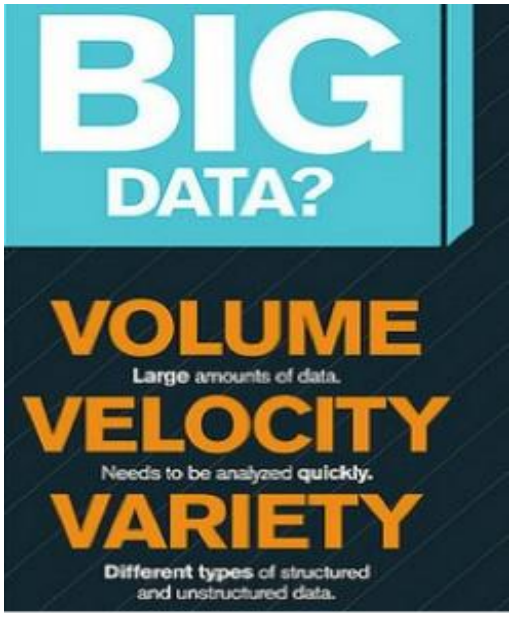
9311 ADVANCED TOPICS - INTRODUCTION TO BIG DATA MANAGEMENT

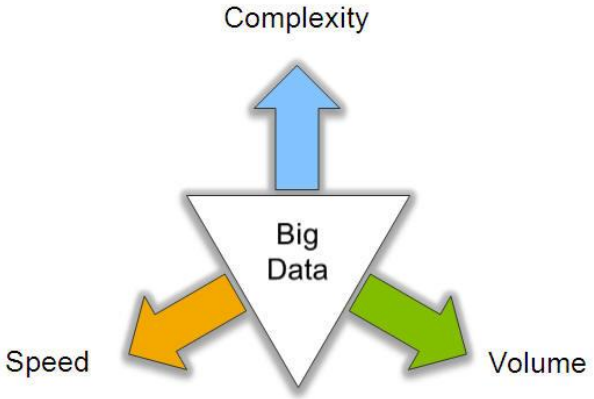
Outline

- Big Data
  - The 3V’ characteristics
  - Cloud Computing
- MapReduce
  - Basic Idea
  - An example

BIG DATA

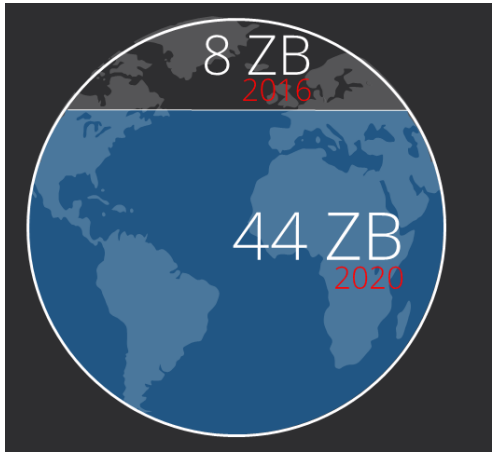
Big Data Characteristics: 3V



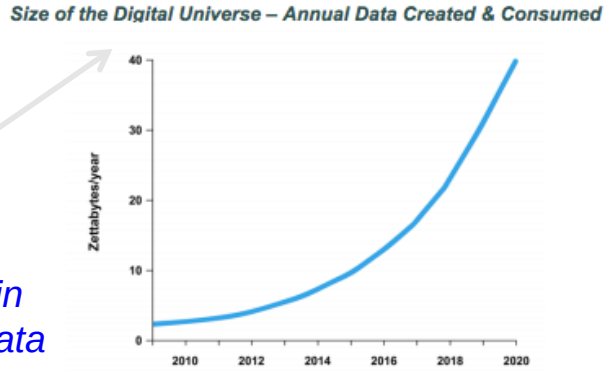


Volume (Scale)

- Data Volume
  - Growth 40% per year
  - From 8 zettabytes (2016) to 44zb (2020)
- Data volume is increasing exponentially



Exponential increase in collected/generated data



Data Source: IDC Digital Universe 2013

Volume (Scale)

Using SSD of 6G/s, a linear scan of a data set ***D*** would take

- 1.9 days when ***D*** is of 1PB (10^{15}B)
- 5.28 years when ***D*** is of 1EB (10^{18}B)

Volume (Scale)

640K ought to be enough for anybody.

Variety (Complexity)

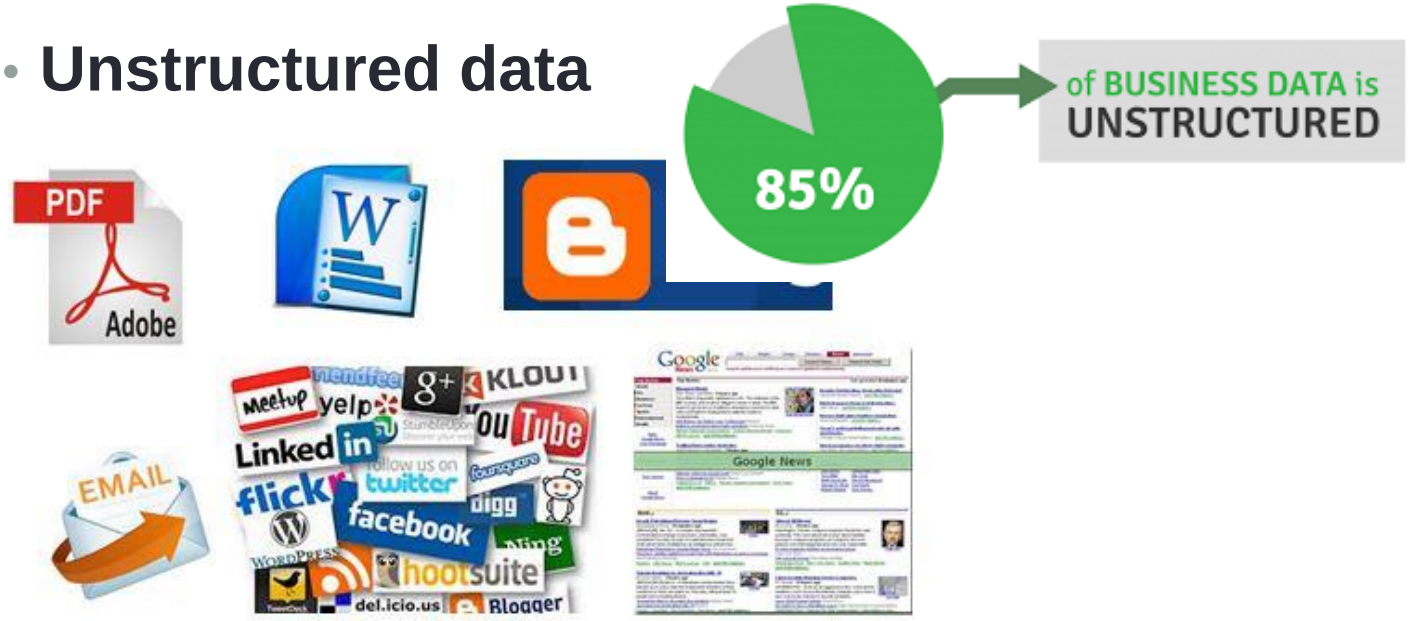
- Different Types:
  - Relational Data (Tables/Transaction/Legacy Data)
  - Text Data (Web)
  - Semi-structured Data (XML)
  - Graph Data
    - Social Network, Semantic Web (RDF), …
  - Streaming Data
    - You can only scan the data once
  - A single application can be generating/collecting many types of data
- Different Sources：
  - Movie reviews from IMDB and Rotten Tomatoes
  - Product reviews from different provider websites

