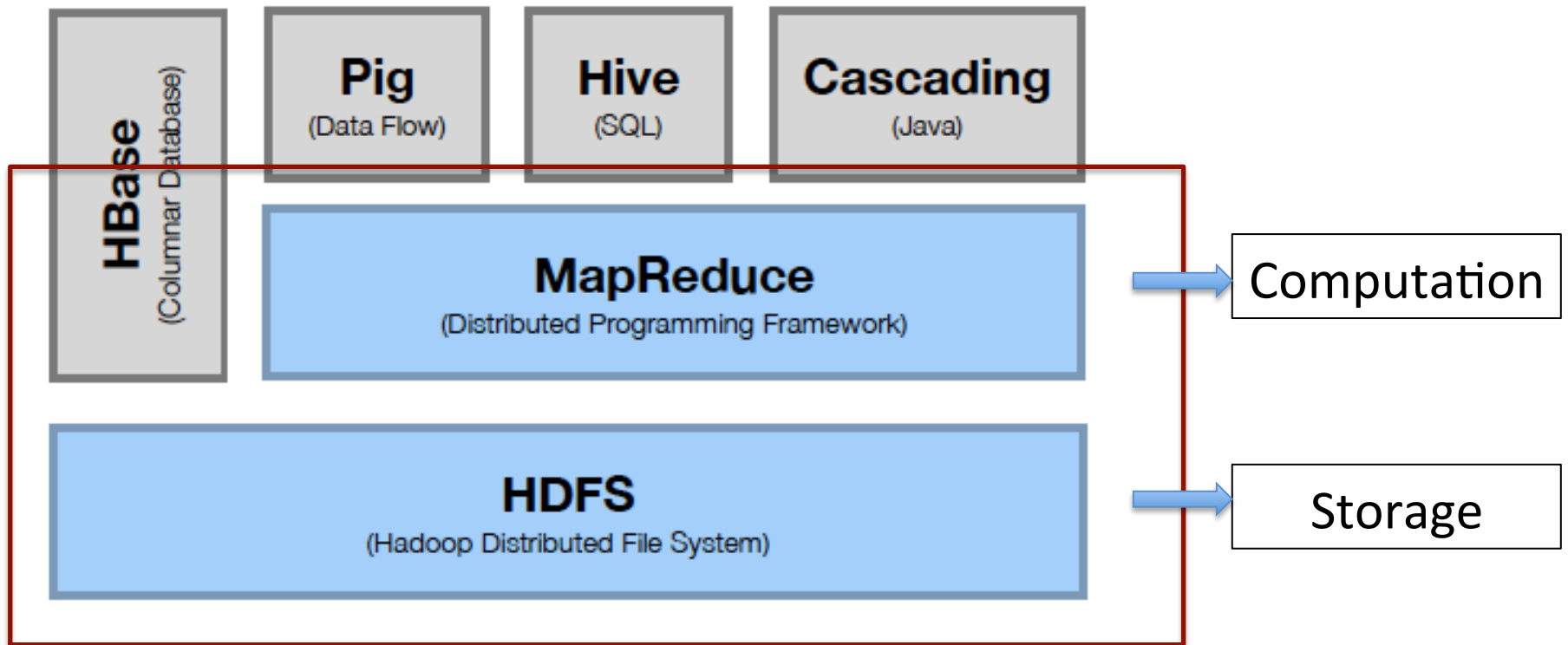
“By 2015, 50% of Enterprise data will be processed by Hadoop” –Yahoo!



Google



Hadoop Stack



Hadoop Resources

- Hadoop at ND:

<http://ccl.cse.nd.edu/operations/hadoop/>

- Apache Hadoop Documentation:

<http://hadoop.apache.org/docs/current/>

- Data Intensive Text Processing with Map-Reduce

<http://lintool.github.io/MapReduceAlgorithms/>

- Hadoop Definitive Guide:

<http://www.amazon.com/Hadoop-Definitive-Guide-Tom-White/dp/1449311520>

HDFS

Hadoop Distributed File System

HDFS Outline

- Motivation
- Architecture and Concepts
- Inside
- User Interface

Motivation Questions

- **Problem 1:** Data is too big to store on one machine.
- **HDFS:** Store the data on multiple machines!

Motivation Questions

- **Problem 2:** Very high end machines are too expensive
- **HDFS:** Run on commodity hardware!

Motivation Questions

- **Problem 3:** Commodity hardware will fail!
- **HDFS:** Software is intelligent enough to handle hardware failure!

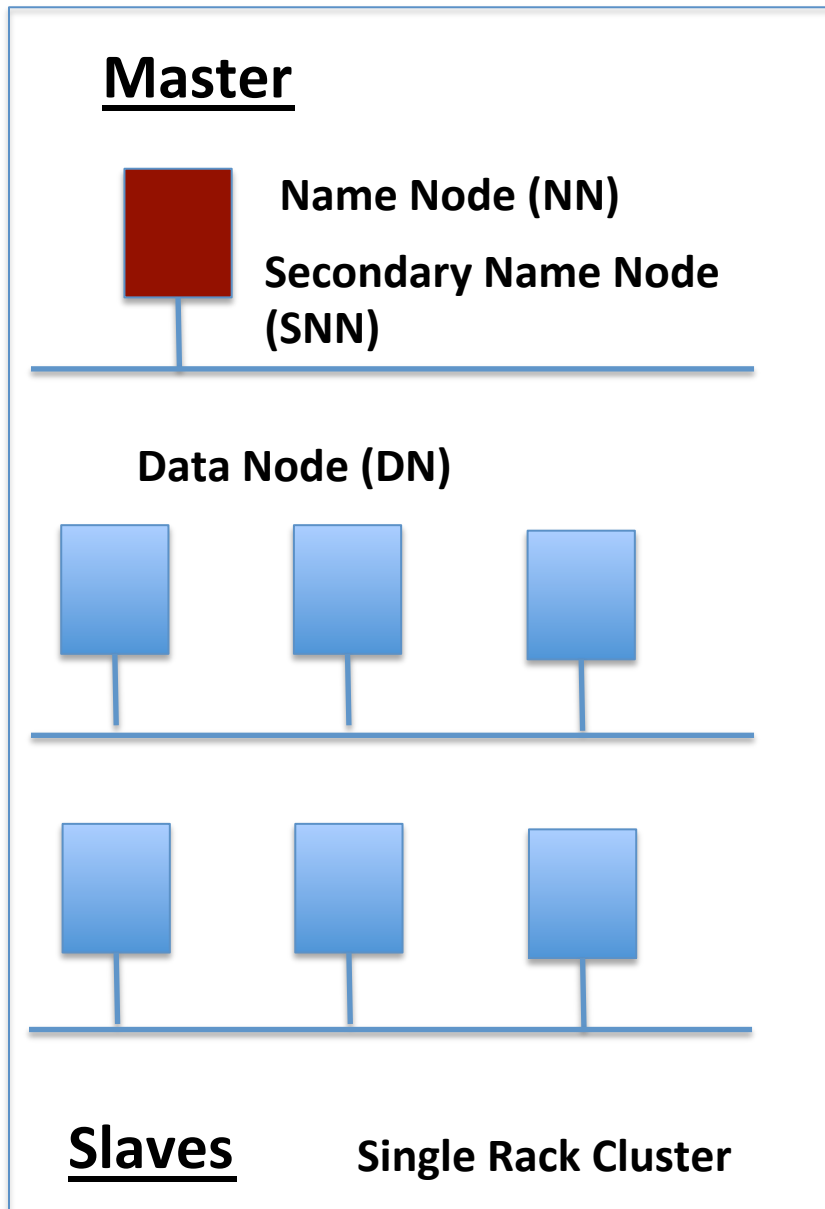
Motivation Questions

- **Problem 4:** What happens to the data if the machine stores the data fails?
- **HDFS:** Replicate the data!

Motivation Questions

- **Problem 5:** How can distributed machines organize the data in a coordinated way?
- **HDFS: Master-Slave Architecture!**

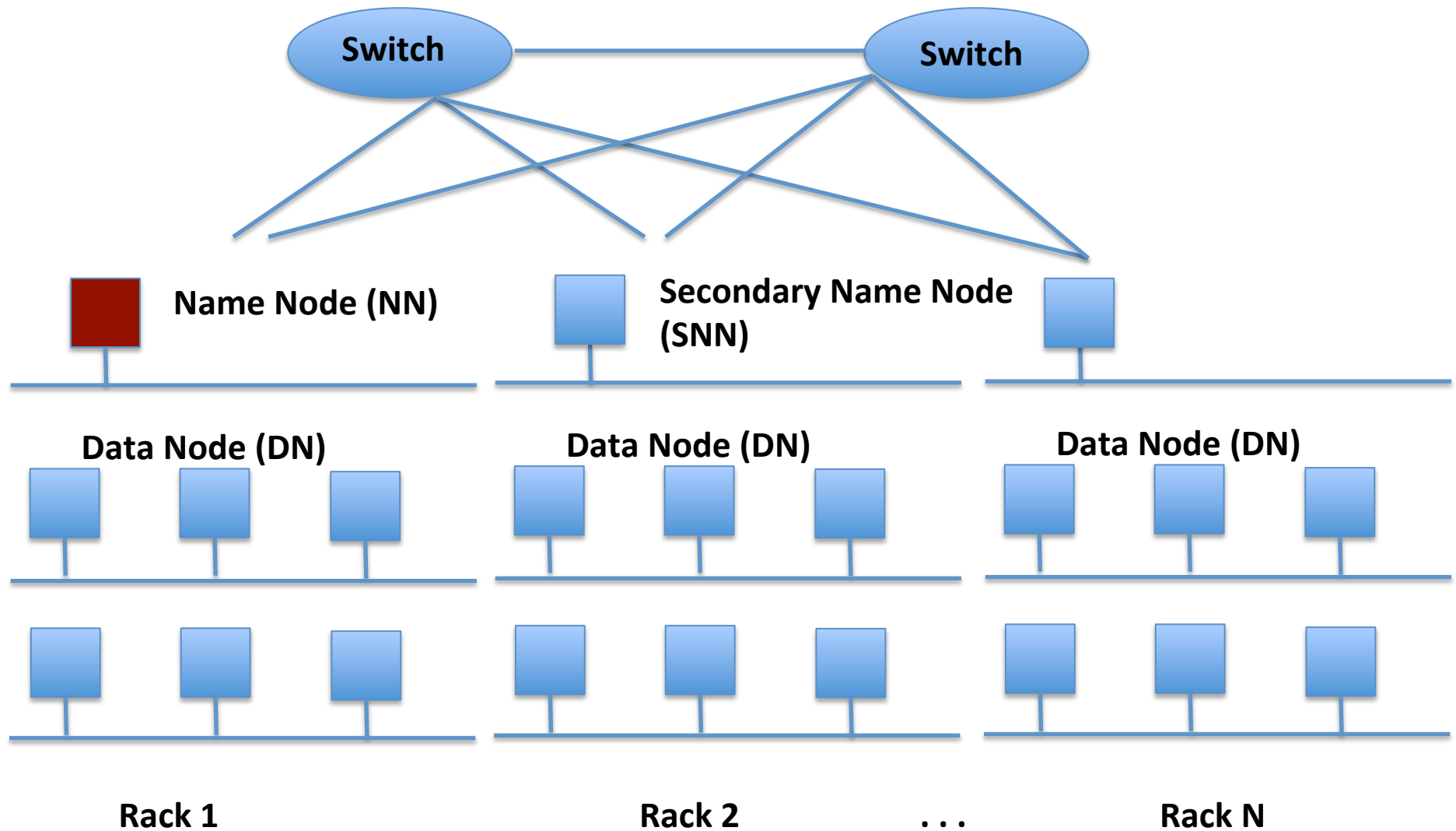
HDFS Architecture: Master-Slave



- **Name Node: Controller**
 - File System Name Space Management
 - Block Mappings
- **Data Node: Work Horses**
 - Block Operations
 - Replication
- **Secondary Name Node:**
 - Checkpoint node

