# HADOOP TECHNOLOGY

## What is Hadoop Technology??

- The most well known technology used for Big Data is Hadoop.
- It is actually a large scale batch data processing system

# Why Hadoop ??

- Distributed cluster system
- Platform for massively scalable applications
- Enables parallel data processing

# Developers of Hadoop Technology:

Michael j. cafarella

Doug cutting

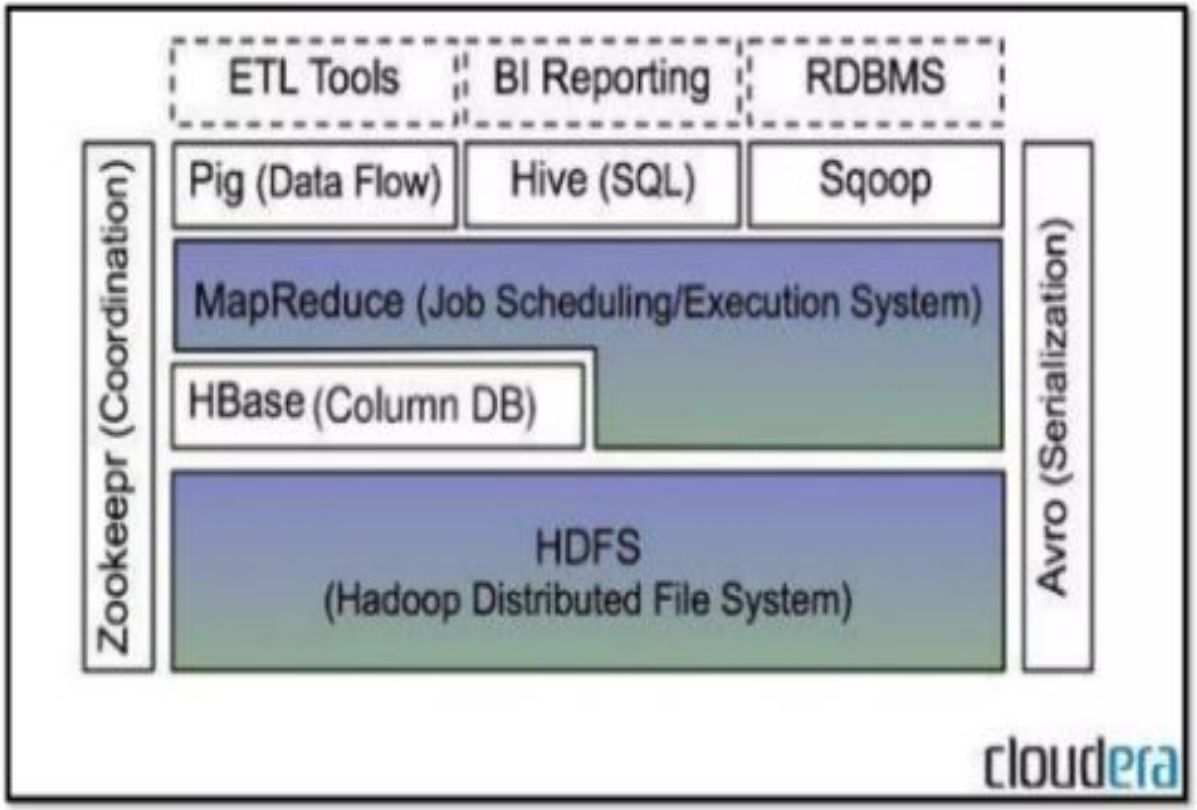
# Famous Hadoop users



# Hadoop Features

- Hadoop provides access to the file systems
- The Hadoop Common package contains the necessary JAR files and scripts
- The package also provides source code, documentation and a contribution section that includes projects from the Hadoop Community.

# HADOOP ARCHITECTURE



# Core-Components of Hadoop:

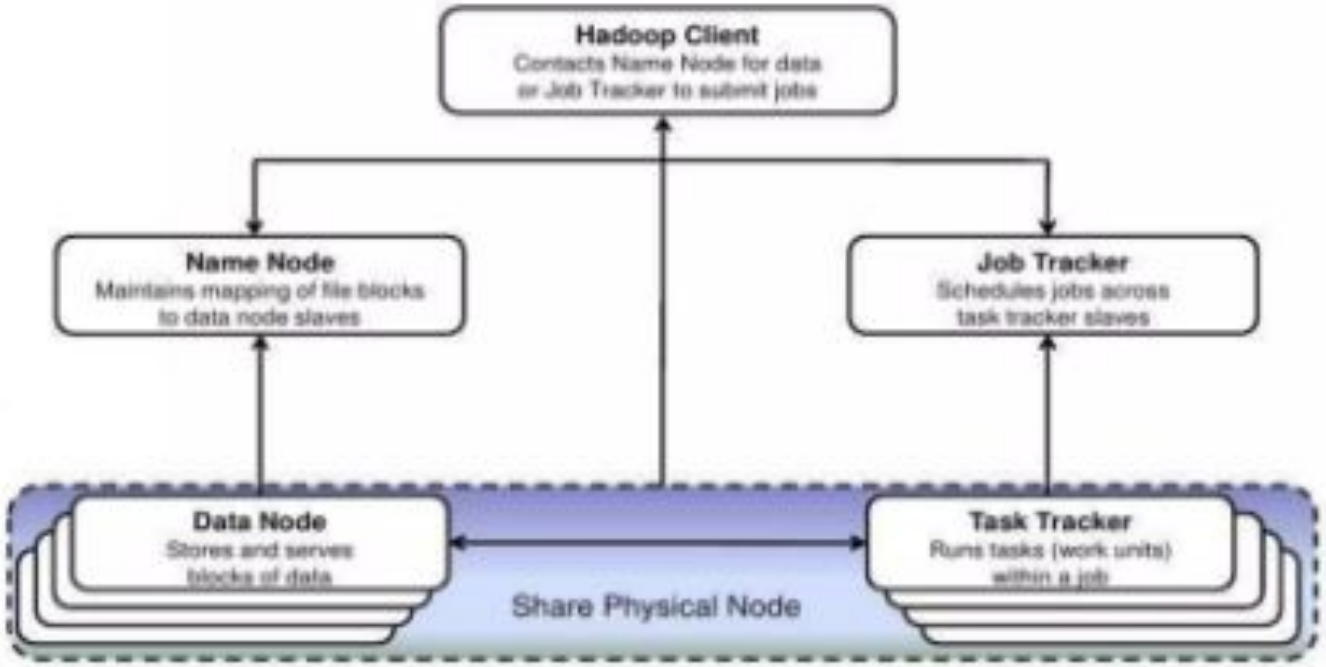

Hadoop distributive file system.


Map reduce.

# What is HDFS ?

- Distributed file system
- Traditional hierarchical file organization
- Single namespace for the entire cluster
- Write-once-read-many access model
- Aware of the network topology

# Hadoop High Level Architechture



# Hadoop cluster

•A Small Hadoop Cluster Include a single master &

multiple worker nodes

Master node:

Data Node

Job Tracker

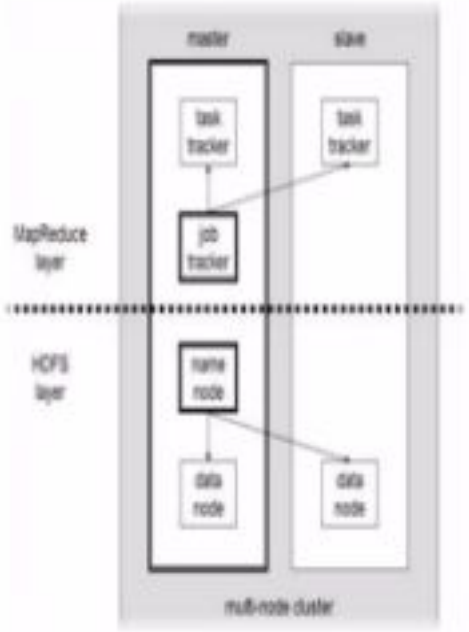
Task Tracker

Name Node

Slave node:

Data Node

Task Tracke

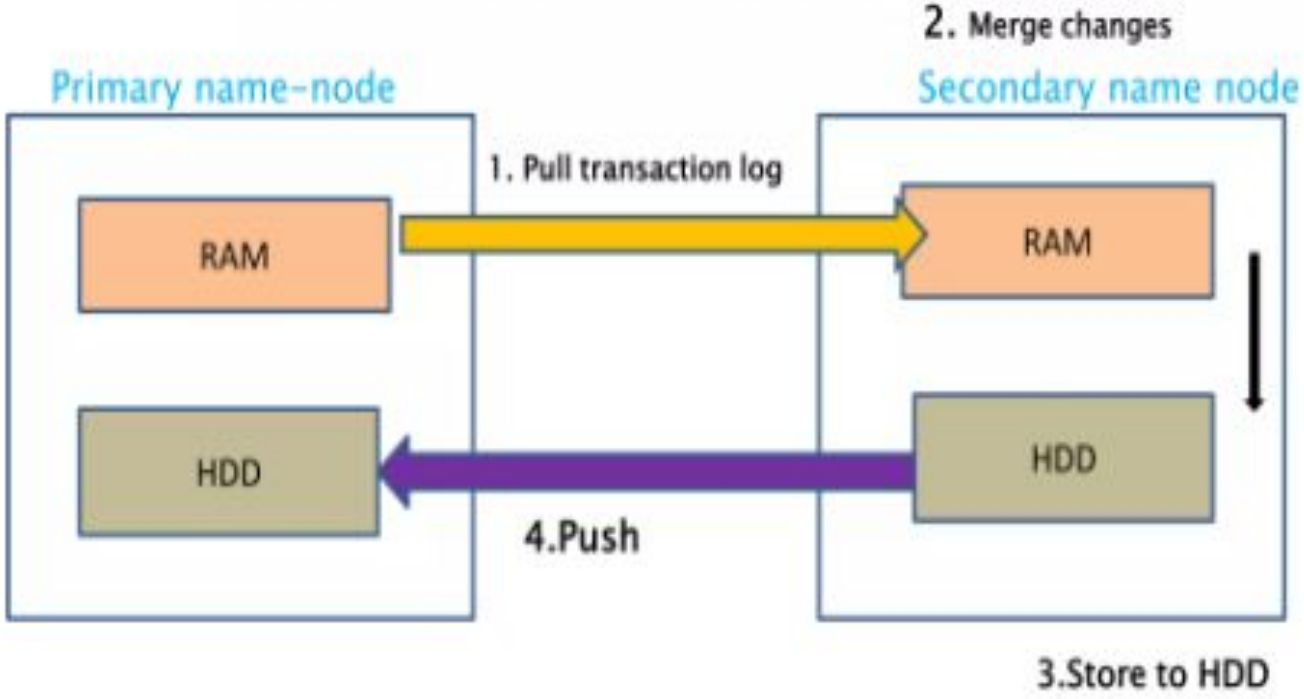


# HDFS – Name Node Features

**Metadata in main memory:**

- List of files
- List of blocks for each file
- List of Data Nodes for each block
- File attributes
- Creation time
- Records every change in the metadata

# HDFS-name node architecture




# HDFS-Data node

- Block Server Stores data in the local file system
- Periodic validation of checksums
- Periodically sends a report of all existing blocks to the Name Node

# Hadoop MAPREDUCE

**Map reduce implementation:**

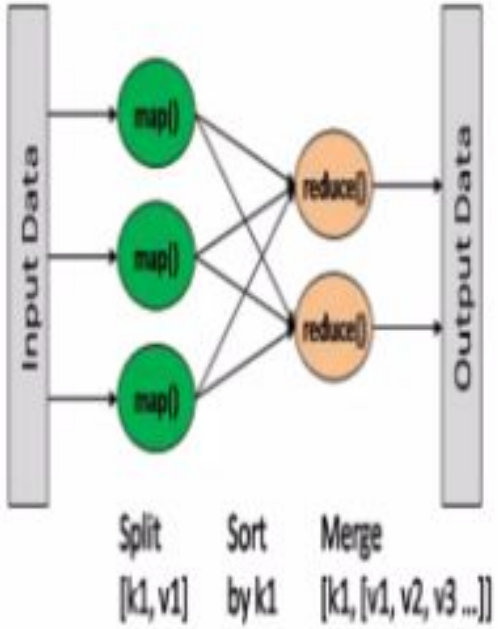
**Job Tracker:**

Splitting into map and reduce tasks

Scheduling tasks on a cluster node

**Task Tracker:**

Runs Map Reduce tasks periodically



# What is Hadoop?

- Hadoop is an open source software programming framework for storing a large amount of data and performing the computation.
- Its framework is based on Java programming with some native code in C and shell scripts.
- Hadoop is used for some advanced level of analytics, which includes Machine Learning and data mining

# Need for Hadoop

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

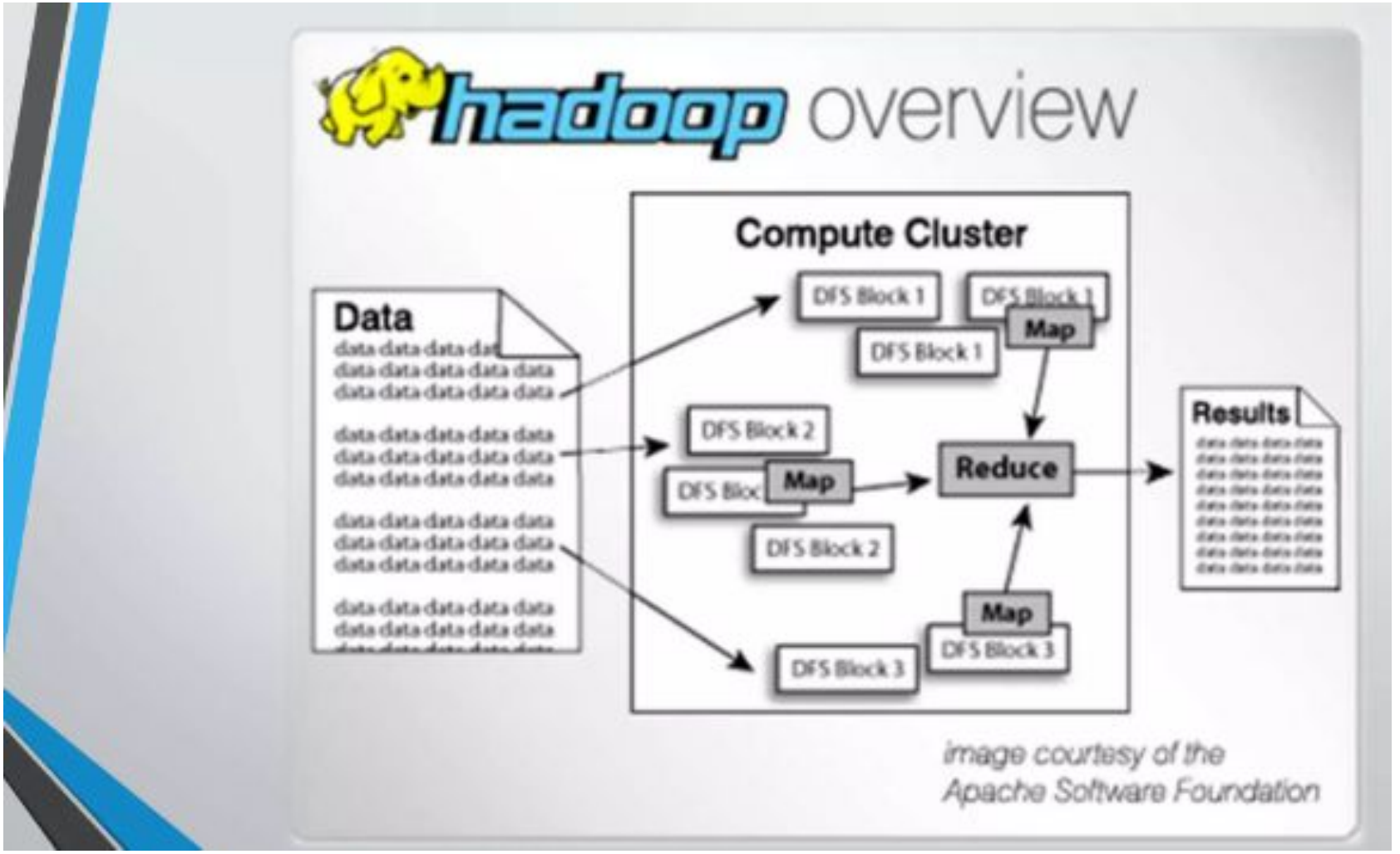
Programmers

*No longer need to worry about*

- Q: Where file is located?
- Q: How to handle failures & data lost?
- Q: How to divide computation?
- Q: How to program for scaling?

# History of Hadoop

- **Apache Software Foundation** is the developers of Hadoop, and it's co-founders are **Doug Cutting** and **Mike Cafarella**.
- It's co-founder Doug Cutting named it on his son's toy elephant. In October 2003 the first paper release was Google File System.
- In January 2006, MapReduce development started on the Apache Nutch which consisted of around 6000 lines coding for it and around 5000 lines coding for HDFS.
- In April 2006 Hadoop 0.1.0 was released.


hadoop overview
Compute Cluster
Data
DFS Block 1
DFS Block 1
DFS Block 1
Map
DFS Block 2
DFS Bloc
Map
DFS Block 2
Reduce
Results
Map
DFS Block 3
DFS Block 3
image courtesy of the
Apache Software Foundation

# Advantages and Disadvantages of Hadoop

**Advantages:**

- Ability to store a large amount of data.
- High flexibility.
- Cost effective.
- High computational power.
- Tasks are independent.
- Linear scaling.