A Single View to the Customer

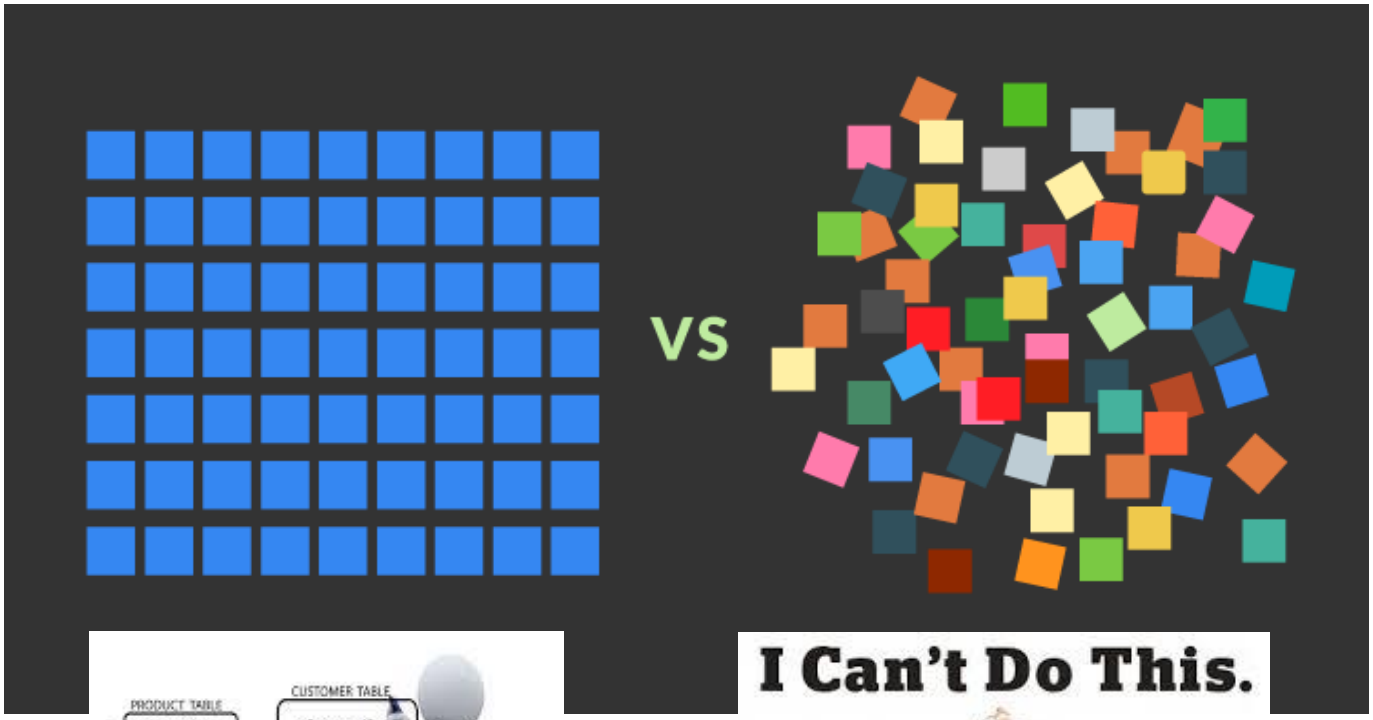
Variety --- Structured vs. Unstrucured

- Relational database (Structured data)
  - Relational table
  - Highly structured and formatted
  - Mathematical guaranteed

- **Unstructured data**



Big (Unstructured) Data



Velocity (Speed)

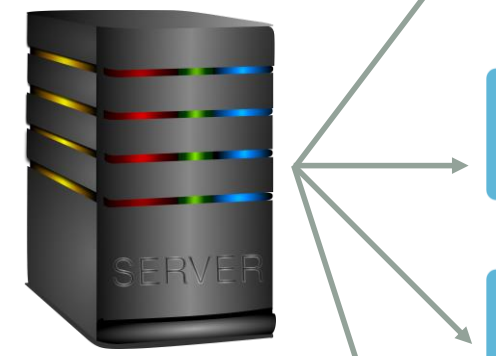
- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities
- Examples
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction
  - **Disaster management and response**

CLOUD COMPUTING

Cloud Computing

- Cloud Computing is a general term used to describe a new class of network based computing that takes place over the Internet
  - A collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform).
  - Using the Internet for communication and transport provides hardware, software and networking services to clients
  - These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API
- A technical point of view
  - Internet-based computing (i.e., computers attached to network)
- A business-model point of view
  - Pay-as-you-go (i.e., rental)

Virtualisation



16 cores
16GB memory
1024GB disk

1 core
1GB memory
64GB disk

1 core
1GB memory
64GB disk

1 core
1GB memory
64GB disk

…

1 core
1GB memory
64GB disk

Create on demand

4 cores
4GB memory
128GB disk

Return while done

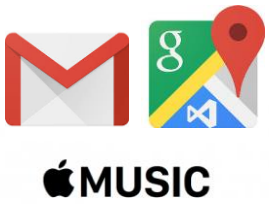
Cloud Computing

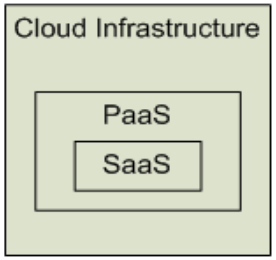
Pay-as-you-go !!!

