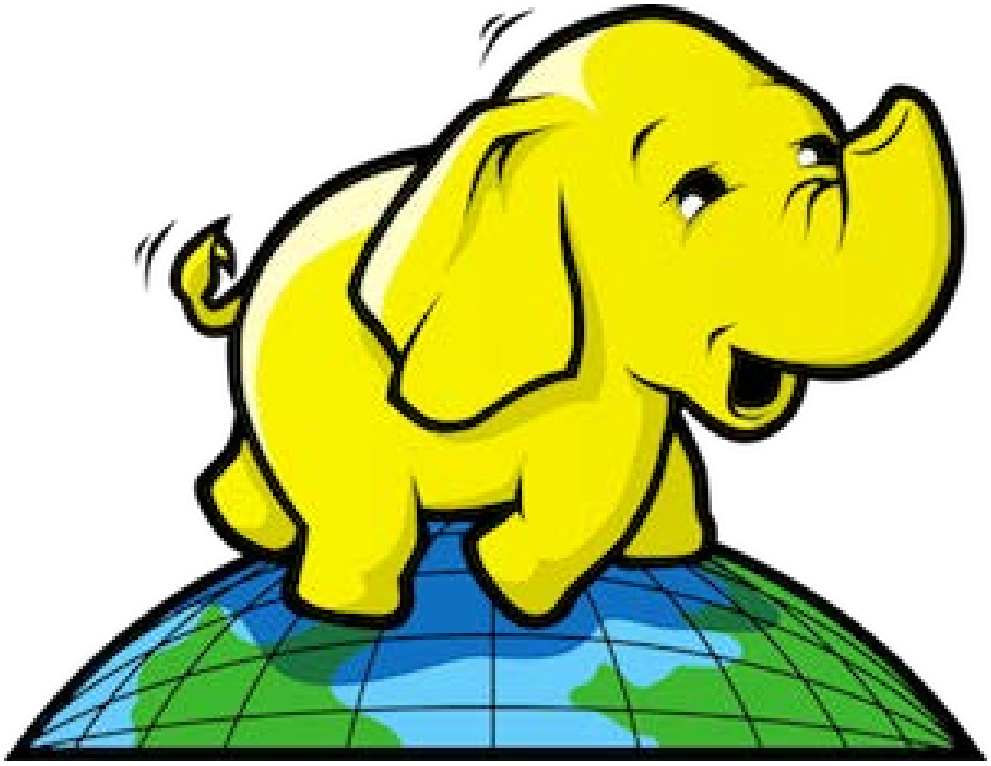
Introducing Hadoop MapReduce



Paolo Merialdo
Luca Menichetti Fabrizio Rebecca

Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

Install Pseudo-Distributed Mode

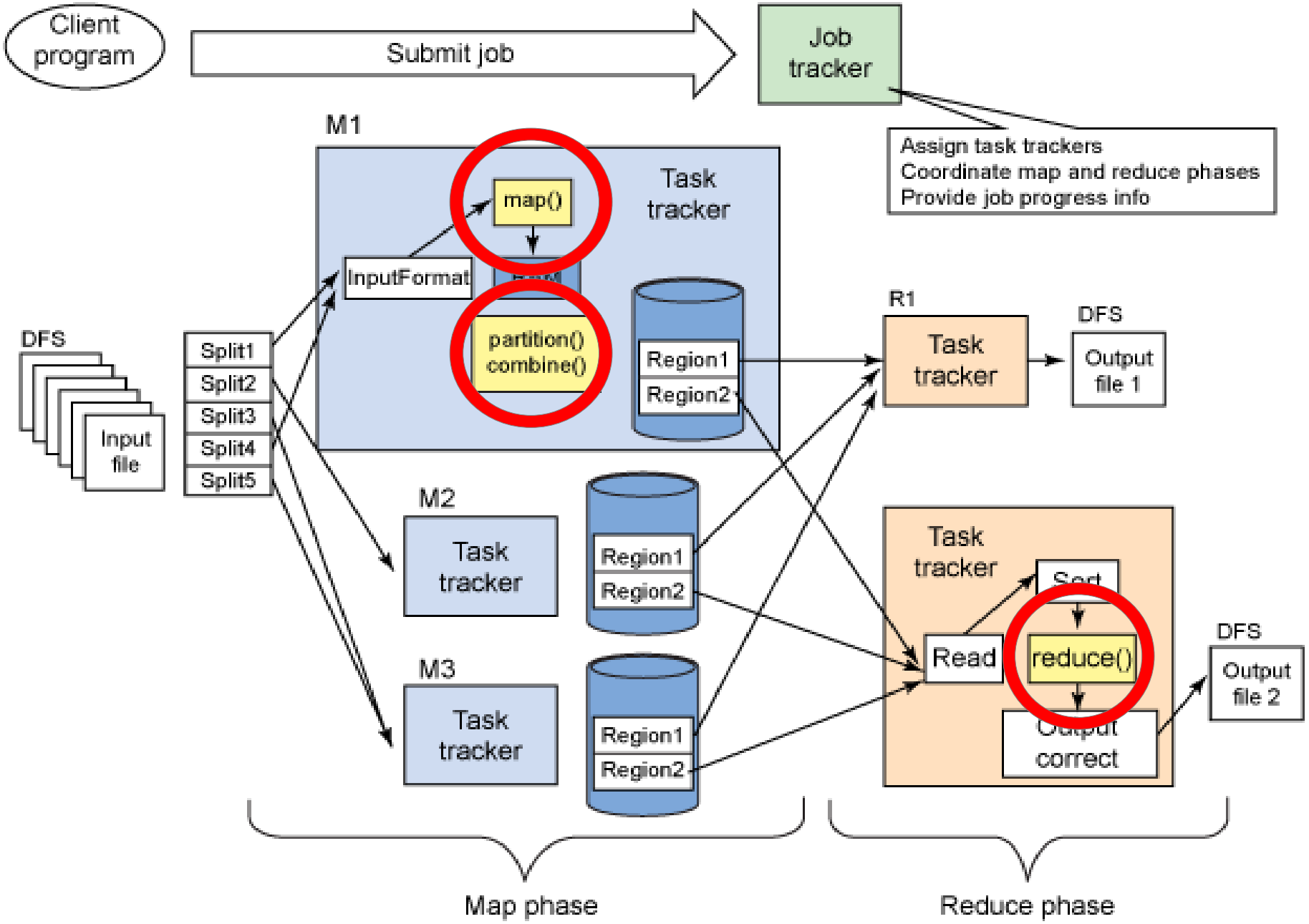
Good luck!

