Number of Book
II nd used

TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE & TECHNOLOGY
EXAMINATION BOARD
KIRTIPUR

Copy No. 11784

Code No.:

Book I

Code No.:

Students are required to write their answers on both sides & a margin of about 1¼ inches should be left on each page.

MARKS OBTAINED

1st Q		11th Q	
2nd Q		12th Q	
3rd Q		13th Q	
4th Q		14 th Q	
5th Q		15th Q	
6th Q		16th Q	
7th Q		17th Q	
8th Q		18th Q	
9th Q		19th Q	
10thQ		20th Q	

TOTAL 39-5

INSTRUCTIONS TO CANDIDATES

[1] Each Candidate will write legibly on the title page his or her ROLL NO. REGISTERED NO. AND THE SCRIPT in which answers are written but not his or her name and the name of the campus from which he or she appears. This should be done in each answer - book used before beginning to write inside.

[2] No loose papers will be provided for scribbling and no other paper should be brought in for this purpose. Any candidate found with loose paper in his or her possession WILL BE EXPELLED. All work must be done in the book provided and pages MUST NOT BE TORN OUT. The book provided CANNOT BE REPLACED BY ANOTHER but, if necessary, an additional book will be given. All work intended for the examiner must be written ON BOTH SIDES of the paper. Anything cancelled will not be looked into. Should a torn leaf be discovered inside an answer book, it should not be removed but crossed out and folded after bringing it to the notice of the invigilator.

[3] Candidate is forbidden to write answers or anything else on the question - paper.

[4] No Candidate will be allowed to leave the room until one hour has passed from the time when the papers are distributed.

[5] Candidate who uses two or more answer-books should see, before handing over to the invigilator, that they are properly stitched together.

INSTRUCTIONS TO THE EXAMINER

1. Aggregate marks of each question should be placed in the given appropriate box.
2. Marks for parts of a qusetion should be totalled under the question inside the answer- book.

Level Masters

Year/Part/Sem. II | I | III

Centre SMS TV

Roll No. 06

Roll No. In Words Six

Arpan Saphota

Reg. No.

Subject Techniques for Bio. Data

Paper

Script English.

Date 2080 | 12 | 15

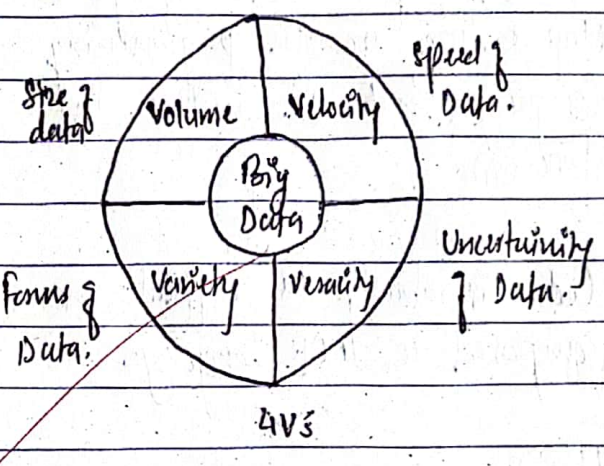
Signature of Scrutiny Board

Signature of Examiner

Group 'A'

1. The different characteristics of big data are:-

(i) Volume
(ii) Velocity
(iii) Variety
(iv) Veracity.



(i) Volume.

In Big data, Volume is one of the key things which tells that the data is the big data. Volume simply means the size of the data.

2.5 Example?

(ii) Velocity

Velocity in big data means the speed of data that is being generated. As the times increases the data also increases due to the velocity characteristics of data.

