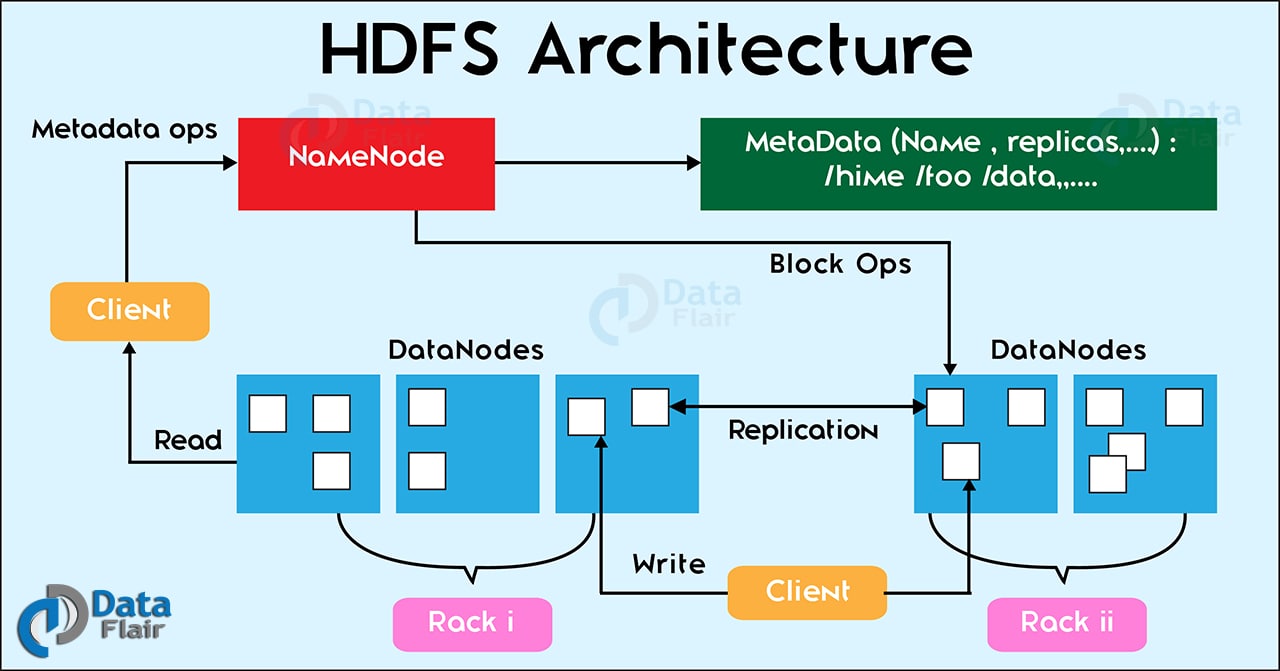
**5.1. What is HDFS?**

[**Hadoop HDFS or Hadoop Distributed File System**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/) is a distributed file system which provides storage in Hadoop in a distributed fashion.

In Hadoop Architecture on the master node, a daemon called ***namenode*** run for HDFS. On all the slaves a daemon called ***datanode*** run for HDFS. Hence slaves are also called as datanode. Namenode stores meta-data and manages the datanodes. On the other hand, Datanodes stores the data and do the actual task.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/hadoop-hdfs-architecture-1.jpg)

*Hadoop Tutorial – Hadoop HDFS Architecture*

HDFS is a highly [**fault tolerant**](http://data-flair.training/blogs/learn-hadoop-hdfs-fault-tolerance/), **distributed**, **reliable** and **scalable** file system for data storage. First Follow this guide to[**learn more about features of HDFS**](http://data-flair.training/blogs/features-hadoop-hdfs-overview-beginners/)and then proceed further with the Hadoop tutorial.

HDFS is developed to handle huge volumes of data. The file size expected is in the range of GBs to TBs. A file is split up into blocks (default 128 MB) and stored distributedly across multiple machines. These blocks replicate as per the replication factor. After replication, it stored at different nodes. This handles the [**failure of a node**](http://data-flair.training/blogs/hadoop-high-availability-tutorial/) in the cluster. So if there is a file of 640 MB, it breaks down into 5 blocks of 128 MB each (if we use the default value).

### 5.2. What is MapReduce?

In this Hadoop Basics Tutorial, now its time to understand one of the most important pillars of Hadoop, i.e. Hadoop MapReduce. The**Hadoop MapReduce** is a programming model. As it is designed for large volumes of data in parallel by dividing the work into a set of independent tasks. MapReduce is the heart of Hadoop, it moves computation close to the data. As a movement of a huge volume of data will be very costly. It allows massive scalability across hundreds or thousands of servers in a [**Hadoop cluster.**](http://data-flair.training/blogs/install-deploy-cloudera-hadoop-cdh5-apache-2-x-centos/)

Hence, Hadoop MapReduce is a framework for distributed processing of huge volumes of data set over a cluster of nodes. As data stores in a distributed manner in HDFS. It provides the way to [**Map**](http://data-flair.training/blogs/mapper-in-hadoop-mapreduce/)–[**Reduce**](http://data-flair.training/blogs/reducer-in-hadoop-mapreduce/) to perform parallel processing.

### 5.3. What is YARN Hadoop?

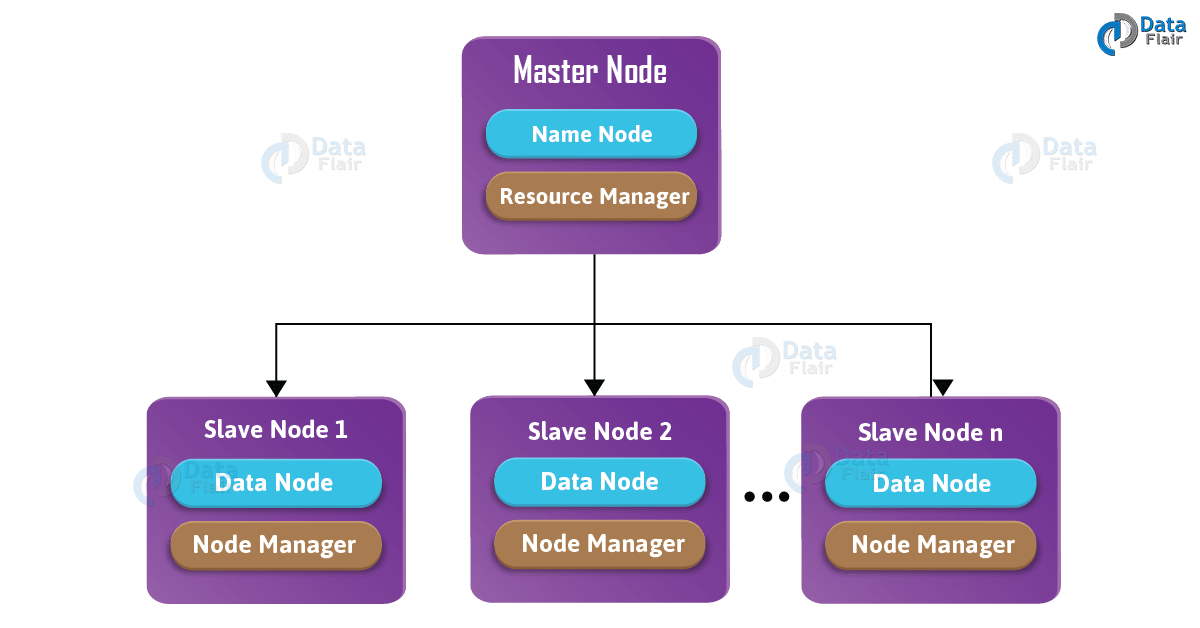
[**YARN – Yet Another Resource Negotiator**](http://data-flair.training/blogs/hadoop-yarn-tutorial/) is the resource management layer of Hadoop. In the multi-node cluster, as it becomes very complex to manage/allocate/release the resources (CPU, memory, disk). **Hadoop Yarn** manages the resources quite efficiently. It allocates the same on request from any application.

On the master node, the [**ResourceManager**](http://data-flair.training/blogs/deep-dive-into-hadoop-yarn-resource-manager/) daemon runs for the YARN then for all the slave nodes [**NodeManager**](http://data-flair.training/blogs/deep-dive-into-hadoop-yarn-node-manager/) daemon runs.

Learn the differences between two resource manager [**Yarn vs. Apache Mesos**](http://data-flair.training/blogs/comparison-between-apache-mesos-vs-hadoop-yarn/). Next topic in the Big Data Hadoop for beginners is a very important part of Hadoop i.e. Hadoop Daemons

## 6. Hadoop Daemons

Daemons are the processes that run in the background. There are mainly 4 daemons which run for Hadoop.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/HADOOP-DAEMONS-ARCHITECTURE-vo.1-01-min.png)

*Hadoop Daemons*

* **Namenode** – It runs on master node for HDFS.
* **Datanode** – It runs on slave nodes for HDFS.
* **ResourceManager** – It runs on master node for Yarn.
* **NodeManager** – It runs on slave node for Yarn.

