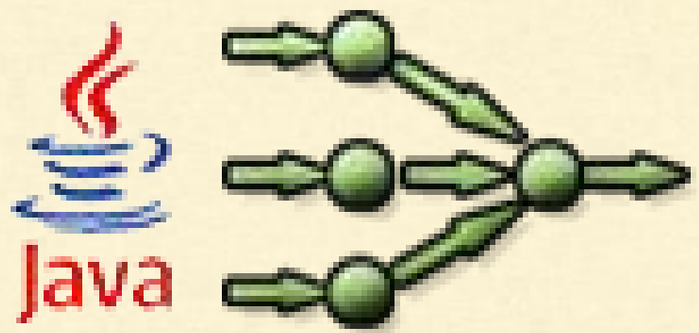
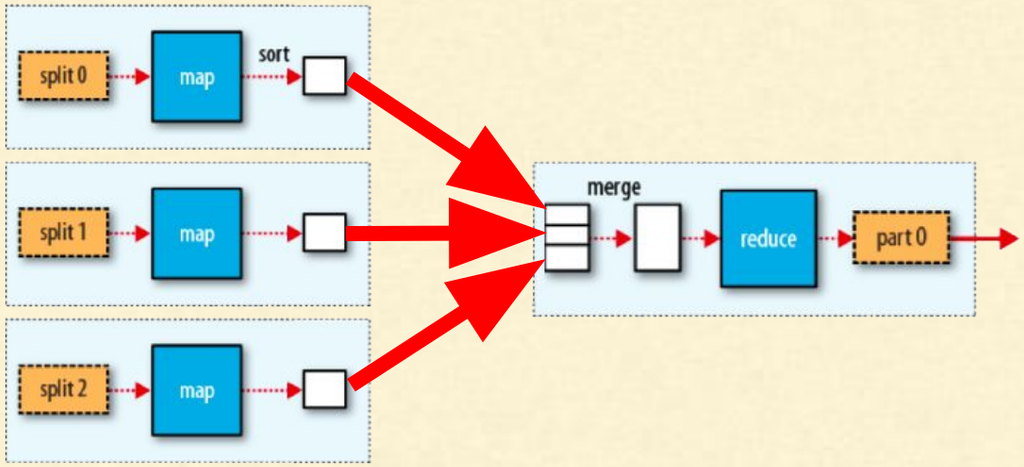


Writing MapReduce with Java

Recap - Why MapReduce?

- Instead of processing Big Data directly
- We breakdown the logic into
  - map()
    - Executed on machines with data
    - Gives out key-value pairs
  - Reduce()
    - Gets output of maps grouped by key
    - Grouping is done by MapReduce Framework
    - Can aggregate data



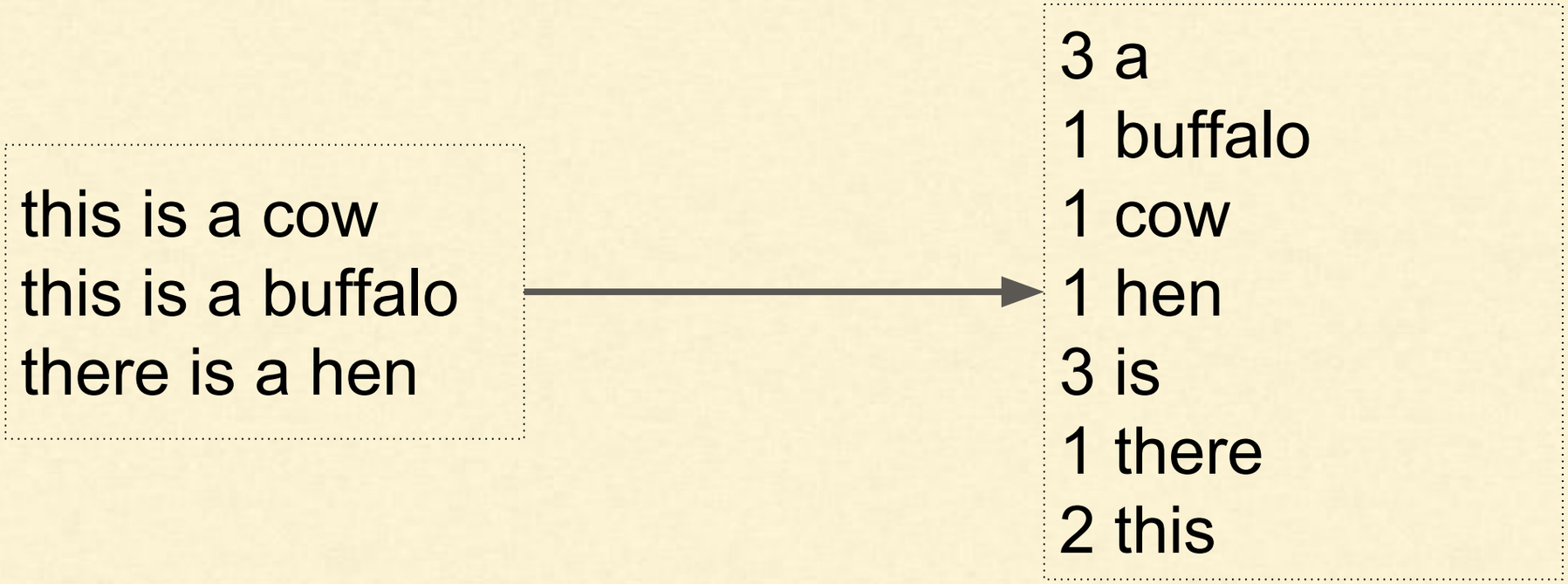
MAP / REDUCE - Why JAVA

Why in Java?