(iii) Variety

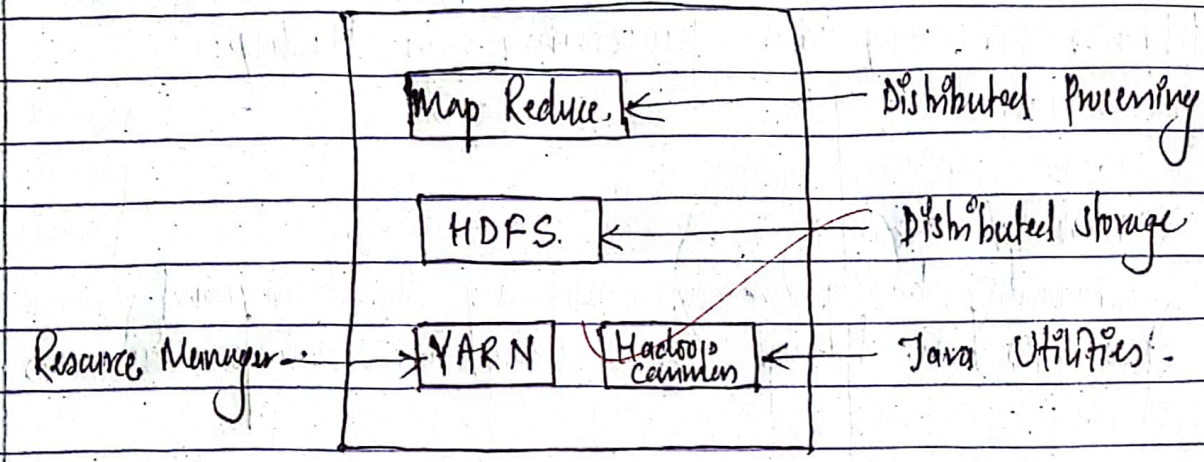
There are different forms of data in Big Data. Structured, Unstructured and semi-structured data are the broadly classified forms of data which is the data variety.

(iv) Veracity.

Veracity is the Uncertainty of the data. It is the accuracy of the data in Big Data.

2.	Functional Programming language	Object-Oriented Programming Language.
(i)	Functional programming Languages in Map Reduce are the programming Language that are written in functions.	Object-Oriented programming Language in Map Reduce are the programming that are written in terms of Objects and classes.
(ii)	Code reusability is difficult as compared to OOP Language.	Code reusability is not difficult as compared to functional programming language.
(iii)	Functional programming Language is a bit old concept and it is difficult to understand.	Object-Oriented programming Language is the new concept and it is easier to understand.
(iv)	It has functions without looping statements.	It has objects, classes, function, loops, and many more.
(v)	LISP is an example of functional programming in MR.	Java is an example of OOP in Map Reduce.

3. The core components of Hadoop



(i) Map Reduce

Map Reduce is one of the core component of Hadoop, where the distributed processing of Big Data is done. It has two components i.e. mapper() and reducer(). The output of mapper will be the input of reducer and it generates the simplified output.

(ii) HDFS

Hadoop Distributed File System (HDFS) is the distributed storage of big data which has Name Node, Secondary Name Node and Data Node. The Name Node stores the metadata and Data Node stores data in block.

(iii) YARN

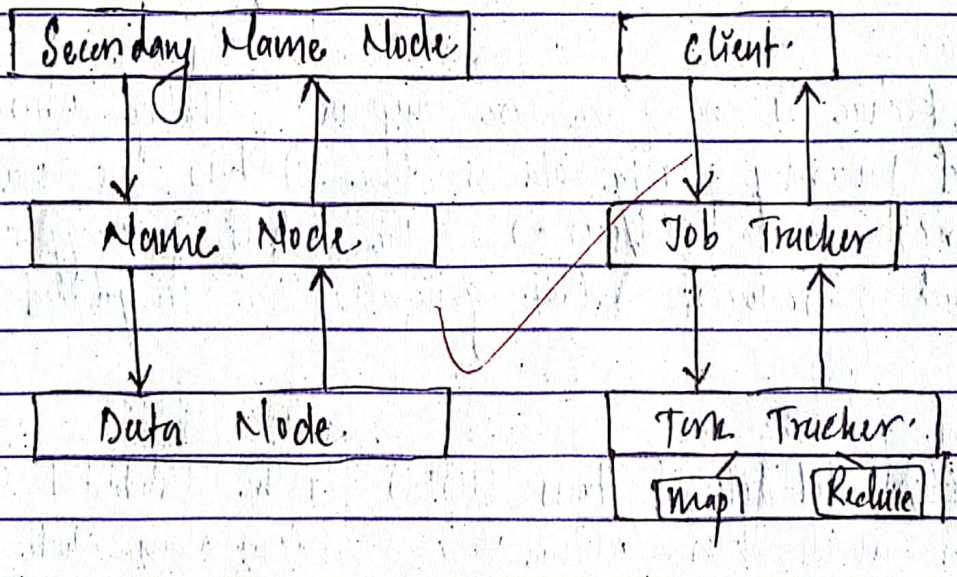
Yet Another Resource Negotiator is the resource manager in Big data Hadoop. It has Node Manager, Resource Manager, Application Master and container, All works actively in coordination to manage the hadoop resources.