These 4 demons run for Hadoop to be functional. Apart from this, there can be secondary NameNode, standby NameNode, Job HistoryServer, etc.

## 7.’How do Hadoop works?’

Till now in Hadoop training we have studied Hadoop Introduction and Hadoop architecture in detail. Now next let us summarize Apache [**Hadoop working**](http://data-flair.training/blogs/how-hadoop-works-internal-working-and-its-components/)step by step:

**i)** Input data breaks into blocks of size 128 Mb (by default) and then moves to different nodes.

**ii)** Once all the blocks of the file stored on datanodes then a user can process the data.

**iii)** Then, master schedules the program (submitted by the user) on individual nodes.

**iv)** Once all the nodes process the data then the output is[**written back to HDFS**](http://data-flair.training/blogs/hdfs-data-write-operation/)**.**

## 8. Hadoop Flavors

This section of Hadoop Tutorial talks about the various flavors of Hadoop.

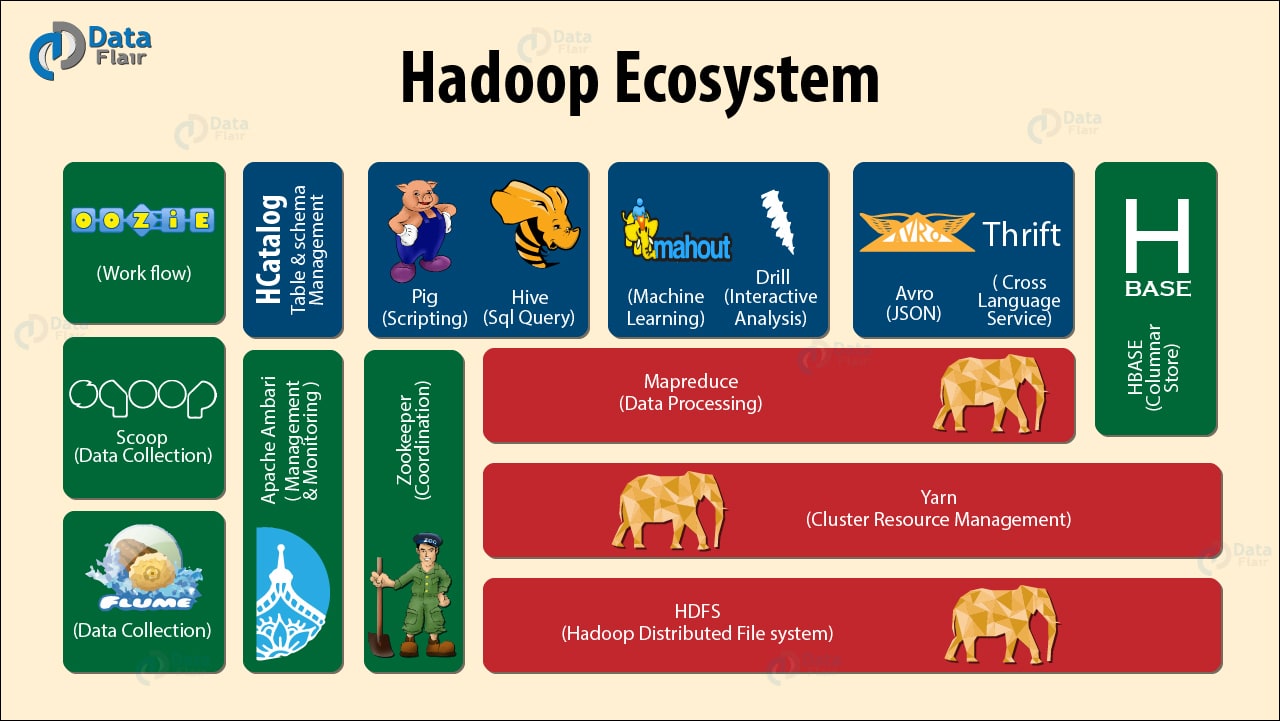
* **Apache** – Vanilla flavor, as the actual code is residing in Apache repositories.
* **Hortonworks** – Popular distribution in the industry.
* **Cloudera** – It is the most popular in the industry.
* **MapR** – It has rewritten HDFS and its HDFS is faster as compared to others.
* **IBM** – Proprietary distribution is known as Big Insights.

All the databases have provided native connectivity with Hadoop for fast data transfer. Because, to transfer data from Oracle to Hadoop, you need a connector.

All flavors are almost same and if you know one, you can easily work on other flavors as well.

## 9. Hadoop Ecosystem Components

In this section of Hadoop tutorial, we will cover Hadoop ecosystem components. Let us see what all the components form the **Hadoop Eco-System**:

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/apache-hadoop-ecosystem-components-2.jpg)

*Hadoop Tutorial – Hadoop Ecosystem Components*

* [**Hadoop HDFS**](http://data-flair.training/blogs/introduction-tutorial-hdfs/)**–** Distributed storage layer for Hadoop.
* [**Yarn Hadoop**](http://data-flair.training/blogs/list-best-books-learn-apache-yarn/)**–** Resource management layer introduced in Hadoop 2.x.
* [**Hadoop Map-Reduce**](http://data-flair.training/blogs/hadoop-mapreduce-flow-how-data-flows-in-mapreduce/)**–** Parallel processing layer for Hadoop.
* [**HBase**](http://data-flair.training/blogs/hbase-tutorial-beginners-guide/)**–** It is a column-oriented database that runs on top of HDFS. It is a NoSQL database which does not understand the structured query. For sparse data set, it suits well.
* [**Hive**](http://data-flair.training/blogs/hive-tutorial-an-introductory-guide-for-beginners/)**–** Apache Hive is a data warehousing infrastructure based on Hadoop and it enables easy data summarization, using SQL queries.
* [**Pig**](http://data-flair.training/blogs/introduction-tutorial-apache-pig/)**–** It is a top-level scripting language. As we use it with Hadoop. Pig enables writing complex data processing without Java programming.
* [**Flume**](http://data-flair.training/blogs/introduction-apache-flume-tutorial-beginners-guide/)**–** It is a reliable system for efficiently collecting large amounts of log data from many different sources in real-time.
* **Sqoop –** It is a tool design to transport huge volumes of data between Hadoop and RDBMS.
* **Oozie –** It is a Java Web application uses to schedule Apache Hadoop jobs. It combines multiple jobs sequentially into one logical unit of work.
* **Zookeeper** – A centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
* **Mahout –** A library of scalable machine-learning algorithms, implemented on top of Apache Hadoop and using the MapReduce paradigm.

Refer this [**Hadoop Ecosystem Components tutorial**](http://data-flair.training/blogs/hadoop-ecosystem-components/) for the detailed study of All the Ecosystem components of Hadoop.

* Introduction to Hadoop Ecosystem

**HDFS  Components:**

There are two major components of Hadoop HDFS- NameNode and DataNode. Let’s now discuss these Hadoop HDFS Components-

**i. NameNode**

It is also known as*Master* node. NameNode does not store actual data or dataset. NameNode stores Metadata i.e. number of [**blocks**](http://data-flair.training/blogs/data-blocks-hdfs-hadoop-distributed-file-system/), their location, on which Rack, which Datanode the data is stored and other details. It consists of files and directories.

**Tasks of HDFS NameNode**

* Manage file system namespace.
* Regulates client’s access to files.
* Executes file system execution such as naming, closing, opening files and directories.

**ii. DataNode**

It is also known as *Slave*. HDFS Datanode is responsible for storing actual data in HDFS. Datanode performs [**read and write operation**](http://data-flair.training/blogs/hadoop-hdfs-data-read-and-write-operations/) as per the request of the clients. Replica block of Datanode consists of 2 files on the file system. The first file is for data and second file is for recording the block’s metadata. HDFS Metadata includes checksums for data. At startup, each Datanode connects to its corresponding Namenode and does handshaking. Verification of namespace ID and software version of DataNode take place by handshaking. At the time of mismatch found, DataNode goes down automatically.