- Primary Support
- Can modify behaviour to a very large extent

MAP / REDUCE - JAVA - Objective

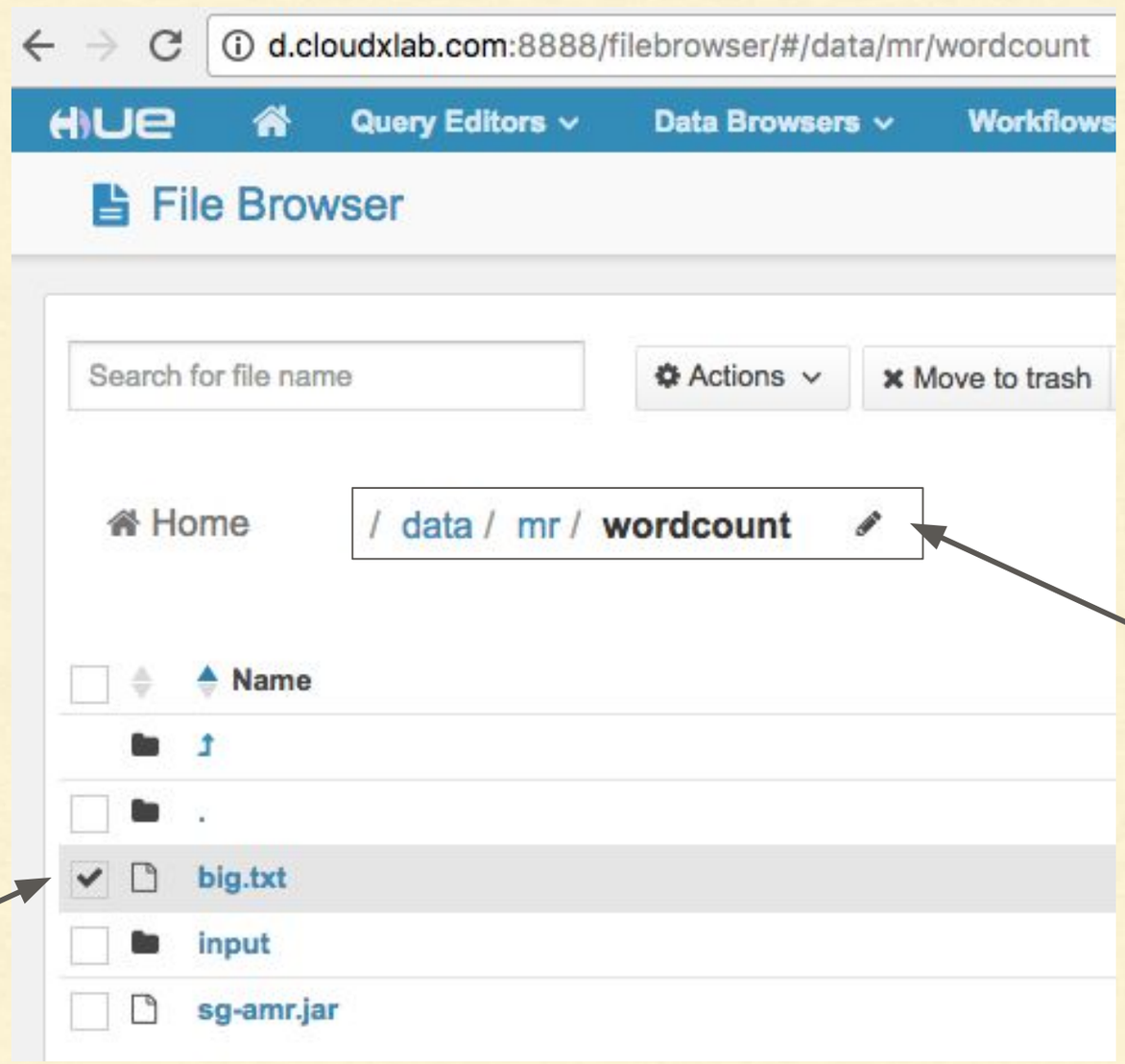
Write a map-reduce job to count unique words



