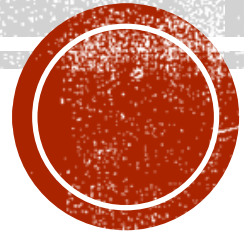
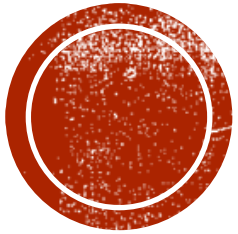
MAPREDUCE – CONT'D



MAPREDUCE DEVELOPMENT WORKFLOW

DEVELOPMENT WORKFLOW

- Understand the problem
- Decomposing a problem into MapReduce Job(s)
- Write code – map, reduce, combine and driver
- Write a unit testing to make sure it works as expected with a set of controlled input
  - MRUnit
- Test it on the local data (single node cluster)
- Run on the real cluster
  - Debugging
- Tuning the tasks
  - Profiling
  - Tune up



BIGDATA MAPREDUCE EXAMPLE

BIGDATA EXAMPLE
A WEATHER DATASET FROM NCDC

- Objective: find the maximum recorded temperature by year from NCDC weather records
- The data format:
  - The data is stored using a line-oriented ASCII format, in which each line is a record.

```
0067011990999991950051507004...9999999N9+00001+99999999999...
0043011990999991950051512004...9999999N9+00221+99999999999...
0043011990999991950051518004...9999999N9-00111+99999999999...
0043012650999991949032412004...0500001N9+01111+99999999999...
0043012650999991949032418004...0500001N9+00781+99999999999...
```

year	temperature	quality
15-19	87-92	93

- Temperature is ***9999*** if the value is missing
- Quality is good if the value is either ***0,1, 4, 5, or 9***

FORMAT OF NCDC RECORD

```
0057
332130   # USAF weather station identifier
99999    # WBAN weather station identifier
19500101 # observation date
0300     # observation time
4
+51317   # latitude (degrees x 1000)
+028783  # longitude (degrees x 1000)
FM-12
+0171    # elevation (meters)
99999
V020
320      # wind direction (degrees)
1        # quality code
N
0072
1
00450    # sky ceiling height (meters)
1        # quality code
C
N
010000   # visibility distance (meters)
1        # quality code
N
9
-0128    # air temperature (degrees Celsius x 10)
1        # quality code
-0139    # dew point temperature (degrees Celsius x 10)
1        # quality code
10268    # atmospheric pressure (hectopascals x 10)
1        # quality code
```

ANALYZING DATA WITH HADOOP

- Map phase
  - Extract needed data (year and temperature) for each record
  - Filter out bad records

- Reduce phase
  - Find max temperature for each year

MAP FUNCTION – EXTRACT AND FILTER

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

  private static final int MISSING = 9999;

  @Override
  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));
    }
  }
}
```

REDUCE FUNCTION - FINDING MAX TEMPERATURE

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

  @Override
  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));
  }
}
```

DRIVER - FINDING MAX TEMPERATURE

```
public class MaxTemperature {

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

    Job job = new Job();
    job.setJarByClass(MaxTemperature.class);
    job.setJobName("Max temperature");

    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);
  }
}
```

A Job object forms the specification of the job and gives you control over how the job is run.