**Tasks of HDFS DataNode**

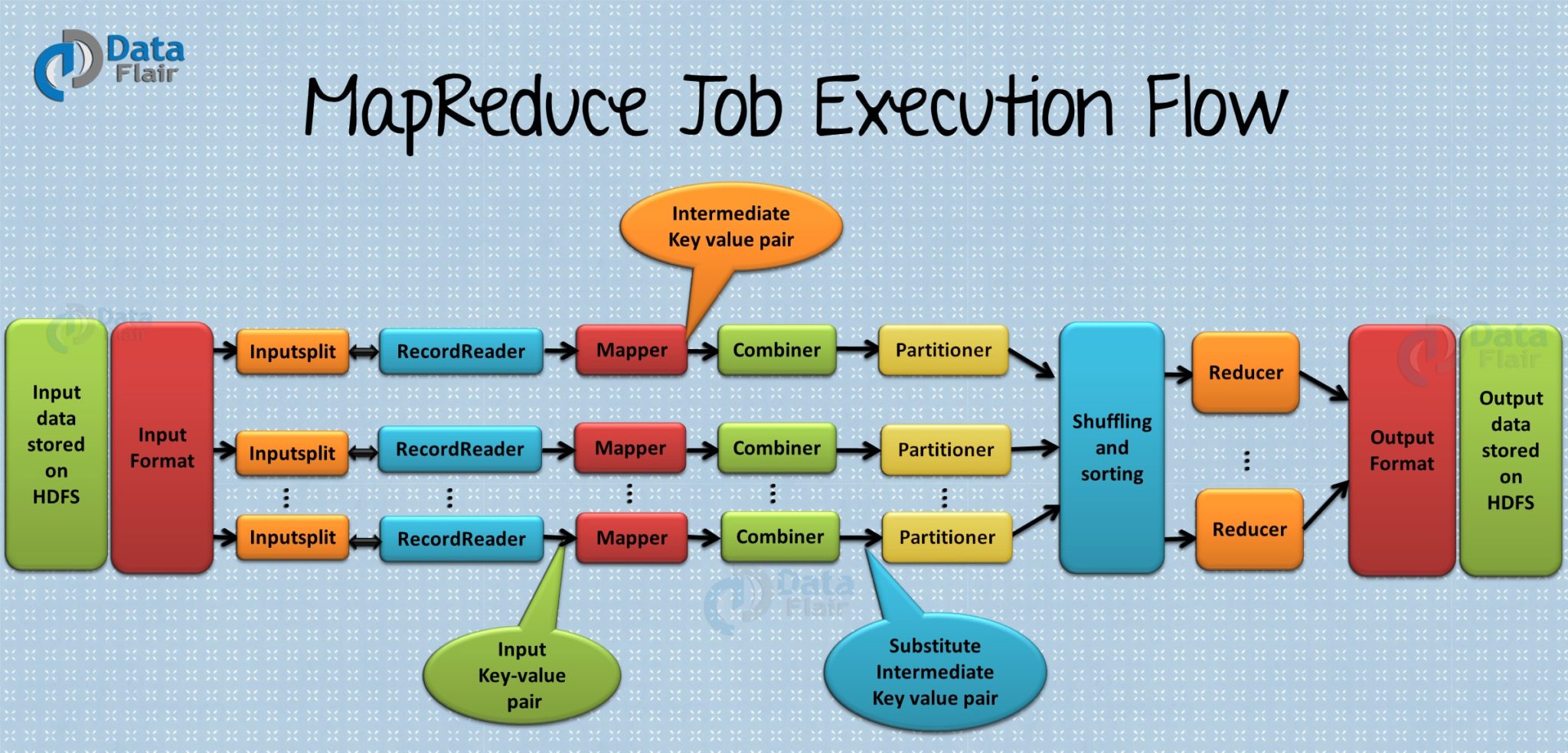
* DataNode performs operations like block replica creation, deletion, and replication according to the instruction of NameNode.
* DataNode manages data storage of the system.

This was all about HDFS as a Hadoop Ecosystem component.

### 2.2. MapReduce

**Hadoop MapReduce** is the core Hadoop ecosystem component which provides data processing. MapReduce is a software framework for easily writing applications that process the vast amount of structured and unstructured data stored in the Hadoop Distributed File system.

MapReduce programs are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster. Thus, it improves the speed and reliability of cluster this parallel processing.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/mapreduce-job-execution-flow-1-1.jpg)

*Hadoop Ecosystem Overview – Hadoop MapReduce*

**Working of MapReduce**

Hadoop Ecosystem component ‘MapReduce’ works by breaking the processing into two phases:

* Map phase
* Reduce phase

Each phase has [**key-value pairs**](http://data-flair.training/blogs/key-value-pairs-hadoop-mapreduce/) as input and output. In addition, programmer also specifies two functions:**map function**and**reduce function**

**Map** **function** takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). [**Read Mapper in detail**.](http://data-flair.training/blogs/mapper-in-hadoop-mapreduce/)

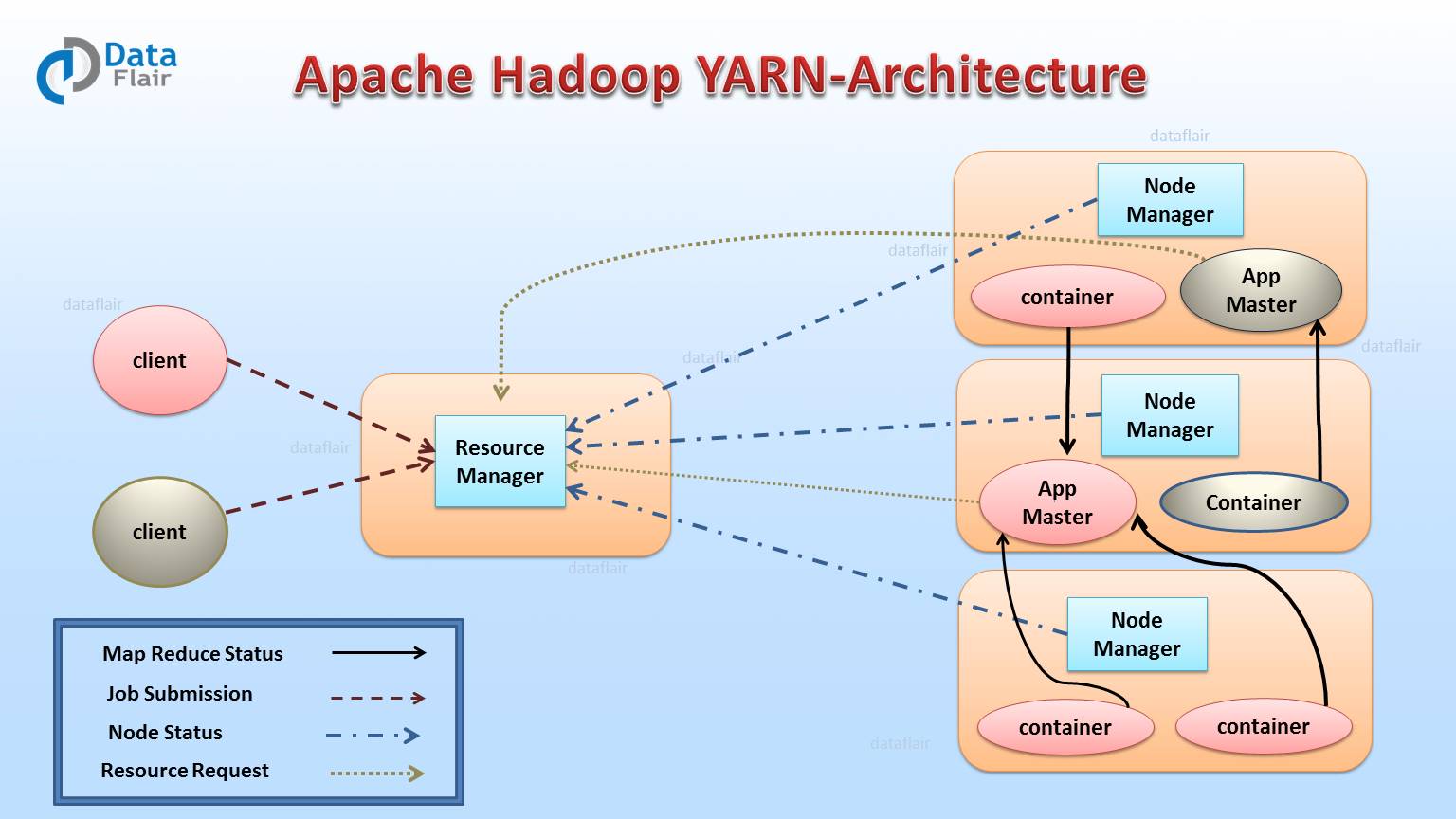
**Reduce** **function**takes the output from the Map as an input and combines those data tuples based on the key and accordingly modifies the value of the key. [**Read Reducer in detail**.](http://data-flair.training/blogs/reducer-in-hadoop-mapreduce/)

**Features of MapReduce**

* **Simplicity –**MapReduce jobs are easy to run. Applications can be written in any language such as **java, C++**, and **python**.
* **Scalability –**MapReduce can process petabytes of data.
* **Speed –**By means of parallel processing problems that take days to solve, it is solved in hours and minutes by MapReduce.
* **Fault Tolerance –**MapReduce takes care of failures. If one copy of data is unavailable, another machine has a copy of the same key pair which can be used for solving the same subtask.