MAP / REDUCE - JAVA - Objective

Write a map-reduce job to count unique words in text file

/data/mr/wordcount/input/big.txt

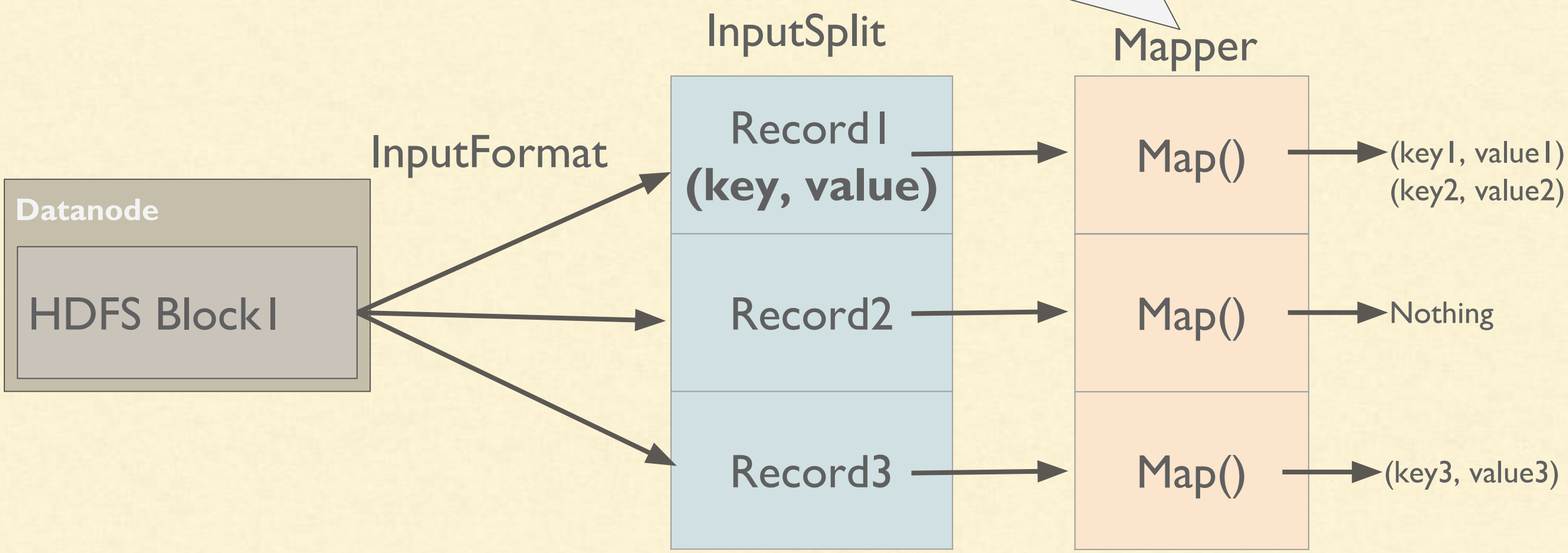


Location in HDFS
In CloudxLab

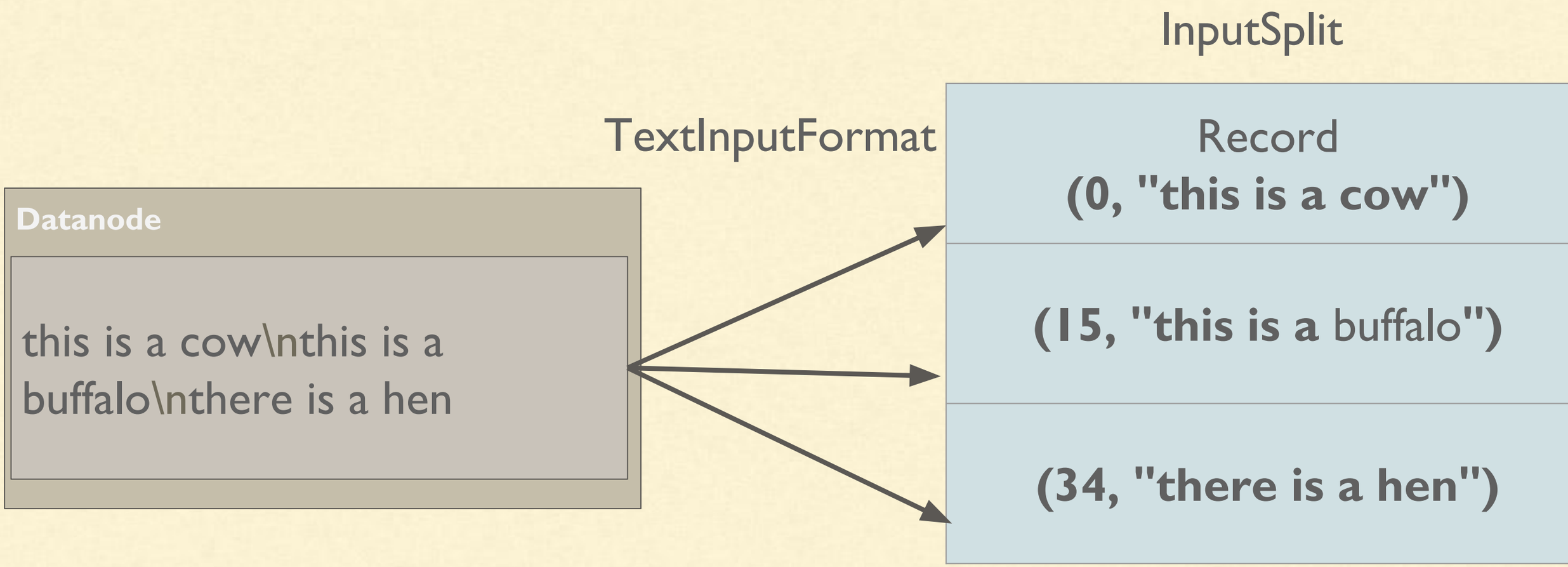
Input File

MAP / REDUCE - JAVA - Mapper

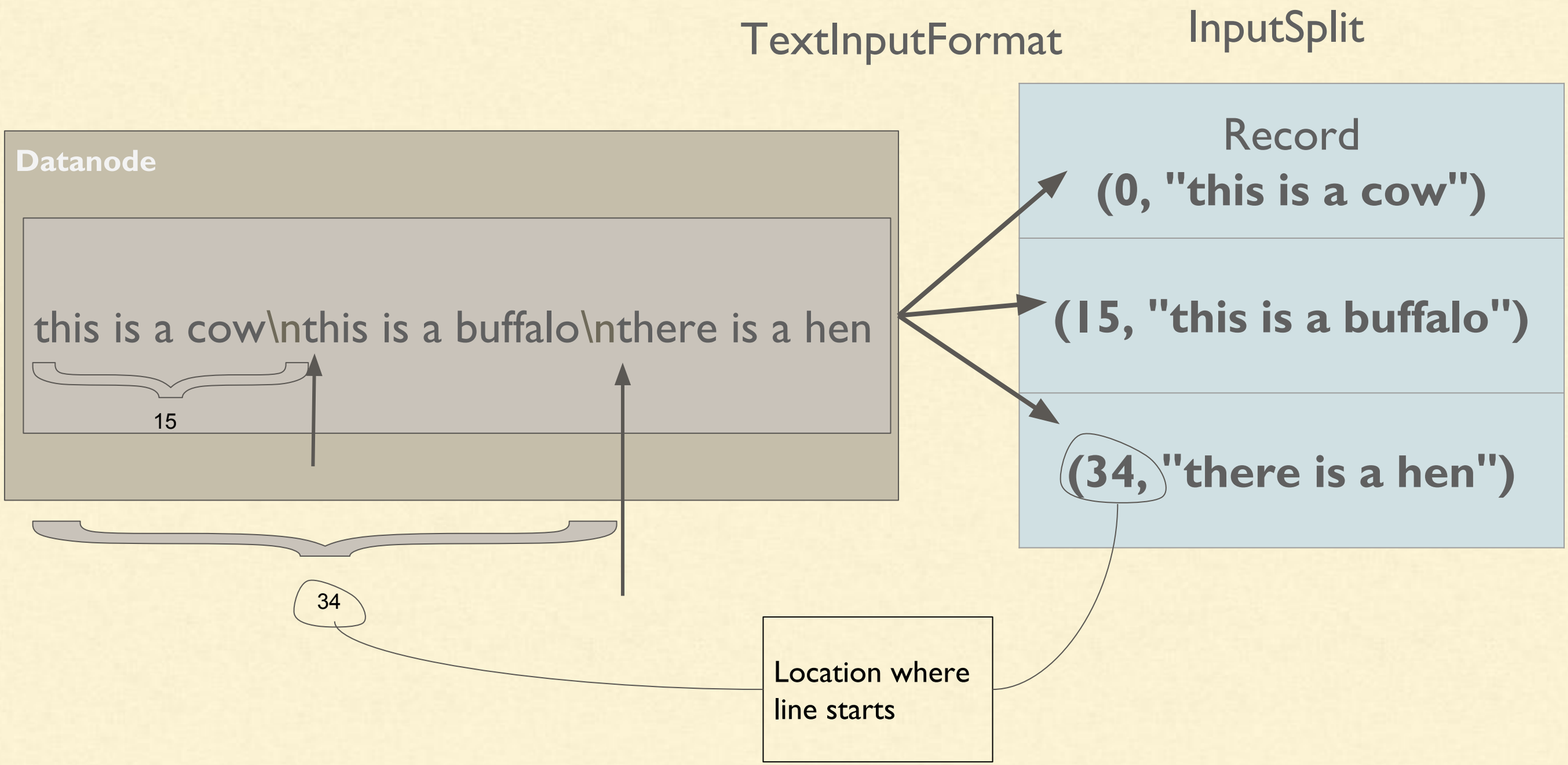
We need to write the code which would break down the input record into key-value.



MAP / REDUCE - JAVA - Mapper



MAP / REDUCE - JAVA - Mapper



MAP / REDUCE - JAVA - Mapper - Class

```
public class StubMapper                                    {

    // A class in java is a complex data type that can have methods in it too
    // Or a class a blue print.
    // Person is a class and sandeep is an object.

}
```

MAP / REDUCE - JAVA - Mapper - Extends

```
public class StubMapper extends Mapper                                    {

    // Out class stubmapper is inheriting the parent class Mapper
    // Which is provided by framework
    // StubMapper is initialized for each input split



}
```

MAP / REDUCE - JAVA - Mapper - Datatypes

Data types of input, ouput key and value

```
public class StubMapper extends Mapper<Object, Text, Text, LongWritable> {

}
```

Data type of Input Key.
In our example, it is number of bytes at which the value is starting

The Data type of input value.
In our case, input value is each line, i.e. Text

The Data type of output key,
We are going to give key as word, therefore it is Text

The Data type of output value,
We are going to give value as 1 therefore it is Long

MAP / REDUCE - JAVA - Mapper - method

```
public class StubMapper extends Mapper<Object, Text, Text, LongWritable> {

    @Override
    public void map(Object key, Text value, Context context)
                                                        {



    }
}
```

MAP / REDUCE - JAVA - Mapper - method

```
public class StubMapper extends Mapper<Object, Text, Text, LongWritable> {

    @Override
    public void map(Object key, Text value, Context context)
                                                        {

        String[] words = value.toString().split("[ \t]+");
        for(String word:words)
        {
            context.write(new Text(word), new LongWritable(1));
        }
    }
}
```

The input line is split by space or tabs into array of strings

MAP / REDUCE - JAVA - Mapper - method

```
public class StubMapper extends Mapper<Object, Text, Text, LongWritable> {

    @Override
    public void map(Object key, Text value, Context context)
                                                        {

        String[] words = value.toString().split("[ \t]+");
        for(String word:words)
        {
            context.write(new Text(word), new LongWritable(1));
        }
    }
}
```

For Each of the words ...