https://github.com/H4ml3t/Introducing-Hadoop

Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

Architecture



Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

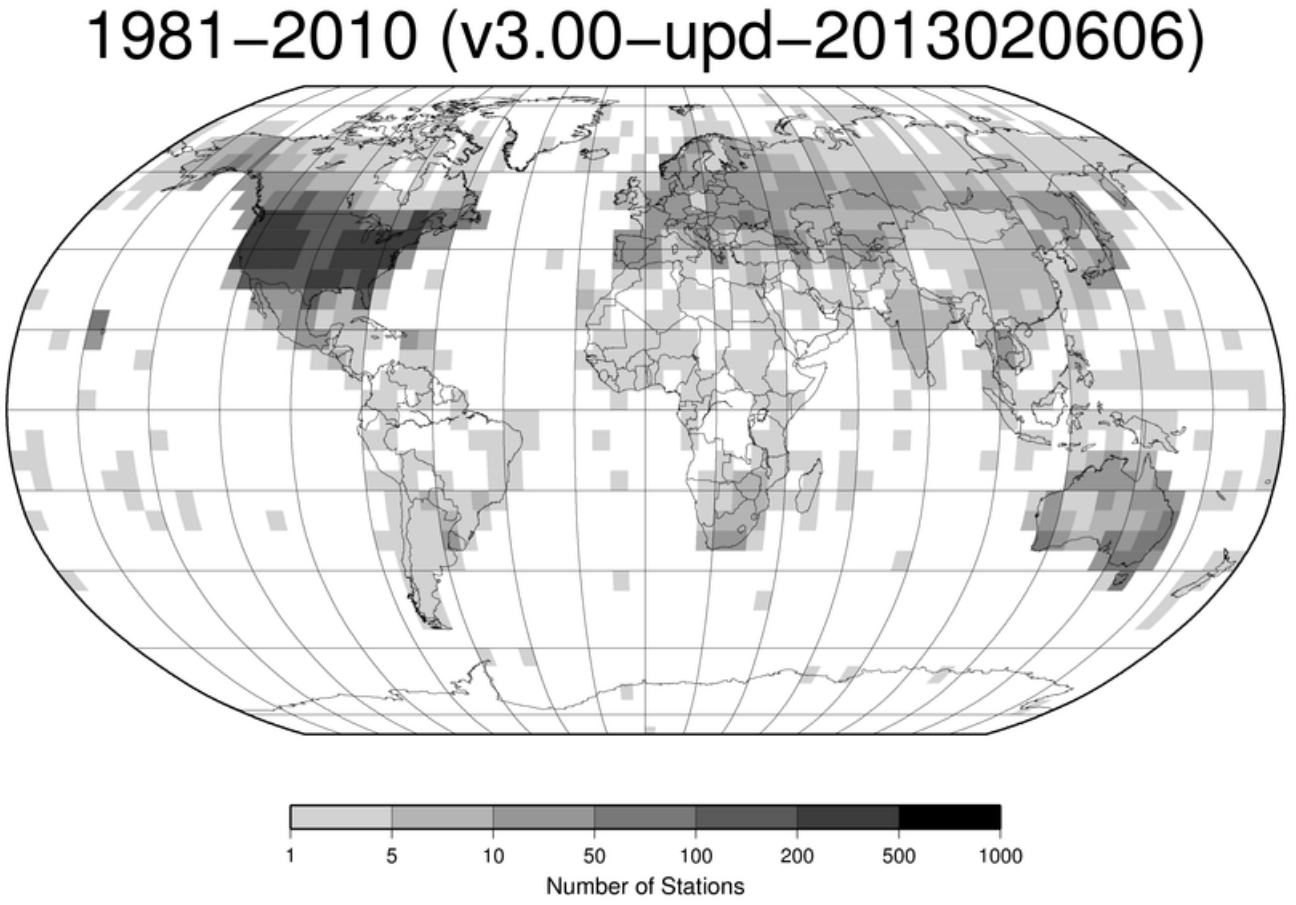
Example: Temperature

We collected information from https://www.ncdc.noaa.gov/ (National Climatic Data Center) about the atmosphere.

For each year (starting from 1763), for each month, for each day, for each hours, for each existing meteorogical stations in the world we have an entry in a file that specifies: data, time, id, air temperature, pressure, elevation, latitude, longitude...

