## List two primary differences between Hadoop version-2 and Hadoop version-3

Solution:

Any two difference among these is correct-

Major difference :

In Hadoop 2 Fault tolerance is ensured using **Replication** whereas in Hadoop-3 it is ensured using **Erasure Coding**. ( Due to replication overhead is 200 percent in Hadoop-2 and due to erasure coding, Hadoop-3 overhead is 50% only).

Hadoop 2 supports one standby Name-Node whereas Hadoop-3 supports **multiple standby Name-Nodes** .

Other differences (that shall be considered)

For Haddop-2 , Java-7 is the minimum compatible version whereas for Hadoop-3 it is **Java -8**.

For balancing disks load inside a Data-Node **Intra data node balancer** is provisioned in Hadoop-3. Where as Hadoop-2 only allows Inter data node balancing using HDFS balancer.

Hadoop-3, **Improve the YARN time line service** - with improving scalability and reliability of this service. In Hadoop-2 YARN Timeline Service has scalability issues

**Improved Scalability** :Hadoop-2 can scale up to 10 thousands cluster nodes where Hadoop-3 can have more than 10 thousands nodes in the cluster

Hadoop 3 has additional library **support for Amazon S3**.

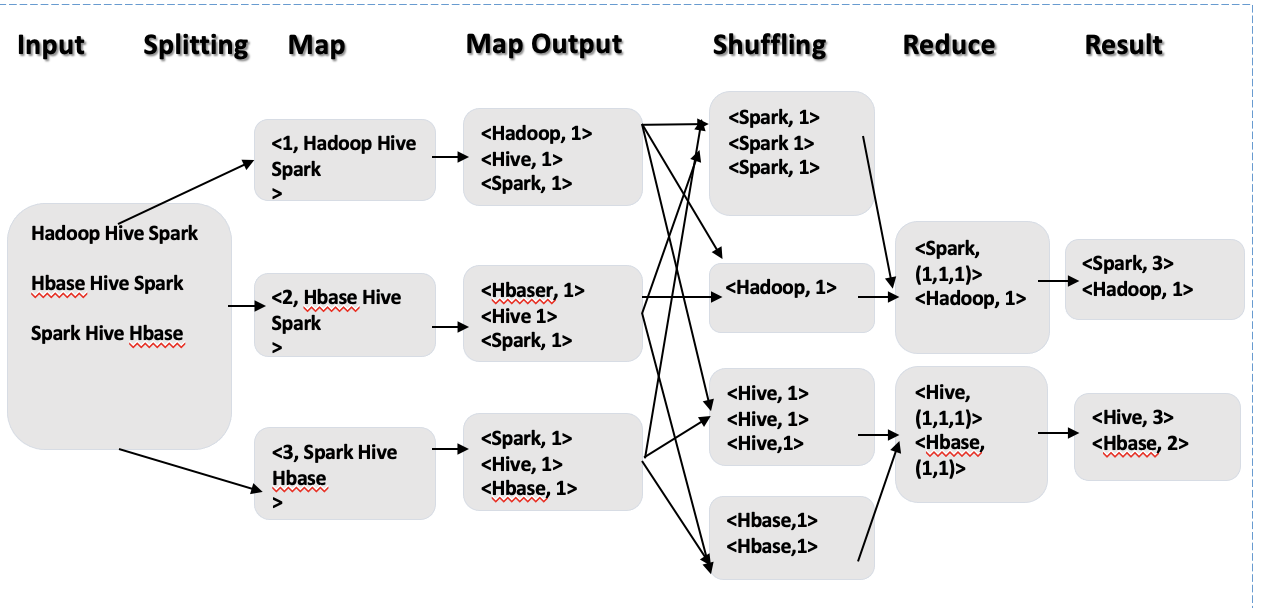
In Haddop-2 **Linux ephemeral port range** (32768-61000)Is being used as default which would sometimes fail to bind to the port due to a conflict with another application. In Hadoop-3 These **conflicting ports have been moved out** of the ephemeral range