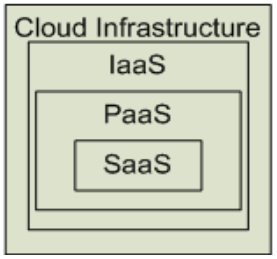
Cloud Computing Services

- Infrastructure as a service (IaaS)
  - Offering hardware related services using the principles of cloud computing. These could include storage services (database or disk storage) or virtual servers.
  - Amazon EC2, Amazon S3
- Platform as a Service (PaaS)
  - Offering a development platform on the cloud.
  - Google's Application Engine, Microsoft Azure
- Software as a service (SaaS)
  - Including a complete software offering on the cloud. Users can access a software application hosted by the cloud vendor on pay-per-use basis. This is a well-established sector.
  - Googles gmail and Microsoft hotmail, Google docs

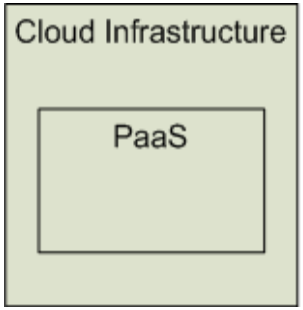
Cloud Computing as Services

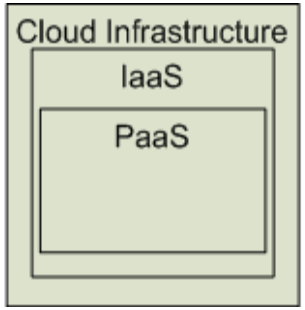






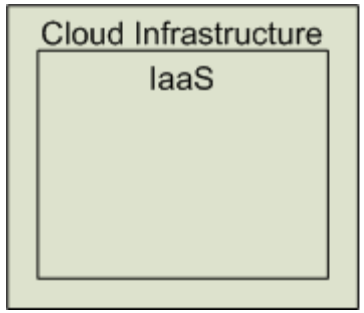
Software as a Service (SaaS) Providers Applications





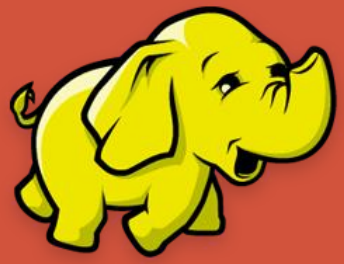
Platform as a Service (PaaS)

Deploy customer created Applications



Infrastructure as a Service (IaaS)

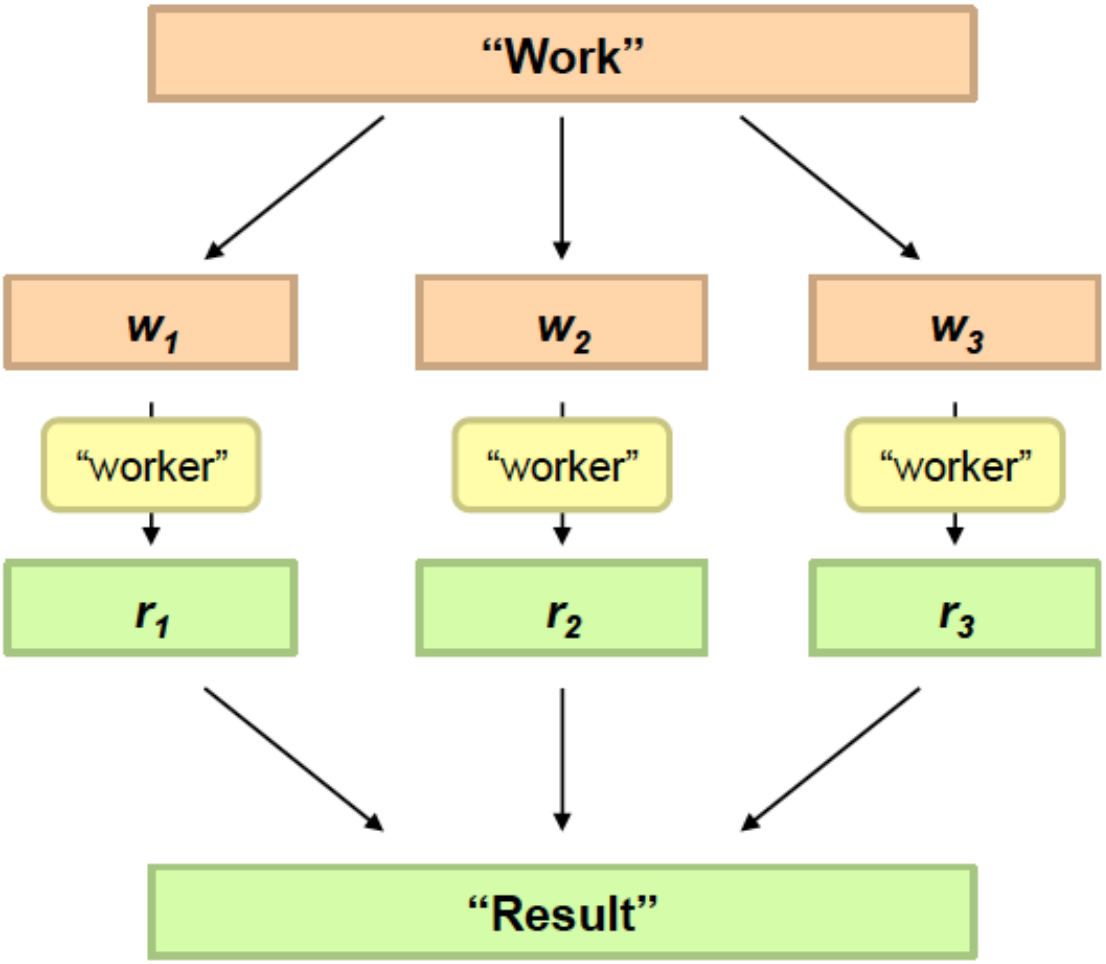
Rent Processing, storage, N/W capacity & computing resources