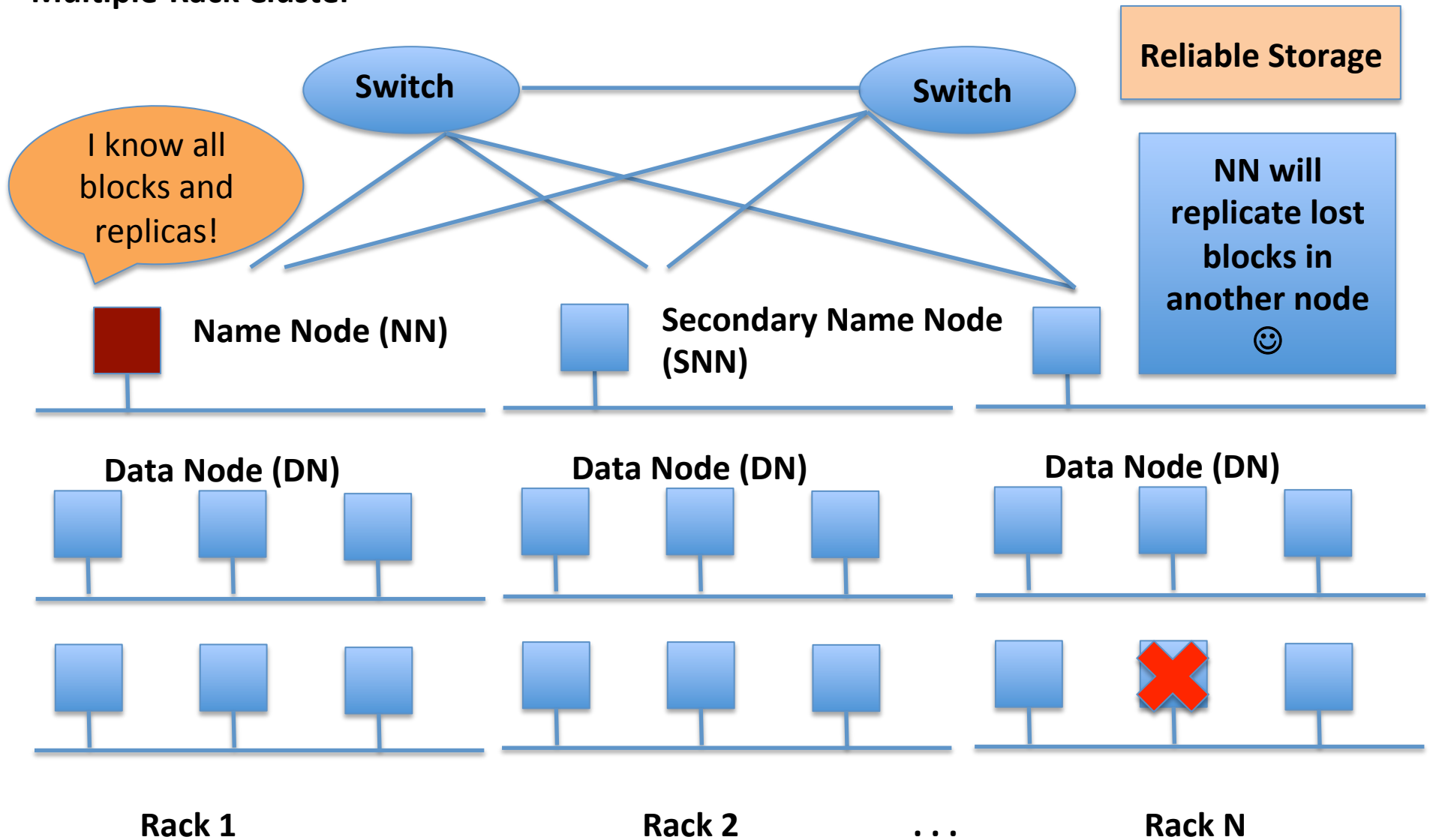
HDFS Architecture: Master-Slave

Multiple-Rack Cluster



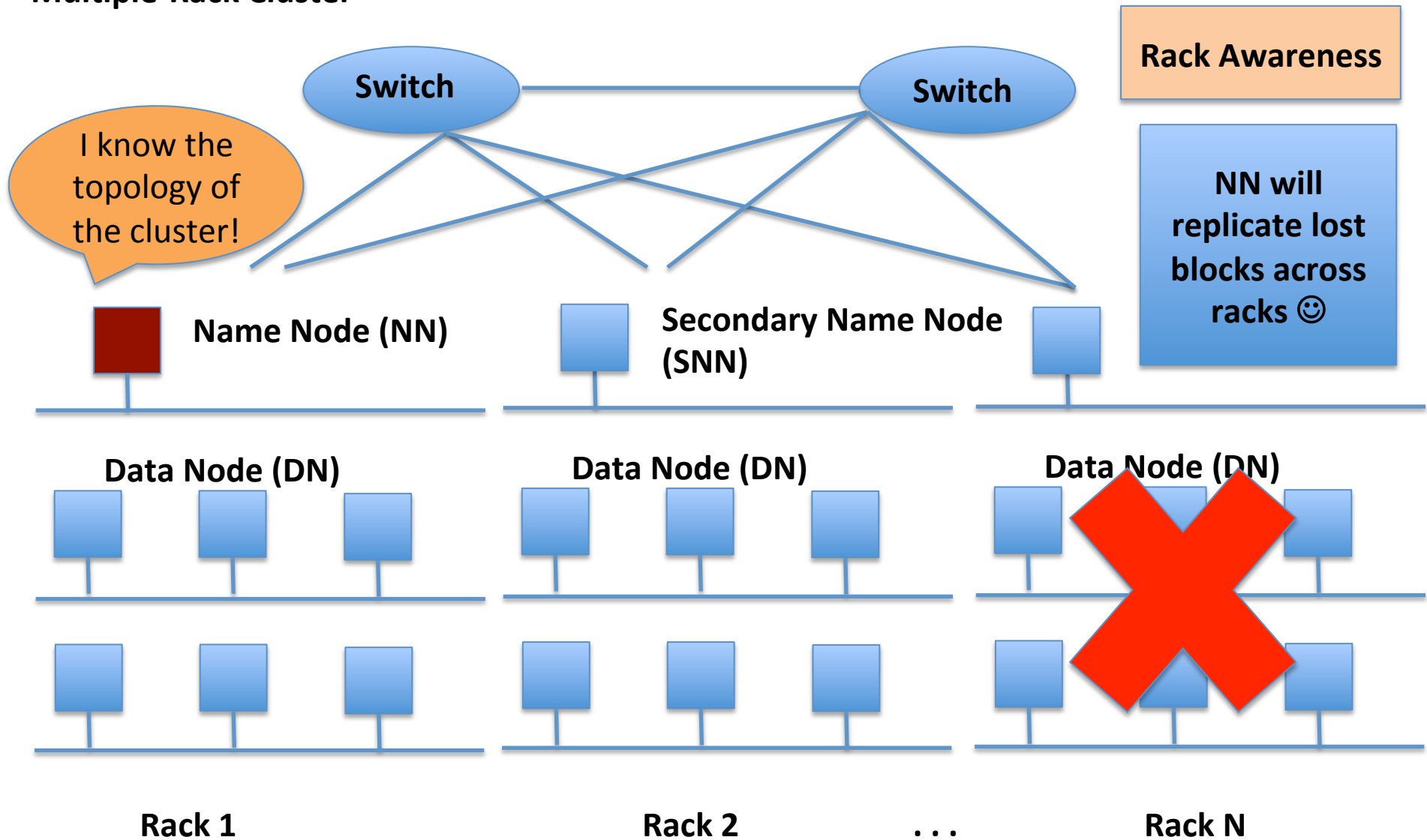
HDFS Architecture: Master-Slave

Multiple-Rack Cluster



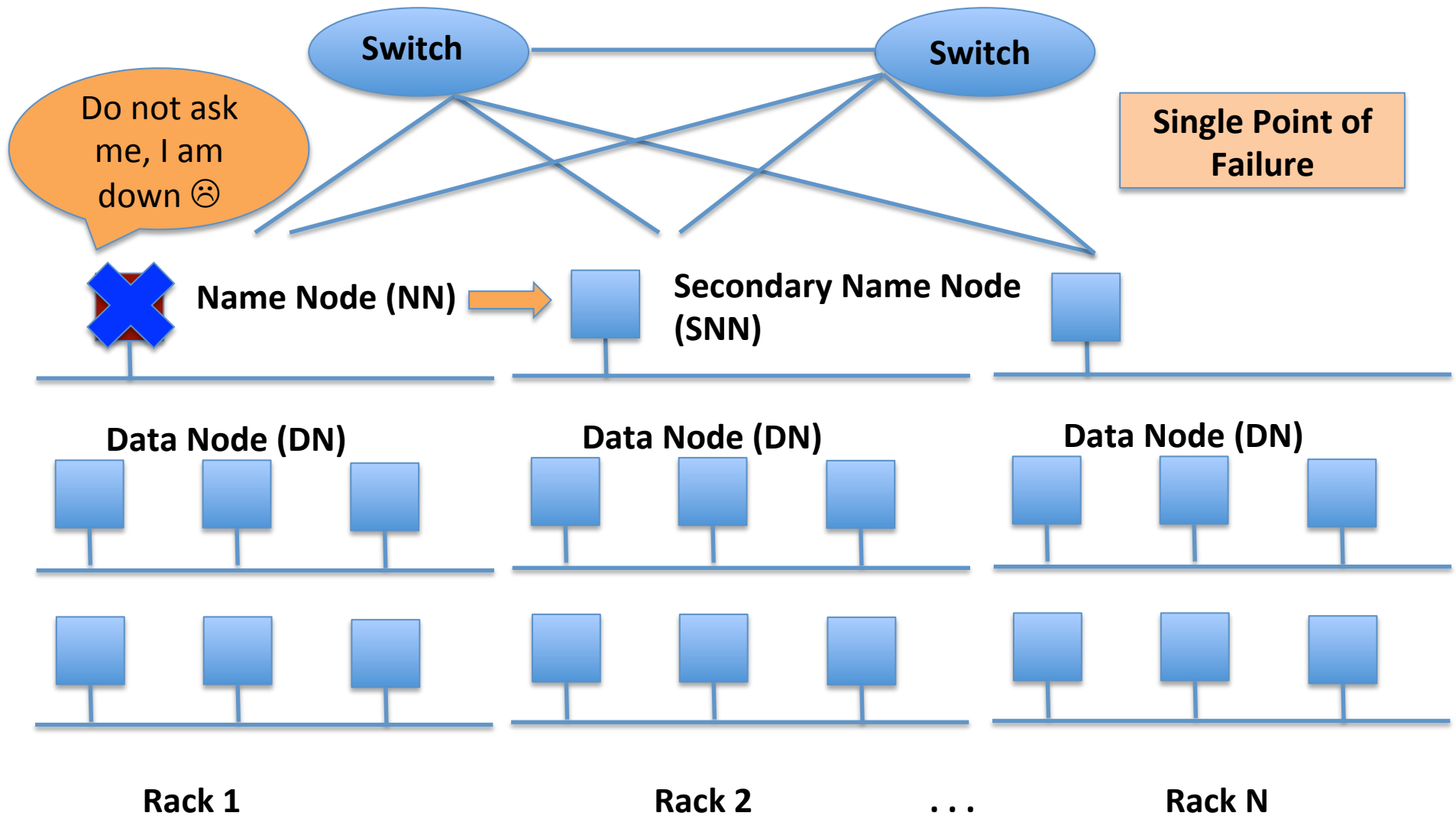
HDFS Architecture: Master-Slave

Multiple-Rack Cluster



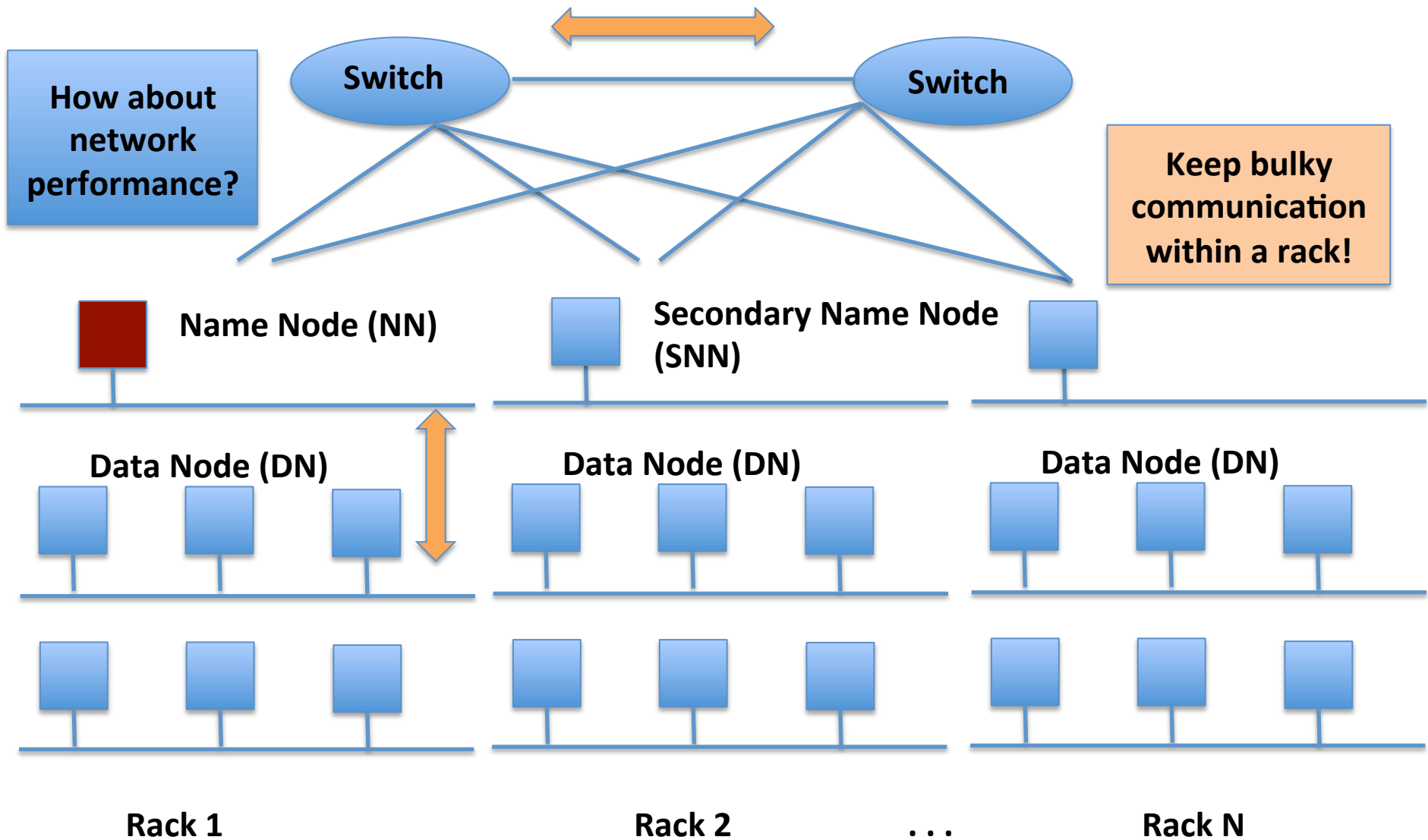
HDFS Architecture: Master-Slave

Multiple-Rack Cluster

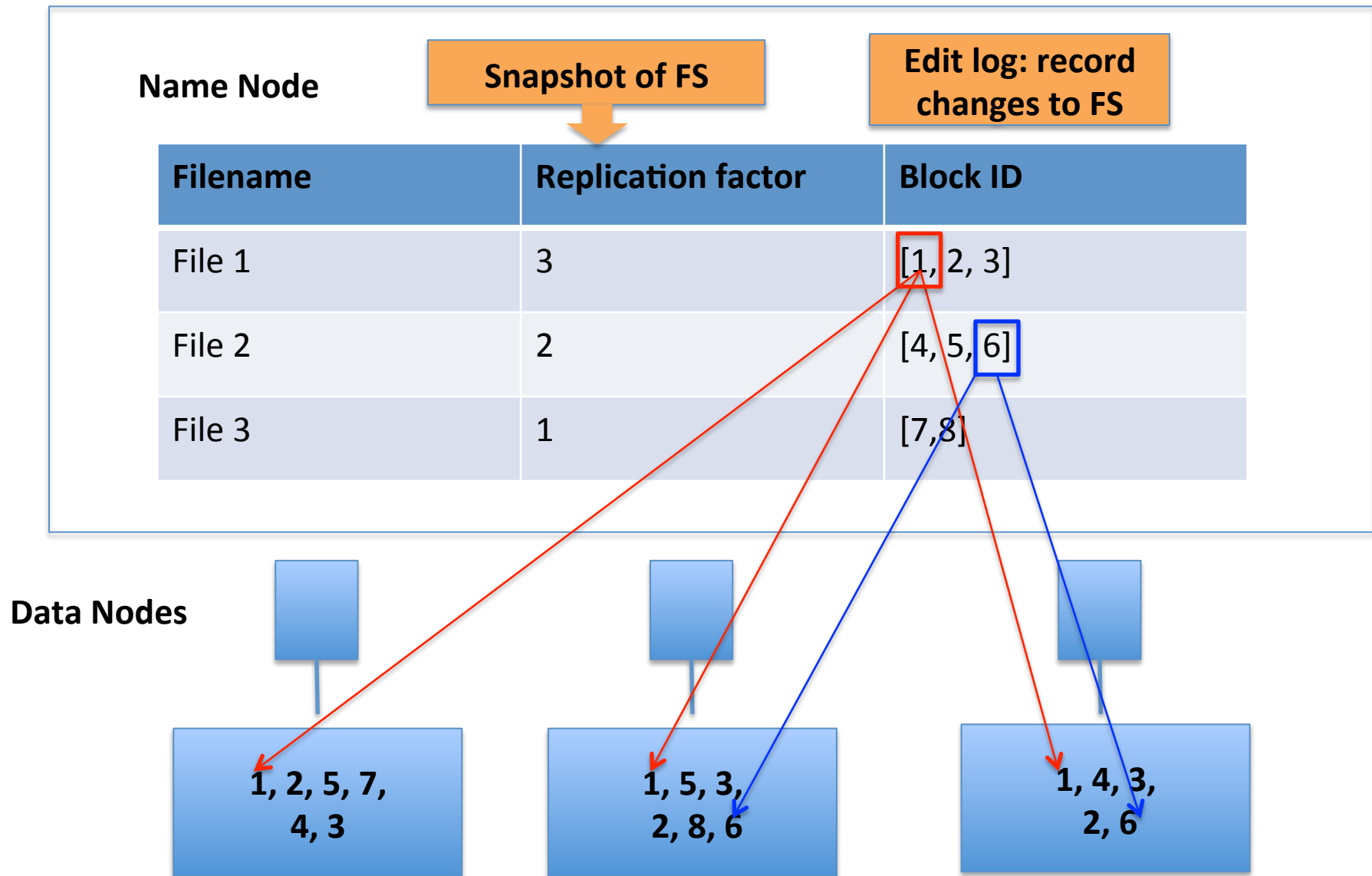


HDFS Architecture: Master-Slave

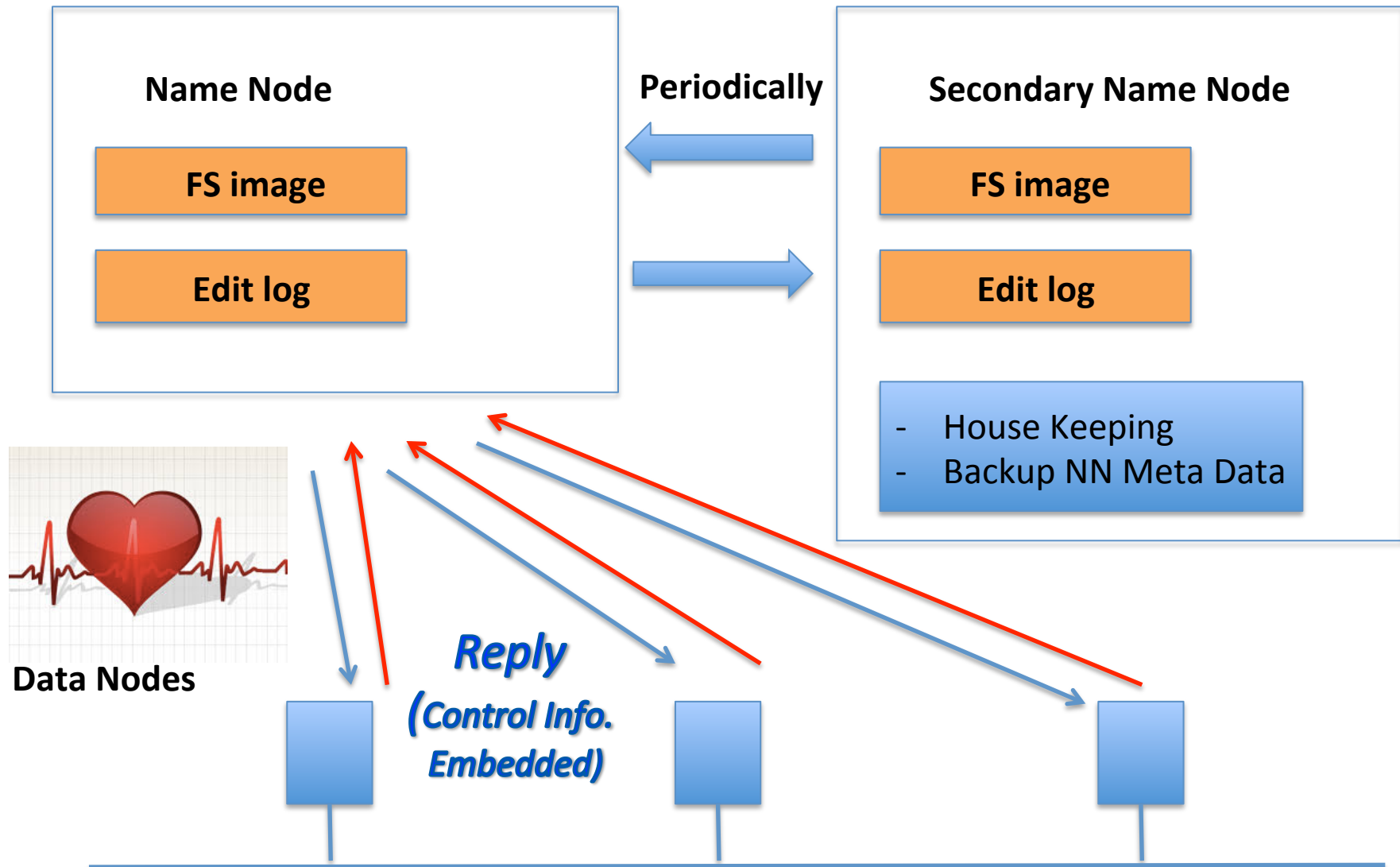
Multiple-Rack Cluster



HDFS Inside: Name Node



HDFS Inside: Name Node



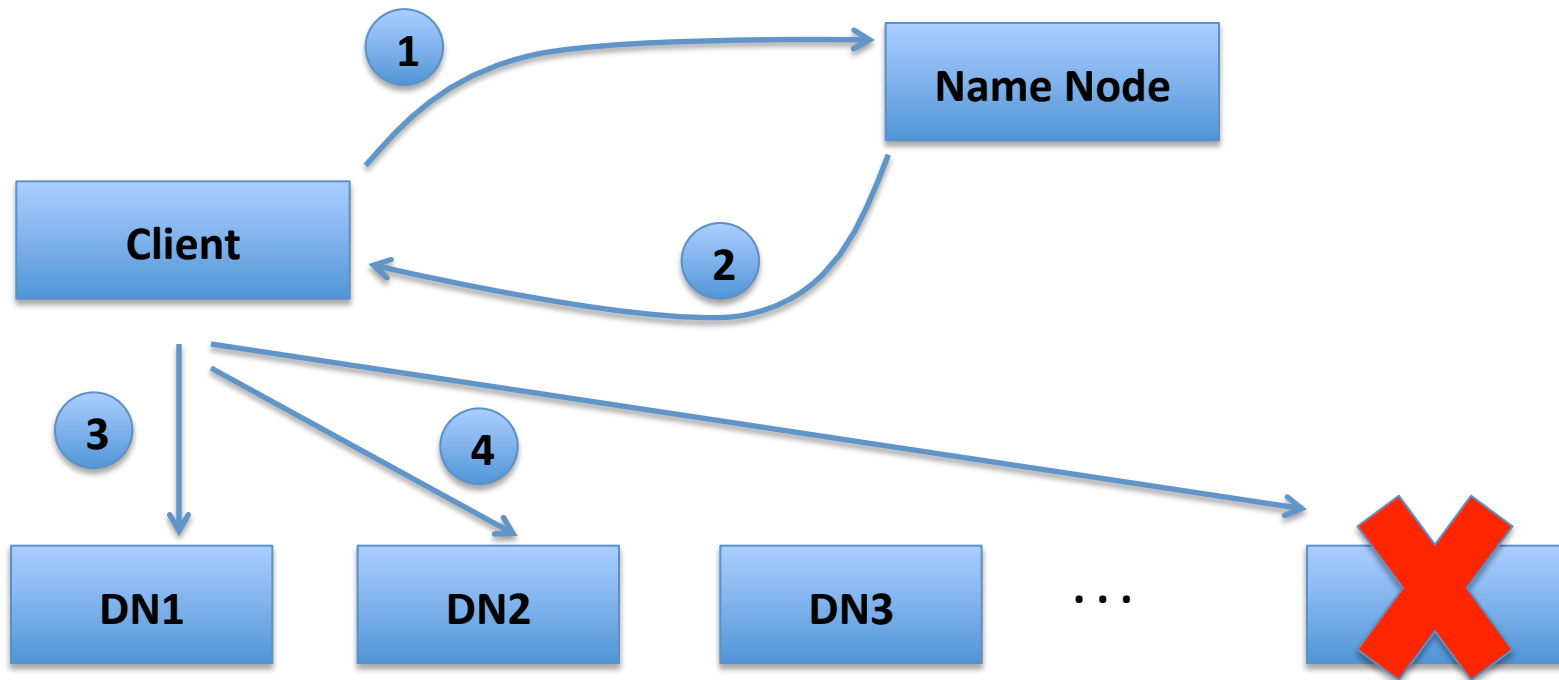
HDFS Inside: Blocks

- Q: Why do we need the abstraction “Blocks” in addition to “Files”?
- Reasons:
 - File can be larger than a single disk
 - Block is of fixed size, easy to manage and manipulate
 - Easy to replicate and do more fine grained load balancing

HDFS Inside: Blocks

- HDFS Block size is by default **64 MB**, why it is much larger than regular file system block?
- Reasons:
 - Minimize overhead: disk seek time is almost constant
 - Example: seek time: 10 ms, file transfer rate: 100MB/s, overhead (seek time/a block transfer time) is 1%, what is the block size?