Example: Temperature

What if we want to know the max temperature in each year?



Temperature - data format

File entries example:

...

0057332130999991958010103004+51317+028783FM-12+017199999V0203201N00721004501CN0100001N9-00211-01391102681

0057332130999991959010103004+51317+028783FM-12+017199999V0203201N00721004501CN0100001N9+00651-01391102681

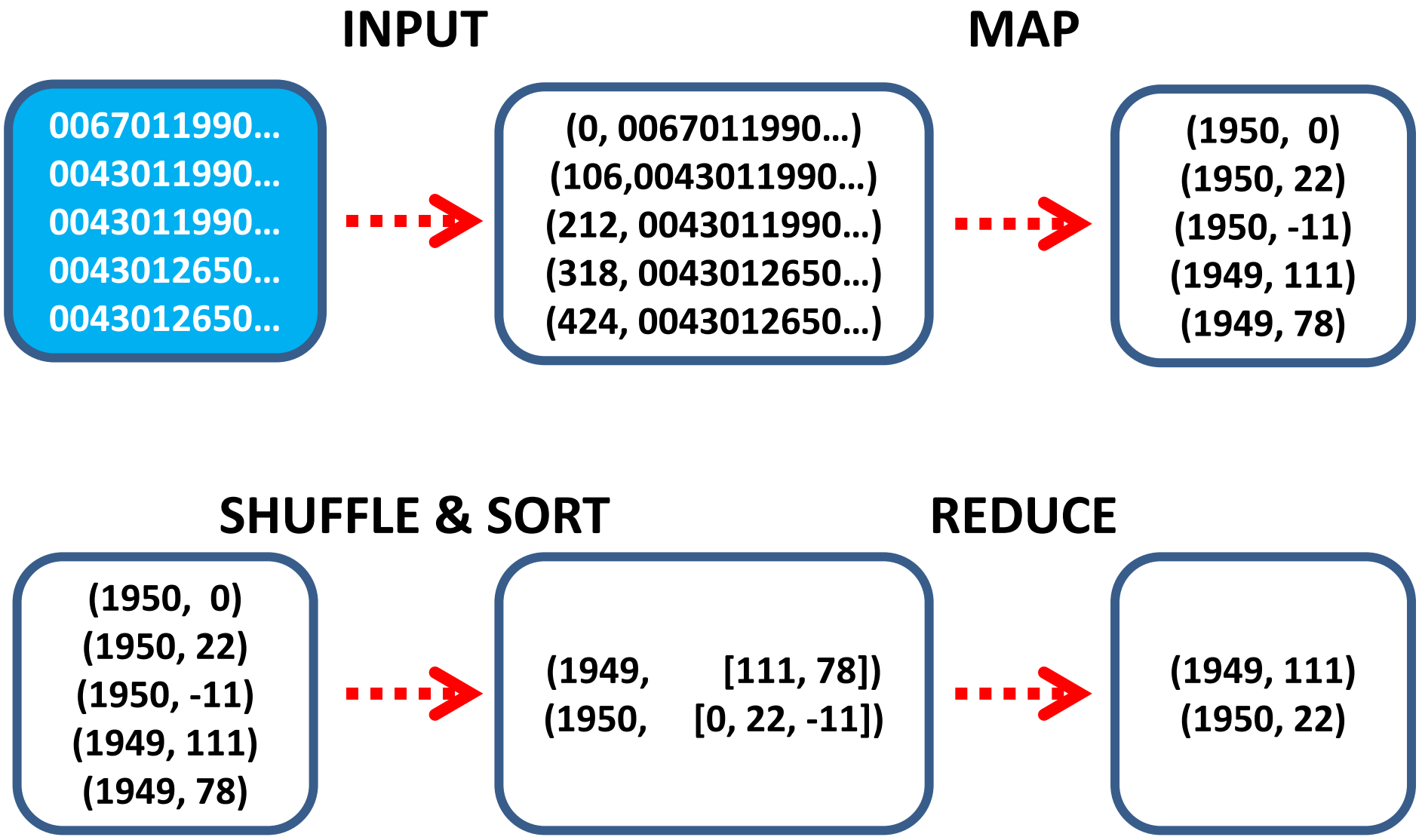
0057332130999991960010103004+51317+028783FM-12+017199999V0203201N00721004501CN0100001N9+00541-01391102681

