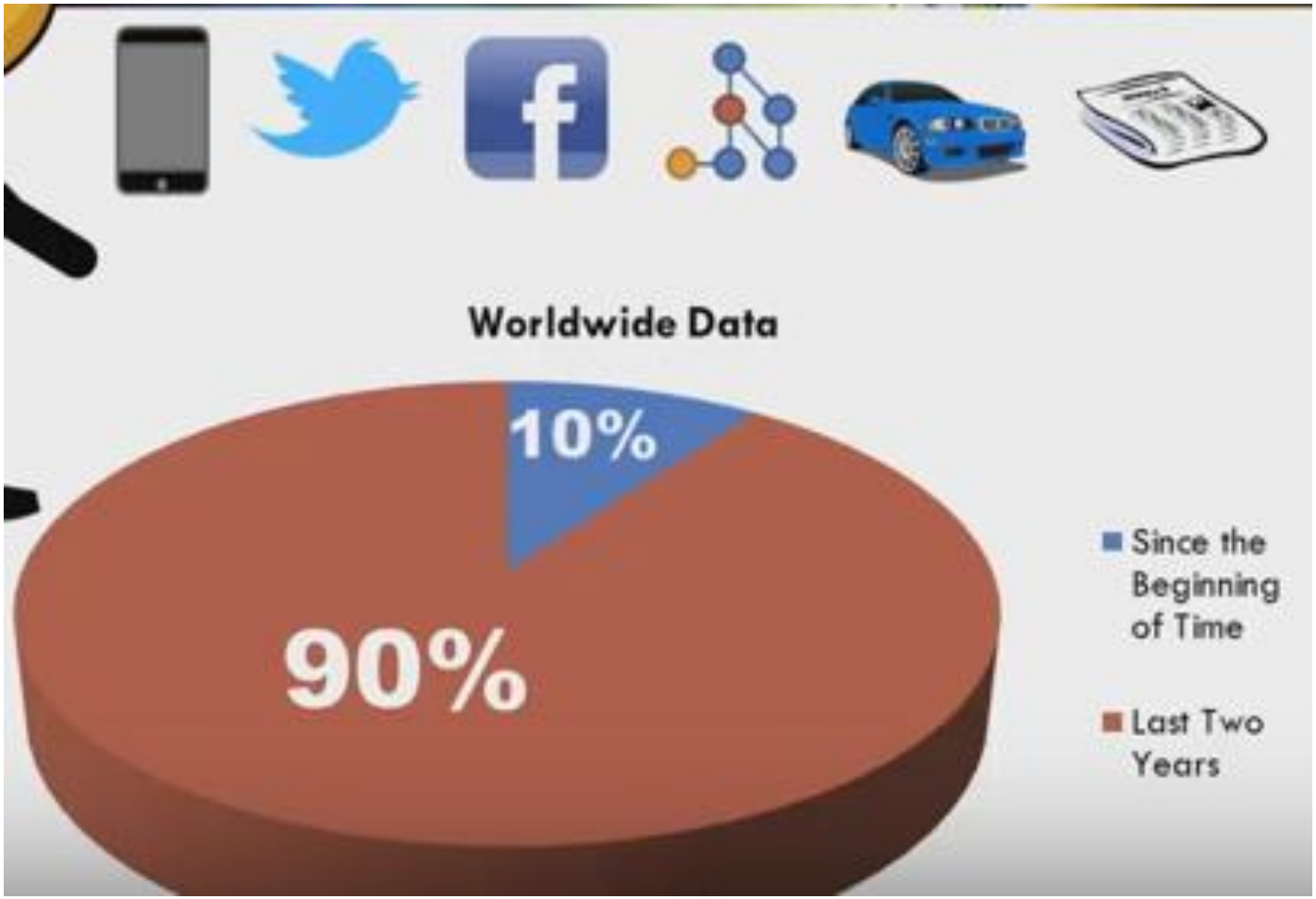
Database & Storage Security

Professor Dr. Mohammad Abu Yousuf
yousuf@juniv.edu

Big Data Security

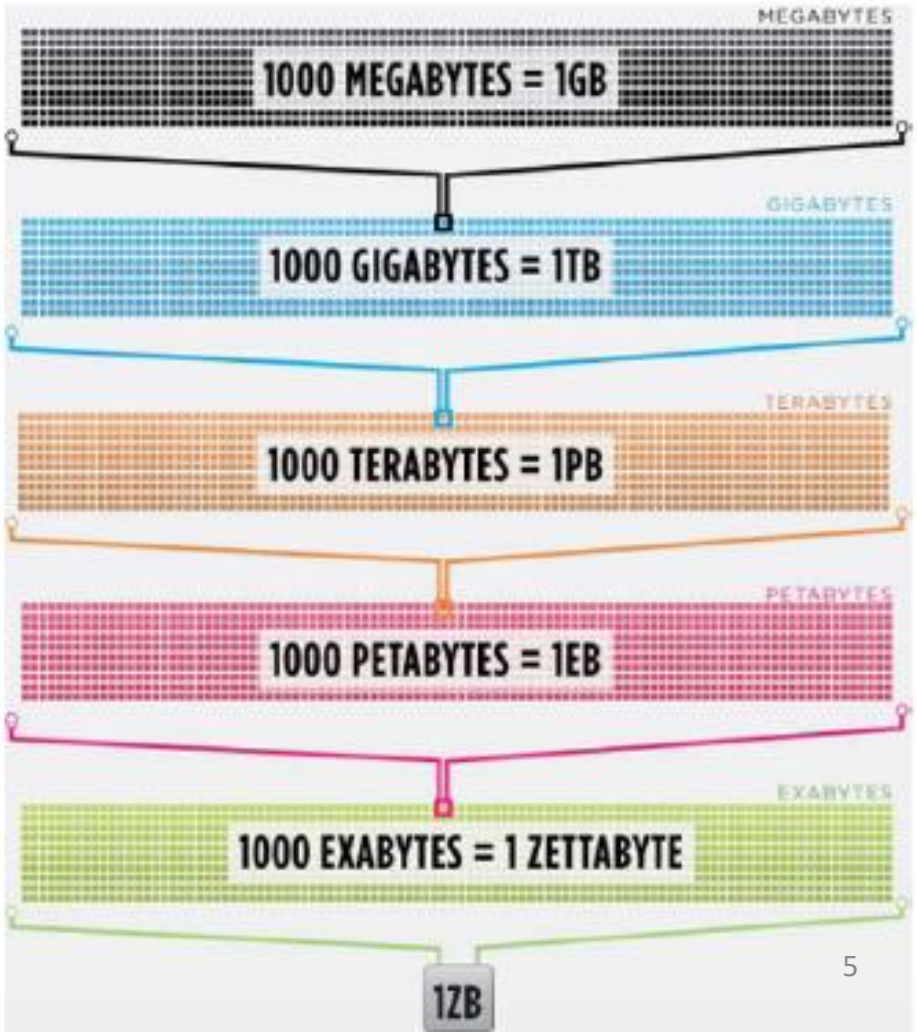
Today's Topics


Worldwide Data
10%
90%
Since the Beginning of Time
Last Two Years

What is Big Data?

Definitions

- **Big data**
  - Datasets so large/complex they become difficult to work with using existing technology
- **Big data technology**
  - Specialized technology developed to manage large/complex data sets



What is Big Data?

- Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques.

- Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.

What is Big Data?

- In Simple Words, Big Data is a technique to solve data problems that are not solvable using Traditional Databases and Tools.

- In other way, BigData means not just huge amount of Data. BigData means huge amount of data generating at very fast rate in different formats.

- Big Data is a Technique to “Store, Process, Manage, Analysis and Report” a huge amount of variety data, at the required speed, and within the required time to allow Real-time Analysis and Reaction.

Characteristics of Big Data

- Big Data is Data with has the following three characteristics:

✓ Extremely Large Volumes of Data

✓ Extremely High Velocity of Data

✓ Extremely Wide Variety of Data

Characteristics of Big Data

- **Volume:** Very vast amount of Data say TB(Tera Bytes) to PB(Peta Bytes) to Exa Byte(EB) and more

- **Velocity:** Velocity means "How fast produce Data".

- **Variety:** Variety means "Different forms of Data".

Types of Data

- Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.
- **Structured data** : Structured Data means Data that is in the form of Rows and Columns. So it is very easy to store even in Relational Databases.

  Example: Relational data.
- **Semi Structured data** : Semi-Structured Data means Data that is formatted in some way. But it is not formatted in the form of Rows and Columns.

  Example: XML data.
- **Unstructured data** : Un-Structured Data means Data that is not formatted in any way. It is not possible to store data in Relational Databases.

  Example: Audio files, Videos, Photos, Sensor Data, Web Data,Mobile Data,GPS Data.

Types of Data

Multi-Structured Data = Structured Data
+ Semi-Structured Data
+ Un-Structured Data

What Comes Under Big Data?

- Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.
- **Black Box Data** : It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- **Social Media Data** : Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
- **Stock Exchange Data** : The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.
- **Power Grid Data** : The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data** : Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data** : Search engines retrieve lots of data from different databases.

Big Data life cycle management

Big Data Technologies

- To harness the power of big data, we would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real time and can protect data privacy and security.

- There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. While looking into the technologies that handle big data, we examine the following two classes of technology:

Big Data Technologies

- Operational Big Data

  This include systems like **MongoDB** that provide operational capabilities for real-time, interactive workloads where data is primarily captured and stored.

- Analytical Big Data

  This includes systems like Massively Parallel Processing (MPP) database systems and **MapReduce** that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

MongoDB

- MongoDB is a document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database:

- Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
- A record in MongoDB is a document, which is a data structure composed of field and value pairs.

- https://www.youtube.com/watch?v=CvIr-2lMLsk

MongoDB

Below given table shows the relationship of RDBMS terminology with MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)

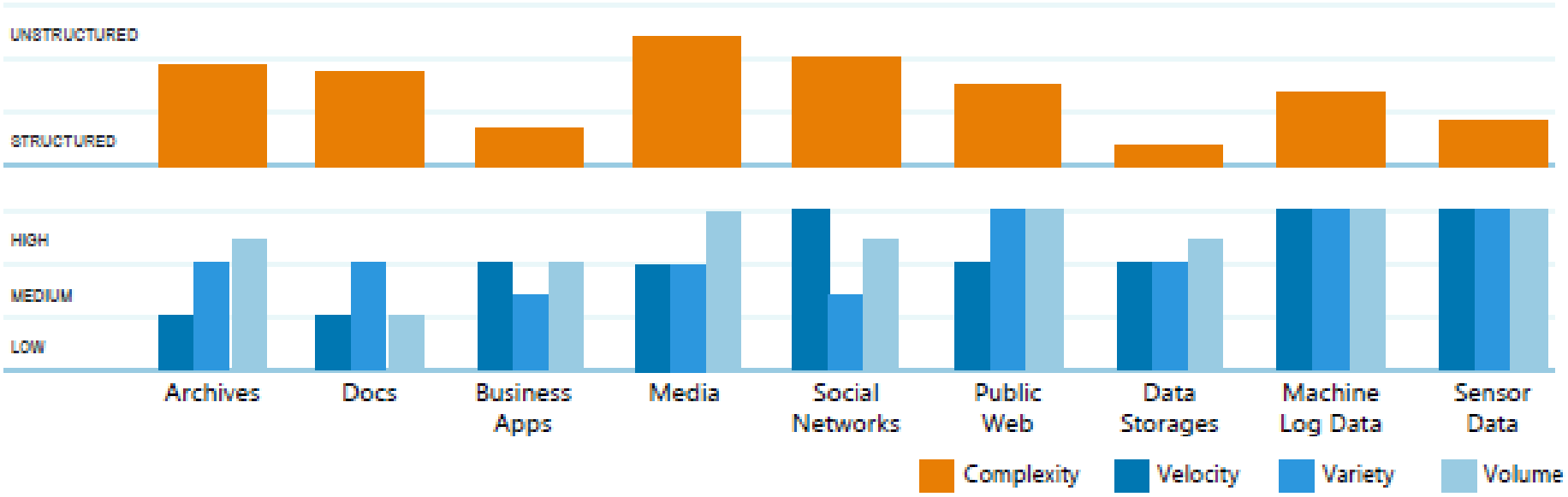
Big Data Technologies

- Different Big Data and Cloud Platforms Solutions available in the current market:

- ❑ Amazon Web Services (AWS),
- ❑ Google Cloud Platform,
- ❑ Microsoft Azure,
- ❑ IBM Bluemix,
- ❑ Pivotal Cloud Foundry,
- ❑ Yahoo Cloud Platform etc.