### 2.3. YARN

**Hadoop YARN** (**Y**et **A**nother **R**esource **N**egotiator) is a Hadoop ecosystem component that provides the resource management. Yarn is also one the most important component of Hadoop Ecosystem.  YARN is called as the operating system of Hadoop as it is responsible for managing and monitoring workloads. It allows multiple data processing engines such as real-time streaming and batch processing to handle data stored on a single platform.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Apache-YARN-architecture-min-1.jpg)

*Apache Hadoop Ecosystem – Hadoop Yarn Diagram*

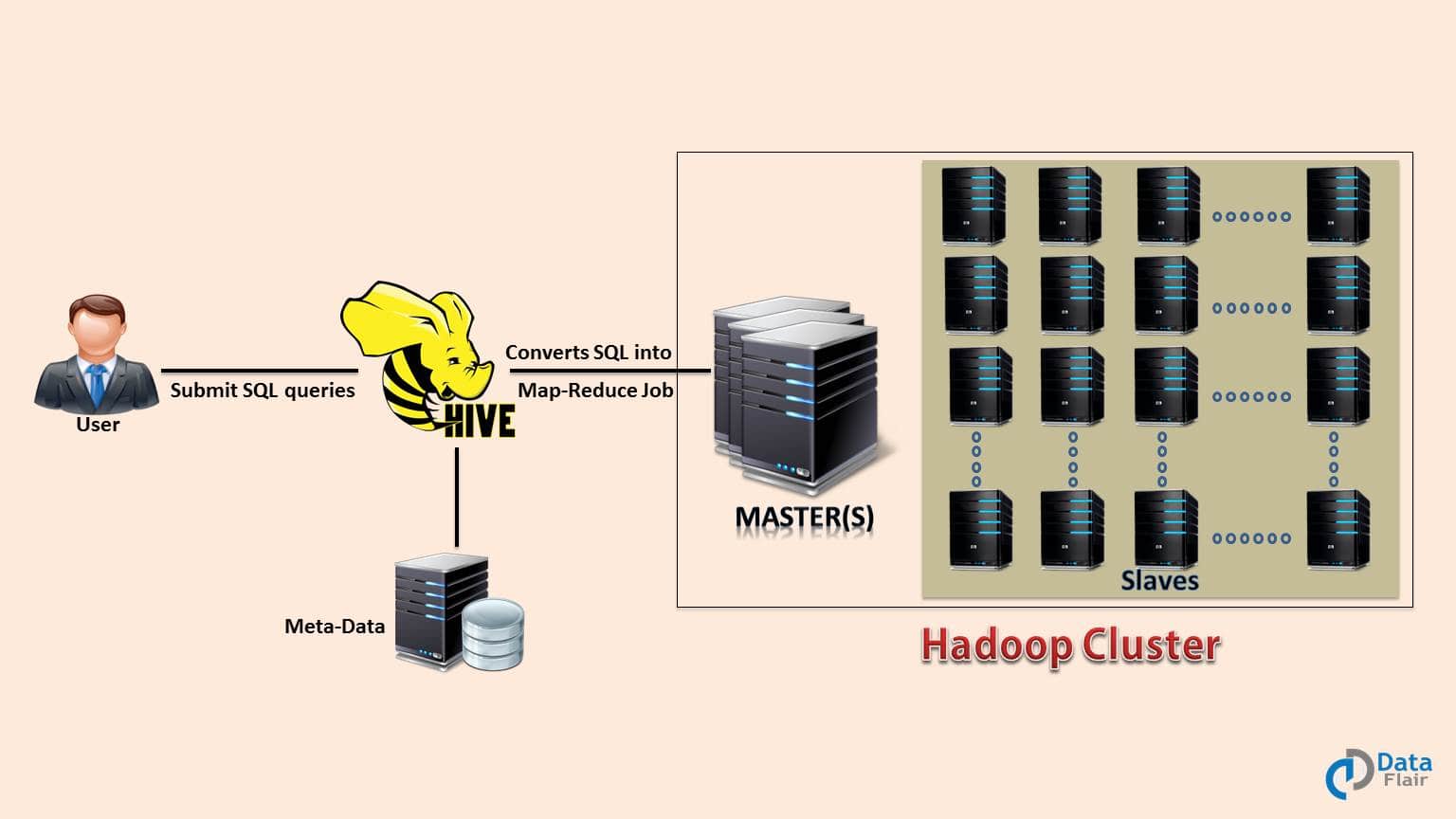
YARN has been projected as a data operating system for [**Hadoop2**](http://data-flair.training/blogs/setup-hadoop-2-yarn-psedo-distributed-mode/). Main features of YARN are:

* **Flexibility –** Enables other purpose-built data processing models beyond MapReduce (batch), such as interactive and streaming. Due to this feature of YARN, other applications can also be run along with Map Reduce programs in Hadoop2.
* **Efficiency –** As many applications run on the same cluster, Hence, efficiency of Hadoop increases without much effect on quality of service.
* **Shared –** Provides a stable, reliable, secure foundation and shared operational services across multiple workloads. Additional programming models such as graph processing and iterative modeling are now possible for data processing.

### 2.4. Hive

The Hadoop ecosystem component,**Apache Hive,** is an open source data warehouse system for querying and analyzing large datasets stored in Hadoop files. Hive do three main functions: data summarization, query, and analysis.

Hive use language called **HiveQL** (HQL), which is similar to SQL. HiveQL automatically translates SQL-like queries into [**MapReduce jobs**](http://data-flair.training/blogs/hadoop-mapreduce-job-execution-flow/) which will execute on Hadoop.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Apache-Hive-Introduction-Tutorial-Training-1.jpg)

*Components of Hadoop Ecosystem – Hive Diagram*

**Main parts of Hive are:**

* **Metastore –** It stores the metadata.
* **Driver –** Manage the lifecycle of a HiveQL statement.
* **Query compiler –** Compiles HiveQL into Directed Acyclic Graph(DAG).
* **Hive server –** Provide a thrift interface and JDBC/ODBC server.

### 2.7. HCatalog

It is a table and storage management layer for Hadoop. HCatalog supports different components available in Hadoop ecosystem like MapReduce, Hive, and Pig to easily read and write data from the cluster. HCatalog is a key component of Hive that enables the user to store their data in any format and structure.

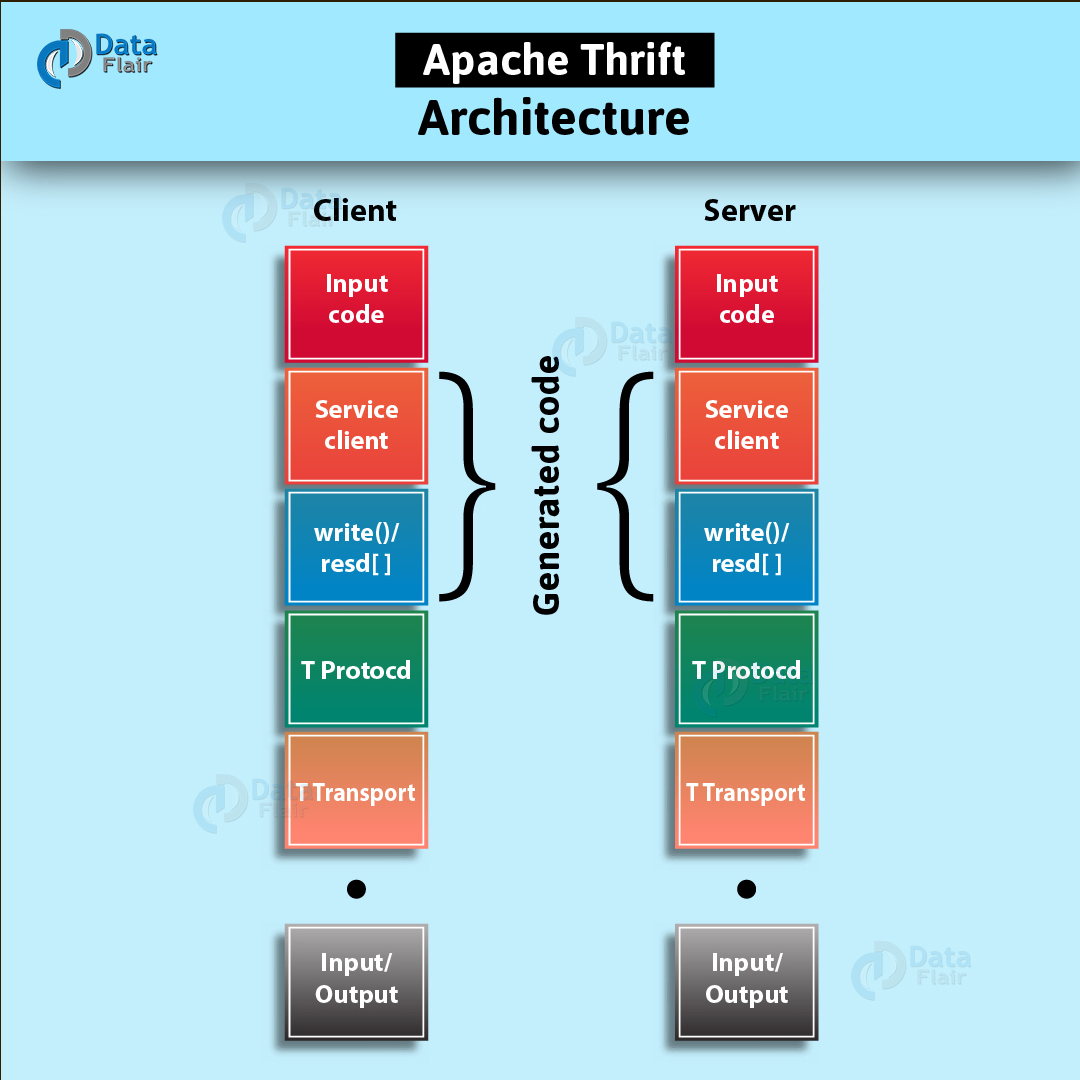
By default, HCatalog supports RCFile, CSV, JSON, sequenceFile and ORC file formats.