...

Year

Degrees Celsius x10

Temperature – logical data flow



Temperature – Mapper

map (k1, v1) → [(k2, v2)]

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class MaxTemperatureMapper extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, IntWritable> {

        private static final int MISSING = 9999;

        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,,Reporter reporter)
                throws IOException {

                String line = value.toString();
                String year = line.substring(15, 19);

                int airTemperature;

                if (line.charAt(87) == '+') {
                        airTemperature = Integer.parseInt(line.substring(88, 92));
                } else {
                        airTemperature = Integer.parseInt(line.substring(87, 92));
                }

                if ( airTemperature != MISSING ) {
                        output.collect(new Text(year), new IntWritable(airTemperature));
                }
        }
}
```

Temperature – Reducer (1)

reduce (k2, [v2]) → [(k3, v3)]

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class MaxTemperatureReducer extends MapReduceBase
        implements Reducer<Text, IntWritable, Text, DoubleWritable> {

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
    DoubleWritable> output, Reporter reporter) throws IOException {

        int maxValue = Integer.MIN_VALUE;

        while (values.hasNext()) {
                maxValue = Math.max(maxValue, values.next().get());
        }

        output.collect(key, new DoubleWritable(((double)maxValue)/10));
    }

}
```

Temperature – Reducer (2)

reduce (k2, [v2]) → [(k3, v3)]

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class MaxTemperatureReducer extends MapReduceBase
        implements Reducer<Text, IntWritable, Text, DoubleWritable> {

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
    DoubleWritable> output, Reporter reporter) throws IOException {

        int maxValue = Integer.MIN_VALUE;

        while (values.hasNext()) {
                maxValue = Math.max(maxValue, values.next().get());
        }

