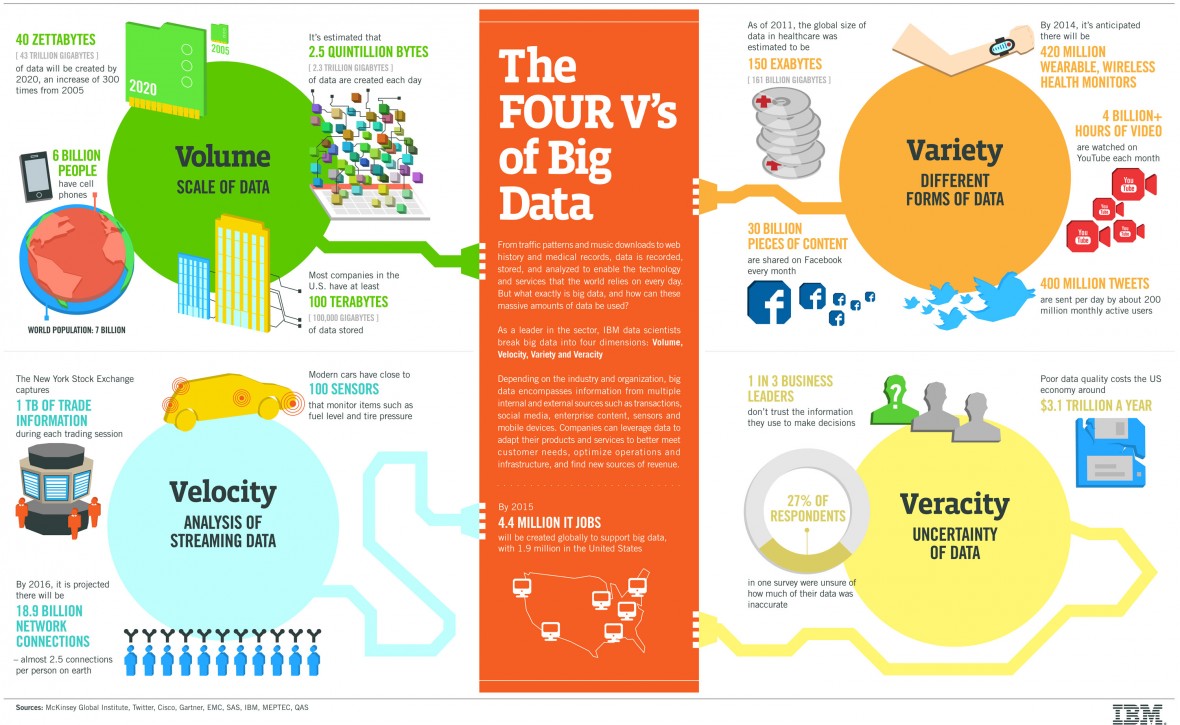
**Source of BigData [**Store the data to process]

1. Sensors, CC Cam, Social Network – Facebook, Online Shopping, Travel Industry etc.

2. NCDC - National Climate Data Center.

**4v’s of BigData**

1. **Volume** - [Giga, Tera, Peta, Exa, Zetta, Yotta… Luma (10^63)].
2. **Variety** - Structured [ RDBMS], Semi Structured [ logs ], Unstructured [ multimedia files ]
3. **Velocity** – How fast is your data growing.
4. **Veracity** – Uncertainty or quality of data.