HADOOP

THE OPEN-SOURCE BIG DATA ENGINE

Philosophy to Scale for Big Data Processing



Divide Work

Combine Results

Distributed processing is non-trivial

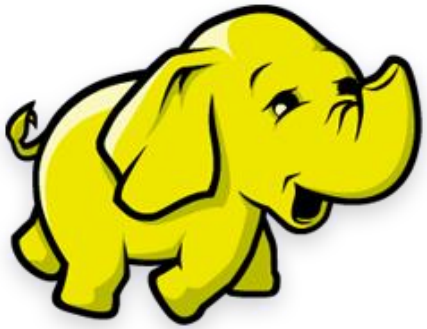
- How to assign tasks to different workers in an efficient way?
- What happens if tasks fail?
- How do workers exchange results?
- How to synchronize distributed tasks allocated to different workers?

Challenge : Failure is the Norm

# of Machines	Probability of Failure
	
	1 - (1 - 1%)^1000 = **99.996%**

What is Hadoop

- Open-source data storage and processing platform
- Before the advent of Hadoop, storage and processing of big data was a big challenge
- Massively scalable, automatically parallelizable
  - Based on work from Google
    - Google: GFS + MapReduce + BigTable (Not open)
    - Hadoop: HDFS + Hadoop MapReduce + HBase（opensource)
- Named by Doug Cutting in 2006 (worked at Yahoo! at that time), after his son's toy elephant.



Hadoop offers

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

Programmers

No longer need to worry about

Q: Where file is located?

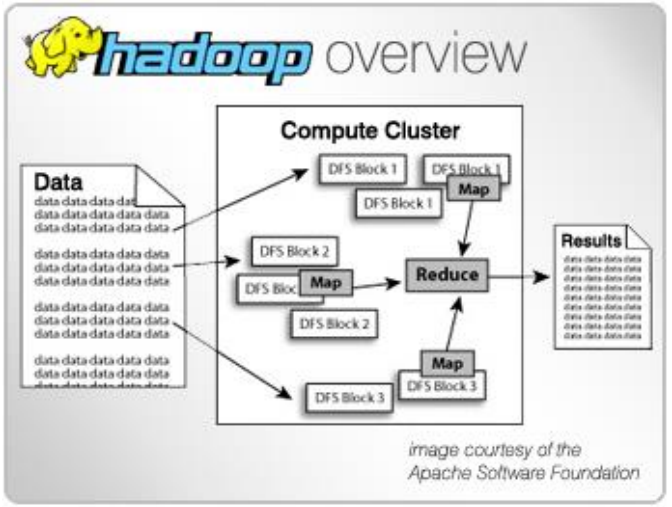
Q: How to handle failures & data lost?

Q: How to divide computation?

Q: How to program for scaling?

Why Use Hadoop?

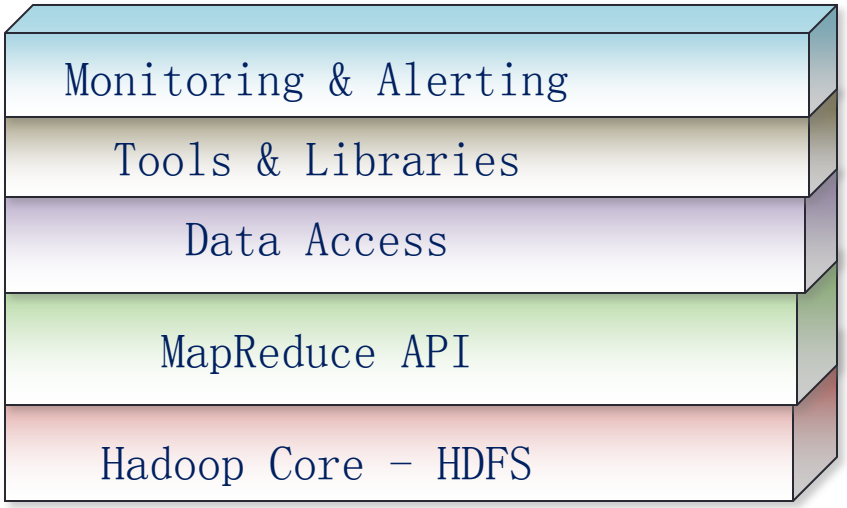
- Cheaper
  - Scales to Petabytes or more easily
- Faster
  - Parallel data processing
- Better
  - Suited for particular types of big data problems



Companies Using Hadoop

Hadoop is a set of Apache Frameworks and more…

- Data storage (HDFS)
  - Runs on commodity hardware (usually Linux)
  - Horizontally scalable
- Processing (MapReduce)
  - Parallelized (scalable) processing
  - Fault Tolerant
- Other Tools / Frameworks
  - Data Access
    - HBase, Hive, Pig, Mahout
  - Tools
    - Hue, Sqoop
  - Monitoring
    - Greenplum, Cloudera



