hadoop

Sqoop

HIVE

apache
PIG

APACHE
Spark™

FLUME

HUE

Hello!

I am Aravinda

You can find me at @

aravinda.cv@nitte.edu.in

“Life is not a COMPETITION

Its about HELPING & INSPIRING

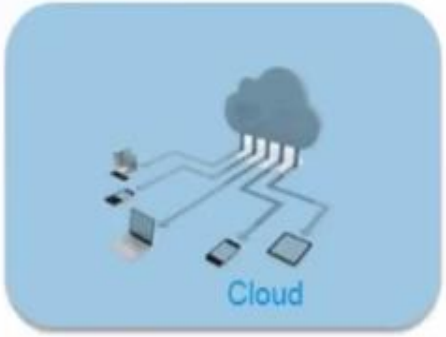
So we can reach our POTENTIAL”.

AGENDA

- Evolution of Data
- What is Big Data
- Big Data as an Opportunity
- Problems in Enhancing Opportunity
- Hadoop as Solution
- Hadoop Ecosystem

Evolution of Technology

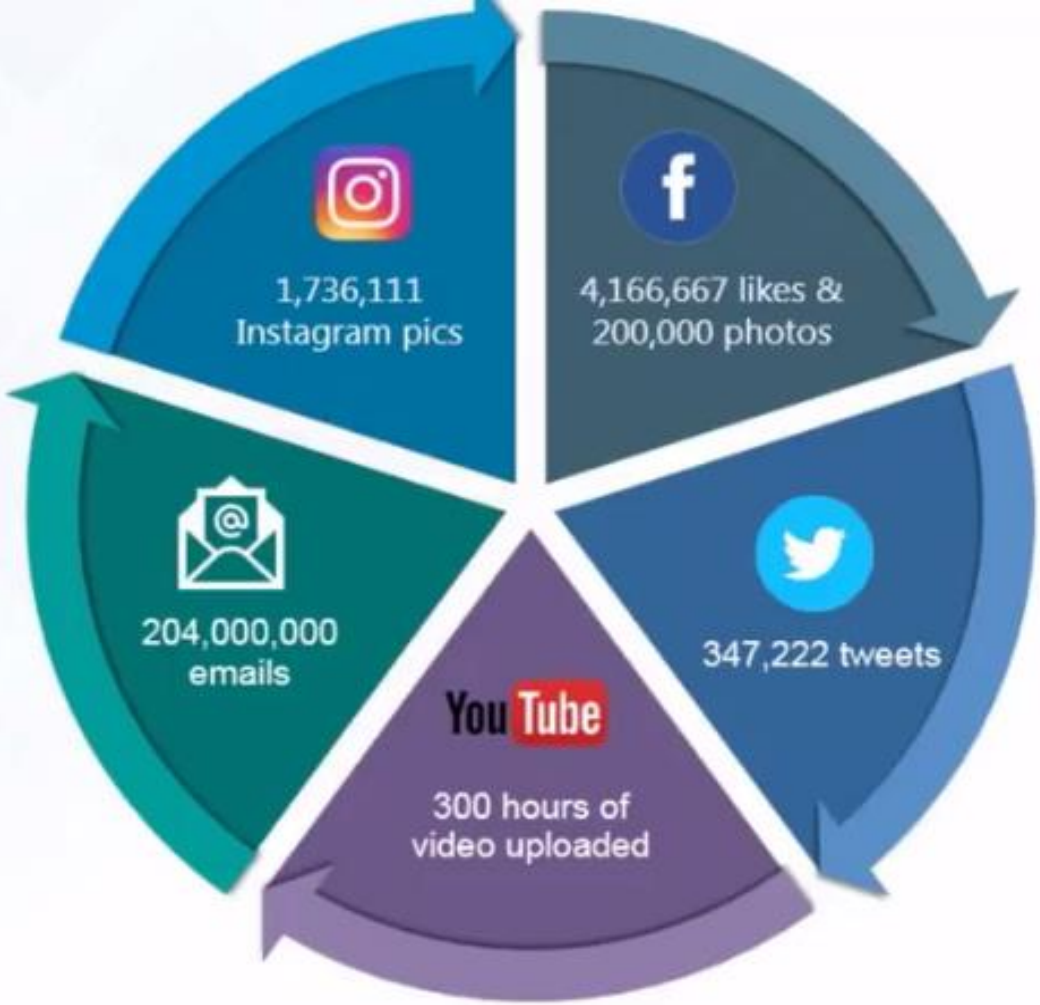
1. Evolution of Technology
2. IOT
3. Social Media
4. Data evolved to Big Data



1. Evolution of Technology
2. **IOT**
3. Social Media
4. Data evolved to Big Data

IOT: 50 Billion devices by 2020

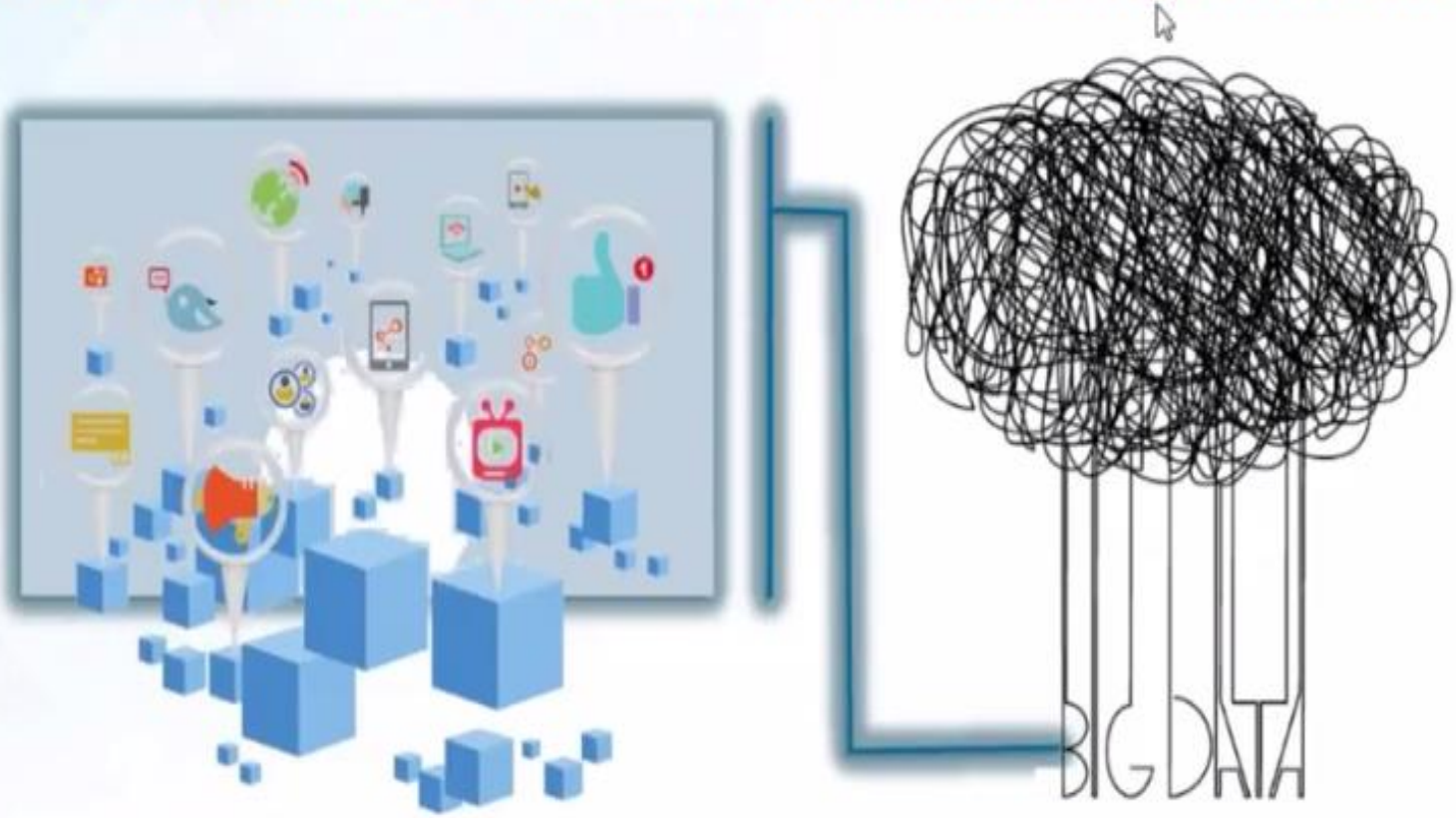
1. Evolution of Technology
2. IOT
3. **Social Media**
4. Data evolved to Big Data



1. Evolution of Technology
2. IOT
3. Social Media
4. Other Factors

What Is Big Data?

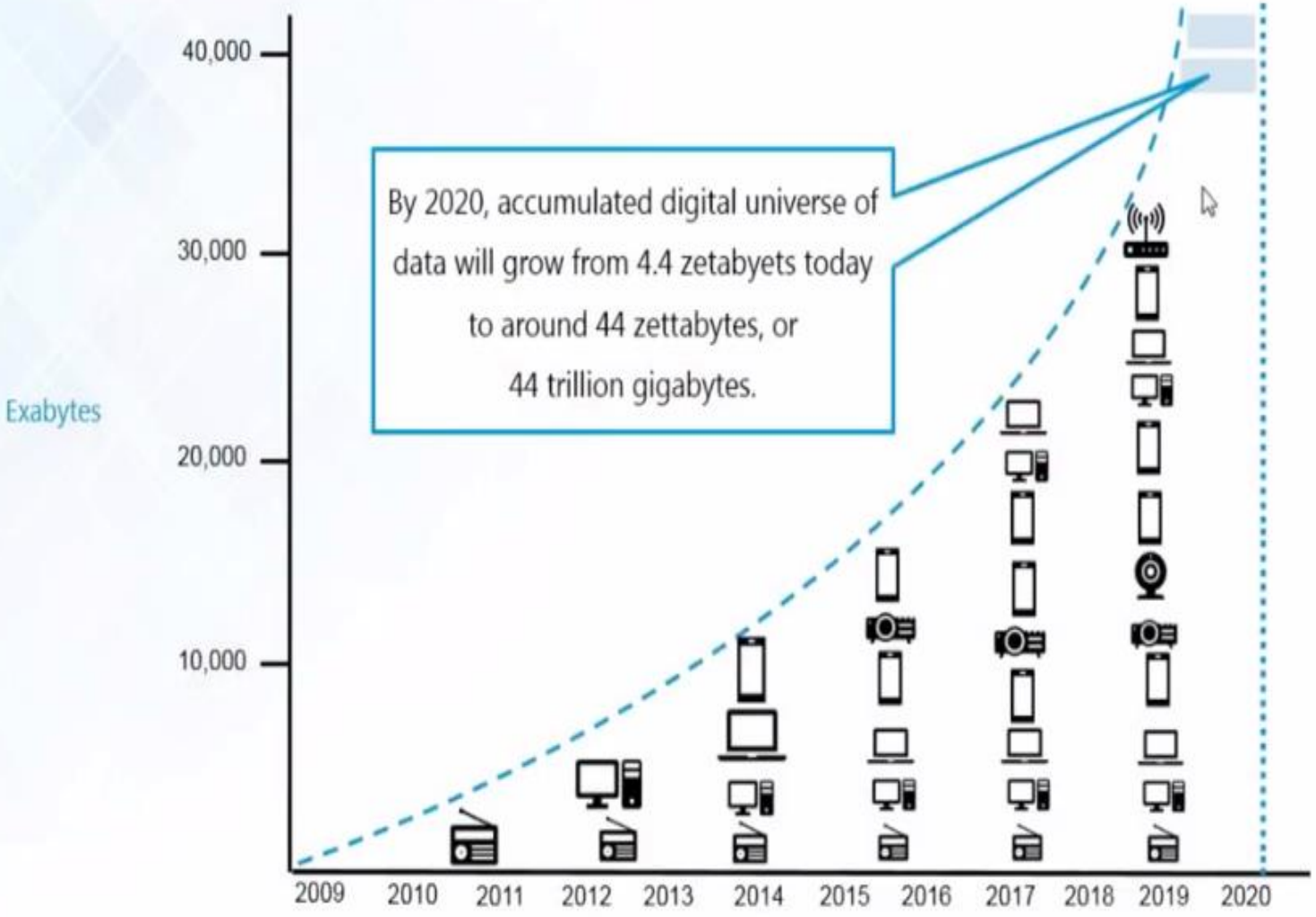
Big data is the term for collection of data sets so large and complex that it becomes difficult to process using on-hand database system tools or traditional data processing applications



5 V's of Big Data


1 Volume
2 Variety
3 Velocity
4 Value
5 Veracity
5 V's of Big Data

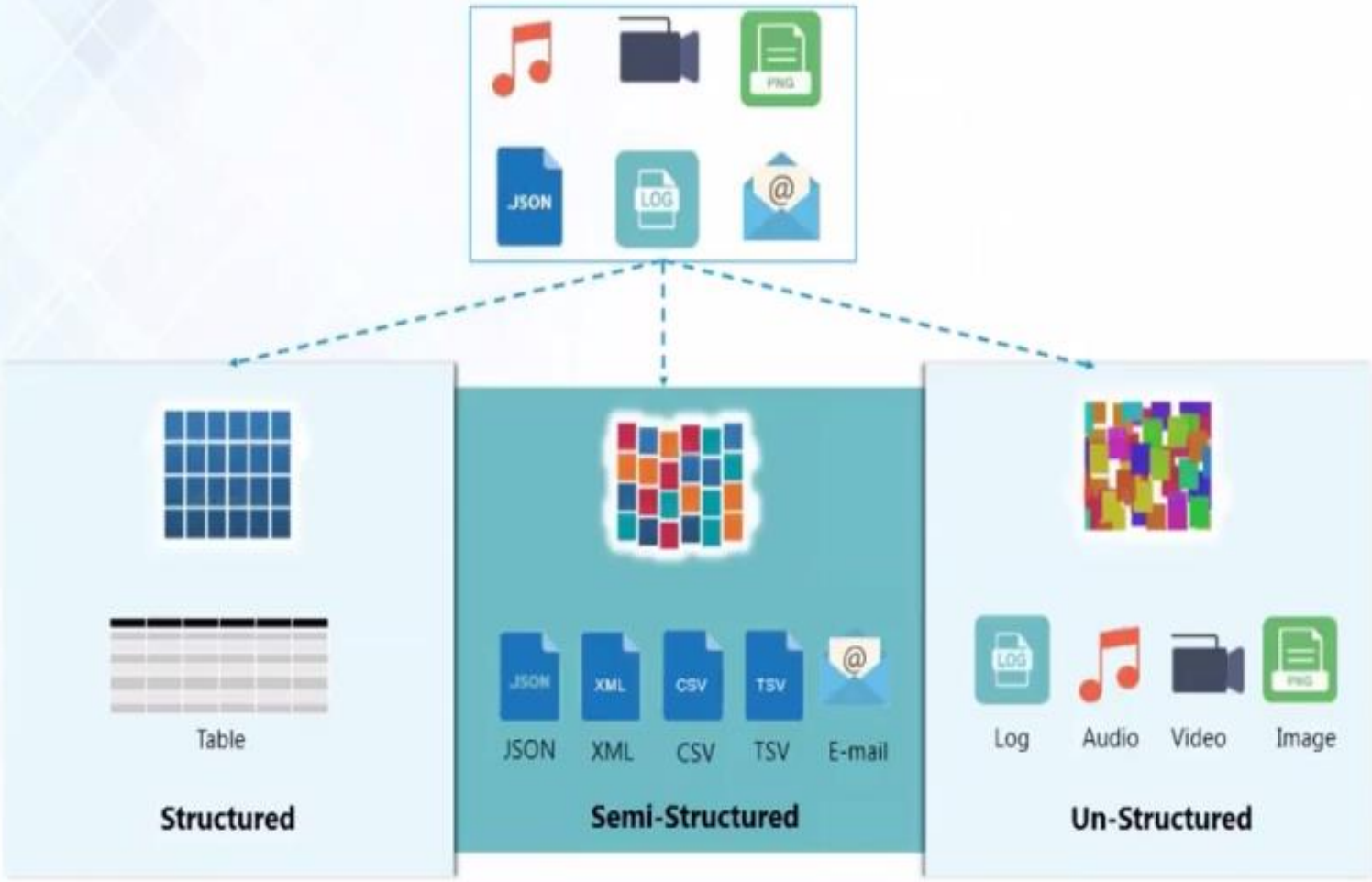
1 **Volume**

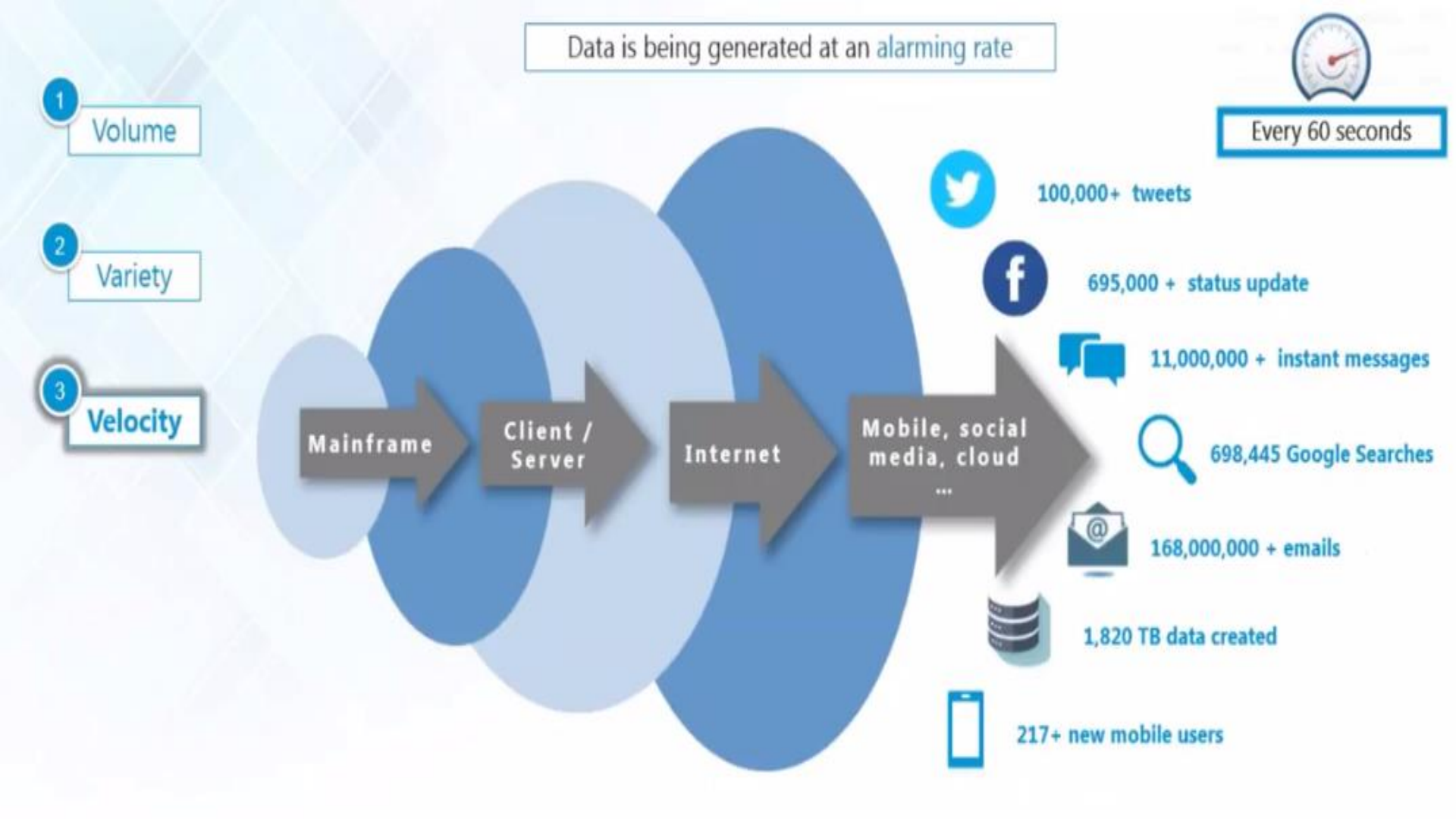

40,000
30,000
20,000
10,000
Exabytes
By 2020, accumulated digital universe of data will grow from 4.4 zetabyets today to around 44 zettabytes, or 44 trillion gigabytes.
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

1. Volume

2. Variety

Different kinds of data is being generated from various sources




Data is being generated at an alarming rate
1
Volume
2
Variety
3
Velocity
Mainframe
Client / Server
Internet
Mobile, social media, cloud
...
Every 60 seconds
100,000+ tweets
695,000 + status update
11,000,000 + instant messages
698,445 Google Searches
168,000,000 + emails
1,820 TB data created
217+ new mobile users

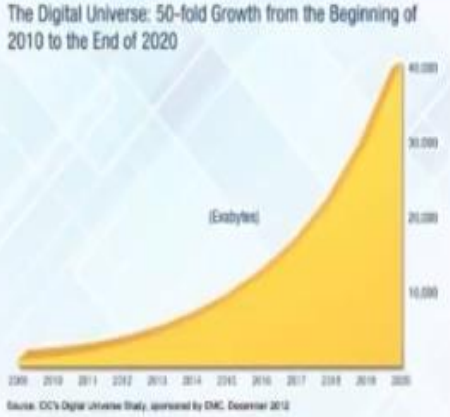
Mechanism to bring the correct meaning out of the data

1. Volume
2. Variety
3. Velocity
4. **Value**

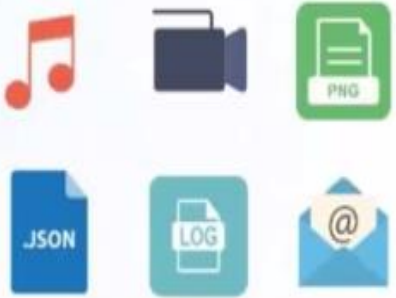
1. Volume
2. Variety
3. Velocity
4. Value
5. **Veracity**

Min	Max	Mean	SD
4.3	?	5.84	0.83
2.0	4.4	3.05	50000000
15000	7.9	1.20	0.43
0.1	2.5	?	0.76

Uncertainty and inconsistencies in the data



Volume



Different kinds of data is being generated from various sources

Variety

Data is being generated at an alarming rate

Velocity

Mechanism to bring the correct meaning out of the data

Value

Min	Max	Mean	SD
4.3	?	5.84	0.83
2.0	4.4	3.05	50000000
15000	7.9	1.20	0.43
0.1	2.5	?	0.76

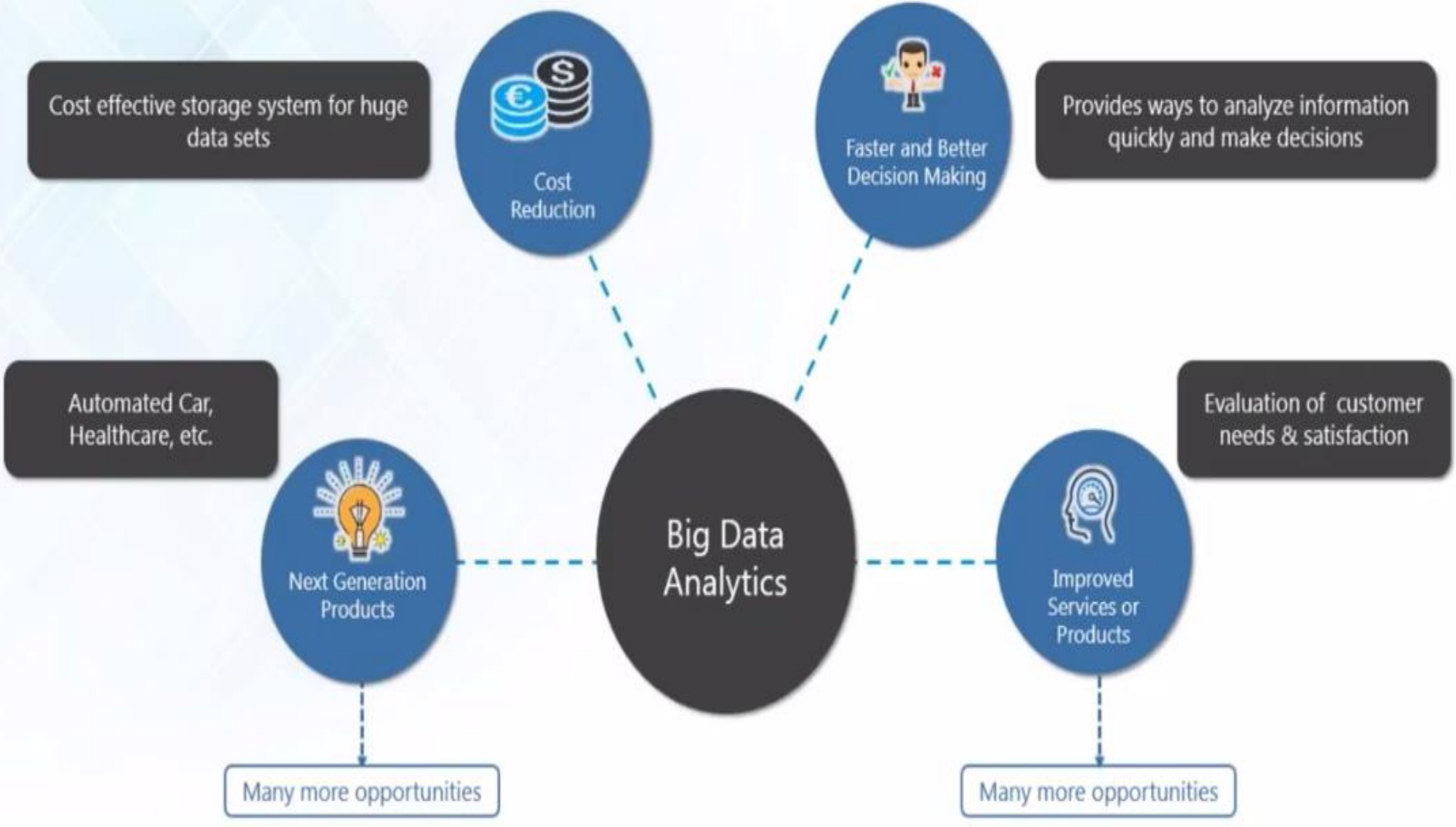
Uncertainty and inconsistencies in the data

Veracity

. . . . . .

V's associated with Big Data may grow with time

Big Data as an Opportunity


Cost effective storage system for huge data sets
Cost Reduction
Faster and Better Decision Making
Provides ways to analyze information quickly and make decisions
Automated Car, Healthcare, etc.
Next Generation Products
Big Data Analytics
Improved Services or Products
Evaluation of customer needs & satisfaction
Many more opportunities
Many more opportunities

Tools for Big-Data

Storage : HDFS, YARN

Ingestion : SQOOP,

Analysis : HIVE, PIG

Processing : introduction to MR, SPARK (CORE, SQL)

System requirements 4 Big-Data

User Laptop/Computer Requirements:

For Administrator-

12 GB RAM, 4 CPU, 50GB HDD

For developer-

8 GB RAM, 4 CPU, 50GB HDD

What is Data

Data is collection of raw facts and figures. means some values or text available in a file or in secured server that can be useful to get some information .for example : banking data, student data, reservation systems

we can read this type of data to get something valuable in terms of information.

What is Information

Information is processed data. means if I have a student table with 100 no of records and I want to know what is name of roll no 21 then values available in table (100 records) is data and after processing name of the student having roll no 21 is Information.

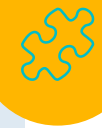
What is Database

Database is collection of data. this collection can be done using a file system or DBMS (DataBase Management System)

DATABASE: data stored in tables. tables will have rows (records) & columns.

What is Filesystem

Data is stored in the form of files. like files on hard disk. for example this text file. but file system is not secure, it does not support transaction, we can not read data easily .They may be NTFS, FAT32 on windows, EXt3, EXT5 o linux: It contain files like .txt, .doc, .xls, .mp3, .mp4, .avi. .dll



What is DBMS

is a set of computer program to manage data like oracle, db2, MySQL

What is RDBMS

which follows codd rules.i.e 6 or more rules considered as RDBMS.

RDBMS stands for Relational Database Management System.

(in 1970 EF Codd, Employee of IBM presentaed a paper called "Database for large shared data banks" in this paper he has give 13 rules rule 0 to 12. if any DBMS follows 6 or more codd rules that can be considered as RDBMS.)

why because RDBMS is schema on write means RDBMS must have a schema before you insert anything into it. that i the reason it can only store structured data.

what is DB server

A database server is a server providing database services. Such a server runs the database software

.MYSQL, ORACLE, IBM DB2 these are DB servers.

How data is stored on your machine (RDBMS)

in your database server which provides RDBMS service. where we can create tables, insert data into that, update and delete data. RDBMS is limited to store PB of data.

Types of Data

we can have Structured, Semi Structured or unstructured Data.

Structured Data: data in RDBMS or text files with specific no of values

1, Inky,C,9742200000

2, Pinky,P,6026500000

Semi Structured Data: JSON (Java script object notation) and XML (eXtensible Markup Language) files

```
<student>
<rollno>1</rollno>
<fname>Inky</fname>
<lname>C</lname>
<class>Computer Engineering</class>
</student>
```

unstructured Data: server logs, text files

Introduction to Big-Data

- Big data is a term that describes the large volume of data generated that can be structured, unstructured or semi structured
- Big Data is similar to 'small data' but larger in size.

Definition of Big data differs for organizations.

when we can not process data using single machine that is called big data

- Below are measurement frequently used for big data-

- 1 TB(1000GB)
- 1 PB(1000TB)
- 1 EB(1000PB)
- 1ZB(1000EB)
- 1YB(1000ZB)

- examples of Big Data--
- Server Log files, social media updates, websites like facebook, twitter, instagram generate huge amount of data which is one of the example of Big Data

Sources of Big-Data

Google - Google processes 100 billion searches a month. That's an average of 40,000 search queries every second

Facebook- Every 60 seconds on Facebook: 510 comments are posted, 293,000 statuses are updated, and 136,000 photos are uploaded. Facebook stores 300 PB of data, with an incoming rate of ~600 TB/day, this is 3 times then last year

Twitter- 500 million people visit Twitter each month without logging in. There are 500 million Tweets sent each day. That's 6,000 Tweets every second

YouTube- 300 hours of video are uploaded to YouTube every minute. There are 3.25 billion hours of video watched each month

IBM Big Data Analytics

A Case Study

Big Data Collected by Smart Meter

Data was collected in 1 Month

Earlier

Data is collected in 15 Minutes

Now

Managing the large volume and velocity of information generated by short-interval reads of smart meter data can overwhelm existing IT resources

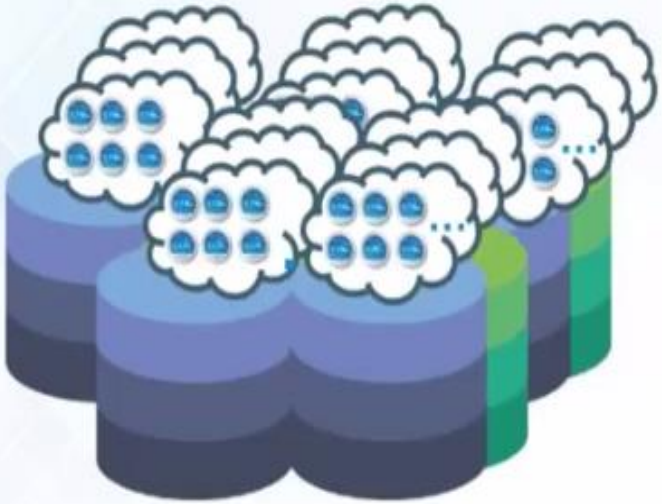
96 million reads per day for every million meters

Big Data generated by Smart Meter

IBM

Problem with Smart Meter Big Data

To manage and use this information to gain insight, utility companies must be capable of high-volume data management and advanced analytics designed to transform data into actionable insights.



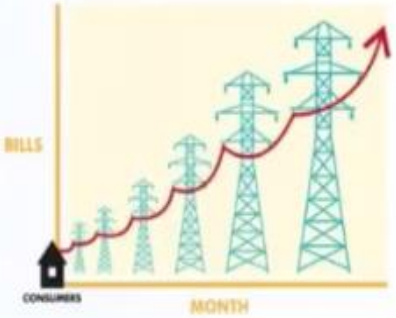
Store

Analyze

IBM

How Smart Meter Big Data Is Analysed

Before analyzing Big Data



Energy utilization and billing has increased

After analyzing Big Data

During peak-load the users require more energy

During off-peak times the users required less energy

Time-of-use pricing encourages cost-savvy retail like industrial heavy machines to be used at off-peak times

IBM Smart Meter Solution

IBM offers an integrated suite of products designed to enable IT to leverage big data in a variety of ways that can contribute to the success of energy companies

IBM Solution

1. Managing smart meter data
2. Monitoring the distribution grid
3. Optimizing unit commitment
4. Optimizing energy trading
5. Forecasting and scheduling loads

ONCOR using IBM Smart Meter Solution

ONCOR

Oncor Electric Delivery has incorporate IBM Smart Meter service

1. **Instrumented** — Utilizes smart electricity meters to accurately measure the electricity usage of a household
2. **Interconnected** — Unprecedented access to detailed information about their electricity use
3. **Intelligent** — Consumers monitor and control their electricity usage through near-real time readings of electricity meters

Customers in Oncor's service territory showed last year during the company's biggest energy saver contest that by using the information from Oncor's advanced meter

Users reduced their electric usage and bills by 25 percent or more

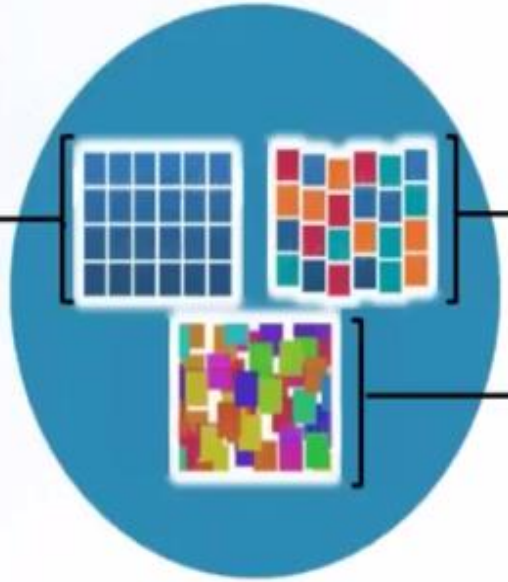
Problems with Big Data

Problem 1: Storing exponentially growing huge datasets

- Data generated in past **2 years** is more than the previous history in total
- By 2020, total digital data will grow to **44 Zettabytes** approximately
- By 2020, about **1.7 MB** of new info will be created every second for every person

Problems with Big Data

Problem 2: Processing data having complex structure



Structured

- Organized data format
- Data schema is fixed
- Ex: RDBMS data, etc.

Semi – Structured

- Partial organized data
- Lacks formal structure of a data model
- Ex: XML & JSON files, etc.

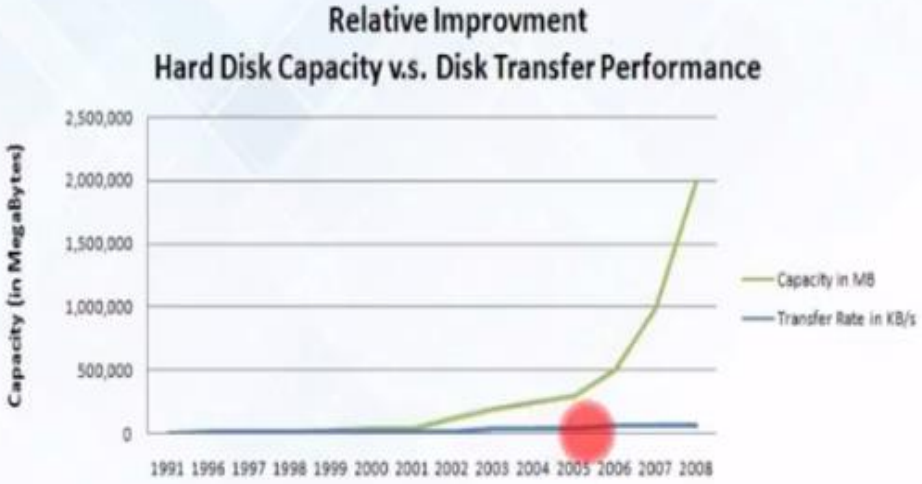
Unstructured

- Un-organized data
- Unknown schema
- Ex: multi-media files, etc.

Problems with Big Data

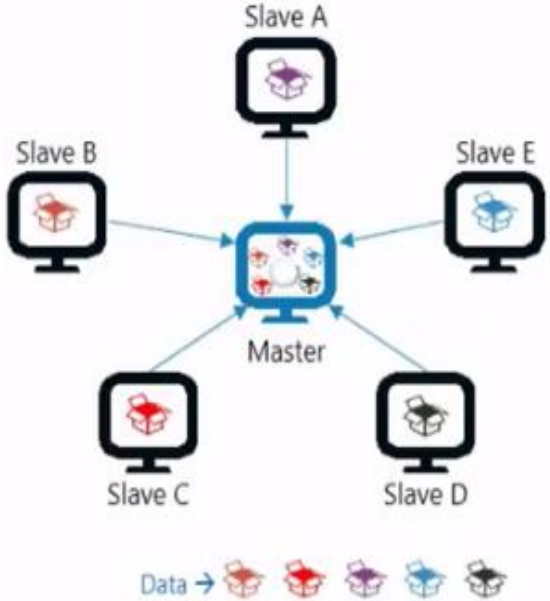
Problem 3: Processing data faster

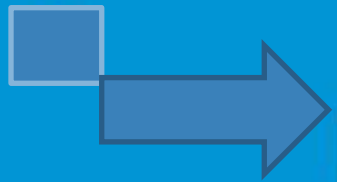
The data is growing at much faster rate than that of disk read/write speed



Source: *Tom's Hardware*

Bringing huge amount of data to computation unit becomes a bottleneck





Hadoop-as-a-Solution

Hadoop

Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion

HDFS (Storage)

Allows to dump any kind of data across the cluster

MapReduce (Processing)

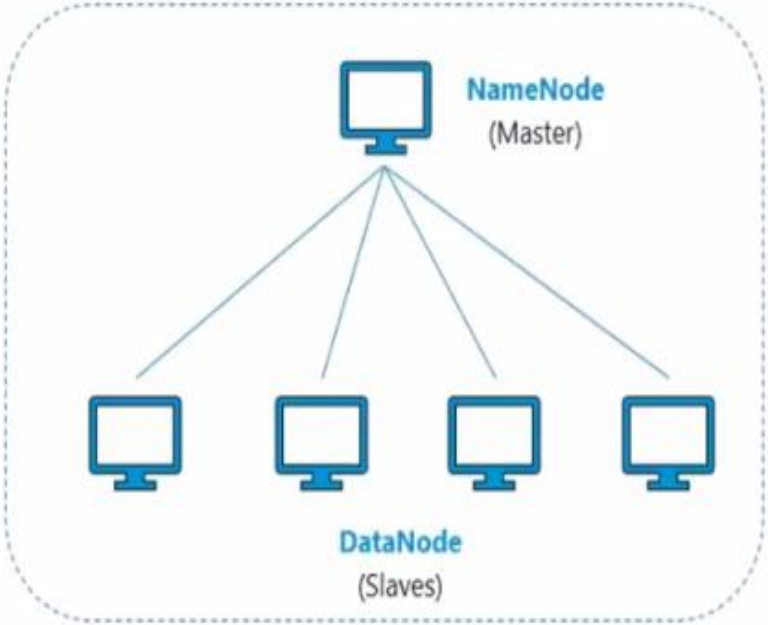
Allows parallel processing of the data stored in HDFS

Hadoop Distributed File System

HDFS creates a level of abstraction over the resources, from where we can see the whole HDFS as a single unit.

HDFS has two core components, i.e. NameNode and DataNode.

- The *NameNode* is the main node that contains metadata about the data stored.
- Data is stored on the *DataNodes* which are commodity hardware in the distributed environment.