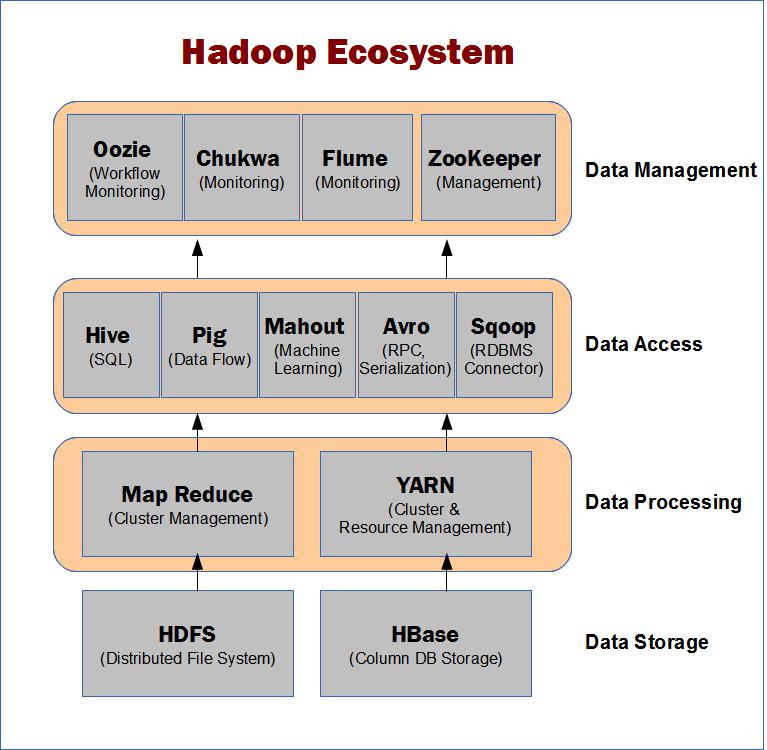
**History of Hadoop**

1. Google storage technique **GFS** and Google processing technique **MapReduce.** Google provide the whitepaper but not the implementation. Yahoo uses the GFS whitepaper and came up with **HDFS [Hadoop Distributed File System]** and **MapReduce**. **Doug Cutting** is the father of Hadoop. Elephant is the mascot, a toy played by the son of Doug.
2. **Hadoop** is an Open Source Framework which is overseen [Made it open source] by **Apache Software Foundation** for **storing** and **processing** huge amount of dataset with cluster of **Commodity Hardware.**
3. **MapReduce** is a Processing Algorithm which solves – **Parallelization**, **Distribution** & **Fault-Tolerance**.

**Hadoop Deployment Mode**

* Standalone Mode
* Pseudo Distributed Mode
* Distributed Mode

**Hadoop EcoSystem**



**Hadoop Distributed File System**

1. Is a file system for storing huge amount **dataset** using **cluster of commodity hardware** with **Streaming Access Pattern [W**rite once, **R**ead any no of time. Don’t **C**hange]**.**
2. **64 MB(HDFS) & 128 MB(HDFS2)** default block size (cluster) is used in HDFS [Not recommended for small dataset] but can be changed by Hadoop Administrator. Extra space in cluster is used for others files.
3. HDFC provides 3 replication on data nodes.

**Services of HDFS**

1. **Master Services**
2. Name Node
3. Secondary Name Node
4. Job Tracker
5. **Slave/Daemon Services**
6. Data Node
7. Task Tracker
8. Master services can communicate among each other.
9. Slave services can communicate among each other.
10. Name Node can communicate Data Node similarly Job Tracker can communicate Task Tracker and vice versa.

**Name Node [single point of failover]**

* **1 Name Node/Cluster** hence must be in high reliable hardware.
* Name Node manages in-memory metadata [**inode and image**] about data stored in Data Node.
* **Inode and image** are persisted in native file system called **Checkpoint.**
* Modification logs of checkpoint is called **Journal**
* Client talks with Name Node. Name Node returns metadata to client. More the blocks more the metadata.
* Client use the metadata to communicate with Data Node.
* It received BlockReport and updates the metadata. If metadata is lost Data Node cluster is inaccessible.
* If no HeartBeat is not received from Data Node than Name Node remove the Data Node from metadata.

**Checkpoint Node**

Special Name Node. It returns a new checkpoint to name node based on image and journal provided by Name Node.

**Backup Node**

Standby node of Named Node.

**Data Node**

* Can be in Commodity Hardware.
* Saves the client file and ack.
* Sends **Block Report** to the Name Node regarding the saved files.
* Sends **HeartBeat [data is alive]** to Name Node.

**Job Tracker [single point of failover]**

* Program on data are send to Job Tracker. It collects metadata for the program from Name Node.
* Send to nearest Task Tracker or the one which is quick using Task Tracker heartbeat.
* Job Tracker communicates with **Task Tracker** which runs the program on Data Node.
* InputFile 200 MB **->** **split file 64MB \* 4 Data Node**
* **No of split file = No of Mapper**.
* If Job Tracker fails Task Tracker cannot be reachable.

**Task Tracker**

* Actually run the program on Data Node.
* Send **HeartBeat** to Job Tracker for its alive.

**HashPartitioner**

* It’s an interface called Partitioner.
* There can be more than one Reducer. setNoOfRedTask(n).
* Its uses intermediate data to split the key value among multiple Reducer.

