Software Systems and Distributed Systems SS 2017

Shen Gao
Bibek Paudel
May 3, 2017

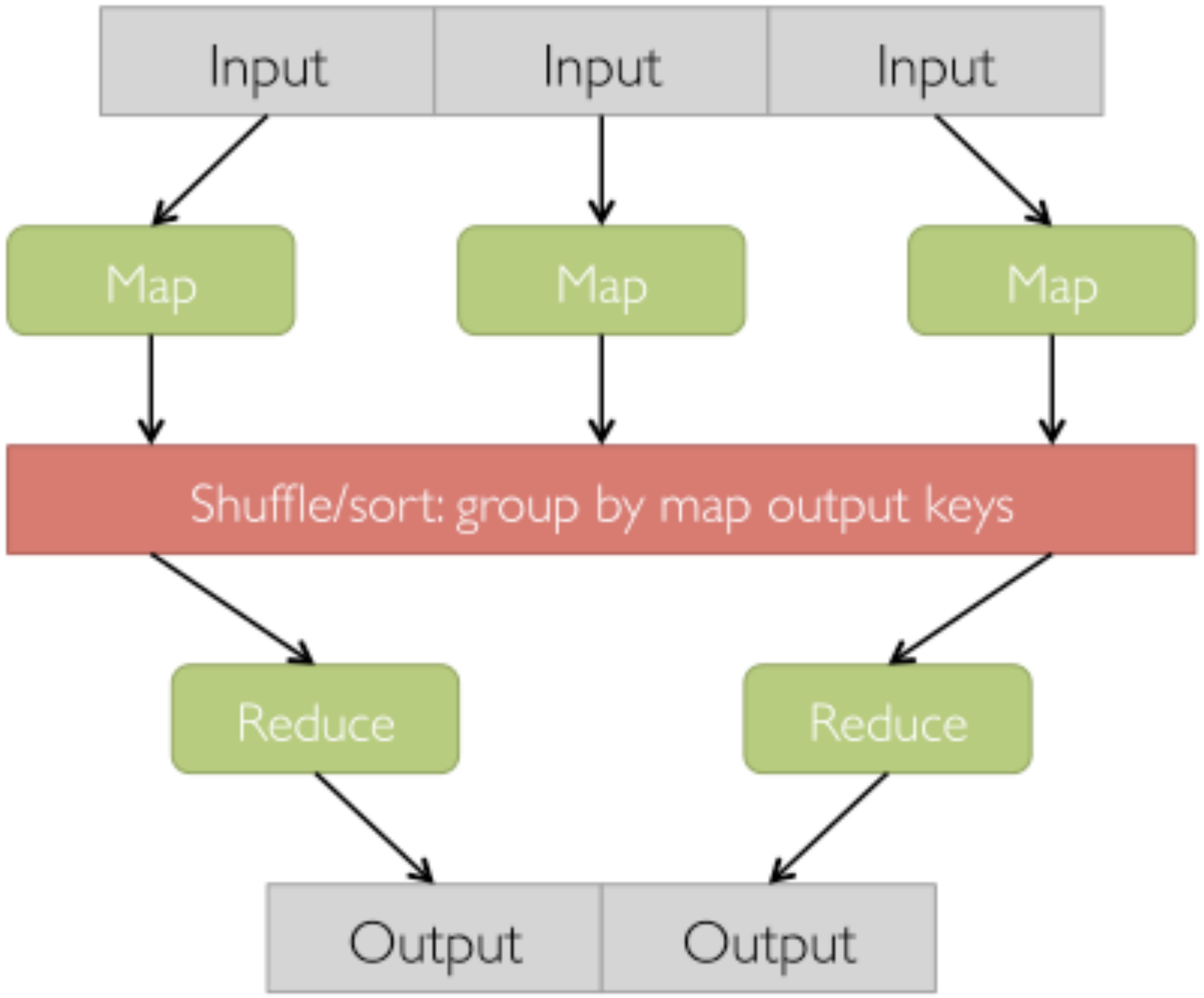
MapReduce Programs

Mapper<Input Key, Input Value, Output Key, Output Value>

Reducer<Input Key, Input Value, Output Key, Output Value>

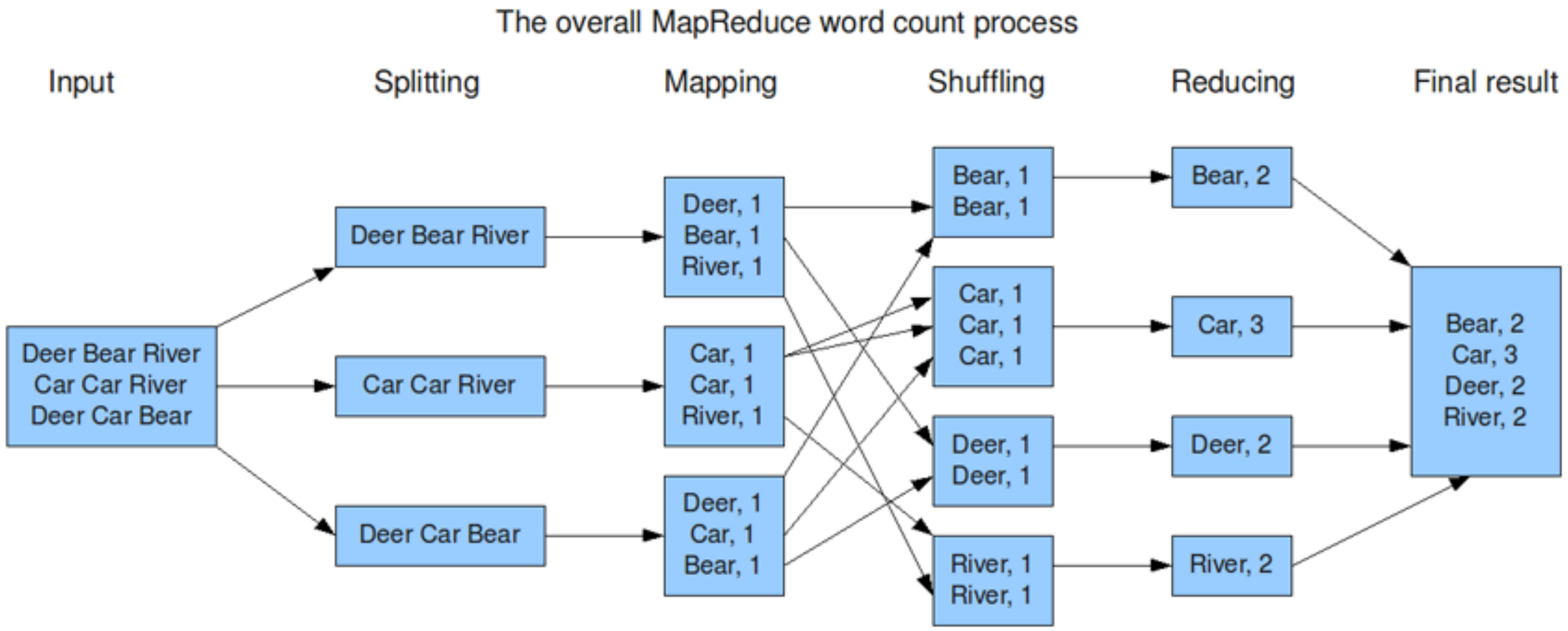
```
main () {
  set mapper and reducer
  set input/output types
  set input/output paths
  set other parameters
  run job
  }
```

MapReduce Programs



Picture: tessera.io

WordCount



Picture: bigdataprojects.org

WordCount

```
public class WordCount{

  public static void main(String[] args)throws Exception{
  Configuration conf = new Configuration();
    Job job = Job.getInstance(conf);

    job.setMapperClass(CountMapper.class);
    job.setReducerClass(CountReducer.class);
    job.setJarByClass(WordCount.class);
    job.setNumReduceTasks(1);
```

```
        job.setJarByClass(WordCount.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(NullWritable.class);

        FileInputFormat.addInputPath(job, new Path("/user/hue/
data/plot_summaries.txt"));
        FileSystem fs = FileSystem.get(conf);

        Path outputDestination = new Path(args[0]);
        if (fs.exists(outputDestination)) {
          fs.delete(outputDestination, true);
        }

        FileOutputFormat.setOutputPath(job, outputDestination);
    job.waitForCompletion(true) ? 0 : 1;
}
```

Mapper

```
public static class CountMapper
   extends Mapper<Object, Text, Text, IntWritable>{

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

     String line = value.toString();
     //data-processing (removal of comma, etc.) here

     StringTokenizer tokenizer = new StringTokenizer(line);
     tokenizer.nextToken();

     while (tokenizer.hasMoreTokens()) {
      context.write(new Text("1"), new IntWritable(1));
      }
       }
   }
```

Mapper

```
public static class CountMapper
   extends Mapper<Object, Text, Text, IntWritable>{

      Set<String> stopWords;
      @Override
      public void setup(Context context) throws IOException {
      }

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

Reducer