(iv) Hadoop Common.

Hadoop Common is also one of the core component where Java utilities are present and we can use the Java libraries in Big Data processing in Hadoop.

4. "running Hadoop"

Running hadoop means, running the background processes in hadoop i.e. executing the hadoop elements.



While running the hadop, its following elements will be running in background.

(i) Secondary Name Node.
(ii) Name Node
(iii) Data Node
(iv) Job Tracker
(v) Task Tracker.

i.e. In Combine, we can visualize the daemons in the below figure.

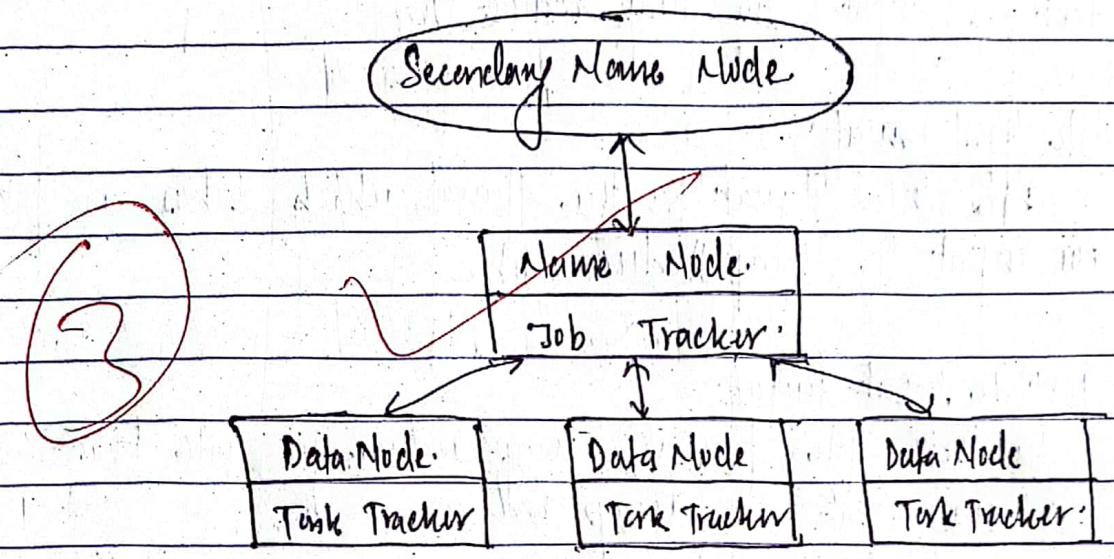


fig: running Hadoop in Big Data Cluster.

5. The different map reduce input and output formats are:

(i) TextInputFormat
(ii) FileInputFormat
(iii) TextFileInputFormat
(iv) TextSequenceInputFormat
(v) FileSequenceInputFormat
(vi) KeyInputFormat
(vii) TextOutputFormat
(viii) FileOutputFormat

(i) Text Input Format.

It is a text input format in Map Reduce where the text is inputed in map reduce job.

(ii) File Input Format.

File input format as the format, which takes as file as an input in Map Reduce Job.

(iii) Textfile Input format

In this file format, map reduce job will take text file as an input in map reduce.

(iv) Text Output Format

Text Output format is the output format, which is obtained as the text output from the map reduce job.

(v) File Output Format.

In this output format, map reduce job will results the file as output.

Similarly we have many other input and output formats in map reduce, which works accordingly. We can use any input and output formats as per our requirements.

Group 'B'

10. Some examples of HDFS commands.

(i) List.
bin/hdfs dfs -ls /hdfs/test

(ii) Copy from local.
bin/hdfs dfs -copyfromlocal /Desktop/test /user/hdfs/test.