MAP / REDUCE - JAVA - Mapper - method

```
public class StubMapper extends Mapper<Object, Text, Text, LongWritable> {

    @Override
    public void map(Object key, Text value, Context context)
                                                        {

        String[] words = value.toString().split("[ \t]+");
        for(String word:words)
        {
            context.write(new Text(word), new LongWritable(1));
        }
    }
}
```

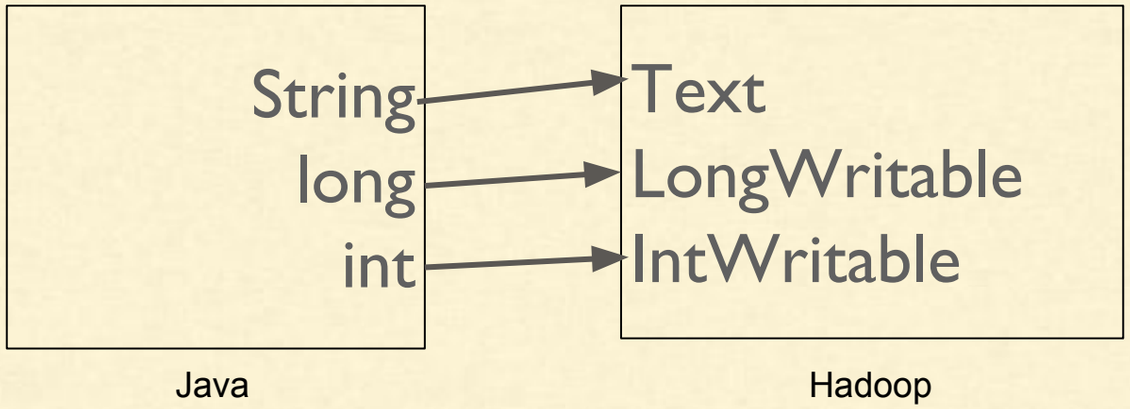
… we give out the word as key ...

… and numeric 1 as the value.

MAP / REDUCE - JAVA - Writable

What is "***new Text(word)*** "?

Usual types of Java to represent numbers and text were not efficient. So, mapreduce team designed their own classes called writables



MAP / REDUCE - JAVA - Writable

What is "***new Text(word)*** "?

Before handling over anything to MapReduce, you need to wrap it into corresponding writable class or create a new one.

Wrapping

```
new Text(word)

new LongWritable(word)
```

Unwrapping

```
value.toString()
```

MAP / REDUCE - Java - Full Code

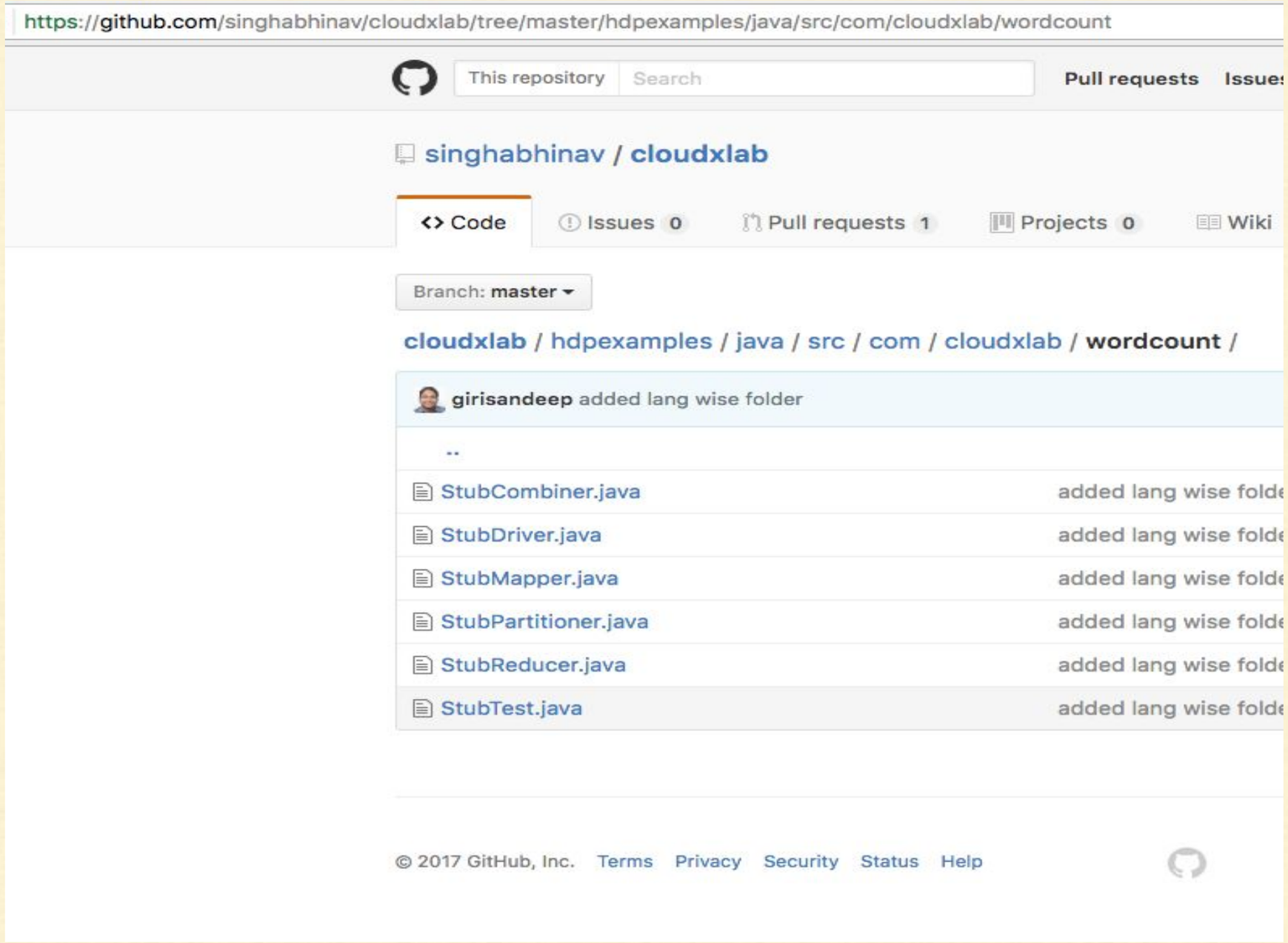
Create a Mapper

```
public class StubMapper extends Mapper<Object, Text, Text,
LongWritable> {

  @Override
  public void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {

     String[] words = value.toString().split("[ \t]+");
     for(String word:words)
     {
        context.write(new Text(word), new LongWritable(1));
     }
  }
}
```