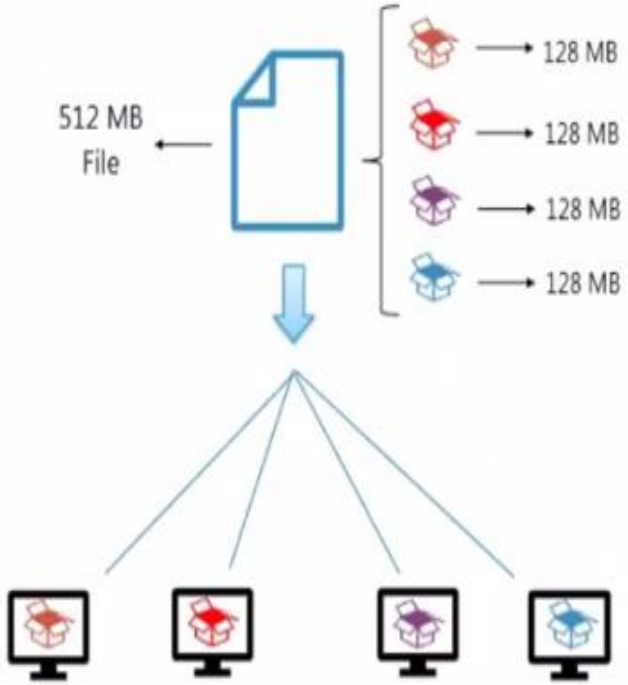
Hadoop Cluster

Storing Data (Solution)

Problem 1: Storing exponentially growing huge datasets

Solution: HDFS

- Storage unit of Hadoop
- It is a Distributed File System
- Divide files (input data) into smaller chunks and stores it across the cluster
- Scalable as per requirement

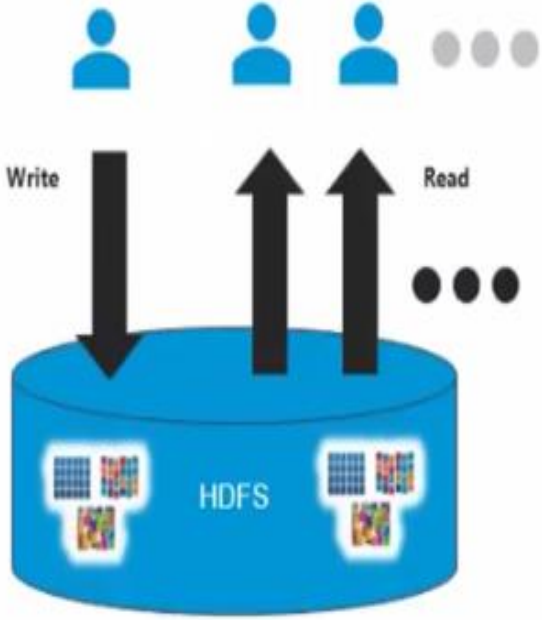


Store Different Kinds Of Data (Solution)

Problem 2: Storing unstructured data

Solution: HDFS

- Allows to store any kind of data, be it structured, semi-structured or unstructured
- Follows WORM (Write Once Read Many)
- No schema validation is done while dumping data

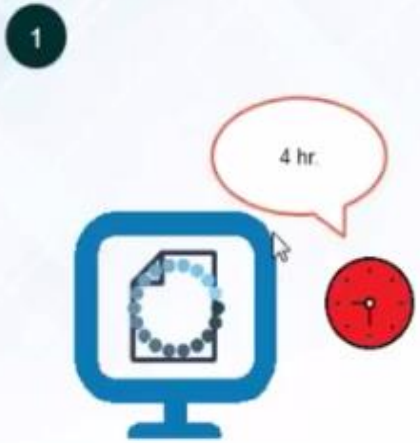


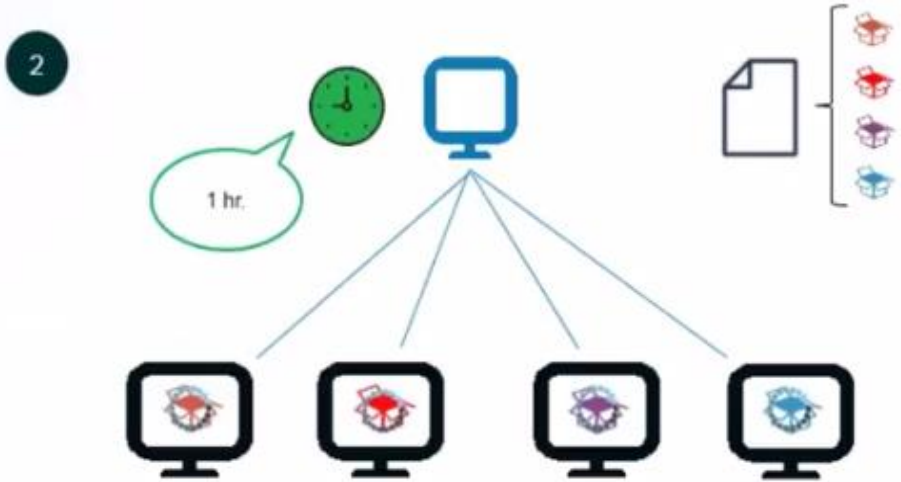
Store Different Kinds Of Data (Solution)

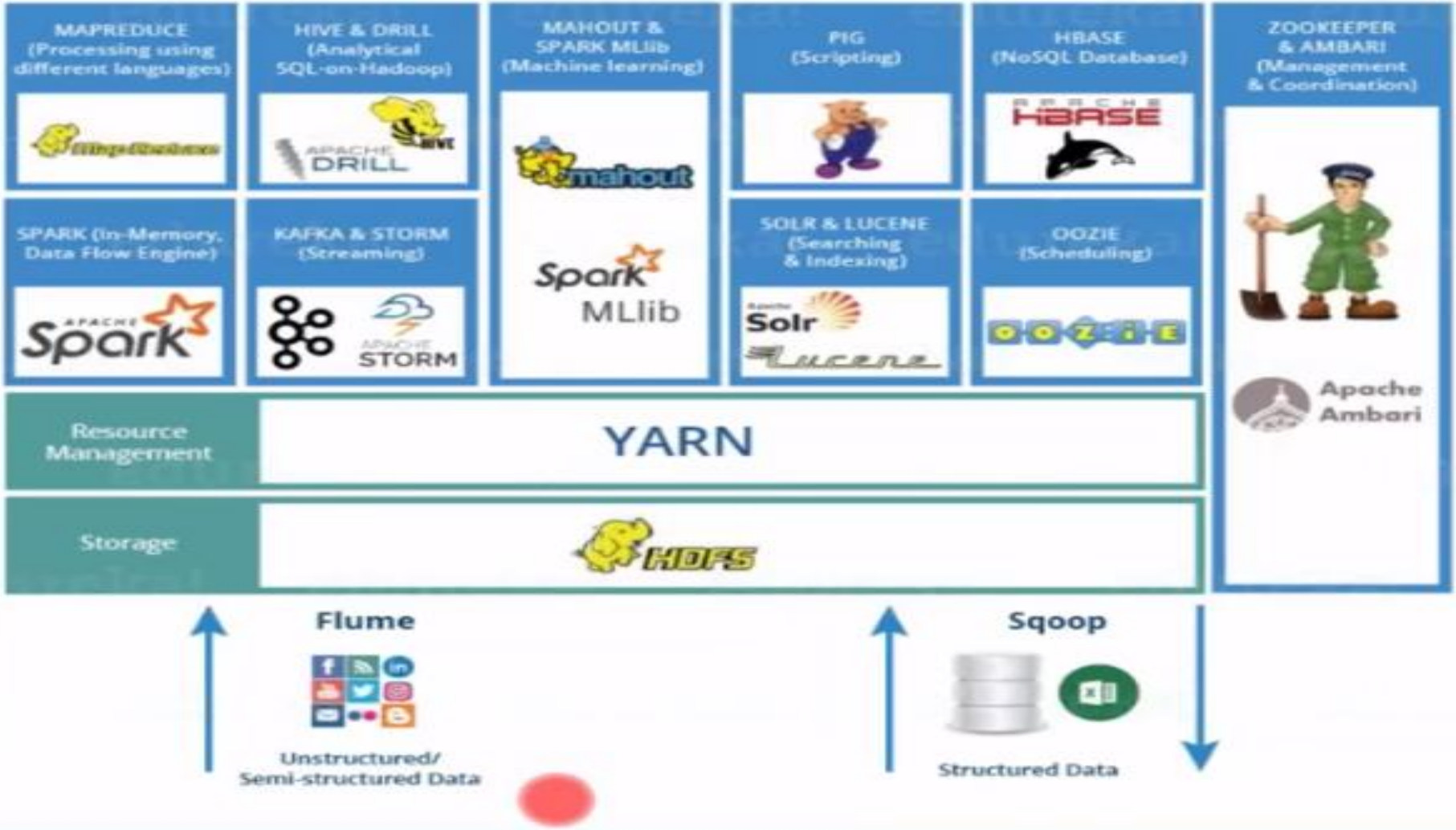
Problem 3: Processing data faster

Solution: Hadoop MapReduce

- Provides parallel processing of data present in HDFS
- Allows to process data locally i.e. each node works with a part of data which is stored on it






MAPREDUCE
(Processing using different languages)
HIVE & DRILL
(Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib
(Machine learning)
PIG
(Scripting)
HBASE
(NoSQL Database)
ZOOKEEPER & AMBARI
(Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM
(Streaming)
SOLR & LUCENE
(Searching & Indexing)
OOZIE
(Scheduling)
Resource Management
YARN
Storage
Flume
Unstructured/
Semi-structured Data
Sqoop
Structured Data

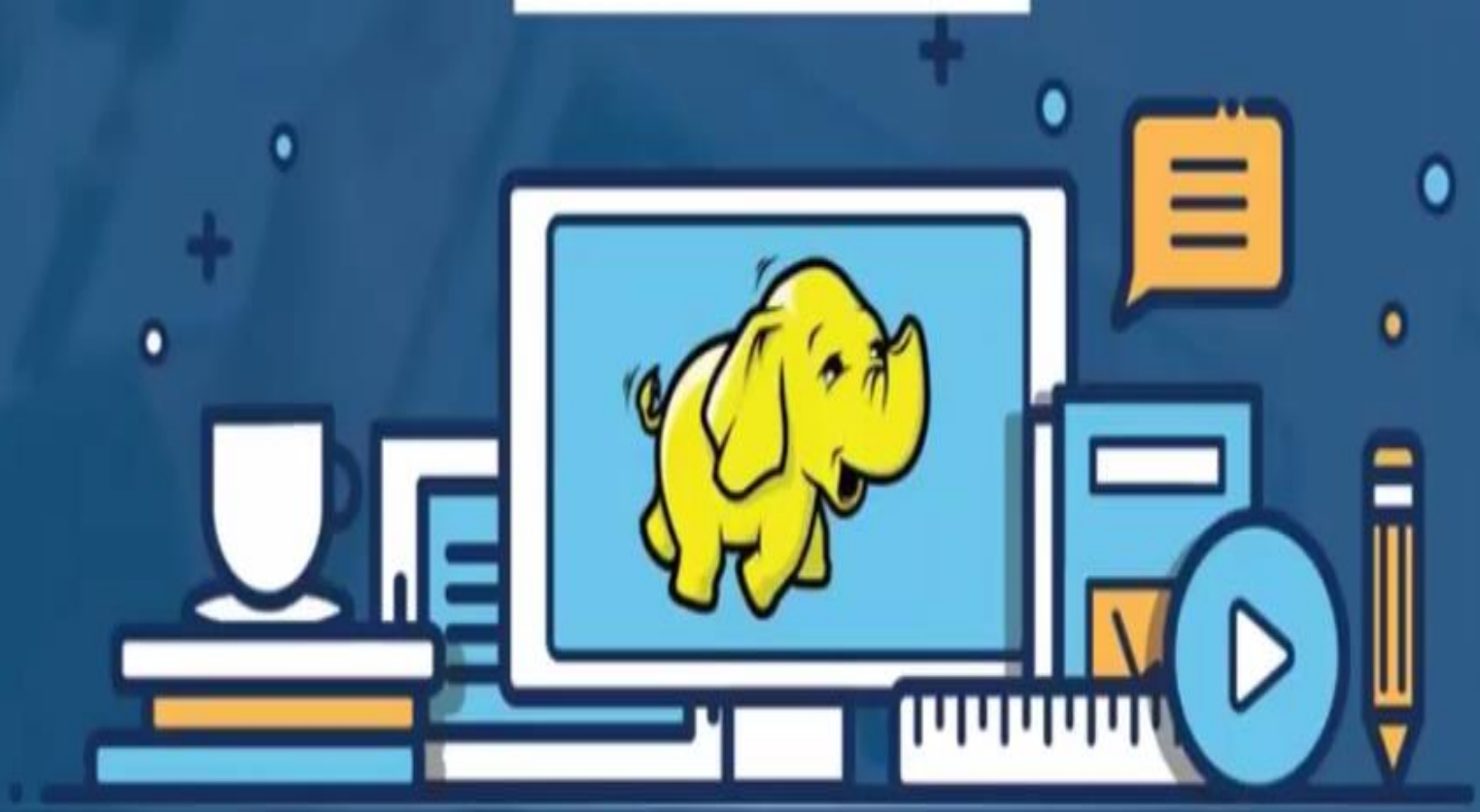
Hadoop provides a scalable solution to store and process huge data sets in parallel and distributed fashion.

Apache Hive is a data warehousing tool that allows us to perform big data analytics using Hive Query Language which is very similar to SQL.

Apache Pig is a platform, used to analyze large data sets representing them as data flows.

Apache Spark is an in-memory data processing engine that allows us to efficiently execute streaming, machine learning or SQL workloads and requires fast iterative access to datasets.

Apache HBase is a NoSQL database that allows us to store unstructured and semi – structured data with ease and provides real time read/write access.



Hadoop Tutorial

- Big Data Growth Drivers
- What is Big Data?
- Hadoop Introduction
- Hadoop Master/Slave Architecture
- Hadoop Core Components
- HDFS Data Blocks
- HDFS Read/Write Mechanism
- What is MapReduce
- MapReduce Program
- MapReduce Job Workflow
- Hadoop Ecosystem
- Hadoop Use Case: Analyzing Olympic Dataset

Big Data Growth Drivers

Data Generated
Every 60 Seconds

Users like
4,166,667
posts

Users send
347,222
tweets

Users cast
18,327
votes

Users like
1,736,111
posts

Users upload
300 ho rs

Global Mobile Data Traffic ,2015-2020

Cisco Forecasts 30.6 Exabytes per Month of Mobile Data Traffic by 2020

3 major trends contributing to the growth of mobile data traffic:

- Adapting to Smarter Mobile Devices
- Defining Cell Network Advances—2G, 3G, and 4G (5G Perspectives)
- Reviewing Tiered Pricing—Unlimited Data and Shared Plans

"Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications"

Volume

Processing increasing huge data sets

Variety

Processing different types of data

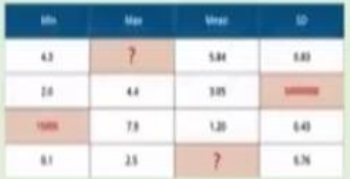
Velocity

Data is being generated at an alarming rate

Value

Finding correct meaning out of the data

Veracity

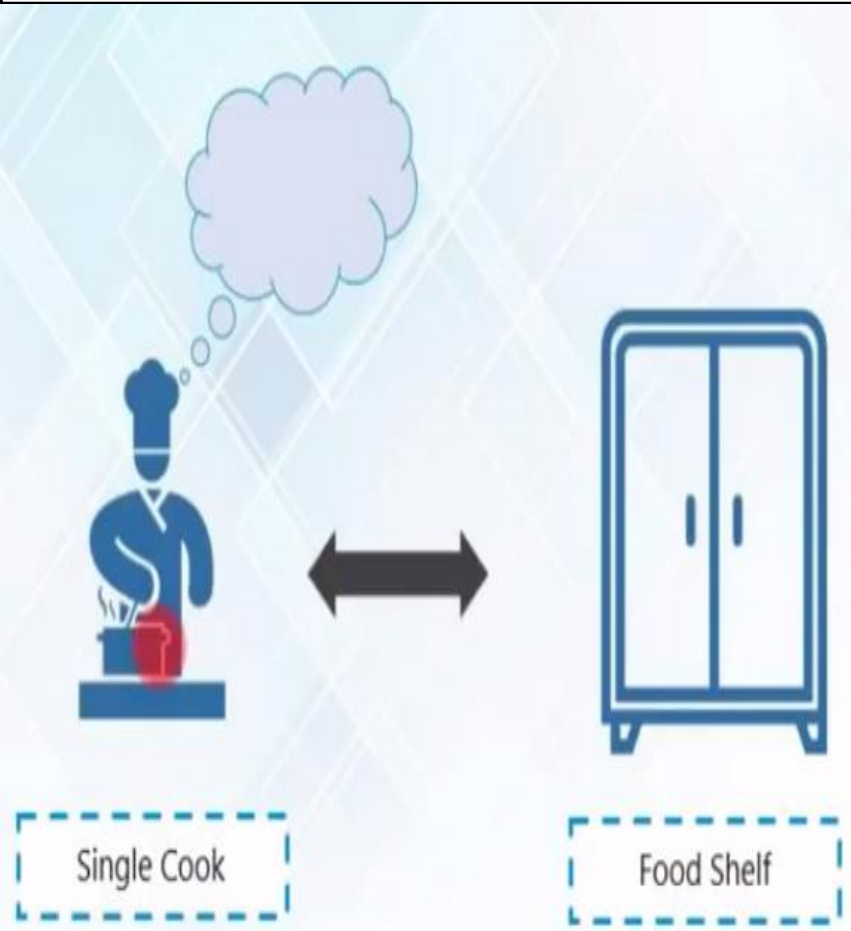


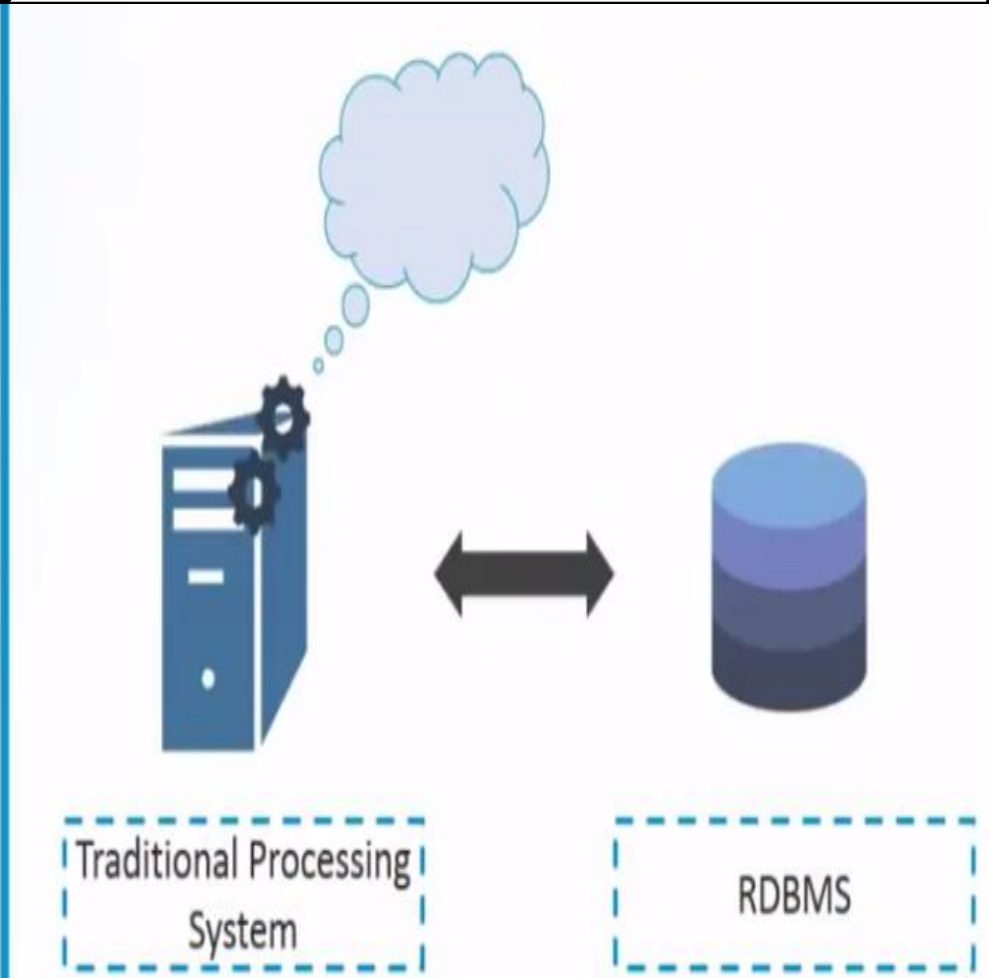
Uncertainty and inconsistencies in the data

Story of Big Data & Traditional System

Scenario:
Bob has opened a small restaurant in his city

Traditional System Scenario





Traditional System Scenario

Traditional Scenario:

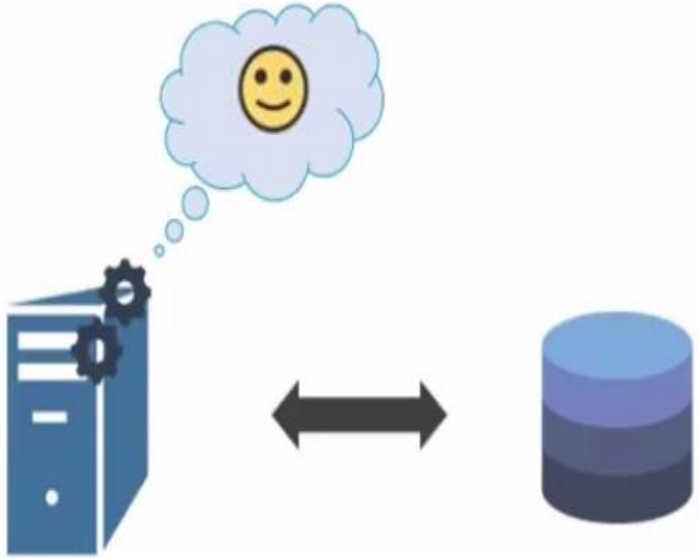
2 orders per hour

Single Cook

Food Shelf

Traditional Scenario:

Data is generated at a steady rate and is structured in nature



Traditional Processing System

RDBMS

Failure of Traditional System Scenario

Scenario 2:

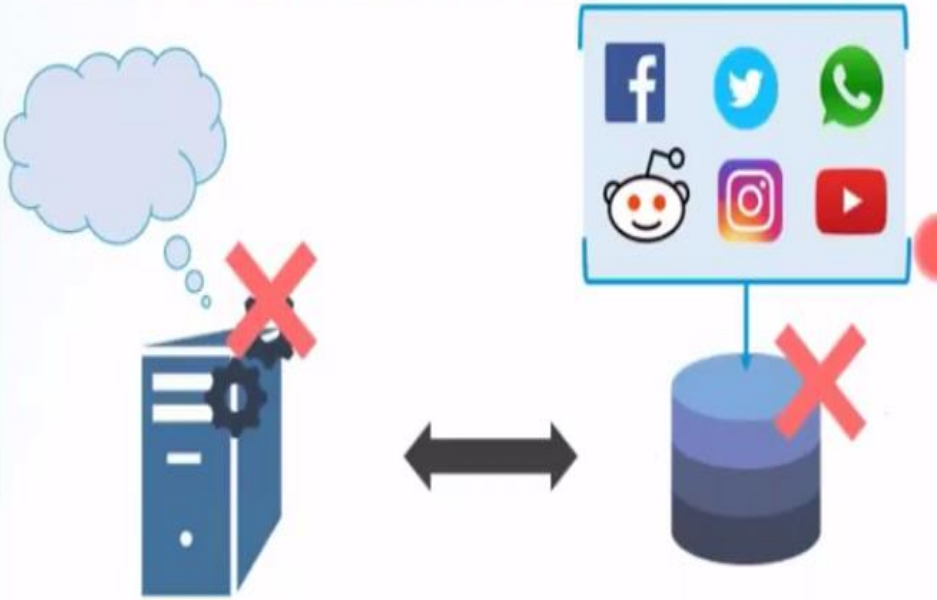
- They started taking Online orders
- 10 orders per hour

Single Cook

Food Shelf

Big Data Scenario:

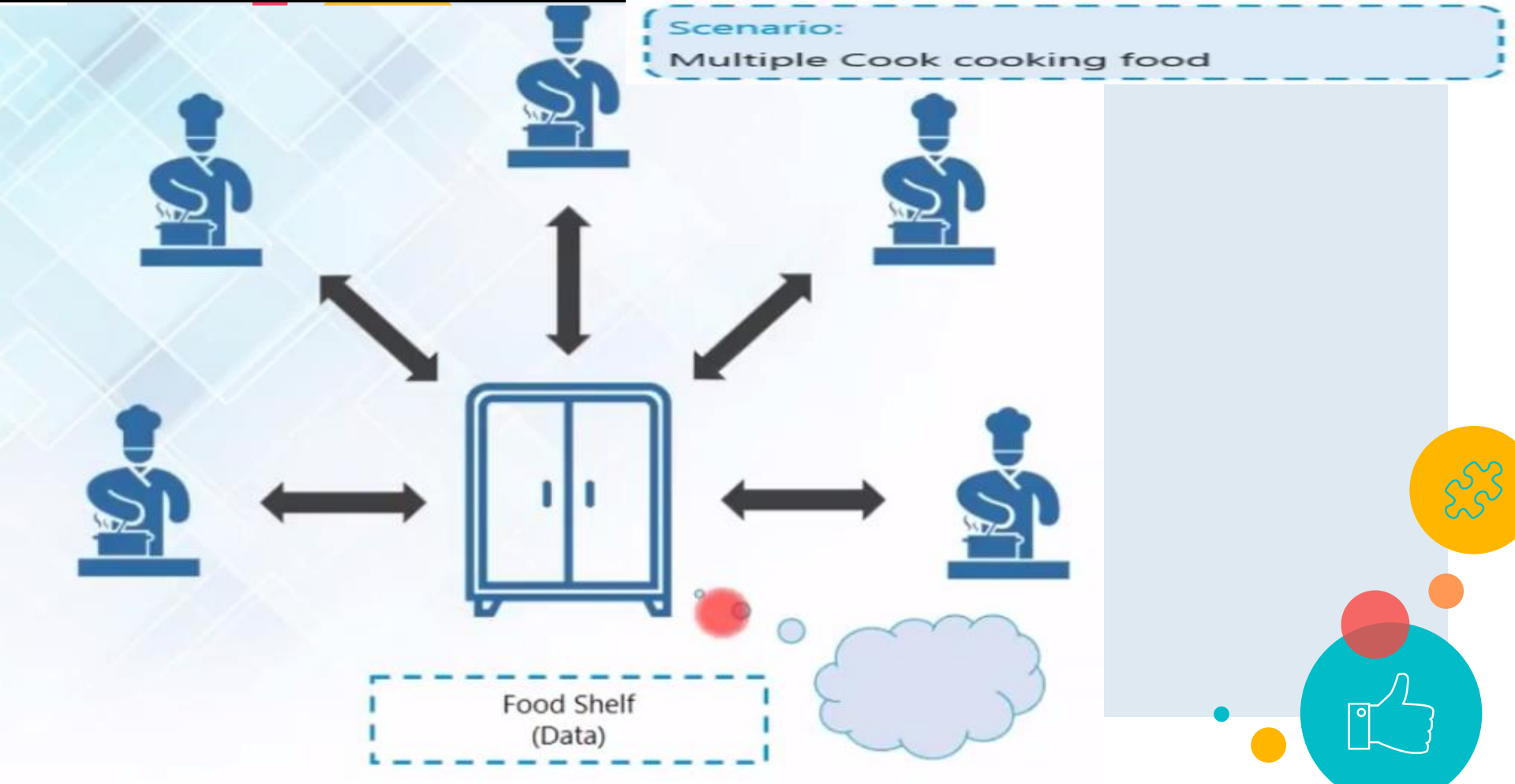
Heterogenous data is being generated at an alarming rate by multiple sources



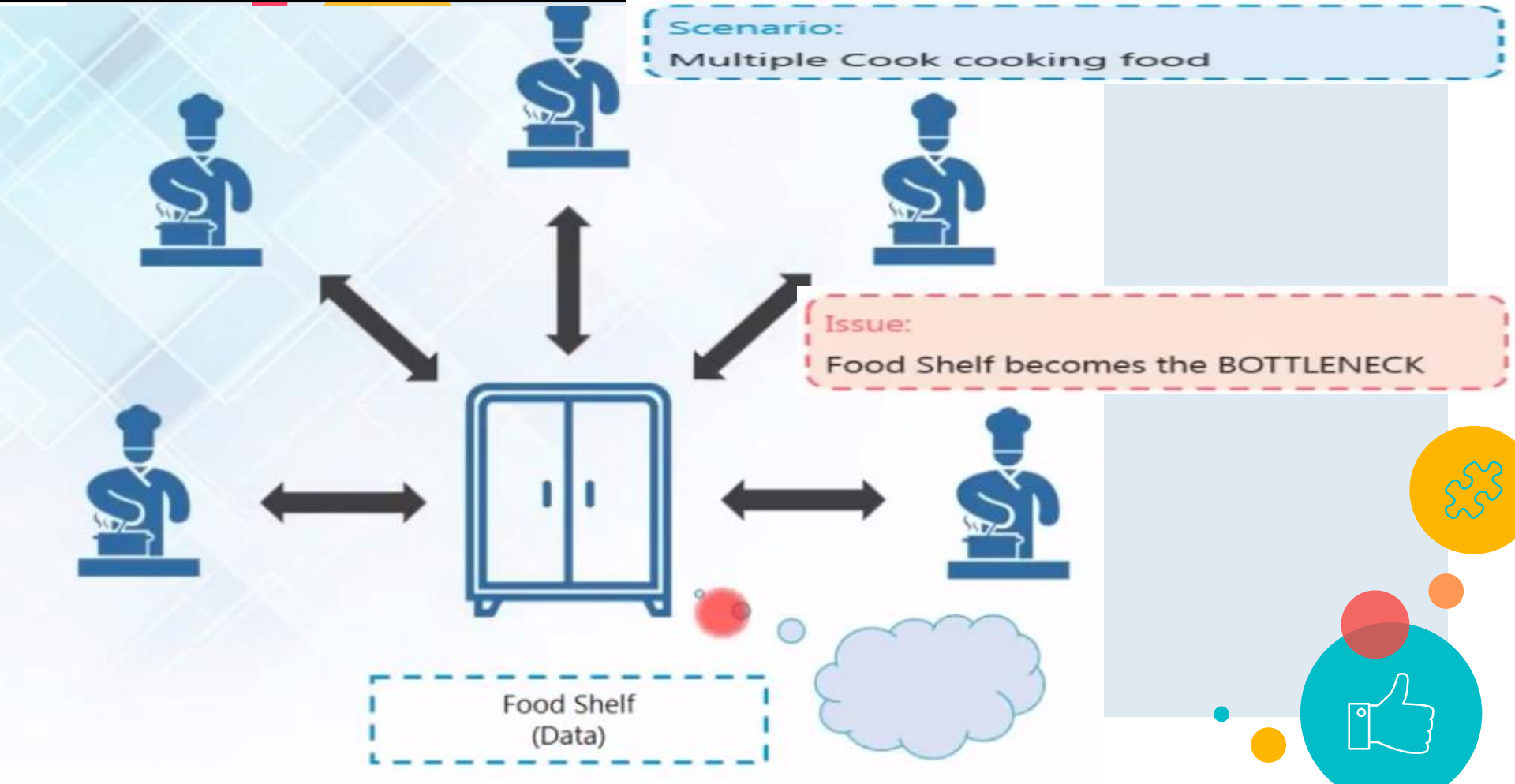
Traditional Processing System

RDBMS

Need of Effective Solution



Need of Effective Solution

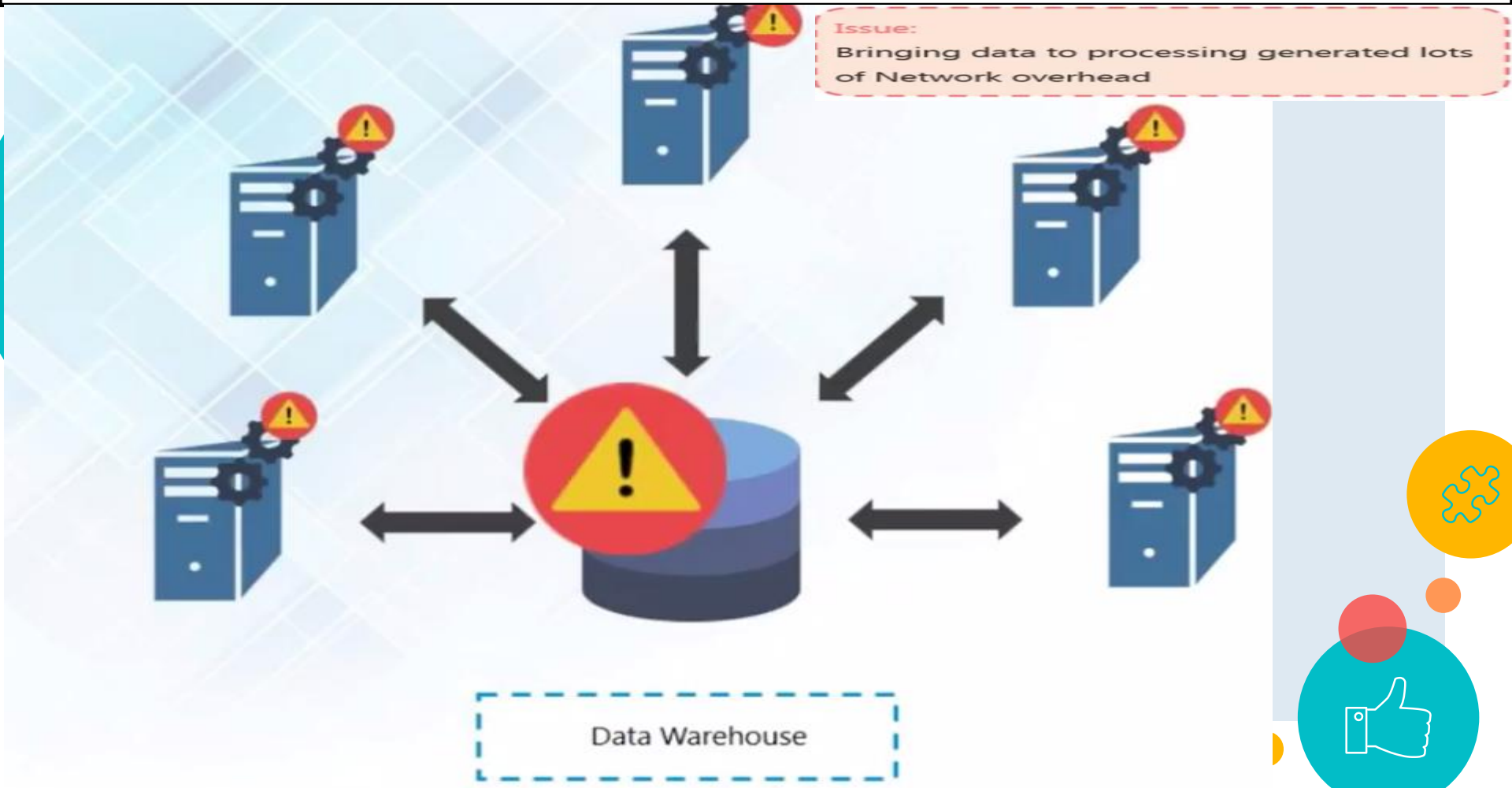


Need of Effective Solution

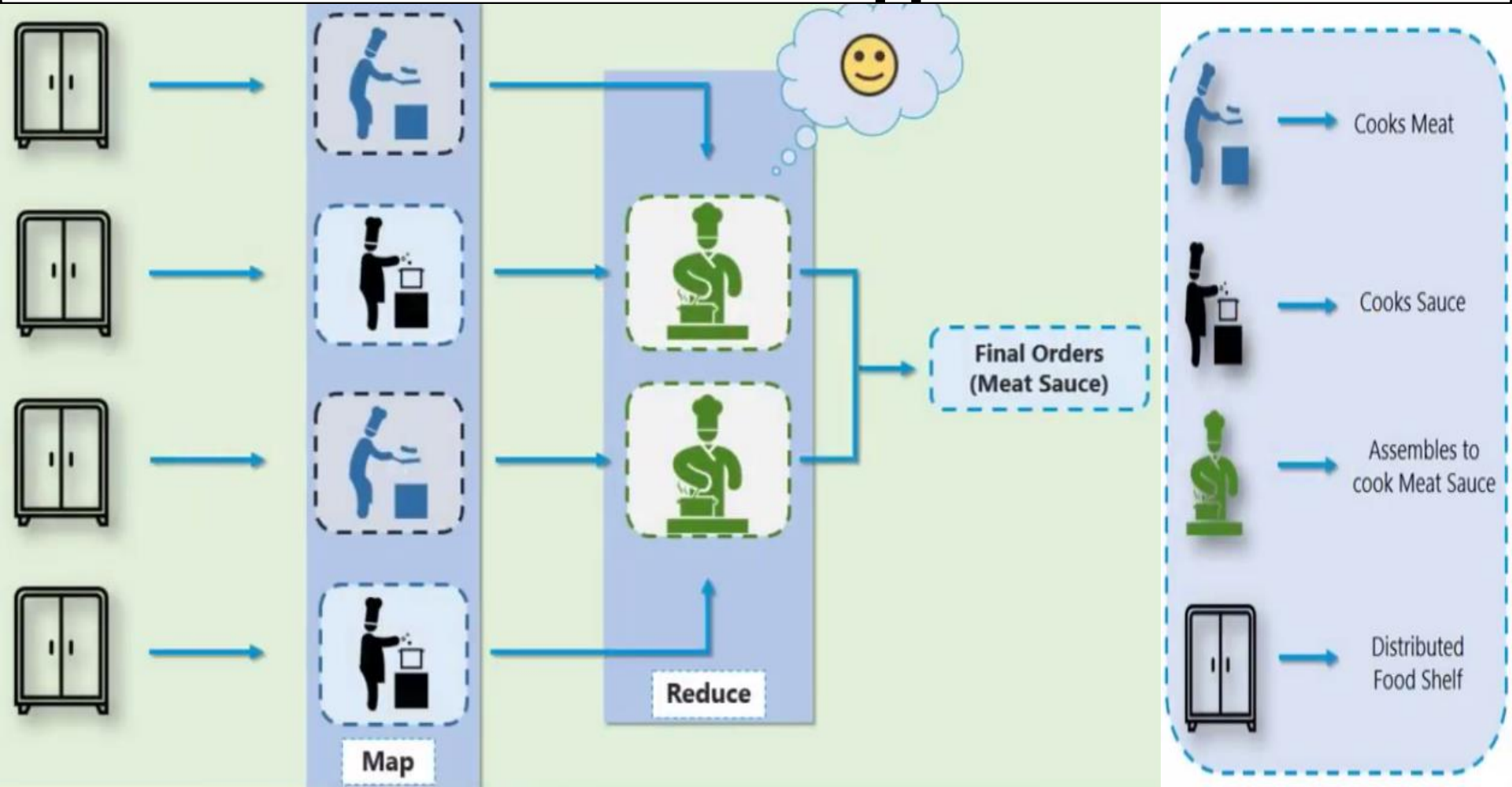
Scenario:

Multiple Processing Unit for data processing

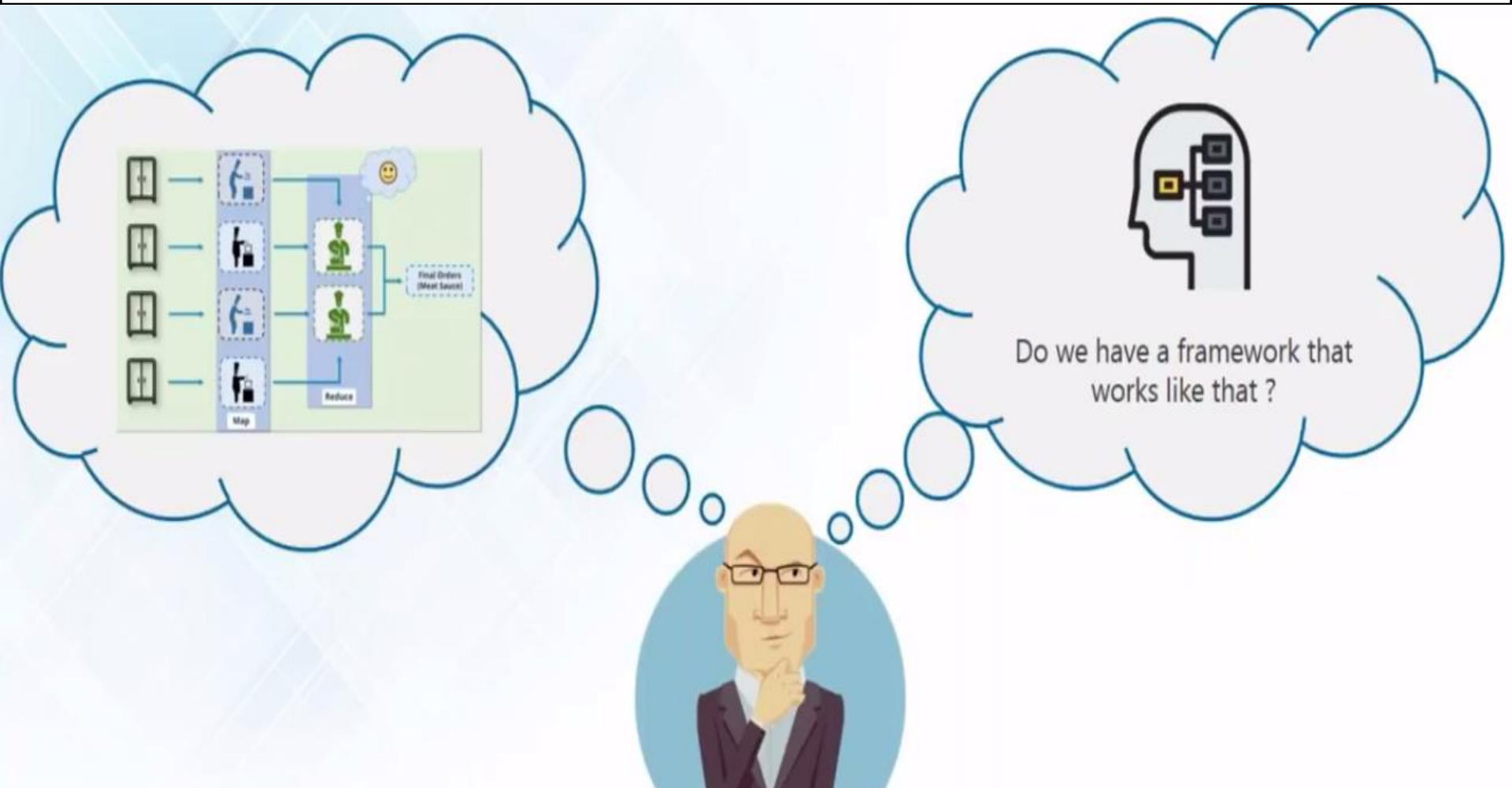
Need of Effective Solution



Distributed and Parallel Approach Solution

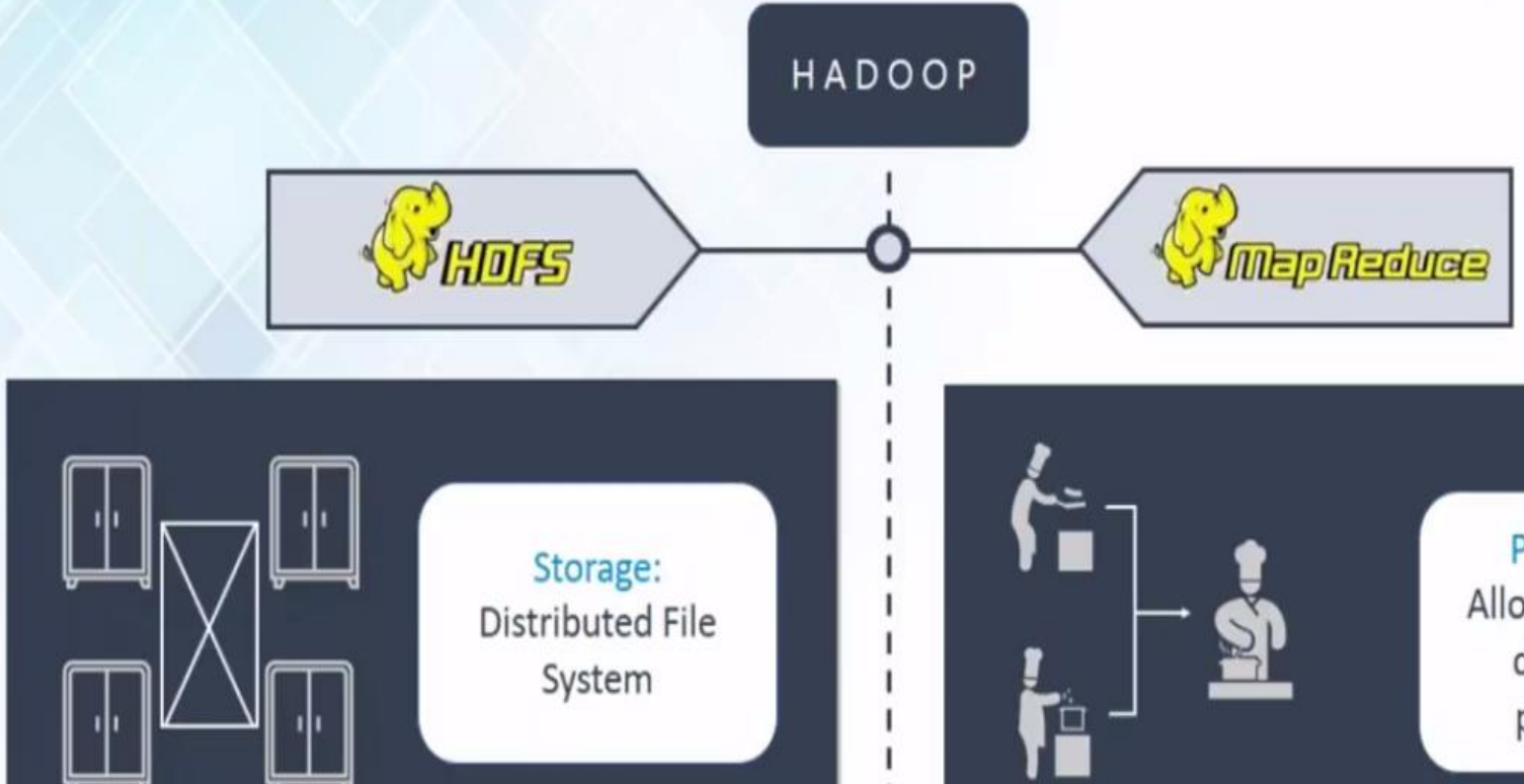


Need for a FrameWork



Apache Hadoop Framework to Process Big Data

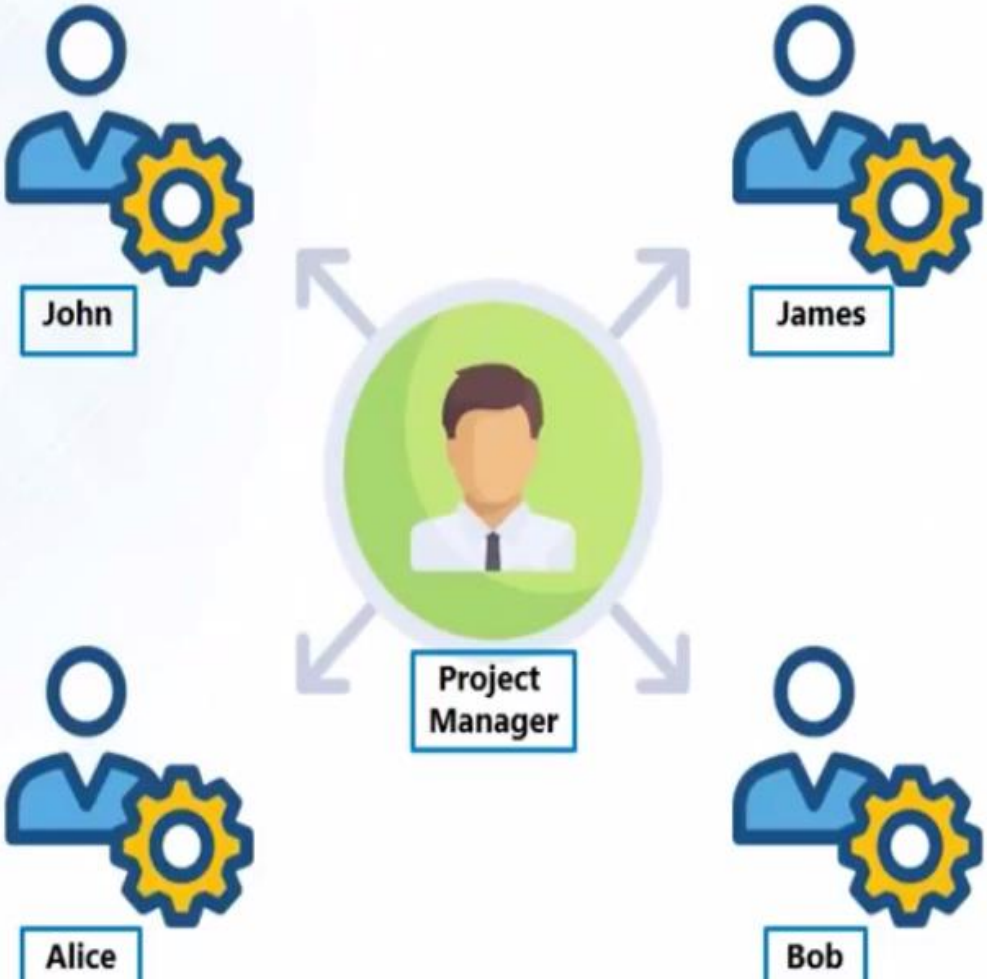
Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion



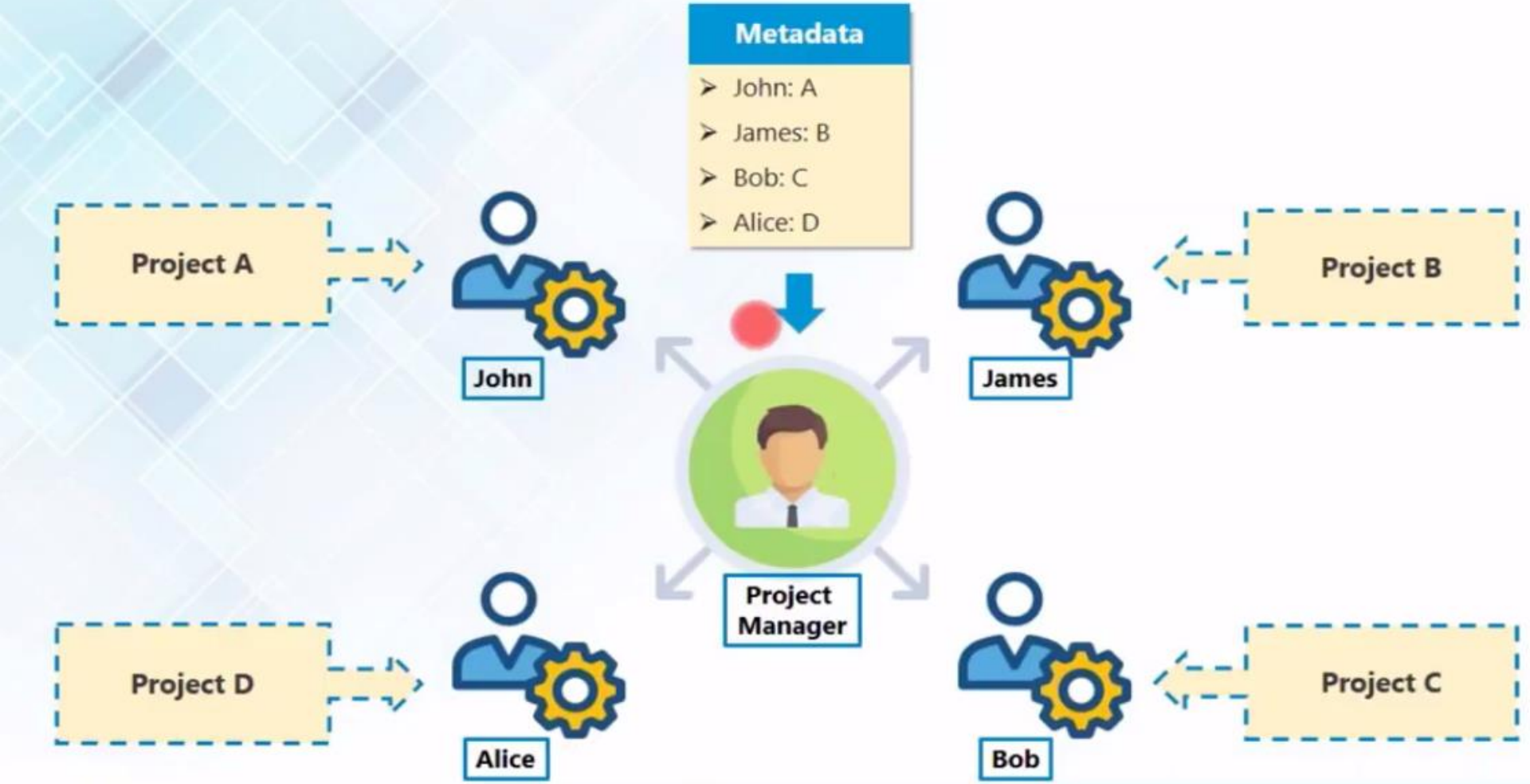
Apache Hadoop Master & Slave Architecture

Scenario:

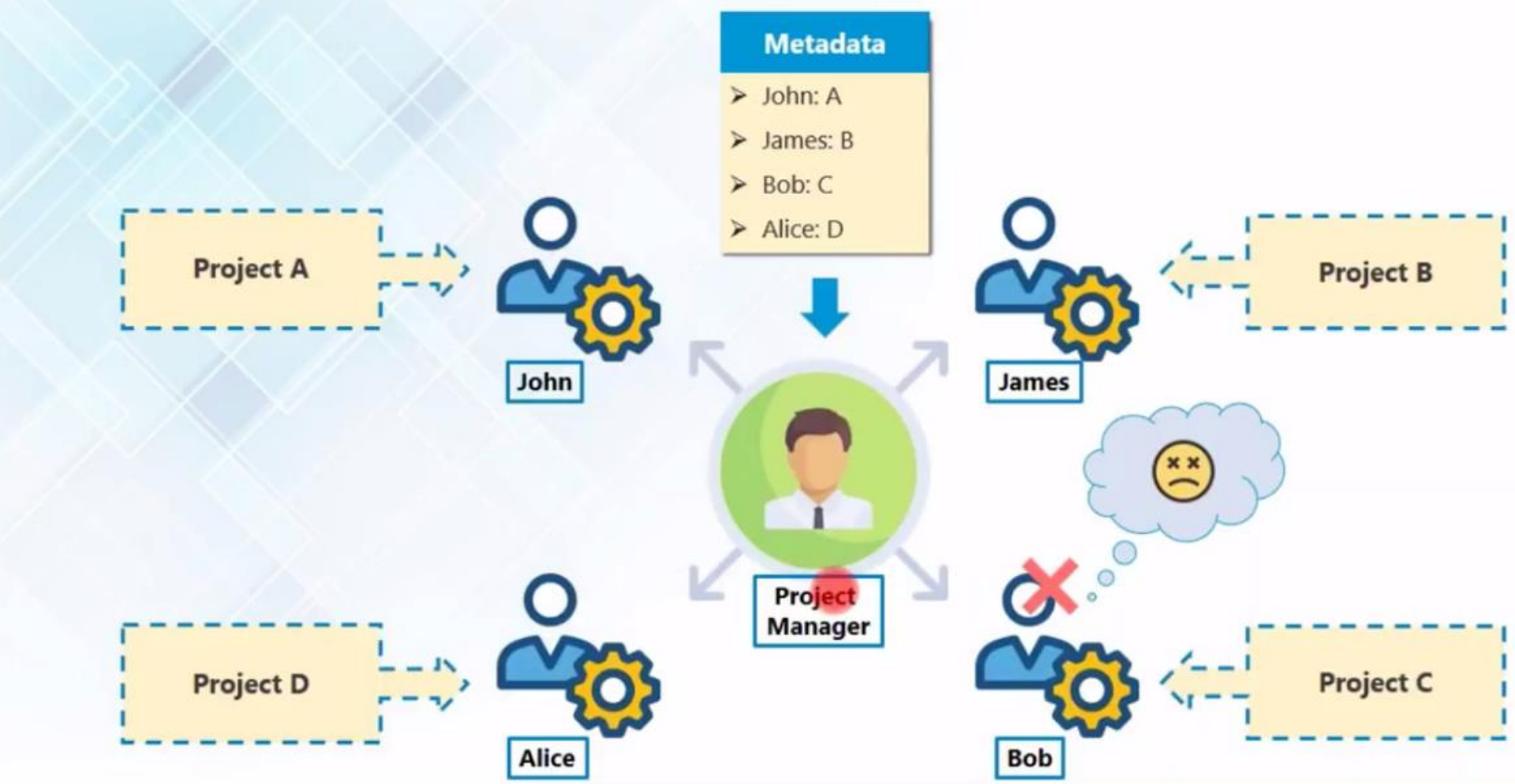
A project Manager managing a team of four employees. He assigns project to each of them and tracks the progress



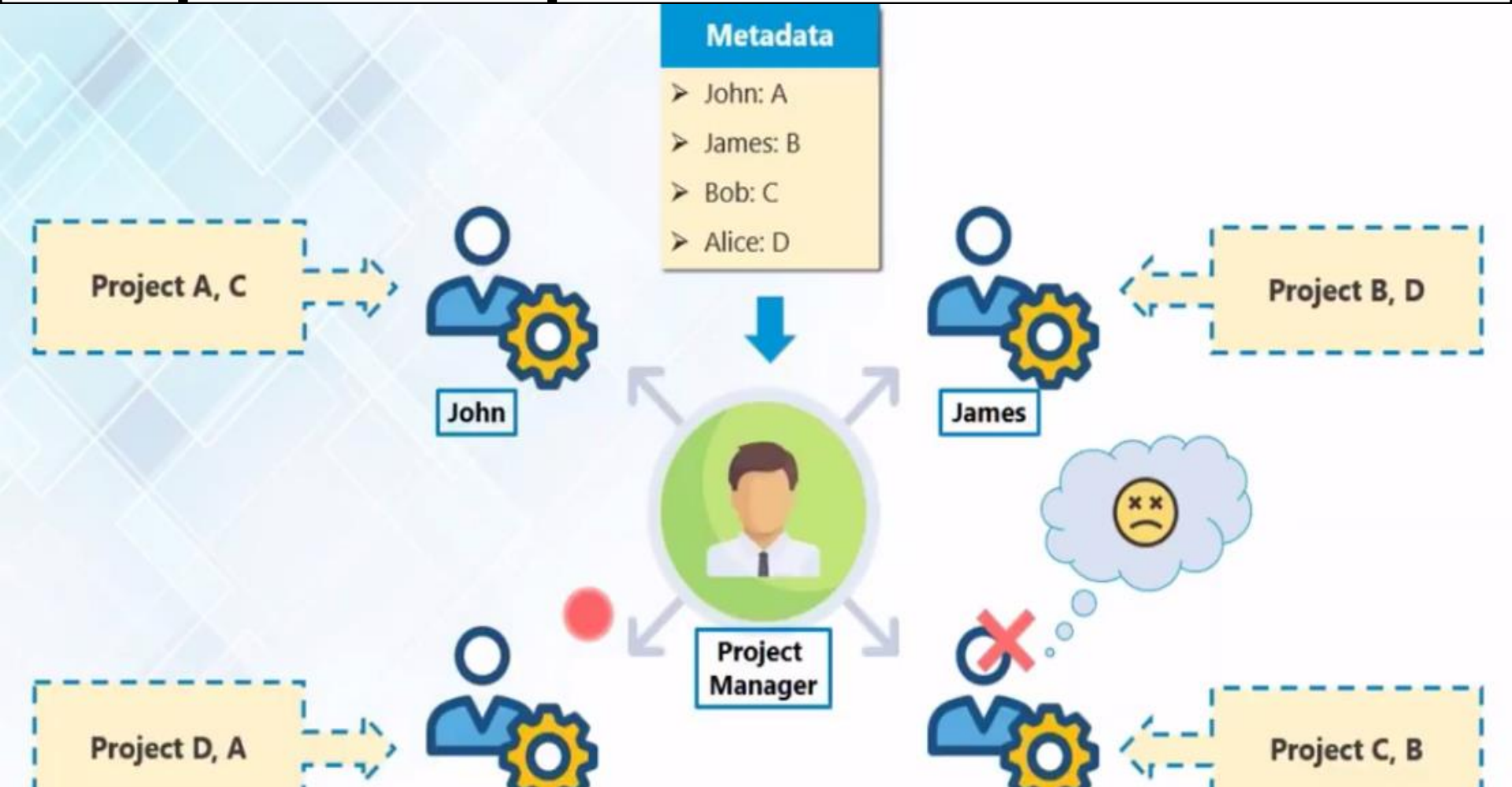
Apache Hadoop Master & Slave Architecture



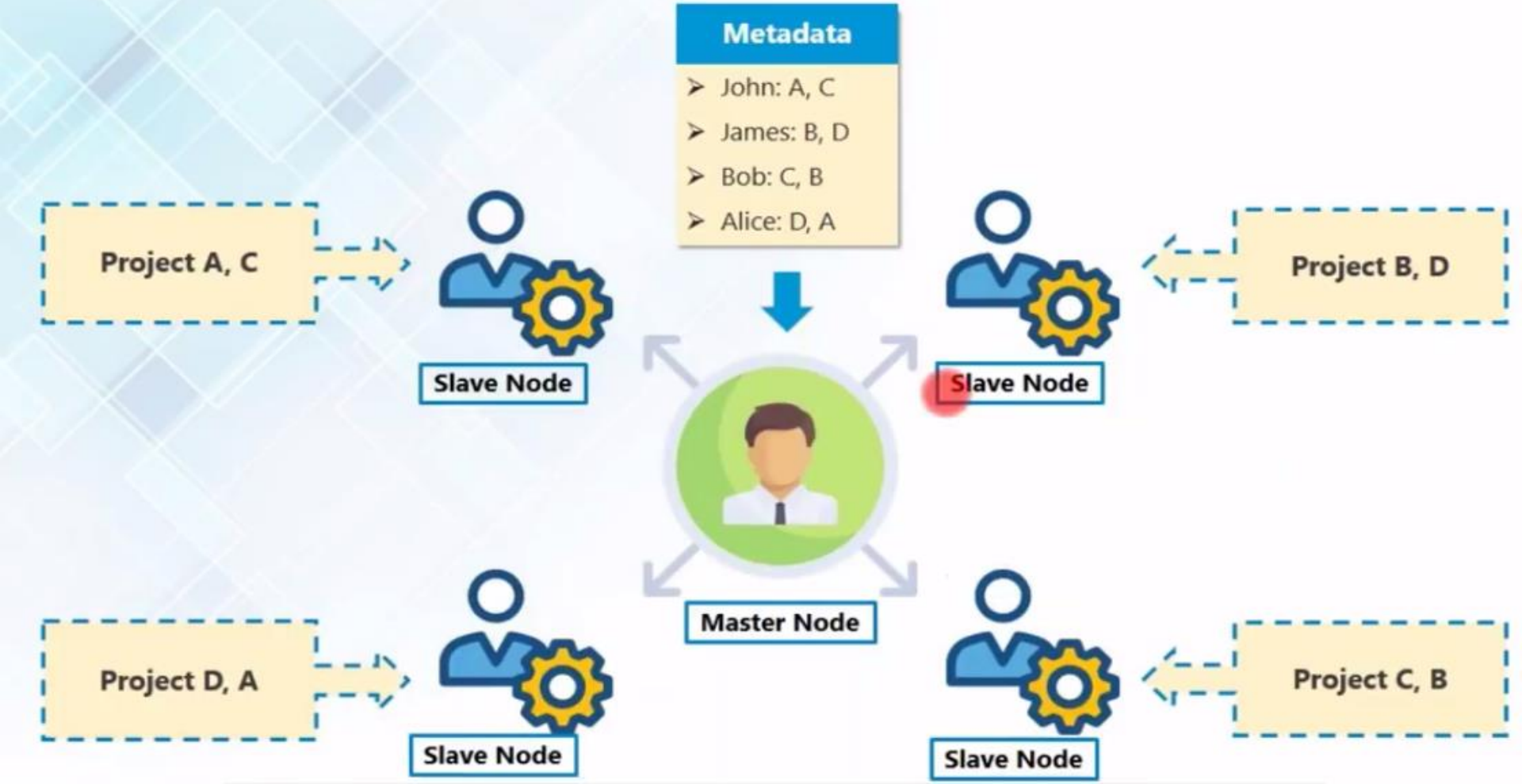
Apache Hadoop Master & Slave Architecture



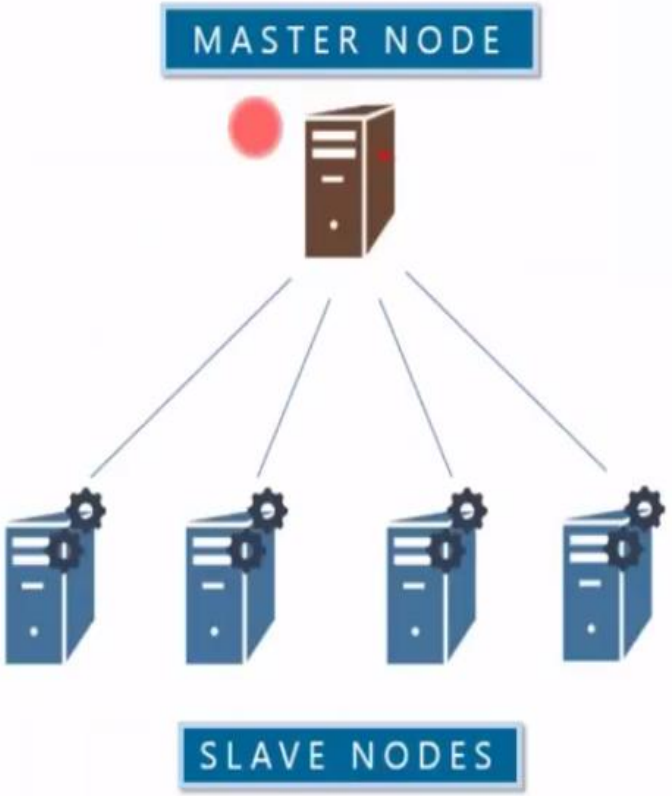
Apache Hadoop Master & Slave Architecture



Apache Hadoop Master & Slave Architecture



Apache Hadoop Master & Slave Architecture



HADOOP CORE COMPONENTS

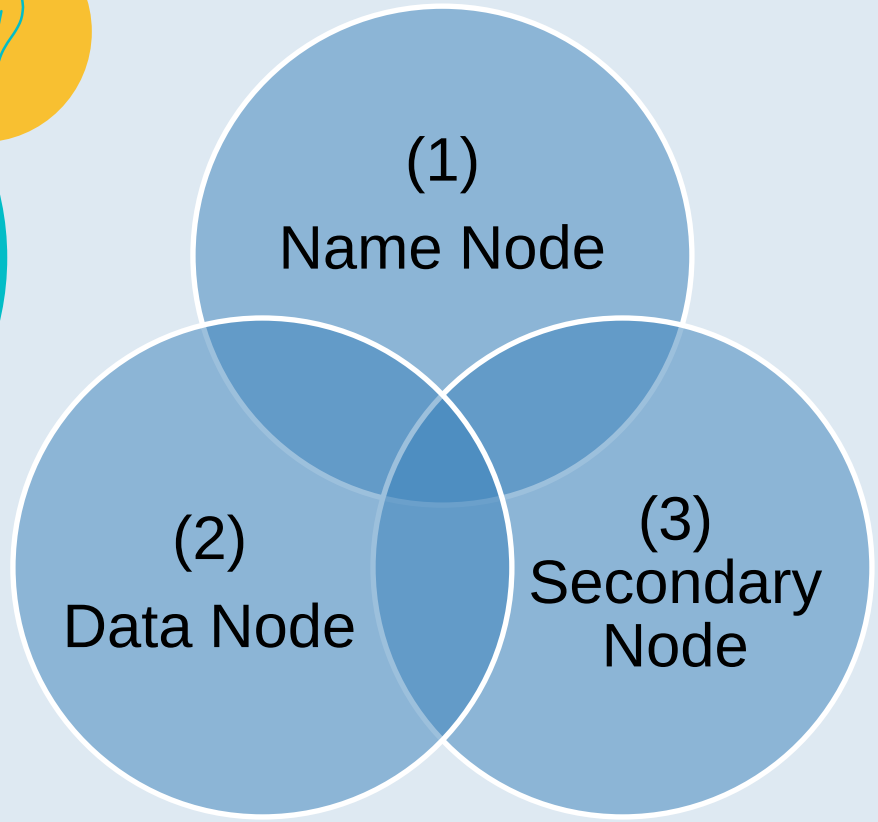
HDFS

MapReduce

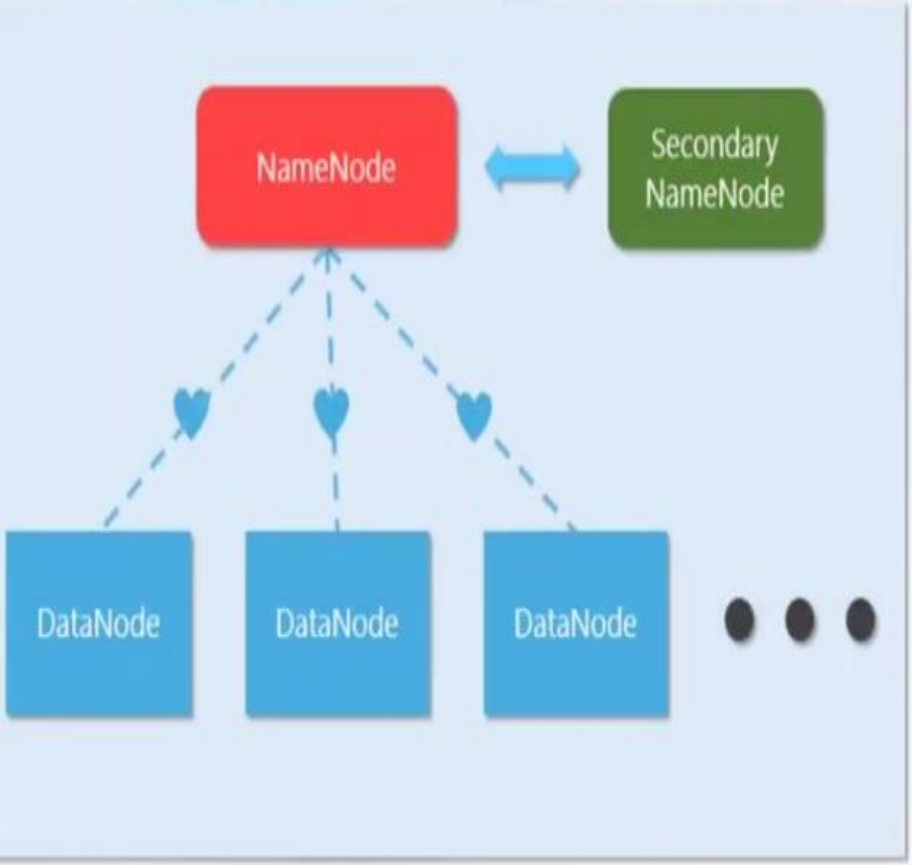
Storage:
Distributed File System

Processing:
Allows parallel & distributed processing

HDFS Core Components



HDFS Core Components-Name Node & Data Node



NameNode:

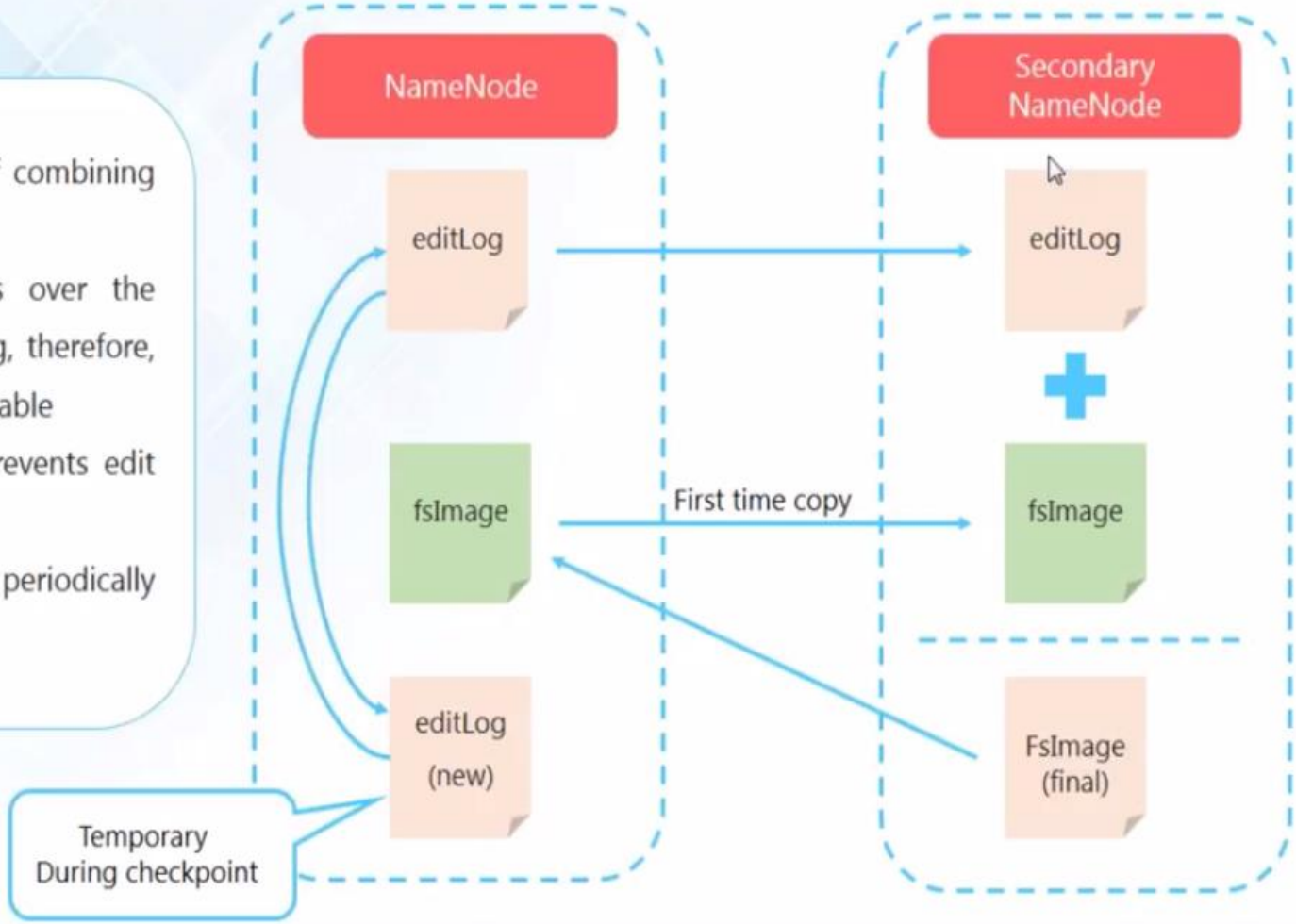
- Maintains and Manages DataNodes
- Records metadata i.e. information about data blocks e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

DataNode:

- Slave daemons
- Stores actual data
- Serves read and write requests from the clients

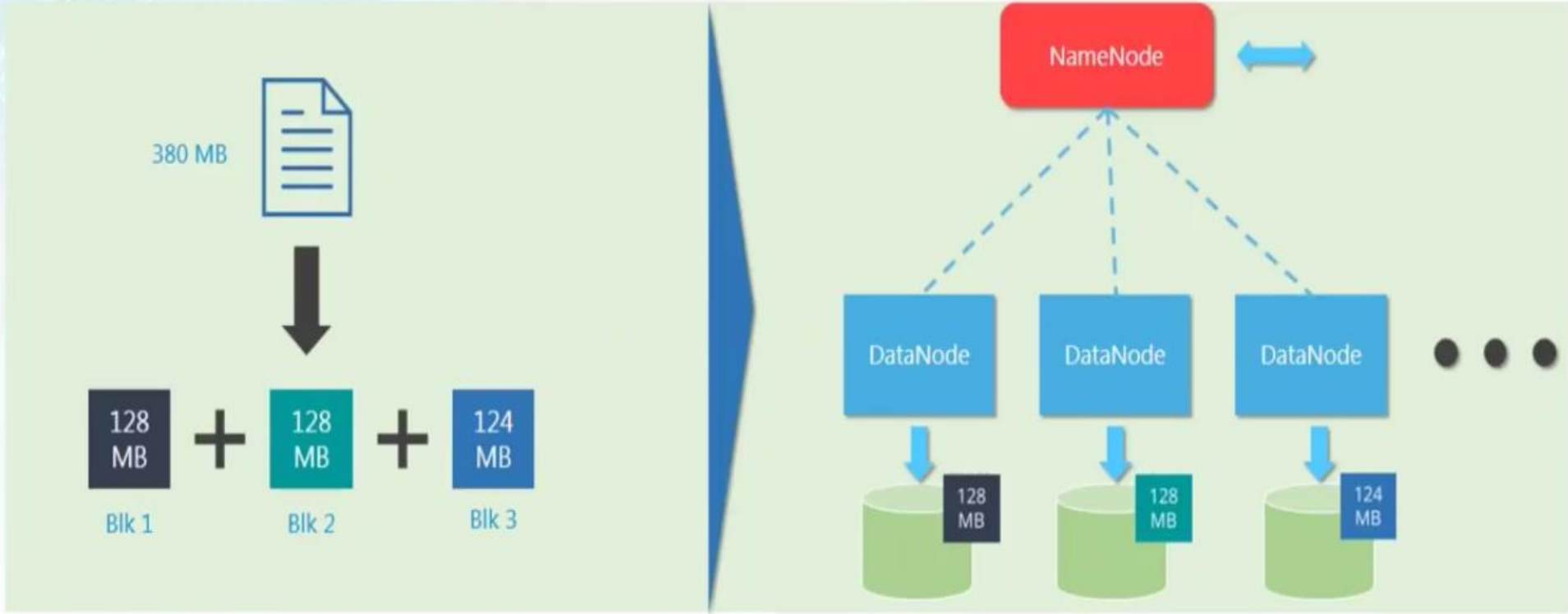
Secondary Name Node & Check pointing

- Checkpointing is a process of combining edit logs with FsImage
- Secondary NameNode takes over the responsibility of checkpointing, therefore, making NameNode more available
- Allows faster Failover as it prevents edit logs from getting too huge
- Checkpointing happens periodically (default: 1 hour)



HDFS Data Blocks

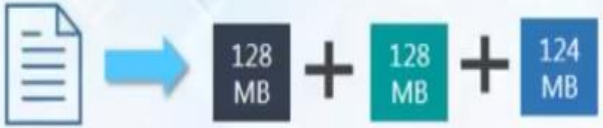
- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x)