(iii) Copy to local
bin/hdfs dfs -copyTolocal /user/hdfs/test /Desktop/test

(iv) rmr (remove recursively)
bin/hdfs dfs -rmr /users/hdfs/test.

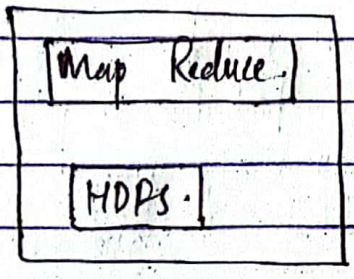
(v) mv (move).
bin/hdfs dfs -mv /users/hdfs/test1 /users/hdfs/test2.

(vi) cp (copy)
bin/hdfs dfs -cp /users/hdfs/test5 /users/hdfs/test10

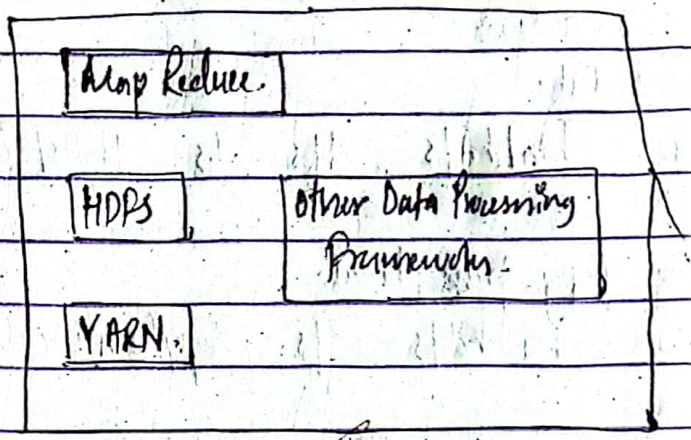
Similarly we have other many commands in HDFS.

Single point of failure in Hadoop version 1 is the Job Tracker component which is single and there was bottleneck in this component to manage the resources between the client and the Nodes. So, here if the Job tracker fails then everything fails. Thus, Hadoop V2.0 introduced.

Hadoop v. 1.0

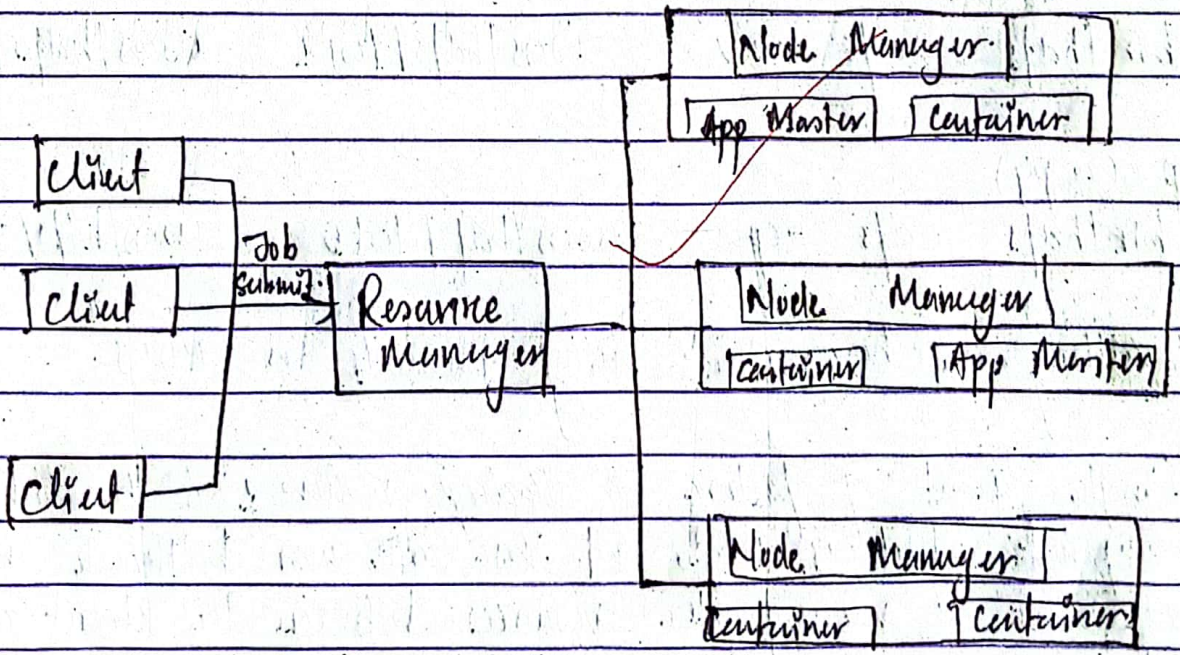


Hadoop v. 2.0



Here, In Hadoop v. 2.0, we have YARN as a resource manager and it manages the resources to avoid the single point of failure.