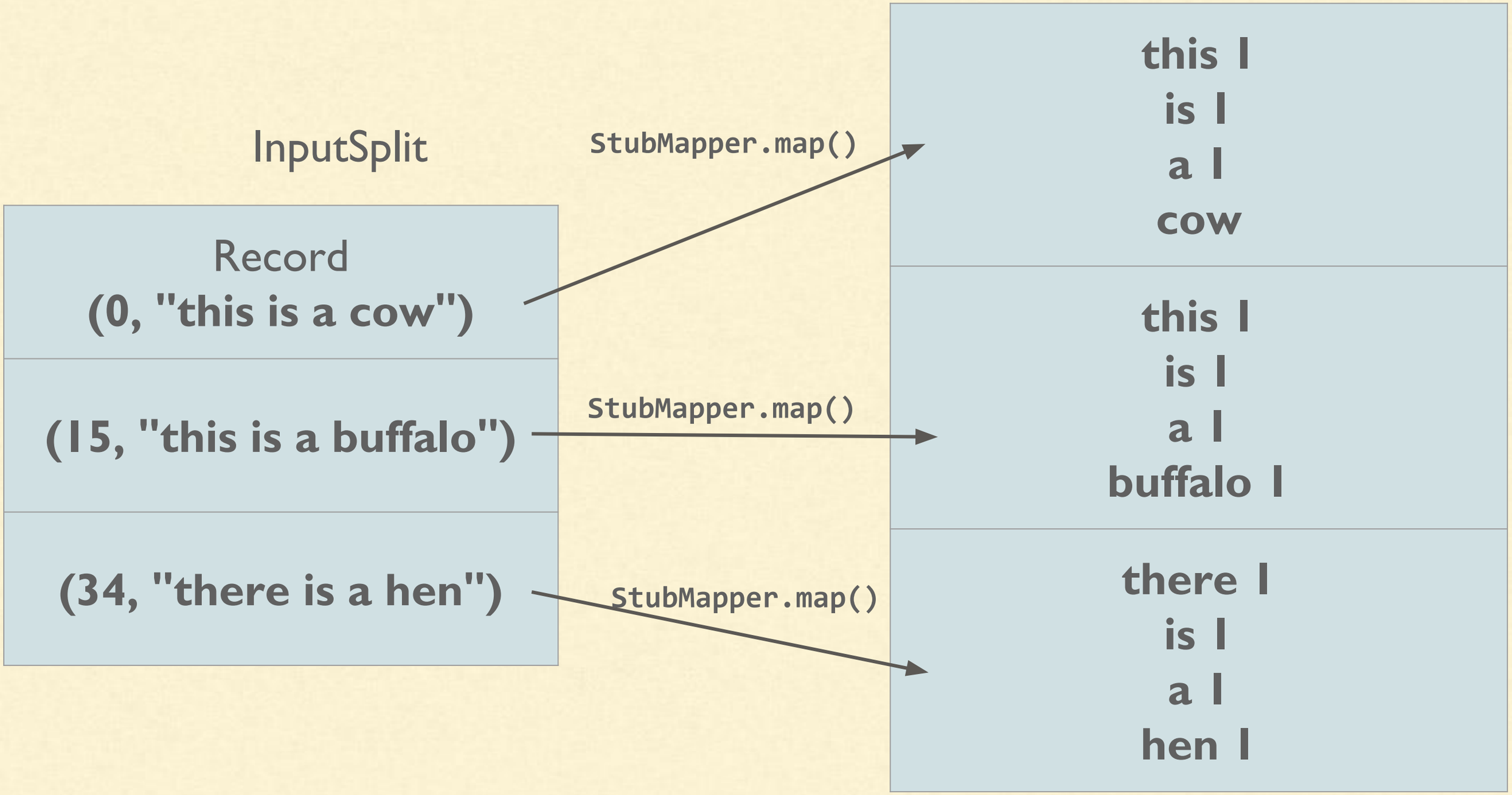
MAP / REDUCE - Java - Full Code



Take a look at complete code at gihub folder:
https://github.com/singhabhinav/cloudxlab/tree/master/hdpexamples/java/src/com/cloudxlab/wordcount

MAP / REDUCE - Java - Complete Code

The output of Mapper



MAP / REDUCE - JAVA - Reducer

Create a Reducer

```
public class StubReducer extends Reducer<Text, LongWritable, Text,
LongWritable> {

  @Override
  public void reduce(Text key, Iterable<LongWritable> values, Context context)
      throws IOException, InterruptedException {

      long sum = 0;
      for(LongWritable iw:values)
      {
          sum += iw.get();
      }
      context.write(key, new LongWritable(sum));
  }
}
```

MAP / REDUCE - JAVA

Create a Driver

```
public class StubDriver {
    public static void main(String[] args) throws Exception {
        Job job = Job.getInstance();
        job.setJarByClass(StubDriver.class);
        job.setMapperClass(StubMapper.class);
        job.setReducerClass(StubReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

        FileInputFormat.addInputPath(job, new Path("/data/mr/wordcount/input/big.txt");
        FileOutputFormat.setOutputPath(job, new Path("javamrout"));
        boolean result = job.waitForCompletion(true);
        System.exit(result ? 0 : 1);
    }
}
```

MAP / MAP / REDUCE - JAVA -

Writing Map-Reduce in Java (Continued)

9. Export jar
10. scp jar to the hadoop server
11. Run it using the following command:
    hadoop jar sandeep/training2.jar StubDriver <arguments>
    e.g: hadoop jar sandeep/training2.jar StubDriver
    /users/root/wordcount/input
    /users/root/wordcount/output16/
12. In case there is a need use -use-lib
13. Testing: Add all the jars provided

Using external Jars:

```
$ export LIBJARS=/path/jar1,/path/jar2
$ hadoop jar my-example.jar com.example.MyTool -libjars ${LIBJARS}
```

MAP / MAP / REDUCE - JAVA - Hands-ON

```
## These are the examples of Map-Reduce
git clone https://github.com/singhabhinav/cloudxlab.git
cd cloudxlab/hdpexamples/java
ant jar

To Run wordcount MapReduce, use:
hadoop jar build/jar/hdpexamples.jar
com.cloudxlab.wordcount.StubDriver
```

MAP / REDUCE - INPUT SPLITS (CONT.)

