Introduction to MapReduce Programming

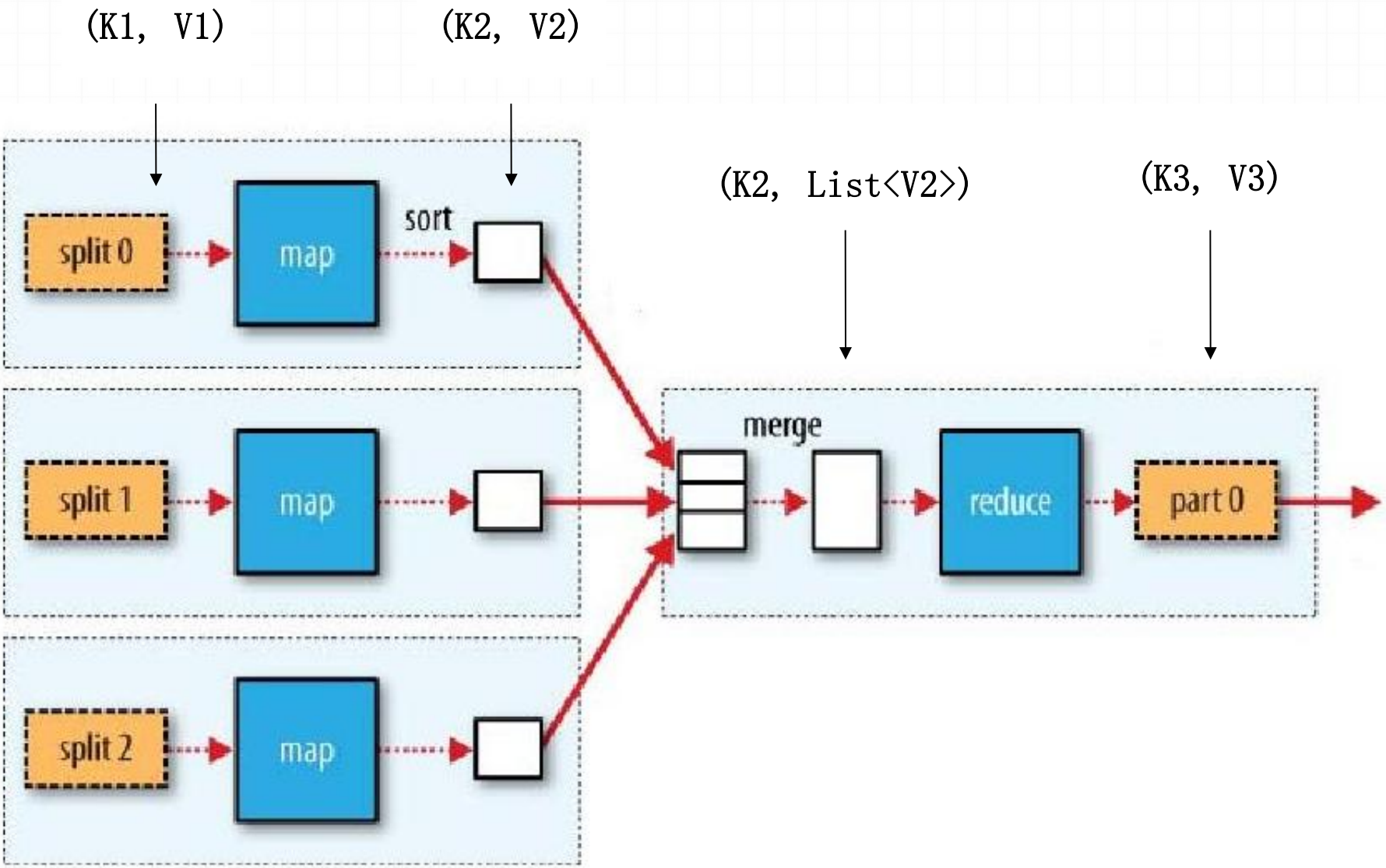
&

Local Hadoop Cluster Accesses Instructions

Rozemary Scarlat

August 31, 2011

Dataflow in a MR Program

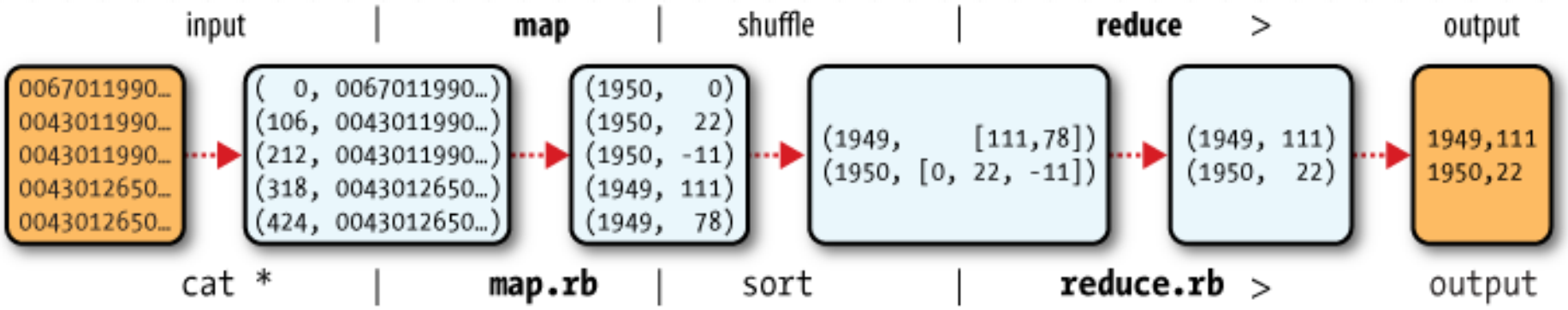


- We have temperature readings for the years 1901 – 