 - 100 MB (HDFS-> 128 MB)

HDFS Inside: Read



1. Client connects to NN to read data
2. NN tells client where to find the data blocks
3. Client reads blocks directly from data nodes (without going through NN)
4. In case of node failures, client connects to another node that serves the missing block

HDFS Inside: Read

- Q: Why does HDFS choose such a design for read? Why not ask client to read blocks through NN?
- Reasons:
 - Prevent NN from being the bottleneck of the cluster
 - Allow HDFS to scale to large number of concurrent clients
 - Spread the data traffic across the cluster

HDFS Inside: Read

- Q: Given multiple replicas of the same block, how does NN decide which replica the client should read?
- HDFS Solution:
 - Rack awareness based on network topology

HDFS Inside: Network Topology

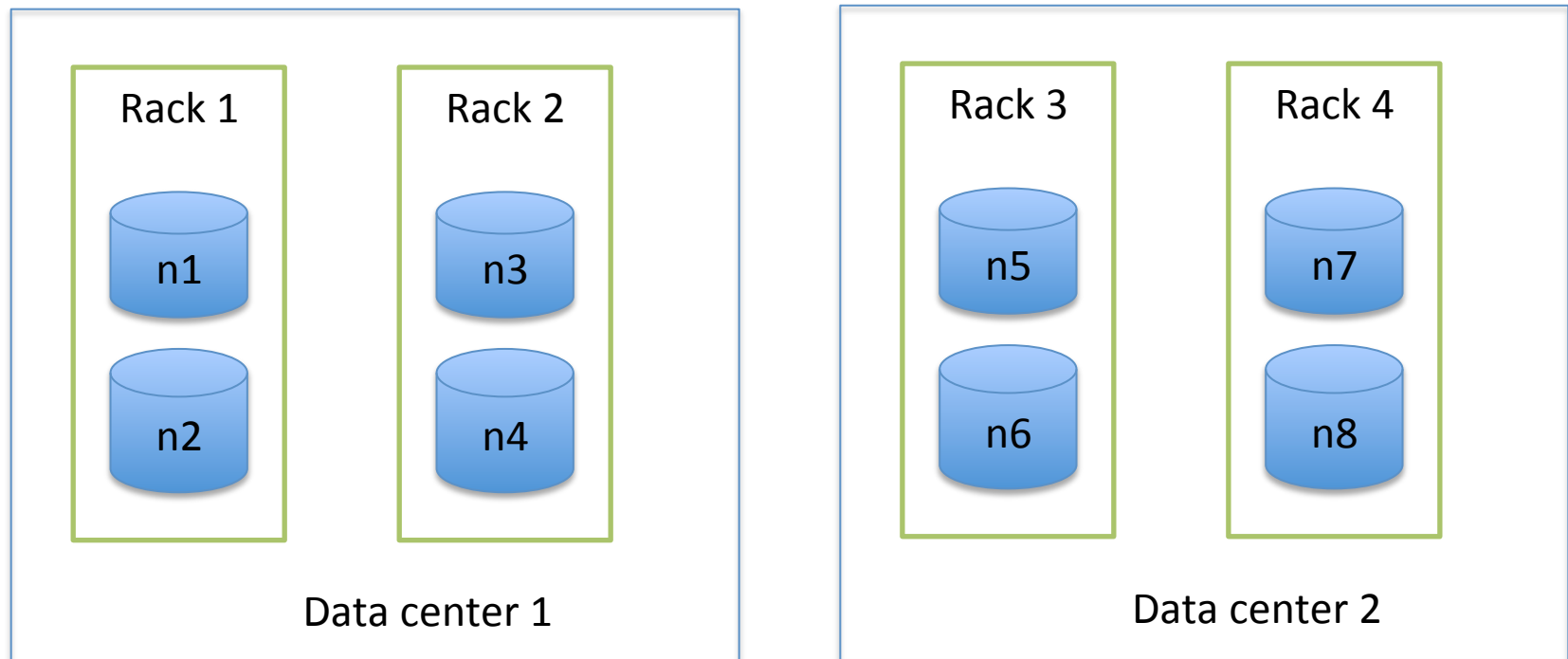
- The critical resource in HDFS is **bandwidth**, distance is defined based on that
- Measuring bandwidths between any pair of nodes is too complex and **does not scale**
- **Basic Idea:**
 - Processes on the same node
 - Different nodes on the same rack
 - Nodes on different racks in the same data center (cluster)
 - Nodes in different data centers