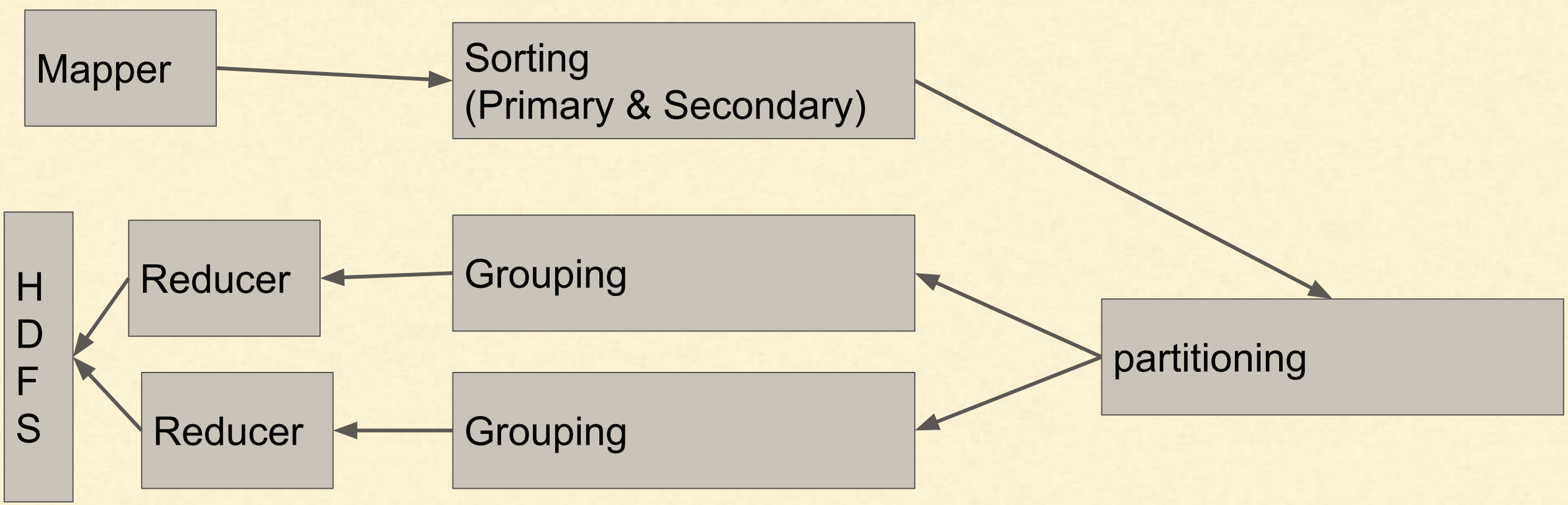
- Has length and locations
- Largest gets processed first
- InputFormat creates splits
  - Default one is TextInputFormat
  - Extend it to custom splits/records

```
public abstract class InputSplit {
    public abstract long getLength()
    public abstract String[] getLocations()
}
```

```
public abstract class InputFormat {
  List getSplits (JobContext);
  RecordReader createRecordReader
        (InputSplit, TaskAttemptContext);
}
```

MAP / REDUCE - Secondary Sorting

- The key-value pairs generated by Mapper are sorted by key
- Reducer recieves the values for each key.
- These values are not sorted.
- To have these sorted, you need to use Secondary Sorting.



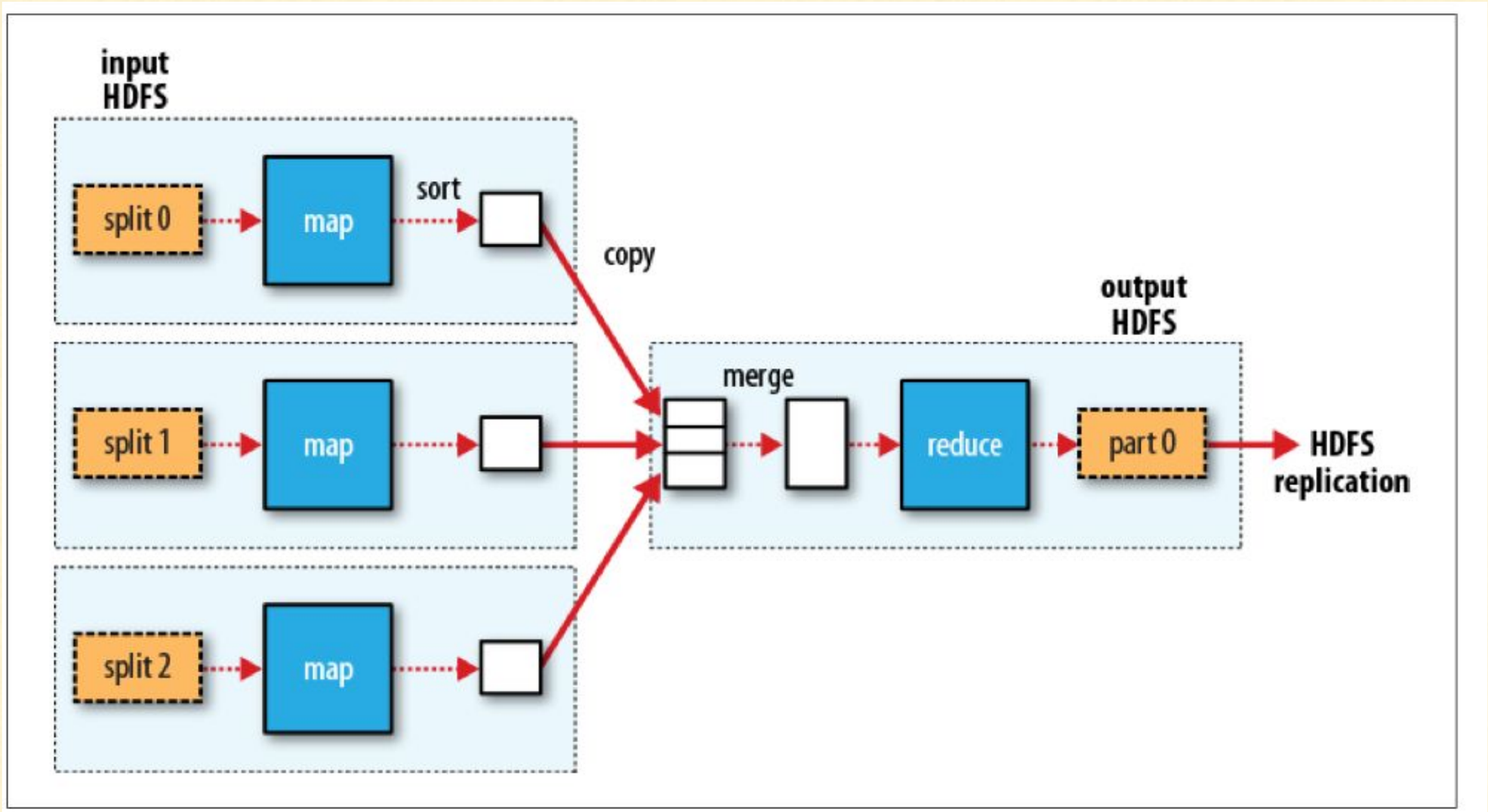
MAP / REDUCE - Secondary Sorting

1. Define Sorting:
   a. Create a *WritableComparable* class instead of “key”
   b. In this class, return Primary and Secondary Key.
2. Define Grouping
   a. Create Grouping class by extending *WritableComparator*
3. Define Paritioning
   a. Extend Partitioner and implement how to parition on PK

See the folder “nextword” from “Session 5” project

More: here and here and here and in “The Definitive Guide of Hadoop”.