Big Data Challenges

Big Data Challenges



Archives
Scanned documents, statements, medical records, e-mails etc..

Docs
XLS, PDF, CSV, HTML, JSON etc.

Business Apps
CRM, ERP systems, HR, project management etc.

Media
Images, video, audio etc.

Social Networks
Twitter, Facebook, Google+, LinkedIn etc.

Public Web
Wikipedia, news, weather, public finance etc

Data Storages
RDBMS, NoSQL, Hadoop, file systems etc.

Machine Log Data
Application logs, event logs, server data, CDRs, clickstream data etc.

Sensor Data
Smart electric meters, medical devices, car sensors, road cameras etc.

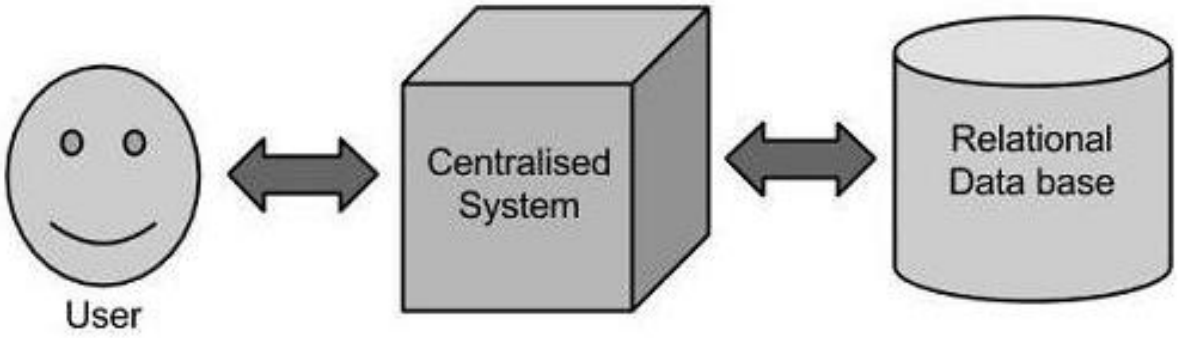
Big Data Challenges

- **Big Data Challenges**

  The major challenges associated with big data are as follows:
  - Capturing data
  - Curation
  - Storage
  - Searching
  - Sharing
  - Transfer
  - Analysis
  - Presentation

Traditional Approach

- In this approach, an enterprise will have a computer to store and process big data. Here data will be stored in an RDBMS like Oracle Database, MS SQL Server or DB2 and sophisticated software can be written to interact with the database, process the required data and present it to the users for analysis purpose.



Traditional Approach

Limitation:

- Traditional approach works well where we have less volume of data that can be accommodated by standard database servers, or up to the limit of the processor which is processing the data.

- But when it comes to dealing with huge amounts of data, it is really a tedious task to process such data through a traditional database server.

MapReduce Algorithm

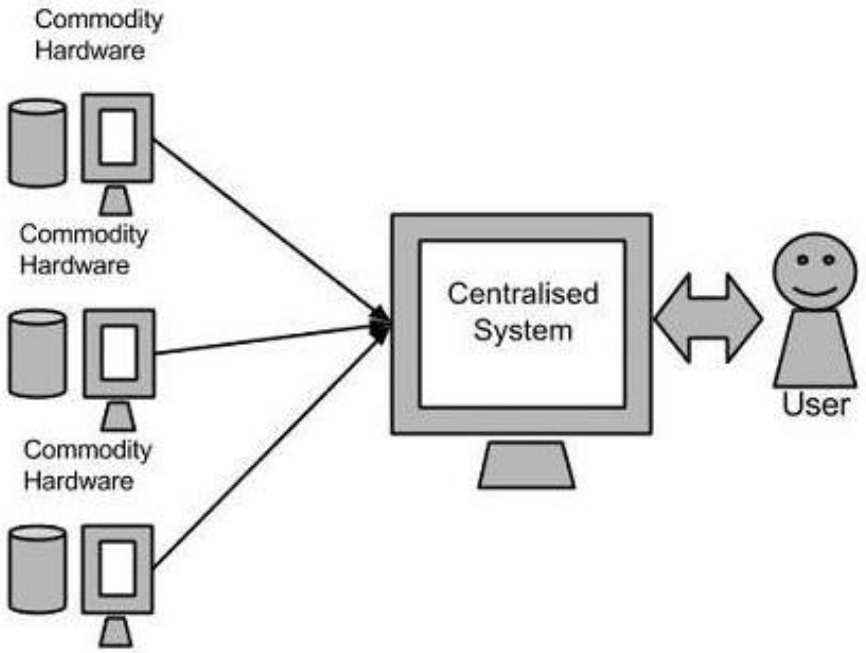
Google's Solution:

Google solved this problem using an algorithm called MapReduce.

MapReduce Algorithm

Google's Solution

- This algorithm divides the task into small parts and assigns those parts to many computers connected over the network, and collects the results to form the final result dataset.



- Above diagram shows various commodity hardware which could be single CPU machines or servers with higher capacity.

Hadoop- Big Data Solutions

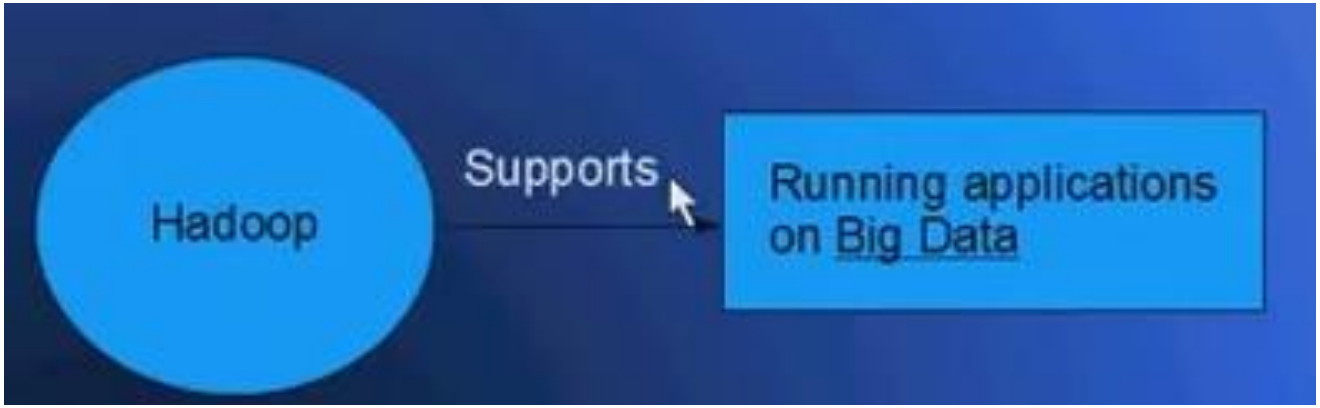
- Map reduce algorithm was used to develop an open-source framework called Hadoop is that allows to store and process big data in a distributed environment across clusters of computers using simple programming models.

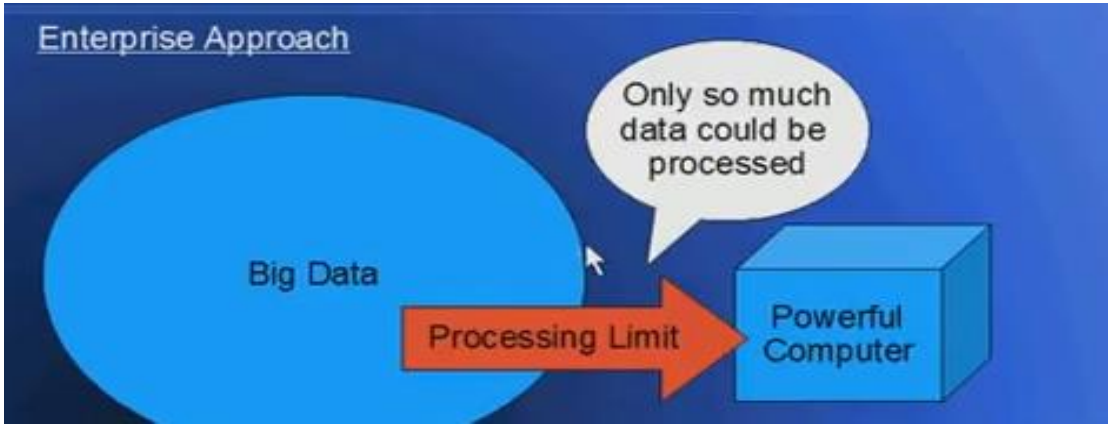
- It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

- Hadoop is an Open Source framework from Apache Software Foundation to solve BigData Problems. It is completely written in Java Programming Language.

Hadoop- Big Data Solutions

- Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.