        output.collect(key, new DoubleWritable(((double)maxValue)/10));
    }

}
```

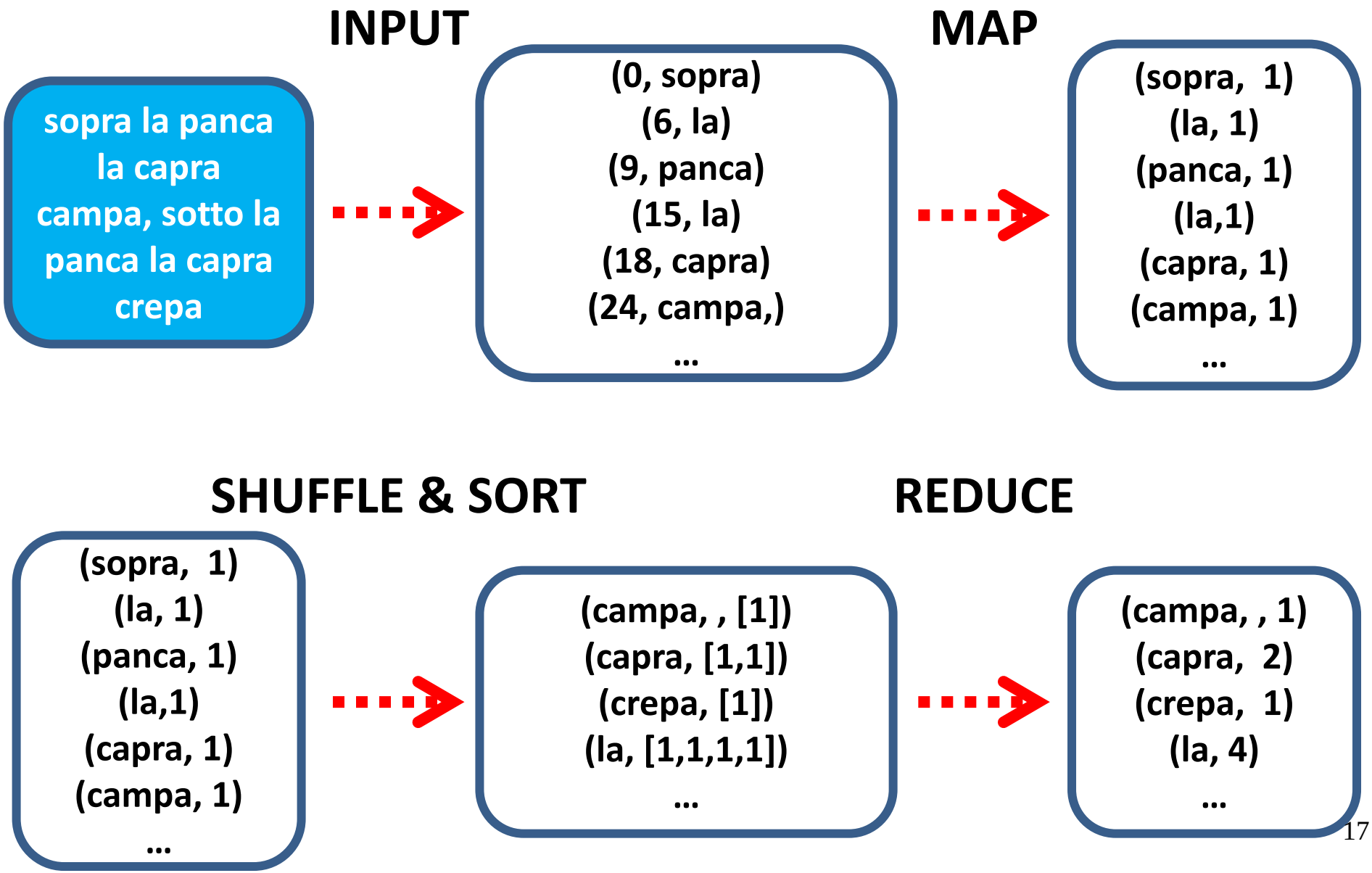
Temperature - Job

```
public class MaxTemperature {

    public static void main(String[] args) throws IOException {

        JobConf conf = new JobConf(MaxTemperature.class);
        conf.setJobName("Max temperature");

        FileInputFormat.addInputPath(conf, new Path("temperature.txt"));
        FileOutputFormat.setOutputPath(conf, new Path("output"));

        conf.setMapperClass(MaxTemperatureMapper.class);
        conf.setReducerClass(MaxTemperatureReducer.class);

        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(DoubleWritable.class);

        JobClient.runJob(conf);
    }
}
```

Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

Word Count – logical data flow



Word Count – Mapper (1)

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {

        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);

        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

Word Count – Mapper (2)

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {

        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);

        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

Word Count – Reducer (1)

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

        int sum = 0;

        for (IntWritable value : values) {
                sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }
}
```

Word Count – Reducer (2)

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

        int sum = 0;