**Benefits of HCatalog:**

* Enables notifications of data availability.
* With the table abstraction, HCatalog frees the user from overhead of data storage.
* Provide visibility for data cleaning and archiving tools.

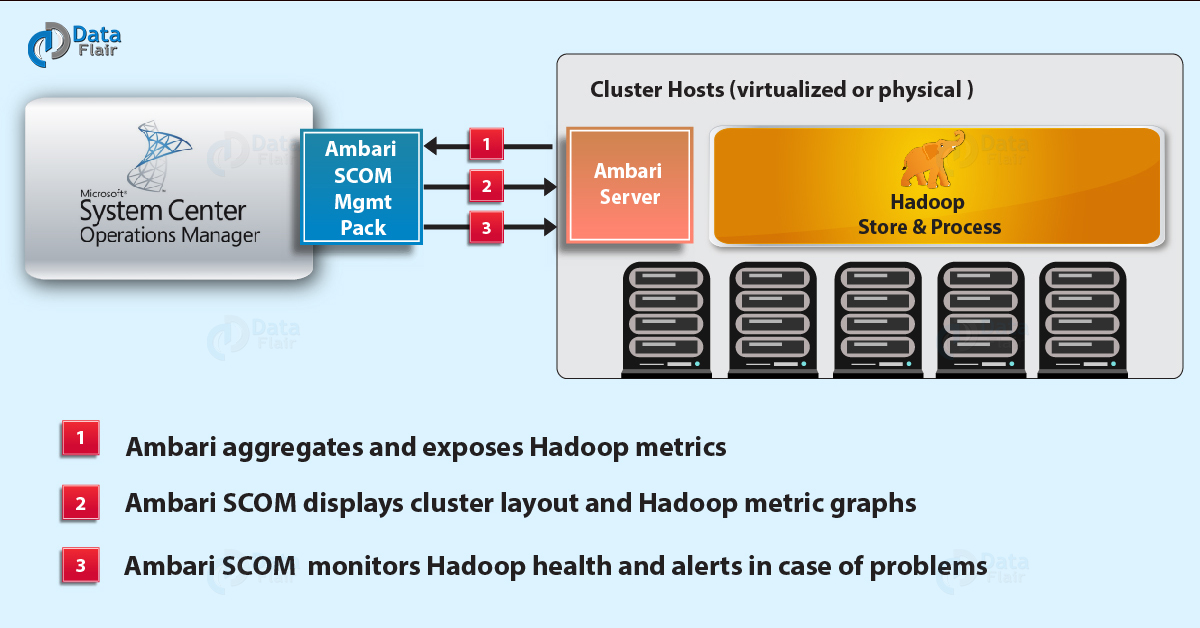
**2.9. Thrift**

* It is a software framework for scalable cross-language services development. Thrift is an interface definition language for RPC(Remote procedure call) communication. Hadoop does a lot of RPC calls so there is a possibility of using Hadoop Ecosystem componet Apache Thrift for performance or other reasons.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Apache-Thrift-Architecture-01.jpg)

### 2.14. Ambari

Ambari, another Hadop ecosystem component, is a management platform for provisioning, managing, monitoring and securing apache Hadoop cluster. Hadoop management gets simpler as Ambari provide consistent, secure platform for operational control.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/ambari-scom-01.jpg)

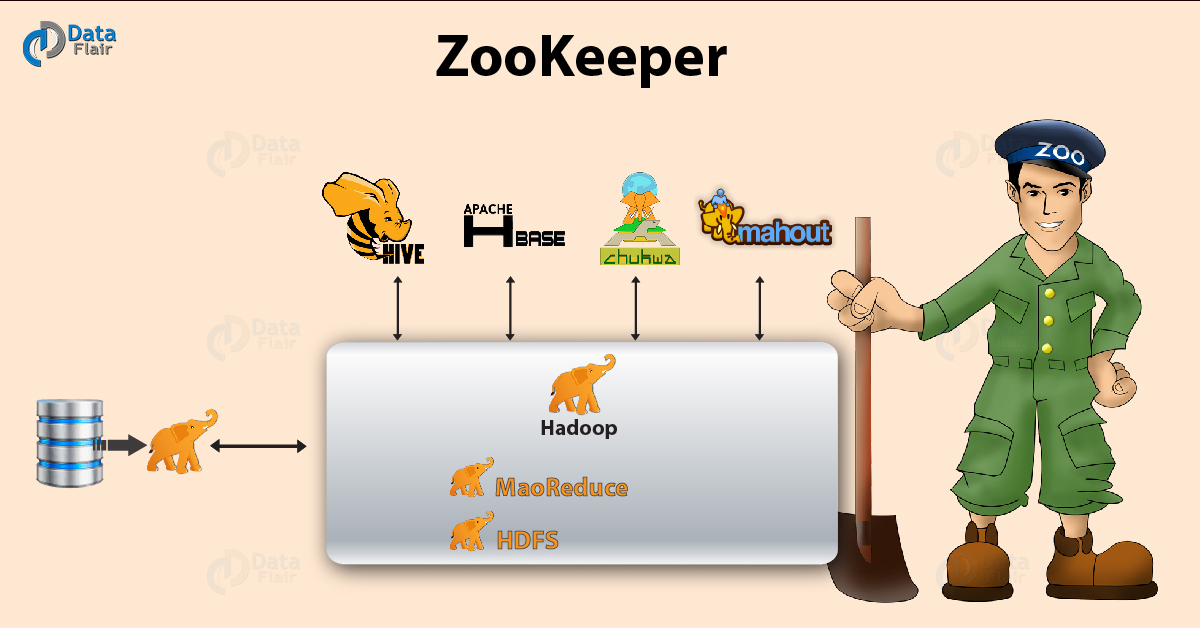
*Hadoop Ecosystem Tutorial – Ambari Diagram*

**Features of Ambari:**

* **Simplified installation, configuration, and management –** Ambari easily and efficiently create and manage clusters at scale.
* **Centralized security setup –**Ambari reduce the complexity to administer and configure cluster security across the entire platform.
* **Highly extensible and customizable –** Ambari is highly extensible for bringing custom services under management.
* **Full visibility into cluster health –** Ambari ensures that the cluster is healthy and available with a holistic approach to monitoring.

### 2.15. Zookeeper

**Apache Zookeeper** is a centralized service and a Hadoop Ecosystem component for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Zookeeper manages and coordinates a large cluster of machines.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/ZooKeeper-01.jpg)

*Hadoop Ecosystem Explained – ZooKeeper Diagram*

**Features of Zookeeper:**

* **Fast –** Zookeeper is fast with workloads where reads to data are more common than writes. The ideal read/write ratio is 10:1.
* **Ordered –** Zookeeper maintains a record of all transactions.