Enterprise Approach
Only so much data could be processed
Big Data
Processing Limit
Powerful Computer


Hadoop's Approach
Big Data
Is broken into pieces

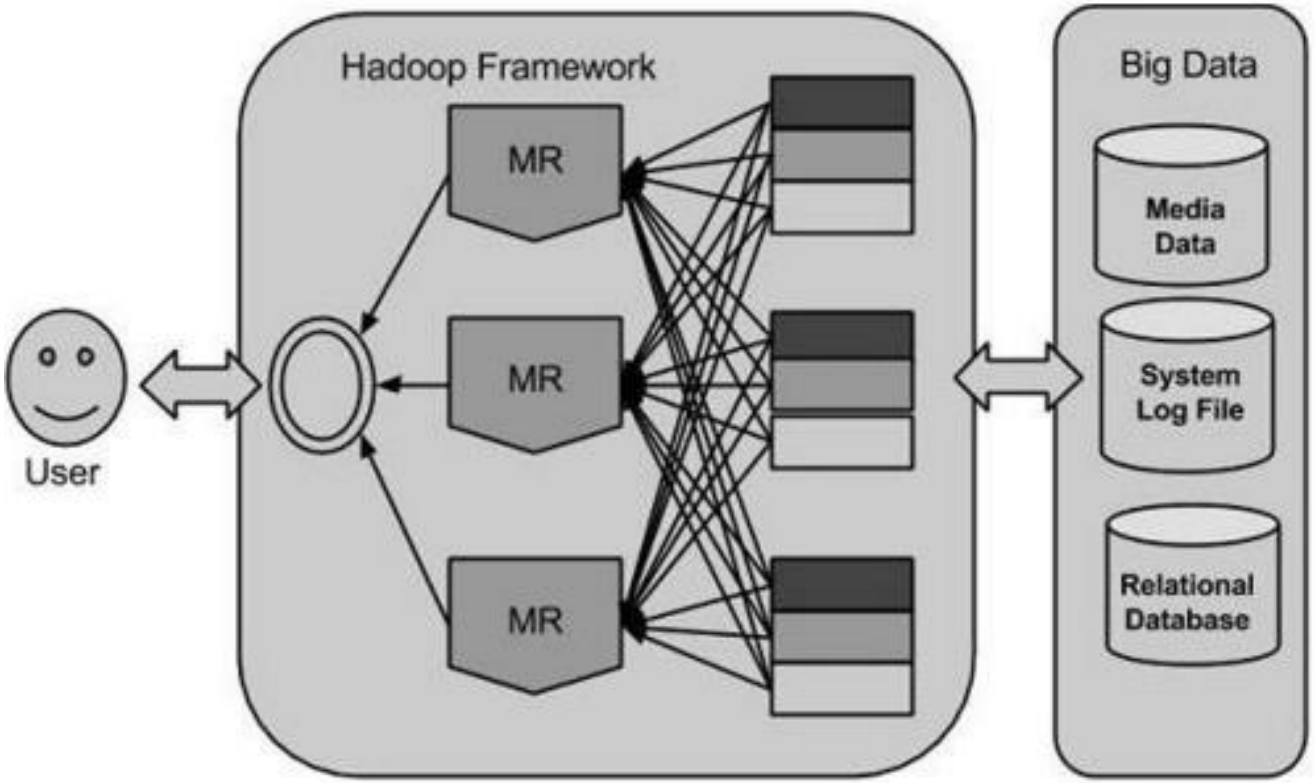

Hadoop's Approach
Computation
Combined Result
Computation
Big Data
Computation

Hadoop- Big Data Solutions

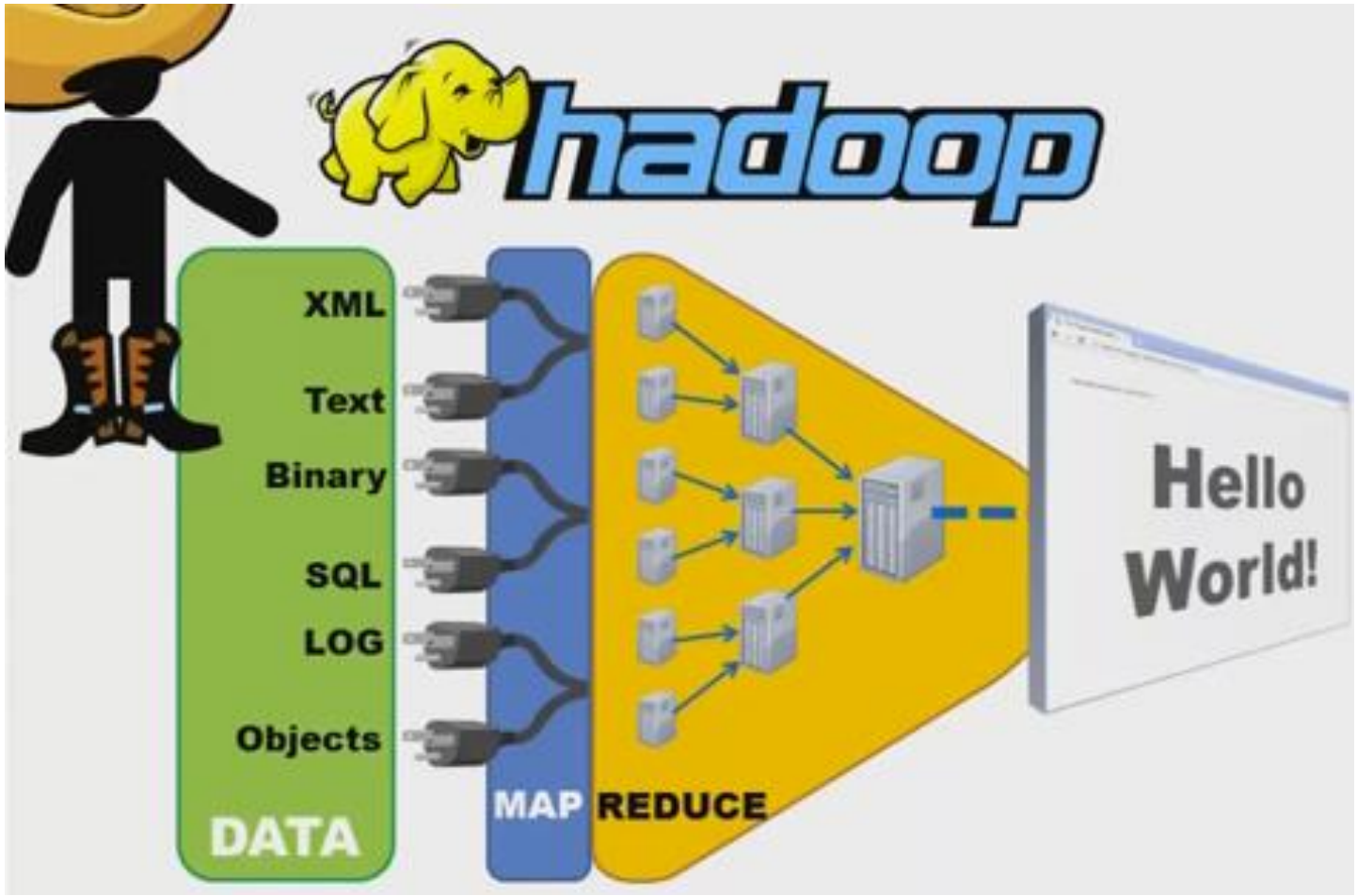
- **Doug Cutting**, **Mike Cafarella** and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant.

- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capabale enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data (fig: next slide).

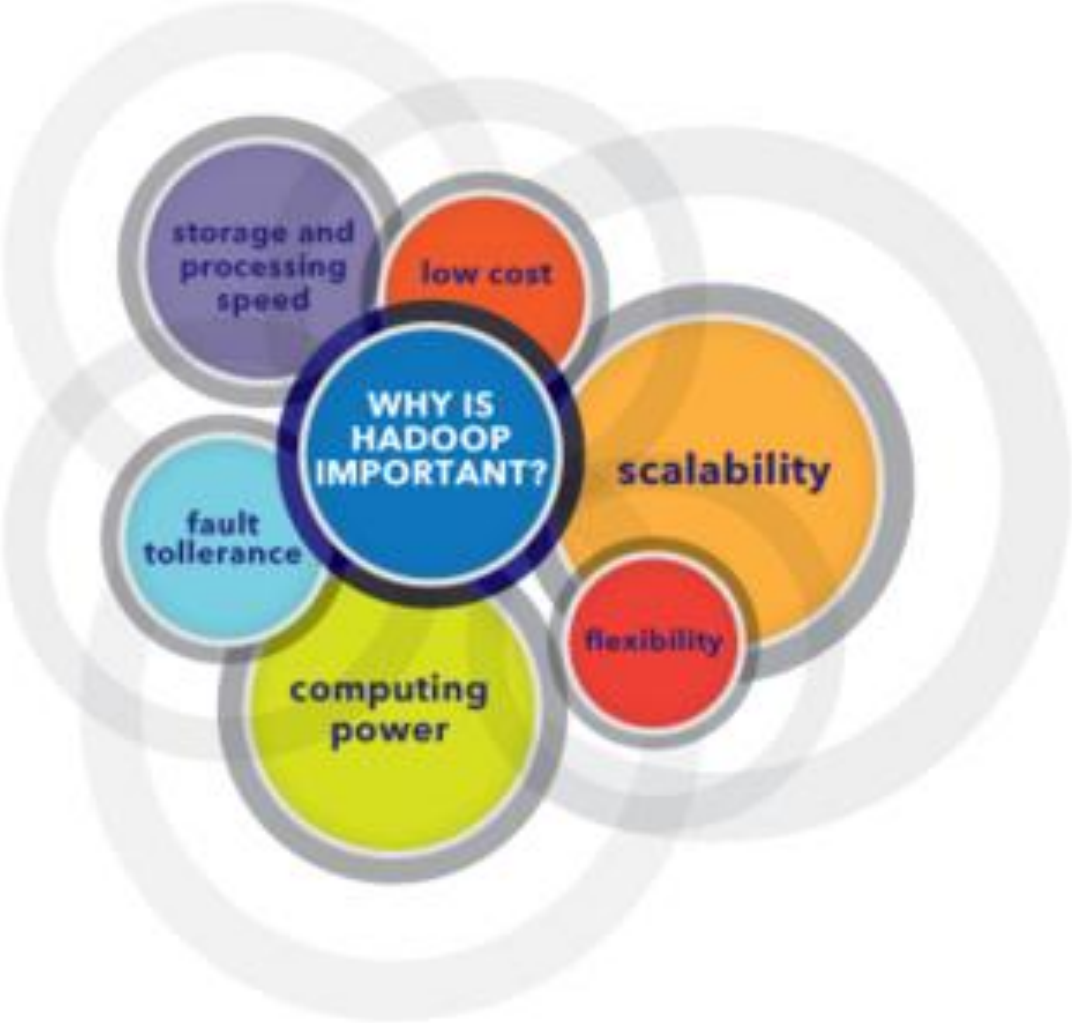
Hadoop- Big Data Solutions



Hadoop- Big Data Solutions



Why is Hadoop important?



Why is Hadoop important?

- **Ability to store and process huge amounts of any kind of data, quickly.**
- **Computing power.** Hadoop's distributed computing model processes big data fast.
- **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
- **Flexibility.** Unlike traditional relational databases, you don’t have to preprocess data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like images and videos.
- **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Scalability.** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

Hadoop Architecture

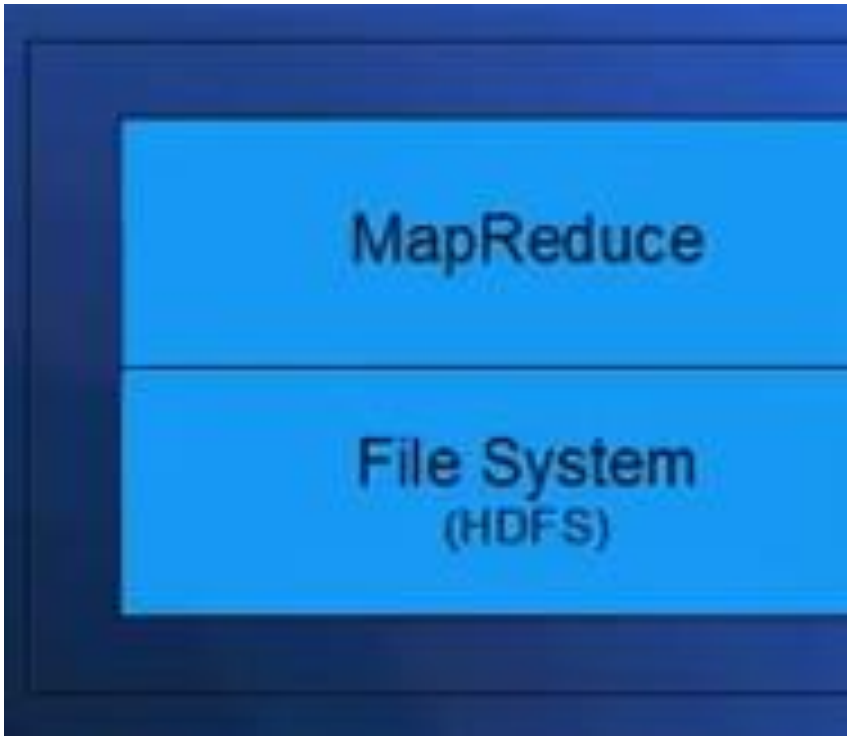
- Hadoop framework includes following four modules:
- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides file system and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.