        for (IntWritable value : values) {
                sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }
}
```

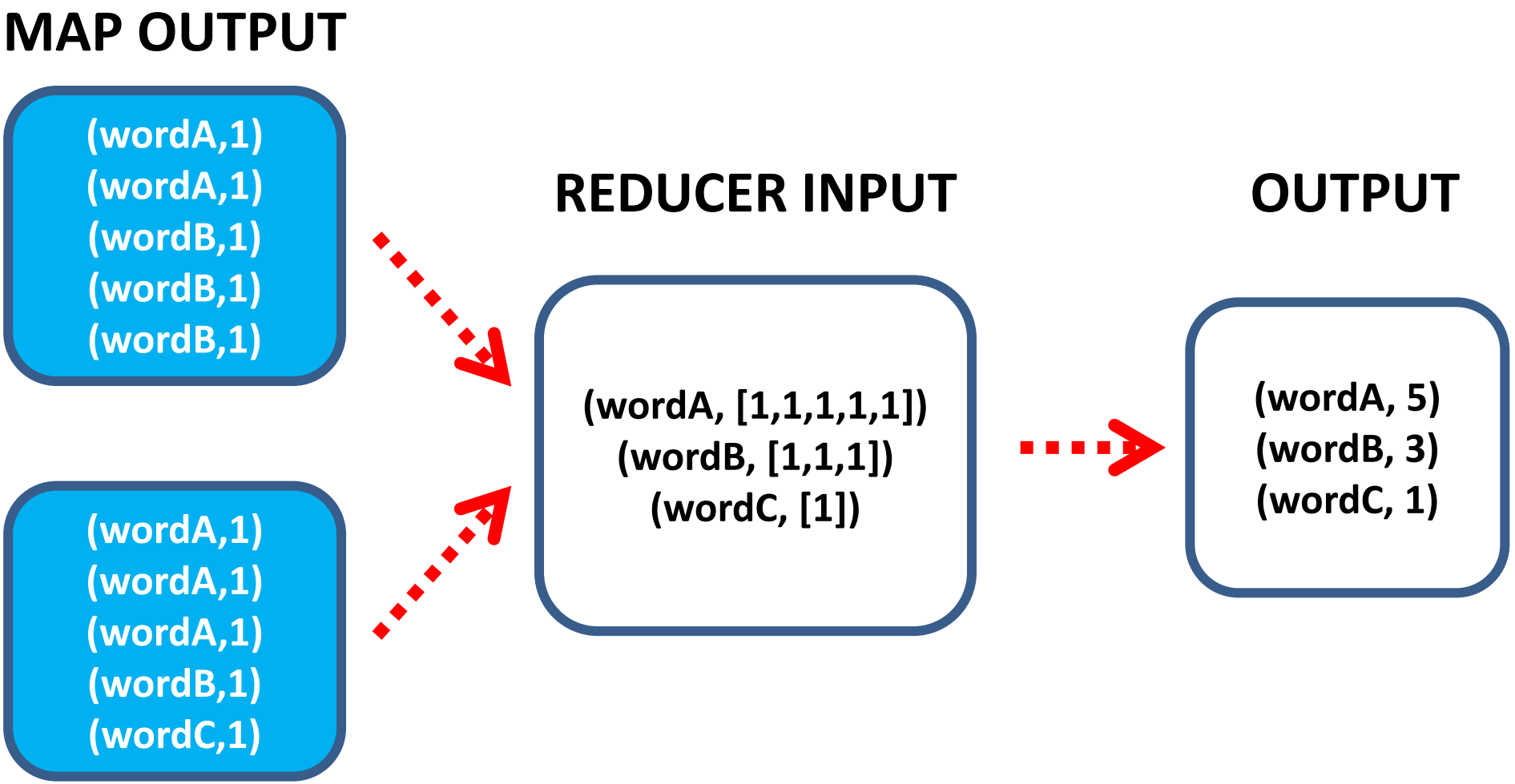
Word Count – Job

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

public class WordCount {

    public static void main(String[] args) throws Exception {

        Job job = new Job(new Configuration(), "WordCount");
        job.setJarByClass(WordCount.class);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.waitForCompletion(true);
    }
}
```

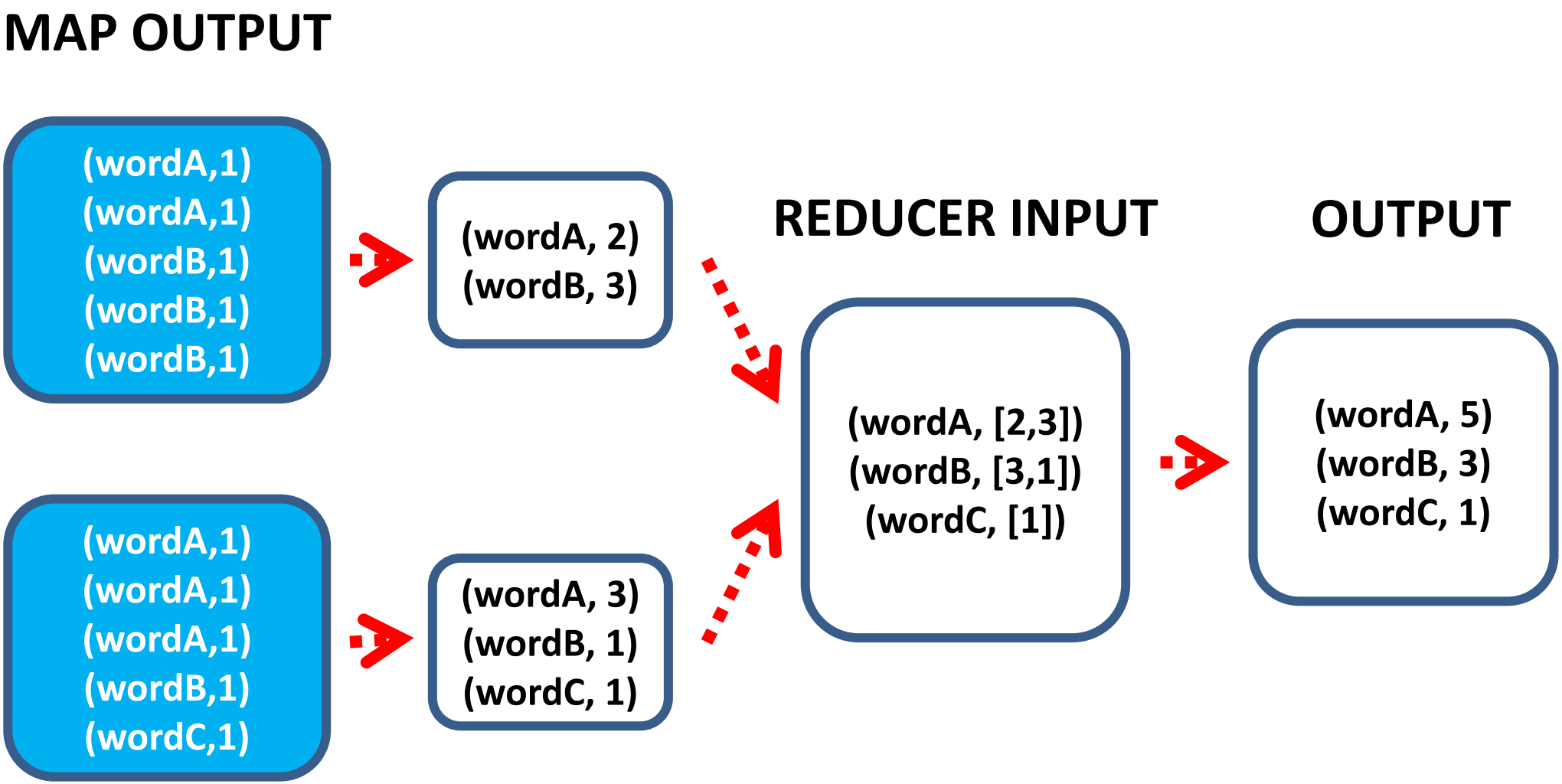
Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