Fault Tolerance in Apache Spark

To achieve fault tolerance for all the generated RDDs, the achieved data is replicated among multiple Spark executors in worker nodes in the cluster. This results in two types of data that needs to be recovered in the event of failure:

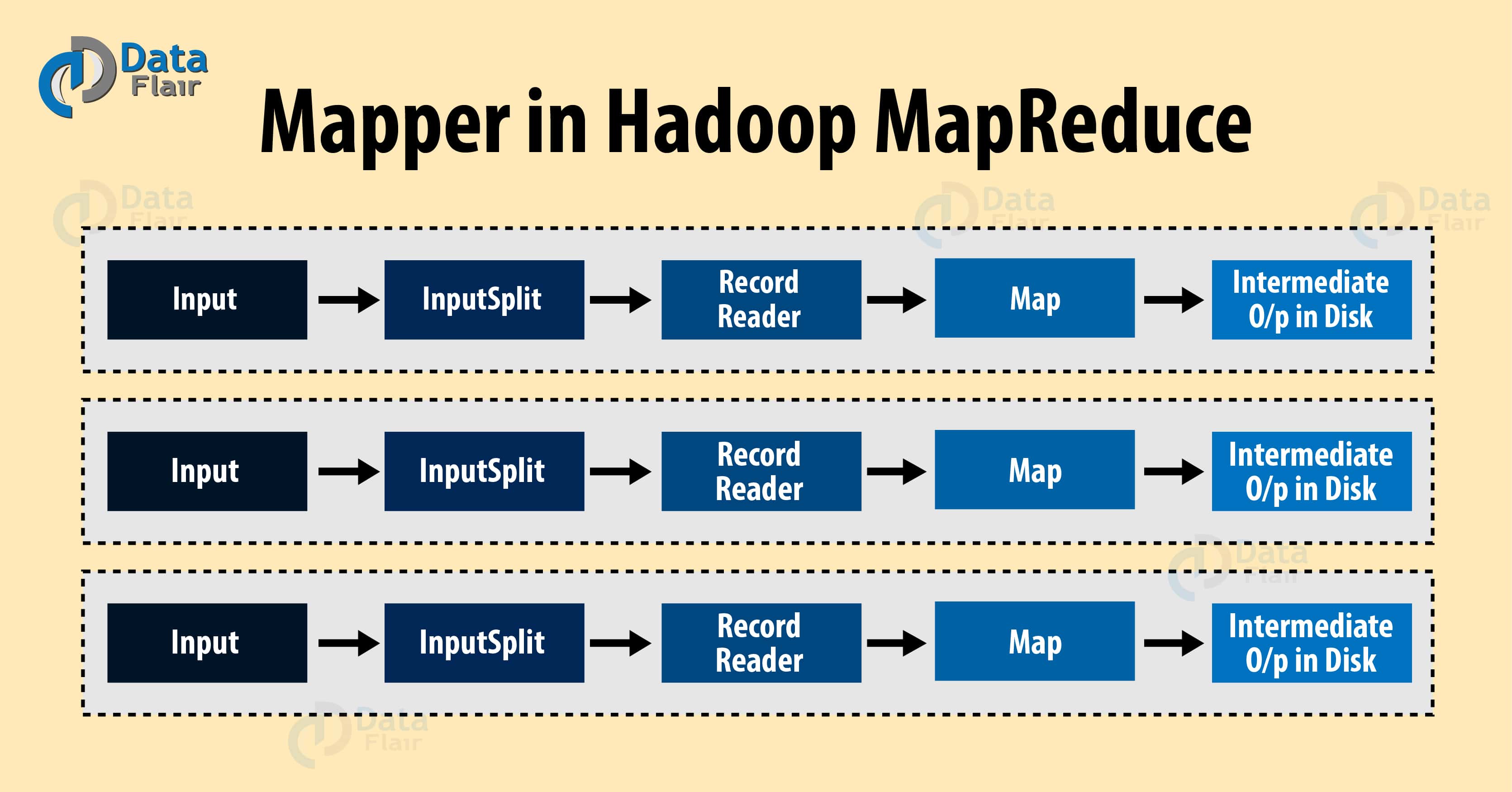
* **Data received and replicated –** In this, the data gets replicated on one of the other nodes thus the data can be retrieved when a failure.
* **Data received but buffered for replication –** The data is not replicated thus the only way to recover fault is by retrieving it again from the source.

Failure also occurs in worker as well as driver nodes.

* **Failure of worker node –** The node which runs the application code on the [**Spark cluster**](http://data-flair.training/blogs/apache-spark-installation-on-multi-node-cluster-step-by-step-guide/) is Spark worker node. These are the slave nodes. Any of the worker nodes running executor can fail, thus resulting in loss of [**in-memory**](http://data-flair.training/blogs/apache-spark-in-memory-computing/) If any receivers were running on failed nodes, then their buffer data will be lost.
* **Failure of driver node –** If there is a failure of the driver node that is running the Spark Streaming application, then SparkContent is lost and all executors with their in-memory data are lost(Killing).

## 1. Hadoop Mapper Tutorial – Objective

We will also discuss the number of mapper in Hadoop MapReduce for running any program and how to calculate the number of mappers required for a given data.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/introduction-to-hadoop-mapper.jpg)

*Hadoop Mapper*

## 2. What is Hadoop Mapper?

**Hadoop Mapper** task processes each input record and it generates a new <key, value> pairs. The <key, value> pairs can be completely different from the input pair. In mapper task, the output is the full collection of all these <key, value> pairs. Before writing the output for each mapper task, [**partitioning**](http://data-flair.training/blogs/partitioner-in-hadoop-mapreduce-hadoop-internals/) of output take place on the basis of the key and then [**sorting**](http://data-flair.training/blogs/shuffling-sorting-hadoop-mapreduce/) is done. This partitioning specifies that all the values for each key are grouped together.

MapReduce frame generates one map task for each InputSplit (we will discuss it below.) generated by the [**InputFormat**](http://data-flair.training/blogs/hadoop-inputformat/) for the job.

Mapper only understands <key, value> pairs of data, so before passing data to the mapper, data should be first converted into <key, value> pairs.

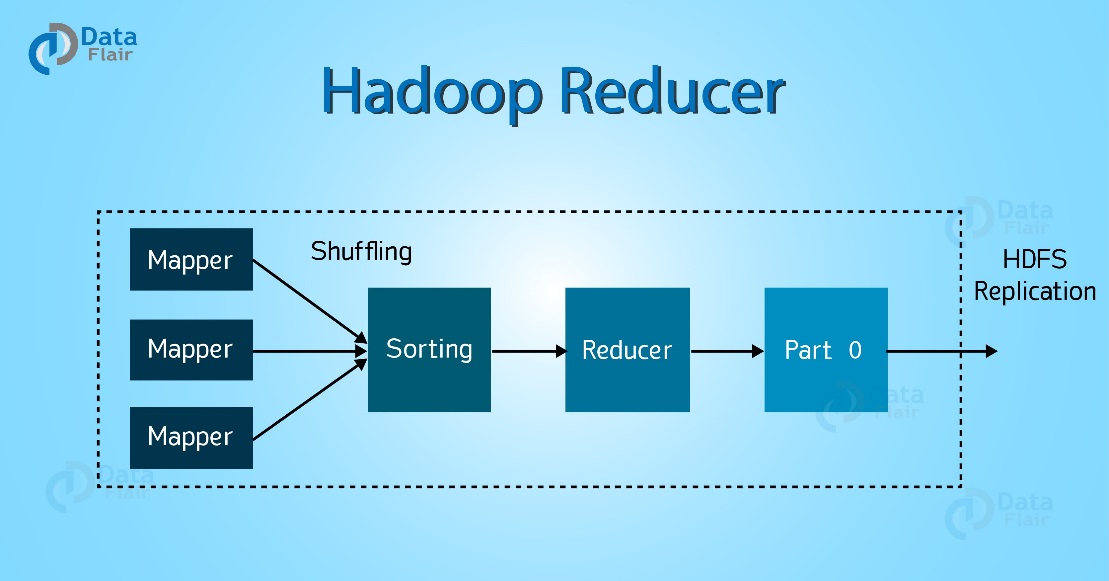
**Read:**[**What is MapReduce?**](https://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)

## 3. How is key value pair generated in Hadoop?

Let us now discuss the key-value pair generation in Hadoop.

* **InputSplit –**It is the logical representation of data. It describes a unit of work that contains a single map task in a MapReduce program. Learn about[InputSplit](http://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/)in detail[.](http://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/)
* **RecordReader –**It communicates with the InputSplit and it converts the data into key-value pairs suitable for reading by the Mapper. By default, it uses TextInputFormat for converting data into the key-value pair. RecordReader communicates with the Inputsplit until the file reading is not completed. Learn about[**RecordReader**](http://data-flair.training/blogs/recordreader-in-hadoop-mapreduce/)in detail[.](http://data-flair.training/blogs/recordreader-in-hadoop-mapreduce/)

Read [**Reducer in Hadoop**](https://data-flair.training/blogs/hadoop-reducer/) to have the knowledge of both mapper and reducer in hadoop.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/introduction-to-hadoop-reducer.jpg)

*Hadoop Reducer*

## 2. What is Hadoop Reducer?

Let’s now discuss what is Reducer in MapReduce first.

The Reducer process the output of the mapper. After processing the data, it produces a new set of output. At last [**HDFS**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/)stores this output data.