**Disadvantages:**

- Not very effective for small data.
- Hard cluster management.
- Has stability issues.
- Security concerns.

# Hadoop Distributed File System

- It has distributed file system known as HDFS and this HDFS splits files into blocks and sends them across various nodes in form of large clusters.
- Also in case of a node failure, the system operates and data transfer takes place between the nodes which are facilitated by HDFS.

# Comparing: RDBMS vs. Hadoop

| | Traditional RDBMS | Hadoop / MapReduce |
|---|---|---|
| Data Size | Gigabytes (Terabytes) | Petabytes (Hexabytes) |
| Access | Interactive and Batch | Batch – NOT Interactive |
| Updates | Read / Write many times | Write once, Read many times |
| Structure | Static Schema | Dynamic Schema |
| Integrity | High (ACID) | Low |
| Scaling | Nonlinear | Linear |
| Query Response Time | Can be near immediate | Has latency (due to batch processing) |

# Advantages of HDFS:

- It is inexpensive, immutable in nature, stores data reliably, ability to tolerate faults, scalable, block structured, can process a large amount of data simultaneously and many more.

# Disadvantages of HDFS:

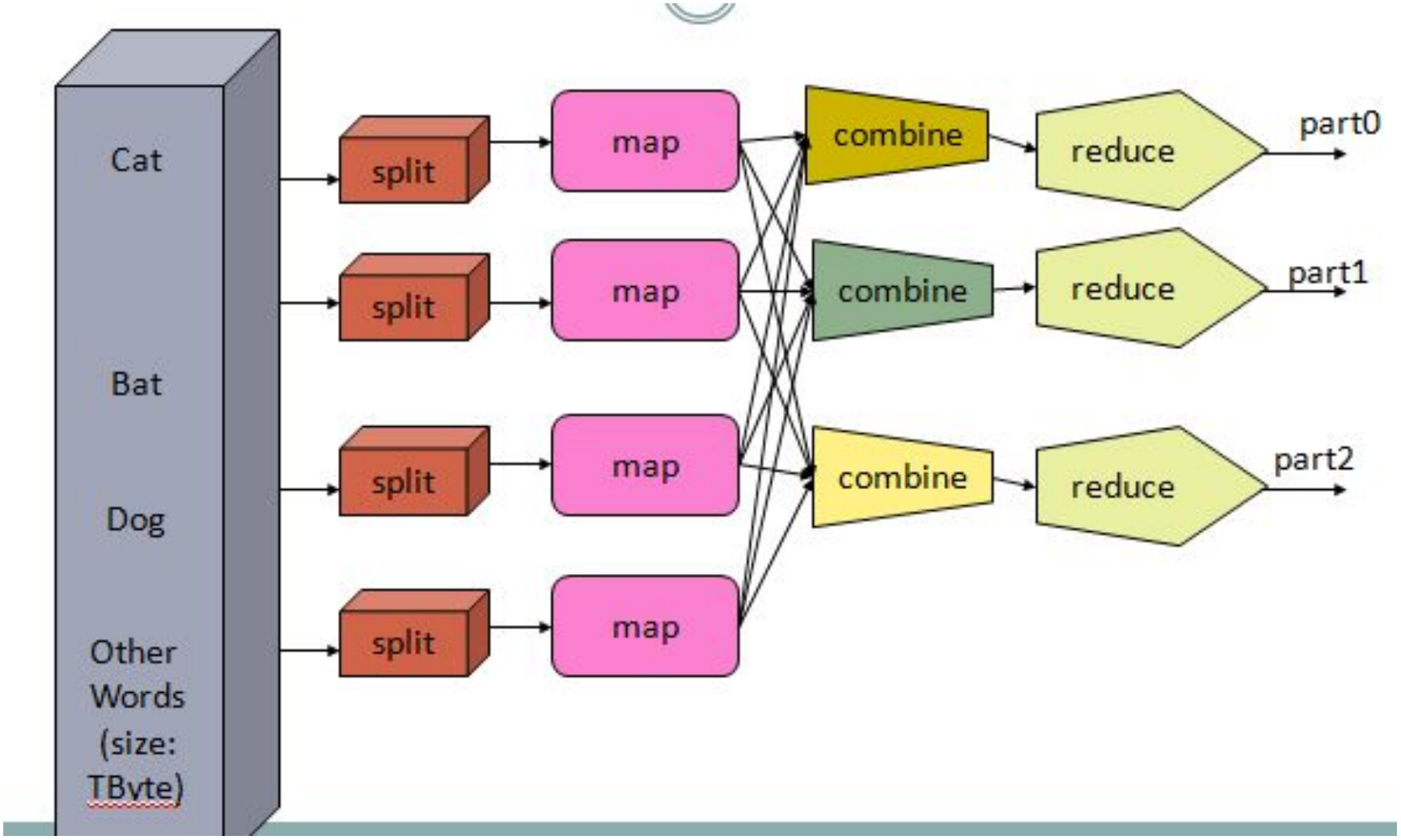
- It's the biggest disadvantage is that it is not fit for small quantities of data. Also, it has issues related to potential stability, restrictive and rough in nature.

# Some common frameworks of Hadoop

- **Hive-** It uses HiveQl for data structuring and for writing complicated MapReduce in HDFS.
- **Drill-** It consists of user-defined functions and is used for data exploration.
- **Storm-** It allows real-time processing and streaming of data.
- **Spark-** It contains a Machine Learning Library(MLlib) for providing enhanced machine learning and is widely used for data processing. It also supports Java, Python, and Scala.
- **Pig-** It has Pig Latin, a SQL-Like language and performs data transformation of unstructured data.
- **Tez-** It reduces the complexities of Hive and Pig and helps in the running of their codes faster.

# Some common frameworks of Hadoop

- **Hive-** It uses HiveQl for data structuring and for writing complicated MapReduce in HDFS.
- **Drill-** It consists of user-defined functions and is used for data exploration.
- **Storm-** It allows real-time processing and streaming of data.
- **Spark-** It contains a Machine Learning Library(MLlib) for providing enhanced machine learning and is widely used for data processing. It also supports Java, Python, and Scala.
- **Pig-** It has Pig Latin, a SQL-Like language and performs data transformation of unstructured data.
- **Tez-** It reduces the complexities of Hive and Pig and helps in the running of their codes faster.


Cat
Bat
Dog
Other Words (size: TByte)
split
split
split
split
map
map
map
map
combine
combine
combine
reduce
reduce
reduce
part0
part1
part2

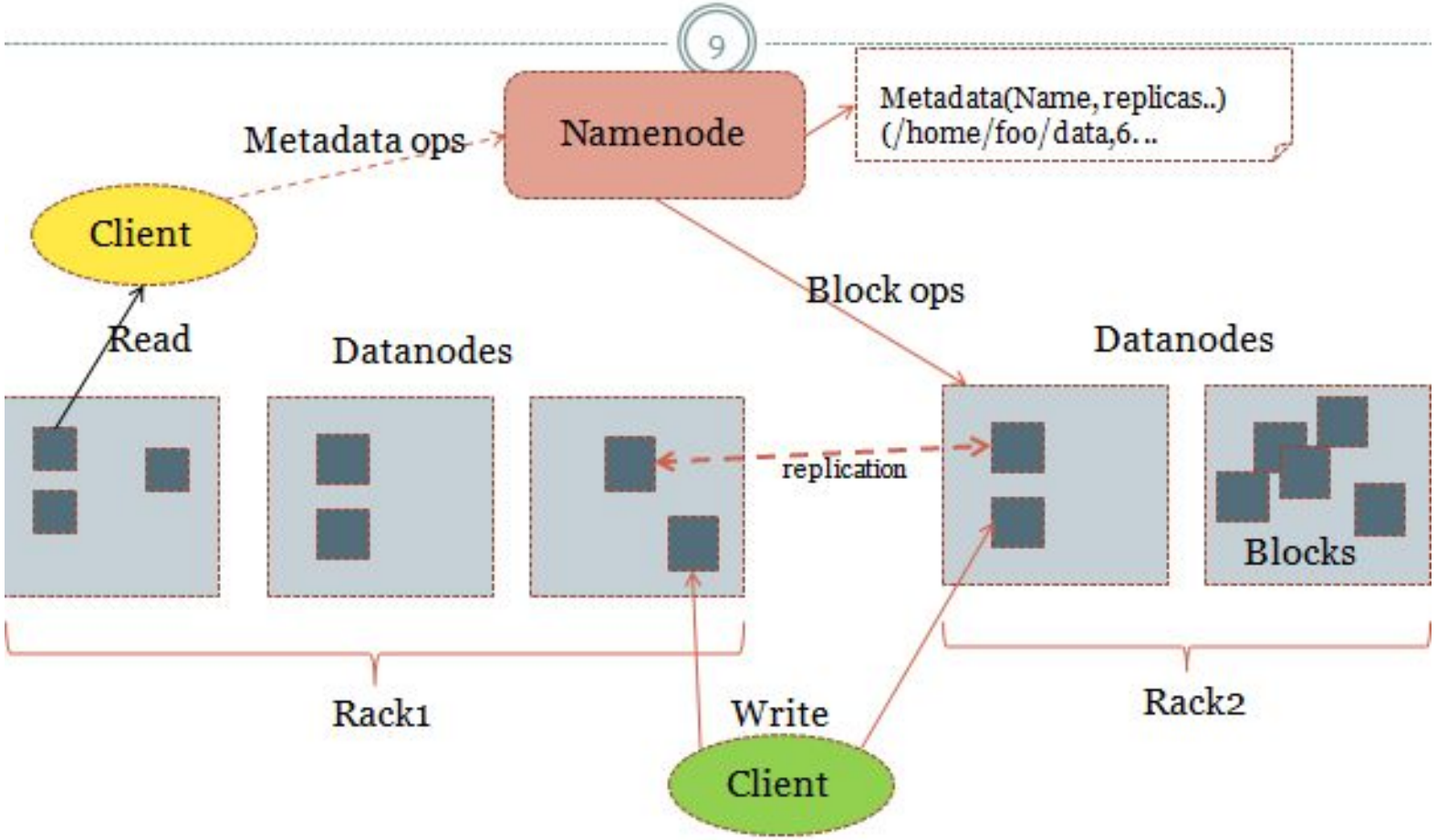
# Namenode and Datanodes



- Master/slave architecture
- HDFS cluster consists of a single **Namenode**, a master server that manages the file system namespace and regulates access to files by clients.
- There are a number of **DataNodes** usually one per node in a cluster.
- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

# HDFS Architecture





# Datanode



- A Datanode stores data in files in its local file system.
- Datanode has no knowledge about HDFS filesystem
- It stores each block of HDFS data in a separate file.
- Datanode does not create all files in the same directory.
- It uses heuristics to determine optimal number of files per directory and creates directories appropriately:
  - Research issue?
- When the filesystem starts up it generates a list of all HDFS blocks and send this report to Namenode: Blockreport.

# The Communication Protocol



- All HDFS communication protocols are layered on top of the TCP/IP protocol
- A client establishes a connection to a configurable TCP port on the Namenode machine. It talks ClientProtocol with the Namenode.
- The Datanodes talk to the Namenode using Datanode protocol.
- RPC abstraction wraps both ClientProtocol and Datanode protocol.
- Namenode is simply a server and never initiates a request; it only responds to RPC requests issued by DataNodes or clients.

# Hadoop Architecture

Hadoop framework includes following four modules:

- Hadoop Common: These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- Hadoop YARN: This is a framework for job scheduling and cluster resource management.
- Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.
- Hadoop MapReduce: This is YARN-based system for parallel processing of large data sets.

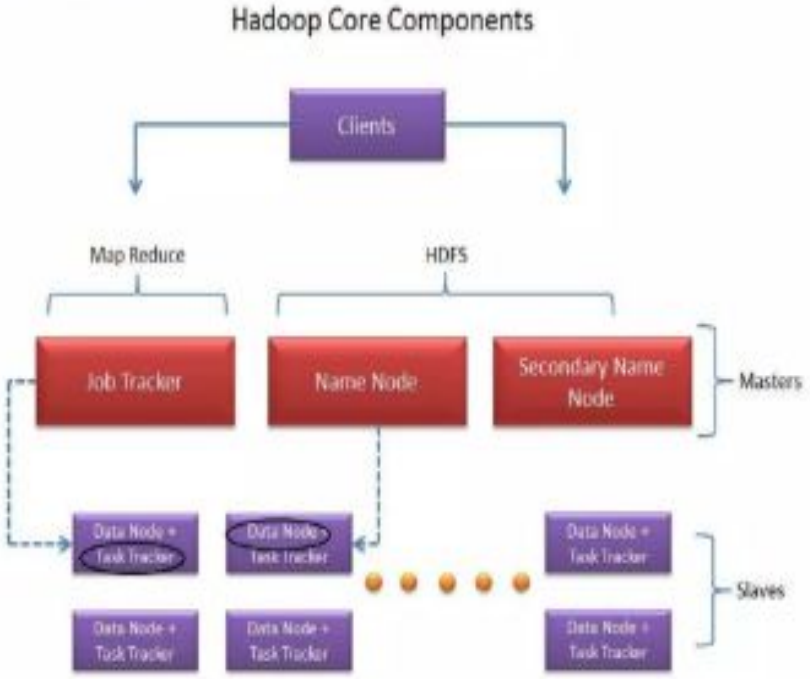
# Hadoop Main Components

- Hadoop consists of MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper. MapReduce and Hadoop distributed file system (HDFS) are the main component of Hadoop.

# Core Components of Hadoop Cluster

**Hadoop cluster has 3 components:**

- Client
- Master
- Slave



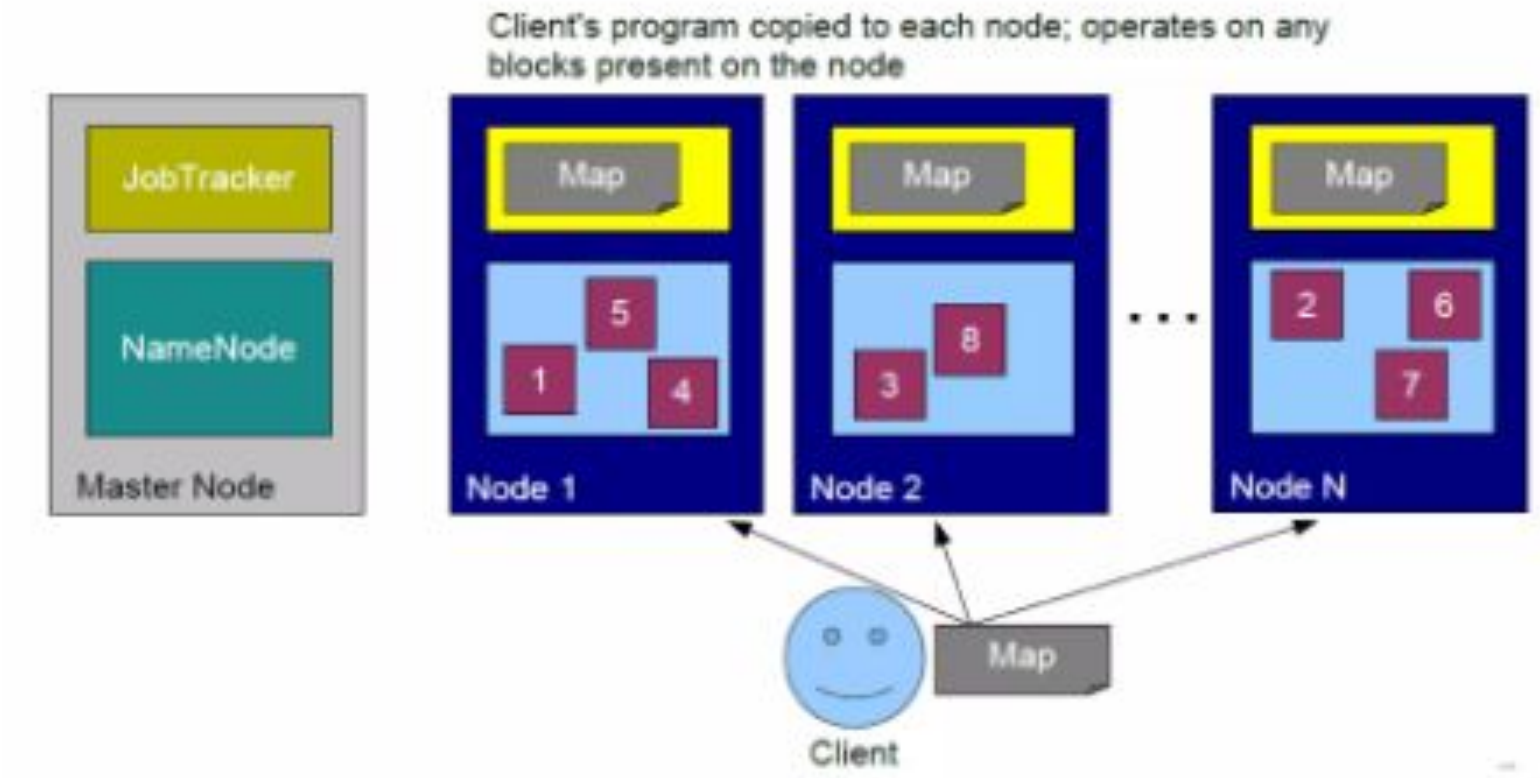
# Task Tracker

1. Each Task Tracker is responsible to execute and manage the individual tasks assigned by Job Tracker.
2. Task Tracker also handles the data motion between the map and reduce phases.
3. One Prime responsibility of Task Tracker is to constantly communicate with the Job Tracker the status of the Task.
4. If the JobTracker fails to receive a heartbeat from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster.