MAPREDUCE

A big-data programming model

What is MapReduce

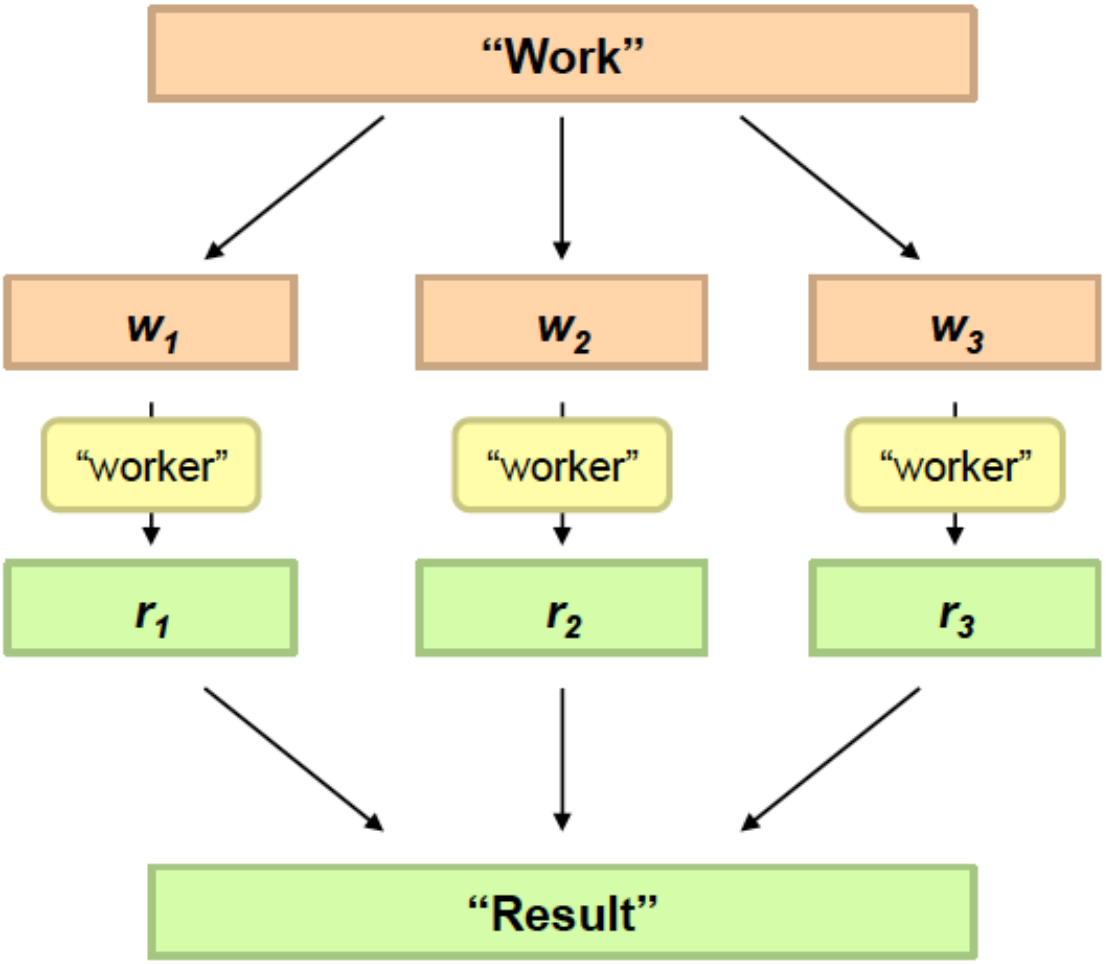
- Origin from Google, [OSDI'04]

  MapReduce: Simplified Data Processing on Large Clusters

  Jeffrey Dean and Sanjay Ghemawat
- A programming model for parallel data processing
- Hadoop can run MapReduce programs written in various languages:
  e.g. Java, Ruby, Python, C++

- We introduce basic principles for MapReduce only in this course. *If interested: COMP9313 – Big Data Management*

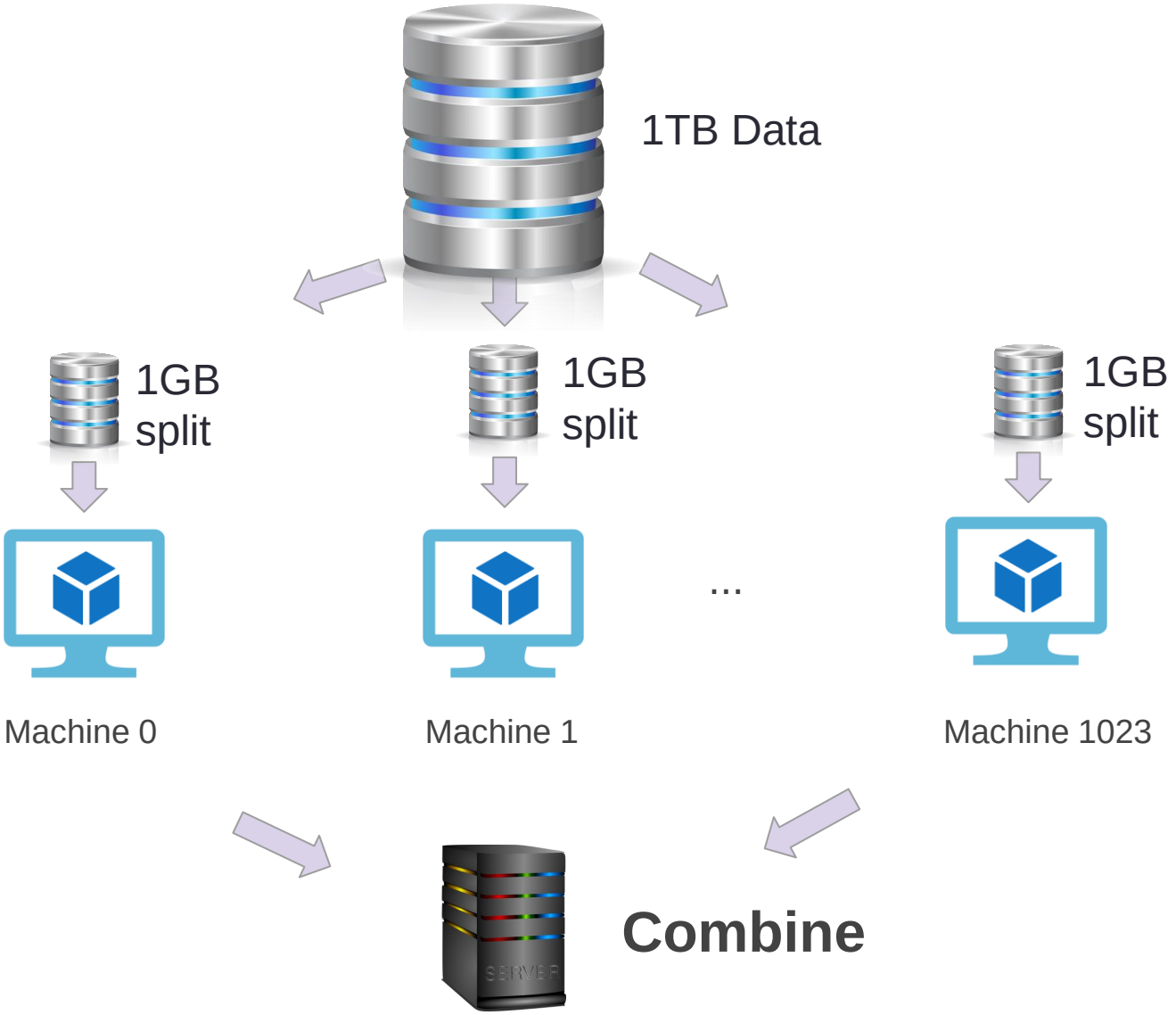
Philosophy Revisit



Divide Work

Combine Results

Divide and Combine



Typical Big Data Problem

- Iterate over a large number of records
- Extract something of interest
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Map