Hadoop Reducer takes a set of an intermediate [**key-value pair**](http://data-flair.training/blogs/key-value-pairs-hadoop-mapreduce/) produced by the mapper as the input and runs a Reducer function on each of them. One can aggregate, filter, and combine this data (key, value) in a number of ways for a wide range of processing. Reducer first processes the intermediate values for particular key generated by the map function and then generates the output (zero or more key-value pair).

One-one mapping takes place between keys and reducers. Reducers run in parallel since they are independent of one another. The user decides the number of reducers. By default number of reducers is 1.

**Read:**[**HDFS Combiner Tutorial**](https://data-flair.training/blogs/combiner-in-hadoop-mapreduce-advantages-disadvantages/)

## 3. Phases of MapReduce Reducer

**[Learn Hadoop from Industry Experts](https://data-flair.training/big-data-hadoop/" \t "_blank)**

As you can see in the diagram at the top, there are 3 phases of Reducer in Hadoop MapReduce. Let’s discuss each of them one by one-

### 3.1. Shuffle Phase of MapReduce Reducer

In this phase, the sorted output from the mapper is the input to the Reducer. In Shuffle phase, with the help of HTTP, the framework fetches the relevant partition of the output of all the mappers.

### 3.2. Sort Phase of MapReduce Reducer

In this phase, the input from different mappers is again sorted based on the similar keys in different Mappers. The shuffle and sort phases occur concurrently.

Learn [**Mapreduce Shuffling and Sorting Phase**](http://data-flair.training/blogs/shuffling-sorting-hadoop-mapreduce/) in detail.

**Read:**[**Features of HDFS**](https://data-flair.training/blogs/features-hadoop-hdfs-overview-beginners/)

**[Test Your Hadoop Knowledge](https://data-flair.training/blogs/category/quiz/hadoop-quiz/" \t "_blank)**

### 3.3. Reduce Phase

In this phase, after shuffling and sorting, reduce task aggregates the key-value pairs. The OutputCollector.collect() method, writes the output of the reduce task to the Filesystem. Reducer output is not sorted.

* **Map:** (K1, V1) -> list (K2, V2)
* **Reduce:** {(K2, list (V2 }) -> list (K3, V3)

**RACK AWARNESS**

In a large cluster of Hadoop, in order to improve the network traffic while [reading/writing HDFS file](http://data-flair.training/blogs/hadoop-hdfs-data-read-and-write-operations/), namenode chooses the datanode which is closer to the same rack or nearby rack to Read/Write request. Namenode achieves rack information by maintaining the rack id’s of each datanode. This concept that chooses closer datanodes based on the rack information is called Rack Awareness in Hadoop.

Rack awareness is having the knowledge of Cluster topology or more specifically how the different data nodes are distributed across the racks of a [Hadoop cluster](https://data-flair.training/blogs/installation-of-hadoop-3-x-on-ubuntu/). Default [Hadoop installation](http://data-flair.training/blogs/install-cloudera-hadoop-cdh5-ubuntu/) assumes that all data nodes belong to the same rack.

## 3. Why Rack Awareness?

In [**Big data**](http://data-flair.training/blogs/big-data-history-use-cases/)Hadoop, rack awareness is required for below reasons:

* To improve data[**high availability**](http://data-flair.training/blogs/hadoop-high-availability-tutorial/)and reliability.
* [**Improve the performance**](http://data-flair.training/blogs/hadoop-mapreduce-performance-tuning-best-practices/) of the cluster.
* To improve network bandwidth.
* Avoid losing data if entire rack fails though the chance of the rack failure is far less than that of node failure.
* To keep bulk data in the rack when possible.
* An assumption that in-rack id’s higher bandwidth, lower latency.

**File Formats**

Following are the Apache Hive different file formats:

* Text File
* Sequence File
* RC File
* AVRO File
* ORC File
* Parquet File

#### **Hive Text File Format**

if you do

create table g00103.iqp\_employee\_dim as select \* from g00103.emp where 1=0

----default will come text file format

If you do

Create table g00103.iqp\_employee\_dim like g00103.emp

----It will copy even the type of the file format

**Hive Text file format** is a default storage format. You can use the text format to interchange the data with other client application. The text file format is very common most of the applications. Data is stored in lines, with each line being a record. Each lines are terminated by a newline character (\n).

The text format is simple plane file format. You can use the compression (*BZIP2*) on the text file to reduce the storage spaces.

Create a TEXT file by add storage option as **‘STORED AS TEXTFILE’**at the end of a Hive CREATE TABLE command.

#### **Hive Sequence File Format**

**Sequence files** are Hadoop flat files which stores values in binary key-value pairs. The sequence files are in binary format and these files are able to split. The main advantages of using sequence file is to merge two or more files into one file.

Create a sequence file by add storage option as **‘STORED AS SEQUENCEFILE’** at the end of a Hive CREATE TABLE command.

#### **Hive RC File Format**

**RCFile** is row columnar file format. This is another form of Hive file format which offers high row level compression rates. If you have requirement to perform multiple rows at a time then you can use RCFile format.

The RCFile are very much similar to the sequence file format. This file format also stores the data as key-value pairs.

#### **Hive AVRO File Format**

**AVRO** is open source project that provides data serialization and data exchange services for Hadoop. You can exchange data between Hadoop ecosystem and program written in any programming languages. Avro is one of the popular file format in Big Data Hadoop based applications.

#### **Hive ORC File Format**

The **ORC file** stands for Optimized Row Columnar file format. The ORC file format provides a highly efficient way to store data in Hive table. This file system was actually designed to overcome limitations of the other Hive file formats. The Use of ORC files improves performance when Hive is reading, writing, and processing data from large tables.

#### **Hive Parquet File Format**

**Parquet** is a column-oriented binary file format. The parquet is highly efficient for the types of large-scale queries. Parquet is especially good for queries scanning particular columns within a particular table. The Parquet table uses compression Snappy, gzip; currently Snappy by default.

* TBLPROPERTIES ("comment"="table\_comment")
* TBLPROPERTIES ("hbase.table.name"="table\_name") – see [HBase Integration](https://cwiki.apache.org/confluence/display/Hive/HBaseIntegration#HBaseIntegration-Usage).
* TBLPROPERTIES ("immutable"="true") or ("immutable"="false") in release 0.13.0+ ([HIVE-6406](https://issues.apache.org/jira/browse/HIVE-6406)) – see [Inserting Data into Hive Tables from Queries](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML#LanguageManualDML-InsertingdataintoHiveTablesfromqueries).
* TBLPROPERTIES ("orc.compress"="ZLIB") or ("orc.compress"="SNAPPY") or ("orc.compress"="NONE") and other ORC properties – see [ORC Files](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC#LanguageManualORC-HiveQLSyntax).
* TBLPROPERTIES ("transactional"="true") or ("transactional"="false") in release 0.14.0+, the default is "false" – see [Hive Transactions](https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions#HiveTransactions-TableProperties).
* TBLPROPERTIES ("NO\_AUTO\_COMPACTION"="true") or ("NO\_AUTO\_COMPACTION"="false"), the default is "false" – see [Hive Transactions](https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions#HiveTransactions-TableProperties).
* TBLPROPERTIES ("compactor.mapreduce.map.memory.mb"="mapper\_memory")

SerDe

| **Syntax** | **Equivalent** |
| --- | --- |
| **Syntax** | **Equivalent** |
| STORED AS AVRO /  STORED AS AVROFILE | ROW FORMAT SERDE    'org.apache.hadoop.hive.serde2.avro.AvroSerDe'    STORED AS INPUTFORMAT    'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'    OUTPUTFORMAT    'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat' |
| STORED AS ORC /  STORED AS ORCFILE | ROW FORMAT SERDE    'org.apache.hadoop.hive.ql.io.orc.OrcSerde'    STORED AS INPUTFORMAT    'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'    OUTPUTFORMAT    'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat' |
| STORED AS PARQUET /  STORED AS PARQUETFILE | ROW FORMAT SERDE    'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'    STORED AS INPUTFORMAT    'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'    OUTPUTFORMAT    'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat' |
| STORED AS RCFILE | STORED AS INPUTFORMAT    'org.apache.hadoop.hive.ql.io.RCFileInputFormat'    OUTPUTFORMAT    'org.apache.hadoop.hive.ql.io.RCFileOutputFormat' |
| STORED AS SEQUENCEFILE | STORED AS INPUTFORMAT    'org.apache.hadoop.mapred.SequenceFileInputFormat'    OUTPUTFORMAT    'org.apache.hadoop.mapred.SequenceFileOutputFormat' |
| STORED AS TEXTFILE | STORED AS INPUTFORMAT    'org.apache.hadoop.mapred.TextInputFormat'    OUTPUTFORMAT    'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' |

Serde

Hive

1.SERDE

createtab\_stmt

CREATE EXTERNAL TABLE `iqp\_baan\_orders\_\_x`(

`sor\_id` string COMMENT '',

`order\_number` string COMMENT '',

`requisition\_number` string COMMENT '',

`order\_line\_number` string COMMENT '',

`customer\_po\_number` string COMMENT '')