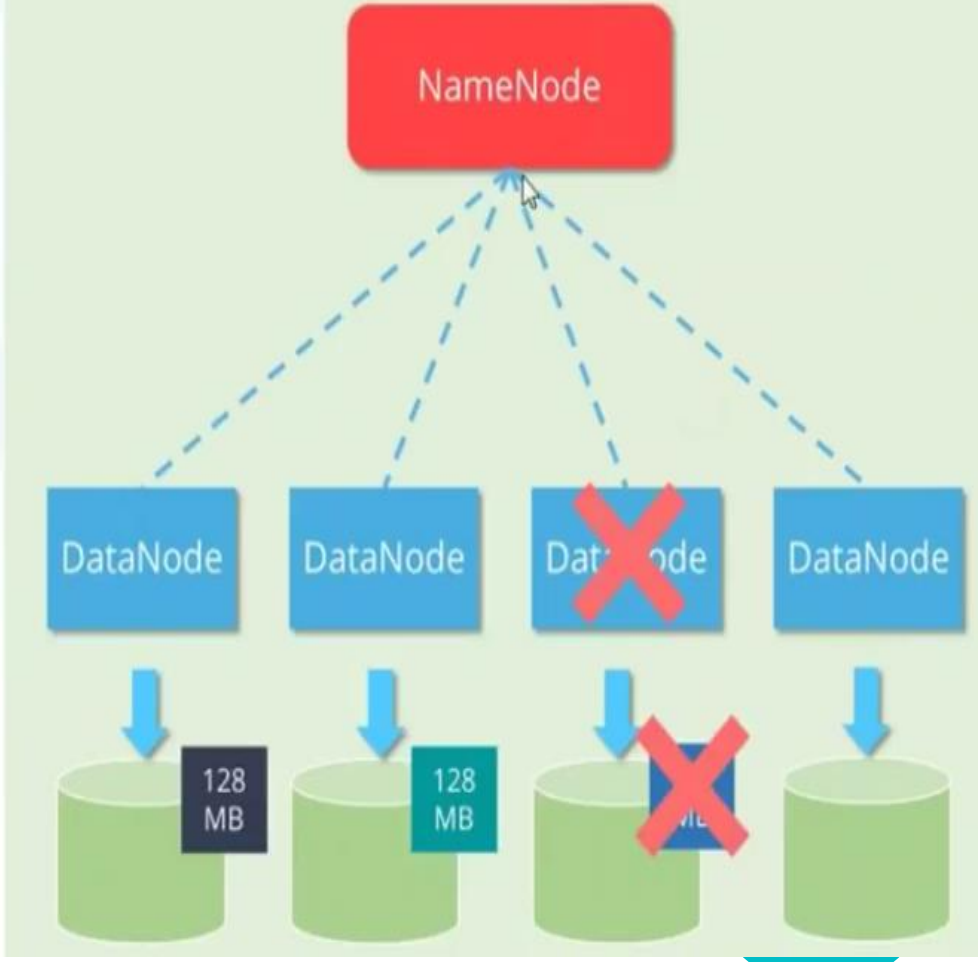


Fault Tolerance :How Hadoop cope up with DataNode Failure

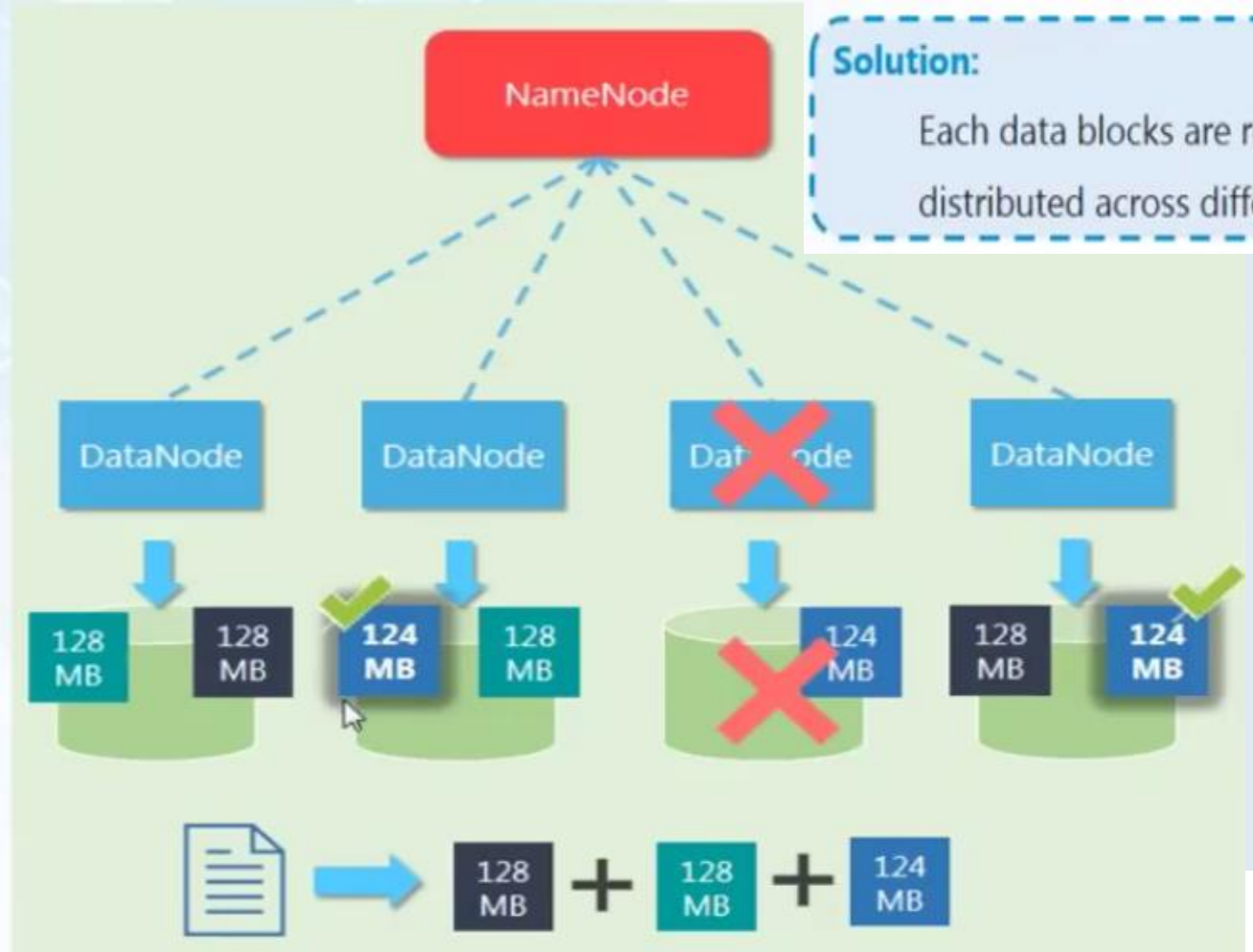
Scenario:

One of the DataNodes crashed containing the data blocks





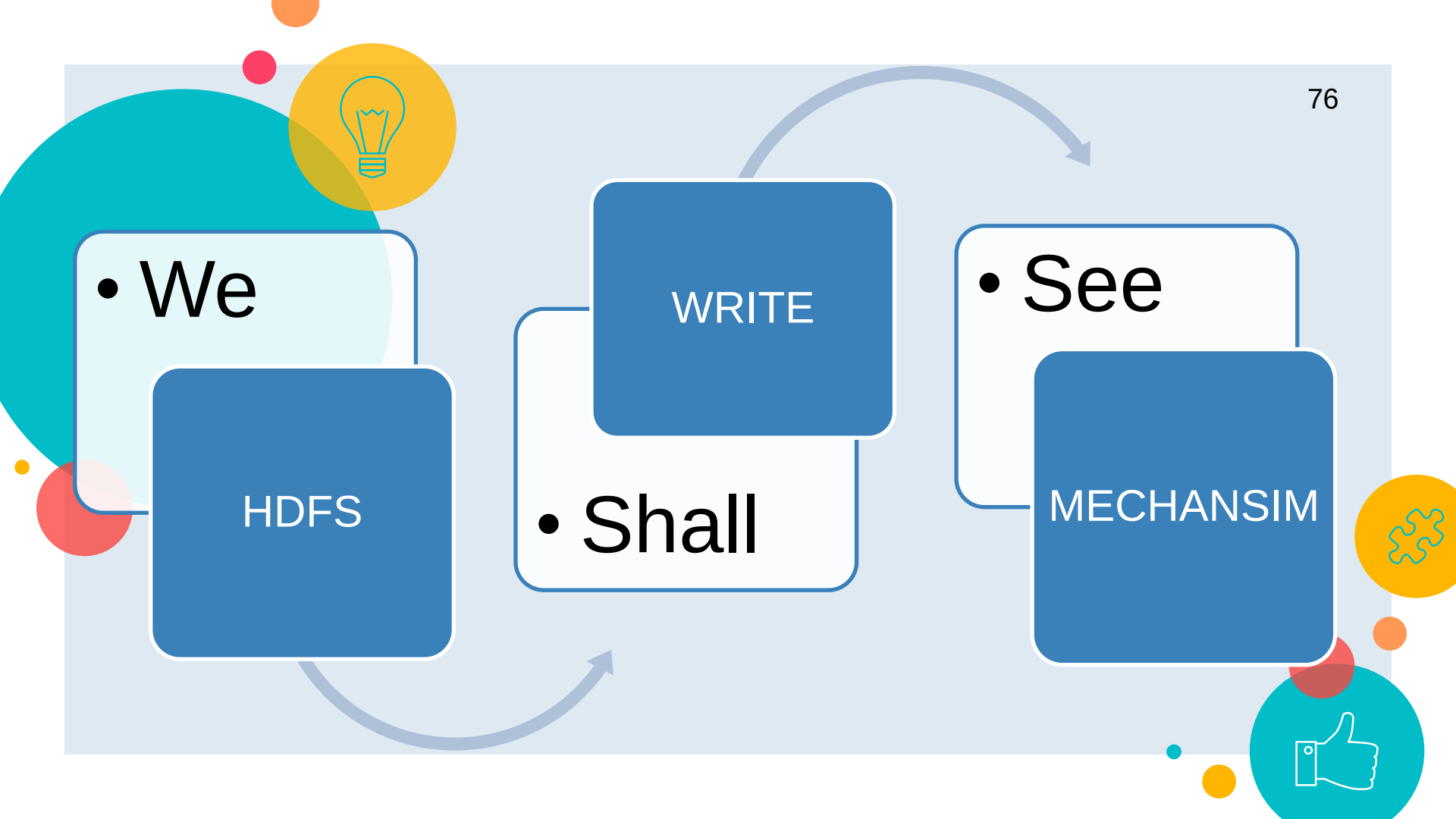
Fault Tolerance :Replication Factor



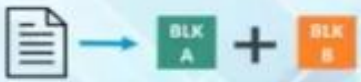
Solution:

Each data blocks are replicated (thrice by default) and are distributed across different DataNodes

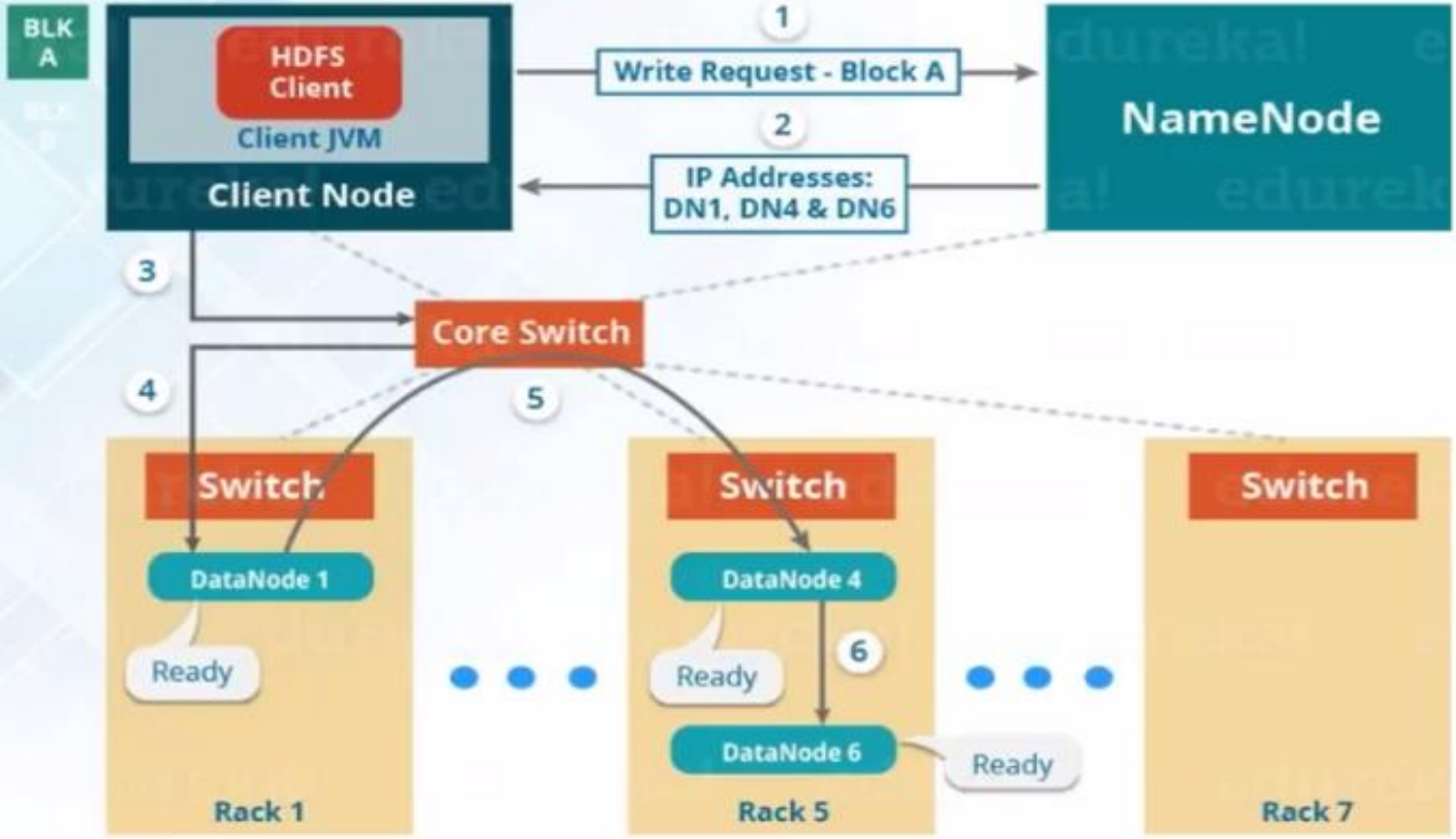
As it is said Never Put All Your Eggs in the Same Basket