Word Count: Without Combiner



Word Count: With Combiner



Combiner

In the job configuration:

…

job.setMapperClass(WordCountMapper.class);

job.setCombinerClass(WordCountReducer.class);

job.setReducerClass(WordCountReducer.class);

…

Watch for types!

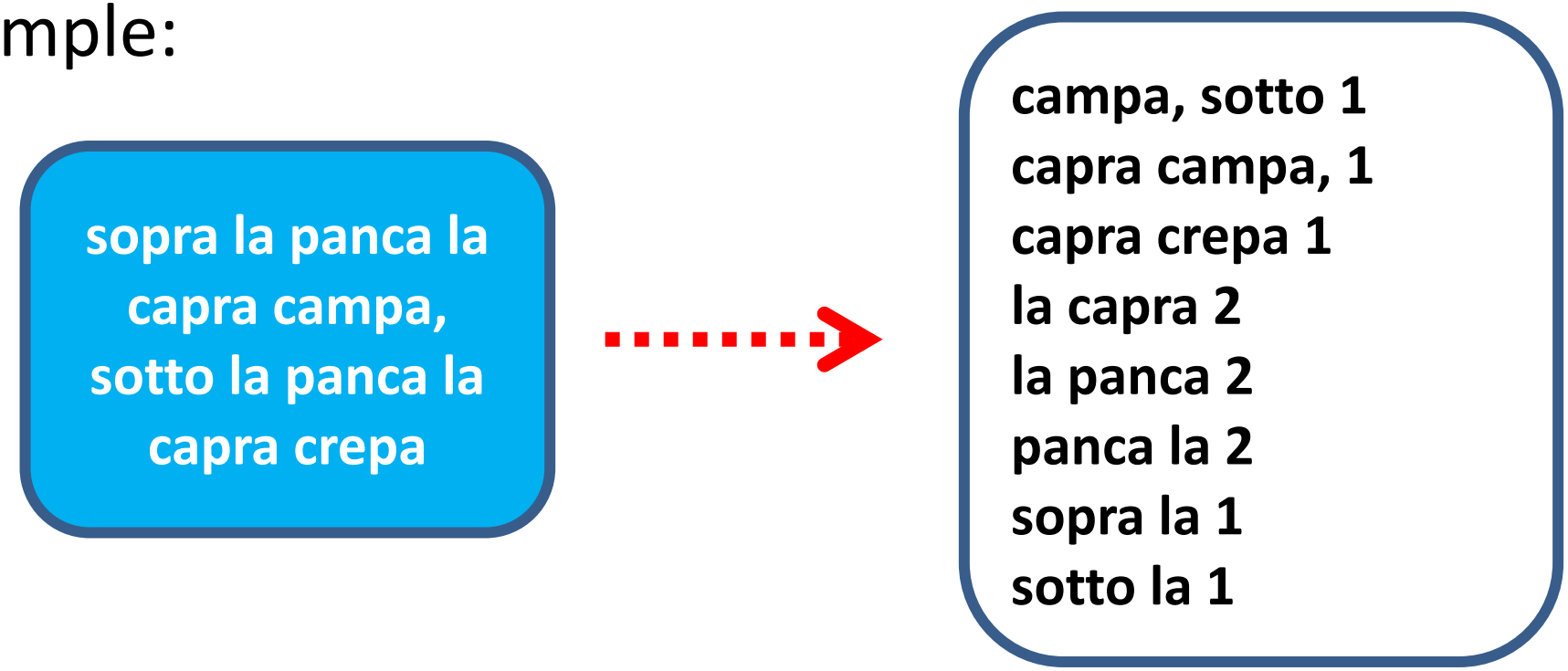
Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