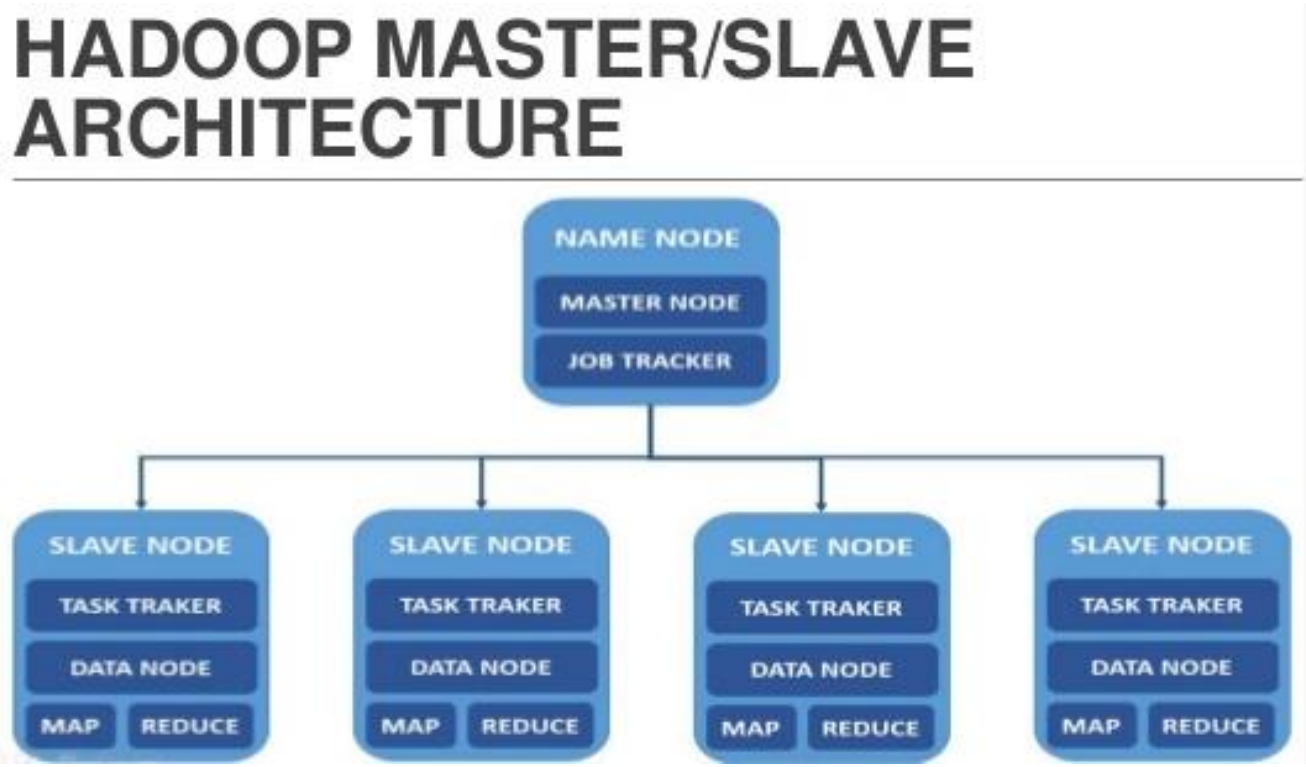
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.

- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

Hadoop Architecture

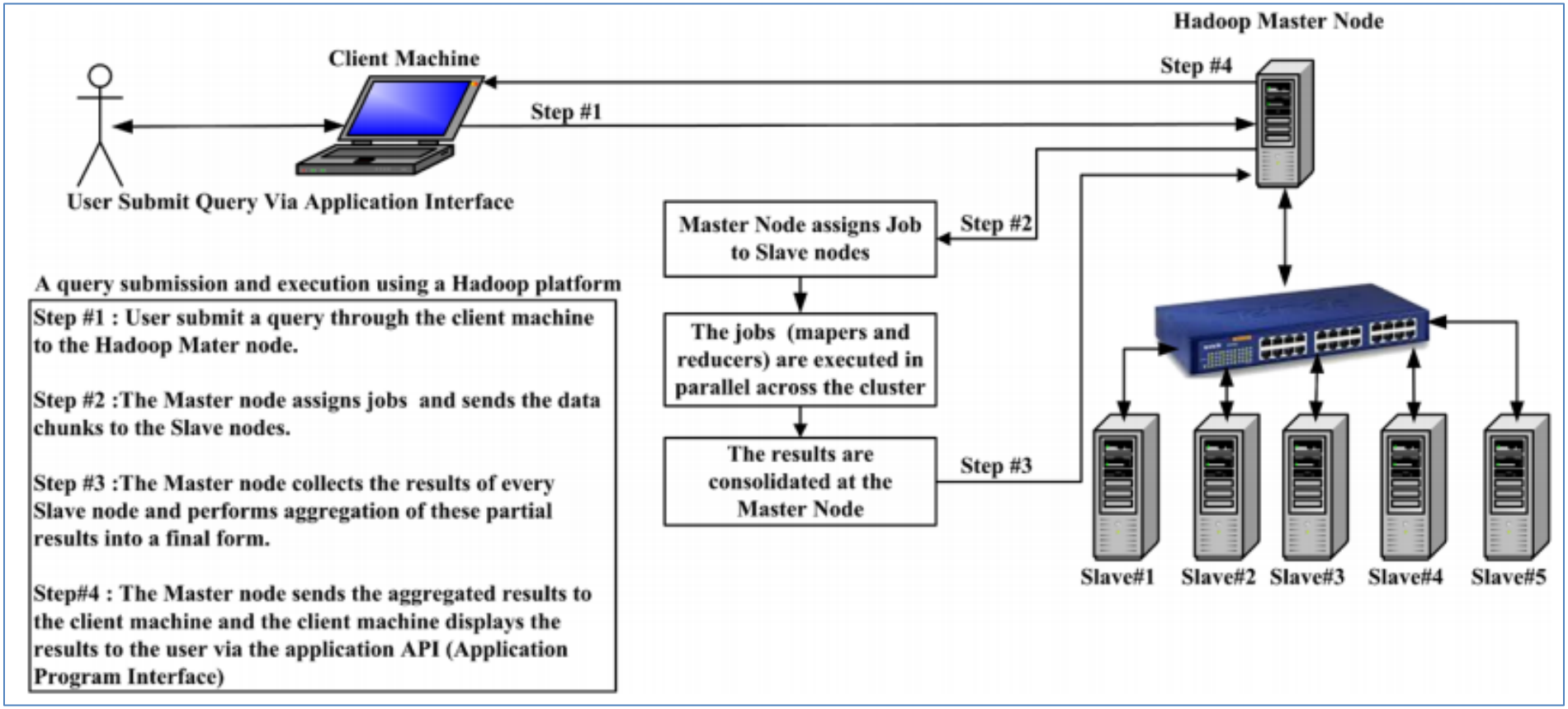


Hadoop Architecture



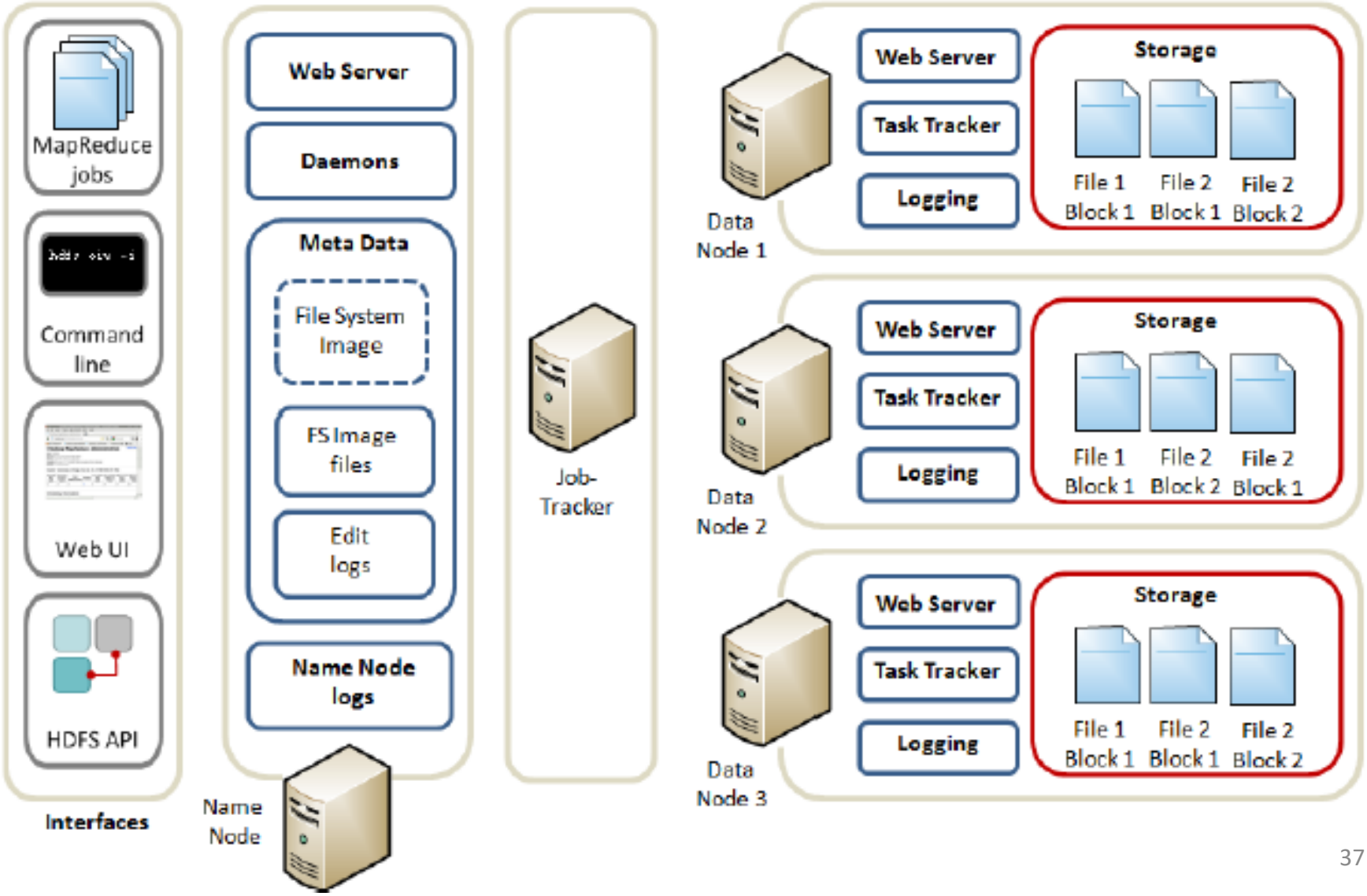
HDFS comprises of 2 important components-NameNode, and DataNode. HDFS operates on a Master-Slave architecture model where the NameNode acts as the master node for keeping a track of the storage cluster and the DataNode acts as a slave node summing up to the various systems within a Hadoop cluster.