MAP / REDUCE - DATA FLOW WITH SINGLE REDUCER

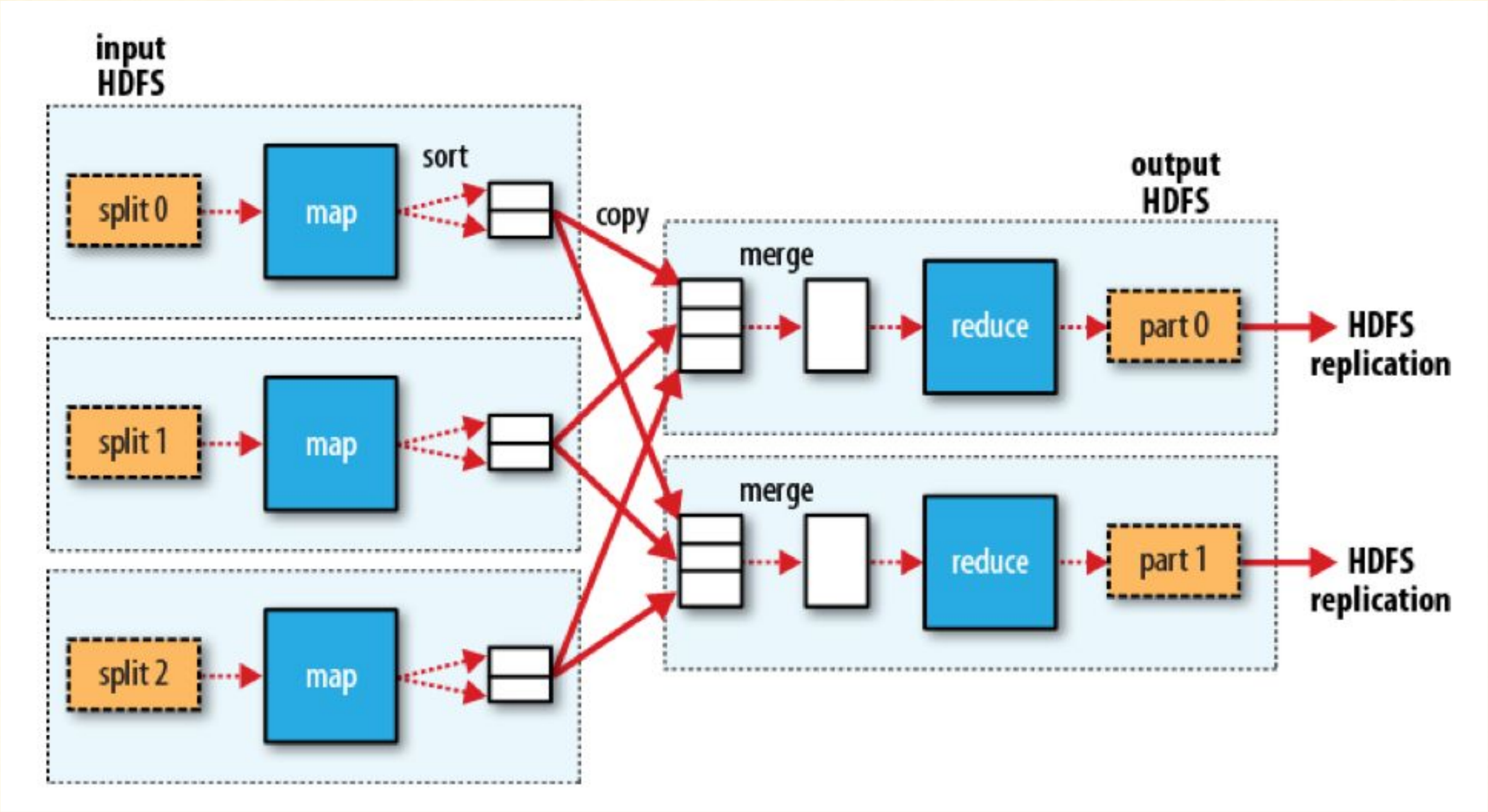


Network Transfer

Local Transfer

Node

MAP / REDUCE - MULTIPLE REDUCERS



MAP / REDUCE - PARTITIONER

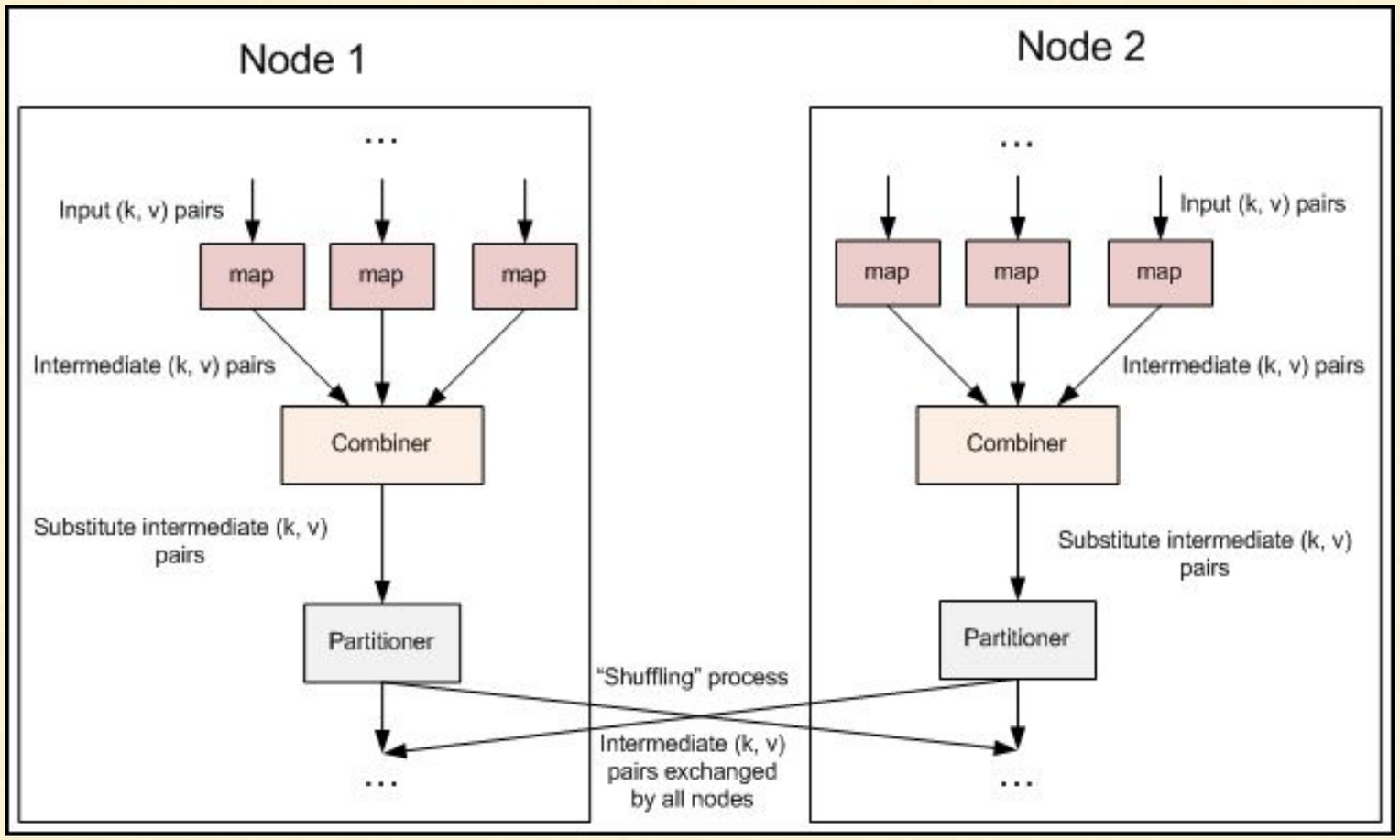
- Defines the key for partitioning
- Decides which key goes to which reducers

```
public static class AgePartitioner extends Partitioner<Text, Text> {
    public int getPartition(Text gender, Text value, int numReduceTasks) {
        if(gender.getString().equals("M"))
        return 0;
       else
        return 1;
    }
}
```

MAP / REDUCE - HOW MANY REDUCERS?