## What is Hive metastore? Can NoSQL Database- HBase can be configured as hive metastore?

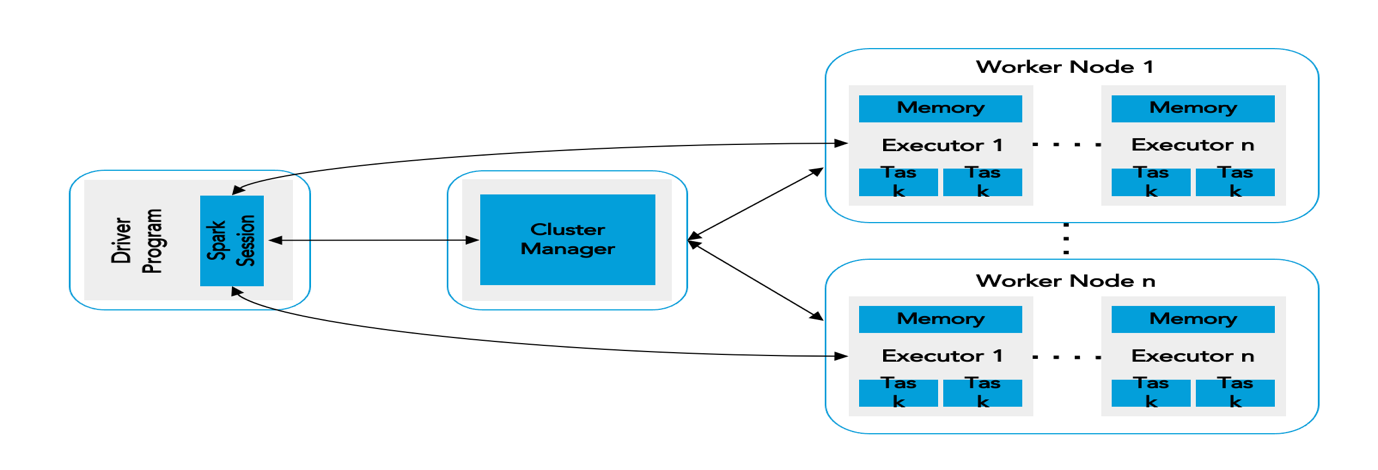
Hive-Metastore is the central repository of Hive Metadata. It is service that mainly **stores the meta data for Hive tables** and relations. For example, Schema and Locations etc.

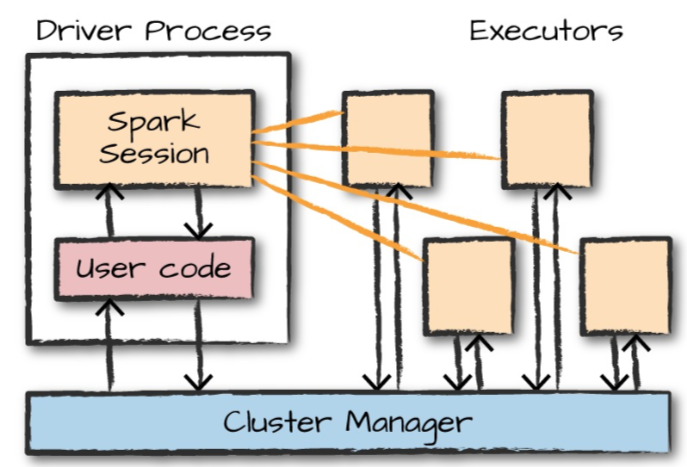
**Yes. HBases can be configured as Hive -Metastore.** In fact, HBase is preferred metastore service for highly scalable production systems.