Hadoop Architecture



The architecture of the Hadoop cluster. Hadoop cluster architecture, showing the distributed computing nodes, which are Master node (NameNode), Slave Nodes (DataNode), and the Ethernet switch

Hadoop Architecture



MapReduce

MapReduce

- While processing large set of data, we should definitely address scalability and efficiency in the application code that is processing the large amount of data.
- Map reduce algorithm (or flow) is highly effective in handling big data.

Illustration of MapReduce

MapReduce

- Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.
- The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:
- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Illustration of MapReduce

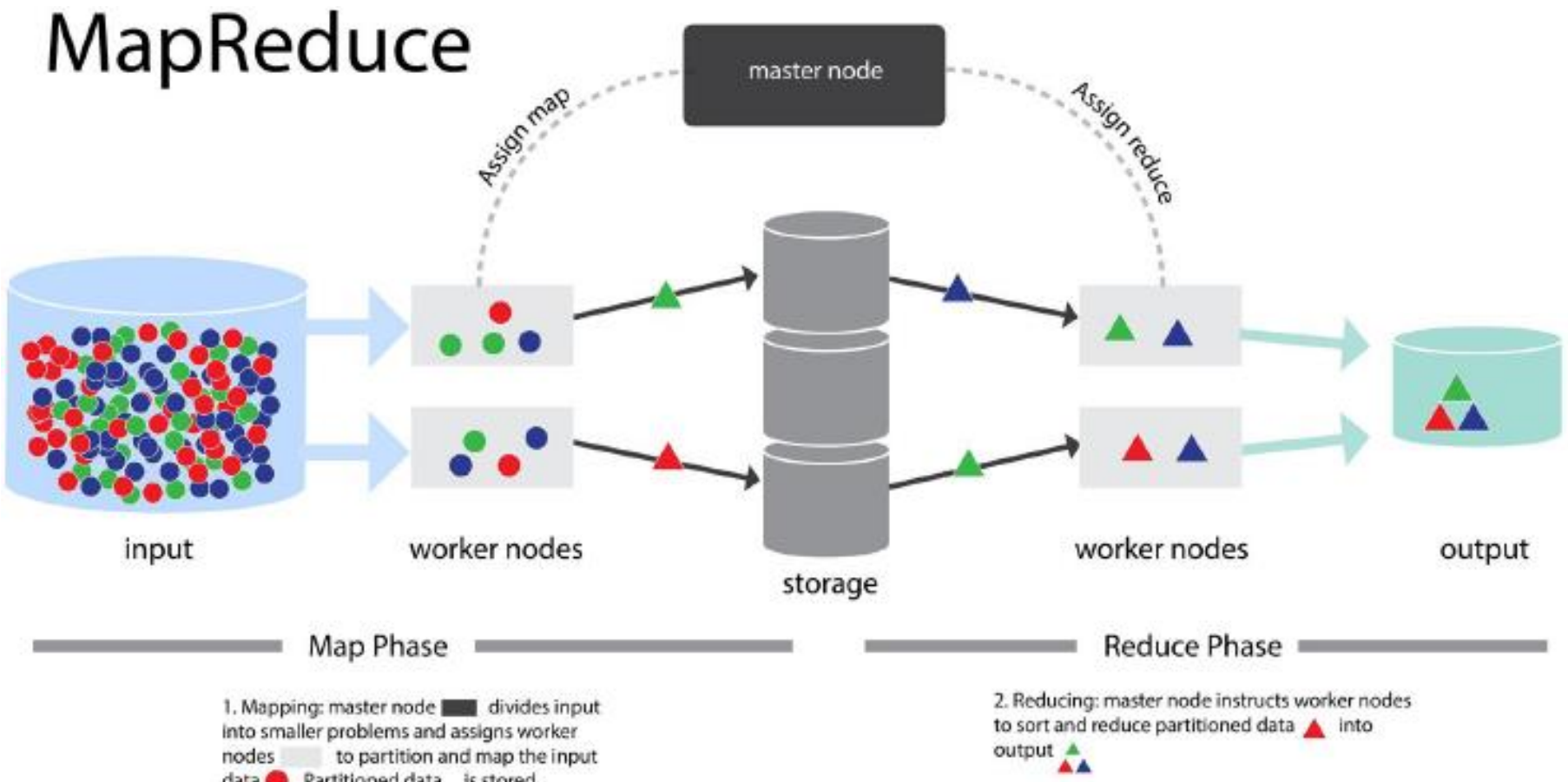


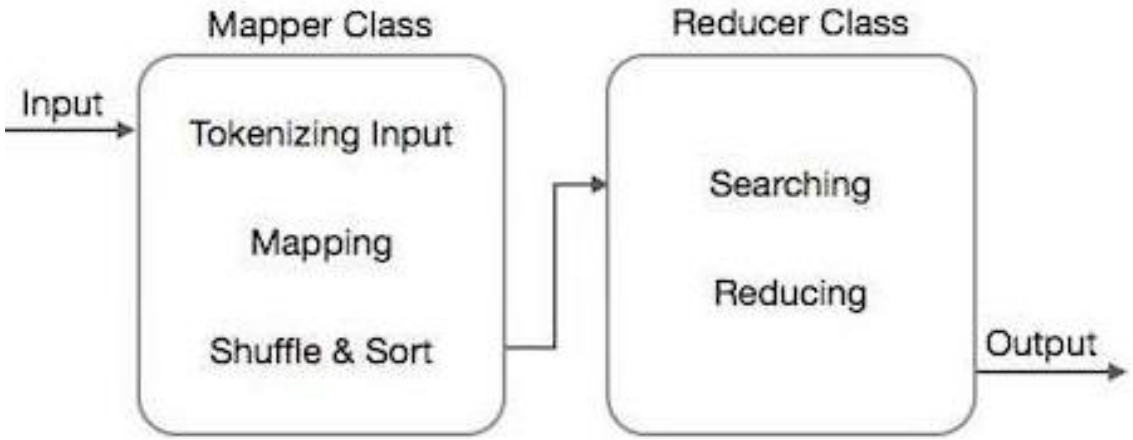
Illustration of MapReduce

- The MapReduce algorithm contains two important tasks, namely Map and Reduce.

  1)The map task is done by means of Mapper Class.

  2) The reduce task is done by means of Reducer Class.

- Mapper class takes the input, tokenizes it, maps and sorts it. The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.



MapReduce Algorithm Steps

- MapReduce Algorithm uses the following three main steps:

  1) Map Function

  2) Shuffle Function

  3) Reduce Function

MapReduce Algorithm Steps

- **Map Function**

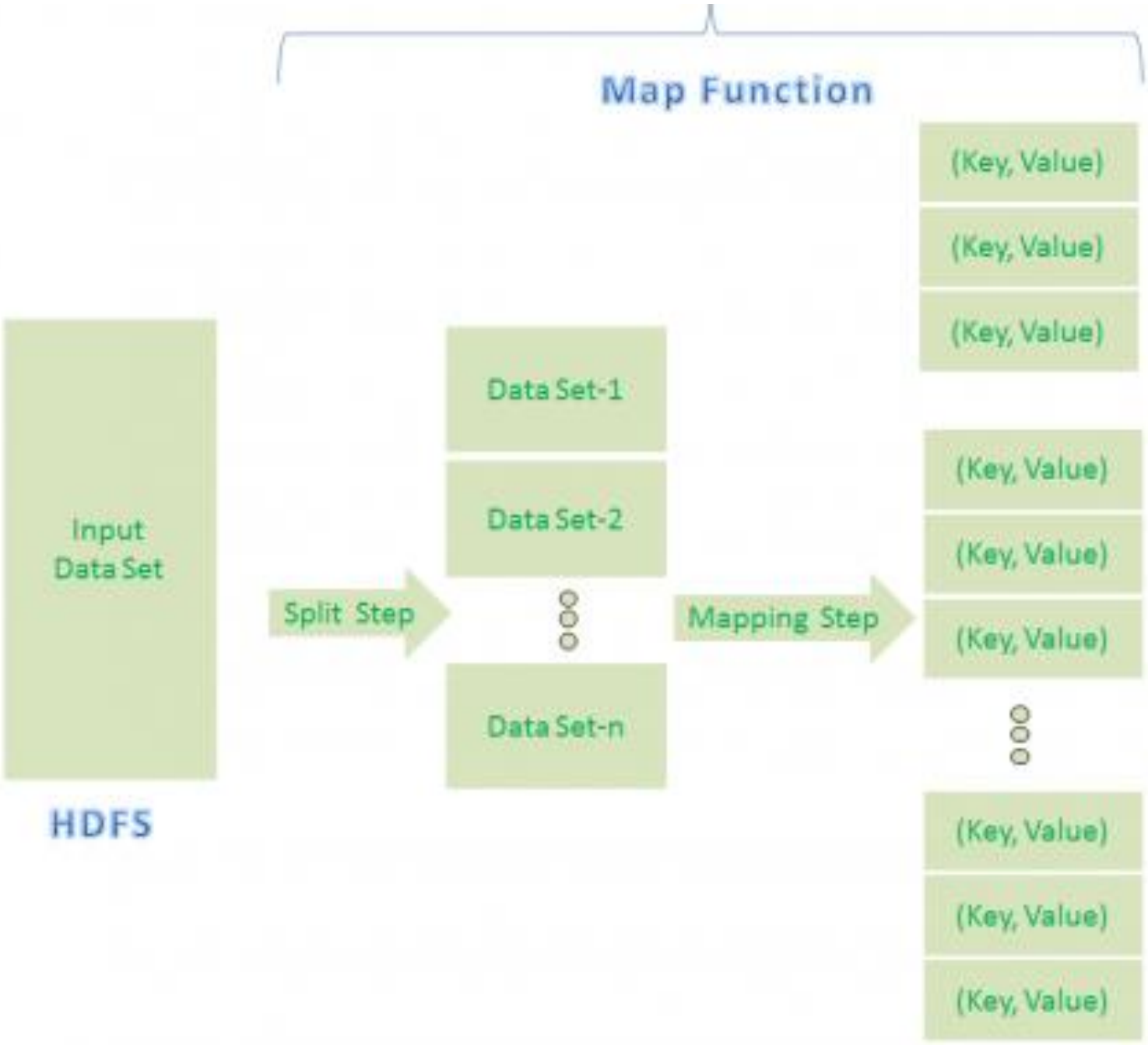
  Map Function is the first step in MapReduce Algorithm. It takes input tasks and divides them into smaller sub-tasks. Then perform required computation on each sub-task in parallel.