- By Default One
- Too many reducers effort of shuffling is high
- Too few reducers, computation takes time
- Tune it to the total number of slots

MAP / REDUCE - COMBINER FUNCTIONS



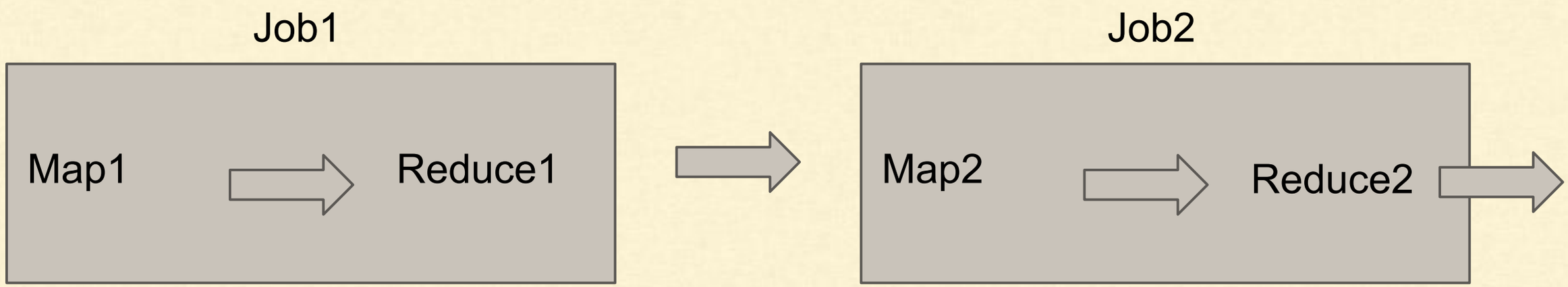
- Runs on the same node after Map has finished
- Processes the output of Map
- Helps in minimise the data transfer
- Does not replace reducer
- Should be commutative and associative

MAP / REDUCE - COMBINER FUNCTIONS

- Defined Using Reducer Class - Same signature as reducer
- No matter in what way it is applied, output should be same
- Examples: Sum, Min, Max
- max(0, 20, 10, 25) = max(max(0, 20), max(10,25)) = max(20, 25) = 25
  - = max(max(0, 10), max(20,25)) = max(10, 25) = 25
- Not: average or mean
  - avg(0, 20, 10, 25) => 11.25
  - = avg(avg(0, 20), avg(10,25)) = avg(10, 17.5) = 13.75
  - = avg(avg(0, 10, 20), avg(25)) = avg(10, 25) = 17.5
- is function f(a, b,c…) = {return sqrt(a*a+b*b+c*c…);}

job.setCombinerClass(MaxTemperatureReducer.class);

MAP / REDUCE - Job Chaining



Method1:

Using our Java Code:

```
if(job1.waitForCompletion(true))
{
    job2.waitForCompletion(true);
}
```

MAP / REDUCE - Job Chaining

Method2: Using Unix

hadoop jar x.jar Driver1 inputdir outputdir1 **&&** hadoop jar x.jar Driver2 outputdir1 outdir2

Method3: Using Oozie

We will discuss it later.

Method4: Using dependencies

//job2 can’t start until job1 completes

job2.addDependingJob(job1);

See [this project.]

In this project we chain our previously done wordcount with new job to order the words in descending order of counts

MAP / REDUCE - Pipes

1. For running C/C++ code
2. Better than streaming
3. You can run as following:

$ bin/hadoop pipes -input inputPath -output outputPath -program path/to/executable

Hadoop & Spark

Thank you.

+1 419 665 3276 (US)
+91 803 959 1464 (IN)

support@knowbigdata.com

Subscribe to our Youtube channel for latest videos -
https://www.youtube.com/channel/UCxugRFe5wETYA7nMH6VGyEA

MAP / REDUCE - JAVA

Writing Map-Reduce in Java

1. Install Eclipse
2. Create a Java project
3. Add Libs: hadoop-mapreduce-client-core.jar, hadoop-common.jar
4. Change JDK to 7.0
5. Change the Java Compiler settings to 1.6

MAP / REDUCE - JAVA

Checkout & Follow instructions at

https://github.com/girisandeep/mrexamples

MAP / REDUCE - JAVA

14. Create Test Case

```
public class StubTest {

  @Before
  public void setUp() {
    mapDriver = new MapDriver<Object, Text, Text, LongWritable>();
    mapDriver.setMapper(new StubMapper(););
    reduceDriver = new ReduceDriver<Text, LongWritable, Text, LongWritable>();
    reduceDriver.setReducer(new StubReducer(););
   mapReduceDriver = new MapReduceDriver<Object, Text, Text, LongWritable, Text,
LongWritable>();
    mapReduceDriver.setMapper(mapper);
    mapReduceDriver.setReducer(reducer);
  }

  @Test
  public void testXYZ() {
     ....
  }
}
```

MAP / REDUCE - JAVA

15. Create a test case

```
@Test
    public void testMapReduce() throws IOException {

        mapReduceDriver.addInput(new Pair<Object, Text>
            ("1", new Text("sandeep giri is here")));
        mapReduceDriver.addInput(new Pair<Object, Text>
            ("2", new Text("teach the map and reduce class is fun.")));
        List<Pair<Text, LongWritable>> output = mapReduceDriver.run();
        for (Pair<Text, LongWritable> p : output) {
            System.out.print(p.getFirst() + "-" + p.getSecond());
            //assert here
            ....
        }
    }
```

Custom Writable

- Objects that are serialized need to extend Writable
- Examples: Text, IntWritable, LongWritable, FloatWritable, BooleanWritable etc. (See)
- You can define you own

```
public interface Writable {
  void readFields(DataInput in);
  void write(DataOutput out);
}
```

AVAILABLE INPUT SPLITS

Notes

- You can directly read files inside your mapper:
  - *FileSystem fs = FileSystem.get(URI.create(uri), conf);*
- Third Party - GZIp Splittable: http://niels.basjes.nl/splittable-gzip
- https://github.com/twitter/hadoop-lzo