```
public static class CountReducer
extends Reducer<Text, IntWritable, IntWritable, NullWritable> {

   public void reduce(Text key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException{
        int sum = 0;

    for (IntWritable val : values) {
          sum++;
        }
        context.write(new IntWritable(sum), NullWritable.get());
      }
}
```

Word Frequency

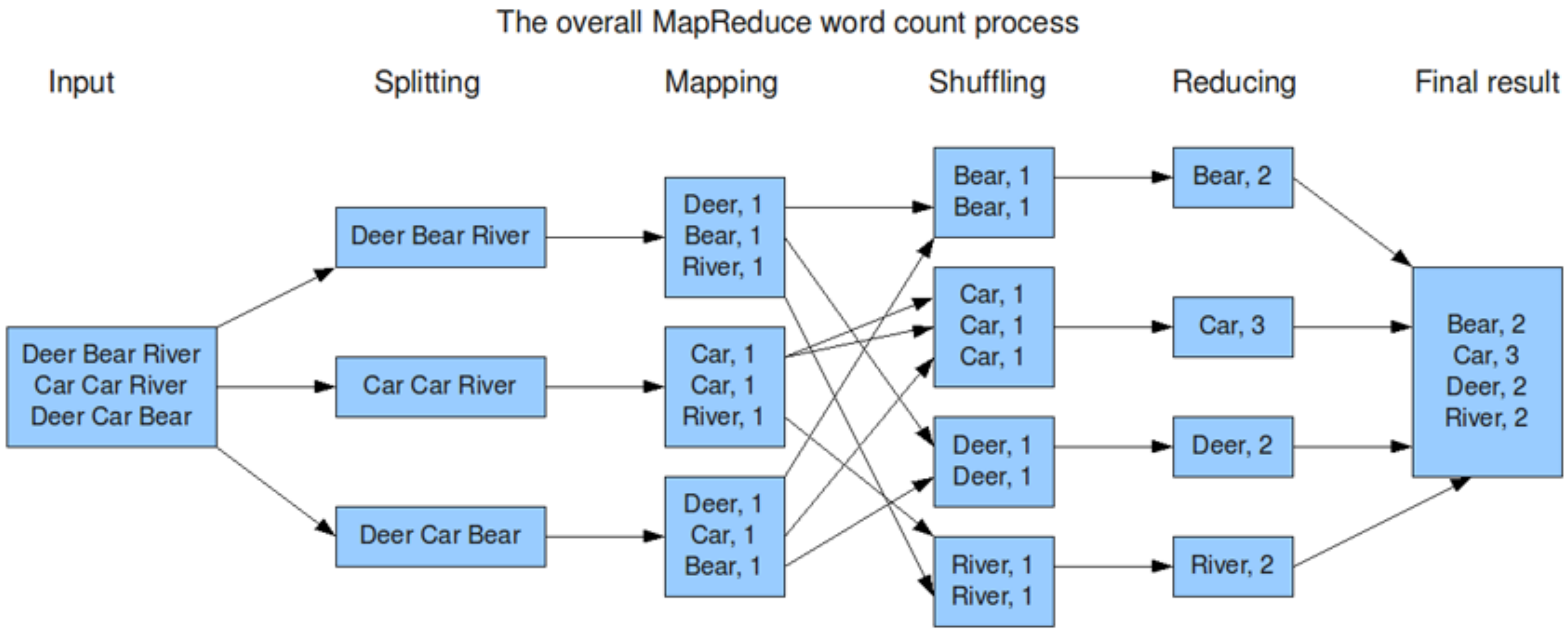
- Job 1:
  - map:
    - input: text
    - emit(word, 1)
  - reduce:
    - input (word, Iterable <Int>),
      sum all values
    - emit: (word, sum of values)

Word Frequency

- Job 1:
  - map:
    - input: text
    - emit(word, 1)
  - reduce:
    - input (word, Iterable <Int>), sum all values
    - emit: (word, sum of values)

- Job 2:
  - map:
    - input: <word, count>
    - emit(count, word)
  - reduce:
    - input: (word, Iterable <Int>), sum all values, put in a map
    - iterate over the map, emit: (word, count) or emit: (count, word)

Word Frequency: Recap



Picture: bigdataprojects.org

Custom Comparator

- LongWritable.DecreasingComparator (output key needs to be LongWritable)
- ReverseOrderComparator (old API)

Distributed Cache

- During job setup:

```
job.addCacheFile(new URI(“hdfs://sandbox.hortonworks.com:
8020”+“/user/hue/stopwords/stop.txt”))
```

- Mapper setup

```
Set<String> stopWords;
@Override
public void setup(Context context) throws IOException {
//Path stopWordsFile =
DistributedCache.getLocalCacheFiles(context.getConfiguration())[0];

Path stopWordsFile = context.getLocalCacheFiles();
//read file and add stopwords to a HashSet

}
```

```
    Set<String> stopWords;
    @Override
    public void setup(Context context) throws IOException {
      stopWords = new HashSet<String>();

      Path stopWordsFile = context.getLocalCacheFiles();
      BufferedReader fis;
      String line = null;
      fis = new BufferedReader(new 
FileReader(stopWordsFile.toString()));
      while ((line = fis.readLine()) != null) {
        stopWords.add(line);
      }
      fis.close();
    }
```