• We
HDFS
WRITE
• Shall
• See
MECHANSIM

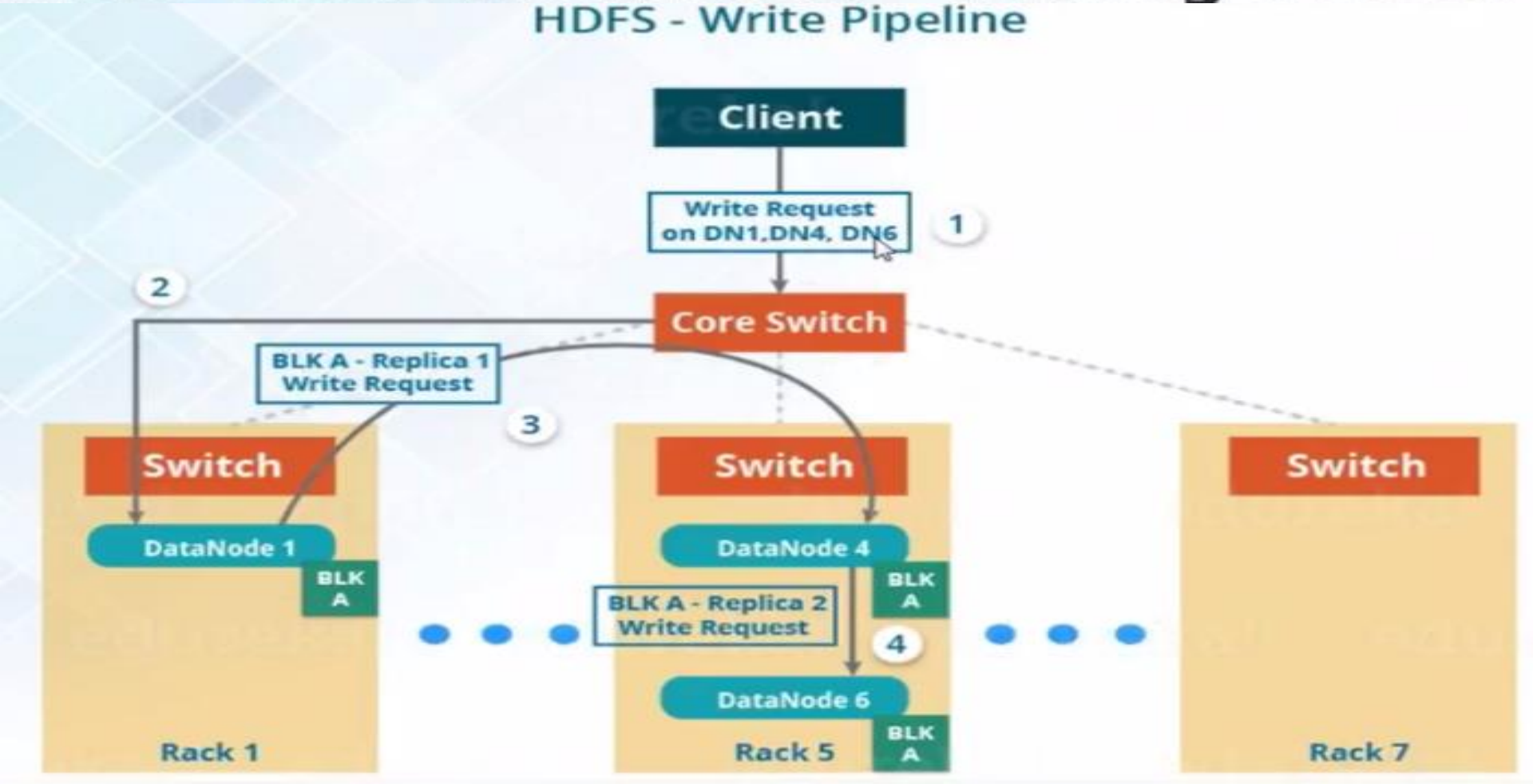
HDFS Write Mechanism – Pipeline Setup



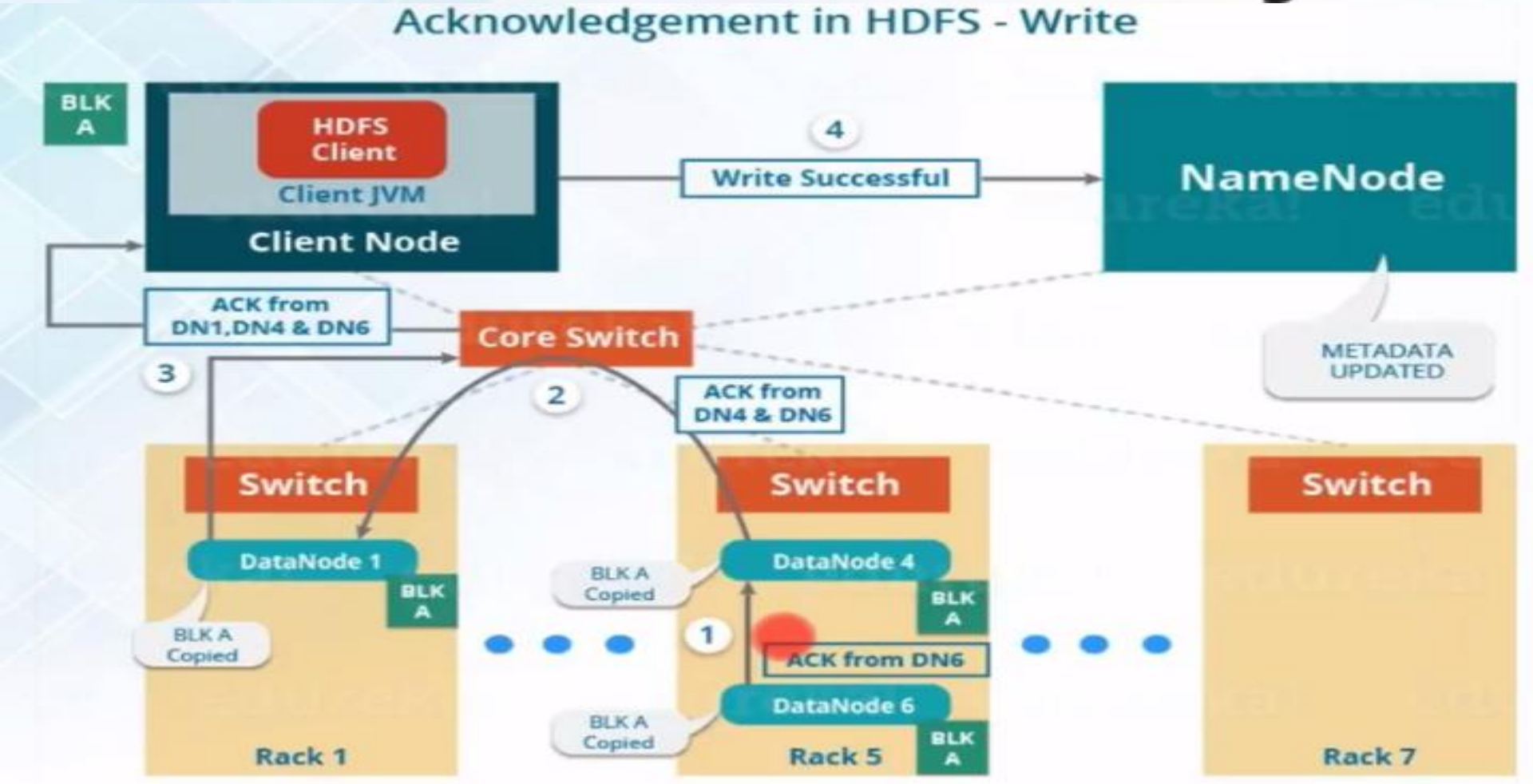
Setting up HDFS - Write Pipeline



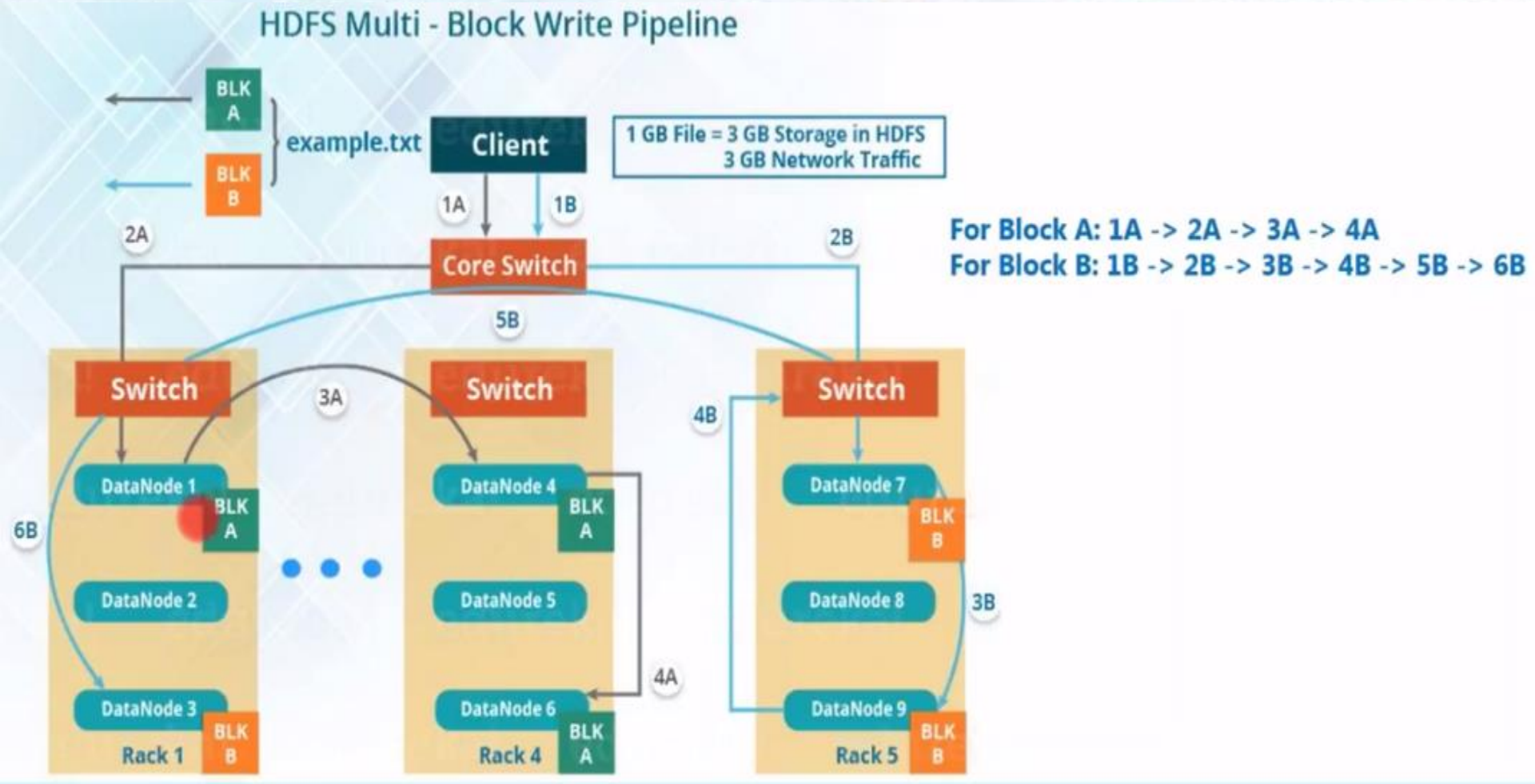
HDFS Write Mechanism – Writing a Block

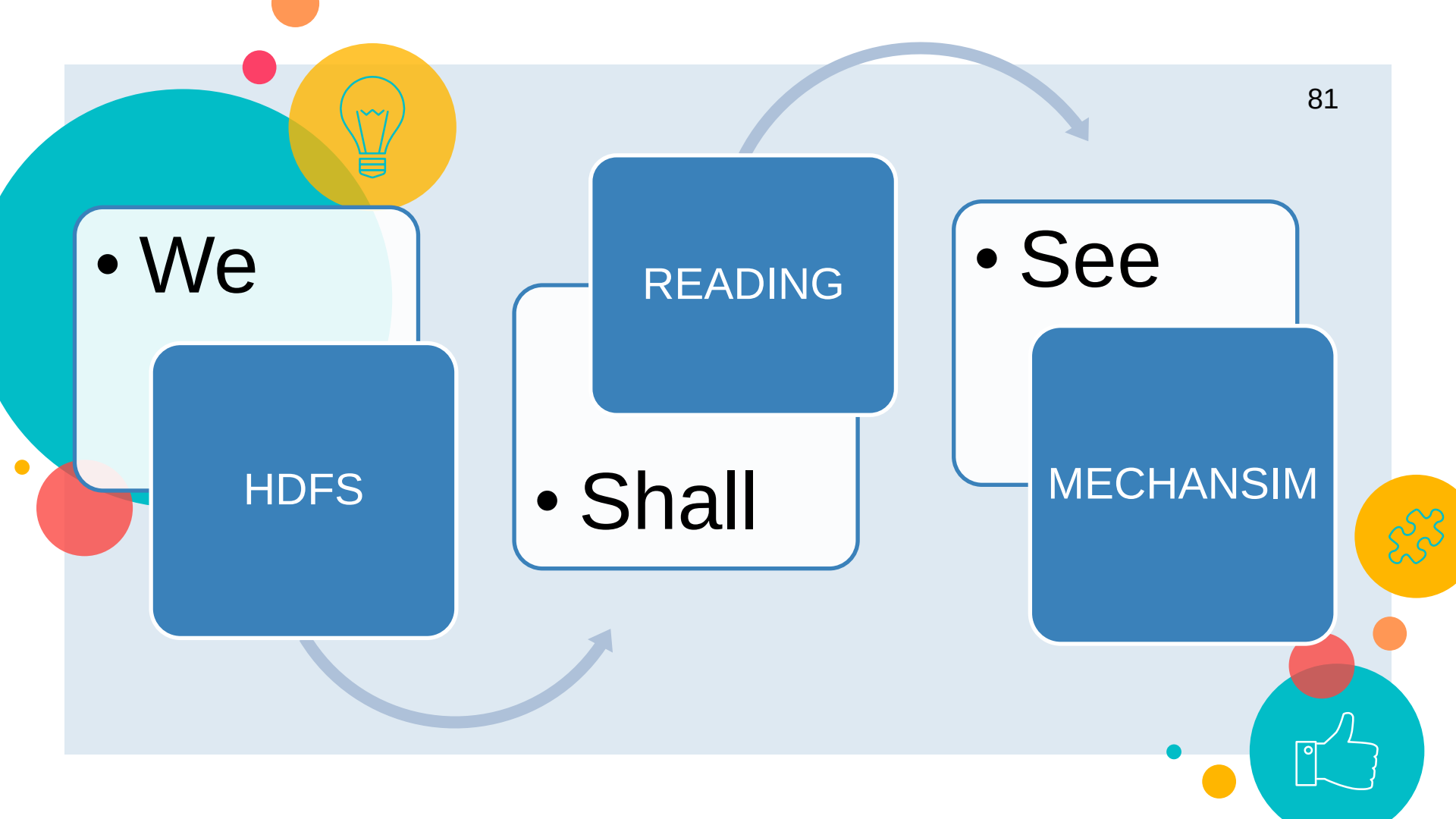


HDFS Write Mechanism - Acknowledgement

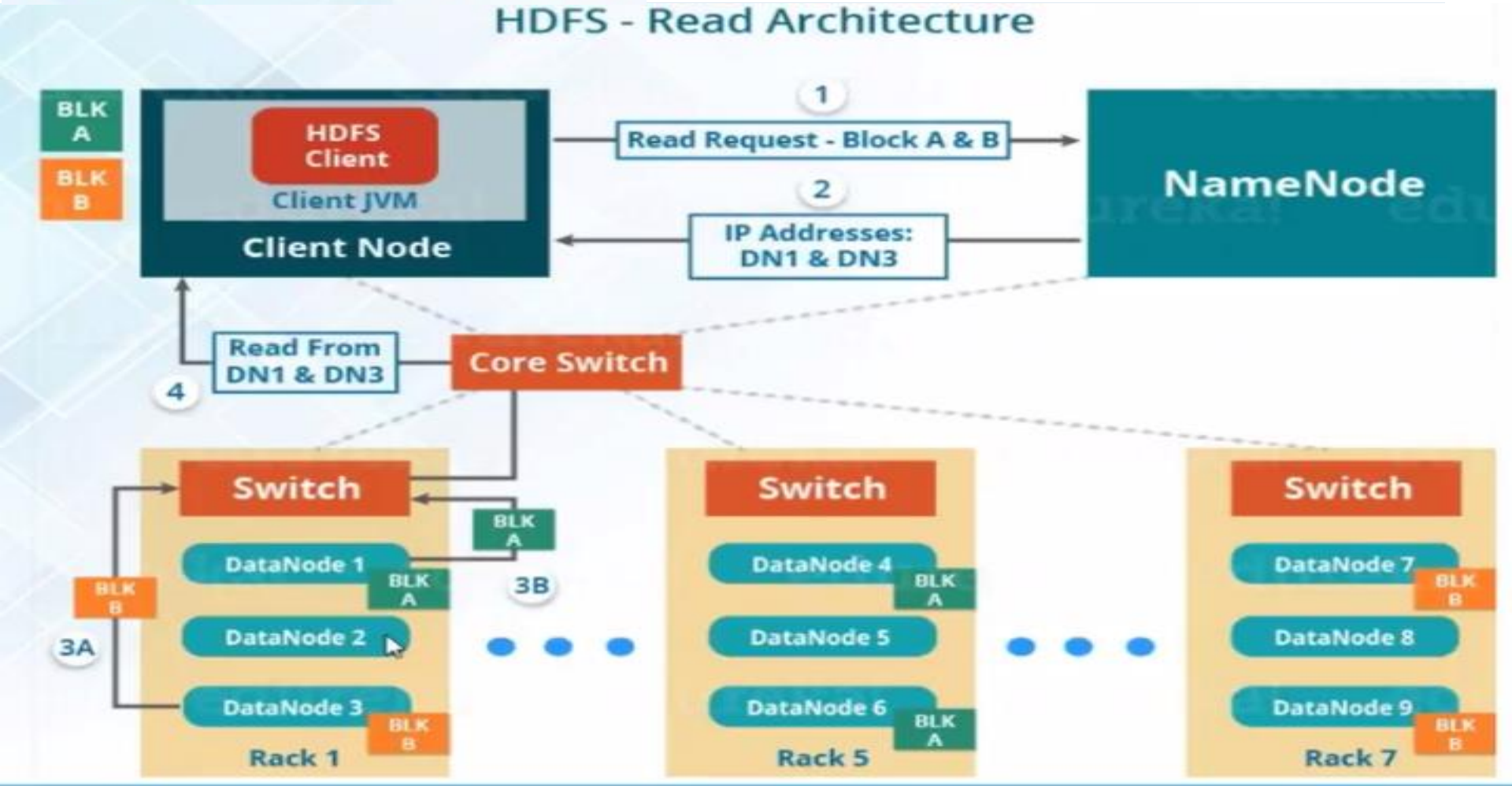


HDFS Multi-Block Write Mechanism




• We
HDFS
READING
• Shall
• See
MECHANSIM

HDFS Read Mechanism



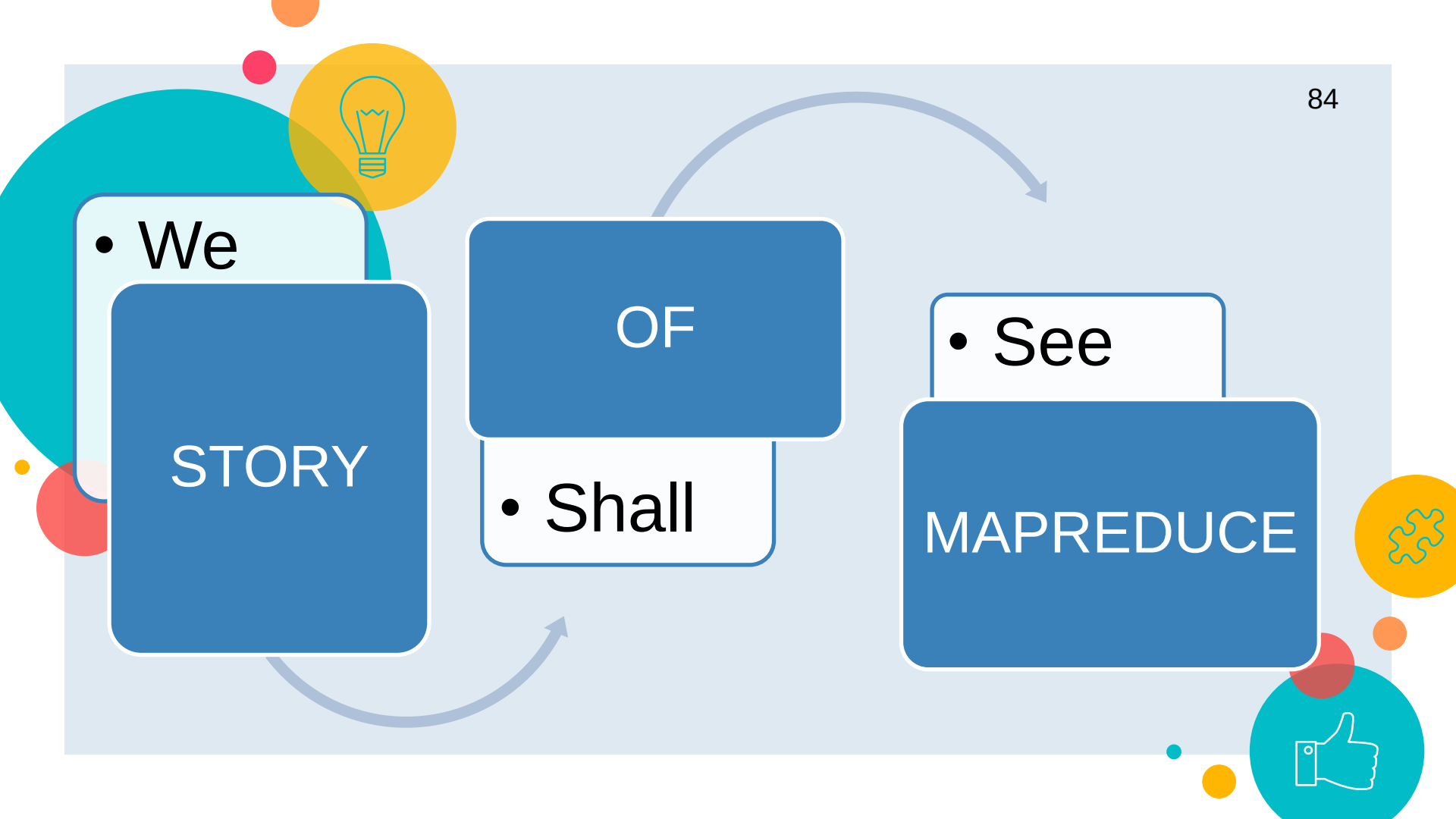
HADOOP CORE COMPONENTS

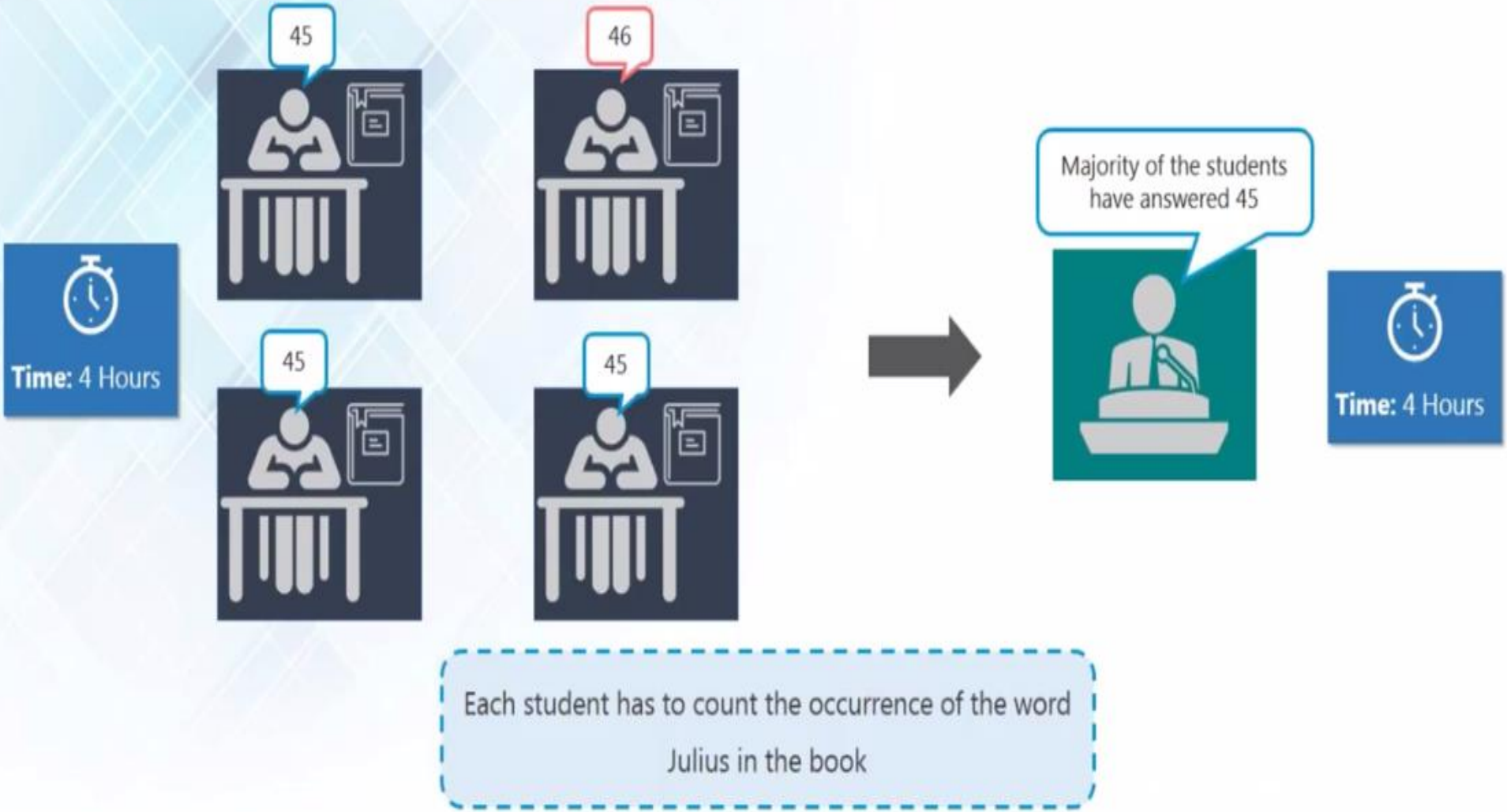
HDFS

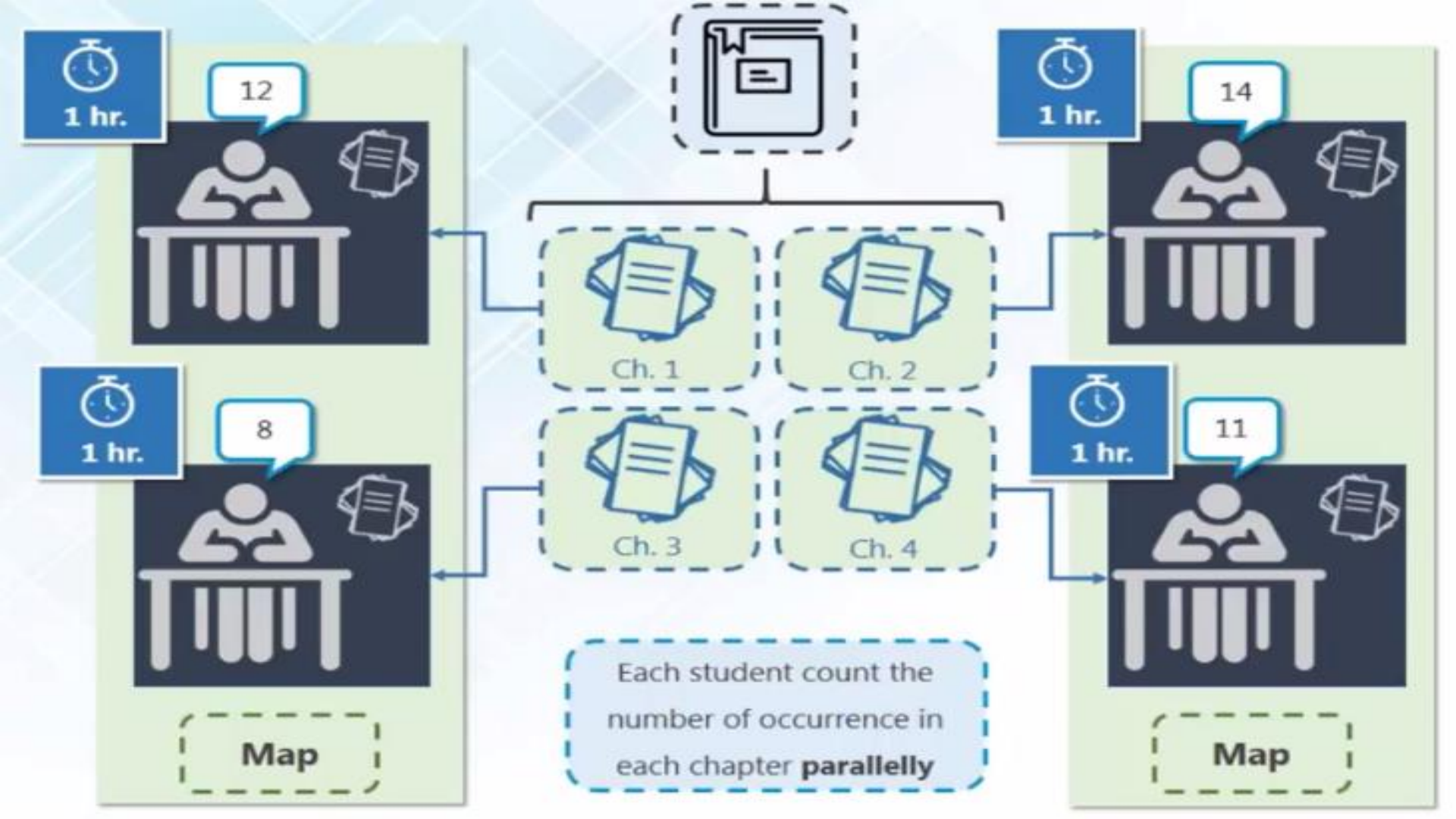
MapReduce

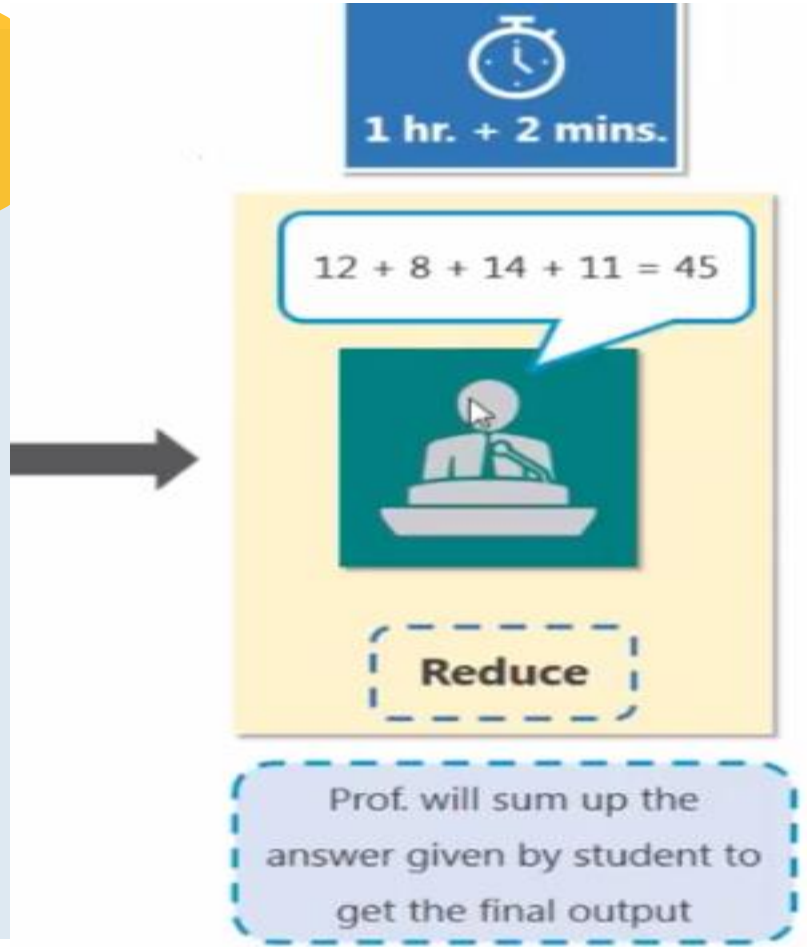
Storage:
Distributed File System

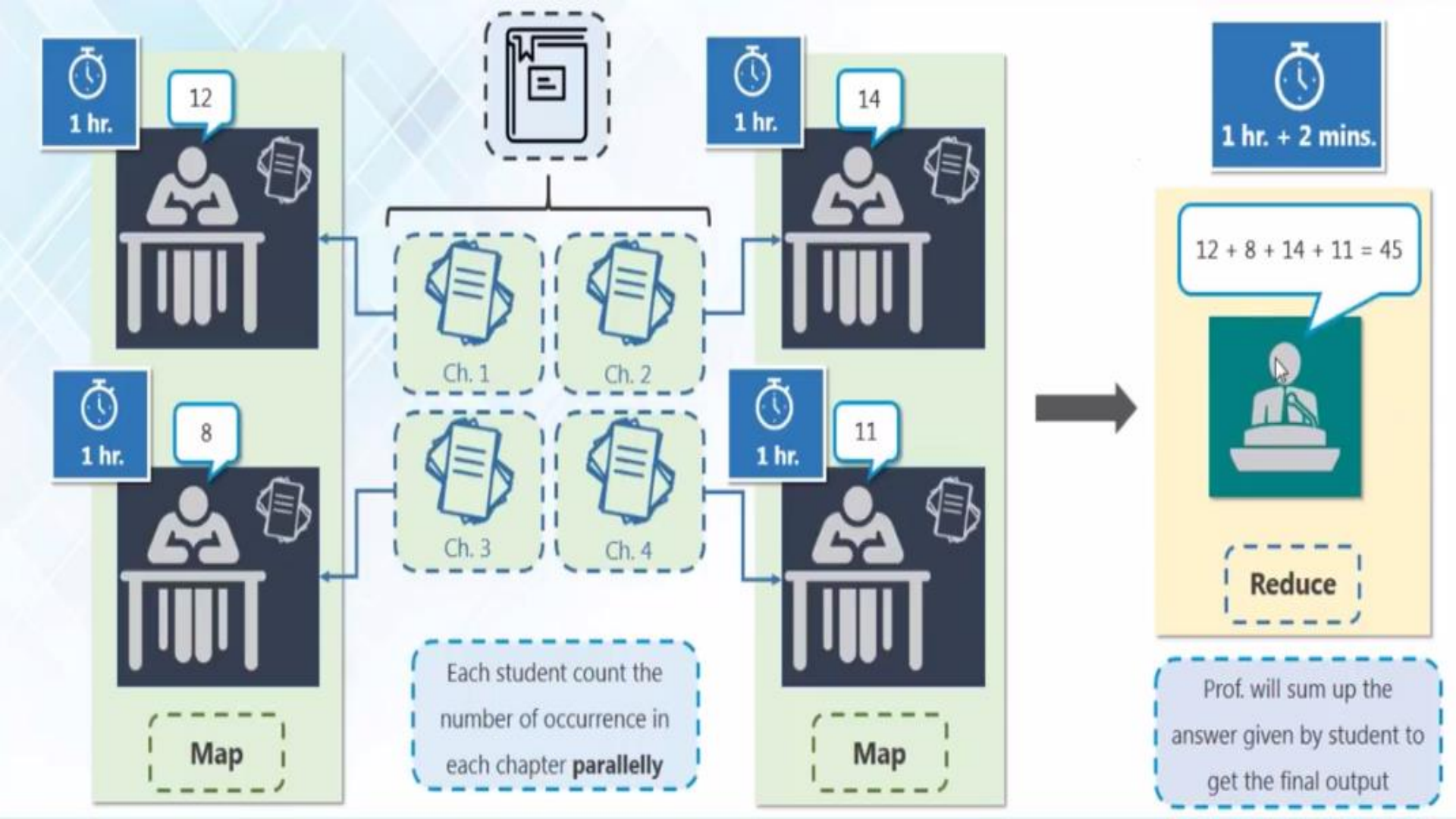
Processing:
Allows parallel & distributed processing

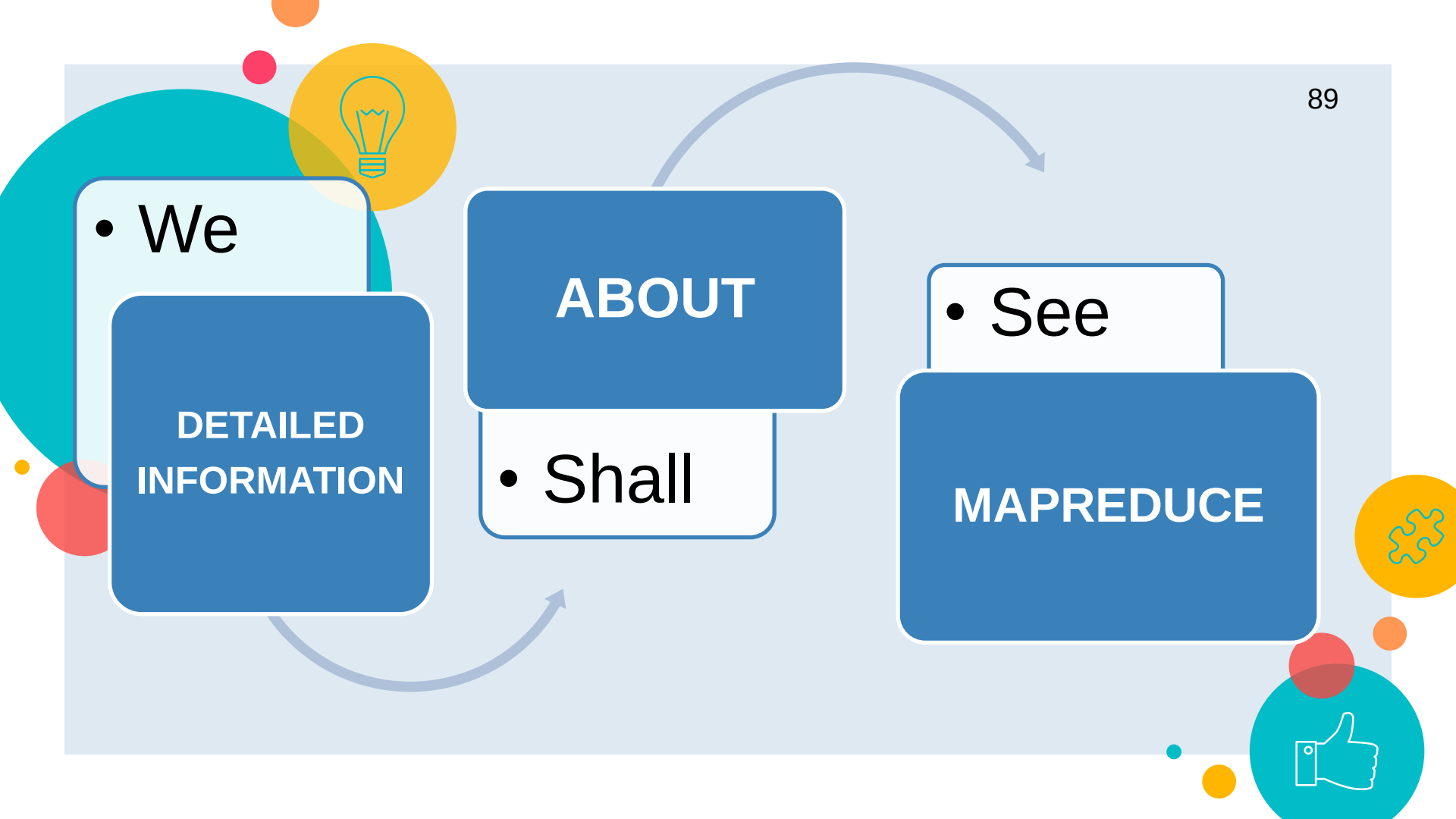

• We
STORY
OF
• Shall
• See
MAPREDUCE


45
46
45
45
Time: 4 Hours
Majority of the students have answered 45
Time: 4 Hours
Each student has to count the occurrence of the word Julius in the book

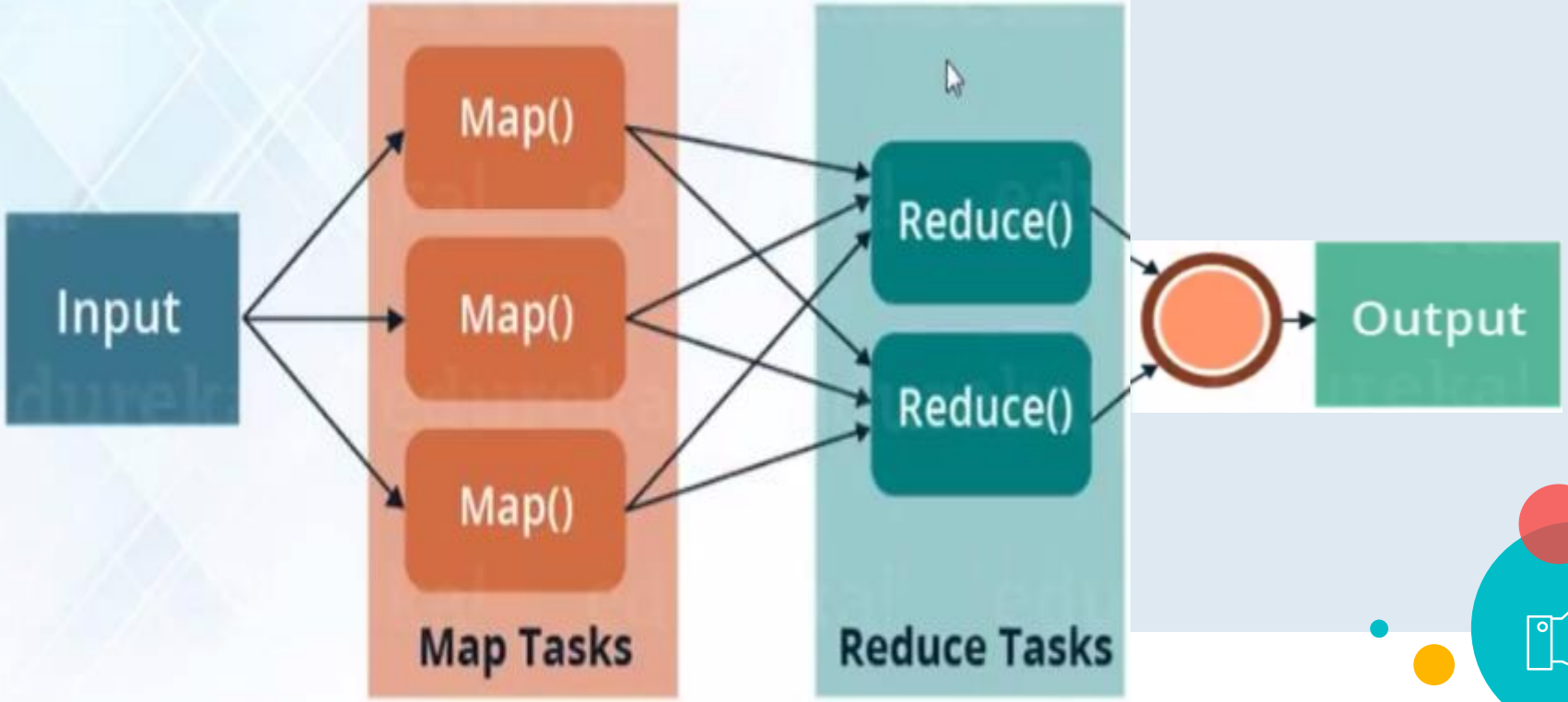

1 hr.
12
1 hr.
14
Ch. 1
Ch. 2
1 hr.
8
Ch. 3
Ch. 4
1 hr.
11
Each student count the number of occurrence in each chapter **parallelly**
Map
Map


1 hr. + 2 mins.
12 + 8 + 14 + 11 = 45
Reduce
Prof. will sum up the answer given by student to get the final output


1 hr.
12
1 hr.
14
1 hr. + 2 mins.
Ch. 1
Ch. 2
12 + 8 + 14 + 11 = 45
1 hr.
8
Ch. 3
Ch. 4
1 hr.
11
Reduce
Each student count the number of occurrence in each chapter **parallelly**
Map
Map
Prof. will sum up the answer given by student to get the final output


• We
DETAILED INFORMATION
ABOUT
• Shall
• See
MAPREDUCE

MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment

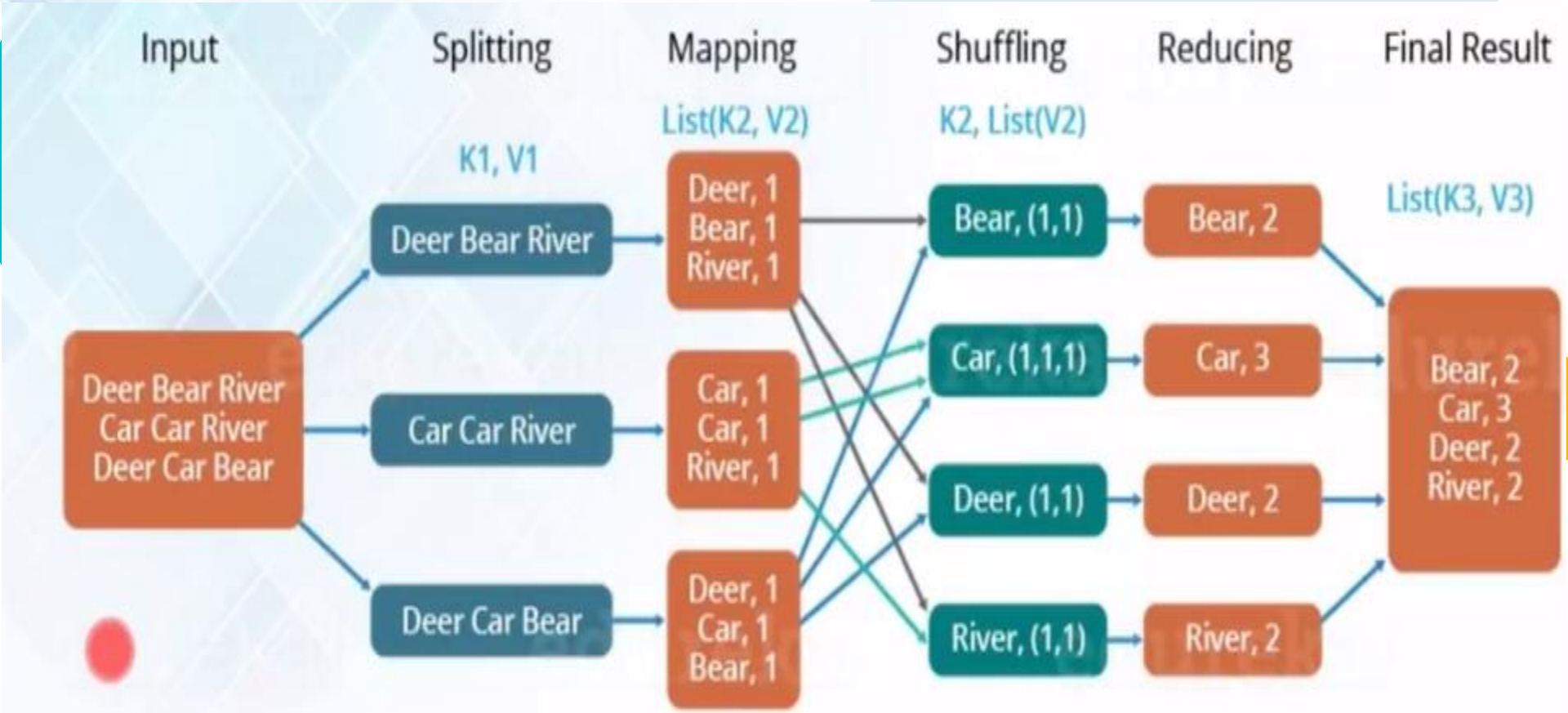


MAP

Reduce

WORD Count Program

The Overall MapReduce Word Count Process



Three Major Parts of MapReduce Program:

1

Mapper Code:

You write the mapper logic over here i.e. how map task will process the data to produce the key-value pair to be aggregated

2

Reducer Code:

You write reducer logic here which combines the intermediate key-value pair generated by Mapper to give the final aggregated output

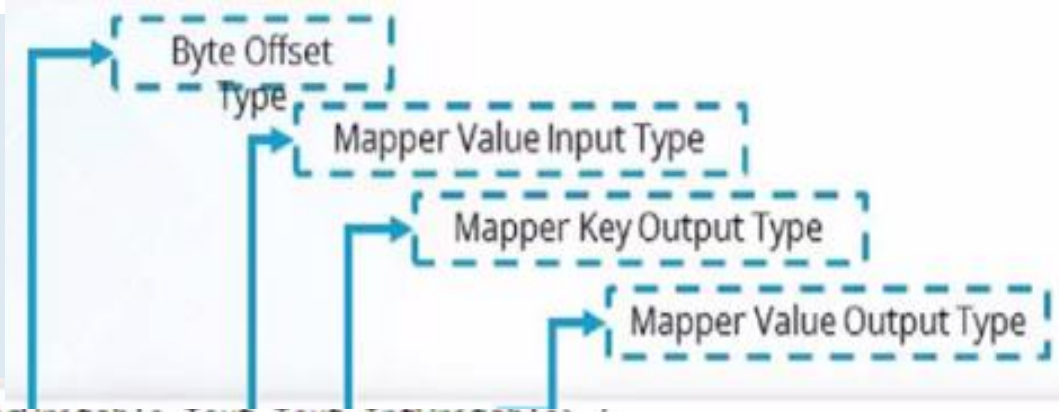
3

Driver Code

You specify all the job configurations over here like job name, Input path, output path, etc.

Input Text File

Key	Value
0	Dear Bear River
121	Car Car River
226	Deer Car Bear




```
public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> {

public void map(LongWritable key, Text value, Context context) throws IOException,InterruptedException {

String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
value.set(tokenizer.nextToken());
context.write(value, new IntWritable(1));
}
```

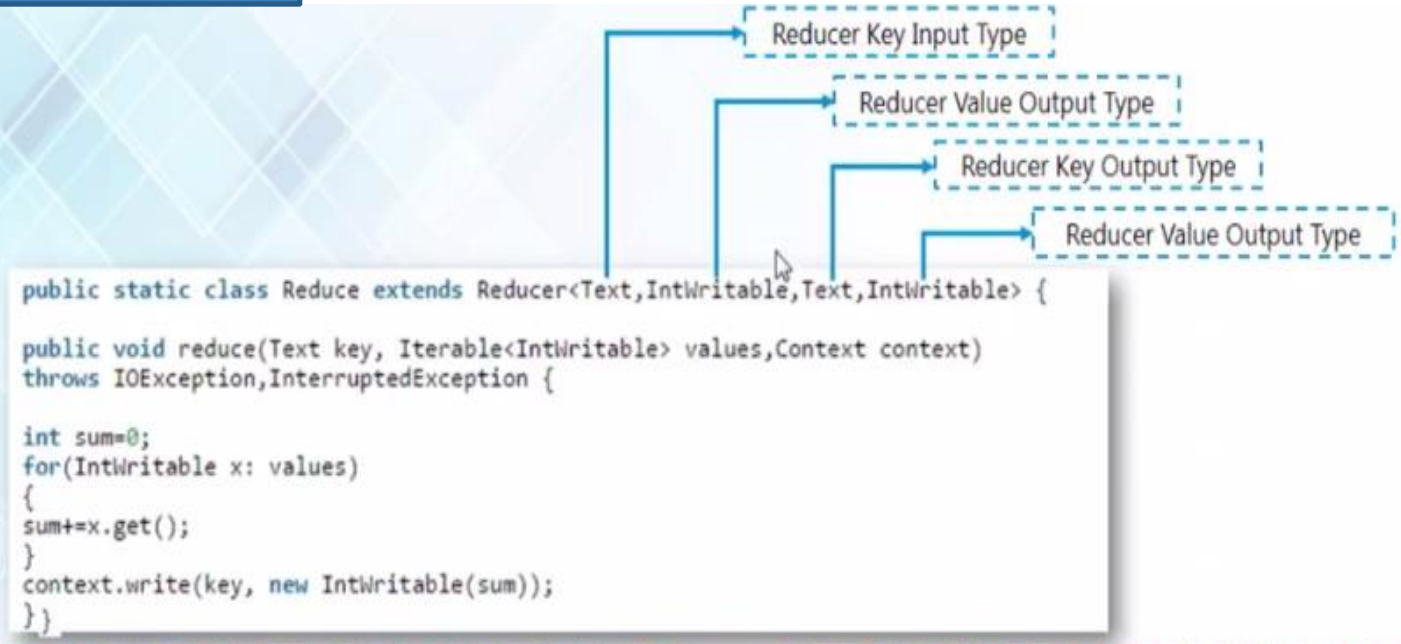
Mapper Input:

- The key is nothing but the offset of each line in the text file: *LongWritable*
- The value is each individual: *Text*

Mapper Output:

- The key is the tokenized words: Text
- We have the hardcoded value in our case which is 1: IntWritable
- Example – Dear 1, Bear 1, etc.

REDUCER CLASS



Reducer Input:

- Keys are unique words which have been generated after the sorting and shuffling phase: Text
- The value is a list of integers corresponding to each key: IntWritable
- Example: Bear, [1, 1], etc.

Reducer Output:

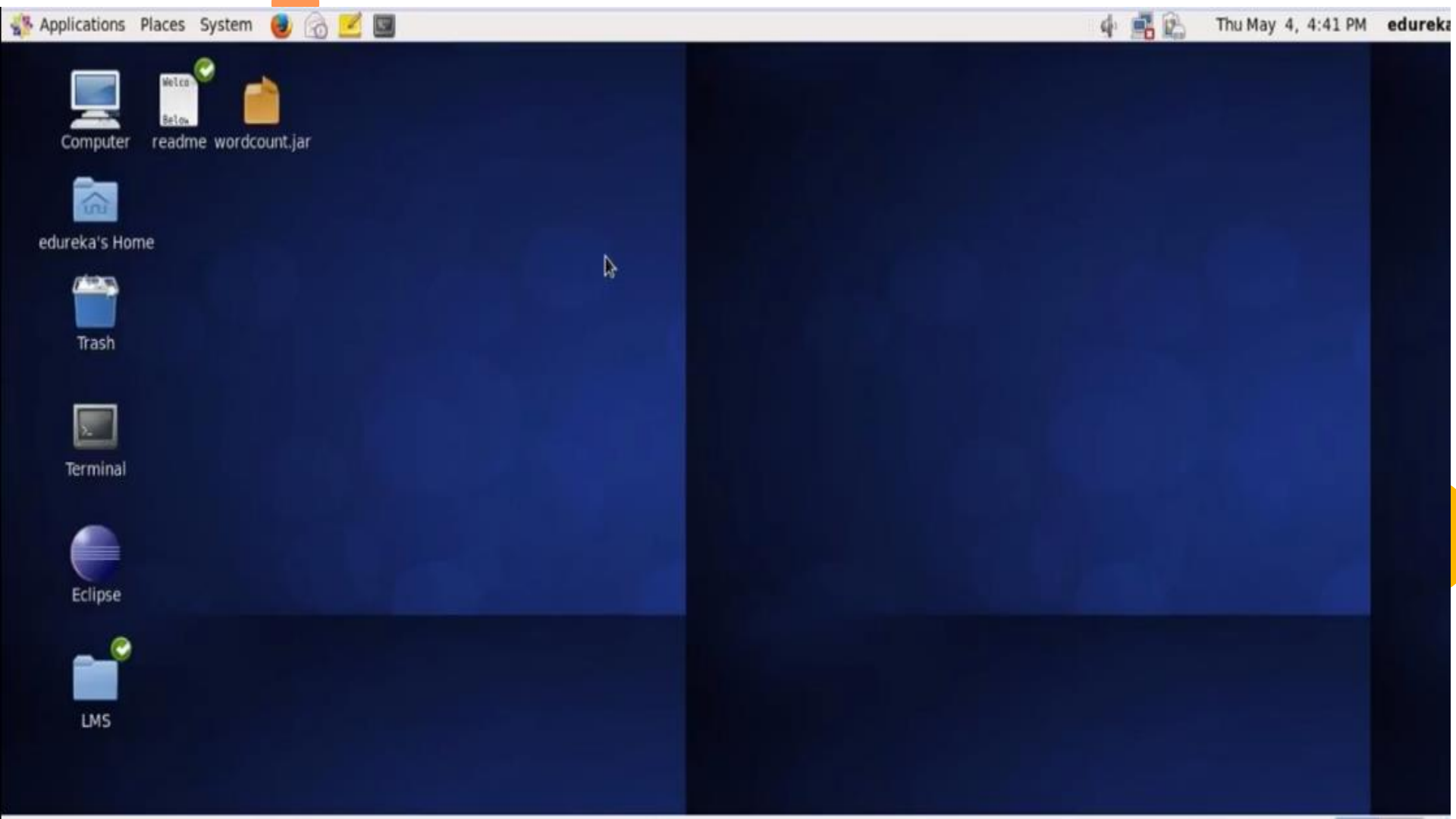
- The key is all the unique words present in the input text file: Text
- The value is the number of occurrences of each of the unique words: IntWritable
- Example: Bear, 2; Car, 3, etc. .