**Bandwidth
becomes less**

HDFS Inside: Network Topology

- HDFS takes a simple approach:
 - See the network as a tree
 - **Distance between two nodes is the sum of their distances to their closest common ancestor**



HDFS Inside: Network Topology

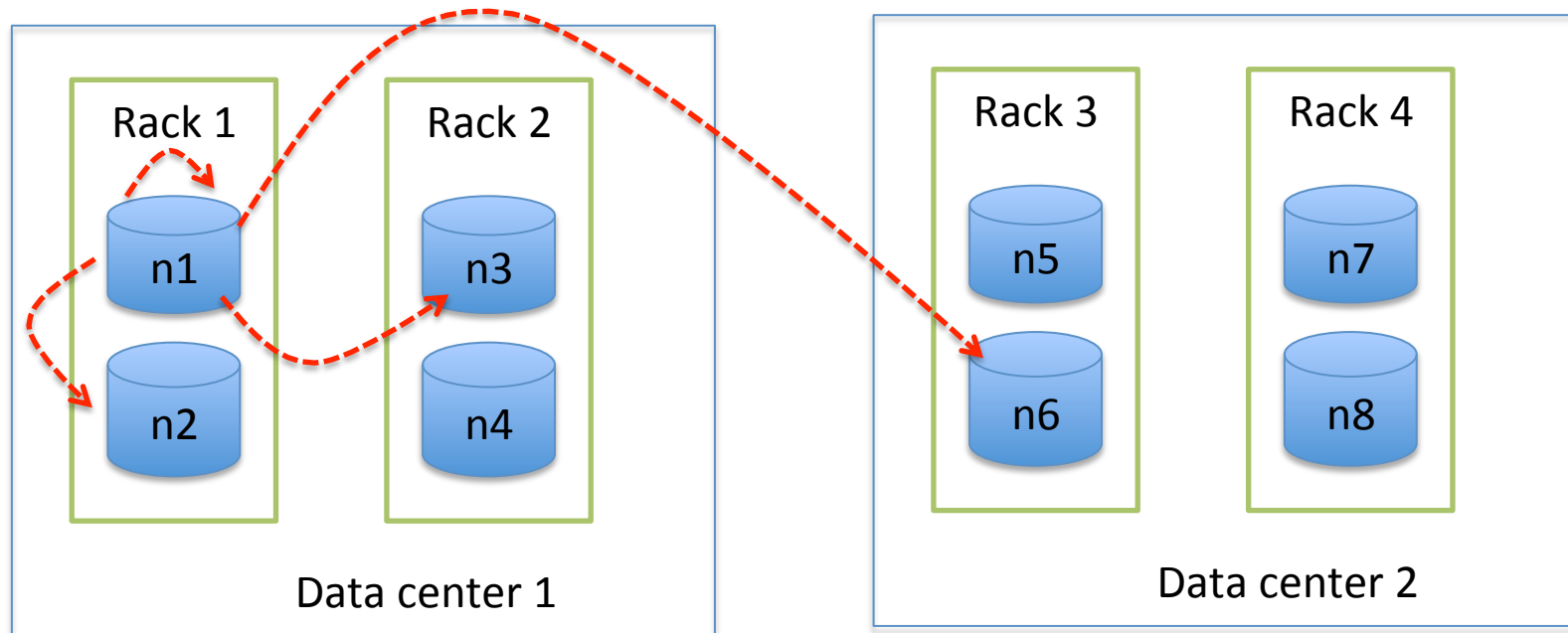
- What are the distance of the following pairs:

Dist (d1/r1/n1, d1/r1/n1)= 0

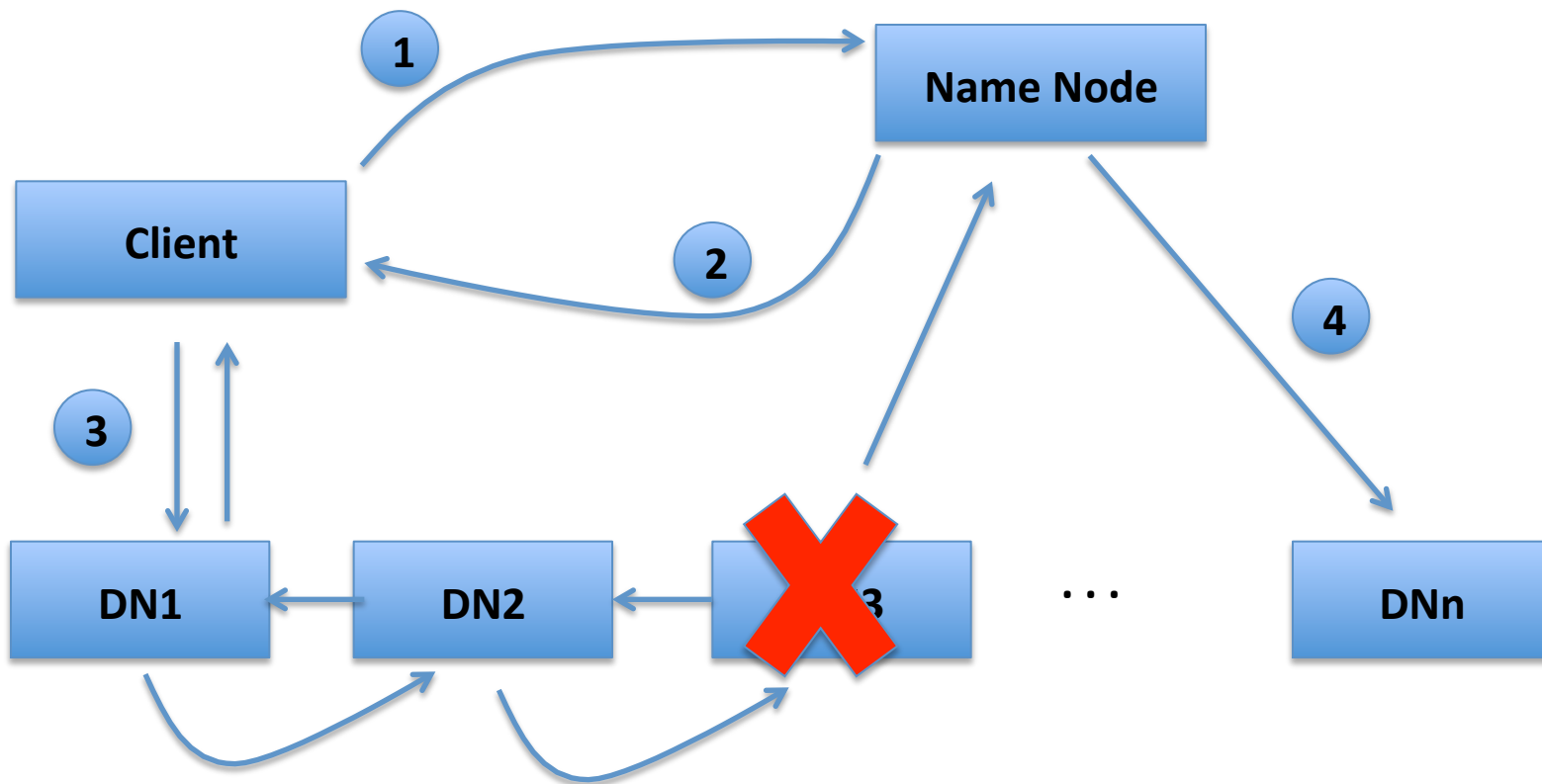
Dist(d1/r1/n1, d1/r1/n2)= 2

Dist(d1/r1/n1, d1/r2/n3)= 4

Dist(d1/r1/n1, d2/r3/n6)= 6



HDFS Inside: Write



1. Client connects to NN to write data
2. NN tells client write these data nodes
3. Client writes blocks directly to data nodes with desired replication factor
4. In case of node failures, NN will figure it out and replicate the missing blocks







HDFS Inside: Write

- Q: Where should HDFS put the three replicas of a block? What tradeoffs we need to consider?
- Tradeoffs:
 - Reliability
 - Write Cost
 - Read Cost

Q: What are some possible strategies?










HDFS Inside: Write

- Replication Strategy vs Tradeoffs

	Reliability	Write Cost	Read Cost
Put all replicas on one node			
Put all replicas on different racks			

HDFS Inside: Write

- Replication Strategy vs Tradeoffs

	Reliability	Write Cost	Read Cost
Put all replicas on one node			
Put all replicas on different racks			
HDFS: 1-> same node as client 2-> a node on different rack 3-> a different node on the same rack as 2			

HDFS Interface

- Web Based Interface
 - <http://ccl.cse.nd.edu/operations/hadoop/>
- Command Line: Hadoop FS Shell
 - <https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html>

HDFS-Web UI

NameNode 'disc01.crc.nd.edu:8020'

Started: Mon Sep 08 12:24:52 EDT 2014

Version: 0.21.0, 985326

Compiled: Tue Aug 17 01:02:28 EDT 2010 by tomwhite from branches/branch-0.21

Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)

[NameNode Logs](#)

Cluster Summary

1580702 files and directories, 1550126 blocks = 3130828 total.

Heap Memory used 1.08 GB is 82% of Committed Heap Memory 1.3 GB. Max Heap Memory is 1.74 GB.

Non Heap Memory used 22.26 MB is 85% of Committed Non Heap Memory 26.12 MB. Max Non Heap Memory is 132 MB.

Configured Capacity	:	179.08 TB
DFS Used	:	26.98 TB
Non DFS Used	:	9.12 TB
DFS Remaining	:	142.98 TB
DFS Used%	:	15.07 %
DFS Remaining%	:	79.84 %
Live Nodes	:	25
Dead Nodes	:	0

HDFS-Web UI

Contents of directory [/users/dwang5](#)

Goto :

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
staging	dir				2014-09-14 23:18	rwX-----	dwang5	supergroup
aggoutput	dir				2014-09-06 14:38	rwXr-Xr-X	dwang5	supergroup
aggtest	dir				2014-09-06 16:44	rwXr-Xr-X	dwang5	supergroup
hw03	dir				2014-09-13 20:10	rwXr-Xr-X	dwang5	supergroup
meanout	dir				2014-09-06 17:53	rwXr-Xr-X	dwang5	supergroup
medianout	dir				2014-09-06 18:06	rwXr-Xr-X	dwang5	supergroup
normaloutputs	dir				2014-08-30 23:30	rwXr-Xr-X	dwang5	supergroup
outputs	dir				2014-08-24 00:33	rwXr-Xr-X	dwang5	supergroup
pages	dir				2014-09-13 23:48	rwXr-Xr-X	dwang5	supergroup
test_hw3	dir				2014-09-14 17:30	rwXr-Xr-X	dwang5	supergroup
tests.txt	file	0.19 KB	3	64 MB	2014-09-06 15:03	rw-r--r--	dwang5	supergroup
testsort.txt	file	0.07 KB	3	64 MB	2014-09-06 17:40	rw-r--r--	dwang5	supergroup
topkout	dir				2014-09-06 20:45	rwXr-Xr-X	dwang5	supergroup
topkeywordout	dir				2014-09-06 21:03	rwXr-Xr-X	dwang5	supergroup
wordcountout	dir				2014-09-07 00:33	rwXr-Xr-X	dwang5	supergroup
wordcountperl	dir				2014-09-07 10:03	rwXr-Xr-X	dwang5	supergroup