ROW FORMAT SERDE

'org.apache.hadoop.hive.serde2.RegexSerDe'

WITH SERDEPROPERTIES (

'input.regex'='(.{10})(.{30})(.{30})(.{10})(.{30})(.{10})(.{10})(.{10})(.{10})(.{10})(.{50})(.{50})(.{50})(.{50})(.{50})(.{20})(.{20})(.{5})(.{10})(.{10})(.{50})(.{50})(.{50})(.{50})(.{50})(.{20})(.{20})(.{5})(.{10})(.{10})(.{50})(.{50})(.{20})(.{50})(.{50})(.{50})(.{50})(.{20})(.{50})(.{20})(.{50})(.{20})(.{50})(.{20})(.{50})(.{20})(.{50})(.{30})(.{30})(.{30})(.{30})(.{10})(.{30})(.{15})(.{30})(.{30})(.{30})(.{30})(.{30})(.{30})(.{30})(.{10})(.{1})(.{10})(.{30})(.{30})(.{15})(.{10})(.{10})(.{10})(.{10})(.{10})(.{10}) (.{3})([^\\t]\*).\*',

'output.format.string'='%1$S %2$S %3$S %4$S %5$S %6$S %7$S %8$S %9$S %10$S %11$S %12$S %13$S %14$S %15$S %16$S %17$S %18$S %19$S %20$S %21$S %22$S %23$S %24$S %25$S %26$ %27$S %28$S %29$S %30$S %31$S %32$S %33$S %34$S %35$S %36$S %37$S %38$S %39$S %40$S %41$S %42$S %43$S %44$S %45$S %46$S %47$S %48$S %49$S %50$S %51$S %52$S %53$ %54$S %55$S %56$S %57$S %58$S %59$S %60$S %61$S %62$S %63$S %64$S %65$S %66$S %67$S %68$S %69$S %70$S %71$S %72$S %73$S %74$S %75$S')

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

'hdfs://alphd1px001/tmp/iqp/externalfiles/iqp\_baan\_orders\_\_x'

TBLPROPERTIES (

'numFiles'='1',

'totalSize'='10080',

'transient\_lastDdlTime'='1527252077')

Interview

**5) What is the use of Hcatalog?**

Hcatalog can be used to share data structures with external systems. Hcatalog provides access to hive metastore to users of other tools on Hadoop so that they can read and write data to hive’s data warehouse.

**11) How will you read and write HDFS files in Hive?**

i) TextInputFormat- This class is used to read data in plain text file format.

ii) HiveIgnoreKeyTextOutputFormat- This class is used to write data in plain text file format.

iii) SequenceFileInputFormat- This class is used to read data in hadoop SequenceFile format.

iv) SequenceFileOutputFormat- This class is used to write data in hadoop SequenceFile format.

**12) What are the components of a Hive query processor?**

Query processor in Apache Hive converts the SQL to a graph of MapReduce jobs with the execution time framework so that the jobs can be executed in the order of dependencies. The various components of a query processor are-

* Parser
* Semantic Analyser
* Type Checking
* Logical Plan Generation
* Optimizer
* Physical Plan Generation
* Execution Engine
* Operators
* UDF’s and UDAF’

DESCRIBE vs DESCRIBE EXTENDED

DESCRIBE—only columns and type

EXTENDED-show create table

**16)  What is the use of explode in Hive?**

Explode in Hive is used to convert complex data types into desired table formats. explode UDTF basically emits all the elements in an array into multiple rows.

**18) Difference between HBase and Hive.**

* HBase is a NoSQL database whereas Hive is a data warehouse framework to process Hadoop jobs.
* HBase runs on top of HDFS whereas Hive runs on top of Hadoop MapReduce.

**17) Explain about SORT BY, ORDER BY, DISTRIBUTE BY and CLUSTER BY in Hive.**

SORT BY – Data is ordered at each of ‘N’ reducers where the reducers can have overlapping range of data.

DISTRUBUTE BY – It is used to distribute the rows among the reducers. Rows that have the same distribute by columns will go to the same reducer.

CLUSTER BY- It is a combination of DISTRIBUTE BY and SORT BY where each of the N reducers gets non overlapping range of data which is then sorted by those ranges at the respective reducers.

**31) What are the different components of a Hive architecture?**

Hive Architecture consists of a –

* User Interface – UI component of the Hive architecture calls the execute interface to the driver.
* Driver create a session handle to the query and sends the query to the compiler to generate an execution plan for it.
* Metastore - Sends the metadata to the compiler for the execution of the query on receiving the sendMetaData request.
* Compiler- Compiler generates the execution plan which is a DAG of stages where each stage is either a metadata operation, a map or reduce job or an operation on HDFS.
* Execute Engine- Execution engine is responsible for submitting each of these stages to the relevant components by managing the dependencies between the various stages in the execution plan generated by the compiler.

1. How will you optimize Hive performance?

There are various ways to run Hive queries faster -

* Using Apache Tez execution engine
* Using vectorization
* Using ORCFILE
* Do cost based query optimization.

APACHE TEZ

Apache Tez is more widely adopted and used in production than Spark, if you consider the Hadoop tools that are most widely used in production.  
  
Apache Hive (starting from version 0.13) and Pig (starting from version 0.14)  have released versions that run on Apache Tez as the execution engine. These are being used in production. Tez has proven itself at large scale workloads. Checkout the study done by Yahoo at really large scale –

· Spark is good for in-memory processing while Tez works great for running query on partition tables

## **3. Why Bucketing?**

Basically, the concept of [**Hive Partitioning**](https://data-flair.training/blogs/apache-hive-partitions/) provides a way of segregating hive table data into multiple files/directories. However, it only gives effective results in few scenarios. Such as:  
  
– When there is the limited number of partitions.  
– Or, while partitions are of comparatively equal size.

Although, it is not possible in all scenarios. For example when are partitioning our tables based geographic locations like country. Hence, some bigger countries will have large partitions (ex: 4-5 countries itself contributing 70-80% of total data). While small countries data will create small partitions (remaining all countries in the world may contribute to just 20-30 % of total data). Hence, at that time Partitioning will not be ideal.

Then, to solve that problem of over partitioning, Hive offers Bucketing concept. It is another effective technique for decomposing table data sets into more manageable parts.

iii. Moreover,  to divide the table into buckets we use CLUSTERED BY clause.

MAP JOINS

Although even if queries frequently depend on small table joins, usage of map joins speed up queries’ execution. Moreover, it is the type of join where a smaller table is loaded into memory and the join is done in the map phase of the MapReduce job.

Basically, before the original [**MapReduce**](https://data-flair.training/blogs/hadoop-mapreduce-tutorial/) task, its first step is to create a MapReduce local task. However, from [**HDFS**](https://data-flair.training/blogs/hadoop-hdfs-tutorial/) this map/reduce task read data of the small table. Further save it into an in-memory hash table, then into a hash table file. Afterward, it moves the hash table file to the [**Hadoop**](https://data-flair.training/blogs/install-and-configure-hadoop-2-7-x-on-ubuntu/) Distributed Cache while original join MapReduce task starts, which will populate the file to each mapper’s local disk. Hence, in this way, all the mapper can load this hash table file into the memory and then do the join in Map stage.

So, let’s understand this with an example. let’s suppose, for a join with big table A and small table B, for every mapper for table A, Table B is read completely. Since the smaller table is loaded into memory at first. Afterward, join is performed in the map phase of the MapReduce job, no reducer is needed and reduce phase is skipped. However, map joins in Hive are way faster than the regular joins since no reducers are necessary.

set hive.auto.convert.join=true;

set hive.auto.convert.join.noconditionaltask=true; -- if the task is so small then merge many in one

Using hint

select /\*+ MAPJOIN(a) \*/ a.\* from passwords a, passwords2 b where a.col0=b.col0 ;

Bucket Map Join

Hence, onto each mapper, only the matching buckets of all small tables are replicated. As a result of this, the efficiency of the query is improved drastically. However, make sure data is not sorted in a bucket map join.

To be more specific we use this feature with several scenarios. Like:

i. While all tables are large.  
ii. Also, while all tables are bucketed using the join columns.  
iii. Moreover, while The number of buckets in one table is a multiple of the number of buckets in the other table.  
iii. Also, when all tables are not sorted.

## **Skew Join – Use Case**

* Basically, on the joining column, one table has huge skew values.

However, let’s assume if table A join B, and A has skew data “1” in joining column.  
At First store, the rows with key 1 in an in-memory hash table and read B. Further to read A  run a set of mappers. Afterward, do the following:

* Make sure use the hashed version of B to compute the result since it has key 1.
* Then, send all other keys to a reducer which does the join. Basically, from a mapper, this reducer will get rows of B also.