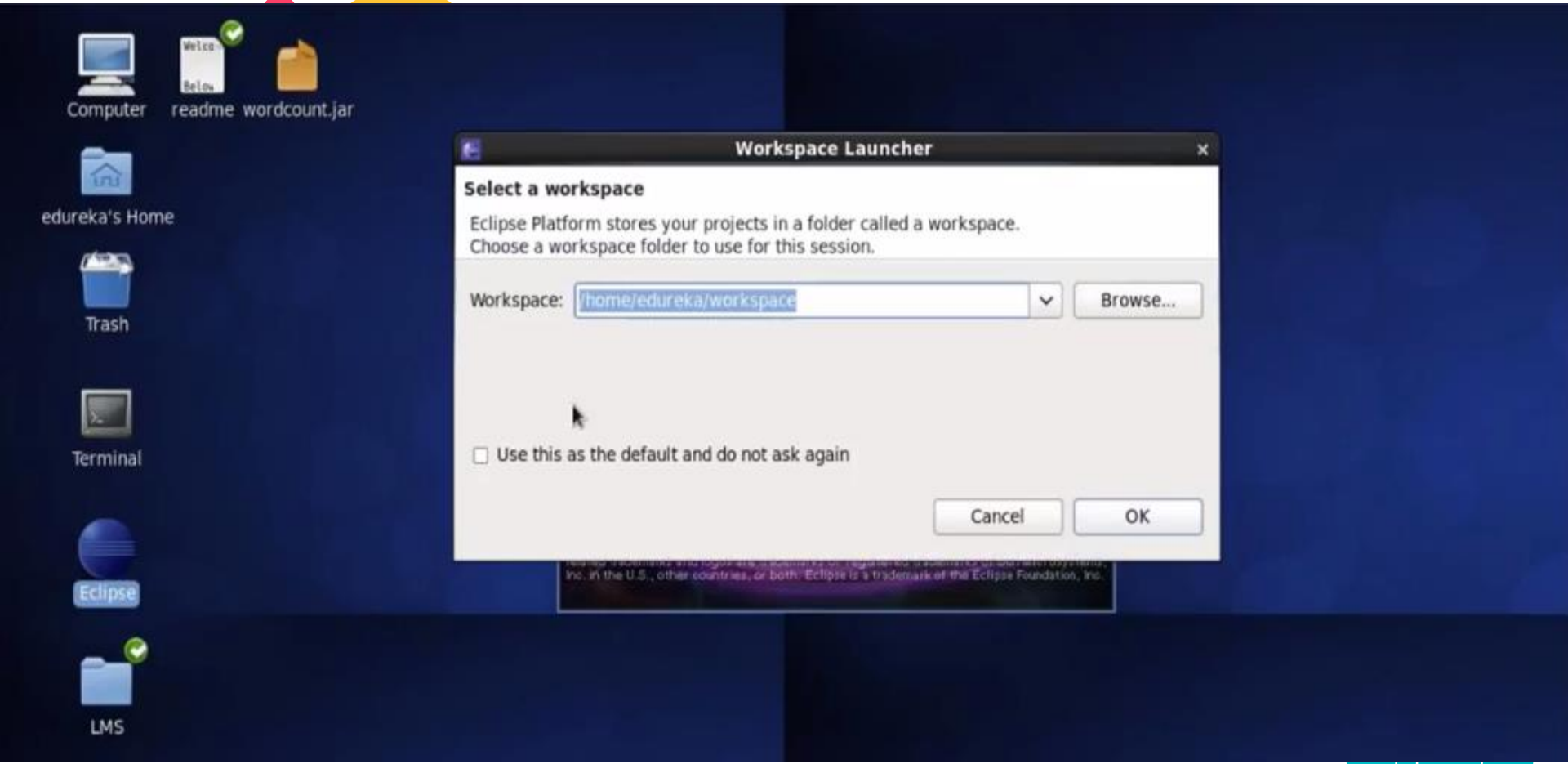
Driver Code

In the driver class, we set the configuration of our MapReduce job to run in Hadoop

```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

- Specify the name of the job , the data type of input/output of the mapper and reducer
- Specify the names of the mapper and reducer classes.
- Path of the input and output folder
- The method setInputFormatClass () is used for specifying the unit of work for mapper
- Main() method is the entry point for the driver


Applications
Places
System
Thu May 4, 4:41 PM
edureka
Computer
readme
wordcount.jar
edureka's Home
Trash
Terminal
Eclipse
LMS


Computer
readme
wordcount.jar
edureka's Home
Trash
Terminal
Eclipse
LMS
Workspace Launcher
Select a workspace
Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.
Workspace:
/home/edureka/workspace
Browse...
Use this as the default and do not ask again
Cancel
OK

File Edit Source Refactor Navigate Search Project Run Window Help

Resource

Project Explorer

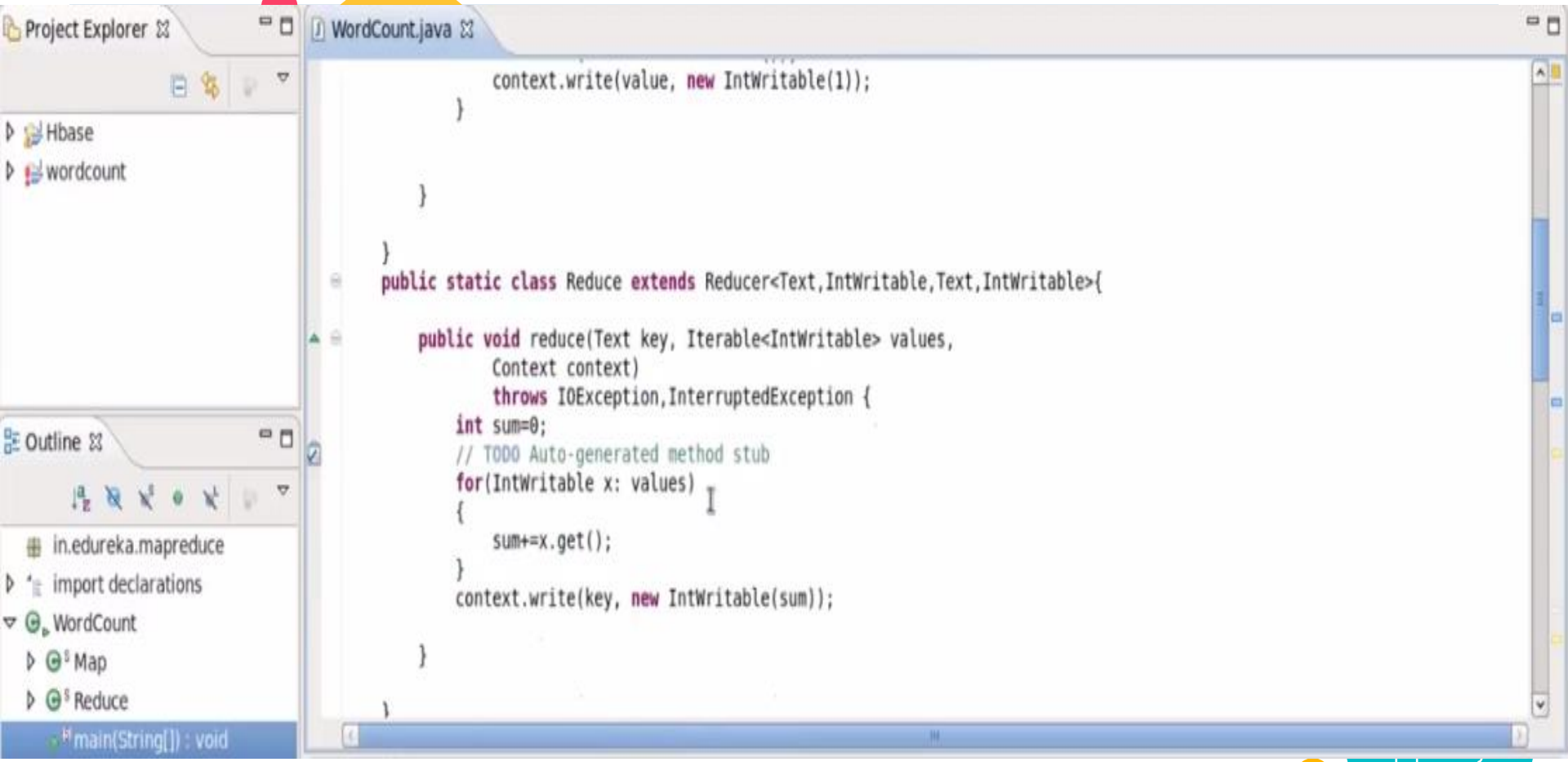
- Hbase
- wordcount

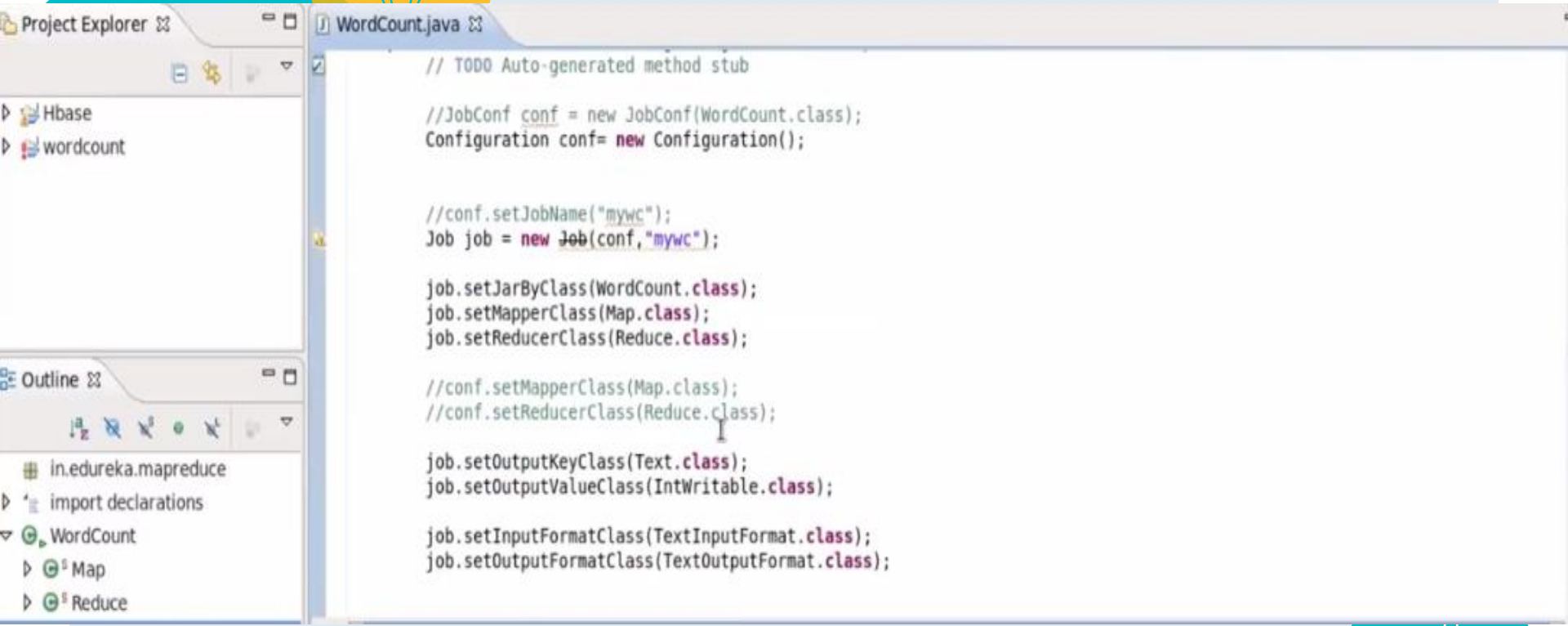
Outline

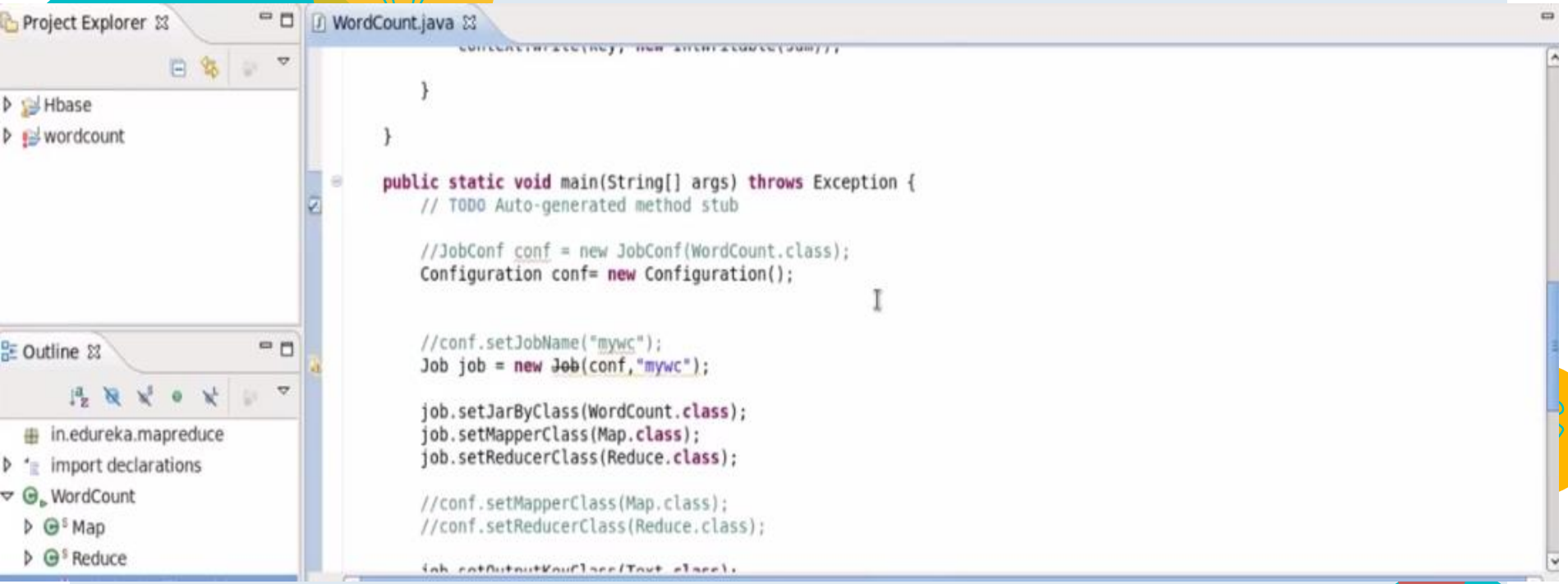
- in.edureka.mapreduce
- import declarations
- WordCount
  - Map
  - Reduce
  - main(String[]) : void

WordCount.java

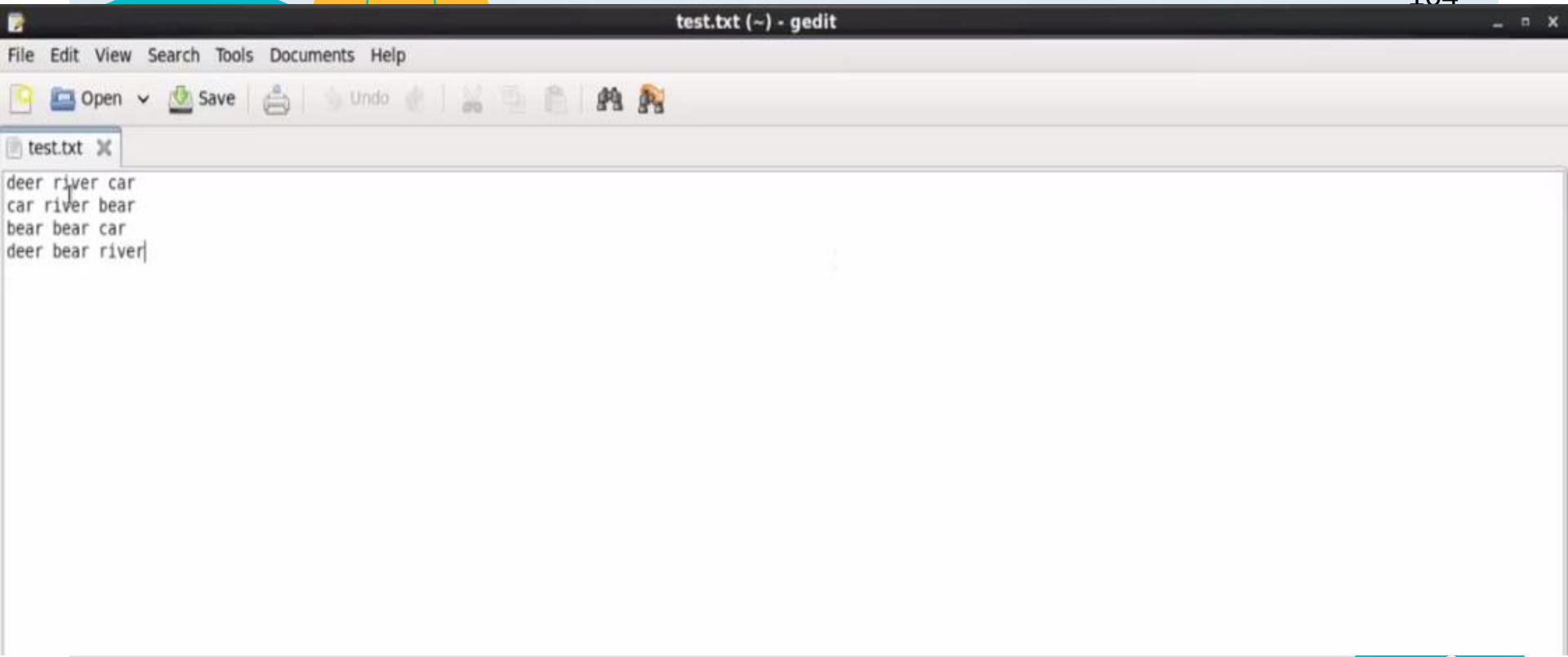
```
import java.io.IOException;

public class WordCount {

    public static class Map extends Mapper<LongWritable,Text,Text,IntWritable>{

        public void map(LongWritable key, Text value,
                Context context)
                throws IOException,InterruptedException {

            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens()) {
                value.set(tokenizer.nextToken());
                context.write(value, new IntWritable(1));
            }
```


Project Explorer
Hbase
wordcount
Outline
in.edureka.mapreduce
import declarations
WordCount
Map
Reduce
main(String[]) : void
WordCount.java
context.write(value, new IntWritable(1));
}
}
public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable>{
public void reduce(Text key, Iterable<IntWritable> values,
Context context)
throws IOException,InterruptedException {
int sum=0;
// TODO Auto-generated method stub
for(IntWritable x: values)
{
sum+=x.get();
}
context.write(key, new IntWritable(sum));
}
}


Project Explorer
Hbase
wordcount
Outline
in.edureka.mapreduce
import declarations
WordCount
Map
Reduce
WordCount.java
// TODO Auto-generated method stub
//JobConf conf = new JobConf(WordCount.class);
Configuration conf= new Configuration();
//conf.setJobName("mywc");
Job job = new Job(conf,"mywc");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
//conf.setMapperClass(Map.class);
//conf.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);


Project Explorer
Hbase
wordcount
Outline
in.edureka.mapreduce
import declarations
WordCount
Map
Reduce
WordCount.java
}
}
public static void main(String[] args) throws Exception {
// TODO Auto-generated method stub
//JobConf conf = new JobConf(WordCount.class);
Configuration conf= new Configuration();
//conf.setJobName("mywc");
Job job = new Job(conf,"mywc");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
//conf.setMapperClass(Map.class);
//conf.setReducerClass(Reduce.class);

```
[edureka@localhost ~]$ hadoop fs -mkdir /wordcount
17/05/04 16:44:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
[edureka@localhost ~]$ hadoop fs -mkdir /wordcount/input
17/05/04 16:45:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
[edureka@localhost ~]$ hadoop fs -mkdir /wordcount/output
17/05/04 16:46:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
[edureka@localhost ~]$
```


test.txt (~) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
test.txt
deer river car
car river bear
bear bear car
deer bear river

```
[edureka@localhost ~]$ hadoop fs -put test.txt /wordcount/input
17/05/04 16:47:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
[edureka@localhost ~]$ hadoop jar wordcount.jar in.edureka.mapreduce/WordCount /wordcount/input /wordcount/outpu
t
17/05/04 16:52:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
17/05/04 16:52:26 INFO Configuration.deprecation: mapreduce.map.class is deprecated. Instead, use mapreduce.job.
map.class
17/05/04 16:52:26 INFO Configuration.deprecation: mapred.job.name is deprecated. Instead, use mapreduce.job.name
17/05/04 16:52:26 INFO Configuration.deprecation: mapreduce.reduce.class is deprecated. Instead, use mapreduce.j
ob.reduce.class
17/05/04 16:52:26 INFO Configuration.deprecation: mapreduce.inputformat.class is deprecated. Instead, use mapred
uce.job.inputformat.class
17/05/04 16:52:26 INFO Configuration.deprecation: mapred.input.dir is deprecated. Instead, use mapreduce.input.f
ileinputformat.inputdir
17/05/04 16:52:26 INFO Configuration.deprecation: mapred.output.dir is deprecated. Instead, use mapreduce.output
.fileoutputformat.outputdir
17/05/04 16:52:26 INFO Configuration.deprecation: mapreduce.outputformat.class is deprecated. Instead, use mapre
duce.job.outputformat.class
17/05/04 16:52:26 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.map
s
17/05/04 16:52:26 INFO Configuration.deprecation: mapred.output.key.class is deprecated. Instead, use mapreduce.
job.output.key.class
17/05/04 16:52:26 INFO Configuration.deprecation: mapred.working.dir is deprecated. Instead, use mapreduce.job.w
orking.dir
17/05/04 16:52:26 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1493892127344_0001
17/05/04 16:52:26 INFO impl.YarnClientImpl: Submitted application application_1493892127344_0001 to ResourceMana
ger at /0.0.0.0:8032
```

File Edit View Search Terminal Help

```
        Reduce shuffle bytes=138
        Reduce input records=12
        Reduce output records=4
        Spilled Records=24
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=158
        CPU time spent (ms)=850
        Physical memory (bytes) snapshot=213401600
        Virtual memory (bytes) snapshot=721362944
        Total committed heap usage (bytes)=137433088
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=60
    File Output Format Counters
        Bytes Written=28
```

```
File  Edit  View  Search  Terminal  Help
[edureka@localhost ~]$ hadoop fs -ls /wordcount/output
17/05/04 16:55:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Found 2 items
-rw-r--r--   3 edureka supergroup          0 2017-05-04 16:52 /wordcount/output/_SUCCESS
-rw-r--r--   3 edureka supergroup         28 2017-05-04 16:52 /wordcount/output/part-r-00000
[edureka@localhost ~]$
```