# HDFS & MapReduce

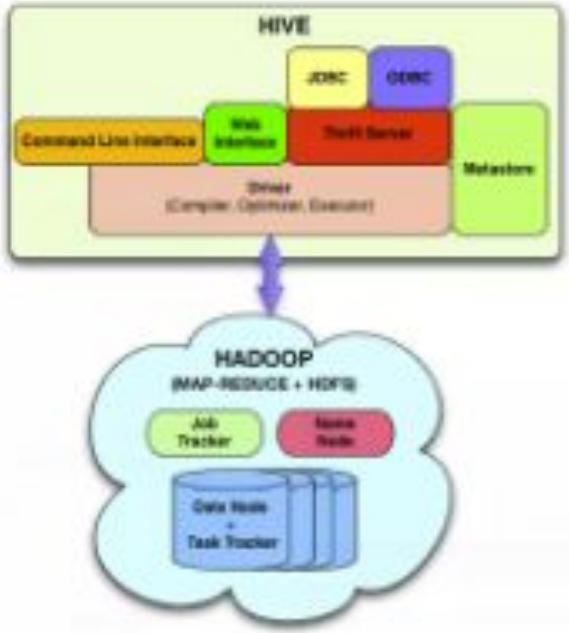
- **Hadoop Distributed File System**
  - A scalable, Fault tolerant, High performance distributed file system
  - Asynchronous replication
  - Write-once and read-many (WORM)
  - Hadoop cluster with 3 DataNodes minimum
  - Data divided into 64MB (default) or 128MB blocks, each block replicated 3 times (default)
  - No RAID required for DataNode
  - Interfaces: API, Thrift, C Library, FUSE, WebDAV, HTTP, FTP
  - **NameNode** holds filesystem metadata
  - Files are broken up and spread over the **DataNodes**
- **Hadoop Map Reduce**
  - Software framework for distributed computation
  - Input | Map() | Copy/Sort | Reduce() | Output
  - **JobTracker** schedules and manages jobs
  - **TaskTracker** executes individual map() and reduce() tasks on each cluster node

# MapReduce


Client's program copied to each node; operates on any blocks present on the node
JobTracker
NameNode
Master Node
Map
5
1
4
Node 1
Map
8
3
Node 2
Map
2
6
7
Node N
Map
Client

# Hive


- Data Warehouse infrastructure that provides data summarization and ad hoc querying on top of Hadoop
  - MapReduce for execution
  - HDFS for storage
- MetaStore
  - Table/Partitions properties
  - Thrift API : Current clients in Php (Web Interface), Python interface to Hive, Java (Query Engine and CLI)
  - Metadata stored in any SQL backend
- Hive Query Language
  - Basic SQL : Select, From, Join, Group By
  - Equi-Join, Multi-Table Insert, Multi-Group-By
  - Batch query



# Pig

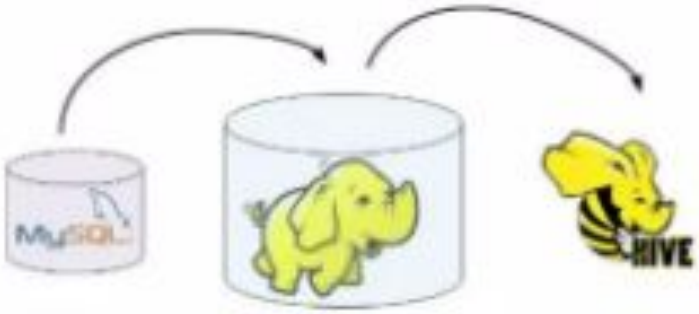
- A high-level data-flow language and execution framework for parallel computation
- Simple to write MapReduce program
- Abstracts you from specific detail
- Focus on data processing
- Data flow
- For data manipulation

PIG Language Example

```
Users = LOAD 'users' AS (name, age);
Fltrd = FILTER Users BY
        age >= 18 AND age <= 25;
Pages = LOAD 'pages' AS (user, url);
Jnd = JOIN Fltrd BY name, Pages BY user;
Grpd = GROUP Jnd BY url;
Smmd = FOREACH Grpd GENERATE group,
       COUNT(Jnd) AS clicks;
Srtd = ORDER Smmd BY clicks DESC;
Top5 = LIMIT Srtd 5;
STORE Top5 INTO 'top5sites';
```

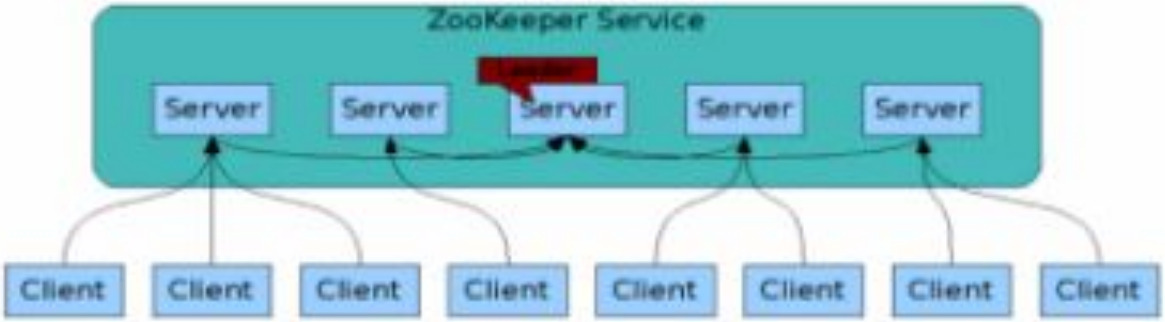
# Sqoop

- Sqoop is a tool designed to help users of large data import existing relational databases into their Hadoop clusters
- Automatic data import
- SQL to Hadoop
- Easy import data from many databases to Hadoop
- Generates code for use in MapReduce applications
- Integrates with Hive



# Zookeeper

- A high-performance coordination service for distributed applications
- ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services



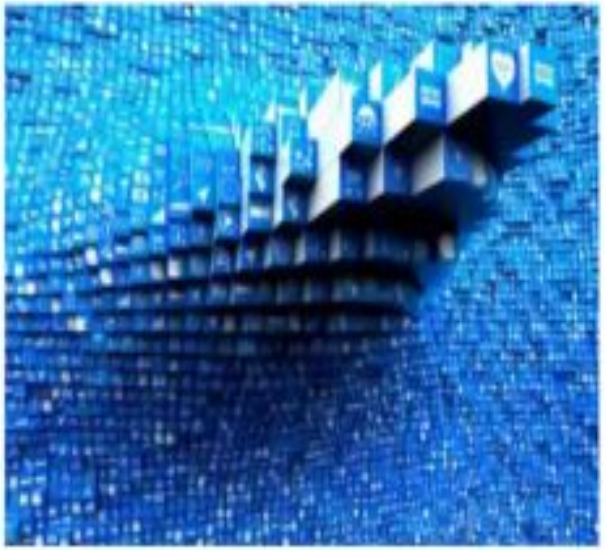

- All servers store a copy of the data
- A leader is elected at startup
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change

# Avro


- A data serialization system that provides dynamic integration with scripting languages
- Avro Data
  - Expressive
  - Smaller and Faster
  - Dynamic
    - Schema store with data
    - APIs permit reading and creating
  - Include a file format and a textual encoding
- Avro RPC
  - Leverage versioning support
  - For Hadoop service provide cross-language access

# What is big data Analytics?

Big data Analytics is a process to extract meaningful insight from big such as hidden patterns, unknown correlations, market trends and customer preferences



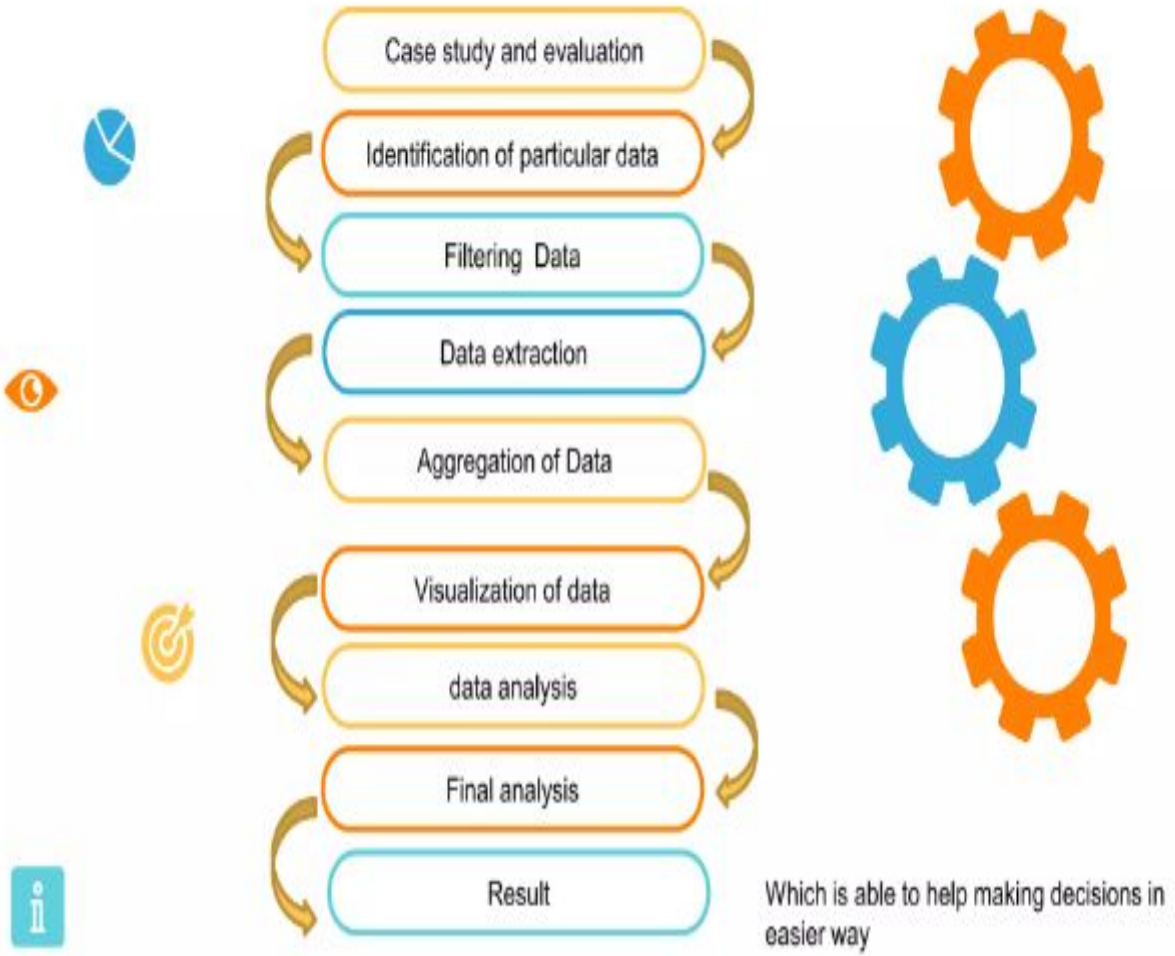
# Types of Big Data Analytics

# Semi-structured Data


***Semi-Structured Data*** refers to the data that does not have a proper structure associated with it. For example, the data that is present within the emails, the log files, and the word documents can be referred to as Semi-Structured Data.

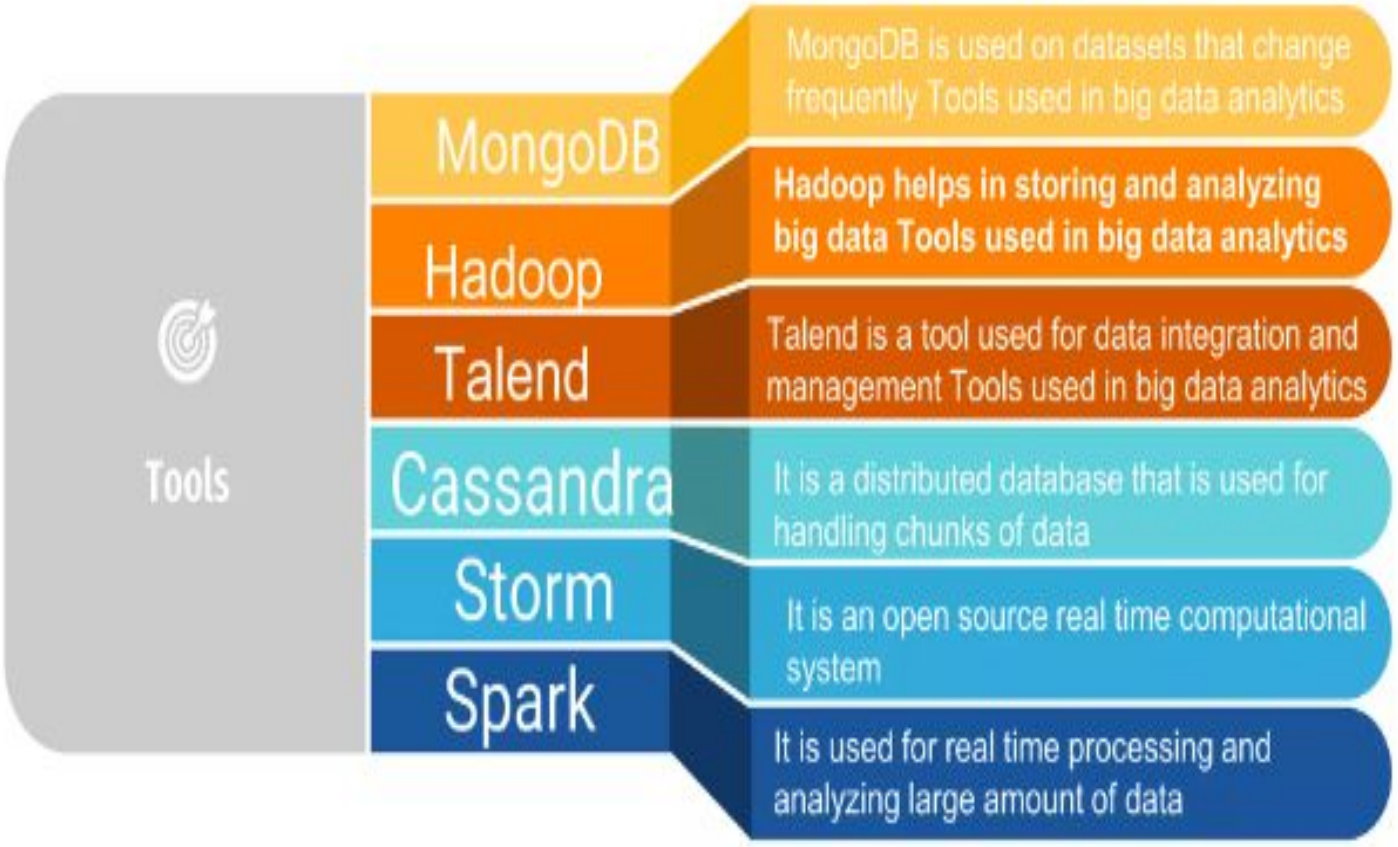
```
<rec><name>Prashant Rao</name><sex>Male</sex><age>35</age></rec>
<rec><name>Seema R.</name><sex>Female</sex><age>41</age></rec>
<rec><name>Satish Mane</name><sex>Male</sex><age>29</age></rec>
<rec><name>Subrato Roy</name><sex>Male</sex><age>26</age></rec>
<rec><name>Jeremiah J.</name><sex>Male</sex><age>35</age></rec>
```

# Process of Big Data Analytics




Which is able to help making decisions in easier way

# Tools used in Big Data Analytics



# Benefits Big Data Analytics

Big data analytics is used for risk management

Big data analytics is used to improve customer experience

Big data analytics is used for product development and innovations

Big data analytics helps in quicker and better decision making in organizations

Google has mastered the domain of big data analytics and it has developed several tools and techniques to capture the data of users which includes their preference, their likes, dislikes, the area of specialization, their requirement etc.