In Hadoop v. 2.0 with YARN.



Here,
Client will submit Job to the Resource Manager
Node Manager will give the status of Node to Resource Manager.
Application Master will give the resource status to Resource Manager
Container will update about the maps reduce task to the Application Master.

In this way the single point of failure (SPOF) in Hadoop V. 1.0 due to single Job Tracker is eliminated with the help of Resource Manager from YARN in Hadoop V. 2.0

8. The different configuration modes to setup Hadoop are:-

(i) Pseudo Distributed Mode
→ Hadoop clusters are falsely distributed in configuration xml file of Hadoop.

(ii) Local standalone ~~Distributed~~ Mode
→ Default mode.
→ Empty configuration xml file

(iii) Fully Distributed Mode:
→ Need to configure xml configuration file.
→ Master - master node hosts Name Nodes, Job Trackers
→ Slaves - hadoop1, hadoop2, hadoop3, ...

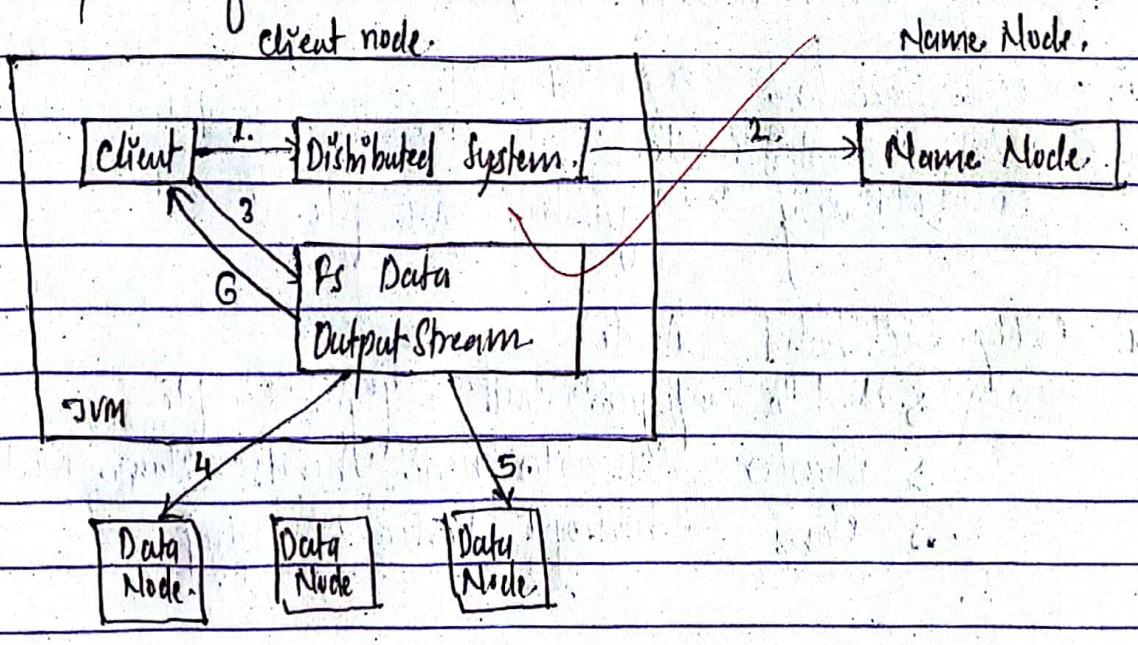
for development mode. Pseudo Distributed Mode is preferred because, this mode can be easily setup as compared to fully Distributed mode. And we can perform any operation of Hadoop tools and technologies in this mode, whereas in Standalone mode we can not experience about the distributed processing. But in pseudo Distributed mode, we can experience distributed storage, processing in our development machine. Also the infrastructure is not that required in pseudo distributed mode. Only one single machine with hadoop compatible is enough for the development in hadoop.

