

## Week-3

Java Primitive	Writable Implementation
Boolean	BooleanWritable
Byte	ByteWritable
Int	IntWritable
Short	ShortWritable
Long	LongWritable
Double	DoubleWritable

Java Class	Writable
String	Text

## MapReduce

Input → Mapper() → Shuffle() & Sort() → Reducer() → Output

### # Sudoku Problem

- vi solve\_this.txt
- hdfs dfs -put /home/hadoop/solve\_this.txt /sudoku/
- /home/hadoop/hadoop-3.3.4/bin/hadoop jar /home/hadoop/hadoop-3.3.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar 'sudoku/solve\_this.txt' /sudoku/solve\_this.txt
- /home/hadoop/hadoop-3.3.4/bin/hadoop jar /home/hadoop/hadoop-3.3.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar 'sudoku/solve\_this.txt' /sudoku/wcount\_result1
- hdfs dfs -cat /sudoku/wcount\_result1/\*

Step 1: Create an input file on the local system.  
(Sudoku & word count)

Step 2: Copy the file from local system to hadoop cluster

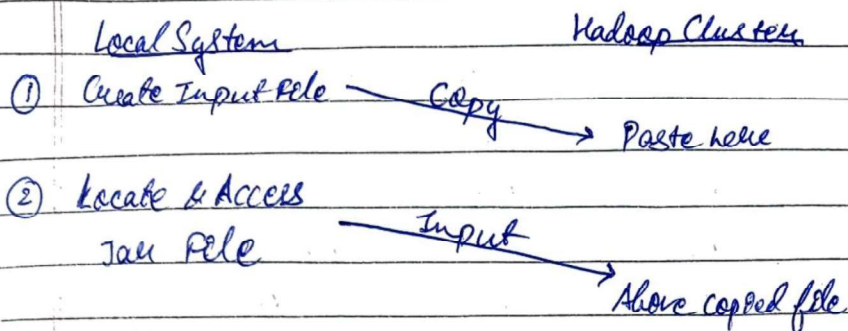
Step 3: Access the jar file present on the local system

Step 4: Give the file on hadoop cluster as the input file

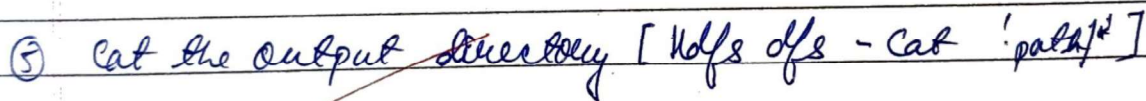
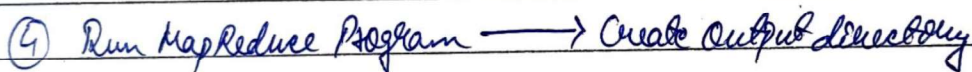
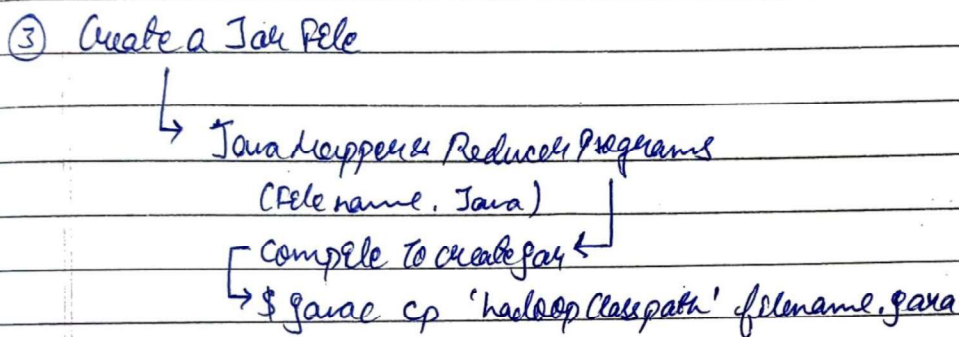
Step 5: Create an output directory on hadoop cluster.

Step 6: Cat the output directory to see the output.

## MapReduce Architecture & Process



or



### # Word Count Problem (Using own jar file)

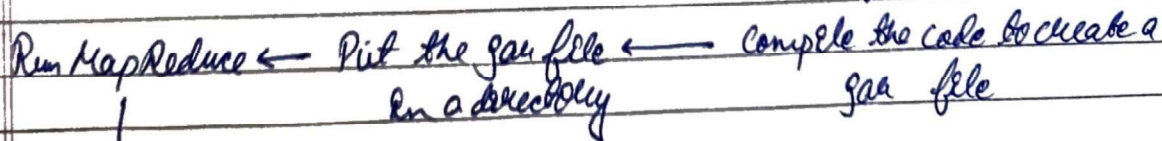
Create Input File

↓

Move File to Hadoop

↓

Write Java Mapper & Reducer Program



↓

Cat Output Files.



- vp wc.txt

Hello world

Hello class

- hdfs dfs -put /home/hadoop/wc.txt /hatch-3/200808042

- wordcount.classes - wordcount.java

- javac cp 'hadoop classpath' wordcount.classes WordCount.java

- jar -crf wordcount.jar -c wordcount.class /\*

- ~~mkdir~~ hadoop jar wordcount.jar /wc.txt wc\_output

- hdfs dfs -cat wc\_output /\*

Hello 2

Class 1

World 1