Reduce

Key idea: provide a functional abstraction for these two operations

Map and Reduce Functions

- Programmers specify two functions:
  - **map** $(k_1, v_1) \rightarrow$ list $[<k_2, v_2>]$
    - Map transforms the input into key-value pairs to process
  - **reduce** $(k_2,$ list $[v_2]) \rightarrow [<k_3, v_3>]$
    - Reduce aggregates the list of values for each key
    - All values with the same key are sent to the same reducer
  - list $[<k_2, v_2>]$ will be grouped according to key k_2 as $(k_2,$ list $[v_2])$

- The MapReduce environment takes in charge of everything else

A WordCount Example

- Given a file, count the occurrence of each word

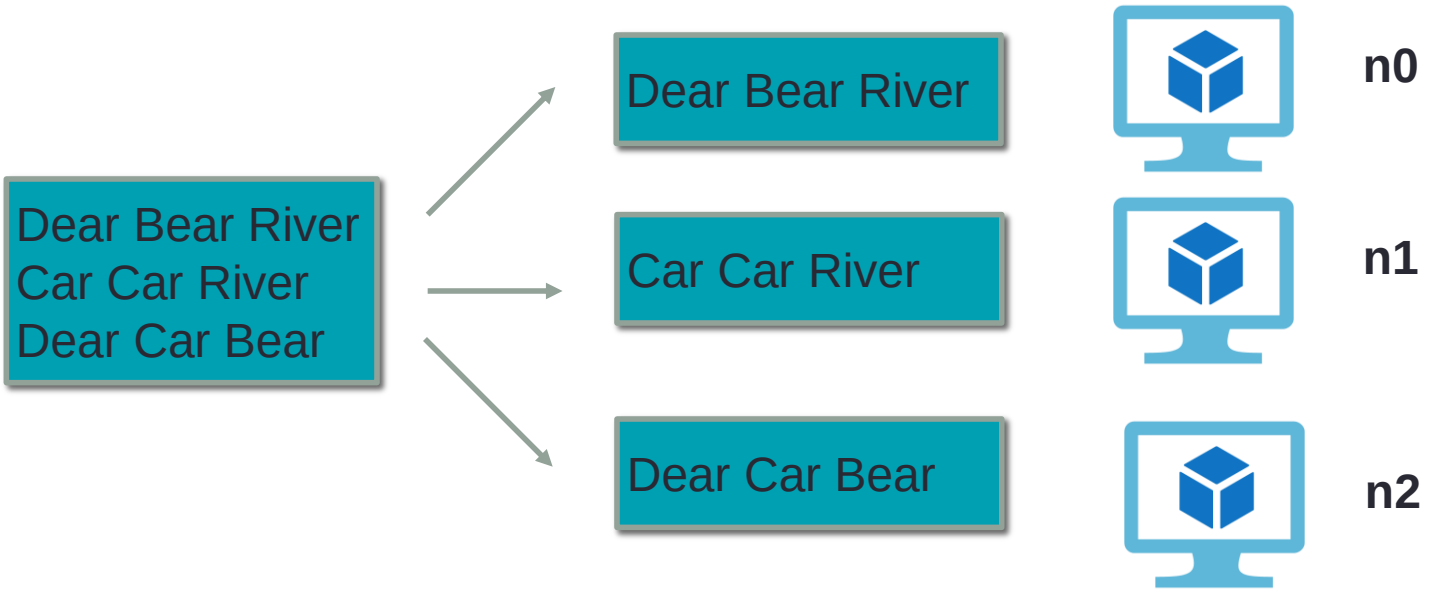
Data File

Dear Bear River
Car Car River
Dear Car Bear

- *E.g., Car* occurs three times
- Imagine we have a file of 1TB

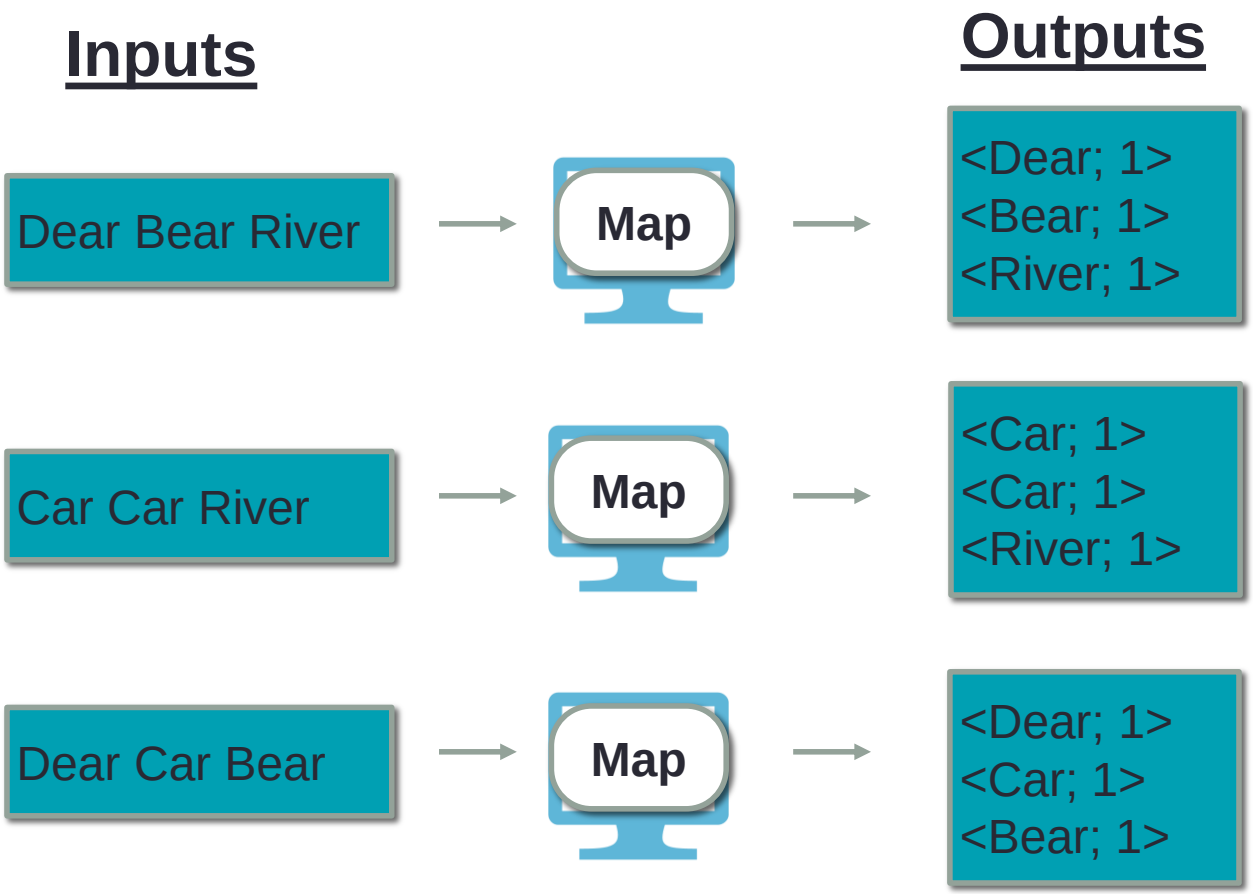
Step 1: Splitting

- Original file is split into smaller pieces



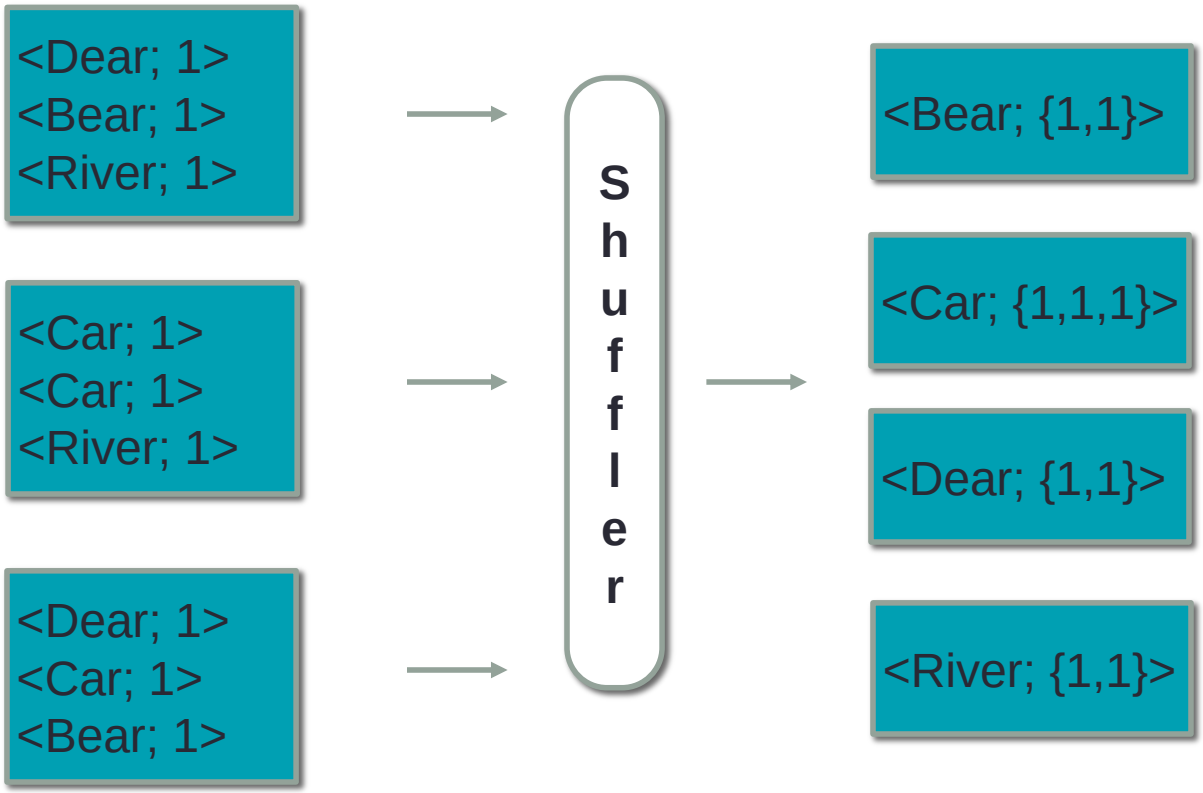
Step 2: Mapping

- Read the file piece, and generate <key; value> pairs by extracting words from lines. Here value is 1 since each word read means it occurs once