```
[edureka@localhost ~]$ hadoop fs -cat /wordcount/output/*0
17/05/04 16:57:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
iltin-java classes where applicable
bear    4
car     3
deer    2
river   3
```

What is Apache Hadoop?

Hadoop is open source Open source software framework designed for storage and processing of large scale data on clusters of commodity hardware.It works on CDH.

CDH is a bundle of all the Hadoop ecosystem tools installed on linux VM.We can use this VM to set up your cluster.Available as express & enterprise edition.

CDH : cloudera distribution for Hadoop, CDH5.8

CDH works with YARN for resource management and Cloudera Manager for Administration

Hadoop can process data that can not be processed by RDBMS. Data is stored in distributed manner

Created by Doug Cutting ,creator of Apache Lucene,text search library in 2005.

Cutting named the program after his son's toy elephant.

Distributions of Hadoop

Cloudera, Hortonworks, Pivotel, IBM, MapR, DELL, AMAZON Web services

History of Hadoop and Creators

- Hadoop was created by Doug Cutting,
- Doug is creator of a text search library called Apache Lucene.
- Root of Hadoop is Apache Nutch, an index based open source web search engine which is part of Lucene project.
- Hadoop is not very Old to have a rich History but starting from-
- 2003 - Google launches project Nutch to handle billions of searches and indexing millions of web pages.
- Oct 2003 - Google releases papers with GFS (Google File System)
- Dec 2004 - Google releases papers with MapReduce
- 2005 - Nutch used GFS and MapReduce to perform operations
- 2006 - Yahoo! created Hadoop based on GFS and MapReduce (with Doug Cutting and team)
- 2007 - Yahoo started using Hadoop on a 1000 node cluster
- Jan 2008 - Apache took over Hadoop
- Jul 2008 - Tested a 4000 node cluster with Hadoop successfully
- 2009 - Hadoop successfully sorted a petabyte of data in less than 17 hours to handle billions of searches and indexing millions of web pages.
- Dec 2011 - Hadoop releases version 1.0

Importance of Big Data

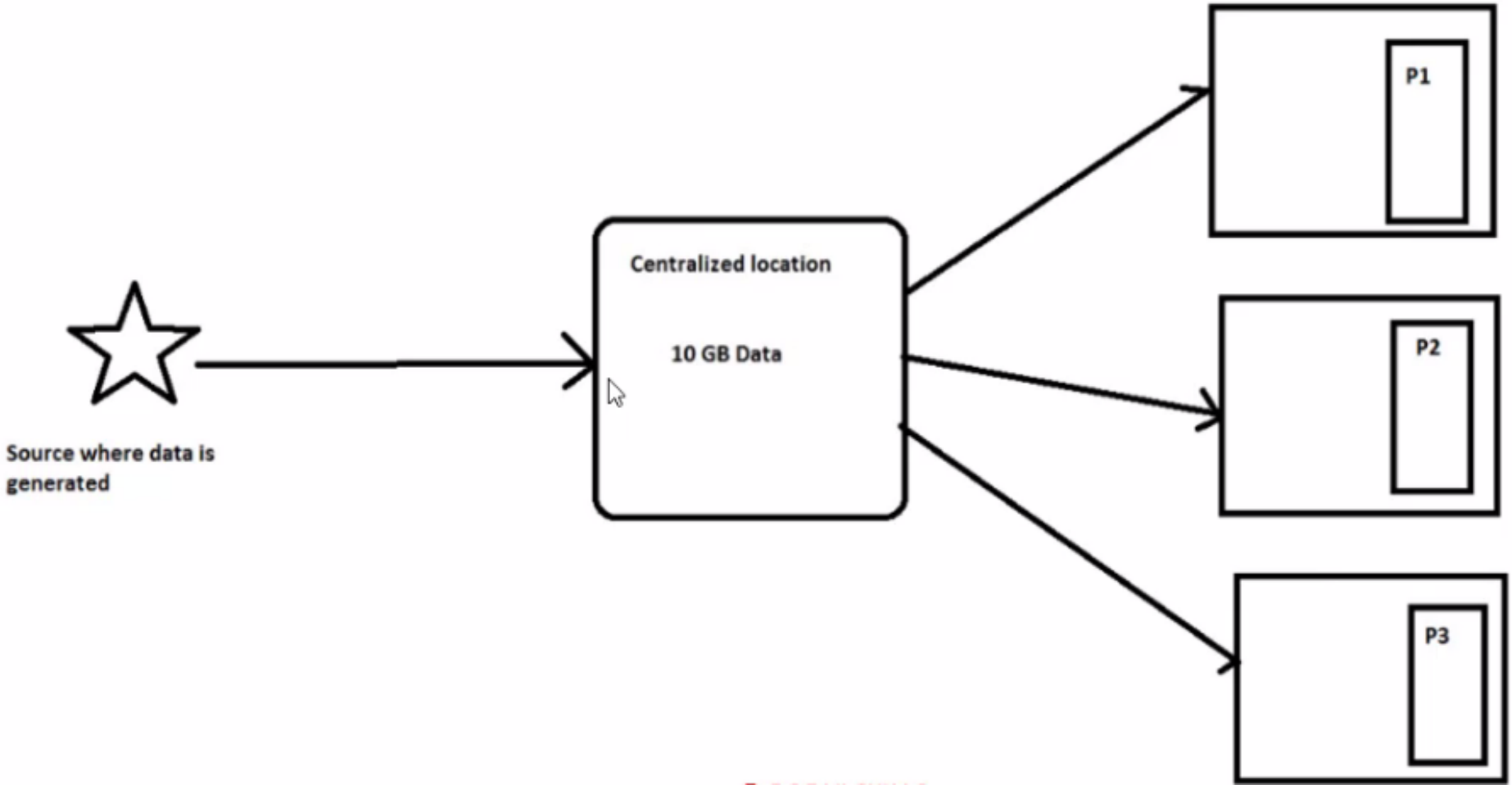
Big Data is important to solve issues like

- Determining root causes of failures, issues and defects in near-real time.
- Generating coupons at the point of sale based on the customer's buying habits.
- Recalculating entire risk portfolios in minutes.
- Detecting fraudulent behavior before it affects your organization.

Limitations of Existing Distributed System

- 1. Partial Failures.- DS data will be copied at the time of processing the same.
- so it will require many network operations. if one node is failed we can not process the data available on failed node.
- 2. Finite Bandwidth to copy data to Processor.Network associated problems
- 3. progamming complexity
- 4. Storage :Distributed system where we store data in a centeralized location
- and distribute when we want to process.
- To avoid above problems we worked to develop Hadoop

Why Hadoop ?

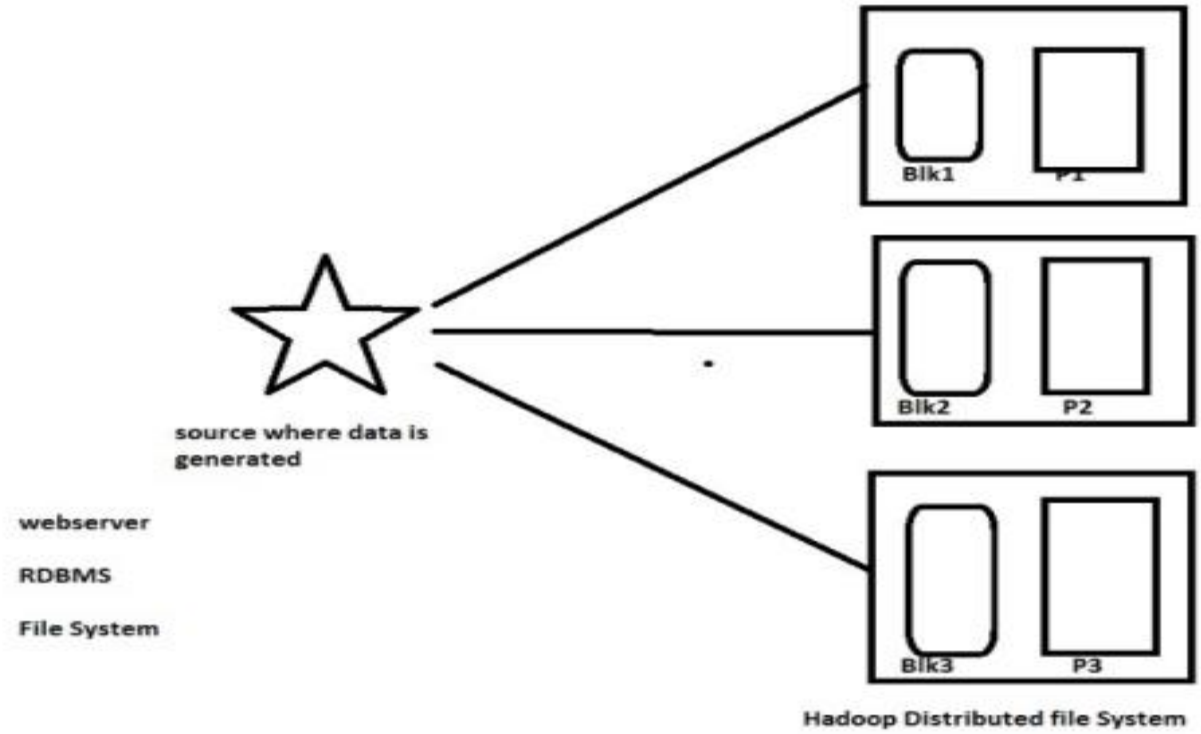

Centralized location
10 GB Data
Source where data is generated
P1
P2
P3

How Hadoop solves problems / limitations of Distributed System.

- to overcome all these issues Hadoop and its eco system was developed by Doug Cutting and his team. major development of Early Hadoop version was took place in Yahoo.
- Hadoop is heavily inspired by Google's white papers like GFS (Google File System) and MapReduce.
- we can see Hadoop as collection of Apace Nutch and Apache Lucene.
- 1. Instead of distributing Data at the time of Processing, Hadoop Distributes Data at the time of STORAGE. it is something like we are giving program to the data instead of giving data to the program

- How Data is stored in Hadoop.
- Hadoop stores data in HDFS(Hadoop Distributed File System).
- HDFS is software written in Java. this will be installed on your operating system.
- once HDFS is installed it will create a new File System on top of existing Linux file system (ext3 or ext4)

DS2


Blk1
P1
Blk2
P2
Blk3
P3
source where data is generated
webserver
RDBMS
File System
Hadoop Distributed file System

why not DS but Hadoop

- using Hadoop we can solve following issues that were there with DS
- 1. Partial Failures / Hadoop is fault tolerant
- By Replicating the data to multiple nodes
- data will be stored in multiple blocks
- 2. Finite Bandwidth
- we will work with single network that will not have any data flow at processing time.
- 3. programming complexity
- all the nodes are working on same environment
- 4. Storage :no centralized storage in Hadoop , your data will be distributed as soon as you are storing that.
- in Hadoop we are storing data with the Processors unlike Distributed system where we store data in a centralized location and distribute when we want to process.

Motivations & Need of Hadoop

- Hadoop is Open source so its free
- Hadoop does not require specilized hardware, it can work with commodity hardware.
- data is stored in distributed manner
- data is replicated to improve high availability
- data is processed on multiple machine in parallel
- can work with multiple OS
- we can manage own users
- Hadoop can process variety of data.
- Hadoop can be used for streaming data or real time data.
- Hadoop is pure processing & file based.

Big data Vs Hadoop

- Big Data and Hadoop Both are different things..
- Big data is a large set of the data that is generated through the FACEBOOK,U TUBE, and many social sites. That is in the form of video,image,text,graphics, sensors, and many other things. Big data has divided in to 3 types Structured, Semi Structured, Unstructured data.
- If data is large enough and complex than it becomes difficult to process using a single computer.
- Hadoop is a tools that manage this kind of massive data and most of the big company used this tools like a oracle, IBM, Google etc. Hadoop is a product of apache . Hadoop is used for storage and large-scale processing of data-sets on clusters of commodity hardware. HADOOP clusters can easily be scaled by adding additional cluster nodes.
- It includes various main components, including a MapReduce and a Hadoop distributed file system (HDFS).

Difference between Hadoop and RDBMS

- 1. RDBMS is transactional that supports insert, update, delete and retrival of data, Hadoop is a storage that does not support transactions.
- 2. RDBMS is for real time data. Hadoop is for archived data
- 3. RDBMS can only work with Structured Data, Hadoop can work with Structured, Semi Structured and Unstructured Data
- 4. RDBMS is dependent on single processor that can have maximum of the processing speed availble. Hadoop works with multiple processors where we can divide our data into blocks
- 5. RDBMS works with specific hardware, Hadoop works with commodity hardware
- 6. if you want to scale RDBMS you have purchase a new server, to scale Hadoop we need new commodity hardware only

- How Big Data is Different from RDBMS Data

- RDBMS is for structured data where as Big Data is collection of structured, semi structured, and unstructured data.

- RDBMS supports transactions, while Big Data transactions are partially supported.

- RDBMS size is in TB, Big data can grow upto PB or more

- What are the different units to measure data?

- Bit - Byte - Kilo Byte - MB - GB - TB - PB - EXA BYTE - ZetaByte - YottaByte - BronotoByte
- current world data is in Exa Bytes (26 EB)

What is HDFS

HDFS (Hadoop distributed File System)

-HDFS is software written in Java. this will be installed on your operating system. once HDFS is installed it will create a new File System on top of existing Linux file system (ext3 or ext4).

HDFS has some deamons. every Deamon is a Java program that runs in saperate JRE.

Hadoop stores data in HDFS(Hadoop Distributed File System).

It is used for streaming data. is write once : means no random reads and write are allowed like RDBMS

HDFS will be collection of Nodes. These Nodes will be running on clusters of *commodity hardware. It is not expensive. every machine is known as Node.

HDFS is scalable. more nodes can be added automatically

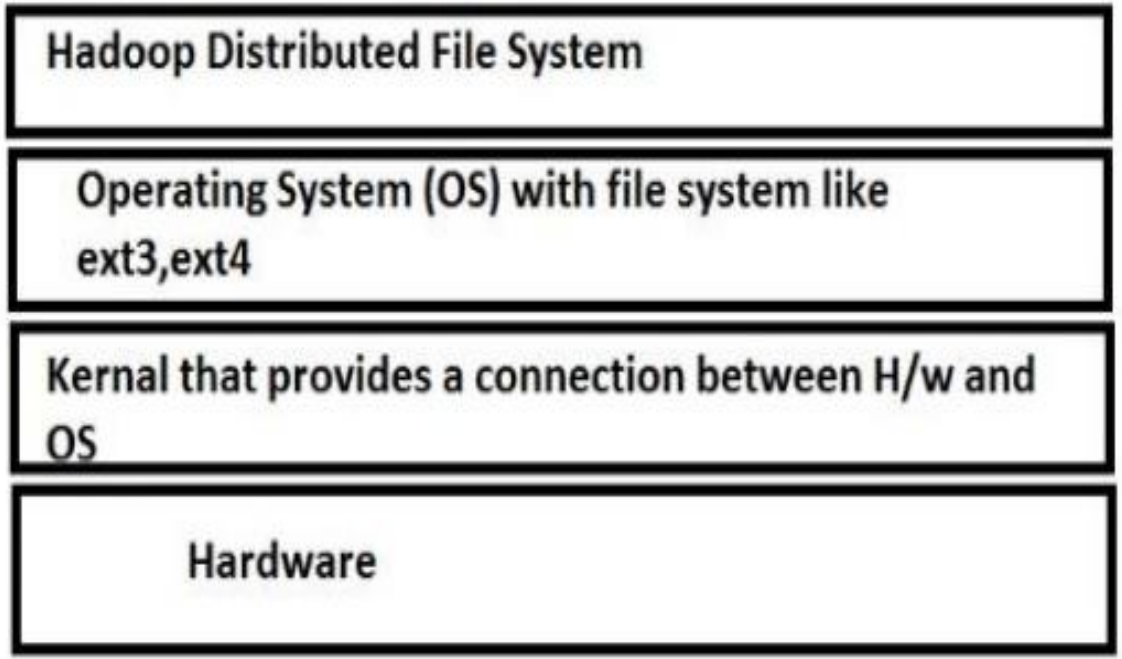
HDFS is fault tolerant .i.e each block of file will be replicated on your cluster by default 3 times which is configurable.

Hadoop is created with 2 core components 1. HDFS 2. MapReduce

- HDFS (Hadoop Distributed File System) is a File System which is installed on Existing File System to store data for Hadoop.
- HDFS remains on Local File File System as a layer where we can keep data with different block size (default 128 MB) and replication (3 Blocks)

HDFS : Classic Mode

HDFS Layer



OS will behave like kernal for HDFS

HDFS Architecture

- **NameNode (NN)—**
- There will be only one NN in your cluster.
- NN is master daemon.
- All other daemons of HDFS are going to report NN.
- NN is responsible to keep meta data which contains filenames, file permission, block location on your cluster. When in used ,this information is stored in main memory of NN to allow fast access. But these information also stored in disk for persistence storage
- Name Node will keep metadata information provided by Data nodes.
- Name Node must be a high configuration machine with good amount of RAM , mostly in 100s of GB
- NN keeps meta data in 2 files
- 1.Fsimage : is a file on Hard disk of NN.
- 2. Edits Log : available in RAM

DataNode (DN)

- **there will be multiple data nodes in a cluster**
- **DN is a slave daemon.** DN is responsible to keep files.
- DN will report to NN with all the file Information. so that NN can get the Metadata info updated. DN keeps the Data Blocks
- whenever files or data is added into DN it will be replicated among the cluster, means one file will have multiple copies.
- Data nodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the name node.
- DN can be a commodity Hardware or comparatively cheaper machine.
- DN will send a heart beat and block report to NN after every 3 seconds. if heart beat is not received for 10 secs DN will be considered dead.
- Receipt of a Heartbeat implies that the Data Node is functioning properly. A Block report contains a list of all blocks on a Data Node

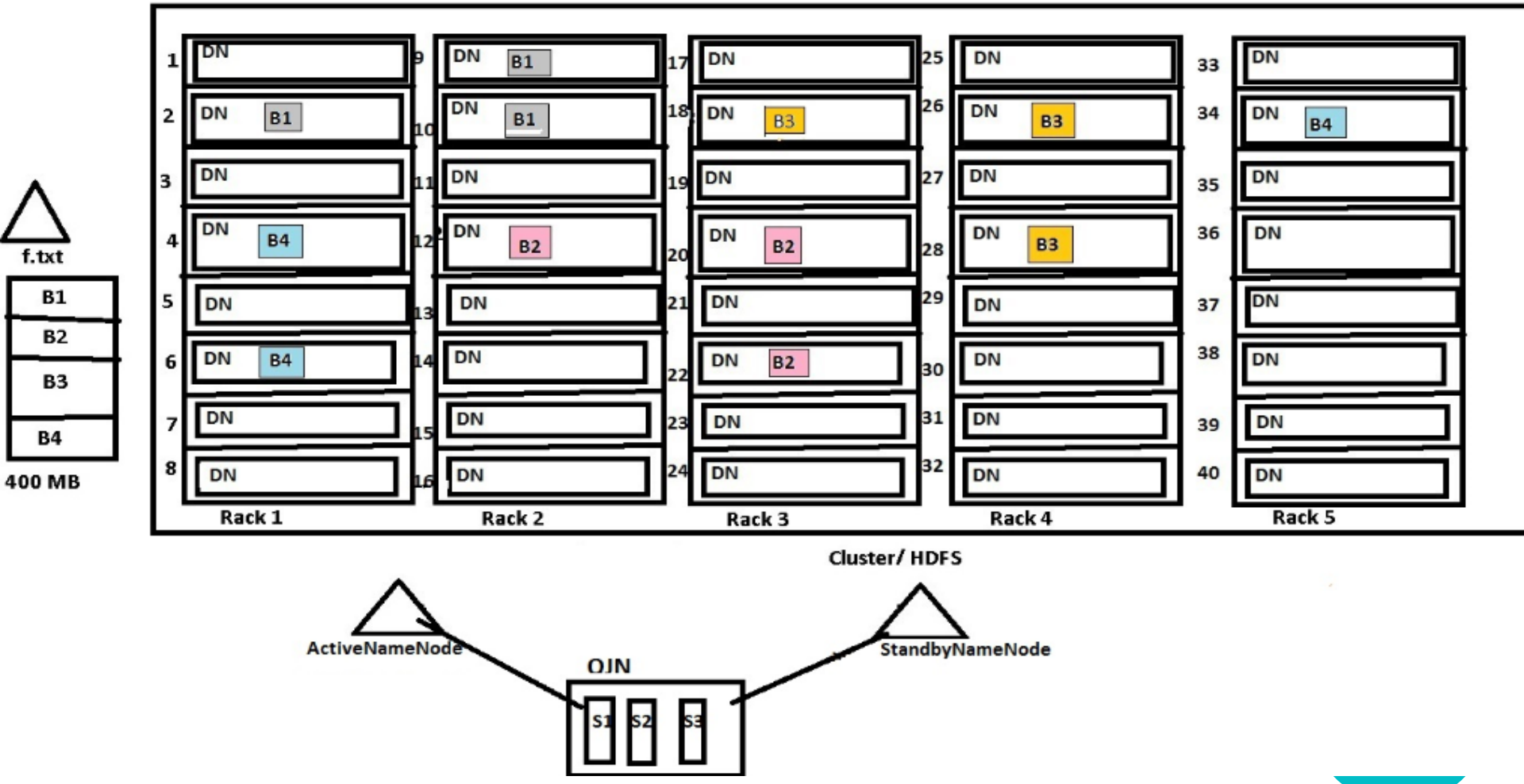
Secondary Name Node

- Secondary Name Node is like an assistant who is responsible to perform bookkeeping task that is merging FS Image with current edits log from name node
- Secondary Name Node is not a replacement for the Name Node. so if NN is failed we will not get the block information.
- Every time when your NN restart it will load current FsImage file into its memory/RAM. then will start listening to Data nodes and keep on updating the Edits Log.
- Secondary Name node does puts a checkpoint in filesystem which will help Name node to function better. Its not the backup for the Name node
- Secondary Name node main purpose is to have a checkpoint in HDFS. Its just a helper node for namenode.That's why it also known as checkpoint node .

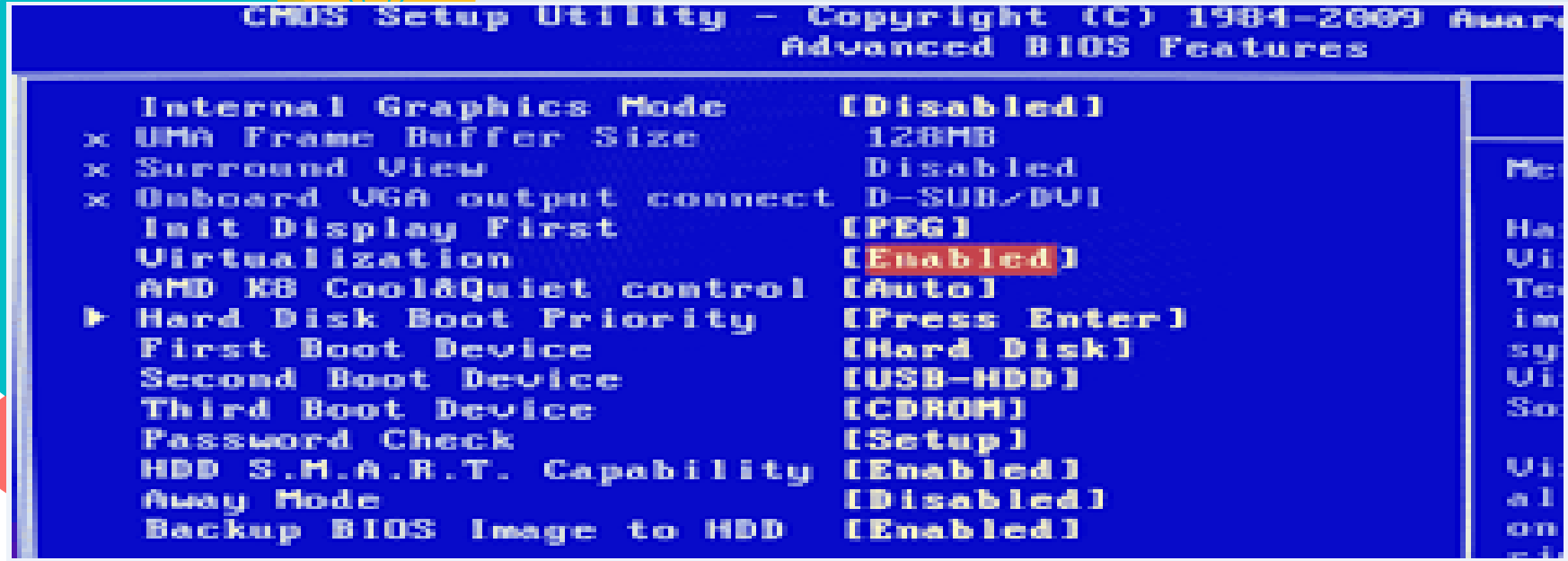
Block-

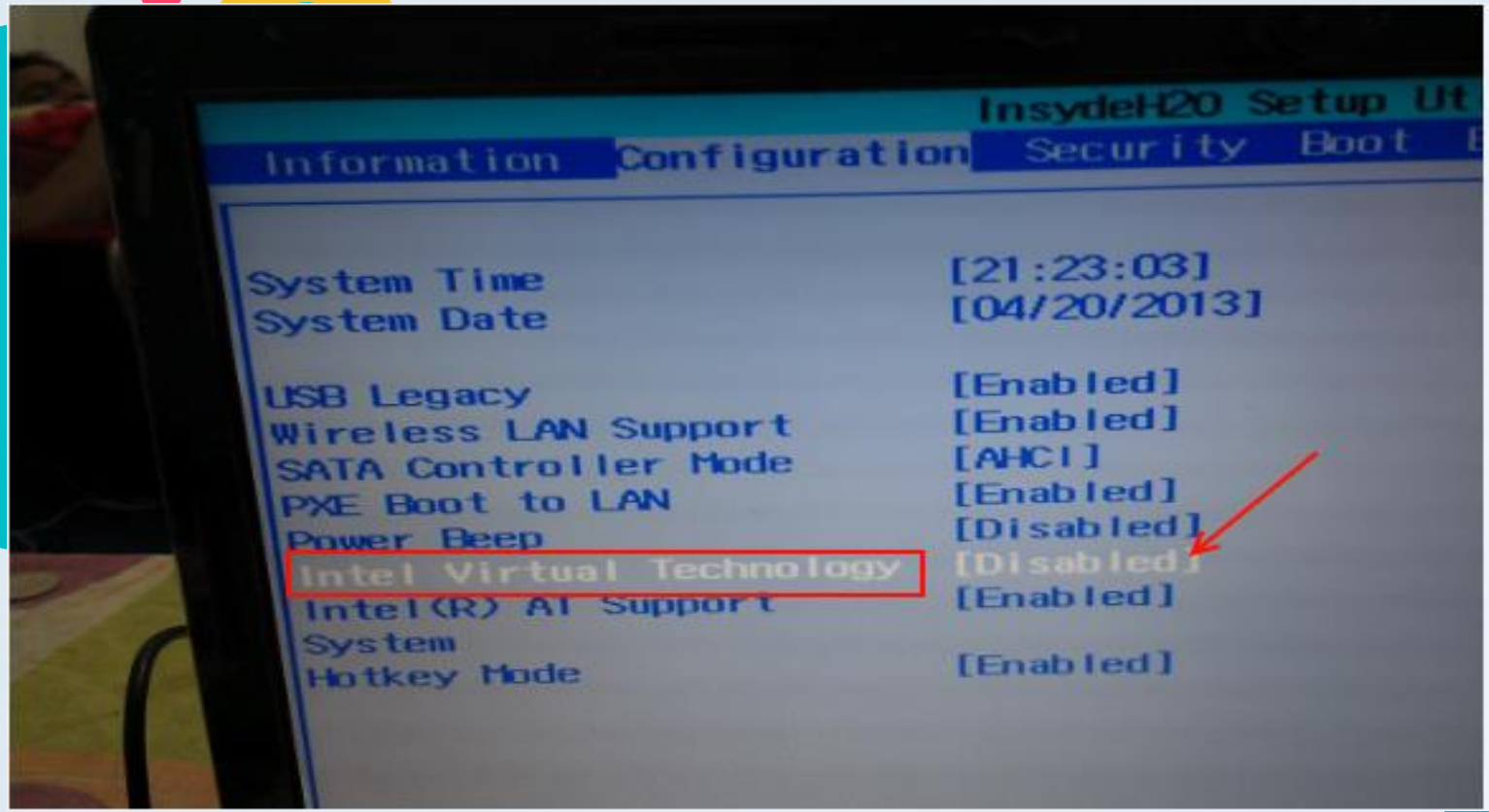
- Block is a part of file to be stored on HDFS which can be of any size & divided into multiple blocks.
- every block will be of 128 MB by default.
- for example if I have a file with 400 MB size I am going to have 4 blocks
- 3*128 MB & 1*16 MB (400 / 128 = 4 blocks = 4 seeks)
- ever block will be replicated 3 times by default.
- keeping in mind rack awareness.
- Rack Awareness means keeping one block in one rack and 2 blocks in another rack
- block size and replication factor is configurable.
- The NameNode makes all decisions regarding replication of blocks
- Why not use Linux File system to keep Blocks ?
- default block size of ext3, ext4,(Linux) NTFS and FAT32 (WINDOWS) is 4 KB. then 400 MB file will be divided in to
- 400 * 1000 / 4 = 100000 no of blocks. and here block size is not cofigurable.
- to read each block on IO is required. in order to read 400 MB I will require 100000 seeks. seek is an expensive operation

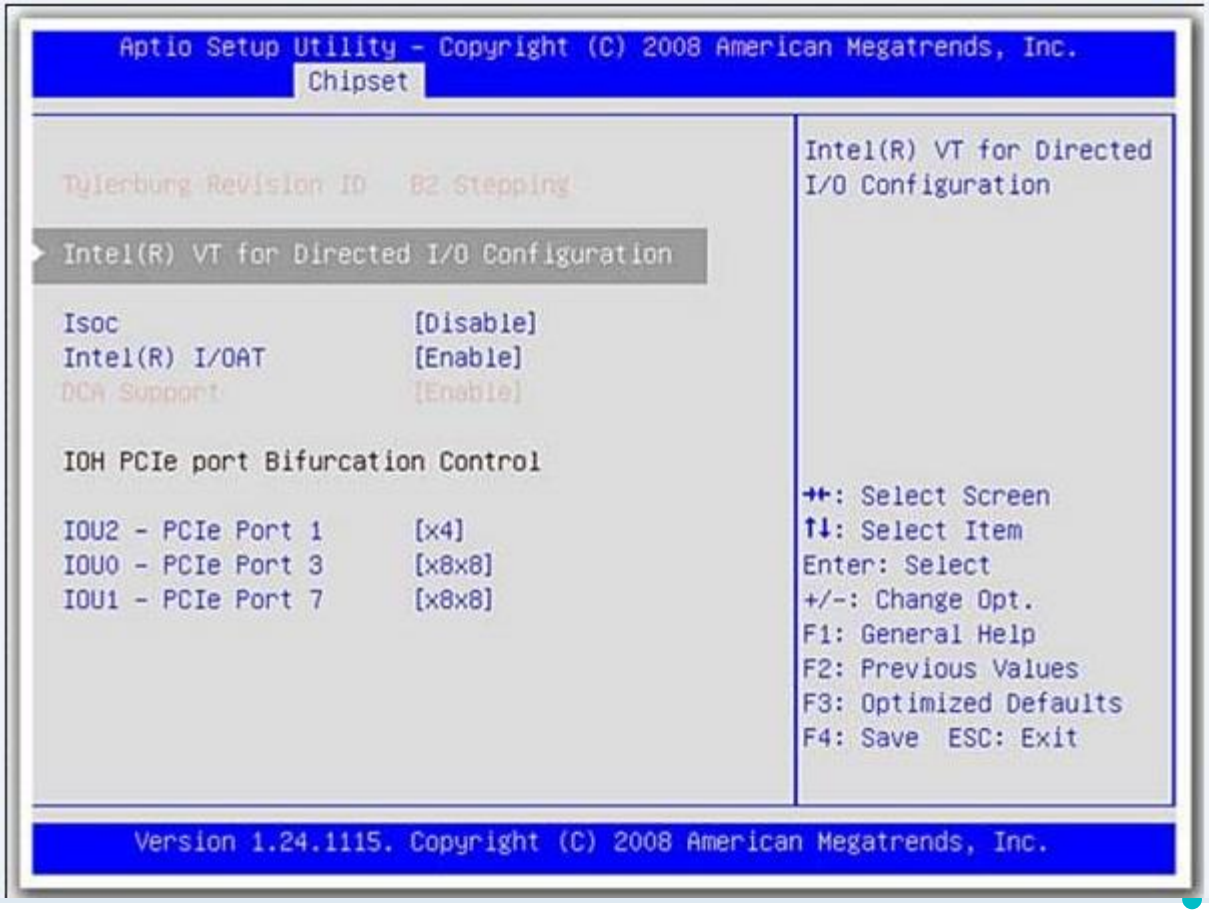
High Availability Mode

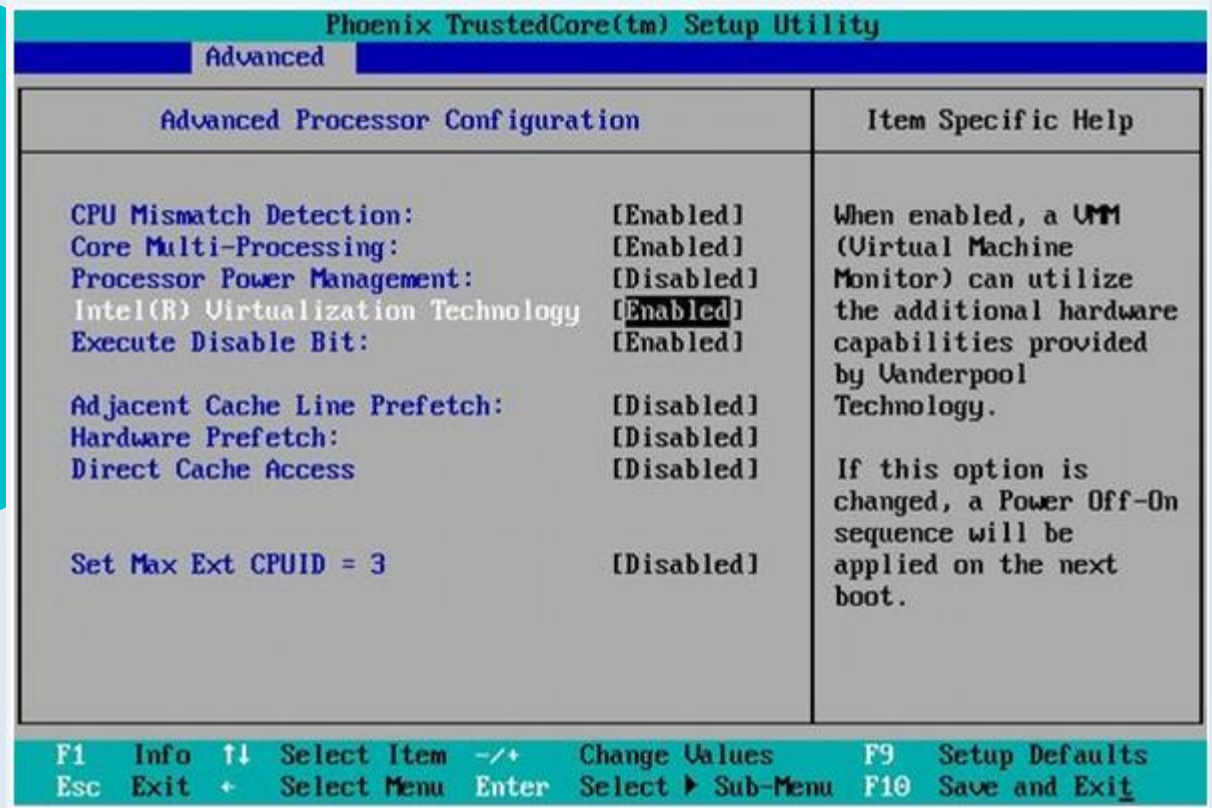


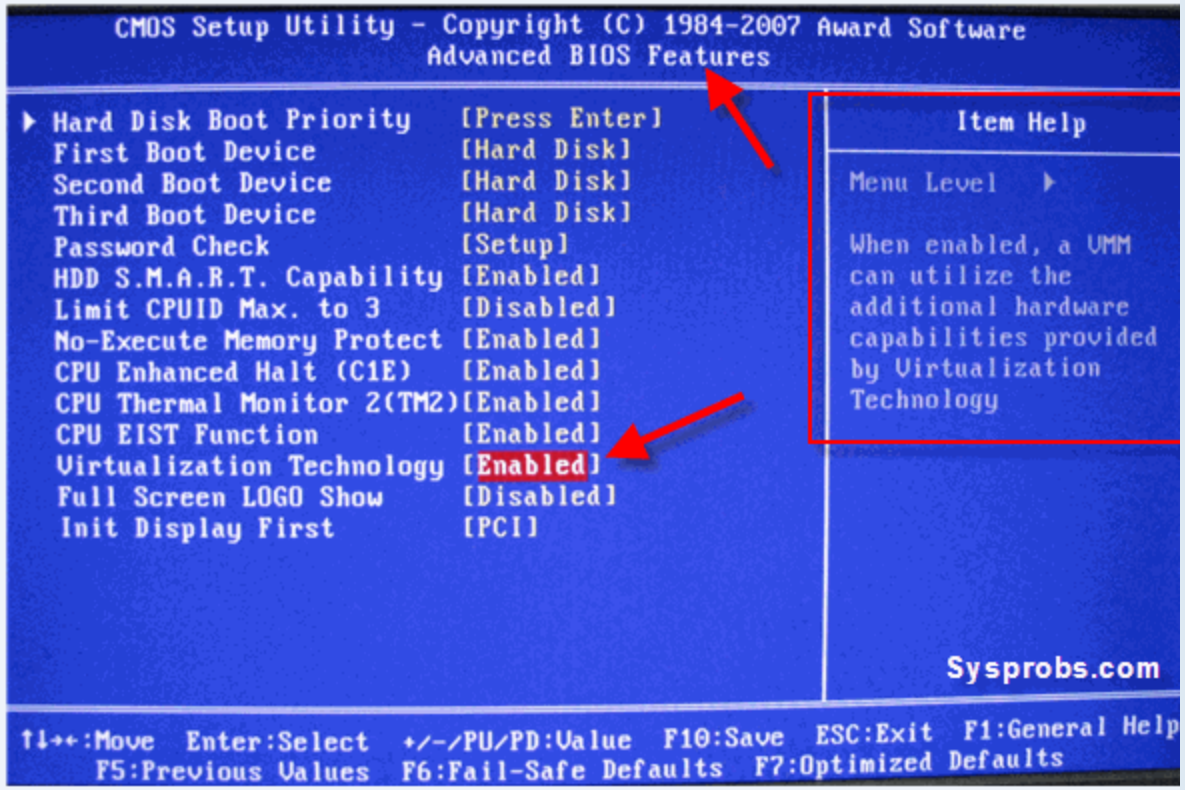

HEWLETT-PACKARD COMPUTER SETUP
Security
Power
Advanced
Setup Password
Power-On Password
Device
USB Se
Slot S
Networ
System
Master
System Security
Data Execution Prevention Enabled
Virtualization Technology (VTx) ▶Enabled
Intel(R) VT-d Disabled
Intel TXT(LT) Support Disabled
F10=Accept, ESC=Cancel
System Security


CMOS Setup Utility - Copyright (C) 1984-2009 Awar
Advanced BIOS Features
Internal Graphics Mode [Disabled]
x UMA Frame Buffer Size 128MB
x Surround View Disabled
x Onboard VGA output connect D-SUB/DVI
Init Display First [PEG]
Virtualization [Enabled]
AMD K8 Cool&Quiet control [Auto]
Hard Disk Boot Priority [Press Enter]
First Boot Device [Hard Disk]
Second Boot Device [USB-HDD]
Third Boot Device [CDROM]
Password Check [Setup]
HDD S.M.A.R.T. Capability [Enabled]
Away Mode [Disabled]
Backup BIOS Image to HDD [Enabled]

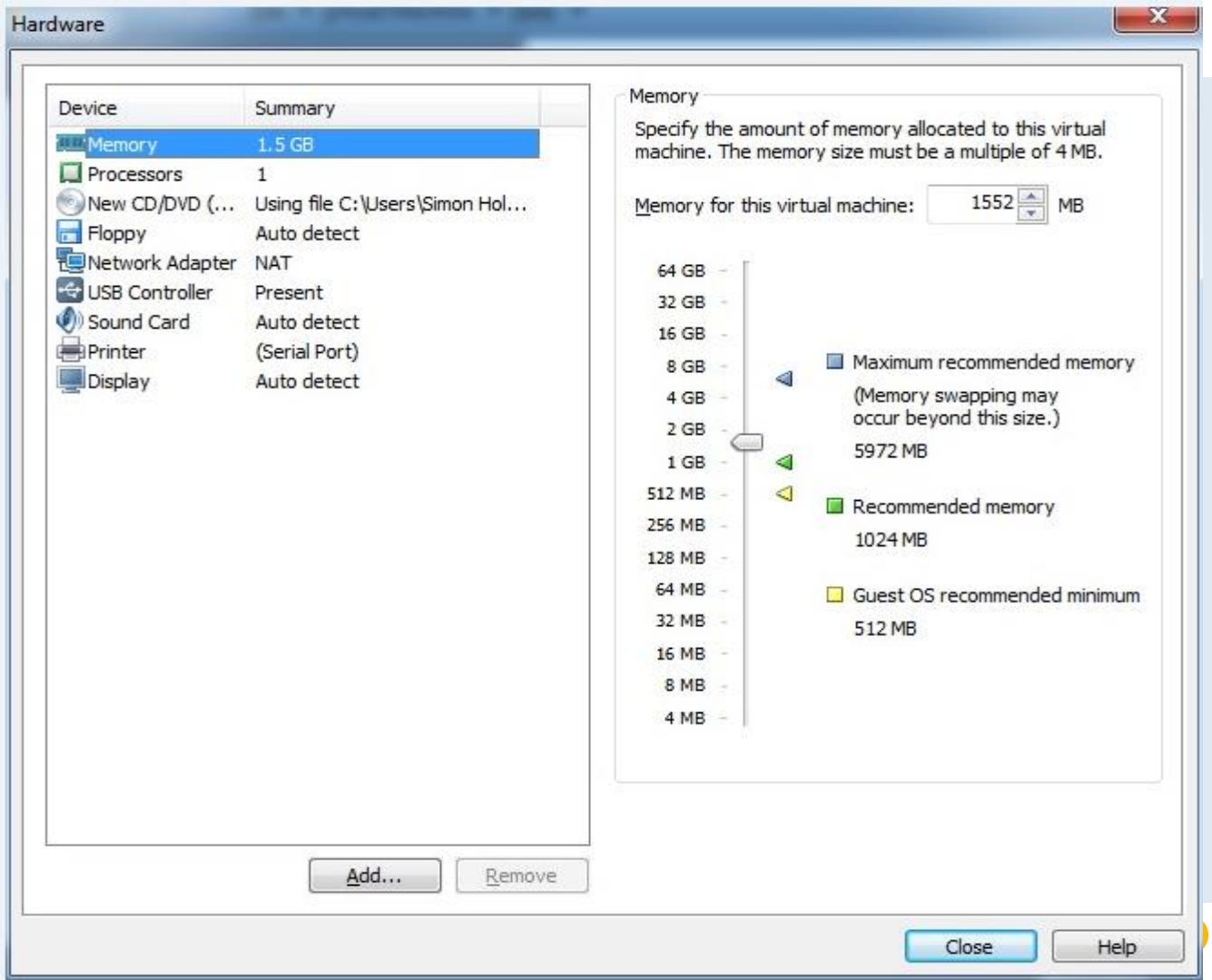

InsydeH20 Setup Ut
Information
Configuration
Security
Boot
System Time
[21:23:03]
System Date
[04/20/2013]
USB Legacy
[Enabled]
Wireless LAN Support
[Enabled]
SATA Controller Mode
[AHCI]
PXE Boot to LAN
[Enabled]
Power Beep
[Disabled]
Intel Virtual Technology
[Disabled]
Intel(R) AI Support
[Enabled]
System
Hotkey Mode
[Enabled]


Aptio Setup Utility – Copyright (C) 2008 American Megatrends, Inc.
Chipset
Tylerburg Revision ID B2 Stepping
Intel(R) VT for Directed I/O Configuration
Isoc [Disable]
Intel(R) I/OAT [Enable]
DCA Support [Enable]
IOH PCIe port Bifurcation Control
IOU2 - PCIe Port 1 [x4]
IOU0 - PCIe Port 3 [x8x8]
IOU1 - PCIe Port 7 [x8x8]
Intel(R) VT for Directed I/O Configuration
→←: Select Screen
↑↓: Select Item
Enter: Select
+/-: Change Opt.
F1: General Help
F2: Previous Values
F3: Optimized Defaults
F4: Save ESC: Exit
Version 1.24.1115. Copyright (C) 2008 American Megatrends, Inc.


Phoenix TrustedCore(tm) Setup Utility
Advanced
Advanced Processor Configuration
Item Specific Help
CPU Mismatch Detection: [Enabled]
Core Multi-Processing: [Enabled]
Processor Power Management: [Disabled]
Intel(R) Virtualization Technology [Enabled]
Execute Disable Bit: [Enabled]
Adjacent Cache Line Prefetch: [Disabled]
Hardware Prefetch: [Disabled]
Direct Cache Access [Disabled]
Set Max Ext CPUID = 3 [Disabled]
When enabled, a VMM (Virtual Machine Monitor) can utilize the additional hardware capabilities provided by Vanderpool Technology.
If this option is changed, a Power Off-On sequence will be applied on the next boot.
F1 Info ↑↓ Select Item -/+ Change Values F9 Setup Defaults
Esc Exit ← Select Menu Enter Select ▶ Sub-Menu F10 Save and Exit


CMOS Setup Utility - Copyright (C) 1984-2007 Award Software
Advanced BIOS Features
Hard Disk Boot Priority [Press Enter]
First Boot Device [Hard Disk]
Second Boot Device [Hard Disk]
Third Boot Device [Hard Disk]
Password Check [Setup]
HDD S.M.A.R.T. Capability [Enabled]
Limit CPUID Max. to 3 [Disabled]
No-Execute Memory Protect [Enabled]
CPU Enhanced Halt (C1E) [Enabled]
CPU Thermal Monitor 2(TM2) [Enabled]
CPU EIST Function [Enabled]
Virtualization Technology [Enabled]
Full Screen LOGO Show [Disabled]
Init Display First [PCI]
Item Help
Menu Level
When enabled, a VMM can utilize the additional hardware capabilities provided by Virtualization Technology
Sysprobs.com
↑↓→←:Move Enter:Select +/-/PU/PD:Value F10:Save ESC:Exit F1:General Help
F5:Previous Values F6:Fail-Safe Defaults F7:Optimized Defaults

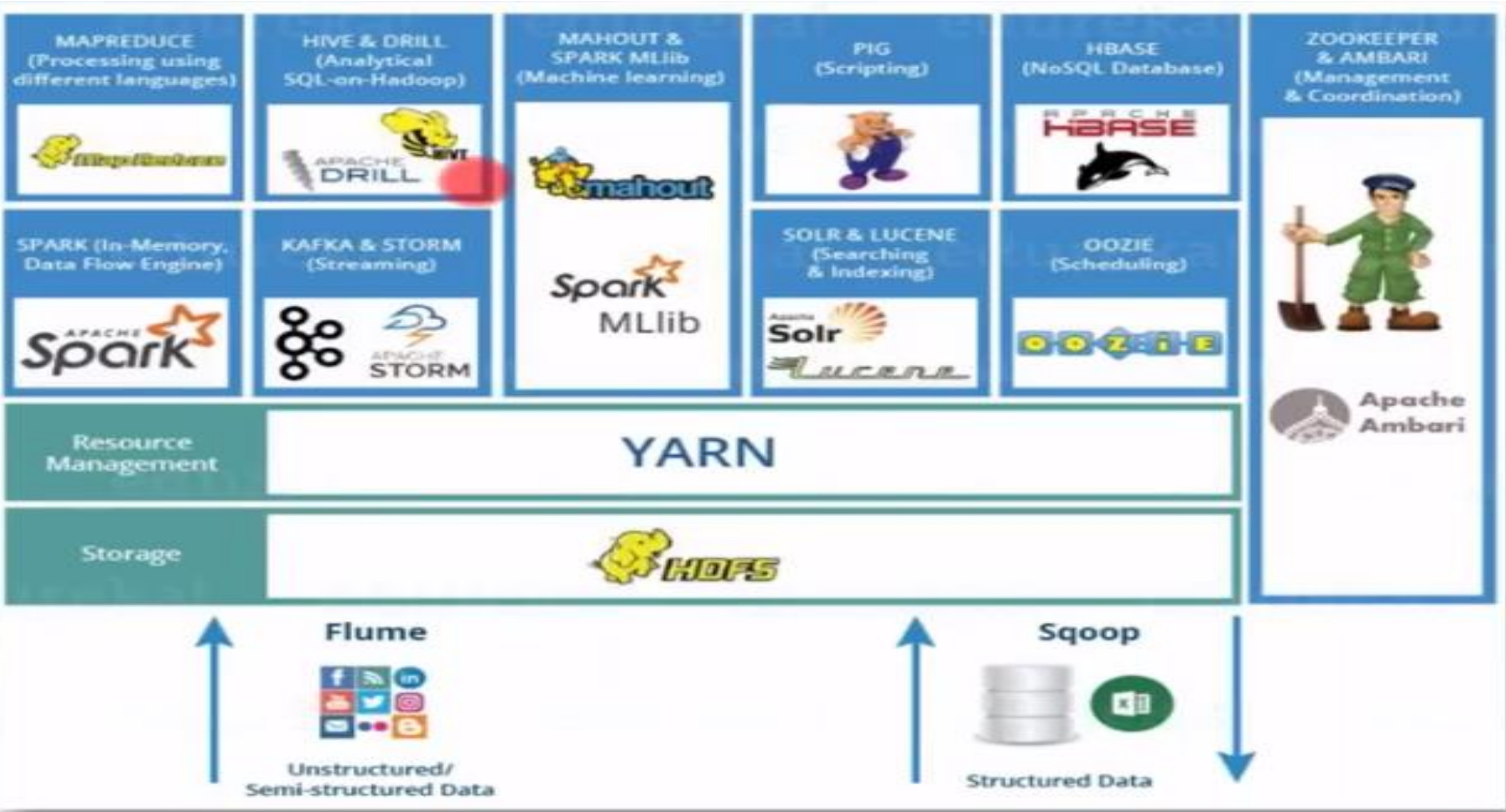

VMware Player
Player
Home
Windows 8.1
Ubuntu 14.10
Welcome to VMware Player
Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.
Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.
Download a Virtual Appliance
Download a virtual appliance from the marketplace. You can then open it in VMware Player.
Help
View VMware Player's help contents.
filehorse.com


Hardware
Device
Summary
Memory
1.5 GB
Processors
1
New CD/DVD (...
Using file C:\Users\Simon Hol...
Floppy
Auto detect
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
(Serial Port)
Display
Auto detect
Add...
Remove
Memory
Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.
Memory for this virtual machine:
1552
MB
64 GB
32 GB
16 GB
8 GB
4 GB
2 GB
1 GB
512 MB
256 MB
128 MB
64 MB
32 MB
16 MB
8 MB
4 MB
Maximum recommended memory
(Memory swapping may occur beyond this size.)
5972 MB
Recommended memory
1024 MB
Guest OS recommended minimum
512 MB
Close
Help

The Hadoop Ecosystem

Hadoop Ecosystem

- HDFS -> Hadoop Distributed File System
- YARN -> Yet Another Resource Negotiator
- MapReduce -> Data processing using programming
- Spark -> In-memory Data Processing
- PIG, HIVE-> Data Processing Services using Query (SQL-like)
- HBase -> NoSQL Database
- Mahout, Spark MLlib -> Machine Learning
- Apache Drill -> SQL on Hadoop
- Zookeeper -> Managing Cluster
- Oozie -> Job Scheduling
- Flume, Sqoop -> Data Ingesting Services
- Solr & Lucene -> Searching & Indexing
- Ambari -> Provision, Monitor and Maintain cluster


MAPREDUCE
(Processing using different languages)
HIVE & DRILL
(Analytical SQL-on-Hadoop)
APACHE DRILL
MAHOUT & SPARK MLlib
(Machine learning)
mahout
Spark MLlib
PIG
(Scripting)
HBASE
(NoSQL Database)
APACHE HBASE
ZOOKEEPER & AMBARI
(Management & Coordination)
Apache Ambari
SPARK (In-Memory, Data Flow Engine)
APACHE Spark
KAFKA & STORM
(Streaming)
APACHE STORM
SOLR & LUCENE
(Searching & Indexing)
Solr
Lucene
OOZIE
(Scheduling)
OOZIE
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

- Stores different types of large data sets (i.e. structured, unstructured and semi structured data)
- HDFS creates a level of abstraction over the resources, from where we can see the whole HDFS as a single unit
- Stores data across various nodes and maintains the log file about the stored data (metadata)
- HDFS has two core components, i.e. NameNode and DataNode

HDFS

- Performs all your processing activities by allocating resources and scheduling tasks
- Two services: ResourceManager and NodeManager
- ResourceManager: Manages resources and schedule applications running on top of YARN
- NodeManager: Manages containers and monitors resource utilization in each container

YARN

- Core component in a Hadoop Ecosystem for processing
- Helps in writing applications that processes large data sets using distributed and parallel algorithms
- In a MapReduce program, Map() and Reduce() are two functions
- Map function performs actions like filtering, grouping and sorting
- Reduce function aggregates and summarizes the result produced by map function

- PIG has two parts: Pig Latin, the language and the pig runtime, for the execution environment
- **1 line of pig latin = approx. 100 lines of Map-Reduce job**
- The compiler internally converts pig latin to MapReduce
- It gives you a platform for building data flow for ETL (Extract, Transform and Load)
- PIG first loads the data, then performs various functions like grouping, filtering, joining, sorting, etc. and finally dumps the data on the screen or stores in HDFS.

- A data warehousing component which analyses data sets in a distributed environment using SQL-like interface
- The query language of Hive is called Hive Query Language(HQL)
- 2 basic components: Hive Command Line and JDBC/ODBC driver
- Supports user defined functions (UDF) to accomplish specific needs

- Provides an environment for creating machine learning applications
- It performs collaborative filtering, clustering and classification
- Provides a command line to invoke various algorithms.
- It has a predefined set of library which already contains different inbuilt algorithms for different use cases.

- A framework for real time data analytics in a distributed computing environment.
- Written in Scala and was originally developed at the University of California, Berkeley.
- It executes in-memory computations to increase speed of data processing over Map-Reduce.
- 100x faster than Hadoop for large scale data processing by exploiting in-memory computations

- An open source, non-relational distributed database - a **NoSQL** database
- Supports all types of data and that is why, it's capable of handling anything and everything
- It is modelled after Google's BigTable
- It gives us a fault tolerant way of storing sparse data
- It is written in Java, and HBase applications can be written in REST, Avro and Thrift APIs

- An open source application which works with distributed environment to analyze large data sets
- Follows the ANSI SQL
- Supports different kinds NoSQL databases and file systems
- For example: Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS and local files
- Combines a variety of data stores just by using a single query

- Oozie is a job scheduler in Hadoop ecosystem
- Two kinds of Oozie jobs: Oozie workflow and Oozie Coordinator
- **Oozie workflow**: Sequential set of actions to be executed
- **Oozie Coordinator**: Oozie jobs which are triggered when the data is made available to it or even triggered based on time

- Ingests unstructured and semi-structured data into HDFS.
- It helps us in collecting, aggregating and moving large amount of data sets.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.

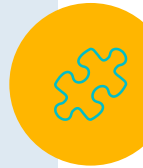
- Another data ingesting service
- Sqoop can import as well as export structured data from RDBMS
- Flume only ingests unstructured data or semi-structured data into HDFS

- Two services which are used for searching and indexing in Hadoop Ecosystem
- Apache Lucene is based on Java, which also helps in spell checking
- Apache Lucene is the engine, Apache Solr is a complete application built around Lucene
- Solr uses Apcahe Lucene Java search library for searching and indexing

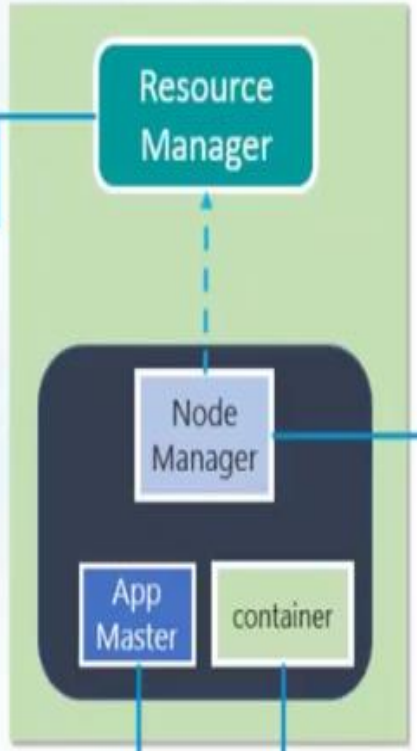
- An open-source server which enables highly reliable distributed coordination
- Apache Zookeeper coordinates with various Hadoop services in a distributed environment
- Performs synchronization, configuration maintenance, grouping and naming

- Software for provisioning, managing and monitoring Apache Hadoop clusters
- Gives us step by step process for installing Hadoop services
- Handles configuration of Hadoop services
- Provides a central management service for starting, stopping and re-configuring Hadoop services
- Monitors health and status of the Hadoop cluster

YARN COMPONENTS



YARN Components



ResourceManager:

- Master daemon that manages all other daemons & accepts job submission
- Allocates first container for the AppMaster

NodeManager:

- Responsible for containers, monitoring their resource usage i.e. (cpu, memory, disk, network) & reports the same to RM

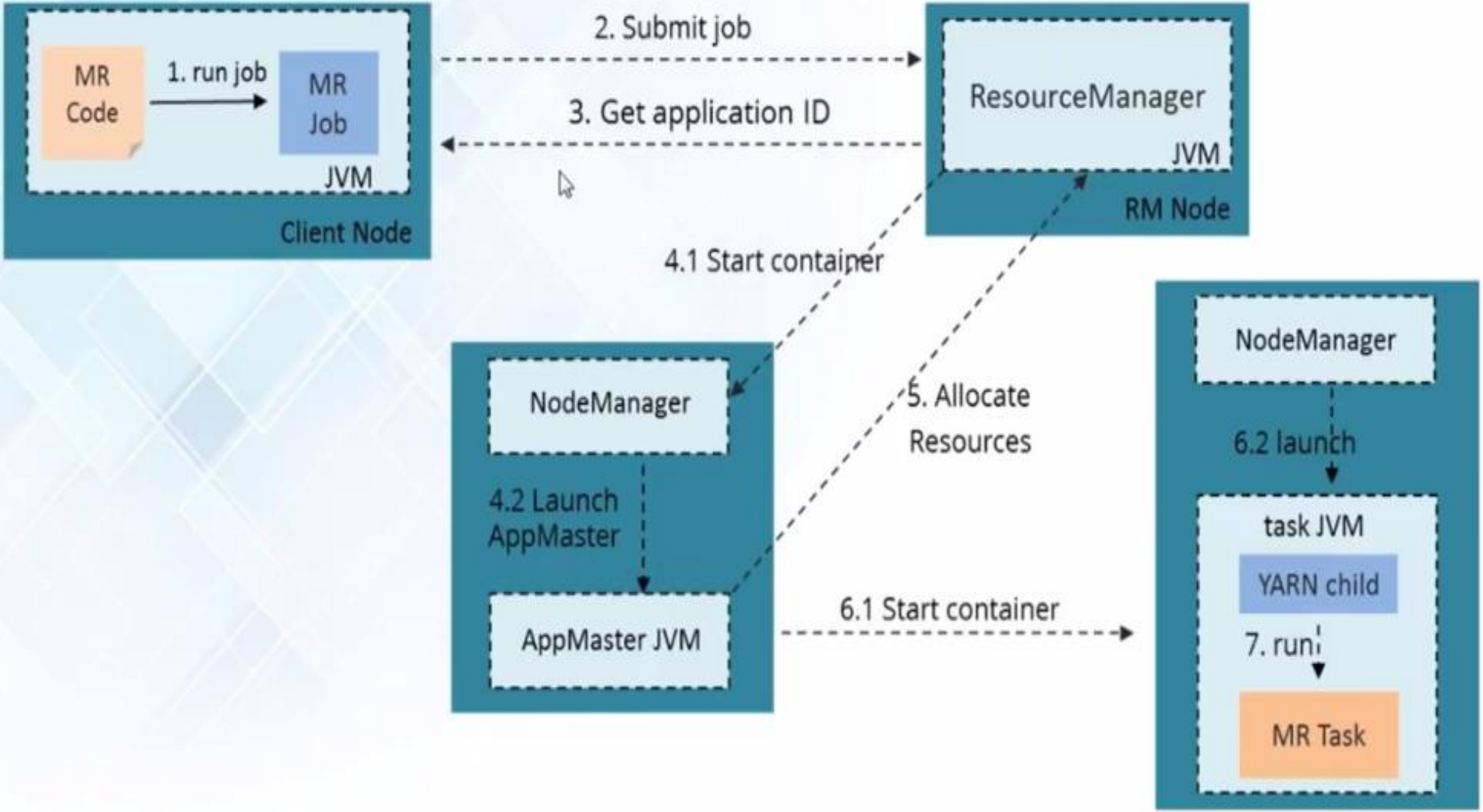
AppMaster:

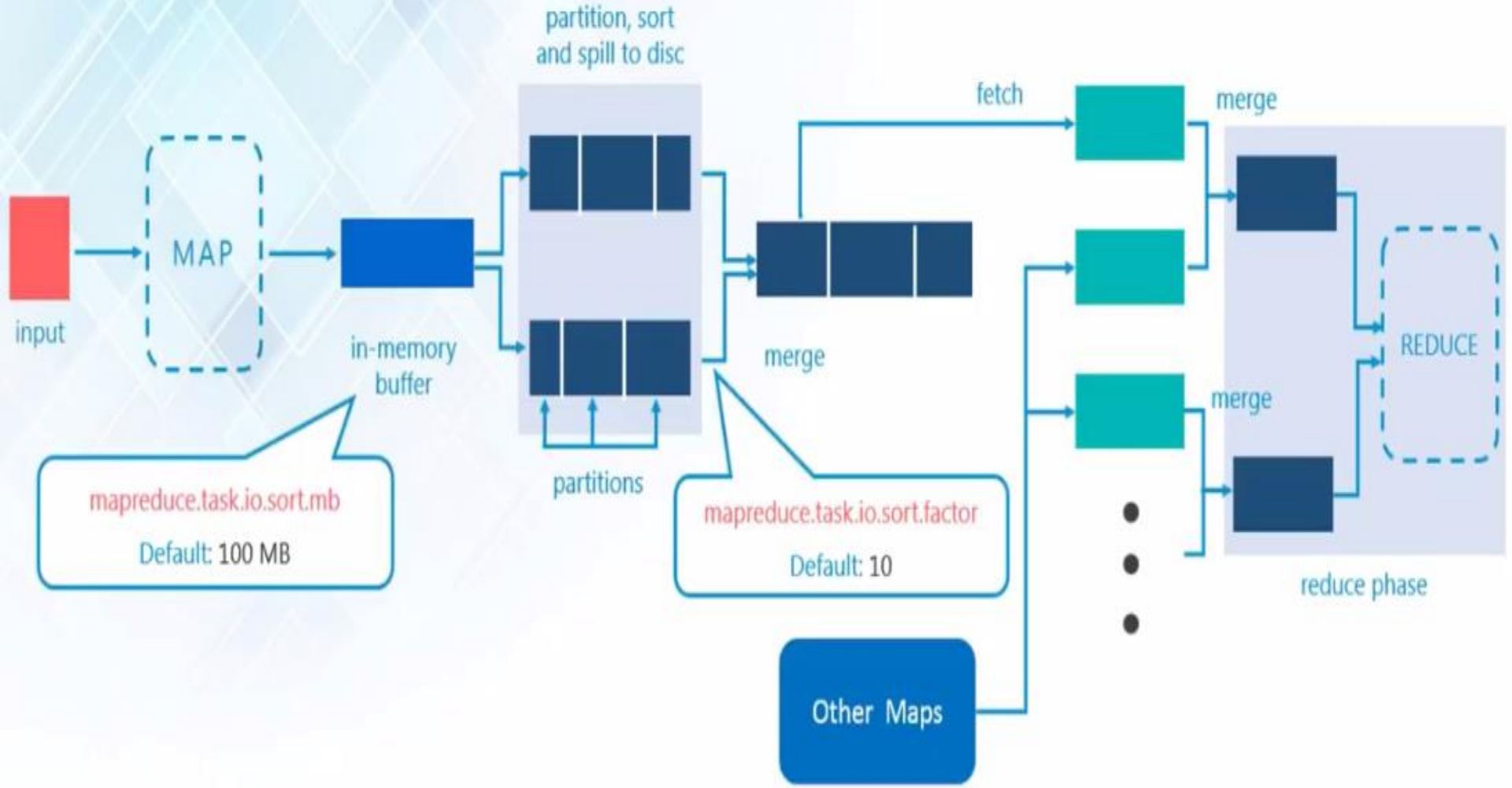
- One per application
- Coordinates and manages MR Jobs
- Negotiates resources from RM

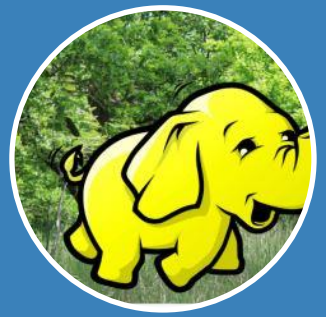
Container:

- Allocates certain amount of resources (memory, CPU etc.) on a slave node (NM)

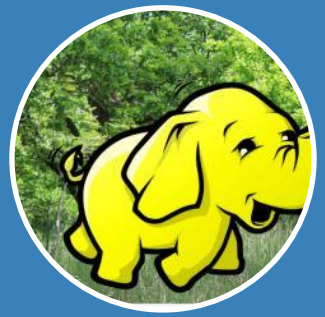
MAPREDUCE JOB WORKFLOW


MR Code
1. run job
MR Job
JVM
Client Node
2. Submit job
3. Get application ID
ResourceManager
JVM
RM Node
4.1 Start container
NodeManager
4.2 Launch AppMaster
AppMaster JVM
5. Allocate Resources
6.1 Start container
NodeManager
6.2 launch
task JVM
YARN child
7. run
MR Task

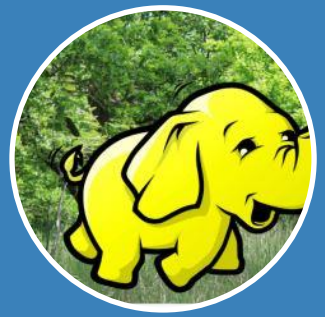

partition, sort
and spill to disc
fetch
merge
input
MAP
in-memory
buffer
merge
REDUCE
merge
partitions
mapreduce.task.io.sort.mb
Default: 100 MB
mapreduce.task.io.sort.factor
Default: 10
Other Maps
reduce phase



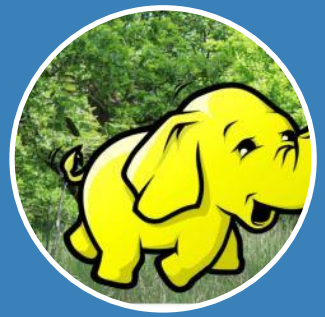
HADOOP ARCHITECTURE-



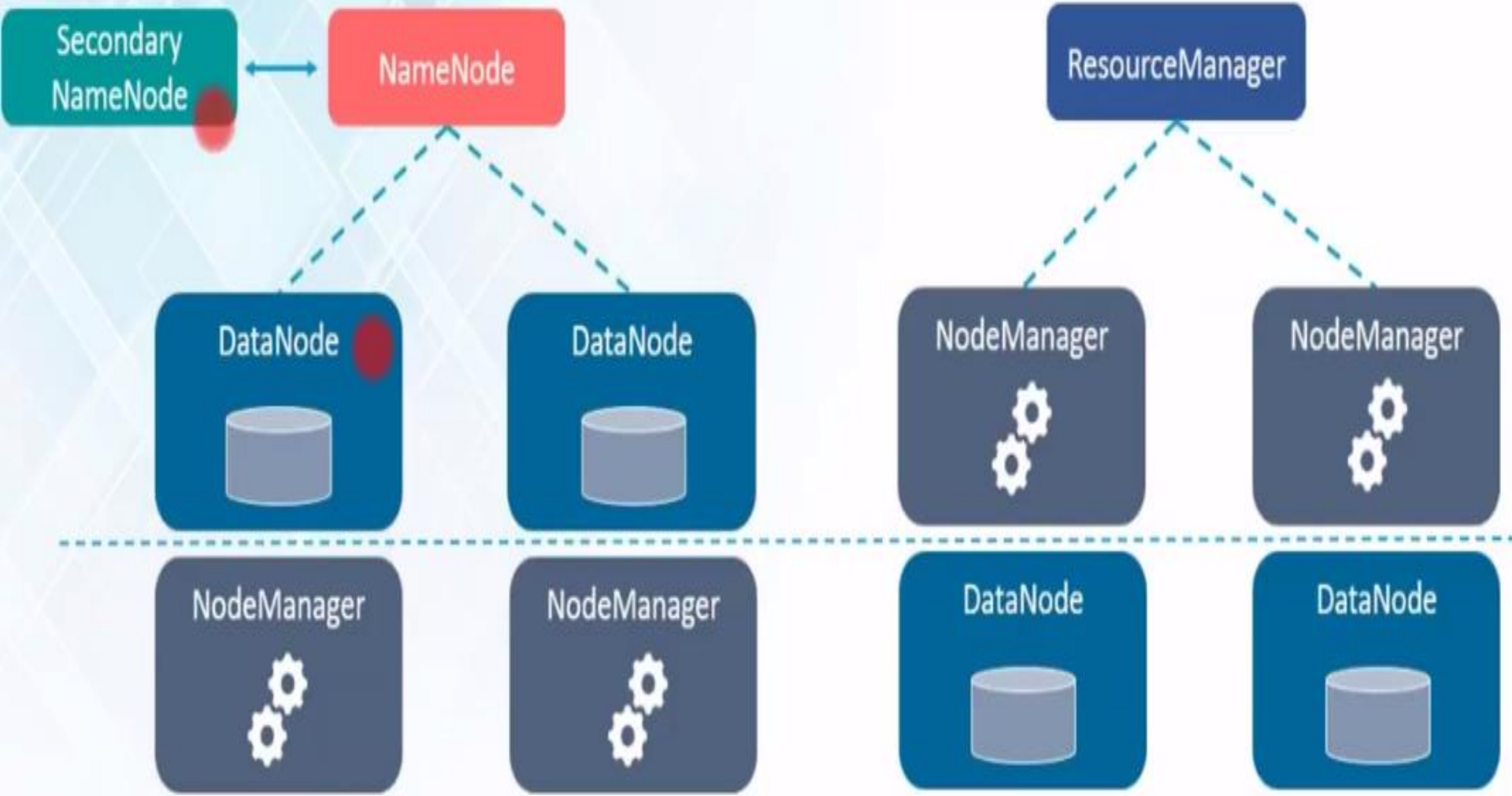
HDFS



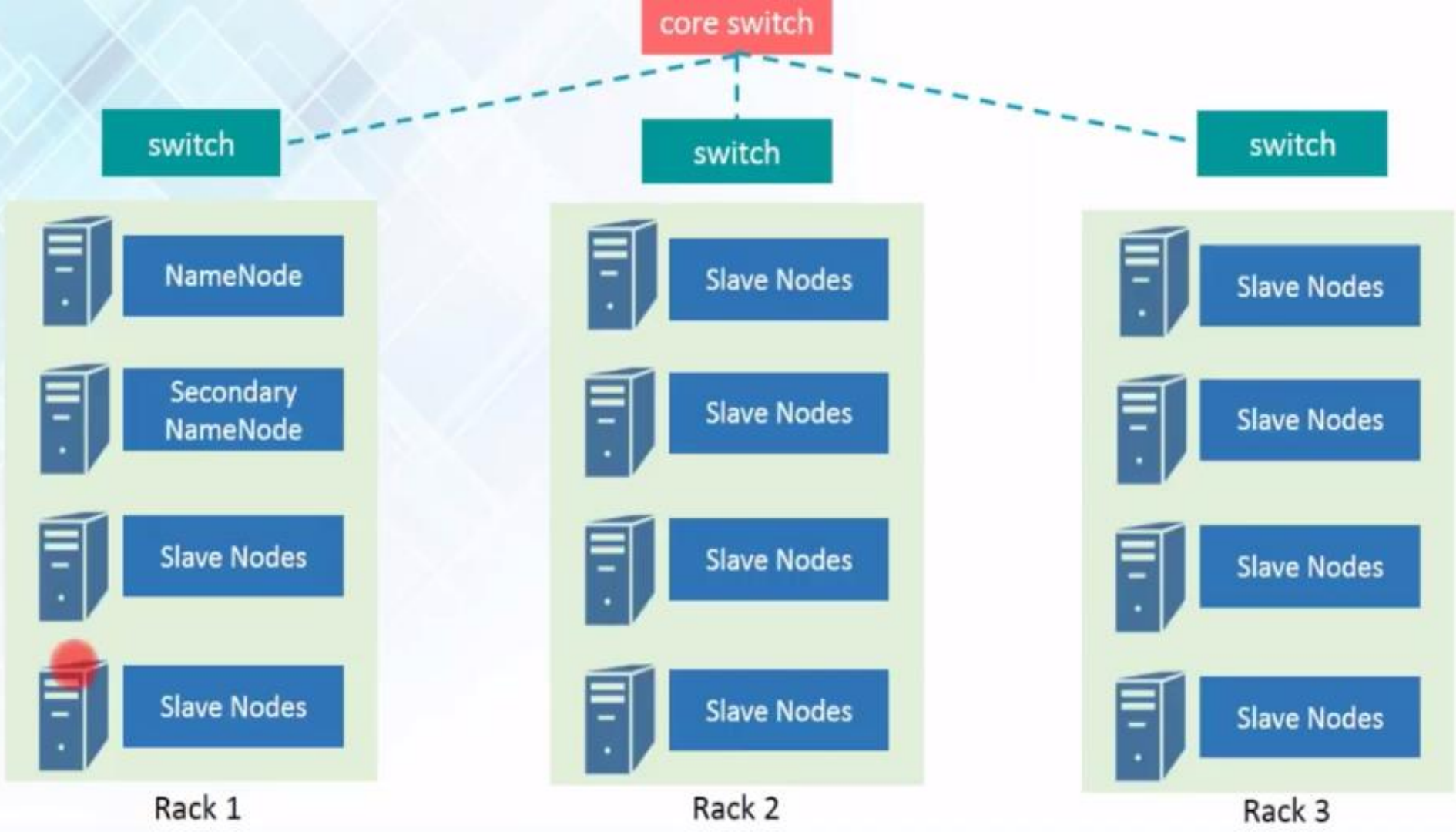
&

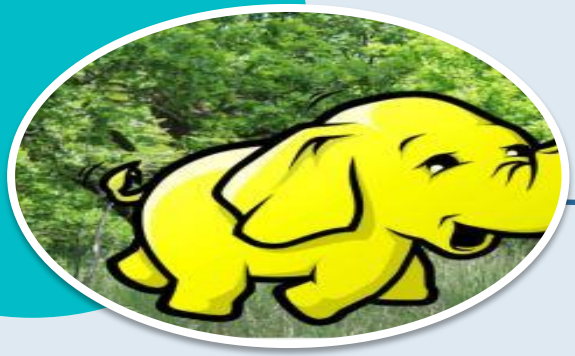


YARN


Secondary NameNode
NameNode
ResourceManager
DataNode
DataNode
NodeManager
NodeManager
NodeManager
NodeManager
DataNode
DataNode

HADOOP CLUSTER


core switch
switch
switch
switch
NameNode
Secondary NameNode
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Slave Nodes
Rack 1
Rack 2
Rack 3



HADOOP CLUSTER MODES

Standalone (or Local) Mode

- No daemons, everything runs in a single JVM
- Suitable for running MapReduce programs during development
- Has no DFS or Distributed File System

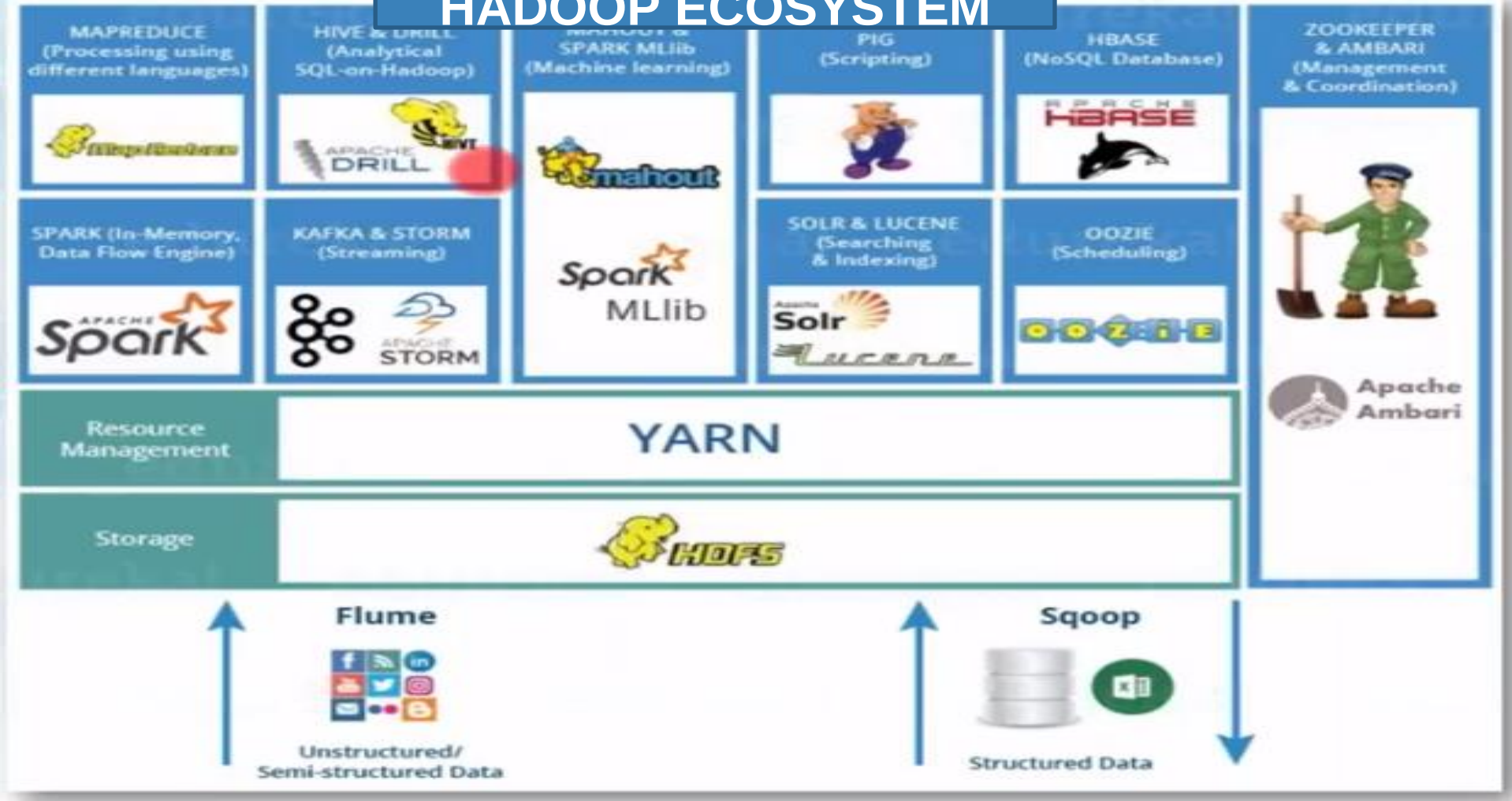
Pseudo Distributed Mode

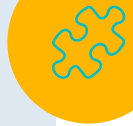
- All Hadoop daemons run on the local machine

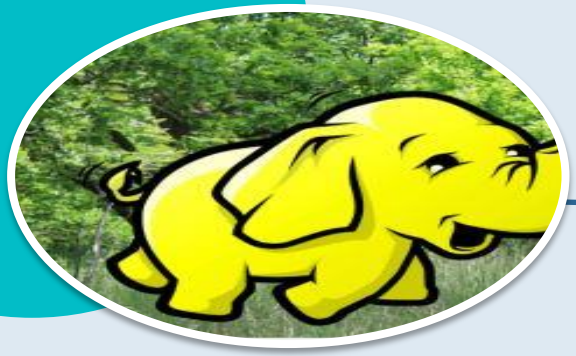
Multi-Node Cluster Mode

- Hadoop daemons run on a cluster of machines

HADOOP ECOSYSTEM







HADOOP USE CASE ANALYZING OLYMPIC DATASET

Hadoop Use Case: Analyzing Olympic Dataset

Problem statement:

- Find the list of top 10 countries won the highest medals
- Find the total number of gold medals won by each country
- Which countries have won the most number of medals in swimming?

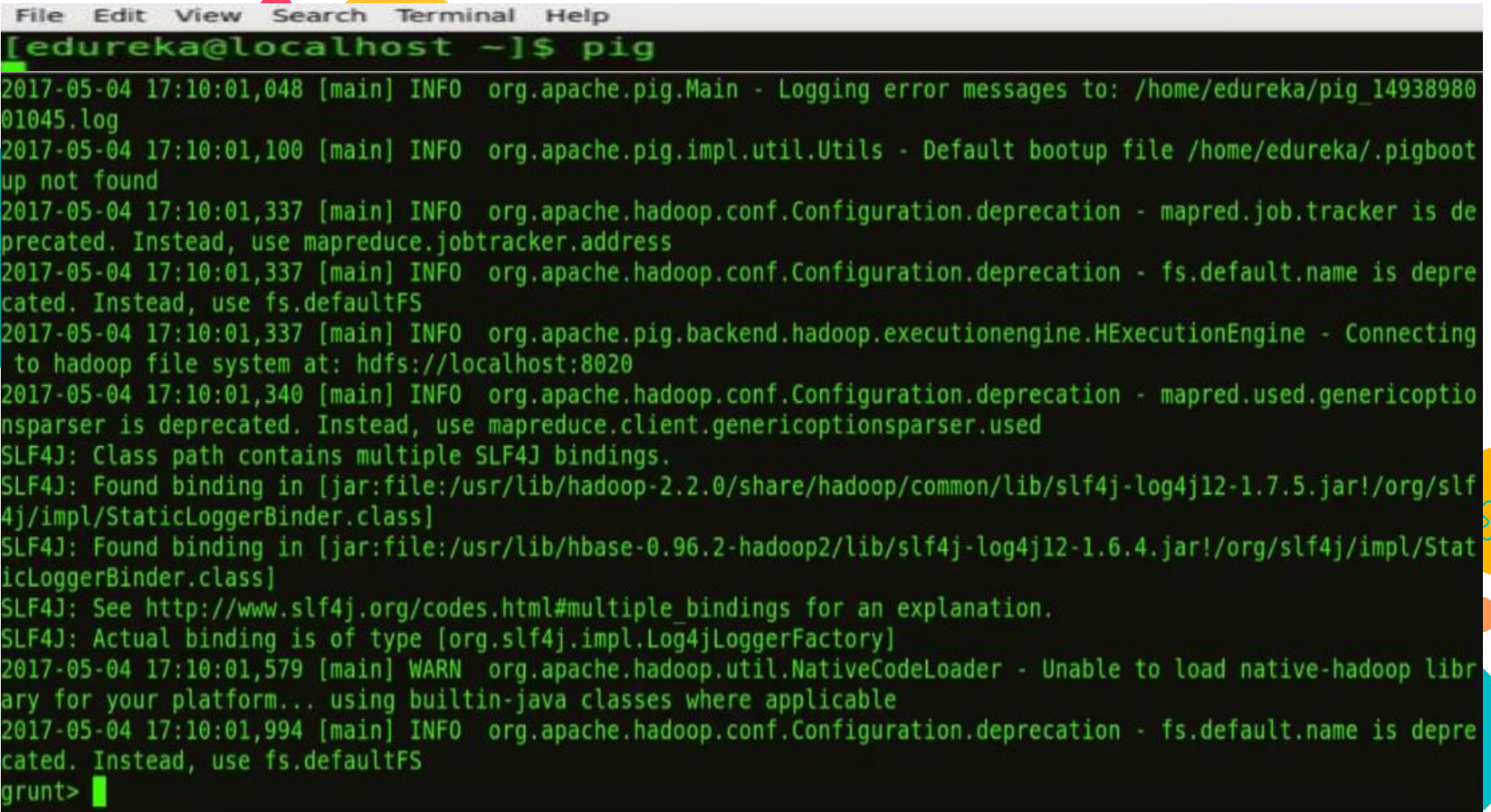
Dataset Description

The data set consists of the following fields:

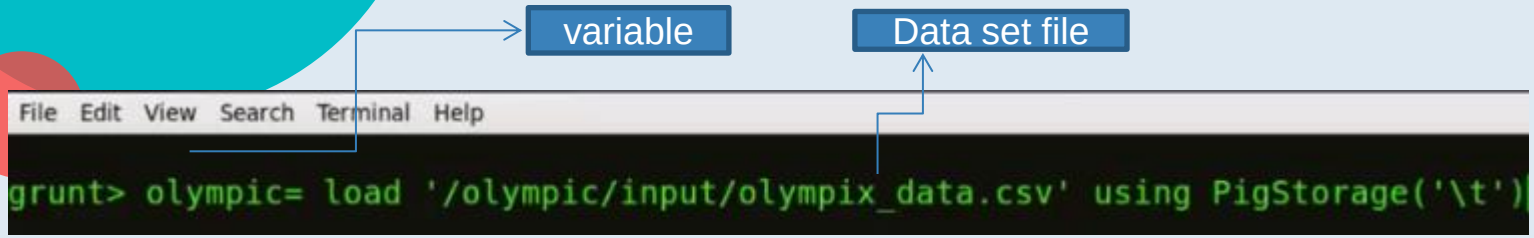
- **Athlete**: This field consists of the athlete name
- **Age**: This field consists of athlete ages
- **Country**: This fields consists of the country names which participated in Olympics
- **Year**: This field consists of the year
- **Closing Date**: This field consists of the closing date of ceremony
- **Sport**: Consists of the sports name
- **Gold Medals**: No. of Gold medals
- **Silver Medals**: No.of Silver medals
- **Bronze Medals**: No.of Bronze medals
- **Total Medals**: Consists of total no of medals

Dataset Description

Athlete	Age	Country	Year	Closing Ceremony Date	Sport	Gold Medals	Silver Medals	Bronze Medals	Total Medals
Michael Phelps	23	United States	2008	08-24-08	Swimming	8	0	0	8
Michael Phelps	19	United States	2004	08-29-04	Swimming	6	0	2	8
Michael Phelps	27	United States	2012	08-12-12	Swimming	4	2	0	6
Natalie Coughlin	25	United States	2008	08-24-08	Swimming	1	2	3	6
Aleksey Nemov	24	Russia	2000	10-01-00	Gymnastics	2	1	3	6
Alicia Coutts	24	Australia	2012	08-12-12	Swimming	1	3	1	5
Missy Franklin	17	United States	2012	08-12-12	Swimming	4	0	1	5
Ryan Lochte	27	United States	2012	08-12-12	Swimming	2	2	1	5
Allison Schmitt	22	United States	2012	08-12-12	Swimming	3	1	1	5
Natalie Coughlin	21	United States	2004	08-29-04	Swimming	2	2	1	5
Ian Thorpe	17	Australia	2000	10-01-00	Swimming	3	2	0	5
Dara Torres	33	United States	2000	10-01-00	Swimming	2	0	3	5
Cindy Klassen	26	Canada	2006	02-26-06	Speed Skating	1	2	2	5
Nastia Liukin	18	United States	2008	08-24-08	Gymnastics	1	3	1	5
Marit Bjørgen	29	Norway	2010	02-28-10	Cross Country Skiing	3	1	1	5
Sun Yang	20	China	2012	08-12-12	Swimming	2	1	1	4
Kirsty Coventry	24	Zimbabwe	2008	08-24-08	Swimming	1	3	0	4
Libby Lenton-Trickett	23	Australia	2008	08-24-08	Swimming	2	1	1	4
Ryan Lochte	24	United States	2008	08-24-08	Swimming	2	0	2	4
Inge de Bruijn	30	Netherlands	2004	08-29-04	Swimming	1	1	2	4
Petria Thomas	28	Australia	2004	08-29-04	Swimming	3	1	0	4
Ian Thorpe	21	Australia	2004	08-29-04	Swimming	2	1	1	4
Inge de Bruijn	27	Netherlands	2000	10-01-00	Swimming	3	1	0	4
Gary Hall Jr.	25	United States	2000	10-01-00	Swimming	2	1	1	4
Michael Klim	23	Australia	2000	10-01-00	Swimming	2	2	0	4
Susie O'Neill	27	Australia	2000	10-01-00	Swimming	1	3	0	4


File Edit View Search Terminal Help
[edureka@localhost ~]$ pig
2017-05-04 17:10:01,048 [main] INFO org.apache.pig.Main - Logging error messages to: /home/edureka/pig_1493898001045.log
2017-05-04 17:10:01,100 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/edureka/.pigbootup not found
2017-05-04 17:10:01,337 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2017-05-04 17:10:01,337 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2017-05-04 17:10:01,337 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8020
2017-05-04 17:10:01,340 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.used.genericoptionsparser is deprecated. Instead, use mapreduce.client.genericoptionsparser.used
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hadoop-2.2.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hbase-0.96.2-hadoop2/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2017-05-04 17:10:01,579 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2017-05-04 17:10:01,994 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>

This is the command we should type in terminal
Variable name =path of the directory where the files is stored
Olympic=load '/Olympic/input/olymix_data.csv' using PigStorage('\t')