Therefore, pseudo Distributed mode is preferred for the development in hadoop.

Hadoop Streaming at last (RTO)

7. Anatomy of Read and Write Operation of the HDFS.

Read Operation of HDFS



1. Open.
Here client open the file in Distributed file system.

2. Get blocks location.
Distributed system will get the block location from the Name Node.

3. Read
client will recieve the data from file system output stream.

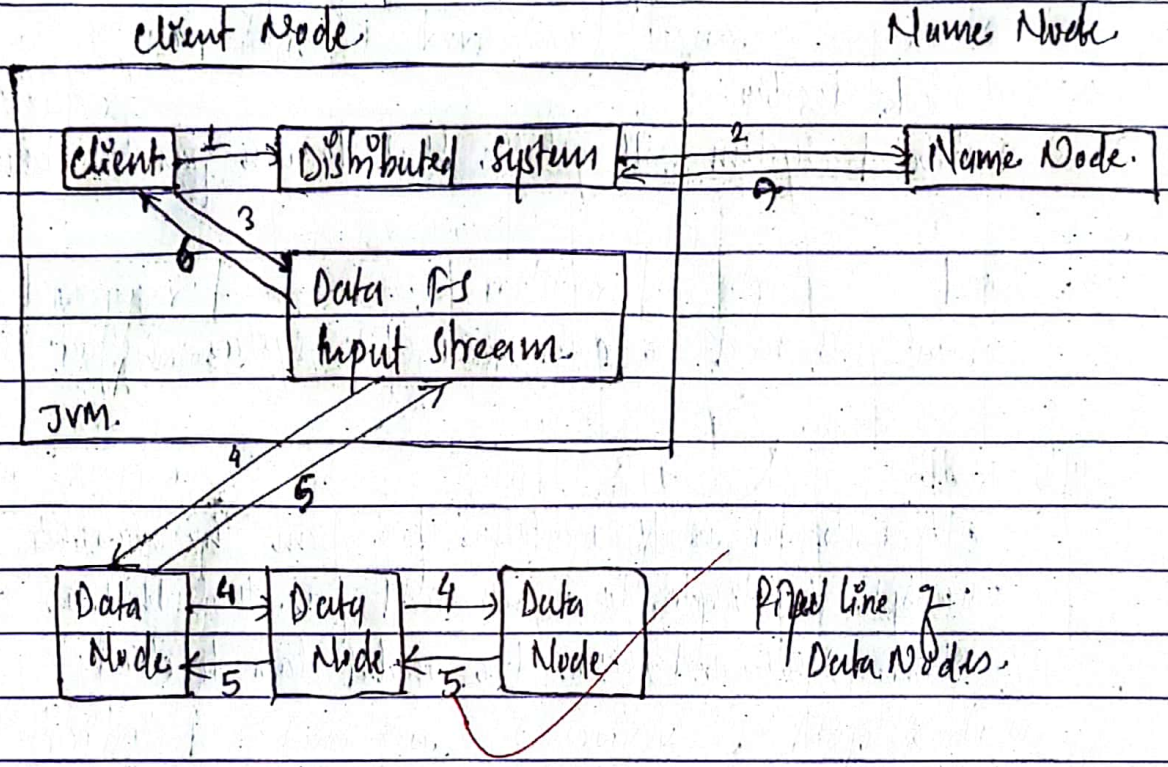
4,5 Read.
Output streams will read the file from the data Node, we have multiple data node, so it will read from that block ID obtained from Name Node. and returns the readed data to client

6. Close.
Reading completed, so closes the file.

In this way, Read Operation happen in Hadoop Distributed file system.

Similarly,

Write Operation of HDFS.



1. Open.
   client will open the file in Distributed file System to write

2. Create.
   Distributed File System create file information file name, file id in Name Node and stores its metadata.

3. Write
   client will write the data in Data Flow system Input Stream.

4. ~~Write Package~~

4. Write Packet -

the Input Stream will write packet to the desired DataNode from the pipeline of data Node. NameNode give the Information of Data Node in which to write.