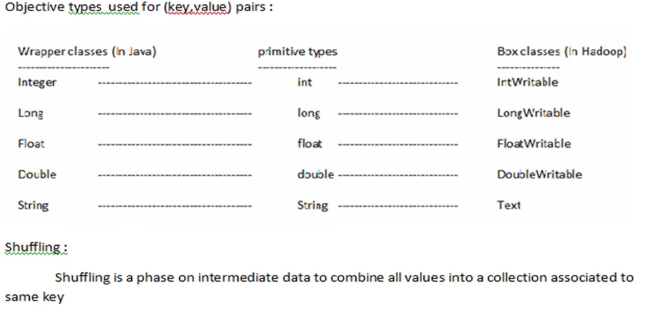
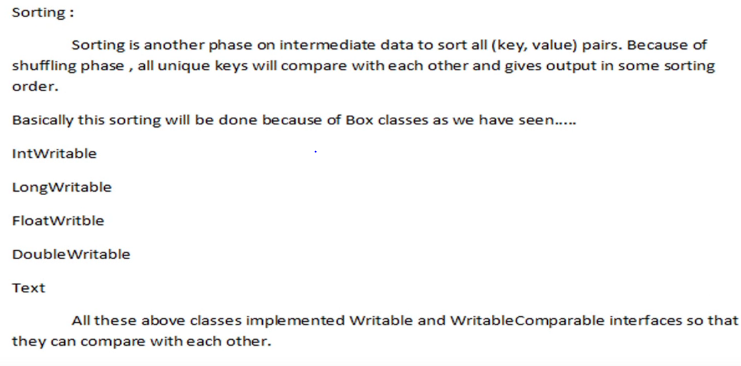
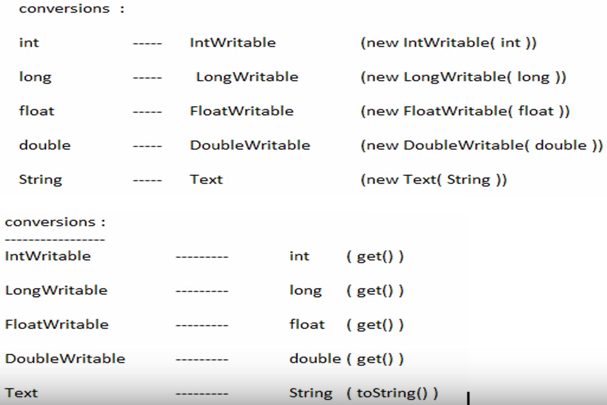
**Mapper [custom program]**

* It’s an interface called Mapper
* Default mapper is **IdentityMapper**
* No of split file = Mapper
* Works on key-value pair input
* It is the client program that is executing i.e. Mapper.java
* Produces the output in key-value pair.

**RecordReader**

* Predefined Interface between splitfile and mapper.
* Read the Record[line] from splitfile and convert to key-value pair.
* Supports predefined file format
  + - TextInputFormat[default]
    - KeyValueInputFormat
    - SequenceFileInputFormat
    - SequenceFileAsTextFormat

**DataType**



**Reducer [custom program]**

* It’s an interface called Reducer.
* Default 1 Reducer will be present.
* **Identity Reducer** is the default reducer provided by Hadoop.
* **No of Reducer = No of output file**
* User can provide there custom Reducer.
* Process **intermediate data** of the Mapper.
* Produce output file by below two phase
  + - Shuffling Phase – Remove the duplicate keys.
    - Sorting Phase – uses box Comparable and compareTo() method

**Combiner [custom program]**

* It’s a mini Reducer.
* It’s do the shuffling and sorting at mapper to reduce traffic at Reducer.