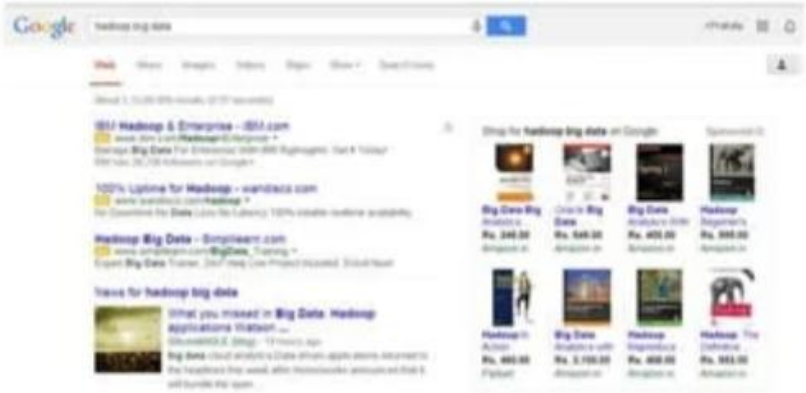
# Structured Data


**Structured Data** refers to the data that has a proper structure associated with it. For example, the data that is present within the databases, the CSV files, and the excel spreadsheets can be referred to as Structured Data.

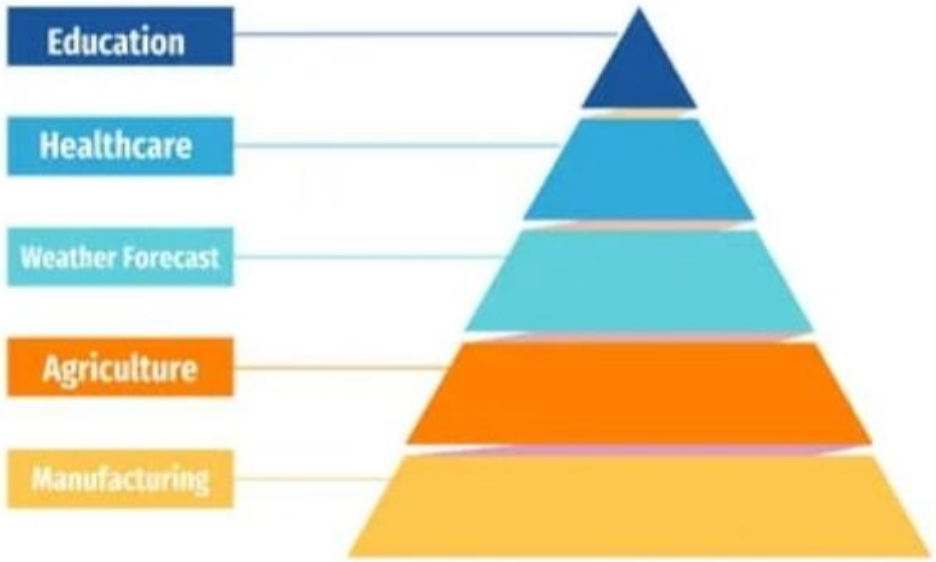
| Employee_ID | Employee_Name | Gender | Department | Salary_In_lacs |
|---|---|---|---|---|
| 2365 | Rajesh Kulkarni | Male | Finance | 650000 |
| 3398 | Pratibha Joshi | Female | Admin | 650000 |
| 7465 | Shushil Roy | Male | Admin | 500000 |
| 7500 | Shubhojit Das | Male | Finance | 500000 |
| 7699 | Priya Sane | Female | Finance | 550000 |

# Unstructured Data


***Un-Structured Data*** refers to the data that does not have any structure associated with it at all. For example, the image files, the audio files, and the video files can be referred to as Un-Structured Data.



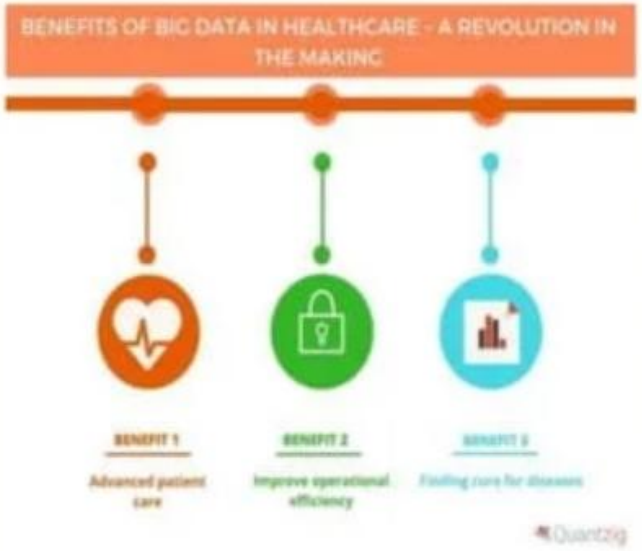
# Big Data application domains



# Education


- Enhancing Student Results
- A better Grading System
- Gaining Attention
- Customized Programs
- Reducing The Number of Dropouts

# HealthCare




- Health Tracking
- Improve the care delivery system / machinery
- Fraud detection and prevention
- Real-time alerts

# Weather Forecast

- Estimates of areas where flooding is likely to be most severe
- The strength and direction of tropical storms
- The most likely amount of snow or rain that will fall in a specific area
- The most likely locations of downed power lines
- Estimates of areas where wind speeds are likely to be greatest
- The locations where bridges and roads most likely to be damaged by storms
- The likelihood of flights being cancelled at specific airports

# Agriculture


- Boosting productivity
- Access to plant genome information
- Predicting yields
- Risk management
- Food safety
- Savings

# Manufacturing


- Product quality
- Defects tracking
- Supply planning
- Manufacturing process defect tracking
- Testing and simulation of new manufacturing processes

# Google Big Data

Google developed several open source tools and techniques that are extensively used in big data ecosystem. With the help of different big data tools and techniques, Google is now capable of exploring millions of websites and **fetch you the right answer or information within milliseconds.**

# Google Big Data

The first question that comes to our mind is how can Google perform such complex operations so efficiently?

The answer is Big data analytics.

Big Data tools and techniques to understand our requirements based on several parameters like search history, locations, trends etc.
Google effortlessly displays the complex calculations which are designed to match the user's requirement.

Google always wanted to develop a search engine that has the ability to think like a human and understand the phrase, logic, and goal of any search query. Semantics has helped Google to accomplish this task to look beyond the literal meaning of any phrase of a search query.

# Google Has adopted the following techniques using Big Data Analytics

# IBM's Weather Forecasting

One example of an application of big data to weather forecasting is IBM's Deep Thunder. Unlike many weather forecasting systems, which give general information about a broad geographical region, Deep Thunder provides forecasts for extremely specific locations, such as a single airport, so that local authorities can get critically important information in real time.


facebook

## New York Stock Exchange

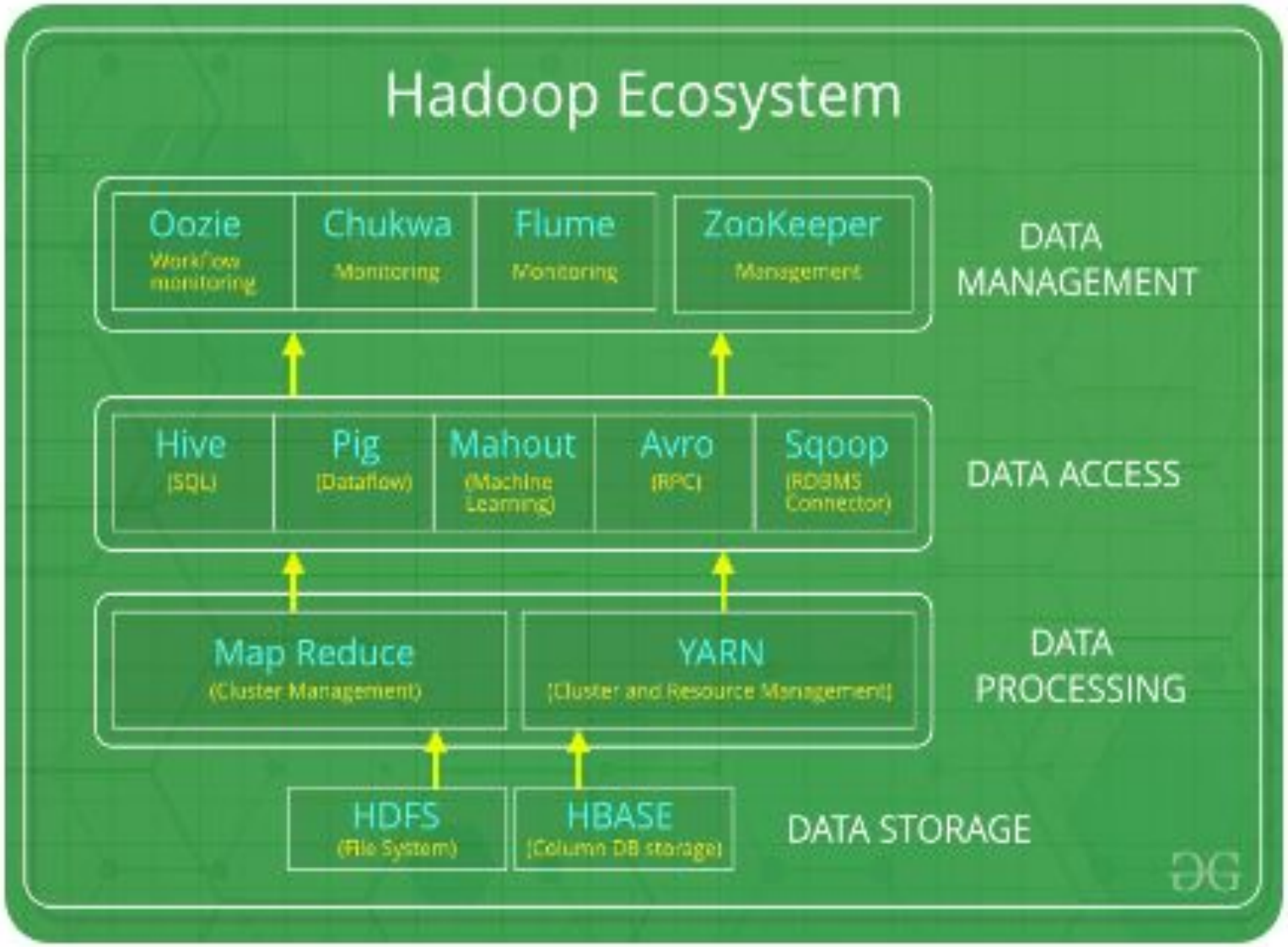
The New York Stock Exchange generates about one terabyte of new trade data per day.

## Social Media

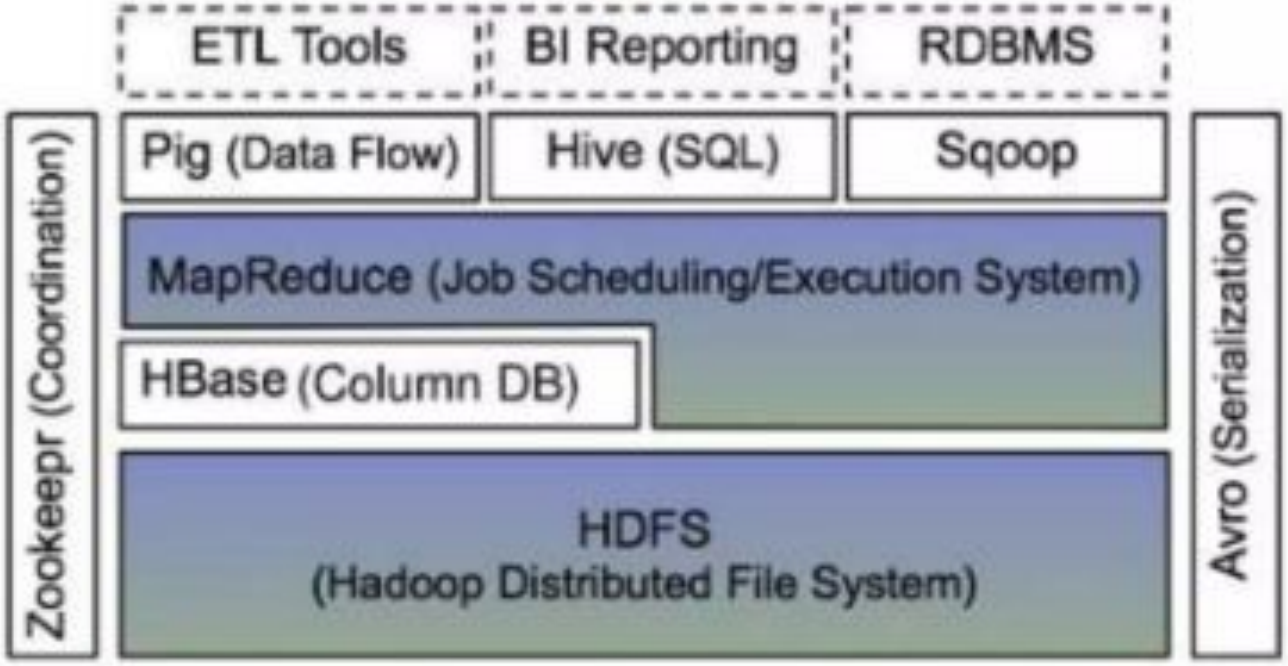
500TB+ data is generated everyday only on Facebook .This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.

## Air Industry

A single Jet engine can generate 10+terabytes of data in 30 minutes of flight time. With many thousand flights per day, generation of data reaches up to many Petabytes.


Hadoop Ecosystem
Oozie
Workflow monitoring
Chukwa
Monitoring
Flume
Monitoring
ZooKeeper
Management
DATA MANAGEMENT
Hive
(SQL)
Pig
(Dataflow)
Mahout
(Machine Learning)
Avro
(RPC)
Sqoop
(RDBMS Connector)
DATA ACCESS
Map Reduce
(Cluster Management)
YARN
(Cluster and Resource Management)
DATA PROCESSING
HDFS
(File System)
HBASE
(Column DB storage)
DATA STORAGE

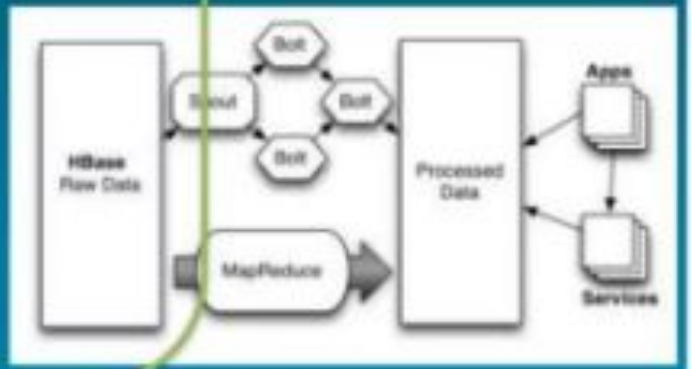
# The Hadoop Ecosystem



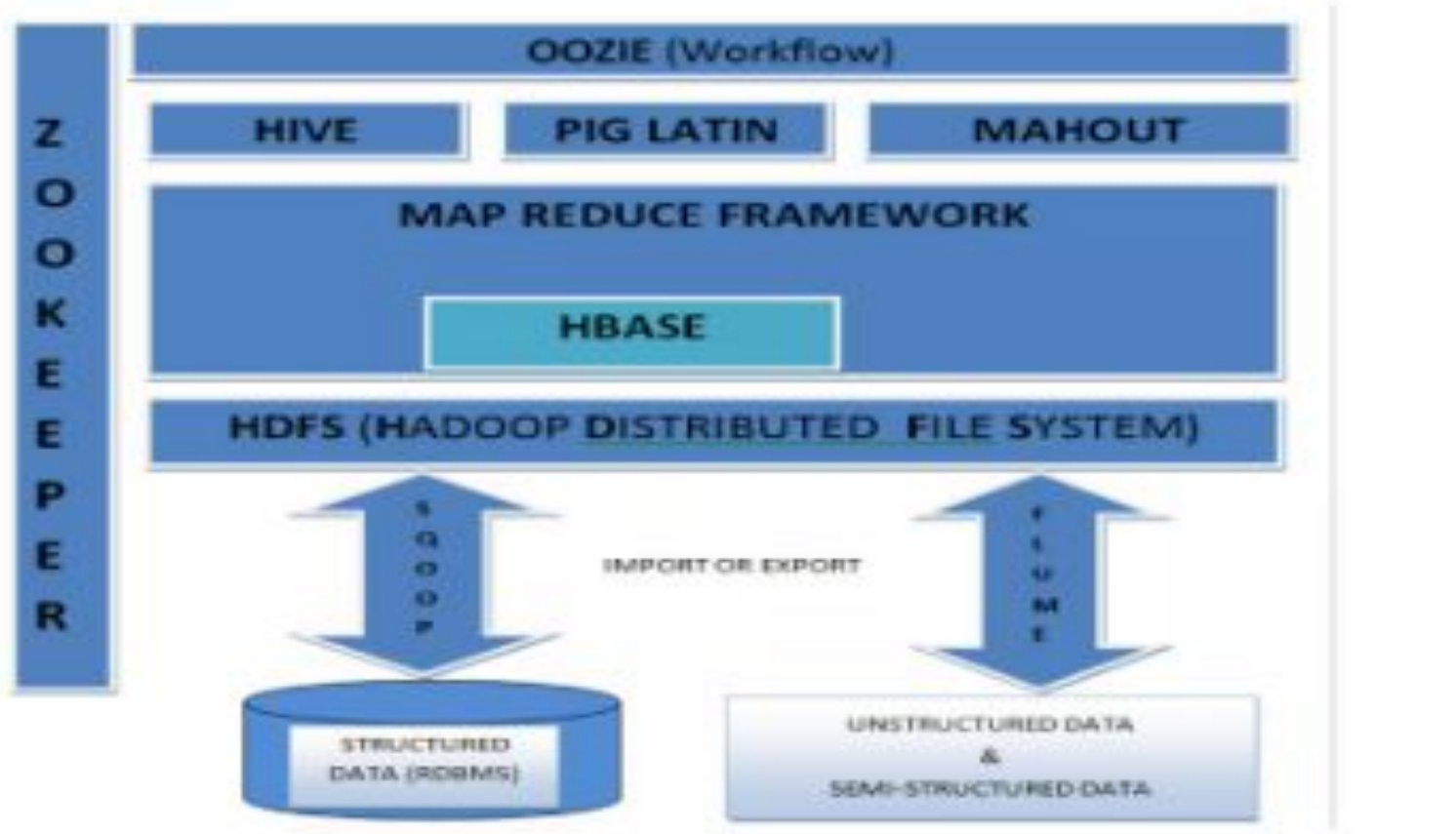
# Roles in Hadoop Ecosystem

- MapReduce for indexing, learning
- HBase for storage and fast access
- Also: Storm for incremental update
- And: relational DB for most recent derived data
- API façade for input; API for querying learning