MapReduce Algorithm Steps

- Map function step performs the following two sub-steps:

  Splitting

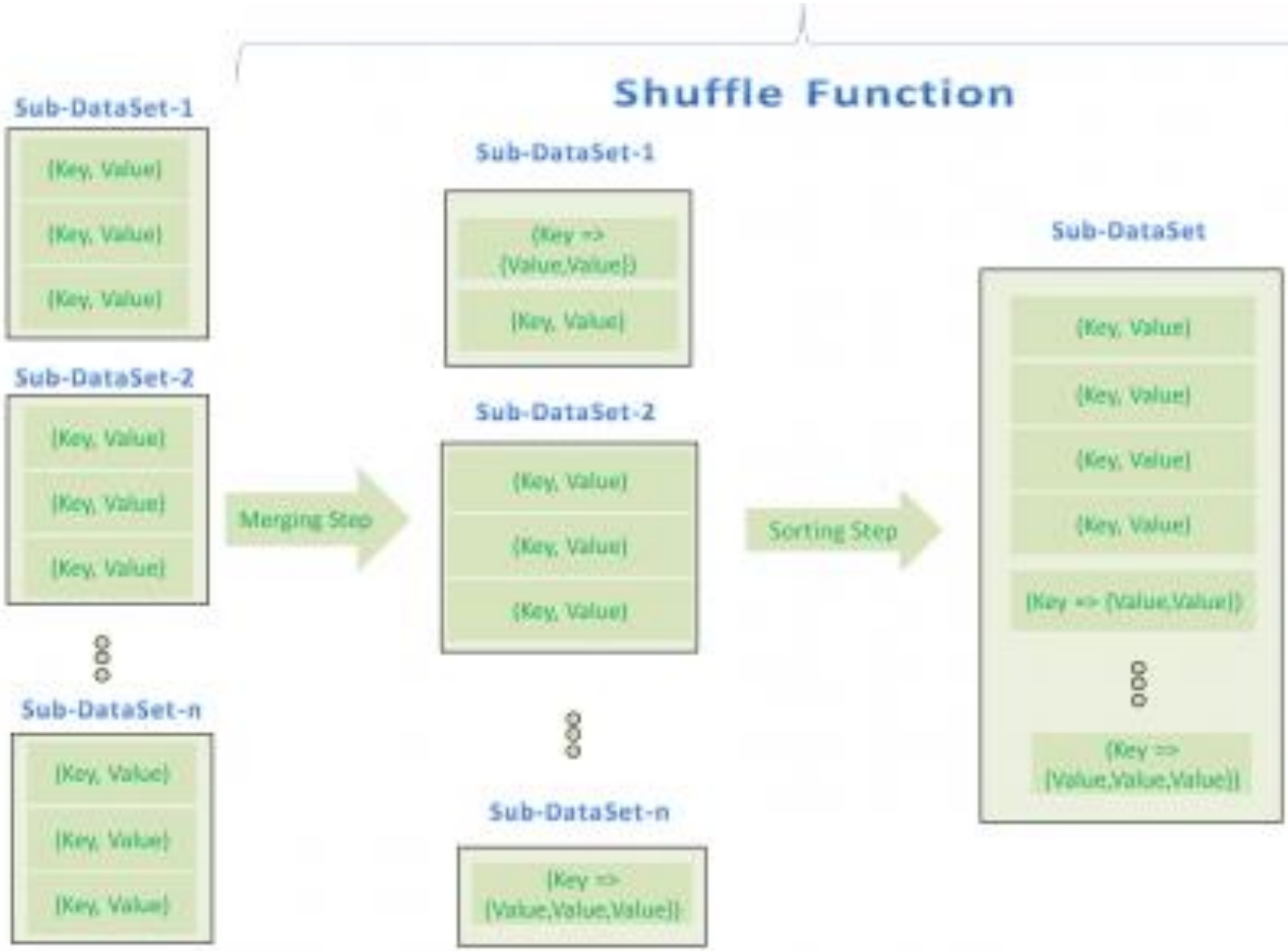
  Mapping
- Splitting step takes input DataSet from Source and divide into smaller Sub-DataSets.
- Mapping step takes those smaller Sub-DataSets and perform required action or computation on each Sub-DataSet.
- The output of this Map Function is a set of key and value pairs as <Key, Value> as shown in the below diagram.


Map Function
(Key, Value)
(Key, Value)
(Key, Value)
Input
Data Set
Data Set-1
Data Set-2
Data Set-n
Split Step
Mapping Step
(Key, Value)
(Key, Value)
(Key, Value)
(Key, Value)
(Key, Value)
(Key, Value)
HDFS

- **Shuffle Function**
  Second step in MapReduce Algorithm. Shuffle Function is also know as “Combine Function”.
- It performs the following two sub-steps:

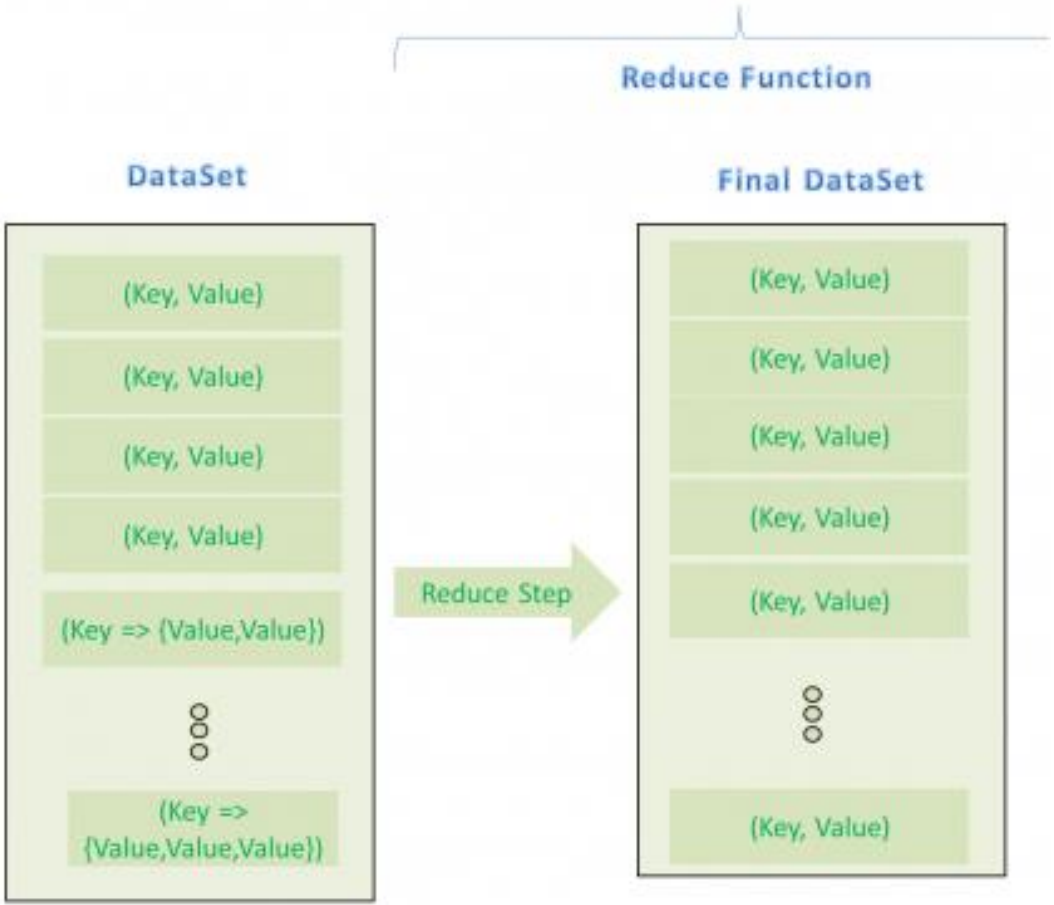
  Merging

  Sorting
- It takes a list of outputs coming from “Map Function” and perform these two sub-steps on each and every key-value pair.
- Merging step combines all key-value pairs which have same keys (that is grouping key-value pairs by comparing “Key”). This step returns <Key, List<Value>>.
- Sorting step takes input from Merging step and sort all key-value pairs by using Keys. This step also returns <Key, List<Value>> output but with sorted key-value pairs.
- Finally, Shuffle Function returns a list of <Key, List<Value>> sorted pairs to next step.


Shuffle Function
Sub-DataSet-1
(Key, Value)
(Key, Value)
(Key, Value)
Sub-DataSet-2
(Key, Value)
(Key, Value)
(Key, Value)
Sub-DataSet-n
(Key, Value)
(Key, Value)
(Key, Value)
Merging Step
Sub-DataSet-1
(Key => (Value,Value))
(Key, Value)
Sub-DataSet-2
(Key, Value)
(Key, Value)
(Key, Value)
Sub-DataSet-n
(Key => (Value,Value,Value))
Sorting Step
Sub-DataSet
(Key, Value)
(Key, Value)
(Key, Value)
(Key, Value)
(Key => (Value,Value))
(Key => (Value,Value,Value))

Reduce Function

- It is the final step in MapReduce Algorithm. It performs only one step : Reduce step.
- It takes list of <Key, List<Value>> sorted pairs from Shuffle Function and perform reduce operation as shown below.

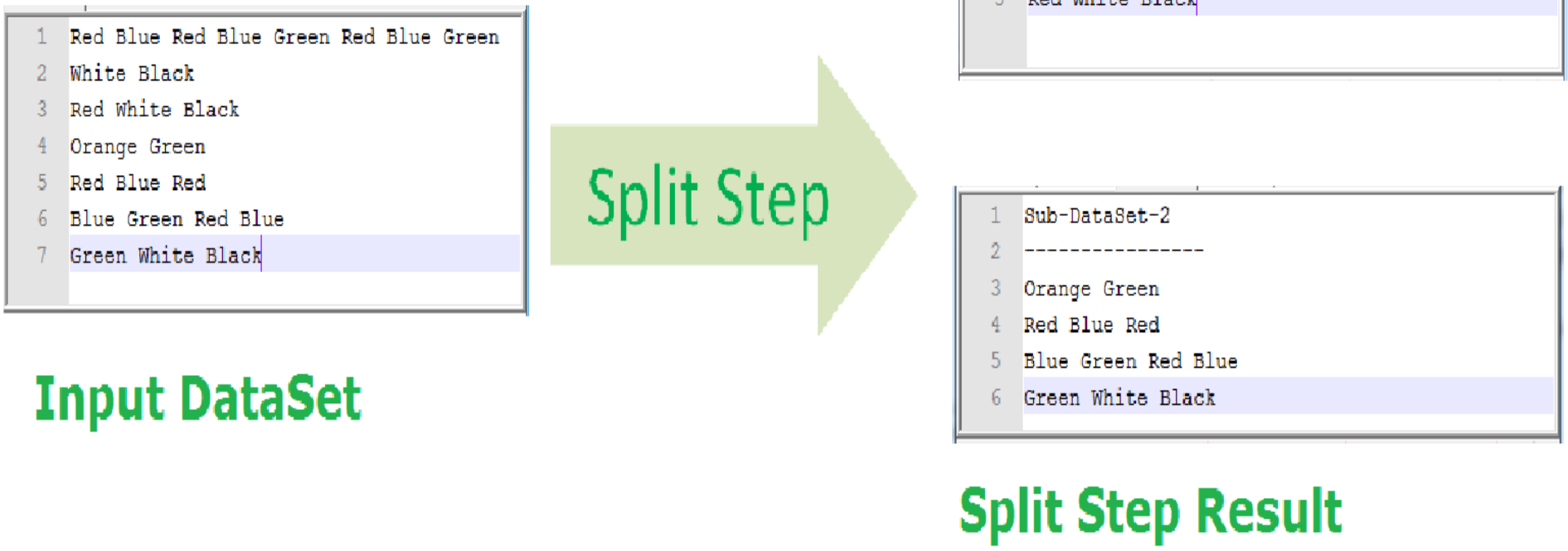


Example of MapReduce

Problem Statement:

Count the number of occurrences of each word available in a DataSet.