HDFS Command Line

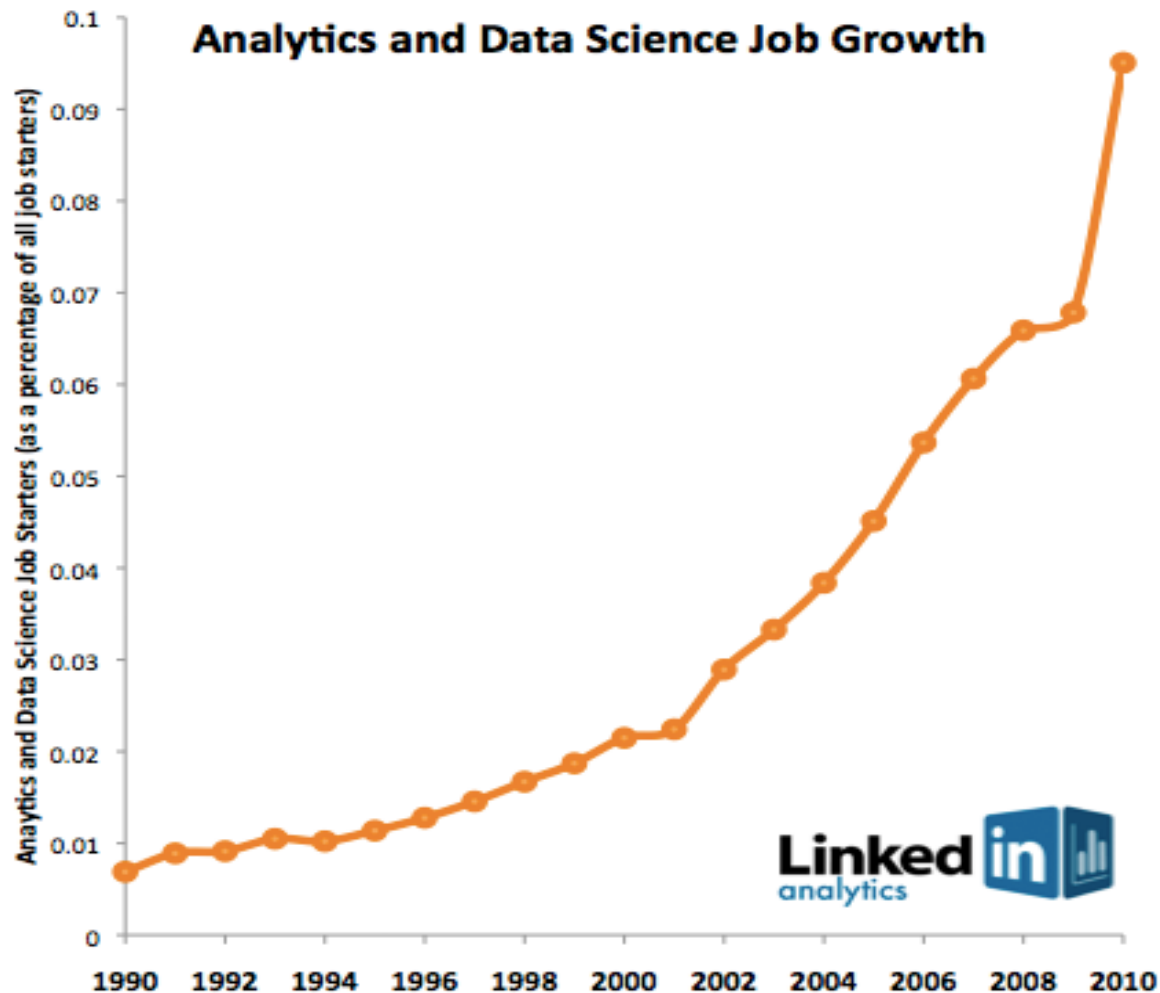
- Hadoop Shell

```
[dwang5@disc01 ~]$ hadoop fs
Jsage: java FsShell
    [-ls <path>]
    [-lsr <path>]
    [-df [<path>]]
    [-du [-s] [-h] <path>]
    [-dus <path>]
    [-count[-q] <path>]
    [-mv <src> <dst>]
    [-cp <src> <dst>]
    [-rm [-skipTrash] <path>]
    [-rmr [-skipTrash] <path>]
    [-expunge]
    [-put <localsrc> ... <dst>]
    [-copyFromLocal <localsrc> ... <dst>]
    [-moveFromLocal <localsrc> ... <dst>]
    [-get [-ignoreCrc] [-crc] <src> <localdst>]
    [-getmerge <src> <localdst> [addnl]]
    [-cat <src>]
    [-text <src>]
    [-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
    [-moveToLocal [-crc] <src> <localdst>]
    [-mkdir <path>]
    [-setrep [-R] [-w] <rep> <path/file>]
    [-touchz <path>]
    [-test [-ezd] <path>]
    [-stat [format] <path>]
```

Big Data Processing 1 Summary

- Big Data and Hadoop background
 - What and Why about Hadoop
 - 4 V challenge of Big Data
- Hadoop Distributed File System (HDFS)
 - Motivation: guide Hadoop design
 - Architecture: Single rack vs Multi-rack clusters
 - Reliable storage, Rack-awareness, Throughput
 - Inside: Name Node file system, Read, Write
 - Interface: Web and Command line

Job Market Demand in Big Data Science in Last Two Decades



Courtesy Linked Corp.

MapReduce

MapReduce Outline

- MapReduce Architecture
- MapReduce Internals
- MapReduce Examples
- JobTracker Interface

MapReduce: A Real World Analogy

Coins Deposit



MapReduce: A Real World Analogy

Coins Deposit



Coins Counting Machine

MapReduce: A Real World Analogy

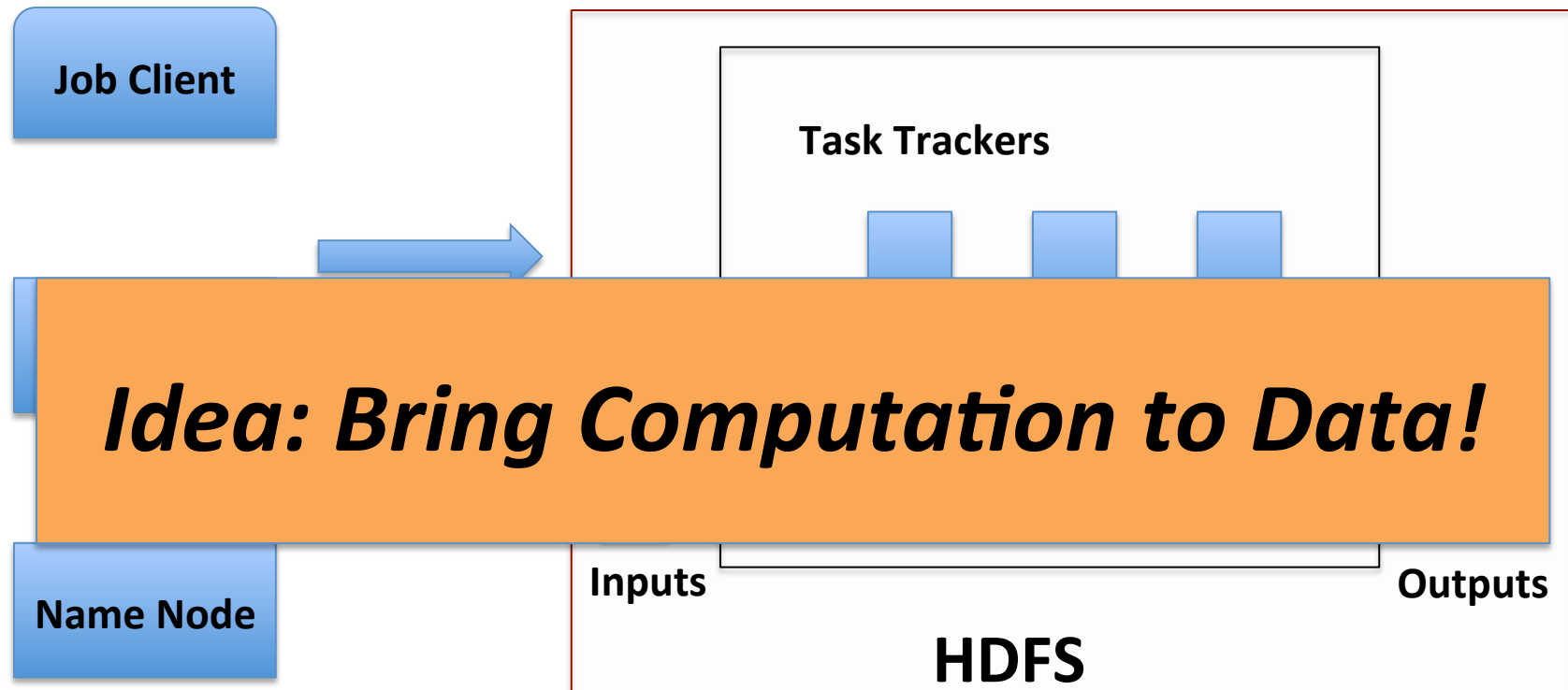
Coins Deposit



Mapper: Categorize coins by their face values

Reducer: Count the coins in each face value *in parallel*

MapReduce Architecture: Master-Slaves



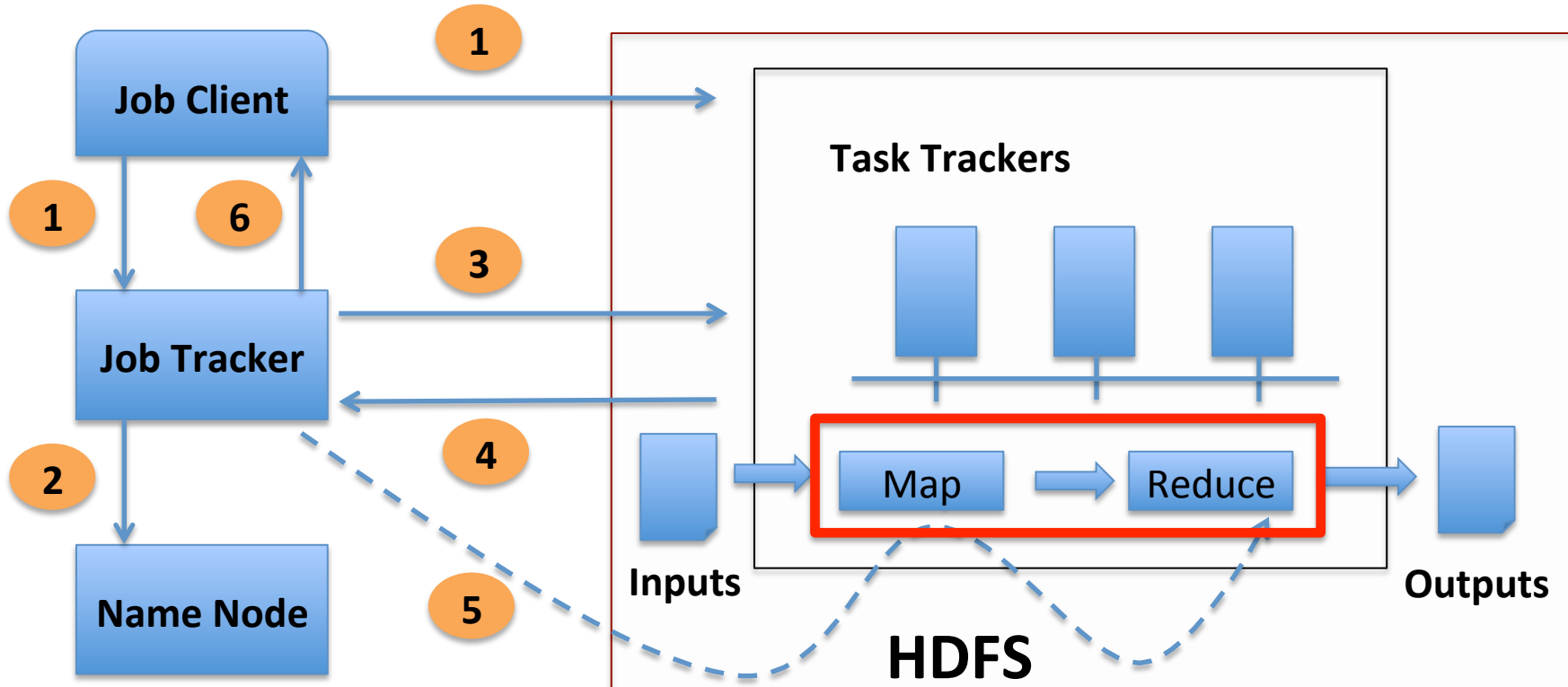
Job Client: Submit Jobs

Task Tracker: Execute Jobs

Job Tracker: Coordinate Jobs

Job: MapReduce Function+ Config
(Scheduling, Phase Coordination, etc.)