Machine Learning



# Working Of Ecosystem



**Introduction:** *Hadoop Ecosystem* is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are *four major elements of Hadoop* i.e. **HDFS, MapReduce, YARN, and Hadoop Common**. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem:

Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

## Apache HBase:

- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data

**HDFS:**

- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
  1. Name node
  2. Data Node

- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

**YARN:**

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
  1. Resource Manager
  2. Nodes Manager
  3. Application Manager

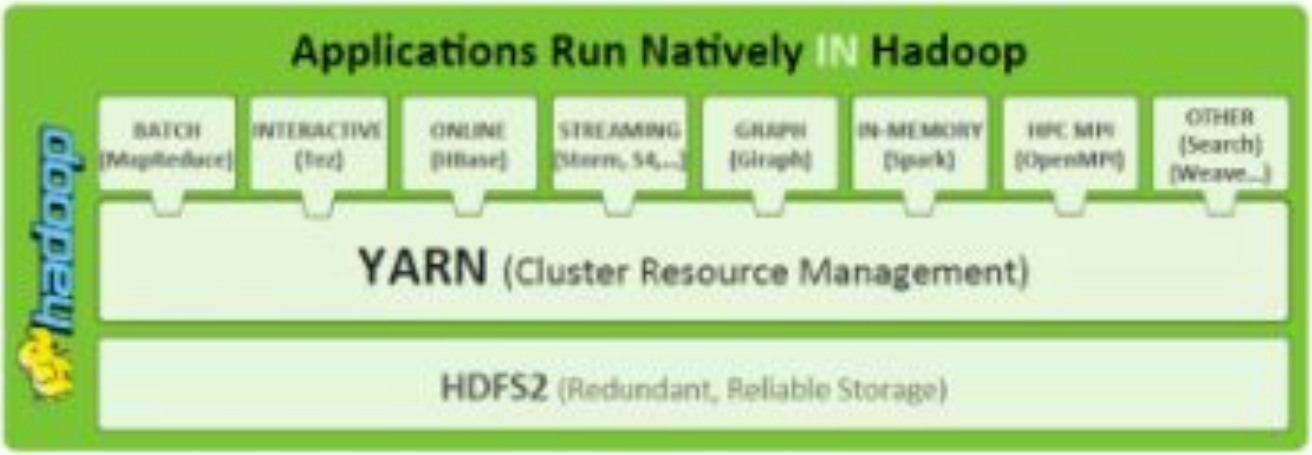
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

# YARN: Taking Hadoop Beyond Batch

**Store ALL DATA in one place…**

**Interact with that data in MULTIPLE WAYS**

**with Predictable Performance and Quality of Service**



# Hadoop 1 Limitations

**Scalability**

- Max Cluster size ~5,000 nodes
- Max concurrent tasks ~40,000
- Coarse Synchronization in JobTracker

**Availability**

- Failure Kills Queued & Running Jobs

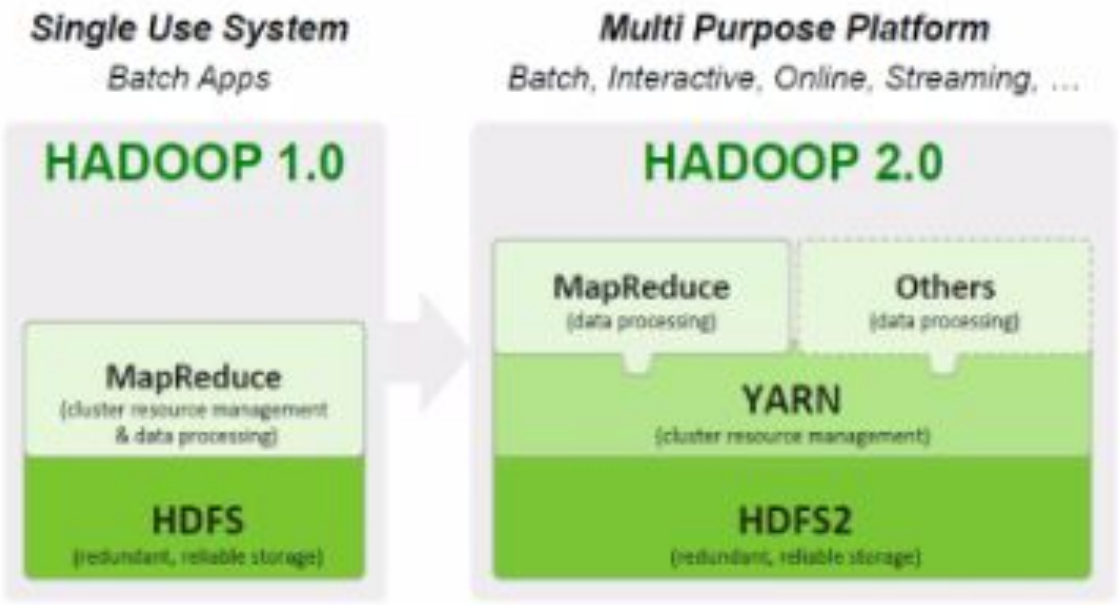
**Hard partition of resources into map and reduce slots**

- Non-optimal Resource Utilization

**Lacks Support for Alternate Paradigms and Services**

- Iterative applications in MapReduce are 10x slower

# Our Vision: Hadoop as Next-Gen Platform



MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
  1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
  2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

## PIG:

Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.

- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.
- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.
- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

## HIVE:

- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: *JDBC Drivers* and *HIVE Command Line*.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

## Apache Spark:

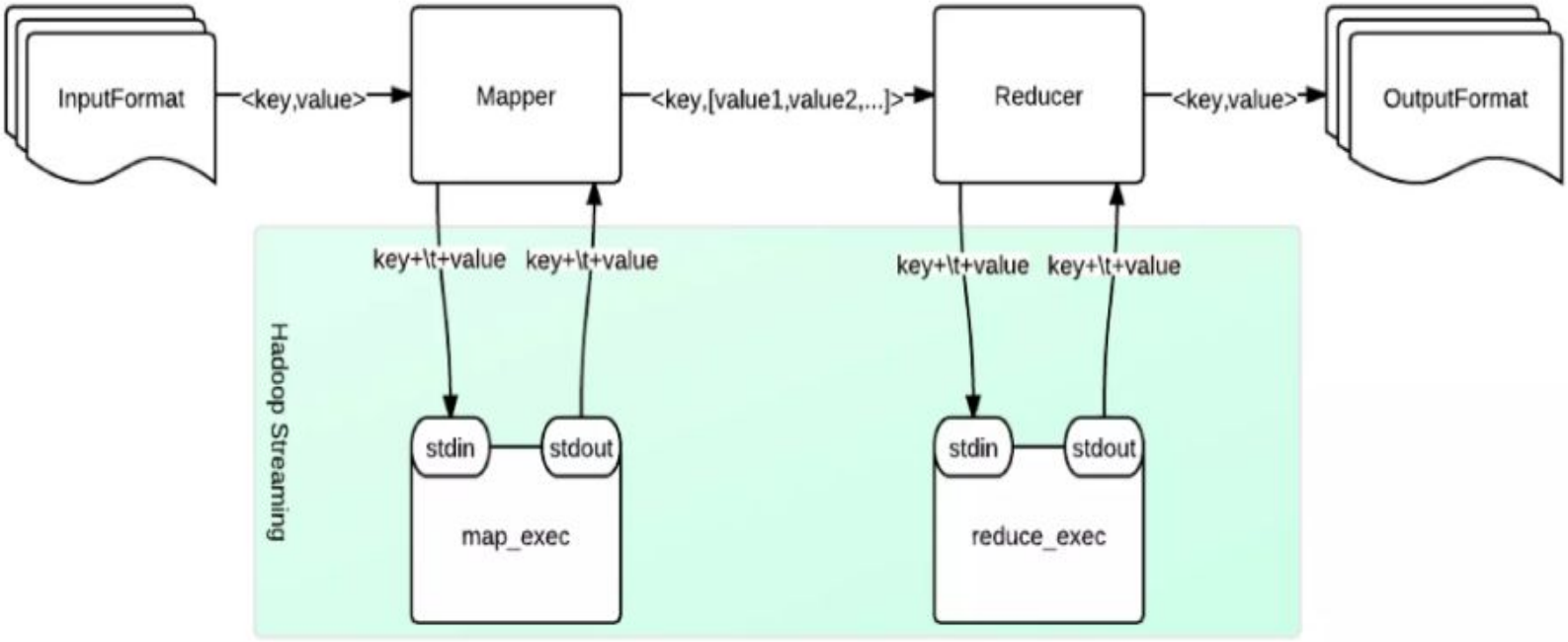
- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

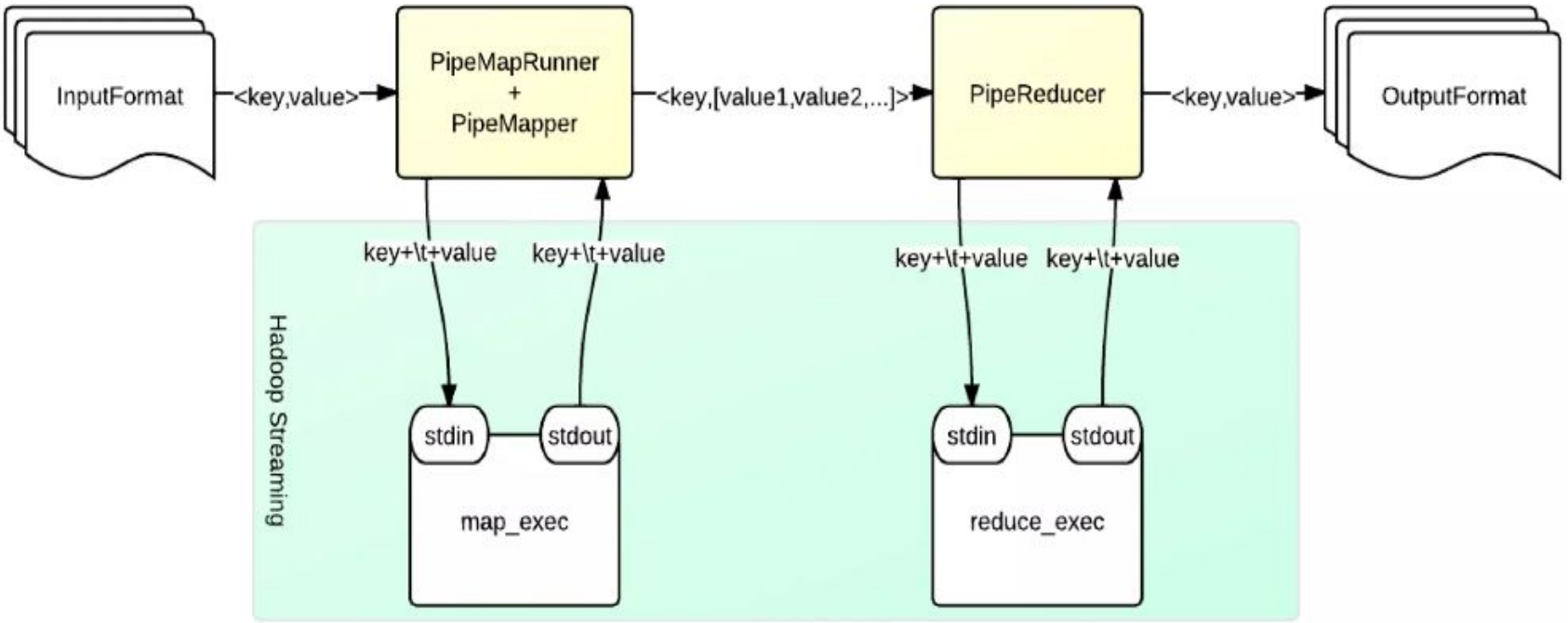
## Apache Spark:

- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

- **Zookeeper:** There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie:** Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

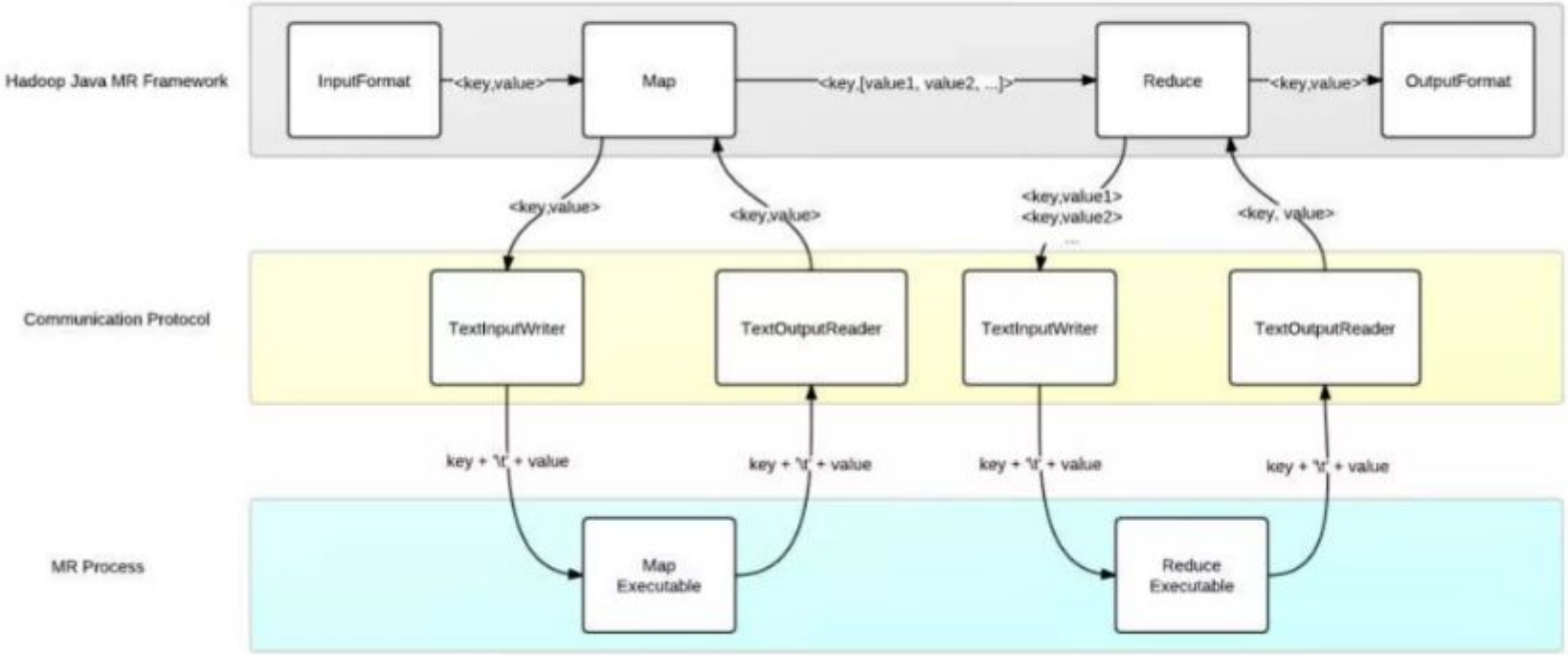
# HADOOP PIPES




InputFormat
<key,value>
PipeMapRunner
+
PipeMapper
<key,[value1,value2,...]>
PipeReducer
<key,value>
OutputFormat
key+\t+value
key+\t+value
key+\t+value
key+\t+value
Hadoop Streaming
stdin
stdout
map_exec
stdin
stdout
reduce_exec

# Streaming Data Flow

HANBORQ



# Streaming I/O Format

- -inputFormat <javaClassName>
  - JobConf.setInputFormat()
- -outputFormat <javaClassName>

# Hadoop Streaming :



- It is a feature that comes with a Hadoop distribution that allows developers or programmers to write the Map-Reduce program using different programming languages like Ruby, Perl, Python, C++, etc.
- We can use any language that can read from the standard input(STDIN) like keyboard input and all and write using standard output(STDOUT).
- Although Hadoop Framework is completely written in java programs for Hadoop do not necessarily need to code in Java programming language.