5. Acknowledge Packet -

Once the data is written then datanode will acknowledge the Input Stream, that the writing has been done.

6. Close.

The file get closed since the write operation has been completed.

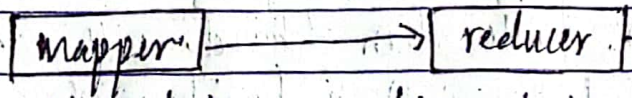
7. complete

The completed signals will be passed to the Name Node from the client Node.

In this way, the Write Operation of HDFS will be completed.

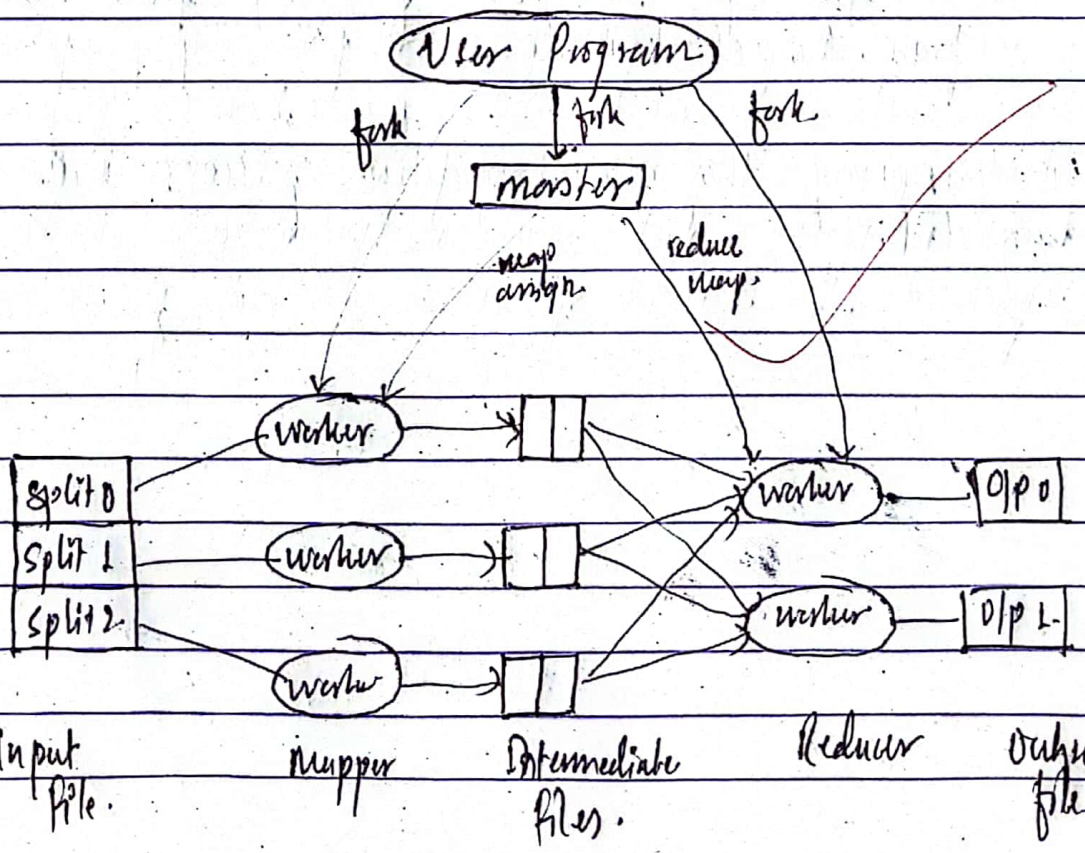
6. Map Reduce is the framework in Big Data Technology where the distributed processing of the big data is performed. In map reduce it has two main components

(i) mapper
(ii) Reducer



In mapper	In reducer
- mapping	- Summing the mapped result
- splitting.	- reduced output

Distributed Execution Overview of Map Reduce:



In the Distributed Execution of Map Reduce, the User program is the program that is written by user. It is a map reduce program which will fork the mapper worker, master and reducer worker.

Master will assign map to the master worker and also it will assign reduce task to reducer.

We have input files as split 0, split 1, split 2 which will goes to the mapper phase in which the mapping and all the transformation happens by the worker.