MapReduce Architecture: Workflow

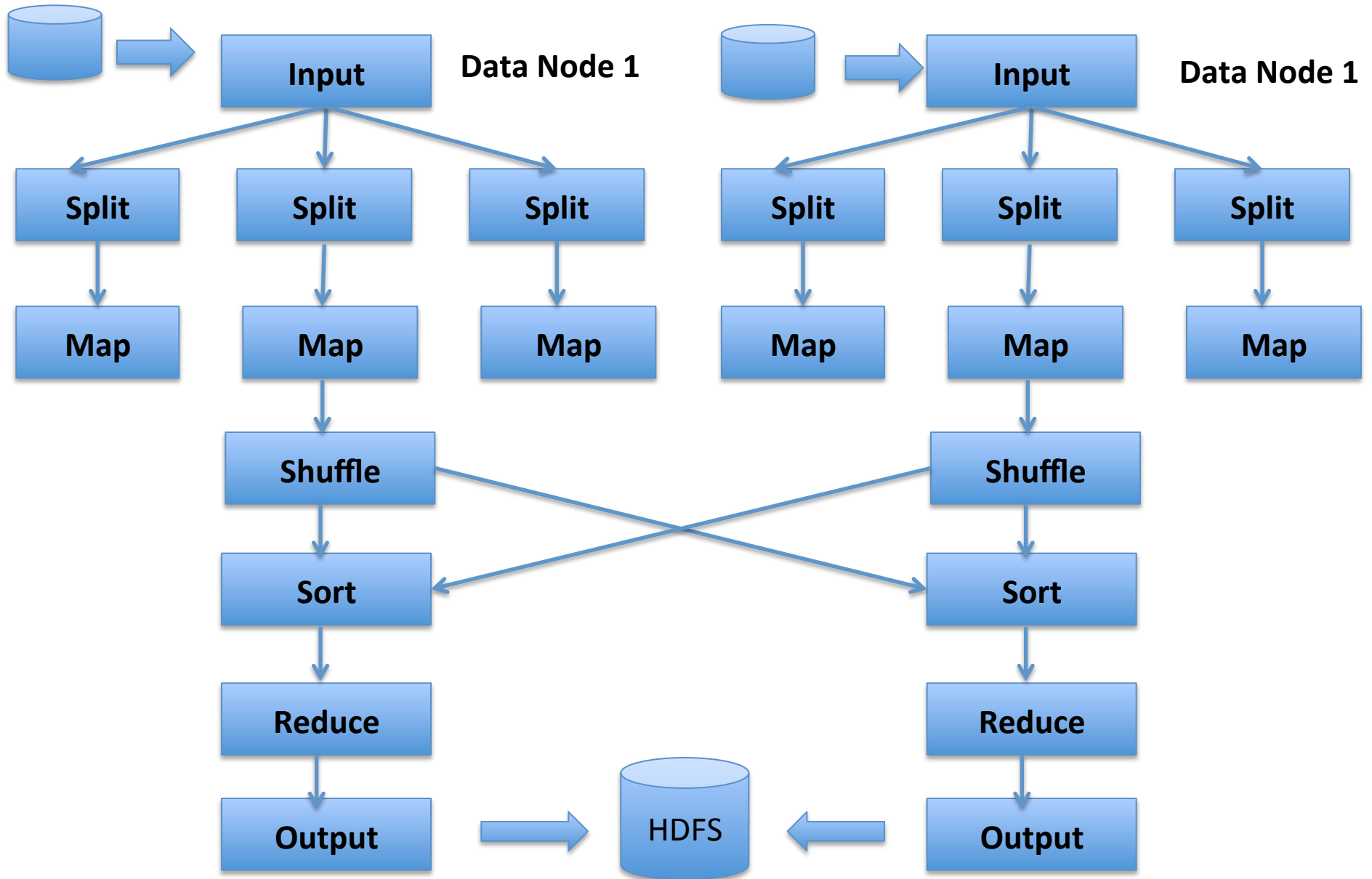


1. Client submits job to Job Tracker and copy code to HDFS
2. Job Tracker talks to NN to find data it needs
3. Job Tracker creates execution plan and submits work to Task Trackers
4. Task trackers do the job and report progress/status to Job Tracker
5. Job Tracker manages task phases
6. Job Tracker finishes the job and updates status

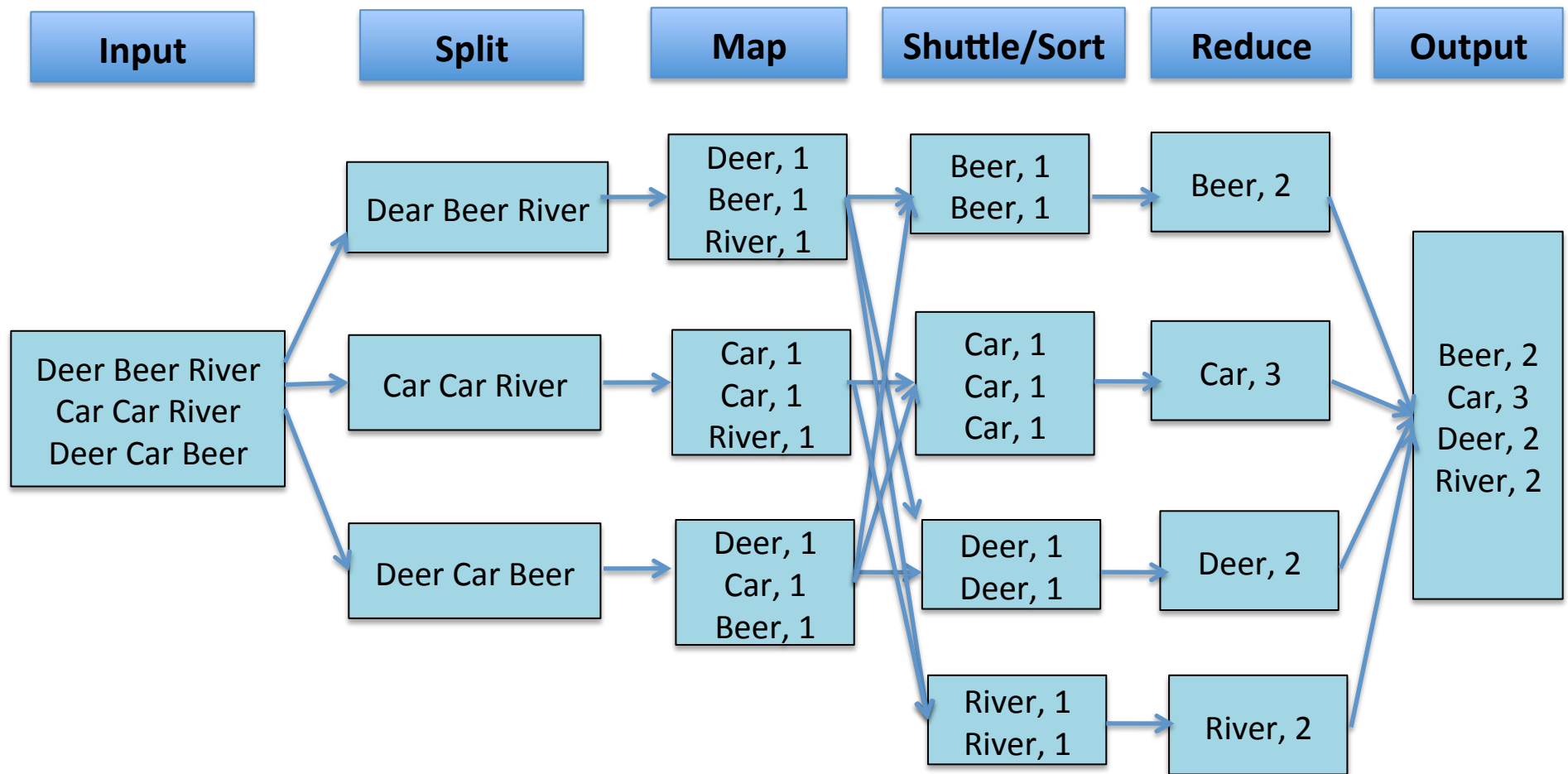
MapReduce Paradigm

- Implement two functions:
 - **Map** (k1,v1) -> list (k2, v2)
 - **Reduce**(k2, list(v2)) -> list (v3)
- Framework handles everything else
- Value with the **same key** go to the same reducer

MapReduce Internal

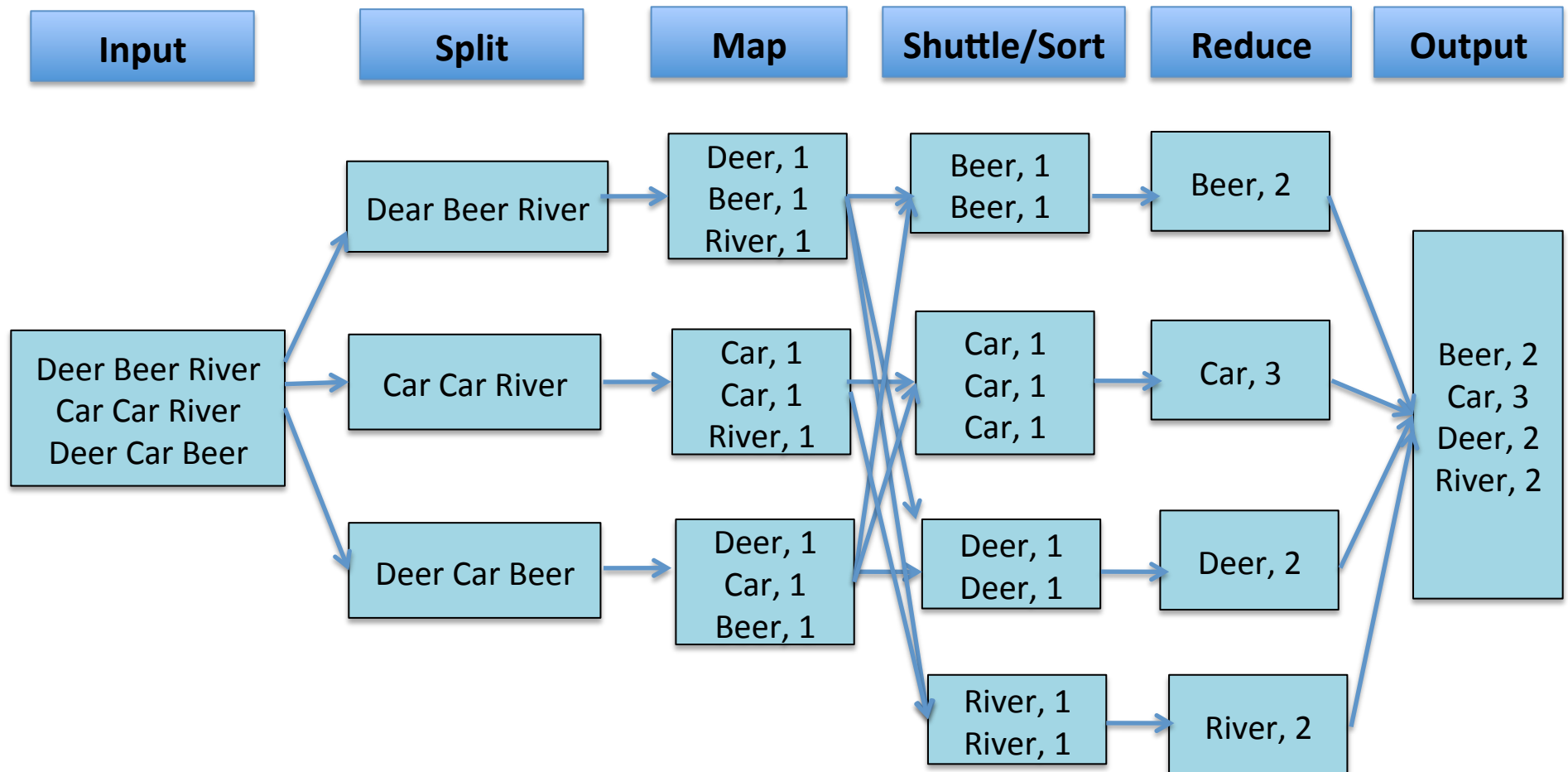


MapReduce Example: Word Count



Similar Flavor of Coins Deposit ? ☺

MapReduce Example: Word Count



Q: What are the Key and Value Pairs of Map and Reduce?

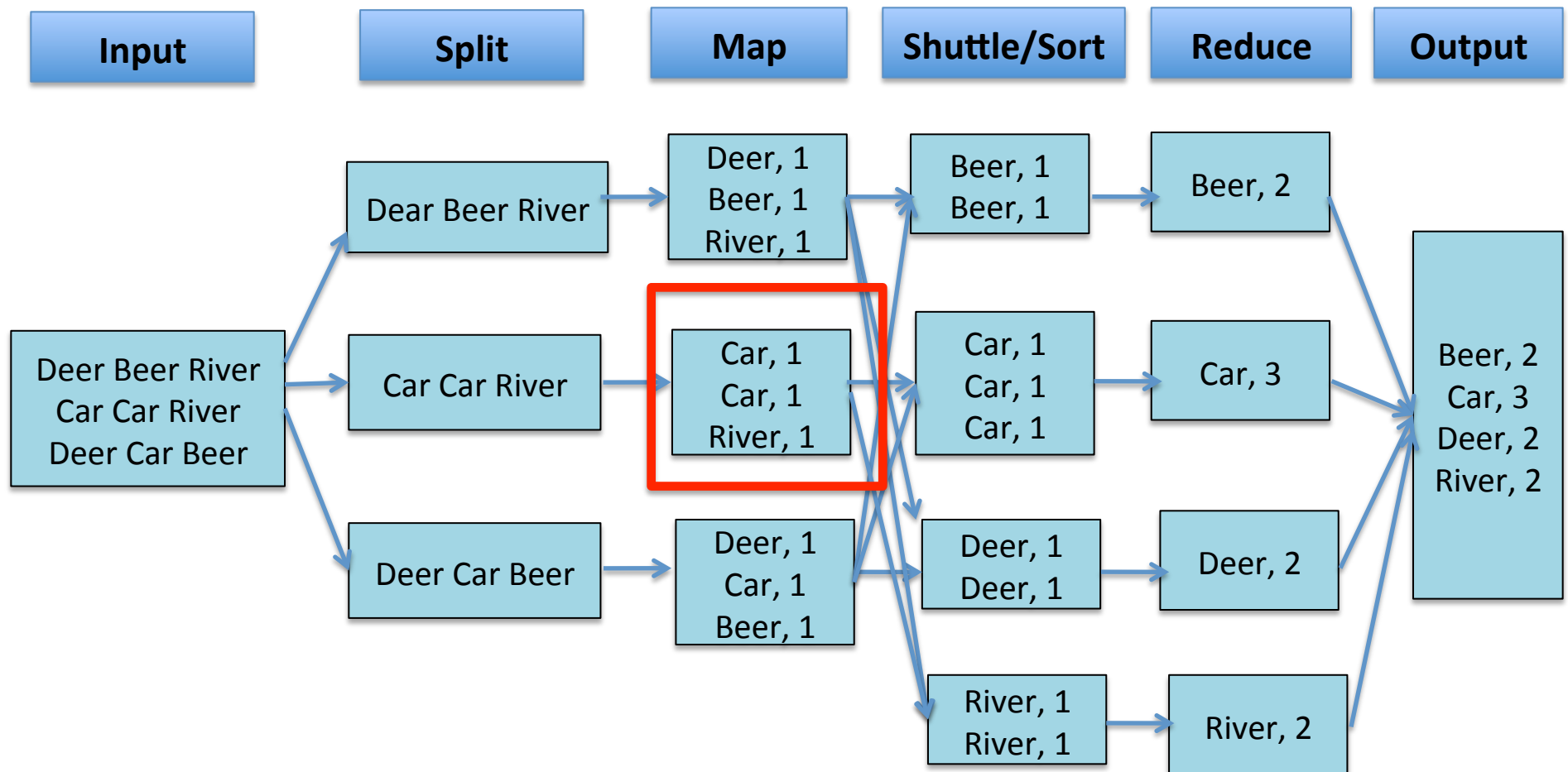
Map: Key=word, Value=1

Reduce: Key=word, Value=aggregated count

Mapper and Reducer of Word Count

- `Map(key, value){`
 // key: line number
 // value: words in a line
 for each word w in value:
 `Emit(w, "1");}`
- `Reduce(key, list of values){`
 // key: a word
 // list of values: a list of counts
 `int result = 0;`
 for each v in values:
 `result += ParseInt(v);`
 `Emit(key, result);}`

MapReduce Example: Word Count

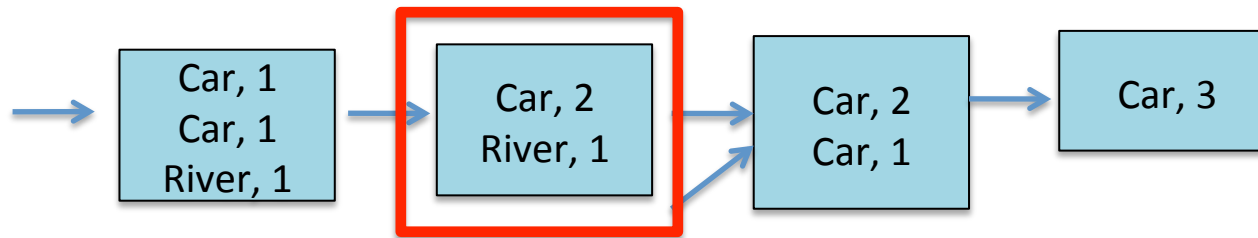


Q: Do you see any place we can improve the efficiency?