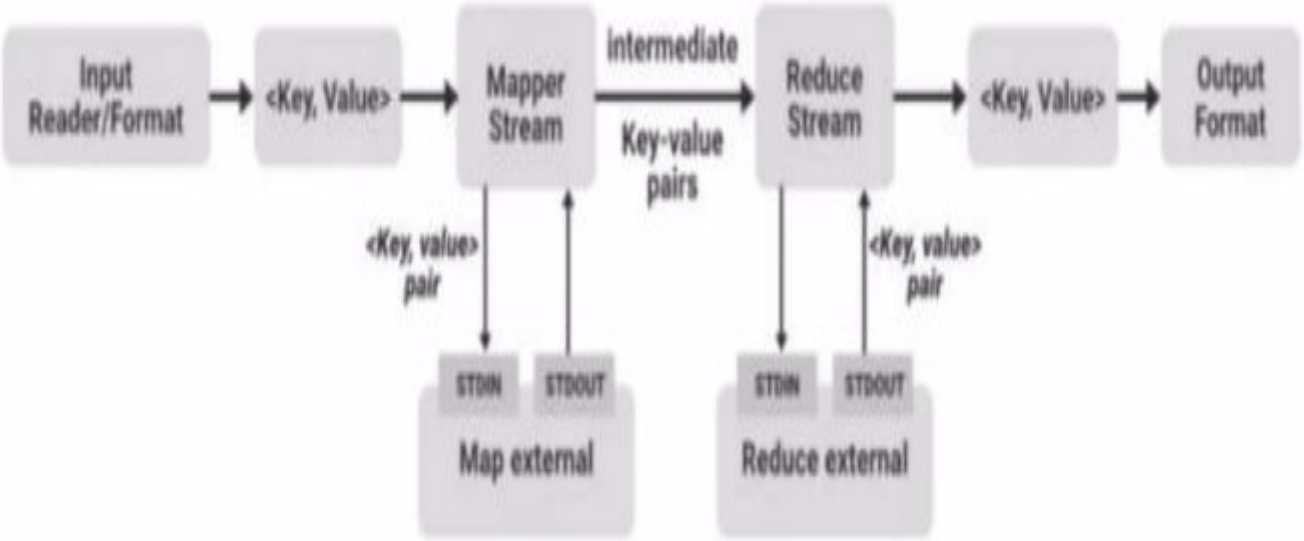
.

# Hadoop Streaming :



- The main advantage of the Hadoop streaming utility is that it allows Java as well as non-Java programmed MapReduce jobs to be executed over Hadoop clusters.
- The Hadoop streaming supports the Perl, Python, PHP, R, and C++ Translate the application logic into the Mapper and Reducer sections with the key and value output elements.
- Three main components: Mapper, Reducer, and Driver.

# Hadoop Streaming

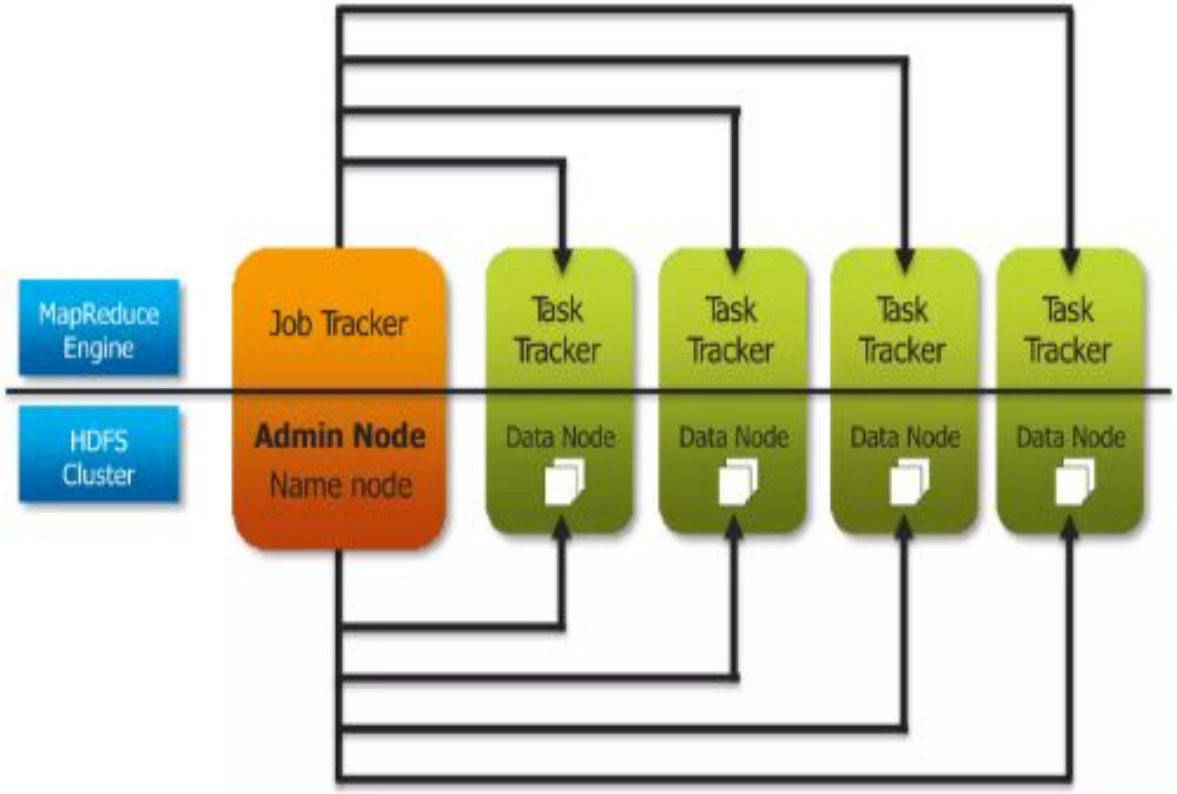


**MAP REDUCE**

# Map Reduce

- HDFS handles the Distributed Filesystem layer
- MapReduce is a programming model for data processing.
- MapReduce
  - Framework for parallel computing
  - Programmers get simple API
  - Don't have to worry about handling
    - parallelization
    - data distribution
    - load balancing
    - fault tolerance
    - Allows one to process huge amounts of data (terabytes and petabytes) on thousands of processors

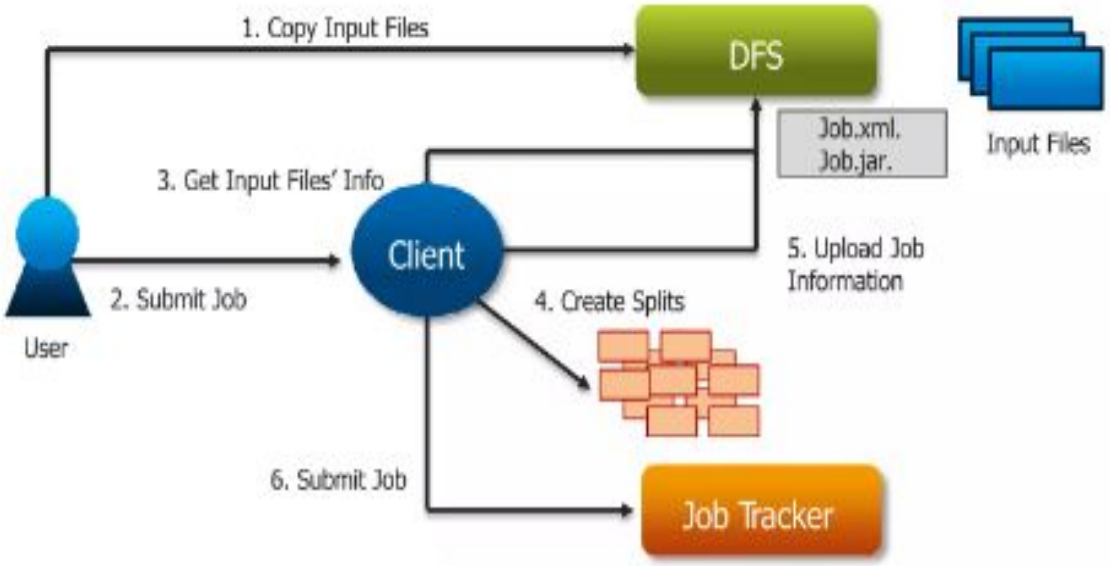
# Map Reduce Concepts (Hadoop-1.0)



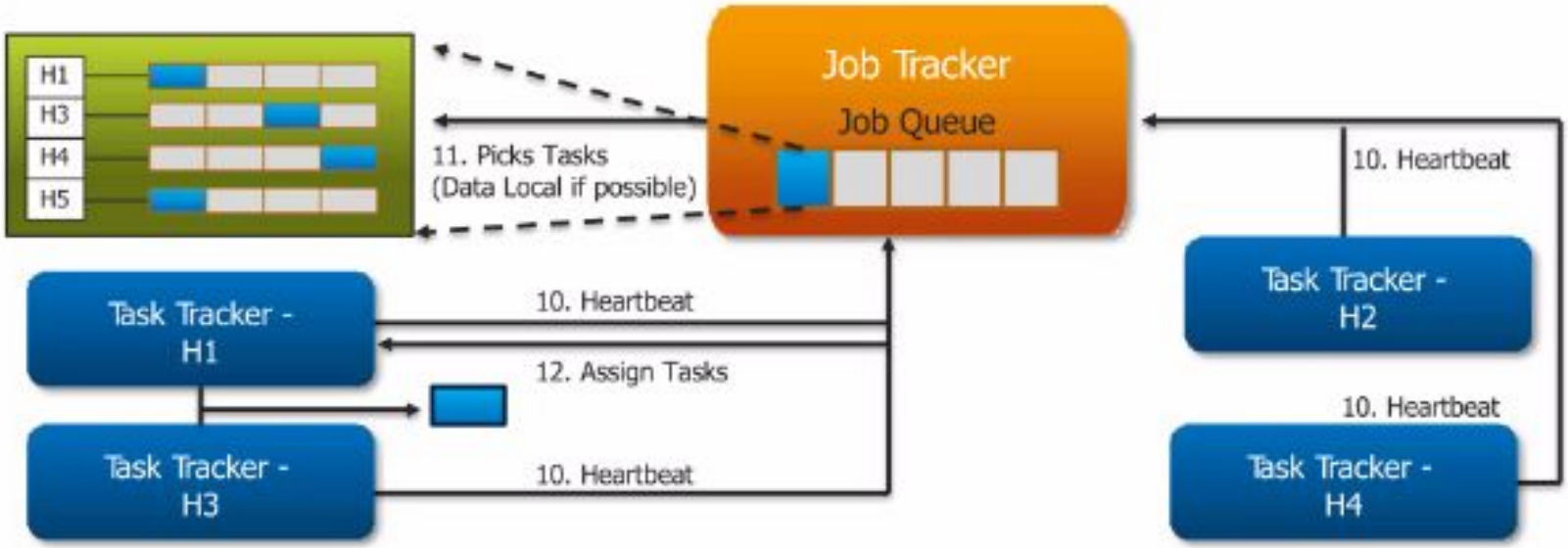
# Map Reduce Concepts

- **Job Tracker**
  - The Job-Tracker is responsible for accepting jobs from clients, dividing those jobs into tasks, and assigning those tasks to be executed by worker nodes.
- **Task Tracker**
  - Task-Tracker is a process that manages the execution of the tasks currently assigned to that node. Each Task Tracker has a fixed number of slots for executing tasks (two maps and two reduces by default).

# Job Tracker


1. Copy Input Files
DFS
Input Files
Job.xml.
Job.jar.
3. Get Input Files' Info
Client
5. Upload Job
Information
2. Submit Job
4. Create Splits
User
6. Submit Job
Job Tracker

# Job Tracker


H1
H3
H4
H5
Job Tracker
Job Queue
11. Picks Tasks
(Data Local if possible)
10. Heartbeat
Task Tracker -
H2
Task Tracker -
H1
10. Heartbeat
12. Assign Tasks
10. Heartbeat
Task Tracker -
H4
Task Tracker -
H3
10. Heartbeat

# Understanding Data Transformations

- In order to write MapReduce applications you need to have an understanding of how data is transformed as it executes in the MapReduce framework.

**From start to finish, there are four fundamental transformations. Data is:**

- Transformed from the input files and fed into the mappers
- Transformed by the mappers
- Sorted, merged, and presented to the reducer
- Transform by reducers and written to output files

# Solving a Programming Problem using MapReduce

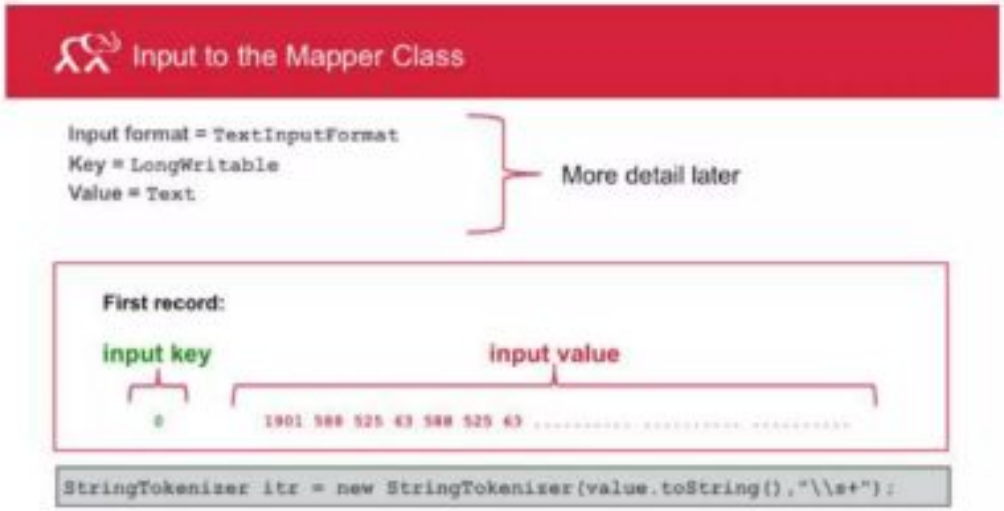
Example Data Set

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2000 | 2025191 | 1788950 | 236241 | 1544607 | 1458185 | 86422 | 480584 | 330765 | 149819 |
| 2001 | 1991082 | 1862846 | 128236 | 1483563 | 1516008 | -32445 | 507519 | 346838 | 160681 |
| 2002 | 1853136 | 2010894 | -157758 | 1337815 | 1655232 | -317417 | 515321 | 355662 | 159659 |
| 2003 | 1782314 | 2159899 | -377585 | 1258472 | 1796890 | -538418 | 523842 | 363009 | 160833 |
| 2004 | 1880114 | 2292841 | -412727 | 1345369 | 1913330 | -567961 | 534745 | 379511 | 155234 |
| 2005 | 2153611 | 2471957 | -318346 | 1576135 | 2069746 | -493611 | 577476 | 402211 | 175265 |
| 2006 | 2406869 | 2655050 | -248181 | 1798487 | 2232981 | -434494 | 608382 | 422069 | 186313 |
| 2007 | 2567985 | 2728686 | -160701 | 1932896 | 2275049 | -342153 | 635089 | 453637 | 181452 |
| 2008 | 2523991 | 2982544 | -458553 | 1865945 | 2507793 | -641848 | 658046 | 474751 | 183295 |
| 2009 | 2104989 | 3517677 | -1412688 | 1450980 | 3000661 | -1549681 | 654009 | 517016 | 136993 |
| 2010 | 2162706 | 3457079 | -1294373 | 1531019 | 2902397 | -1371378 | 631687 | 554682 | 77005 |
| 2011 | 2303466 | 3603059 | -1299593 | 1737678 | 3104453 | -1366775 | 565788 | 498606 | 67182 |
| 2012 | 2450164 | 3537127 | -1086963 | 1880663 | 3029539 | -1148876 | 569501 | 507588 | 61913 |