Bigram Count

A bigram is a couple of two consecutive words.

Example:



To represent a Bigram we want to use a custom Writable type (BigramWritable)

Bigram Count

```
import org.apache.hadoop.io.*;

public class BigramWritable implements WritableComparable<BigramWritable> {

    private Text leftBigram;
    private Text rightBigram;

    public BigramWritable(Text left, Text right){
        this.leftBigram = left;
        this.rightBigram = right;
    }
    @Override
    public void readFields(DataInput in) throws IOException {
        leftBigram = new Text(in.readUTF());
        rightBigram = new Text(in.readUTF());
    }

    @Override
    public void write(DataOutput out) throws IOException {
        out.writeUTF(leftBigram.toString());
        out.writeUTF(rightBigram.toString());
    }
    …
```

Bigram Count

```
…

public void set(Text prev, Text cur) {
        leftBigram = prev;
        rightBigram = cur;
}

@Override
public int hashCode() {
        return leftBigram.hashCode() + rightBigram.hashCode();
}

@Override
public boolean equals(Object o) {
        if (o instanceof BigramWritable) {
                BigramWritable bigram = (BigramWritable) o;
                return leftBigram.equals(bigram.leftBigram) &&
                rightBigram.equals(bigram.rightBigram);
        }
        return false;
}

…
```

Bigram Count

```
        …

        @Override
        public int compareTo(BigramWritable tp) {
                int cmp = leftBigram.compareTo(tp.leftBigram);
                if (cmp != 0) {
                        return cmp;
                }
                return rightBigram.compareTo(tp.rightBigram);
        }
}
```

Bigram Count - Mapper

```
public class BigramCountMapper extends Mapper<LongWritable, Text, BigramWritable,
IntWritable> {

    private static final IntWritable ONE = new IntWritable(1);
    private static final BigramWritable BIGRAM = new BigramWritable();

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

        String line = value.toString();
        String prev = null;
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
                String cur = itr.nextToken();
                if (prev != null) {
                    BIGRAM.set( new Text(prev), new Text(cur) );
                    context.write(BIGRAM, ONE);
                }
                 prev = cur;
        }
     }
}
```

Bigram Count - Reducer

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class BigramCountReducer extends Reducer<BigramWritable,IntWritable,Text, IntWritable> {

    private final static IntWritable SUM = new IntWritable();

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

        int sum = 0;
        Iterator<IntWritable> iter = values.iterator();
        while (iter.hasNext()) {
                sum += iter.next().get();
        }
        SUM.set(sum);
        context.write(new Text( key.toString() ), SUM);
    }
}
```

Bigram Count - Job

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.util.Tool;

public class BigramCount extends Configured implements Tool {

        private BigramCount() {}
        …
        public int run(String[] args) throws Exception {
                …
                Job job = new Job(getConf());
                job.setJobName(BigramCount.class.getSimpleName());
                job.setJarByClass(BigramCount.class);
                …
                job.waitForCompletion(true);
        }

        public static void main(String[] args) throws Exception {
                ToolRunner.run(new BigramCount(), args);
        }

}
```

Summary

- Install Pseudo-Distributed Mode
- Architecture
- Example: Temperature
- Example: Word Count
- Combiner
- Example: Bigram Count (with custom writable)
- How to run a job

How to run a job

How to make a jar file from eclipse:

1) Right click on the project folder
2) Export -> Java -> Runnable Jar
3) Select from the launch configuration the current project
4) Select a destination to export the .jar file
5) Select «Extract required libraries ..» and click Finish

How to copy a file in HDFS (not needed in standalone mode):

hadoop fs –copyFromLocal <src> <localDestination>

How to run a job

How to run a MapReduce job

hadoop jar <filejar> [arguments..]