Local aggregation at mapper will be able to improve MapReduce efficiency.

MapReduce: Combiner

- **Combiner:** do local aggregation/combine task at mapper



- **Q: What are the benefits of using combiner:**
 - Reduce memory/disk requirement of Map tasks
 - Reduce network traffic
- **Q: Can we remove the reduce function?**
 - No, reducer still needs to process records with same key but from *different mappers*
- **Q: How would you implement combiner?**
 - It is the same as Reducer!

MapReduce WordCount 2

- **New Goal:** output all words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
 - Sort words by their counts in the reducer
 - Problem: what happens if we have more than one reducer?

MapReduce WordCount 2

- **New Goal:** output all words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
 - Do two rounds of MapReduce
 - In the 2nd round, take the output of WordCount as input but switch key and value pair!
 - Leverage the sorting capability of *shuffle/sort* to do the global sorting!

MapReduce WordCount 3

- **New Goal:** output the top K words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
 - Use the solution of previous problem and only grab the top K in the final output
 - Problem: is there a more efficient way to do it?

MapReduce WordCount 3

- **New Goal:** output the top K words sorted by their frequencies (total counts) in a document.
- **Question:** How would you adopt the basic word count program to solve it?
- **Solution:**
 - Add a sort function to the *reducer* in the first round and only output the top K words
 - Intuition: the global top K must be a local top K in any reducer!

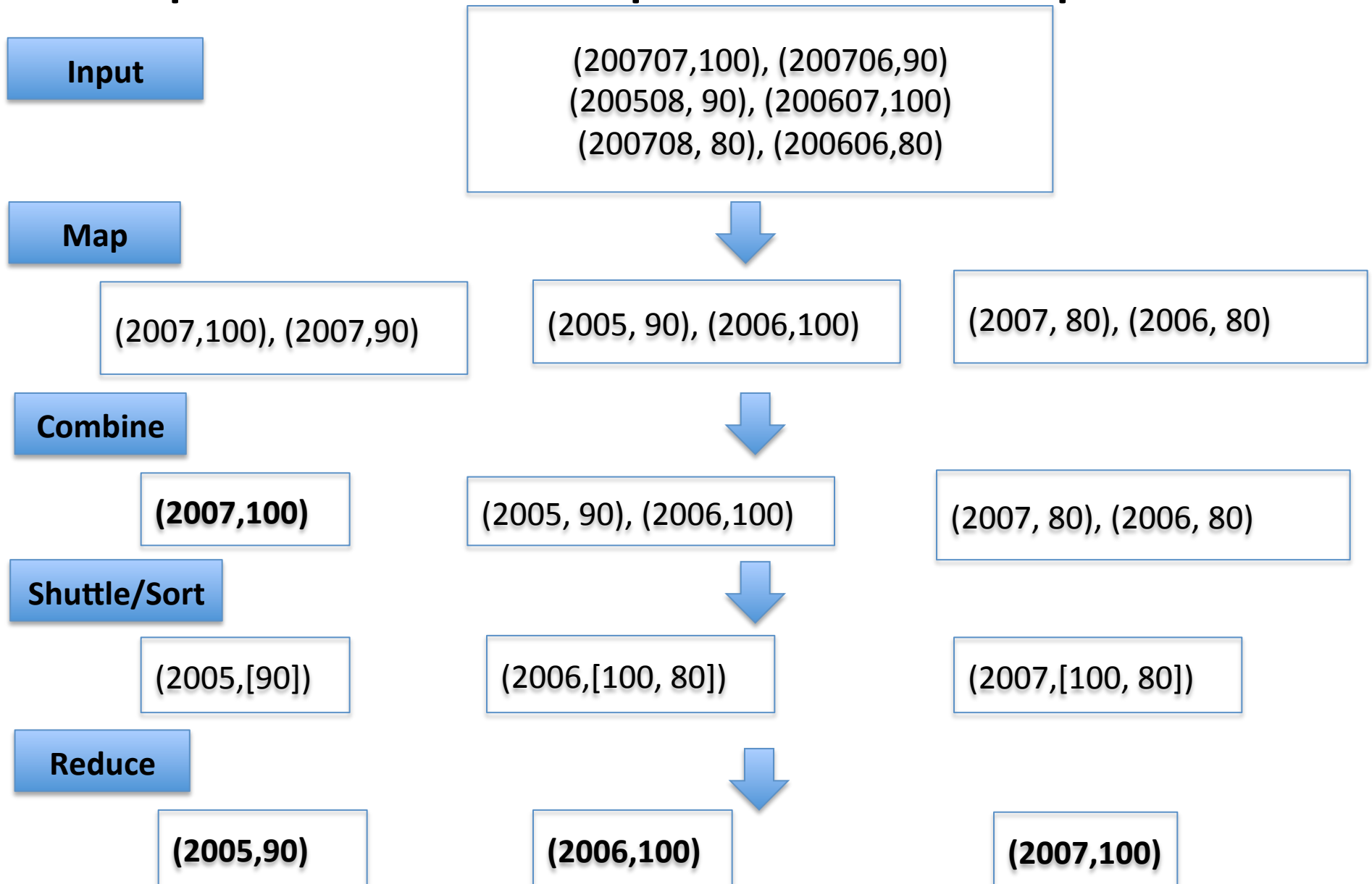
MapReduce In-class Exercise

- **Problem:** Find the maximum monthly temperature for each year from weather reports
- **Input:** A set of records with format as:
 <Year/Month, Average Temperature of that month>
 - (200707,100), (200706,90)
 - (200508, 90), (200607,100)
 - (200708, 80), (200606,80)
- **Question:** write down the Map and Reduce function to solve this problem
 - Assume we split the input by line

Mapper and Reducer of Max Temperature

- Map(key, value){
 // key: line number
 // value: tuples in a line
 for each tuple t in value:
 Emit(t->year, t->temperature);}
 Combiner is the same as Reducer
- Reduce(key, list of values){
 // key: year
 //list of values: a list of monthly temperature
 int max_temp = -100;
 for each v in values:
 max_temp= max(v, max_temp);
 Emit(key, max_temp);}

MapReduce Example: Max Temperature



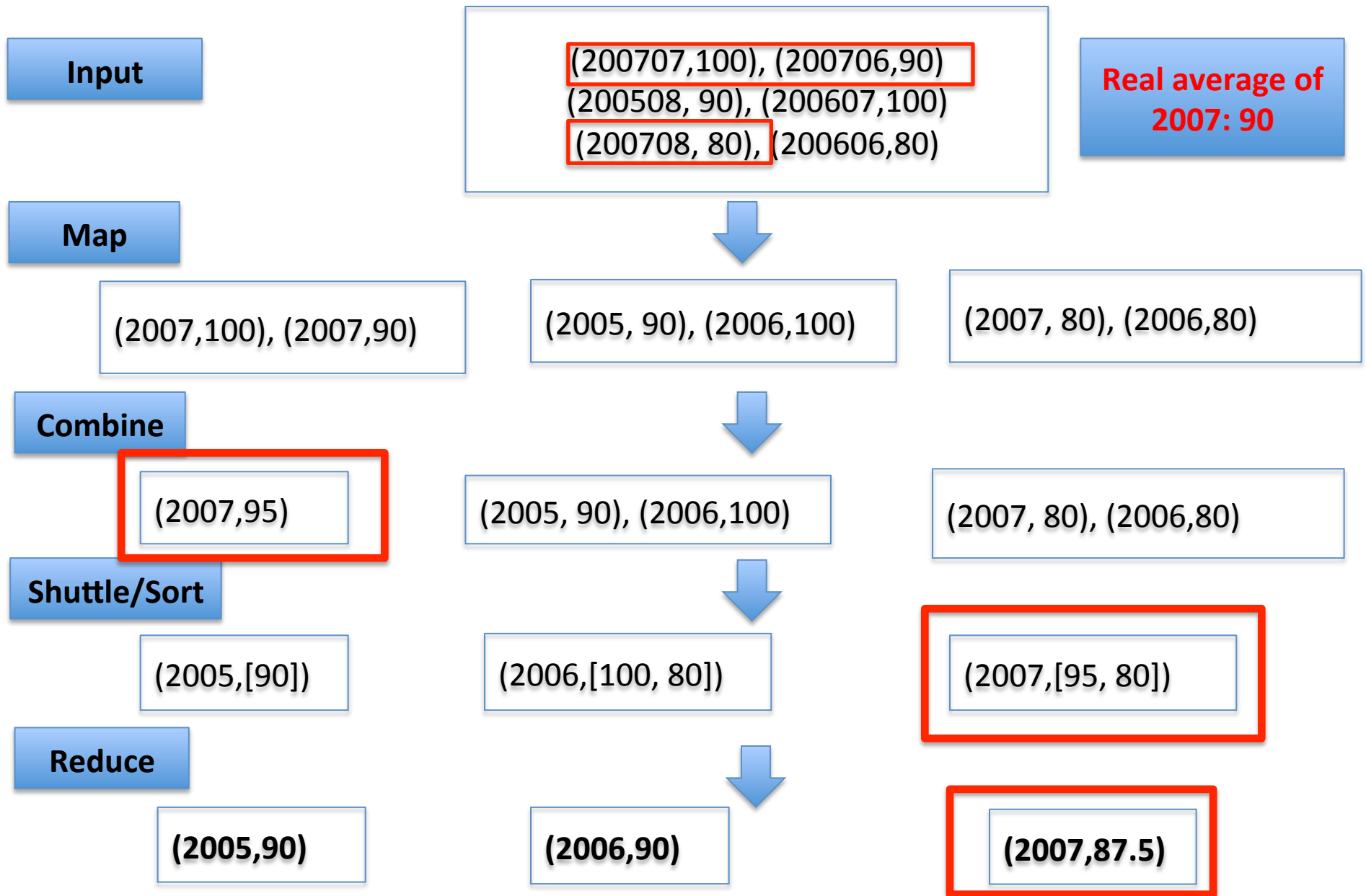
MapReduce In-class Exercise

- **Key-Value Pair of Map and Reduce:**
 - **Map:** (year, temperature)
 - **Reduce:** (year, maximum temperature of the year)
- **Question: How to use the above Map Reduce program (*that contains the combiner*) with slight changes to find the average monthly temperature of the year?**

Mapper and Reducer of Average Temperature

- Map(key, value){
 // key: line number
 // value: tuples in a line
 for each tuple t in value:
 Emit(t->year, t->temperature);}
 - Reduce(key, list of values){
 // key: year
 // list of values: a list of monthly temperatures
 int total_temp = 0;
 for each v in values:
 total_temp= total_temp+v;
 Emit(key, total_temp/size_of(values));}
- Combiner is the same as Reducer