```
File  Edit  View  Search  Terminal  Help

grunt> olympic= load '/olympic/input/olympix_data.csv' using PigStorage('\t')
```

To see the list of file we should use the command called

“Dump Olympic”

```
grunt> olympic= load '/olympic/input/olympix_data.csv' using PigStorage('\t');
2017-05-04 18:25:07,836 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is depre
cated. Instead, use fs.defaultFS
2017-05-04 18:25:07,836 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.l
imit is deprecated. Instead, use mapreduce.job.counters.max
2017-05-04 18:25:07,837 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - dfs.permissions is depre
cated. Instead, use dfs.permissions.enabled
2017-05-04 18:25:07,837 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is
 deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump olympic
```

```
grunt> olympic= load '/olympic/input/olympix_data.csv' using PigStorage('\t');
2017-05-04 18:25:07,836 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is depre
cated. Instead, use fs.defaultFS
2017-05-04 18:25:07,836 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.l
imit is deprecated. Instead, use mapreduce.job.counters.max
2017-05-04 18:25:07,837 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - dfs.permissions is depre
cated. Instead, use dfs.permissions.enabled
2017-05-04 18:25:07,837 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is
 deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump olympic
(Butch Johnson,45,United States,2000,10-01-00,Archery,0,0,1,1)
(Kim Cheong-Tae,20,South Korea,2000,10-01-00,Archery,1,0,0,1)
(Barbara Mensing,39,Germany,2000,10-01-00,Archery,0,0,1,1)
(O Gyo-Mun,28,South Korea,2000,10-01-00,Archery,1,0,0,1)
(Cornelia Pfohl,29,Germany,2000,10-01-00,Archery,0,0,1,1)
(Olena Sadovnycha,32,Ukraine,2000,10-01-00,Archery,0,1,0,1)
(Kateryna Serdiuk,17,Ukraine,2000,10-01-00,Archery,0,1,0,1)
(Wietse van Alten,21,Netherlands,2000,10-01-00,Archery,0,0,1,1)
(Sandra Wagner-Sachse,31,Germany,2000,10-01-00,Archery,0,0,1,1)
(Rod White,23,United States,2000,10-01-00,Archery,0,0,1,1)
grunt>
```

To list out the Top Ten countries of highest medals the command is as follows

Country_final=foreach olympic genteate $2 as Country $9 as total_medals;

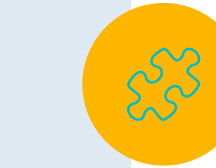
```
File  Edit  View  Search  Terminal  Help

grunt> Country_final= foreach olympic generate $2 as Country, $9 as total_medals;
```

To See the list we should use the command

Dump Country_final

```
grunt> Country_final= foreach olympic generate $2 as Country, $9 as total_medals;
2017-05-04 18:28:09,060 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.l
imit is deprecated. Instead, use mapreduce.job.counters.max
2017-05-04 18:28:09,061 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - dfs.permissions is depre
cated. Instead, use dfs.permissions.enabled
2017-05-04 18:28:09,061 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is
 deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump Country_final
```

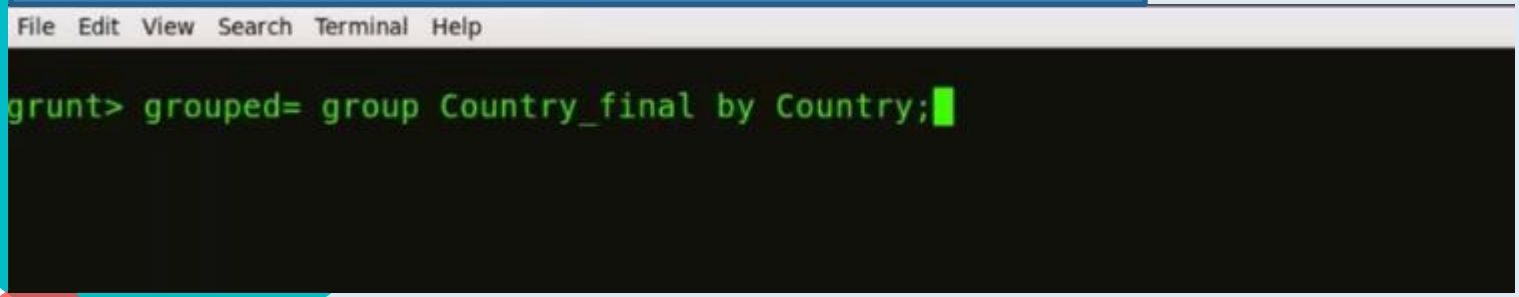
```
File  Edit  View  Search  Terminal  Help
(Chinese Taipei,1)
(Great Britain,1)
(Chinese Taipei,1)
(Japan,1)
(Chinese Taipei,1)
(South Korea,1)
(China,1)
(Italy,1)
(Ukraine,1)
(Italy,1)
(Australia,1)
(Italy,1)
(South Korea,1)
(United States,1)
(South Korea,1)
(Germany,1)
(South Korea,1)
(Germany,1)
(Ukraine,1)
(Ukraine,1)
(Netherlands,1)
(Germany,1)
(United States,1)
grunt>
```



In the previous slide the data was scattered to group the list the command is as follows

Variable name =class name file name by Indexname

Grouped=group Country_final by Country;



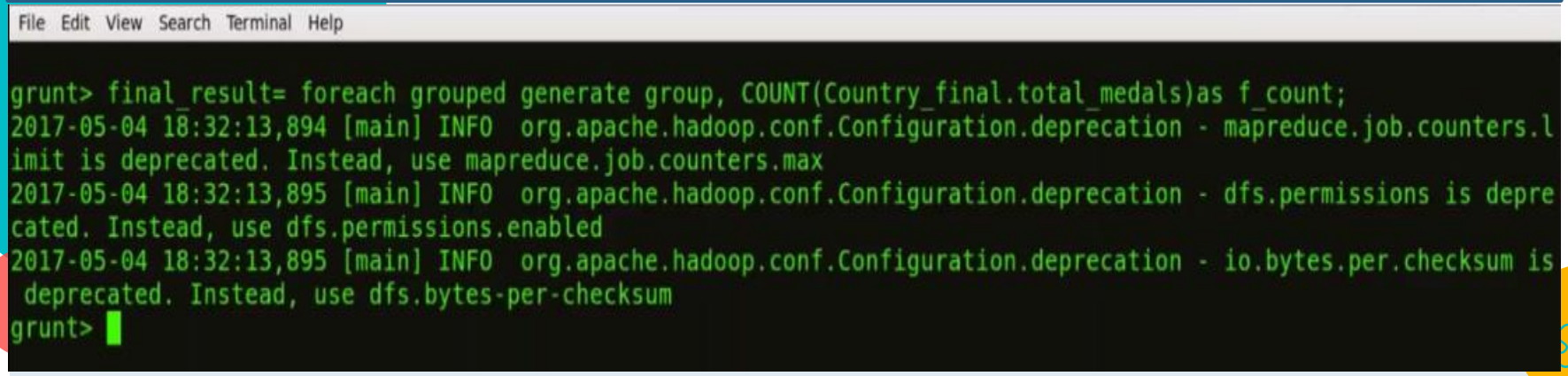
File Edit View Search Terminal Help

```
Republic,1),(Czech Republic,1),(Czech Republic,2),(Czech Republic,2),(Czech Republic,1),(Czech Republic,1),(Czec
h Republic,1),(Czech Republic,2),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Cz
ech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(
Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,2),(Czech Republic,1),(Czech Republic,1)
,(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,
1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,1),(Czech Republic,3),(Czech Republic,1)})
(Dominican Republic,{(Dominican Republic,1),(Dominican Republic,1),(Dominican Republic,1),(Dominican Republic,1)
,(Dominican Republic,1)})
(Trinidad and Tobago,{(Trinidad and Tobago,1),(Trinidad and Tobago,1),(Trinidad and Tobago,1),(Trinidad and Toba
go,1),(Trinidad and Tobago,1),(Trinidad and Tobago,2),(Trinidad and Tobago,2),(Trinidad and Tobago,2),(Trinidad
and Tobago,1),(Trinidad and Tobago,1),(Trinidad and Tobago,1),(Trinidad and Tobago,1),(Trinidad and Tobago,1),(T
rinidad and Tobago,1),(Trinidad and Tobago,1),(Trinidad and Tobago,1)})
(United Arab Emirates,{(United Arab Emirates,1)})
(Serbia and Montenegro,{(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia an
d Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Monten
egro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),
(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia
and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Mont
enegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1
),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbi
a and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Mo
ntenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro,1),(Serbia and Montenegro
,1)})
grunt>
```

To see the final result of all the grouped “Country and Total Medals

Variable name=followed by inbuilt functions

Final_result=foreach grouped generate group, COUNT (Country_final.total_medals) as f_count;

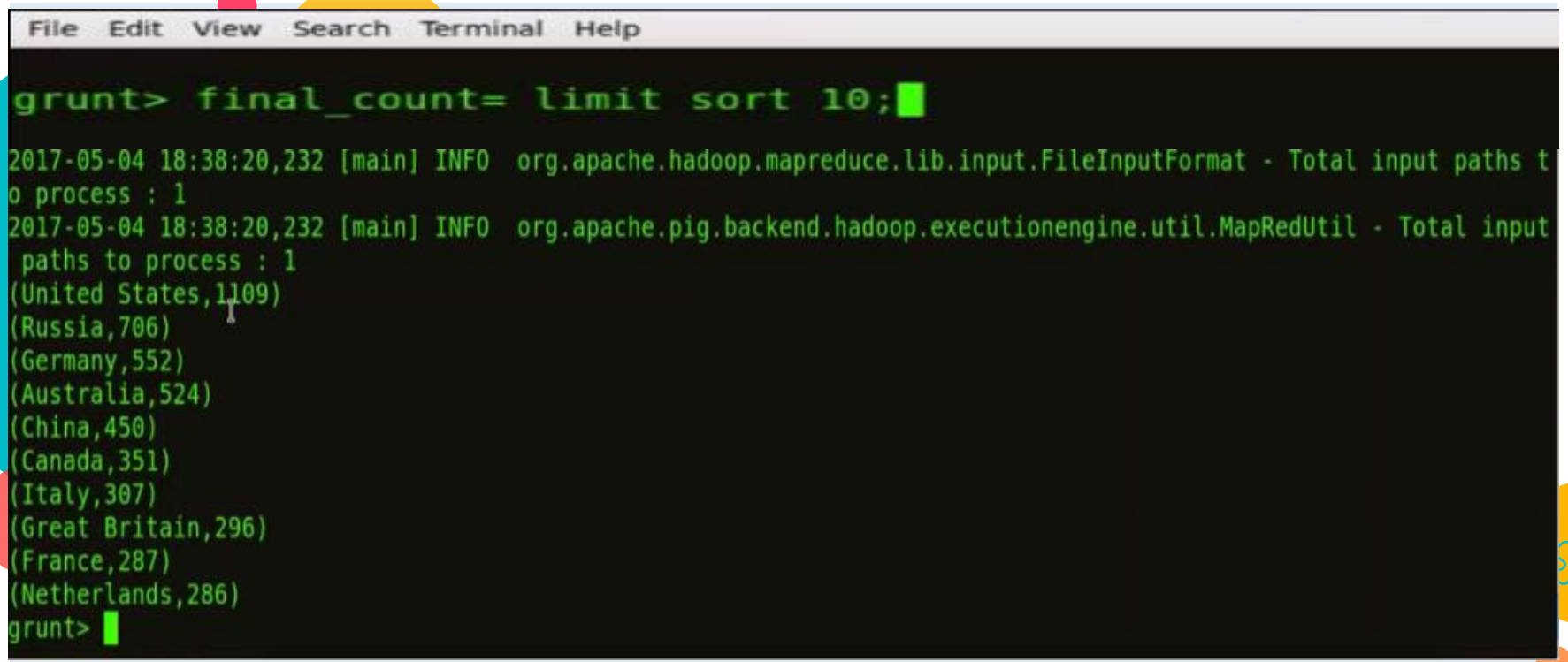


```
File  Edit  View  Search  Terminal  Help
(Kazakhstan,42)
(Kyrgyzstan,3)
(Montenegro,14)
(Mozambique,1)
(Tajikistan,3)
(Uzbekistan,19)
(Afghanistan,2)
(Netherlands,286)
(New Zealand,51)
(North Korea,21)
(Puerto Rico,2)
(South Korea,274)
(Switzerland,87)
(Saudi Arabia,6)
(South Africa,22)
(Great Britain,296)
(United States,1109)
(Chinese Taipei,20)
(Czech Republic,75)
(Dominican Republic,5)
(Trinidad and Tobago,16)
(United Arab Emirates,1)
(Serbia and Montenegro,38)
grunt>
```

To display in sorted order the command is as follows

Sort= order final_result by f_count desc;

```
File  Edit  View  Search  Terminal  Help

grunt> sort= order final_result by f_count desc;
(United States,1109)
(Russia,706)
(Germany,552)
(Australia,524)
(China,450)
(Canada,351)
(Italy,307)
(Great Britain,296)
(France,287)
(Netherlands,286)
(South Korea,274)
(Japan,259)
(Brazil,220)
(Spain,195)
(Cuba,188)
(Sweden,167)
(Norway,158)
(Argentina,141)
(Ukraine,137)
```

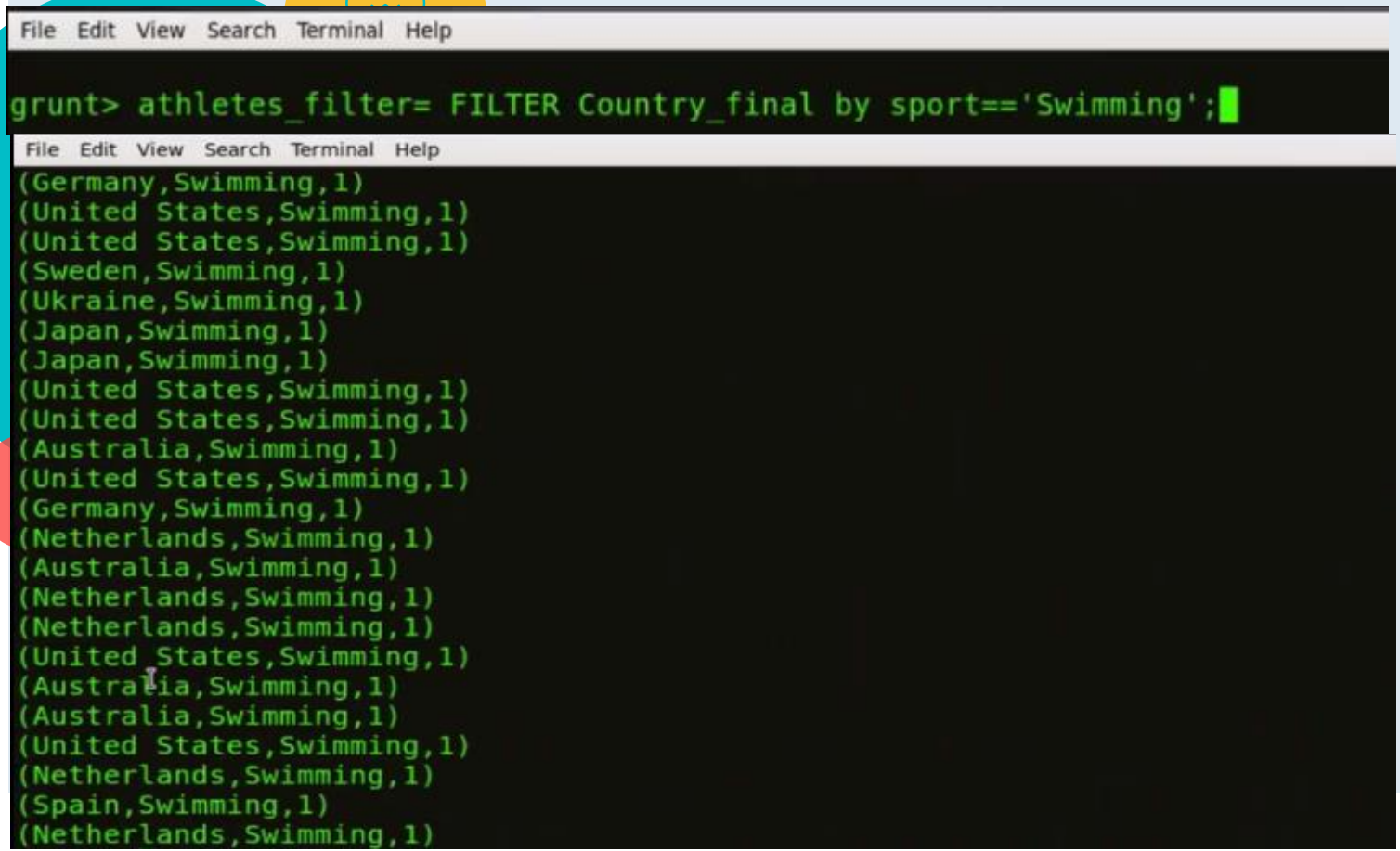

File Edit View Search Terminal Help
grunt> final_count= limit sort 10;
2017-05-04 18:38:20,232 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths t
o process : 1
2017-05-04 18:38:20,232 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input
paths to process : 1
(United States,1109)
(Russia,706)
(Germany,552)
(Australia,524)
(China,450)
(Canada,351)
(Italy,307)
(Great Britain,296)
(France,287)
(Netherlands,286)
grunt>

To store the output we should use the command is as follows

store final_count into '/olympic/output/usecasefirst;

```
File  Edit  View  Search  Terminal  Help

grunt> store final_count into '/olympic/output/usecasefirs
```

```
File  Edit  View  Search  Terminal  Help

grunt> olympic= load '/olympic/input/olympix_data.csv' using PigStorage('\t');
2017-05-04 18:43:51,595 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is depre
cated. Instead, use fs.defaultFS
2017-05-04 18:43:51,595 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.l
imit is deprecated. Instead, use mapreduce.job.counters.max
2017-05-04 18:43:51,596 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - dfs.permissions is depre
cated. Instead, use dfs.permissions.enabled
2017-05-04 18:43:51,596 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is
 deprecated. Instead, use dfs.bytes-per-checksum
grunt> Country_final= foreach olympic generate $2 as Country, $5 as sport,$9 as total_medals;
```


File Edit View Search Terminal Help
grunt> athletes_filter= FILTER Country_final by sport=='Swimming';
File Edit View Search Terminal Help
(Germany,Swimming,1)
(United States,Swimming,1)
(United States,Swimming,1)
(Sweden,Swimming,1)
(Ukraine,Swimming,1)
(Japan,Swimming,1)
(Japan,Swimming,1)
(United States,Swimming,1)
(United States,Swimming,1)
(Australia,Swimming,1)
(United States,Swimming,1)
(Germany,Swimming,1)
(Netherlands,Swimming,1)
(Australia,Swimming,1)
(Netherlands,Swimming,1)
(Netherlands,Swimming,1)
(United States,Swimming,1)
(Australia,Swimming,1)
(Australia,Swimming,1)
(United States,Swimming,1)
(Netherlands,Swimming,1)
(Spain,Swimming,1)
(Netherlands,Swimming,1)


Questions
Queries
Feedback