- **MapReduce – Map Function (Split Step)**



- **MapReduce – Map Function (Mapping Step)**

```
Sub-DataSet-1
----------------
Red Blue Red Blue Green Red Blue Green
White Black
Red White Black
```

```
Sub-DataSet-2
----------------
Orange Green
Red Blue Red
Blue Green Red Blue
Green White Black
```

Split Step Result

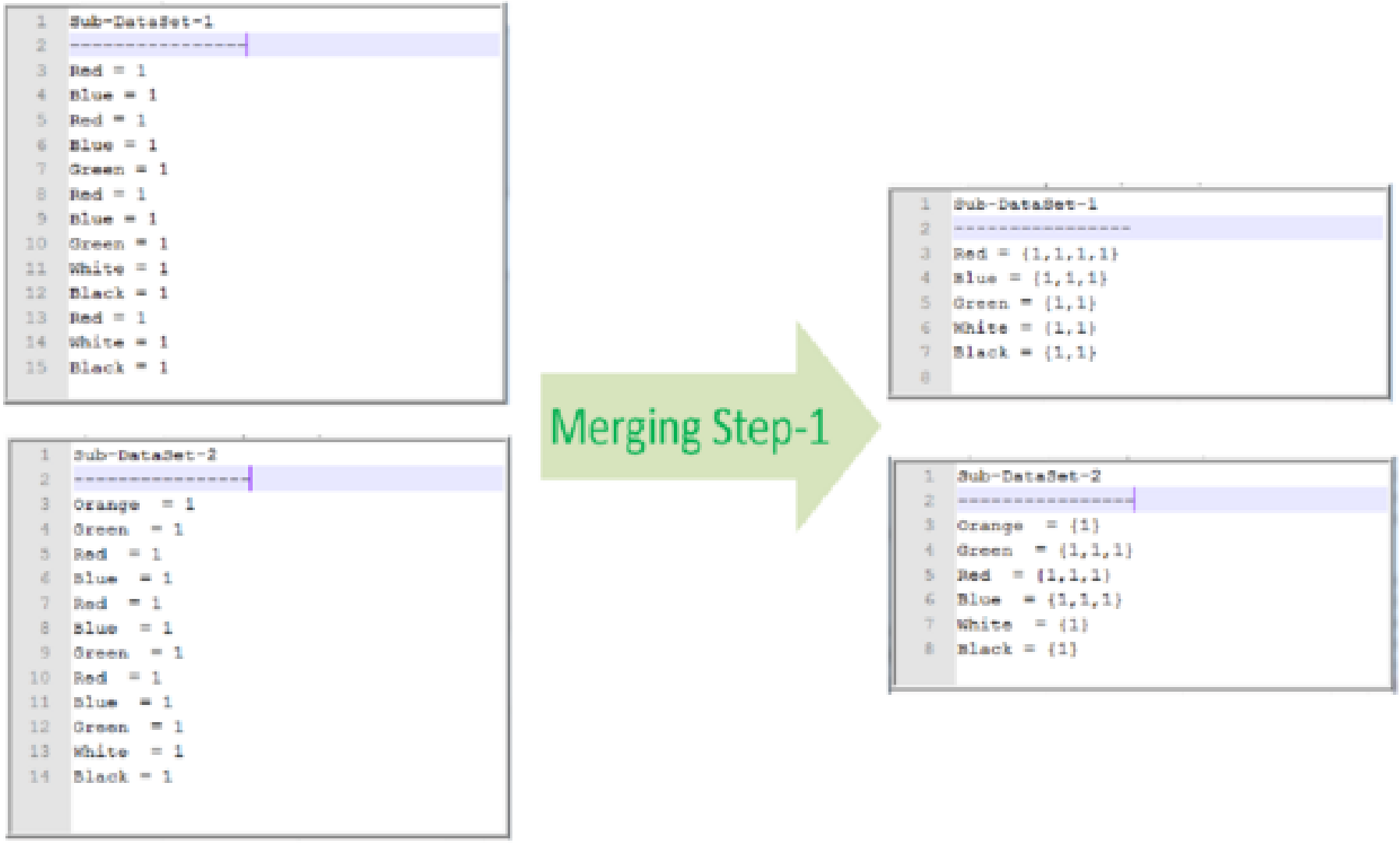
Mapping Step

```
Sub-DataSet-1
----------------
Red = 1
Blue = 1
Red = 1
Blue = 1
Green = 1
Red = 1
Blue = 1
Green = 1
White = 1
Black = 1
Red = 1
White = 1
Black = 1
```

```
Sub-DataSet-2
----------------
Orange  = 1
Green  = 1
Red  = 1
Blue  = 1
Red  = 1
Blue  = 1
Green  = 1
Red  = 1
Blue  = 1
Green  = 1
White  = 1
Black = 1
```

MapReduce - Mapping Step

- **MapReduce – Shuffle Function (Merge Step)**

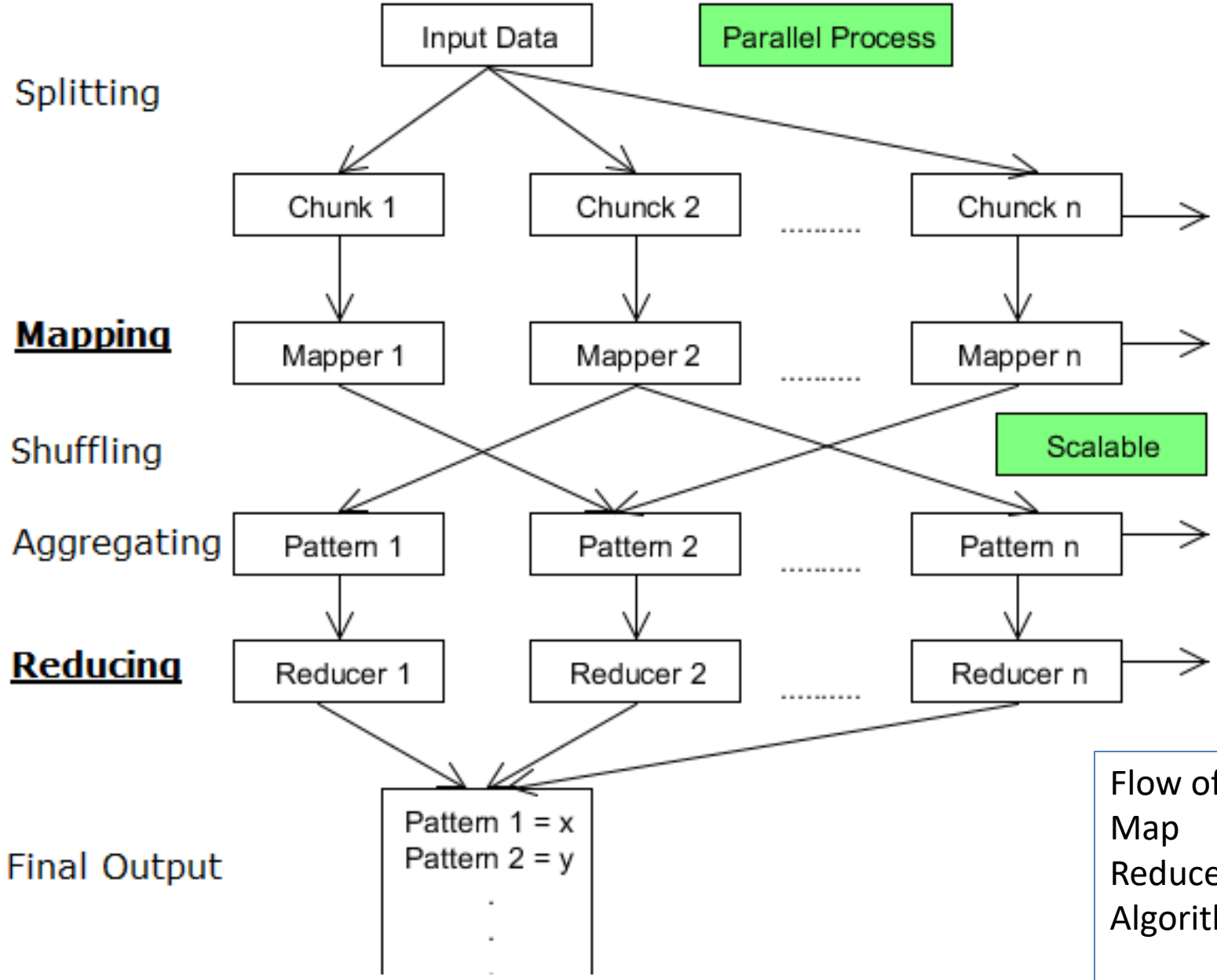


- **MapReduce – Shuffle Function (Sorting Step)**

```
DataSet
----------------
Red = {1,1,1,1,1,1,1}
Blue = {1,1,1,1,1,1}
Green = {1,1,1,1,1}
White = {1,1,1}
Black = {1,1,1}
Orange  = {1}
```


```
DataSet
----------------
Black = {1,1,1}
Blue = {1,1,1,1,1,1}
Green = {1,1,1,1,1}
Orange  = {1}
Red = {1,1,1,1,1,1,1}
White = {1,1,1}
```