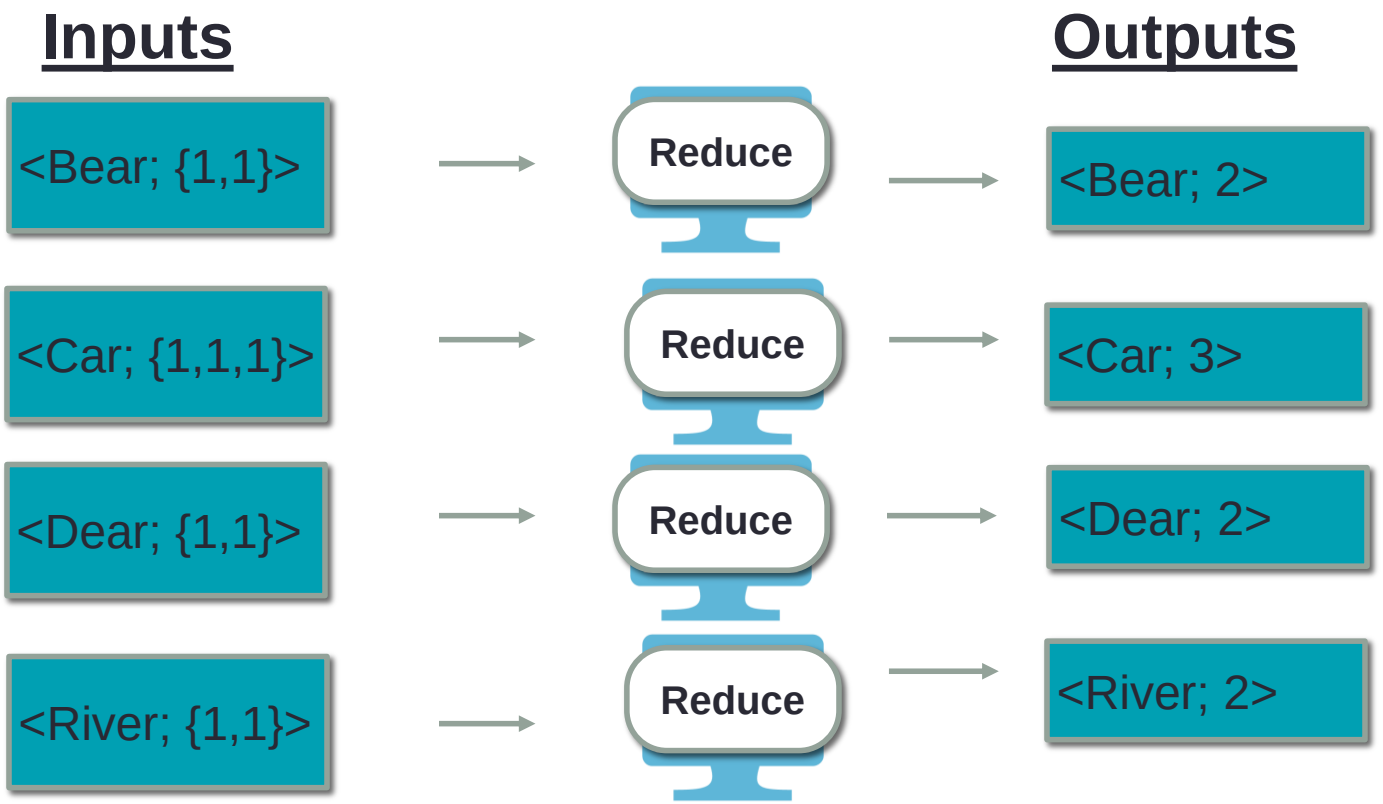
Step 3: Shuffling

- *Hadoop framework handles this step*. Sort the key value pairs and group based on the key value

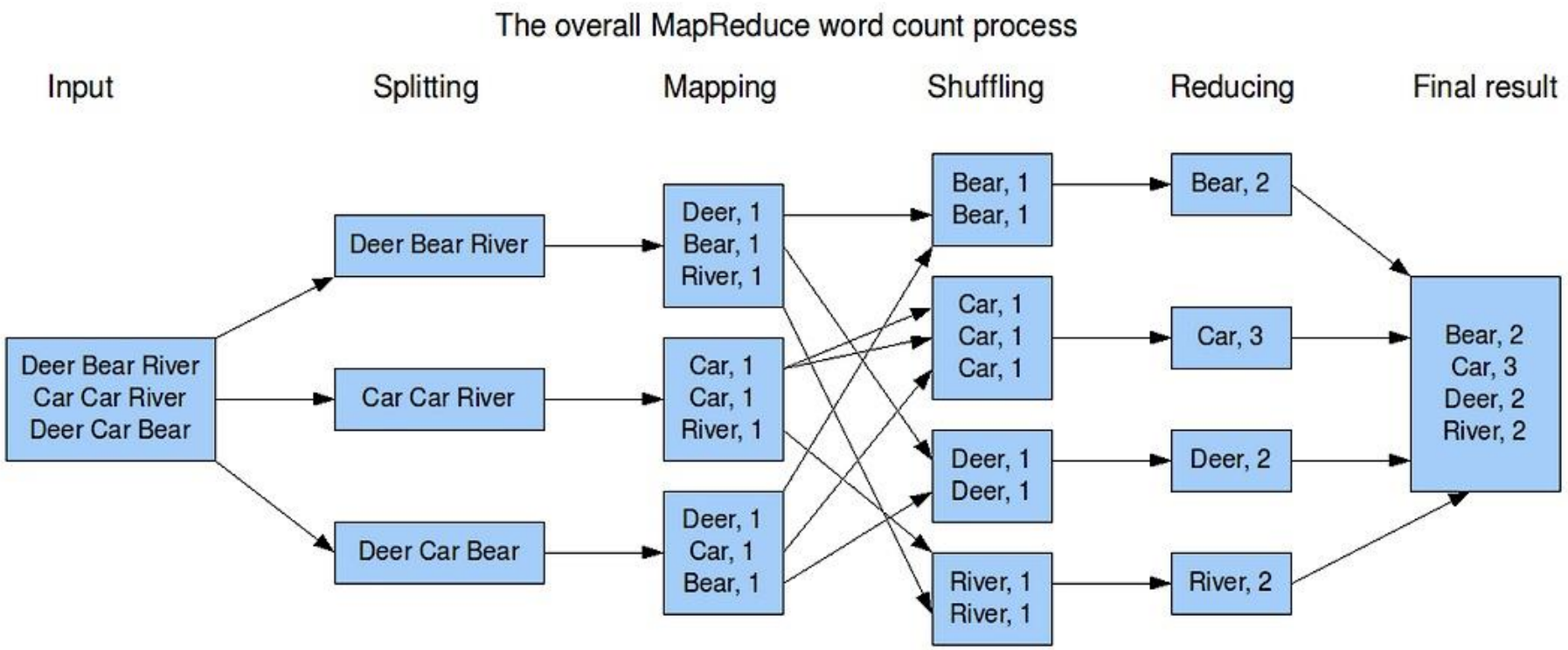


Step 4: Reducing

- Reducer aggregate the list of values of each key and output the result



MapReduce Example - WordCount



“Word Count” in MapReduce

Mapper and Reducer function

```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term t ∈ doc d do
4:             Emit(term t, count 1)

1: class Reducer
2:     method Reduce(term t, counts [c1, c2, ...])
3:         sum ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             sum ← sum + c
6:         Emit(term t, count s)
```

MapReduce

- WordCount example counts the occurrence of each word.
- We can also get other aggregate values, such as sum and maximum.

This part will not be covered in the final exam