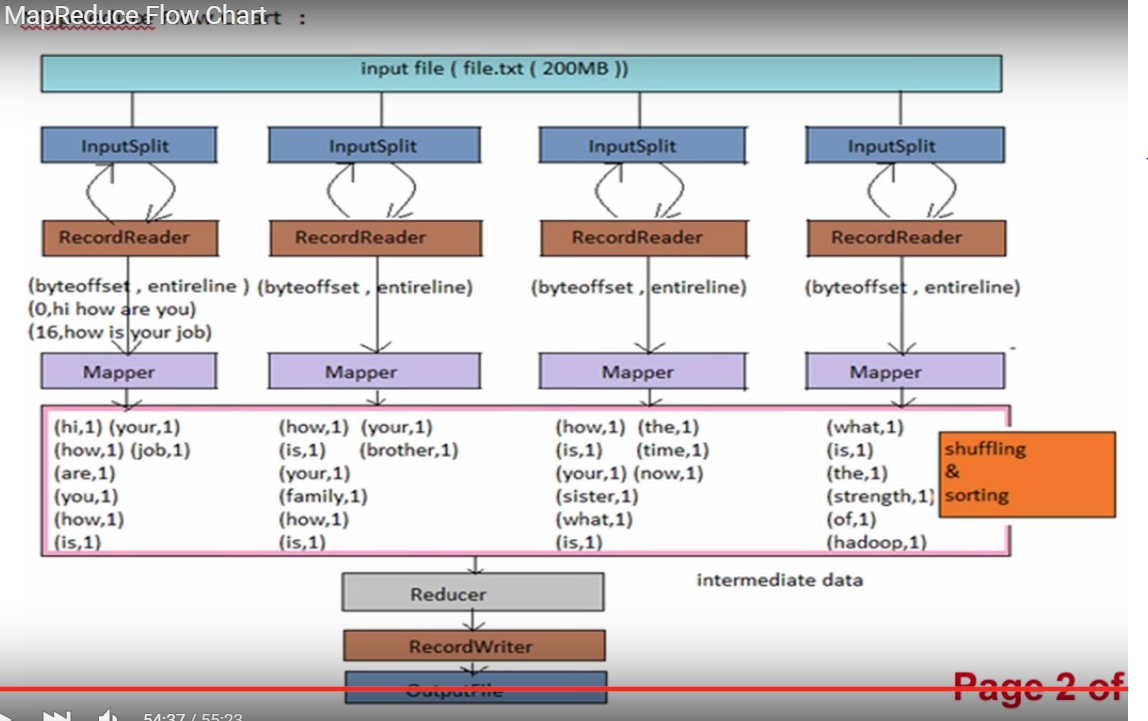
**RecordWriter**

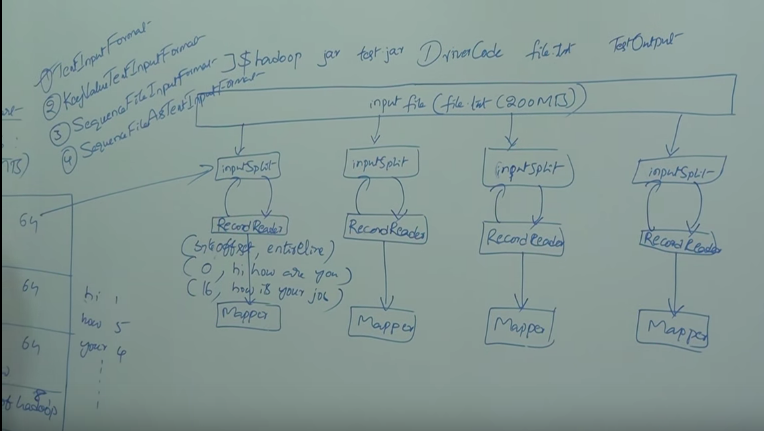
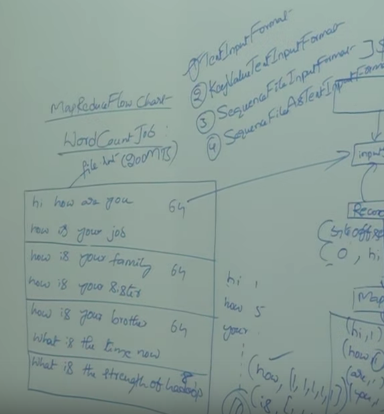
* Its write the content to output file [part-00000] in TextOutput Directory.
* **TextOutput** directory will contain below 3 files \_SUCCESS, \_Logs [directory], part-00000 [output file]

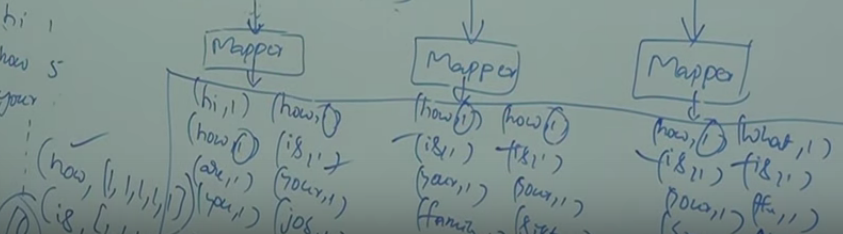
**Execute the above program:**

**$hadoop [jar] [test.jar] [DriverCode] [file.txt] [TextOutput]**

**MapReduce Flow Chart**







**HelloWorld**

* + - Create a file.
    - Load the file to HDFS – **hadoop [fs] [–put] [file.txt] [file]**
    - Create program – Driver.java, Mapper.java, Reducer.java
    - Compile – **javac –classpath hadoop-common-2.7.1.jar \*.java**
    - Run – **Create jar file.** Hadoop cannot run .class file.
      * **Jar cvf test.jar \*.class**
      * **Hadoop [jar] [test.jar] [Driver] [file] [TestOutput]directory**

**HashPartitioner [custom program]**

* Default Partitioner is **HashPartitioner**.
* Splits the key,value pairs among multipler Reducer.
* Partitioner is done based on hashcode.

**CustomPartitioner**

Public class MyPartitioner e