year delta

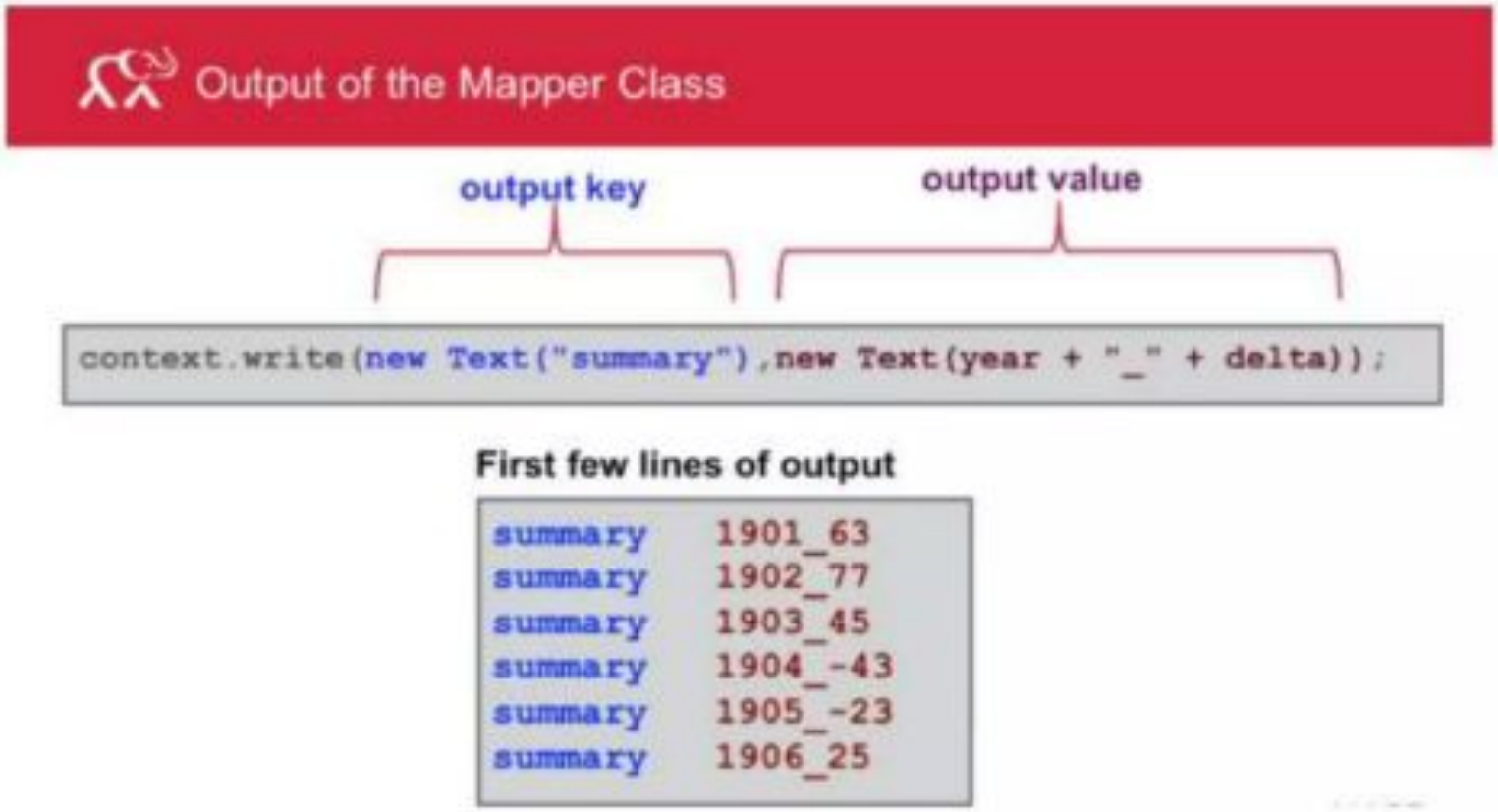
There are a total of 10 fields of information in each line. Our programming objective uses only the first and fourth fields, which are arbitrarily called "year" and "delta" respectively. We will ignore all the other fields of data.

# Designing and Implementing the Mapper Class




Let us define and implement the Mapper class to solve the programming problem.

In this example, we will be using the TextInputFormat class as the type of input to the Mapper class. Using this format, the key from the default record reader associated with TextInputFormat is the byte offset into the file (LongWritable). We won't be using this key for anything in our program, so we will just ignore it. The value from the default record reader is the line read from the input file, in Text.




After grabbing fields 1 and 4 from a record, the Mapper class emits the constant key ("summary") and a composite output value of the year (field 1) followed by an underscore followed by the delta (field 4).

# Design and Implement the Mapper Class

```
public class ReceiptsMapper  extends Mapper
<LongWritable,Text,Text,Text> {
   public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
      StringTokenizer iterator = new
StringTokenizer(value.toString(),"\\s+");
      String year = iterator.nextToken();
      iterator.nextToken();
      iterator.nextToken();
      String delta= iterator.nextToken();
      context.write(new Text("summary"), new Text(year + "_" +
delta);
   }
}
```

# Phases of Developing a MapReduce Application

- Configuration API
- Configuring the Development Environment
- GenericOptionsParser, Tool and ToolRunner
- Writing Unit Tests
- Running locally and in a cluster on Test Data
- The MapReduce Web UI
- Hadoop Logs
- Tuning a Job to improve performance

# Stages 1:Developing a MapReduce Application

- Writing a program in MapReduce follows a certain pattern.
- You start by writing your map and reduce functions, ideally with unit tests to make sure they do what you expect.
- Then you write a driver program to run a job, which can run from your IDE using a small subset of the data to check that it is working.
- If it fails, you can use your IDE's debugger to find the source of the problem.
- With this information, you can expand your unit tests to cover this case and improve your mapper or reducer as appropriate to handle such input correctly.

# Stages 2:Developing a MapReduce Application

- When the program runs as expected against the small dataset, you are ready to unleash it on a cluster.
- Running against the full dataset is likely to expose some more issues, which you can fix as before, by expanding your tests and mapper or reducer to handle the new cases.
- Debugging failing programs in the cluster is a challenge, so we look at some common techniques to make it easier.

# Stage 3: Developing a MapReduce Application

- After the program is working, you may wish to do some tuning, first by running through some standard checks for making MapReduce programs faster and then by doing task profiling.
- Profiling distributed programs is not easy, but Hadoop has hooks to aid the process.

# Example: Word Count

**Task:** Counting the word occurances (frequencies) in a text file (or set of files).

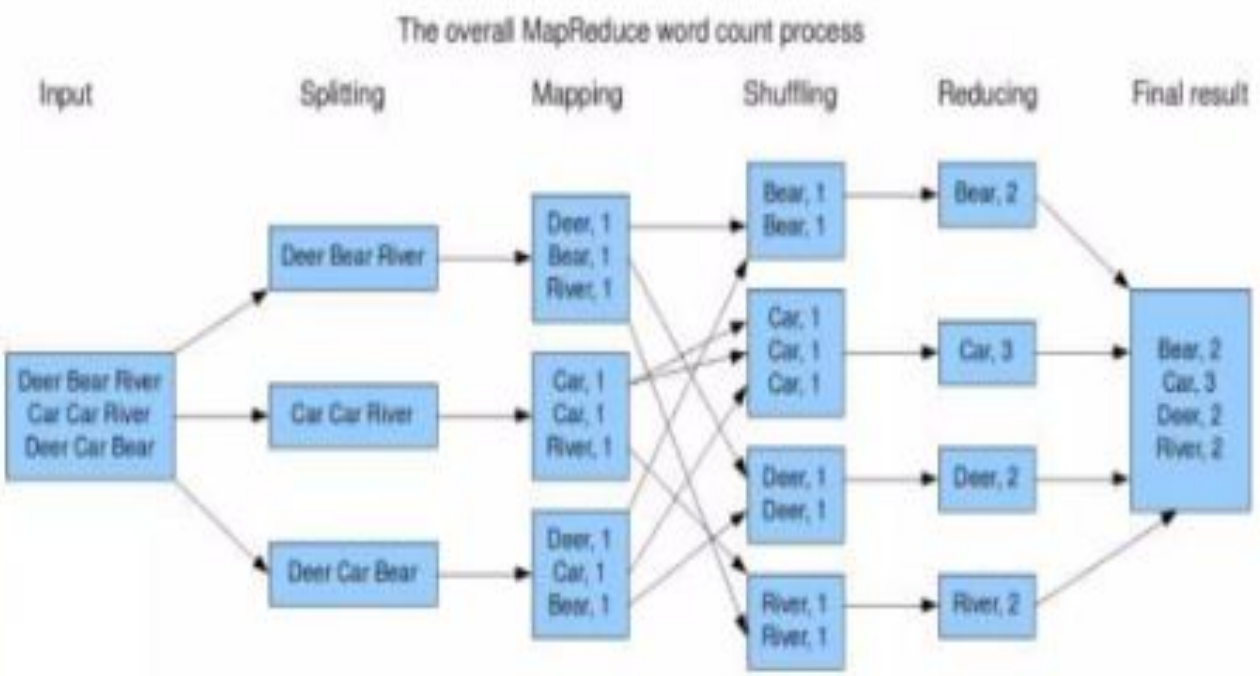
`<word, count >as <key, value >pair`

**Mapper:** Emits <word, 1 >for each word (no counting at this part).

**Shuffle in between:** pairs with same keys grouped together and passed to a single machine.

**Reducer:** Sums up the values (1s) with the same key value

# Example: Word Count



# Example: Job Tracker

## Tasks

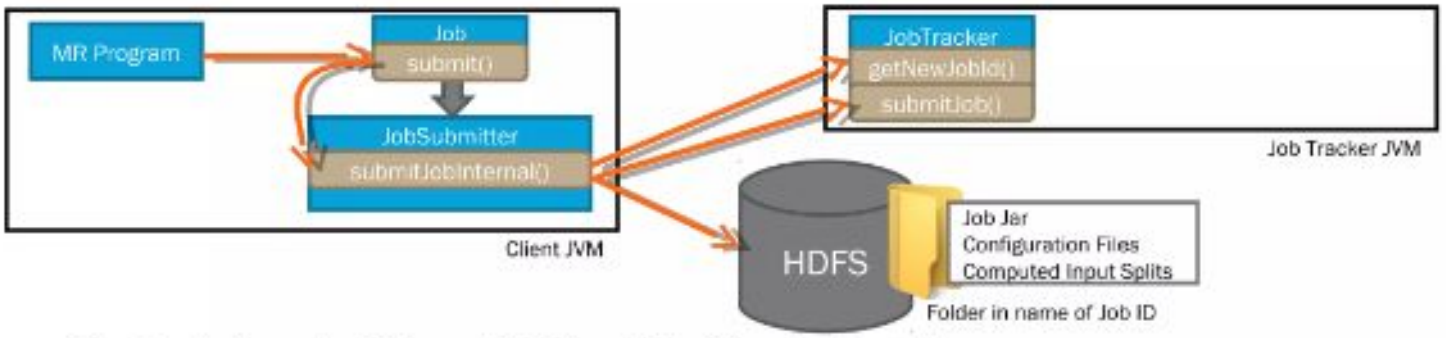
**Hadoop map task list for job_200904110811_0003 on ip-10-250-110-47**

**Completed Tasks**

| Task | Complete | Status | Start Time | Finish Time | Errors | Counters |
|---|---|---|---|---|---|---|
| task_200904110811_0003_m_000043 | 100.00% | hdfs://ip-10-250-110-47.ec2.internal/user/root/input/ncdc/all/1949.gz:0+223338475 | 11-Apr-2009 09:00:06 | 11-Apr-2009 09:01:25 (1mins, 18sec) | | 10 |
| task_200904110811_0003_m_000044 | 100.00% | Detected possibly corrupt record: see logs. | 11-Apr-2009 09:00:06 | 11-Apr-2009 09:01:28 (1mins, 21sec) | | 11 |
| task_200904110811_0003_m_000045 | 100.00% | hdfs://ip-10-250-110-47.ec2.internal/user/root/input/ncdc/all/1970.gz:0+208374610 | 11-Apr-2009 09:00:06 | 11-Apr-2009 09:01:28 (1mins, 21sec) | | 10 |

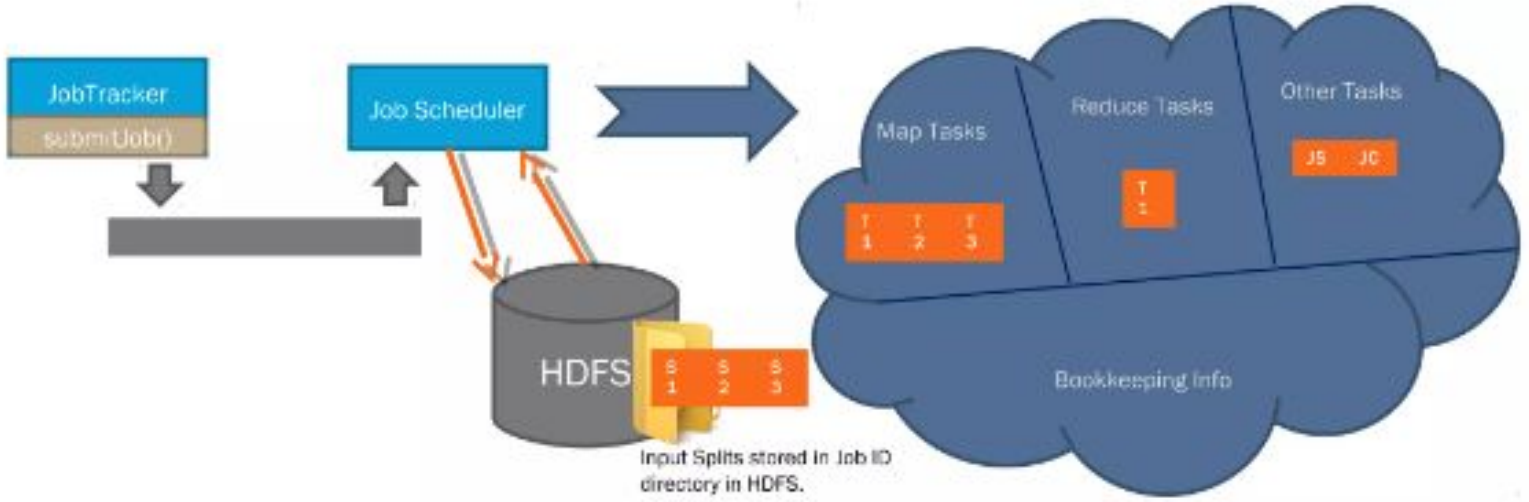
# ANATOMY OF CLASSIC MAPREDUCE IN HADOOP

FIRST A JOB HAS TO BE SUBMITTED TO HADOOP CLUSTER. LET'S SEE HOW JOB SUBMISSION HAPPENS IN HADOOP.



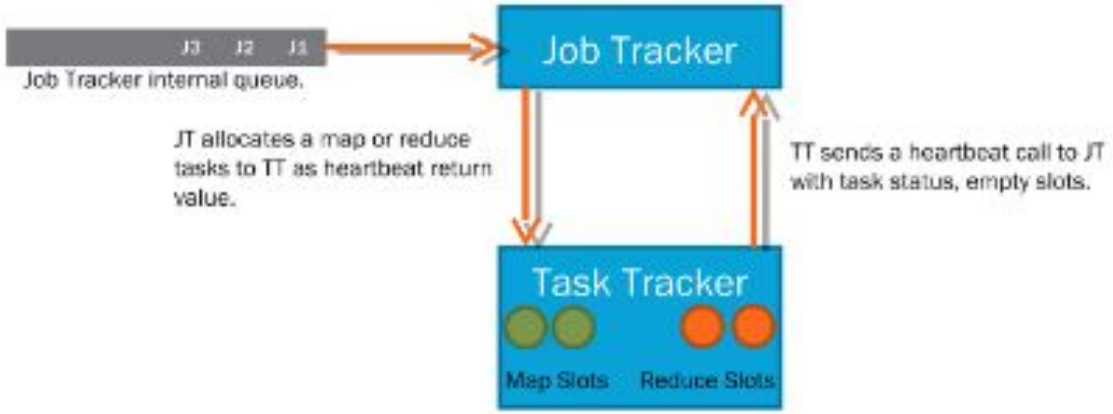

- Client calls the submit() (or waitForCompletion()) method on Job.
- Job creates a new instance of JobSubmitter which has a method called 'submitJobInternal()'.
- Then the Job.submit() method calls JobSubmitter.submitJobInternal() method which does the following;
  - Invokes the JobTracker.getNewJobId() to generate a unique id for the job.
  - Checks the output specification. If output path is not specified or if it already exists, then an exception is thrown.
  - Checks the input specification. In case of invalid input path or if splits couldn't be computed, then an exception is thrown. Else, the input splits are computed.
  - Creates a directory in the name of Job Id in HDFS.
  - Copies Job jar, configuration files and computed input splits to this directory. The Job jar has high replication factor which is configured in 'mapred.submit.replication' property.
  - Informs the JobTracker that the job is ready to be executed by calling submitJob() on JobTracker.
- Instead of Job.submit() the MapReduce program can call Job.waitForCompletion() method. Difference here is, the later waits till the job is complete by polling the JobTracker for every second. Once the job is completed successfully, counters are displayed in console. In case of job failure, the exception is displayed in console. But the submit() method terminates once the job submission is done.

NEXT THE SUBMITTED JOB WILL BE INITIALIZED.
NOW LET'S SEE HOW JOB INITIALIZATION
HAPPENS IN HADOOP.