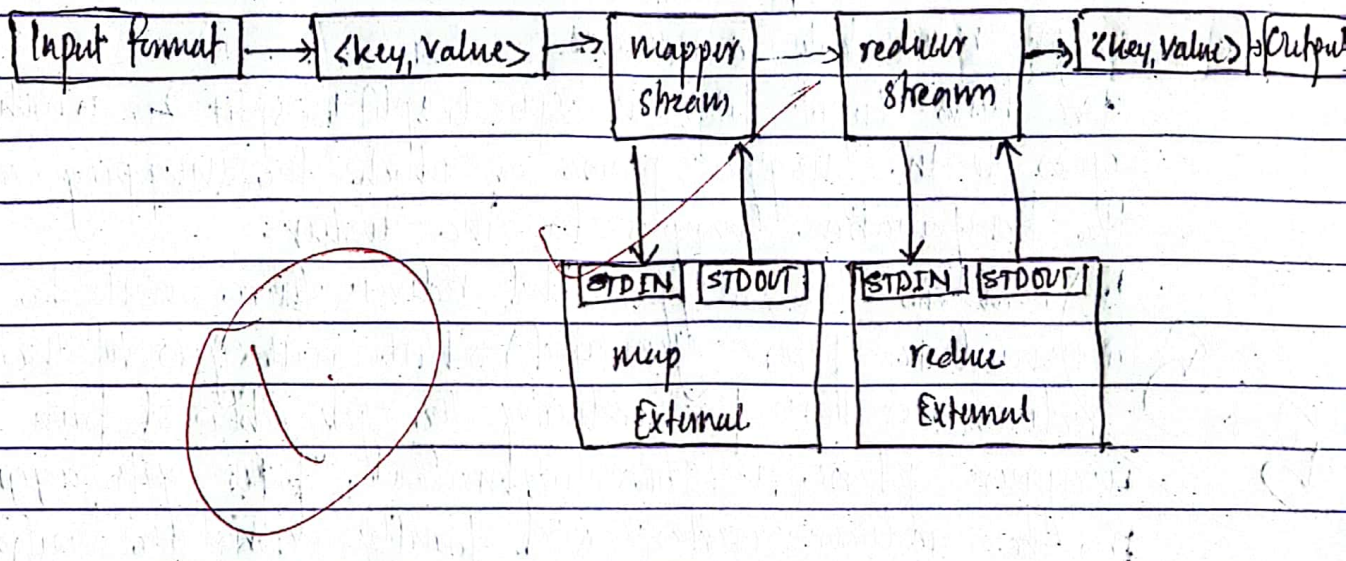
In between the mapper and reducer, there will be intermediate files generated which will goes to the reducer worker for reducing the job obtained from the mapper phase. The intermediate files gets aggregated in the reducer-worker and finally gives the output. The final output is obtained from reducer.

The output is returned to the user program by the master. And finally user gets the map reduced distributed processed result in user end.

In this way, the distributed execution of Map Reduce task happens in Big Data platform.

Q8 Continue ... (missed in prev.)

Hadoop Streaming:



In Hadoop Streaming, we can write map reduce code in any programming language by using the STDIN and STDOUT & map external and reduce External so that input and output can be transferred to any language and convert that to our map reduce task. This is the better version of traditional Map Reduce in Hadoop.

9.

db. order. find({})

(a). db. order, aggregate (

group : [{"_id" : "cust_id"}, {$sum : {"price" : 1}}])

(b). db. order. aggregate (

group : [{"_id" : "cust_id"}, {"orderdt" : $format : {"yyyy-MM-DD"}

}, {$sum : {"price" : 1}}])

(c). ~~db. order. countDocument()~~

db. order. aggregate (

group : [{$cust : "custid"}, $countDocument : {"cust_id" : 1}])

d. db.order.aggregate(

group:[{"_id":"cust_id"} orderdate:{"orderdate":
{$sum:{price:1}},
}

$match:{$sum:{$gt:{$sum:250}}}}]

),

e. db.order.aggregate(

sum:{$sum:{"price":1}},
$match:{"status":"A"}

)

f db.order.aggrate(

group[{"_id":"cust_id"},
$match:{"status":"A"},
sum:{$sum:{"price":1}}}],

$match:{sum:{gt:{sum:250}}})