## **Using an example, depict how MapReduce computes word count**.



## Draw and explain various components of Spark architecture.

Any of the below figure type would be fine



Components of Spark Architecture are :

1. **Driver program/process** :

* Is the central co-ordinator,
* Runs main function,
* Manage/Maintain information about spark application,
* Respond to user program/input,
* Analyze/schedule/distribute work across executor,
* listen for and accept incoming connections from its executors

1. **Executor Process**:

* Executes tasks/code that driver assigns them,
* Reports execution state back to driver

1. **Cluster Manager**

* Keeps track of available resources
* Allocates Worker Nodes
* Supported cluster managers are – Hadoop YARN, Apache Mesos, Kubernetes and Standalone

1. **Worker Node**

* Physical machine on which executor runs,
* one or more Executor runs on Worker Node.

## What is cap theorem ? Where does MongoDB stands in cap theorem

CAP theorem says, A distributed database system can only have 2 of the below 3 properties:

* 1. **Consistency** : All nodes/clients see the same data at the same time. *read* operation will return the value of the most recent *write.* entire transaction gets rolled back if there is an error during any stage in the process.
  2. **Availability** : Every request gets a response on success/failure. the system remains operational 100% of the time.
  3. **Partition Tolerance** : A system that is partition-tolerant can sustain any amount of network failure that doesn’t result in a failure of the entire network. Data records are sufficiently replicated across combinations of nodes and networks to keep the system up through intermittent outages.

MongoDB is a CP data store—it resolves network partitions by maintaining consistency, while compromising on availability

# Section B

## Write HDFC shell commands for the following

To Print Version of installed Hadoop

* **hadoop version**

To Copy ‘file1.txt’ from ‘InputDir’ to ‘OutputDir’ as file2.txt

* **hadoop fs -cp InputDir /file1.txt OutputDir/file2.txt**

To Delete an empty directory named as **XYZ.**

* **hadoop fs -rmdir XYZ**

To list the contents of folder named **SampleDir.**

* **hadoop fs -ls SampleDir**

To fetch the usage instructions of **mkdir** command

* **hadoop fs -help mkdir**

1. **Write a Spark program psudocode to load a textile named as text.txt as spark RDD and compute its wordcounts.**

sc = SparkContext("local)

# read data from text file and split each line into words

words = sc.textFile("**test.txt**".flatMap(lambda line: line.split(" "))

# count the occurrence of each word

wordCounts = words.map(lambda word: (word, 1).reduceByKey(lambda a,b: a +b)

# save the counts to output

wordCounts.saveAsTextFile("output")

1. **Two hive tables are shown below. Write a hive query to perform an inner join on the Table1 and Table 2 on ‘id’ column**

SELECT t1.Id, t1.NAME, t2.Name

FROM Table1 t1 JOIN Table2 t2

ON (t1.Id = t2.Id);

Or

SELECT Table1.\*, Table1.\*  
FROM Table1 JOIN Table2 ON (Table1.Id = Table2.id);

Output order will be based on the query though should have these records

Joe,2,Tie

Hank,4, Coat

## Write commands/query in MongoDB to

1. Create a collection named **orders**.

* db.createCollection(' **orders** ')

1. Insert below record in orders.

{"order\_id”: 1,

"order\_date”: '2013-07-25 00:00:00.0',

"order\_customer\_id”: 11599,

"order\_status”: "CLOSED" }

* **db.orders.insertOne(**

**{**

**"order\_id" : 1,**

**"order\_date" : '2013-07-25 00:00:00.0',**

**"order\_customer\_id" : 11599,**

**"order\_status" : "CLOSED"**

**}**

1. Fetch orders with **order\_status** as COMPLETE.

* db.orders.aggregate(

[

{$match: {order\_status: "COMPLETE"}}

]

1. Compute count of orders with status COMPLETE and CLOSED.

* db.orders.aggregate(

[

{$match : {$or: [{order\_status: {$eq: "COMPLETE"}}, {order\_status: {$eq: "CLOSED"}}]} },

{$group : {\_id: {"status": "$order\_status"}, "count": {$sum: 1}}}

]

)