2001 and we want to compute the maximum for each year
- In our temperature data set, each line looks like this:

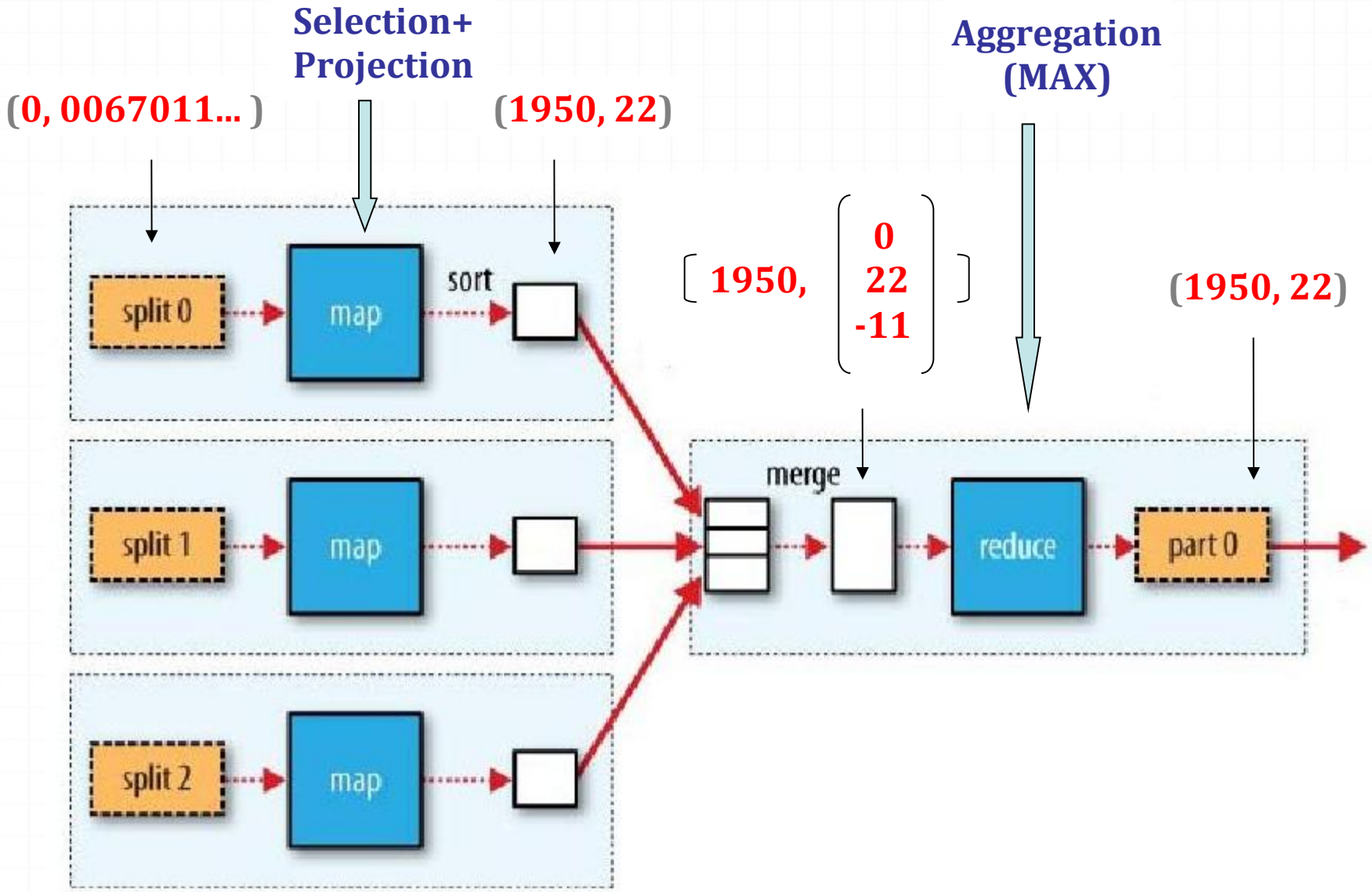
  0067011990999991950051507004...9999999N9+00001+99999999999...
  0043011990999991950051512004...9999999N9+00221+99999999999...