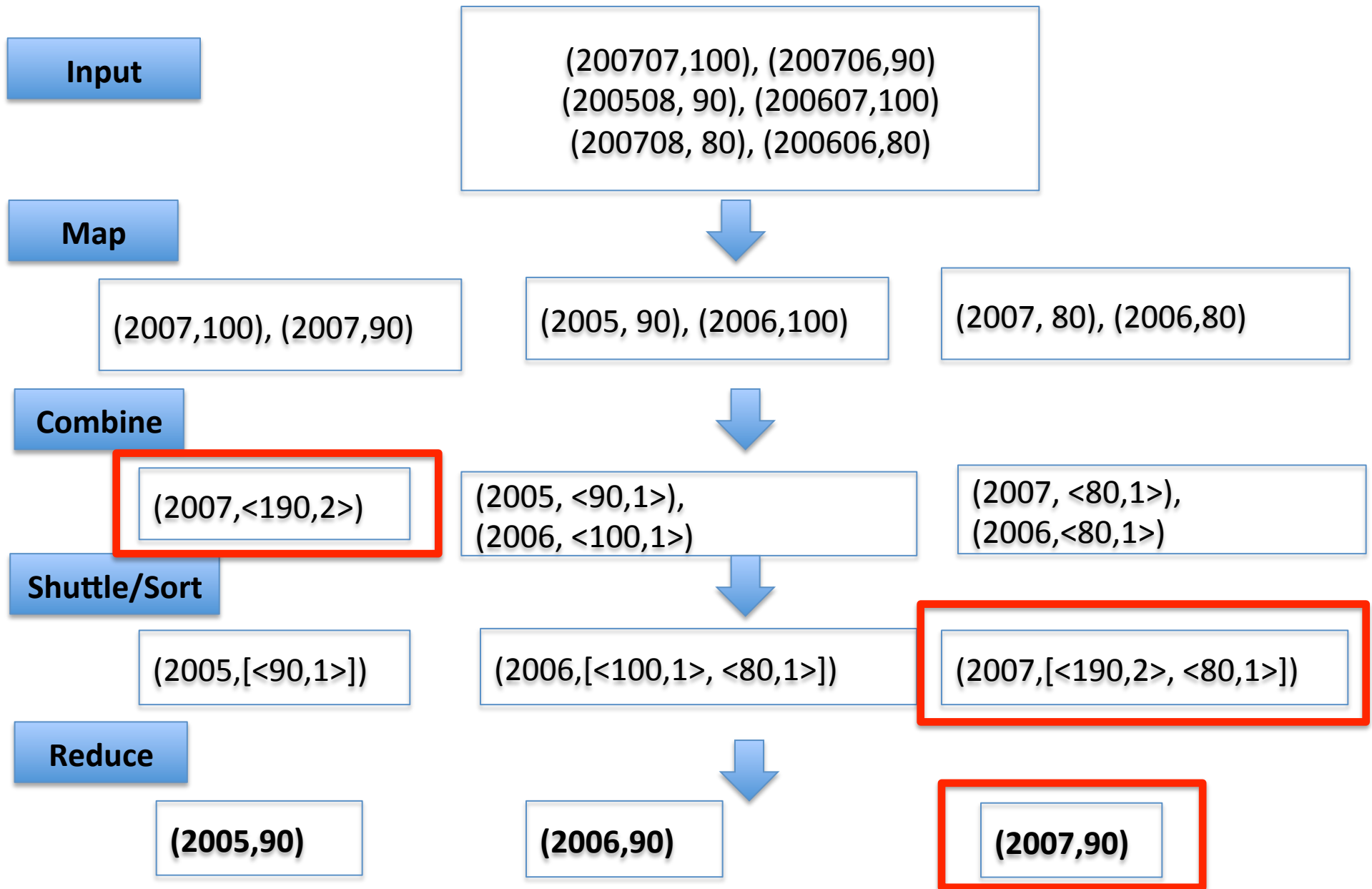
MapReduce Example: Average Temperature



MapReduce In-class Exercise

- The problem is with the combiner!
- Here is a simple counterexample:
 - (2007, 100), (2007,90) -> (2007, 95)
 - (2007,80)->(2007,80)
 - Average of the above is: (2007,87.5)
 - However, the real average is: (2007,90)
- However, we can do a small trick to get around this
 - Mapper: (2007, 100), (2007,90) -> (2007, <190,2>)
 - (2007,80)->(2007,<80,1>)
 - Reducer: (2007,<270,3>)->(2007,90)

MapReduce Example: Average Temperature



Mapper and Reducer of Average Temperature

- **Map(key, value){**
 // key: line number
 // value: tuples in a line
 for each tuple t in value:
 Emit(t->year, t->temperature);}

- **Reduce (key, list of values){**
 // key: year
 // list of values: a list of <temperature
sums, counts> tuples

int total_temp = 0;

int total_count=0;

for each v in values:

total_temp= total_temp+v->sum;

total_count=total_count+v->count;

Emit(key,**total_temp/total_count**);}

- **Combine(key, list of values){**
 // key: year
 // list of values: a list of monthly
temperature

int total_temp = 0;

for each v in values:

total_temp= total_temp+v;

Emit(key,**<total_temp,size_of(list of
values)>**);}

**Combiner is different
from Reducer!**

MapReduce In-class Exercise

- **Functions that can use combiner are called *distributive*:**
 - Distributive: Min/Max(), Sum(), Count(), TopK()
 - Non-distributive: Mean(), Median(), Rank()

Gray, Jim*, et al. "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals." Data Mining and Knowledge Discovery 1.1 (1997): 29-53.

*Jim Gray received Turing Award in 1998

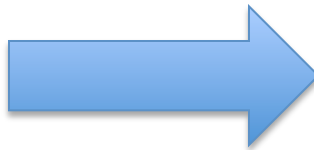


Map Reduce Problems Discussion

- **Problem 1:** Find Word Length Distribution
- **Statement:** Given a set of documents, use Map-Reduce to find the length distribution of all words contained in the documents
- **Question:**
 - What are the Mapper and Reducer Functions?

This is a test data for
the word length
distribution problem

MapReduce



12: 1
7: 1
6: 1
4: 4
3: 2
2: 1
1: 1

Map Reduce Problems Discussion

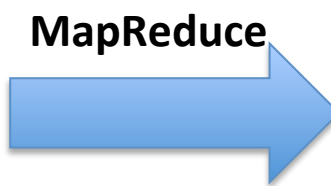
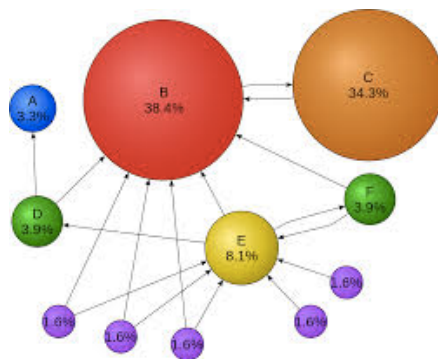
- **Problem 1:** Find Word Length Distribution
- **Mapper and Reducer:**
 - Mapper(document)
 { Emit (**Length(word), word**) }
 - Reducer(output of map)
 { Emit (Length(word), **Size of (List of words at a particular length)**) }

Mapper and Reducer of Word Length Distribution

- `Map(key, value){`
 // key: document name
 // value: words in a document
 for each word `w` in value:
 `Emit(length(w), w);`
- `Reduce(key, list of values){`
 // key: length of a word
 // list of values: a list of words with the same length
 `Emit(key, size_of(list of values));`

Map Reduce Problems Discussion

- **Problem 2:** Indexing & Page Rank
- **Statement:** Given a set of web pages, each page has a **page rank** associated with it, use Map-Reduce to find, for each word, a list of pages (sorted by rank) that contains that word
- **Question:**
 - What are the Mapper and Reducer Functions?



Word 1: [page x1,
page x2, ..]

Word 2: [page y1,
page y2, ...]

...

Map Reduce Problems Discussion

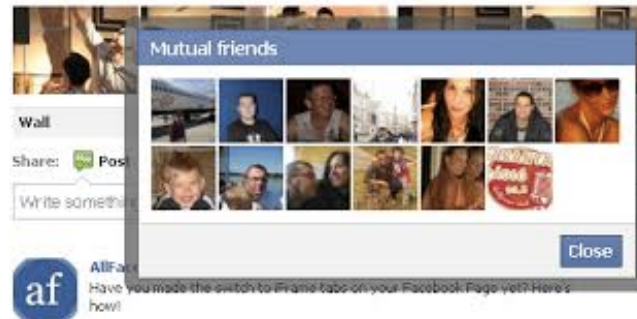
- **Problem 2: Indexing and Page Rank**
- **Mapper and Reducer:**
 - Mapper(page_id, <page_text, page_rank>)
 { Emit (**word**, <**page_id**, **page_rank**>) }
 - Reducer(output of map)
 { Emit (word, **List of pages contains the word sorted by their page_ranks**) }

Mapper and Reducer of Indexing and PageRank

- `Map(key, value){`
 // key: a page
 // value: words in a page, page_rank
 for each word w in value:
 `Emit(w, <page_id, page_rank>;)`
- `Reduce(key, list of values){`
 // key: a word
 // list of values: a list of pages containing that word
 `sorted_pages=sort(list of values, page_rank)`
 `Emit(key, sorted_pages);}`

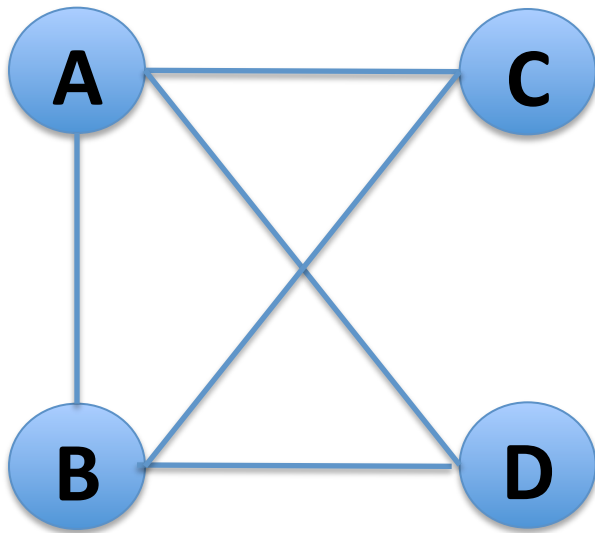
Map Reduce Problems Discussion

- **Problem 3:** Find Common Friends
- **Statement:** Given a group of people on online social media (e.g., Facebook), each has a list of friends, use Map-Reduce to find common friends of any two persons who are friends
- **Question:**
 - What are the Mapper and Reducer Functions?



Map Reduce Problems Discussion

- **Problem 3: Find Common Friends**
- **Simple example:**



Input:

A -> B,C,D

B-> A,C,D

C-> A,B

D->A,B

Output:

(A ,B) -> C,D

(A,C) -> B

(A,D) -> ..

....

MapReduce

Map Reduce Problems Discussion

- **Problem 3: Find Common Friends**
- **Mapper and Reducer:**
 - Mapper(friend list of a person)
 - { for each person in the friend list:
Emit (<**friend pair**>, <**list of friends**>) }
 - Reducer(output of map)
 - { Emit (<friend pair>, **Intersection of two (i.e, the one in friend pair) friend lists**) }

Map Reduce Problems Discussion

- **Problem 3: Find Common Friends**
- **Mapper and Reducer:**

Input:

A -> B,C,D
B-> A,C,D
C-> A,B
D->A,B

Map:

(A,B) -> B,C,D

(A,C) -> B,C,D

(A,D) -> B,C,D

(A,B) -> A,C,D

(B,C) -> A,C,D

(B,D) -> A,C,D

(A,C) -> A,B

(B,C) -> A,B

(A,D) -> A,B

(B,D) -> A,B

Reduce:

(A,B) -> C,D

(A,C) -> B

(A,D) -> B

(B,C) -> A

(B,D) -> A

*Suggest
Fiends 😊*

Mapper and Reducer of Common Friends

- `Map(key, value){`
 `// key: person_id`
 `// value: the list of friends of the person`
 `for each friend f_id in value:`
 `Emit(<person_id, f_id>, value);}`
- `Reduce(key, list of values){`
 `// key: <friend pair>`
 `// list of values: a set of friend lists related with the friend pair`
 `for v1, v2 in values:`
 `common_friends = v1 intersects v2;`
 `Emit(key, common_friends);}`

Map Reduce Problems Discussion

- **Problem 4:** Unique User Count
 - **Statement:** You have a set of documents containing the web browsing records for a company in the form of:
 - *Record x [User_id, Visit_Date, Url]*
- Use Map-Reduce to count the number of unique user visits of the company web pages *per day*.
- **Question:**
 - What are the Mapper and Reducer Functions?



Map Reduce Problems Discussion

- **Problem 4: Unique User Count**
- **Mapper and Reducer:**
 - Mapper(records)
 - { for each record in records:
Emit (**day, user_id**) }
 - Reducer(output of map)
 - { Emit (day, **count of unique user_id appearing on this particular day**) }

Mapper and Reducer of Unique User Count

- `Map(key, value){`
 // key: name of a browsing record document
 // value: the list of browsing records in the document
 for each record r in value:
 `Emit(r->date, r->user_id);}`
- `Reduce(key, list of values){`
 // key: date
 // list of values: a list of users who visit on the day specified
by the key
 `unique_list = unique (values)`
 `Emit(key, size_of(unique_list));}`

Map Reduce Problems Discussion

- **Problem 4: Unique User Count**
- **Q: How to solve this problem by just using the word count program (i.e., do not use the unique function in the reducer)?**

Map Reduce Problems Discussion

- **Problem 4: Unique User Count**
- **Two rounds of Map Reduce:**
 - Round 1:
 - Mapper(record)
 { for each record in records:
 Emit (<**day**, **user_id**>, **1**) }
 - Reducer: same as word count
 - Round 2:
 - Mapper (output of round 1)
 {for each item in list of (<day, user_id>, count)
 Emit (**day**, **1**)}
 - Reducer: same as word count

Mapper and Reducer of Unique User Count

- `Map1(key, value){`
 // key: name of a browsing record document
 // value: the list of browsing records in the document
 for each record `r` in `value`:
 `Emit(<r->date, r->user_id>, 1);`
• `Reduce1`: same as word count reducer
- `Map2(key, list of values){`
 // key: `<date, user_id>`
 // list of values: a list of counts related with the key
 `Emit(key->date, 1);`
• `Reduce2`: same as word count reducer

Map Reduce Problems Discussion

- **Problem 4: Unique User Count**
- **What if we have records from multiple companies mixed together, we want to find the *unique user count per day per company*?**
- **Similar solutions, just change the <key, value> pair.**

Mapper and Reducer of Unique User Count

- Map_p1(key, value){
 // key: name of a browsing record document
 // value: the list of browsing records in the document
 for each record r in value:
 Emit(<r->company_url, r->date, r->user_id>, 1);}
- Reduce_p1: same as word count reducer
- Map_p2(key, list of values){
 // key: <company_url, day, user_id>
 // list of values: a list of counts related with the key
 Emit(<key->company_url, key->date>, 1);}
- Reduce_p2: same as word count reducer

Big Data Processing 2 Summary

Map-Reduce:

- Map-Reduce Paradigm: Mapper and Reducer
- What is the key and what is the value and list of values?
- Combiner and Distributive Functions
- Practice makes perfect 😊

Enjoy MR and Hadoop😊