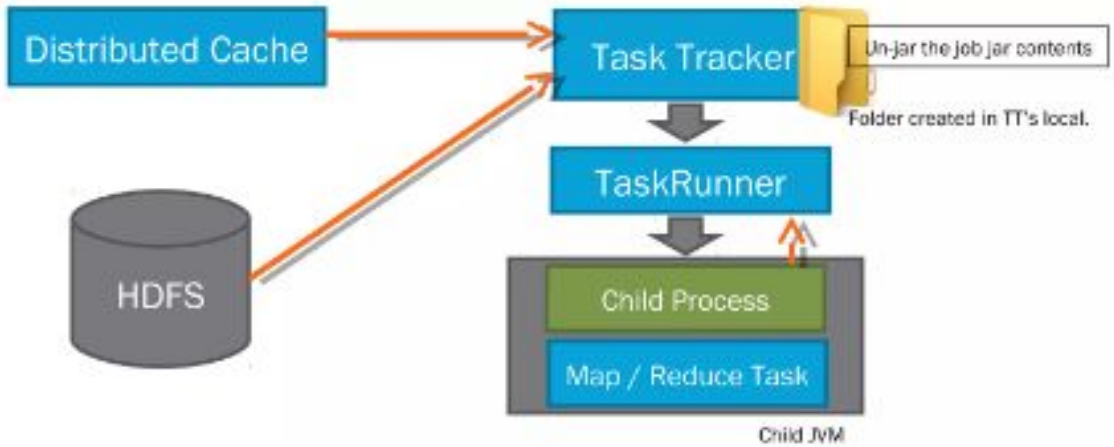

- When JobTracker.submitJob() is called, JobTracker adds a token to an internal queue.
- A job scheduler picks it up and creates an object representation for the job. This representation encapsulates tasks & bookkeeping infos (to track the status & progress of job's tasks).
- Job scheduler then reads the input splits from HDFS.
- Then the scheduler creates one map task for each input split.
- 'mapred.reduce.tasks' property is meant to have an integer. Job scheduler reads this property and creates those many number of reduce tasks. Let's assume the property's value was 1.
- Job setup (JS) & Job cleanup (JC) are the other 2 tasks created. Job setup task is run before any map task and Job cleanup task is run after all reduce tasks are complete.
- Each job has an OutputCommitter (a Java Interface) associated with it. Default implementation is FileOutputCommitter. The OutputCommitter defines what setup and cleanup task (for job & task) should do.
- In case of FileOutputCommitter, the job setup task will create the final output directory for the job and temp working space for task output. For cleanup task, it deletes the temp working space for task output. Please refer the API document for more info.

INITIALIZED JOB IS SPLIT INTO TASKS AND JOB TRACKER ASSIGNS TASKS TO TASK TRACKER. NOW LET'S SEE HOW TASK ASSIGNMENT HAPPENS IN HADOOP.




- Based on factors like available memory, available cores, etc., fixed number of map and reduce slots are configured in a Task Tracker.
- When Task Tracker is up and running, it sends a heartbeat call (for every 5 seconds) to Job Tracker. This is a two way communication between Task Tracker & Job Tracker. Task Tracker uses this call to inform Job Tracker that its alive. Also Task Tracker uses this call to inform Job Tracker about the status of tasks and which all map & reduce slots are available.
- Job Tracker might have many jobs in queue to run. It uses one of the scheduling algorithms to pick a job from queue. Once a job is picked, its associated tasks will be picked.
- When Job Tracker finds that there are empty slots (from the heartbeat call), it allocates tasks to Task Tracker using the heartbeat return value.
- Job Tracker considers the data locality when allocating map tasks; i.e., Job Tracker tries to allocate the map job to a Task Tracker where the block is available (which is called data local). If its not possible, Job Tracker tries to allocate the map task to a map slot in same rack (which is called rack local). There are no such considerations for reduce task.

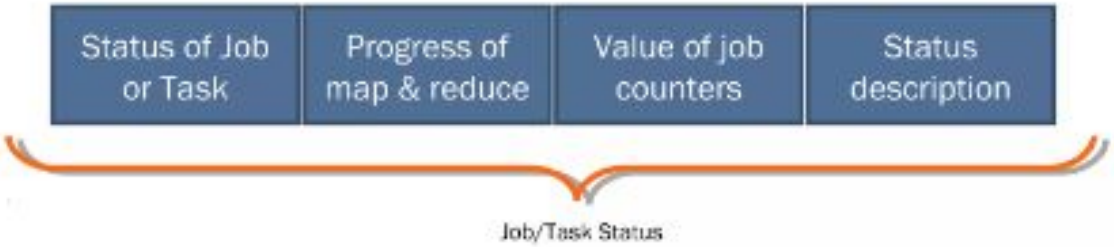
NOW TASKS ARE ASSIGNED TO TASK TRACKER WHICH FOLLOWS A SERIES OF STEPS TO EXECUTE A TASK. LET'S SEE HOW TASKS ARE EXECUTED IN TASK TRACKER.



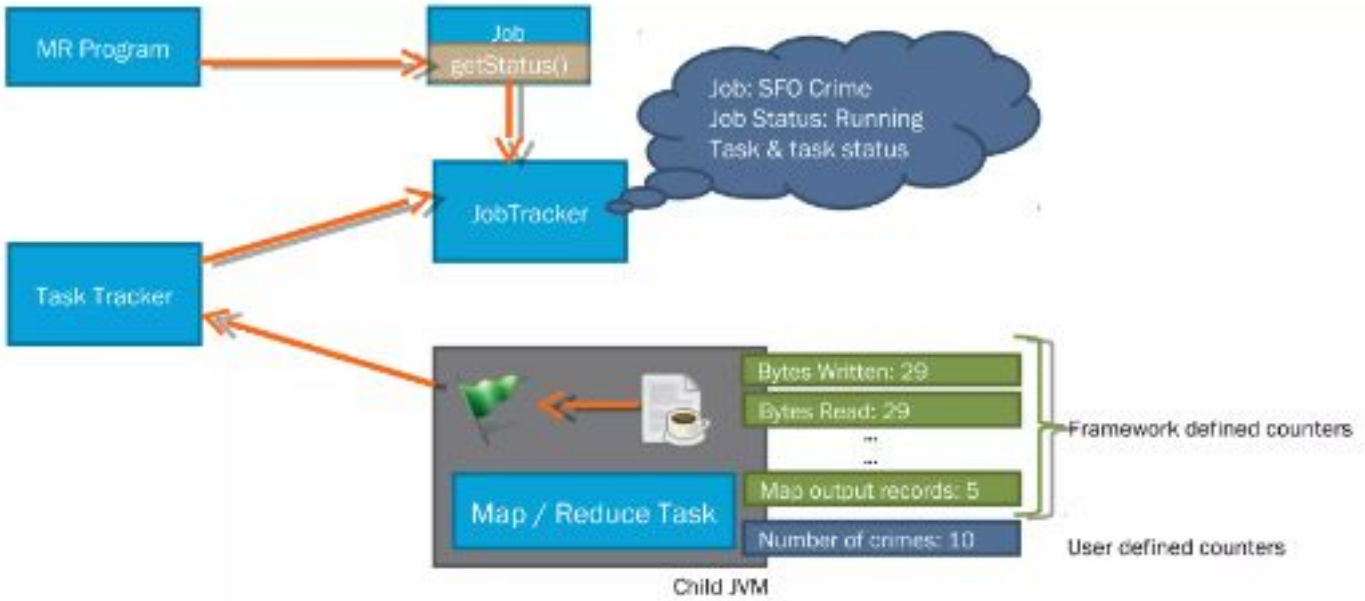

- Now the Task Tracker has been assigned a task.
- Task Tracker copies the job jar from HDFS to task tracker's file system. Also it copies required files from Distributed Cache.
- Task Tracker creates a new folder in task tracker's file system.
- Job jar content is un-jared into the new folder.
- Creates a new instance of TaskRunner.
- TaskRunner launches a new JVM to run each task, so that any bug in the user-defined map & reduce functions don't affect the Task Tracker.
- The child process communicates with its parent through the umbilical interface. It informs the parent of the task's progress every few seconds until the task is complete.
- There is a setup and cleanup tasks executed in the same JVM where the task is executed. The OutputCommiter implementation associated with the job determines what action to be taken during startup and cleanup.

SINCE TASKS ARE EXECUTED IN A DISTRIBUTED ENVIRONMENT, TRACKING THE PROGRESS AND STATUS OF JOB IS TRICKY. LET'S SEE HOW PROGRESS AND STATUS UPDATES ARE TAKEN CARE IN HADOOP.

- MapReduce jobs are long running batch job which takes anything from one minute to hours to run. Because of this user has to get feedback on how job is processing.
- Each job and task have status which comprises of the following;




- Status of Job/Task: Possible values are RUNNING, SUCCESSFULLY COMPLETED and FAILED. Job tracker and Task trackers set the value as the job or task progresses.
- Each task tracks the program by tracking the proportion of the task completed.
  - Map task's progress is measured by the proportion of input processed so far.
  - Measuring Reduce task's progress is little tricky. The system does it by dividing the total progress into 3 parts corresponding to 3 phases of Shuffle.

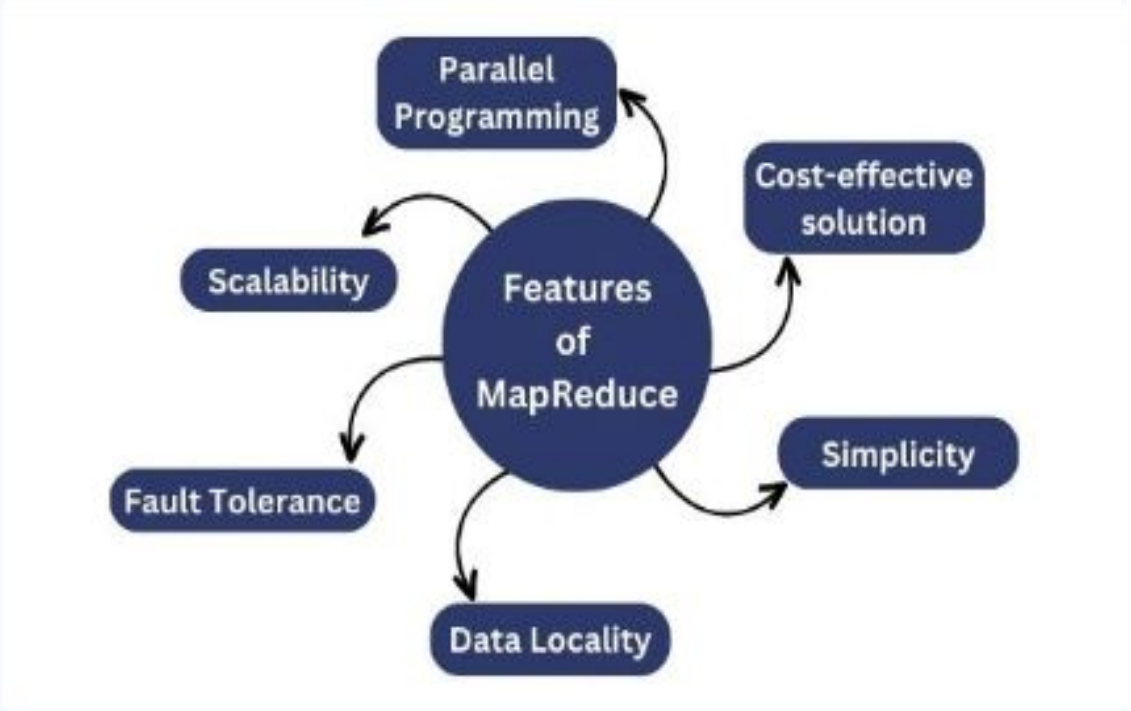



- Each task has set of counters that count various events as the task runs.
- Most of these counters are build into the framework. We can define our own counters. These are called user defined counters.
- As the tasks progress, it sets a flag to indicate that its time to send the progress to Task Tracker.
- This flag is checked every second by a thread and notifies the Task Tracker.
- For every 5 seconds, the Task Tracker sends a heart beat to Job Tracker. The status of all tasks are sent to Job tracker along with the heartbeat call.
- Counters are sent less frequently than every 5 seconds because they can be relatively high-bandwidth.
- Job Tracker combines these updates to produce a global view of the status of all the jobs & its related tasks.
- Clients invoke the API Job.getStatus() for every second to get the status from Job Tracker.

THIS EXECUTION PROCESS CONTINUES TILL ALL THE TASKS ARE COMPLETED. ONCE THE LAST TASK IS COMPLETED, MR FRAMEWORK ENTERS THE LAST PHASE CALLED JOB COMPLETION.

# Key Features of MapReduce

There are some key features of MapReduce below:



## Scalability

MapReduce can scale to process vast amounts of data by distributing tasks across a large number of nodes in a cluster. This allows it to handle massive datasets, making it suitable for Big Data applications.

## Fault Tolerance

MapReduce incorporates built-in fault tolerance to ensure the reliable processing of data. It automatically detects and handles node failures, rerunning tasks on available nodes as needed.

## Data Locality

MapReduce takes advantage of data locality by processing data on the same node where it is stored, minimizing data movement across the network and improving overall performance.

## Simplicity

The MapReduce programming model abstracts away many complexities associated with distributed computing, allowing developers to focus on their data processing logic rather than low-level details.

## Cost-Effective Solution

Hadoop's scalable architecture and MapReduce programming framework make storing and processing extensive data sets very economical.

## Parallel Programming

Tasks are divided into programming models to allow for the simultaneous execution of independent operations. As a result, programs run faster due to parallel processing, making it easier for a process to handle each job. Thanks to parallel processing, these distributed tasks can be performed by multiple processors. Therefore, all software runs faster.

Features explained in detail:

# 1. Highly scalable

A framework with excellent scalability is Apache Hadoop MapReduce. This is because of its capacity for distributing and storing large amounts of data across numerous servers. These servers can all run simultaneously and are all reasonably priced.

By adding servers to the cluster, we can simply grow the amount of storage and computing power. We may improve the capacity of nodes or add any number of nodes (horizontal scalability) to attain high computing power. Organizations may execute applications from massive sets of nodes, potentially using thousands of terabytes of data, thanks to Hadoop MapReduce programming.

## 2. Versatile

Businesses can use MapReduce programming to access new data sources. It makes it possible for companies to work with many forms of data. Enterprises can access both organized and unstructured data with this method and acquire valuable insights from the various data sources.

Since Hadoop is an open-source project, its source code is freely accessible for review, alterations, and analyses. This enables businesses to alter the code to meet their specific needs. The MapReduce framework supports data from sources including email, social media, and clickstreams in different languages.

# 3. Secure

The MapReduce programming model uses the HBase and HDFS security approaches, and only authenticated users are permitted to view and manipulate the data. HDFS uses a replication technique in Hadoop 2 to provide fault tolerance. Depending on the replication factor, it makes a clone of each block on the various machines. One can therefore access data from the other devices that house a replica of the same data if any machine in a cluster goes down. Erasure coding has taken the role of this replication technique in Hadoop 3. Erasure coding delivers the same level of fault tolerance with less area. The storage overhead with erasure coding is less than 50%.

# 4. Affordability

With the help of the MapReduce programming framework and Hadoop's scalable design, big data volumes may be stored and processed very affordably. Such a system is particularly cost-effective and highly scalable, making it ideal for business models that must store data that is constantly expanding to meet the demands of the present.

In terms of scalability, processing data with older, conventional relational database management systems was not as simple as it is with the Hadoop system. In these situations, the company had to minimize the data and execute classification based on presumptions about how specific data could be relevant to the organization, hence deleting the raw data. The MapReduce programming model in the Hadoop scale-out architecture helps in this situation.

# 5. Fast-paced

The Hadoop Distributed File System, a distributed storage technique used by MapReduce, is a mapping system for finding data in a cluster. The data processing technologies, such as MapReduce programming, are typically placed on the same servers that enable quicker data processing.

Thanks to Hadoop's distributed data storage, users may process data in a distributed manner across a cluster of nodes. As a result, it gives the Hadoop architecture the capacity to process data exceptionally quickly. Hadoop MapReduce can process unstructured or semi-structured data in high numbers in a shorter time.

# 6. Based on a simple programming model

Hadoop MapReduce is built on a straightforward programming model and is one of the technology's many noteworthy features. This enables programmers to create MapReduce applications that can handle tasks quickly and effectively. Java is a very well-liked and simple-to-learn programming language used to develop the MapReduce programming model.

Java programming is simple to learn, and anyone can create a data processing model that works for their company. Hadoop is straightforward to utilize because customers don't need to worry about computing distribution. The framework itself does the processing.

# 7. Parallel processing-compatible

The parallel processing involved in MapReduce programming is one of its key components. The tasks are divided in the programming paradigm to enable the simultaneous execution of independent activities. As a result, the program runs faster because of the parallel processing, which makes it simpler for the processes to handle each job. Multiple processors can carry out these broken-down tasks thanks to parallel processing. Consequently, the entire software runs faster.

## 8. Reliable

The same set of data is transferred to some other nodes in a cluster each time a collection of information is sent to a single node. Therefore, even if one node fails, backup copies are always available on other nodes that may still be retrieved whenever necessary. This ensures high data availability.

The framework offers a way to guarantee data trustworthiness through the use of Block Scanner, Volume Scanner, Disk Checker, and Directory Scanner modules. Your data is safely saved in the cluster and is accessible from another machine that has a copy of the data if your device fails or the data becomes corrupt.

## 9. Highly available

Hadoop's fault tolerance feature ensures that even if one of the DataNodes fails, the user may still access the data from other DataNodes that have copies of it. Moreover, the high accessibility Hadoop cluster comprises two or more active and passive NameNodes running on hot standby. The active NameNode is the active node. A passive node is a backup node that applies changes made in active NameNode's edit logs to its namespace.