- We know that characters 16 – 19 represent the year, characters 88 – 92 represent the temperature and character 93 represents the quality code

  (0, 0067011990999991**950**051507004...9999999N9+**0000**1+99999999999...)
  (106, 0043011990999991**950**051512004...9999999N9+**0022**1+99999999999...)



Implementation in MapReduce



Mapper

```
static class MaxTemperatureMapper
    extends Mapper<LongWritable, Text, Text, IntWritable> {

    private static final int MISSING = 9999;

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

      String line = value.toString();
      String year = line.substring(15, 19);
      int airTemperature;
      if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
          airTemperature = Integer.parseInt(line.substring(88, 92));
          } else {
              airTemperature = Integer.parseInt(line.substring(87, 92));
          }
      String quality = line.substring(92, 93);
      if (airTemperature != MISSING && quality.matches("[01459]")) {
        context.write(new Text(year), new IntWritable(airTemperature));
        }
      }
    }
```

Reducer

```
static class MaxTemperatureReducer
    extends Reducer<Text, IntWritable, Text, IntWritable> {

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

        int maxValue = Integer.MIN_VALUE;
        for (IntWritable value : values) {
            maxValue = Math.max(maxValue, value.get());
        }
        context.write(key, new IntWritable(maxValue));
    }
}
```

Main

```
public static void main (String[] args) throws Exception {
    if (args.length != 2) {
        System.err.println("Usage: MaxTemperature <input path> <output path>");
        System.exit(-1);
    }

    Configuration conf = new Configuration();
    Job job = new Job(conf);
    job.setJarByClass(MaxTemperature.class);

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

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

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

    System.exit(job.waitForCompletion(true) ? 0 : 1);

}
```

Beyond MaxTemperature

- What if we want to get the average temperature for a year?
- What if you are only interested in the temperature in Durham? (Assume the station ID at Durham is 212)

Local Hadoop Cluster

- The master node is `hadoop21.cs.duke.edu`
- The slave nodes are `hadoop[22,24-36].cs.duke.edu`
- Online jobtracker address*:

`http://hadoop21.cs.duke.edu:50030/jobtracker.jsp`

- Online HDFS health*:

http://hadoop21.cs.duke.edu:50070/dfshealth.jsp

* Accesible only from within the CS trusted network. Solution:

1. ssh to any node and then use lynx.
2. build "ssh -D port" connection to any node, set proxy in your browser

- Now, let's see how to compile and run a MapReduce job on the local cluster

- You can find the detailed instructions at the course website:
  http://www.cs.duke.edu/courses/fall10/cps216/TA_material/cluster_instructions

Mapper (old API)

```
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) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (airTemperature != MISSING && quality.matches("[01459]")) {
      output.collect(new Text(year), new IntWritable(airTemperature));
    }
  }
}
```

Reducer (old API)

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

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

    int maxValue = Integer.MIN_VALUE;
    while (values.hasNext()) {
      maxValue = Math.max(maxValue, values.next().get());
    }
    output.collect(key, new IntWritable(maxValue));
  }
}
```

Main (old API)

```
public static void main(String[] args) throws IOException {
  if (args.length != 2) {
    System.err.println("Usage: MaxTemperature <input path> <output path>");
    System.exit(-1);
  }

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

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

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

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

  JobClient.runJob(conf);
}
```