- **MapReduce – Reduce Function (Reduce Step)**

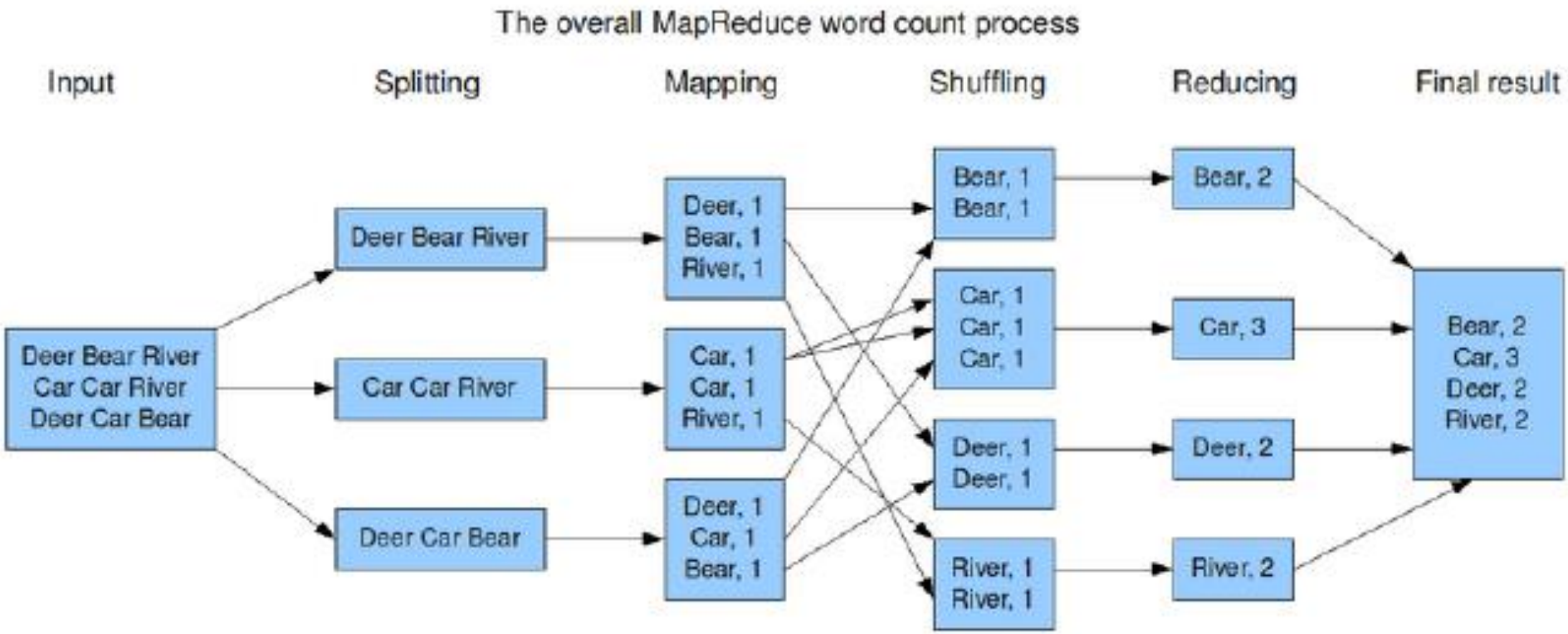


Flow of Map Reduce Algorithm

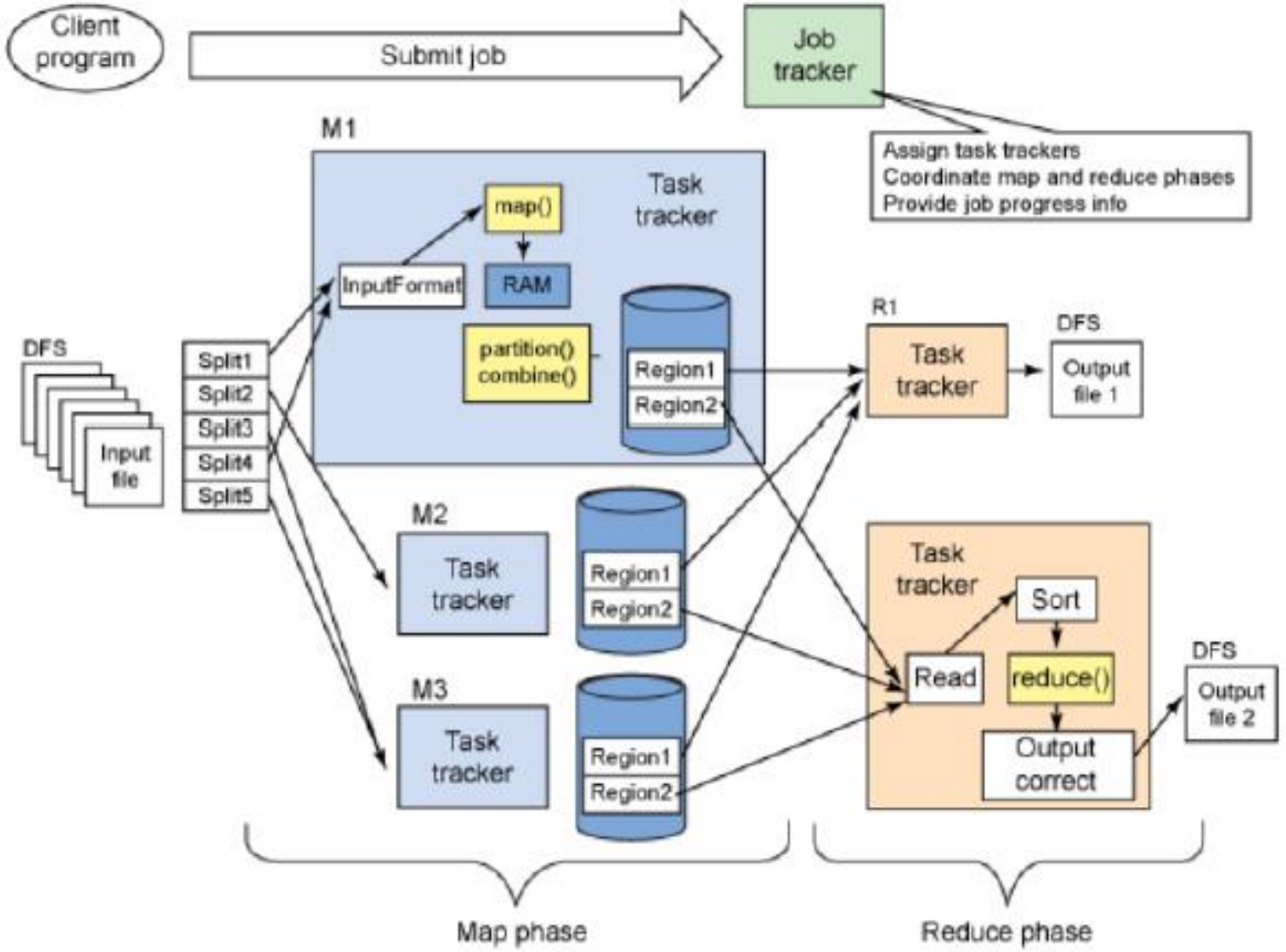
Flow of Map Reduce Algorithm

- In the above map reduce flow:
- The input data can be divided into **n** number of chunks depending upon the amount of data and processing capacity of individual unit.
- Next, it is passed to the mapper functions. Please note that all the chunks are processed simultaneously at the same time, which embraces the parallel processing of data.
- After that, shuffling happens which leads to aggregation of similar patterns.
- Finally, reducers combine them all to get a consolidated output as per the logic.
- This algorithm embraces scalability as depending on the size of the input data, we can keep increasing the number of the parallel processing units.

- **MapReduce example**



MapReduce Data Flow



Algorithms for MapReduce

Algorithms for MapReduce

- MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems.

- These mathematical algorithms may include the following –

  Sorting

  Searching

  Indexing

  TF-IDF

Algorithms for MapReduce

Sorting

- Sorting is one of the basic MapReduce algorithms to process and analyze data. MapReduce implements sorting algorithm to automatically sort the output key-value pairs from the mapper by their keys.

- The set of intermediate key-value pairs for a given Reducer is automatically sorted by Hadoop to form key-values (K2, {V2, V2, …}) before they are presented to the Reducer.

Algorithms for MapReduce

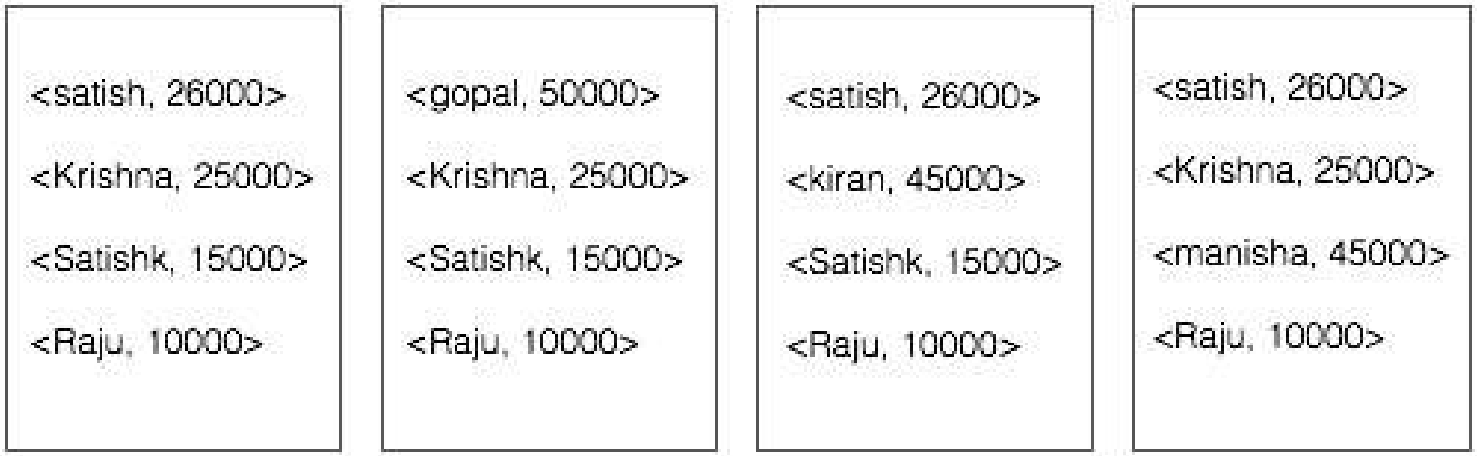
Searching

- Searching plays an important role in MapReduce algorithm.

  Example:

  The following example shows how MapReduce employs Searching algorithm to find out the details of the employee who draws the highest salary in a given employee dataset.

- **The Map phase** processes each input file and provides the employee data in key-value pairs (<k, v> : <emp name, salary>). See the following illustration.



Algorithms for MapReduce

- **The combiner phase** (searching technique) will accept the input from the Map phase as a key-value pair with employee name and salary. Using searching technique, the combiner will check all the employee salary to find the highest salaried employee in each file.

- **Reducer phase** − Form each file, you will find the highest salaried employee. To avoid redundancy, check all the <k, v> pairs and eliminate duplicate entries, if any. The same algorithm is used in between the four <k, v> pairs, which are coming from four input files. The final output should be as follows −

<gopal, 50000>

Algorithms for MapReduce

Indexing

- Normally indexing is used to point to a particular data and its address. It performs batch indexing on the input files for a particular Mapper.

- The indexing technique that is normally used in MapReduce is known as **inverted index.** Search engines like Google and Bing use inverted indexing technique.

- **Example**
- The following text is the input for inverted indexing. Here T[0], T[1], and T[2] are the file names and their content are in double quotes.

```
T[0] = "it is what it is"
T[1] = "what is it"
T[2] = "it is a banana"
```

- After applying the Indexing algorithm, we get the following output .

```
"a": {2}
"banana": {2}
"is": {0, 1, 2}
"it": {0, 1, 2}
"what": {0, 1}
```

- Here "a": {2} implies the term "a" appears in the T[2] file. Similarly, "is": {0, 1, 2} implies the term "is" appears in the files T[0], T[1], and T[2].

Algorithms for MapReduce (TF-IDF)

- TF-IDF for a word in a document is calculated by multiplying two different metrics:

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- While computing TF, all the terms are considered equally important.

Algorithms for MapReduce (TF-IDF)

- **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

  IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

Algorithms for MapReduce (TF-IDF)

- The **inverse document frequency** of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is.

- So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Algorithms for MapReduce (TF-IDF)

- Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

- Since the ratio inside the IDF's log function is always greater than or equal to 1, the value of IDF (and TF–IDF) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the IDF and TF–IDF closer to 0.

Example

- Consider a document containing 1000 words, wherein the word **hive** appears 50 times. The TF for **hive** is then (50 / 1000) = 0.05.

- Now, assume we have 10 million documents and the word **hive** appears in 1000 of these. Then, the IDF is calculated as log(10,000,000 / 1,000) = 4.

- The TF-IDF weight is the product of these quantities – 0.05 × 4 = 0.20.

Thank you