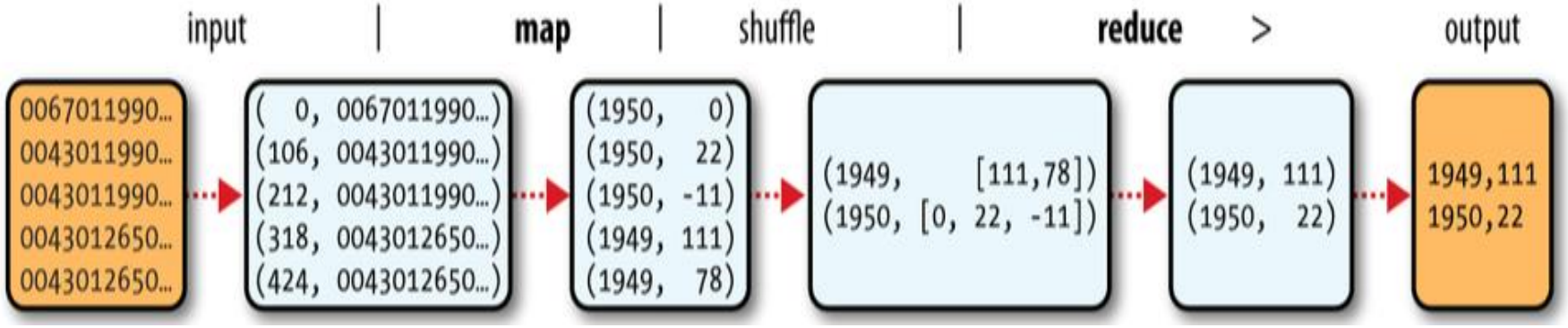
Input directory

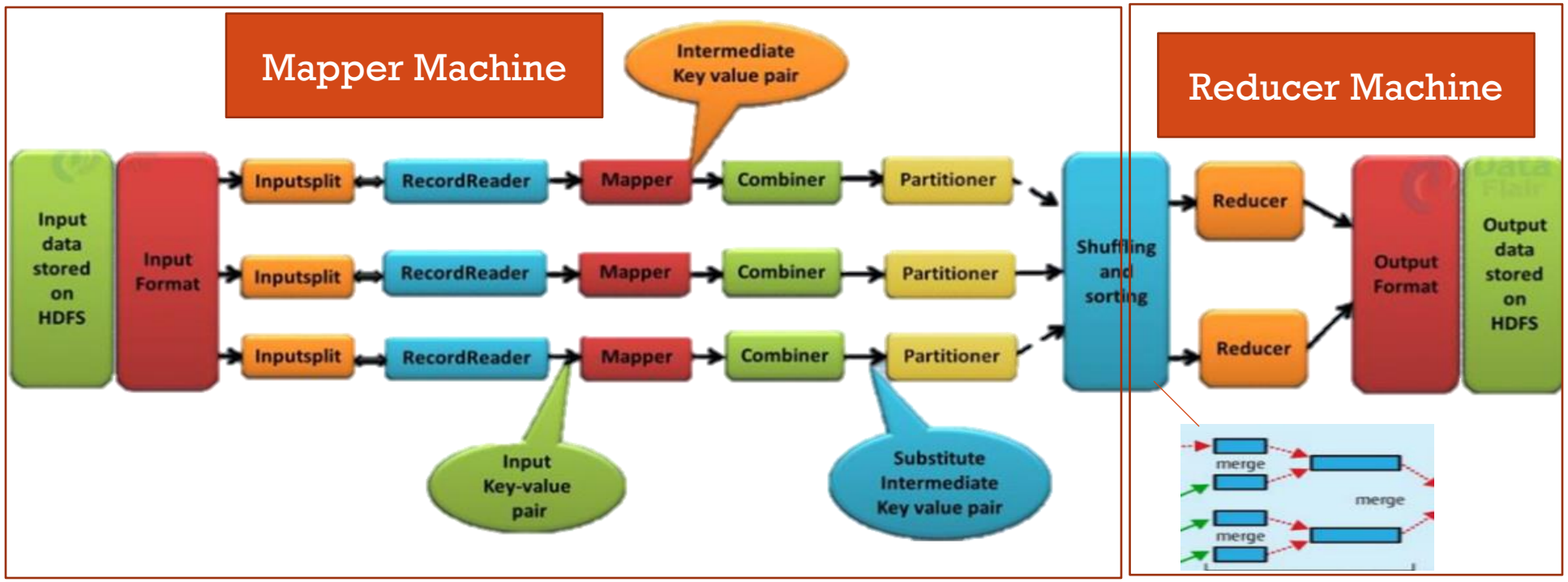
output directory

Reduce's key and value output
Since Map's are the same, no need to specify

flag indicating whether verbose output is generated
When true, the job writes information about its progress to the console

MAPREDUCE IS NOT ONLY MAPREDUCE!





INPUTFORMAT – HANDLE SPLITS AND RECORDREADER

- An abstract class that deals with InputSplits
  - Create input splits and divide them into records

```
public abstract class InputFormat<K, V> {
  public abstract List<InputSplit> getSplits(JobContext context)
      throws IOException, InterruptedException;

  public abstract RecordReader<K, V>
      createRecordReader(InputSplit split, TaskAttemptContext context)
          throws IOException, InterruptedException;
}
```

- The client running the job calculates the splits by calling getSplits(), then send them to app master which uses their storage locations to schedule map tasks
- The map task uses the split to createRecordReader() method to obtain a RecordReader for that split

INPUT SPLITS AND RECORDS

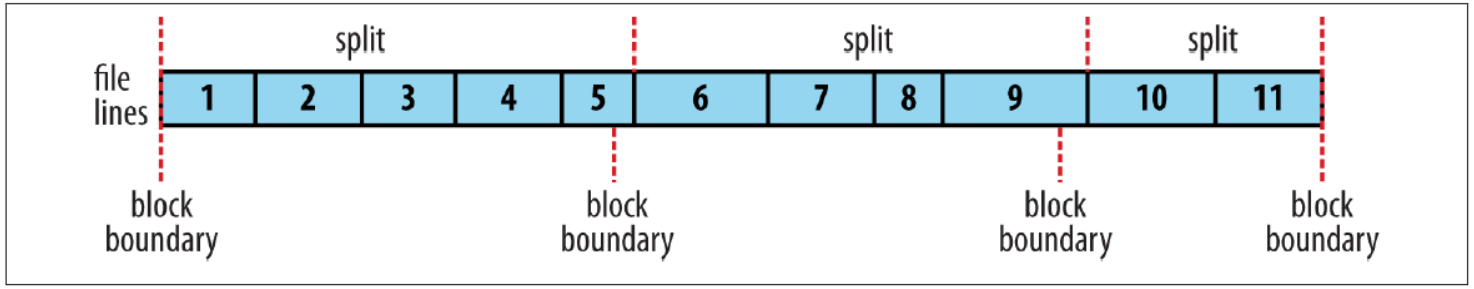
- An input split – a chunk of input that is processed by a single mapper
- Each split is divided into records
- No data, just a reference to data
- Represented by the Java InputSplit class

```
public abstract class InputSplit {
  public abstract long getLength() throws IOException, InterruptedException;
  public abstract String[] getLocations() throws IOException,
      InterruptedException;
}
```

- Location is used to place a map task (closer to split's data)
- Length is used to order splits to be processed, ie. larger split gets processed first to minimize run time (greedy approximation algorithm - https://en.wikipedia.org/wiki/Partition_problem)

THE RELATIONSHIP BETWEEN INPUT SPLITS AND HDFS BLOCKS

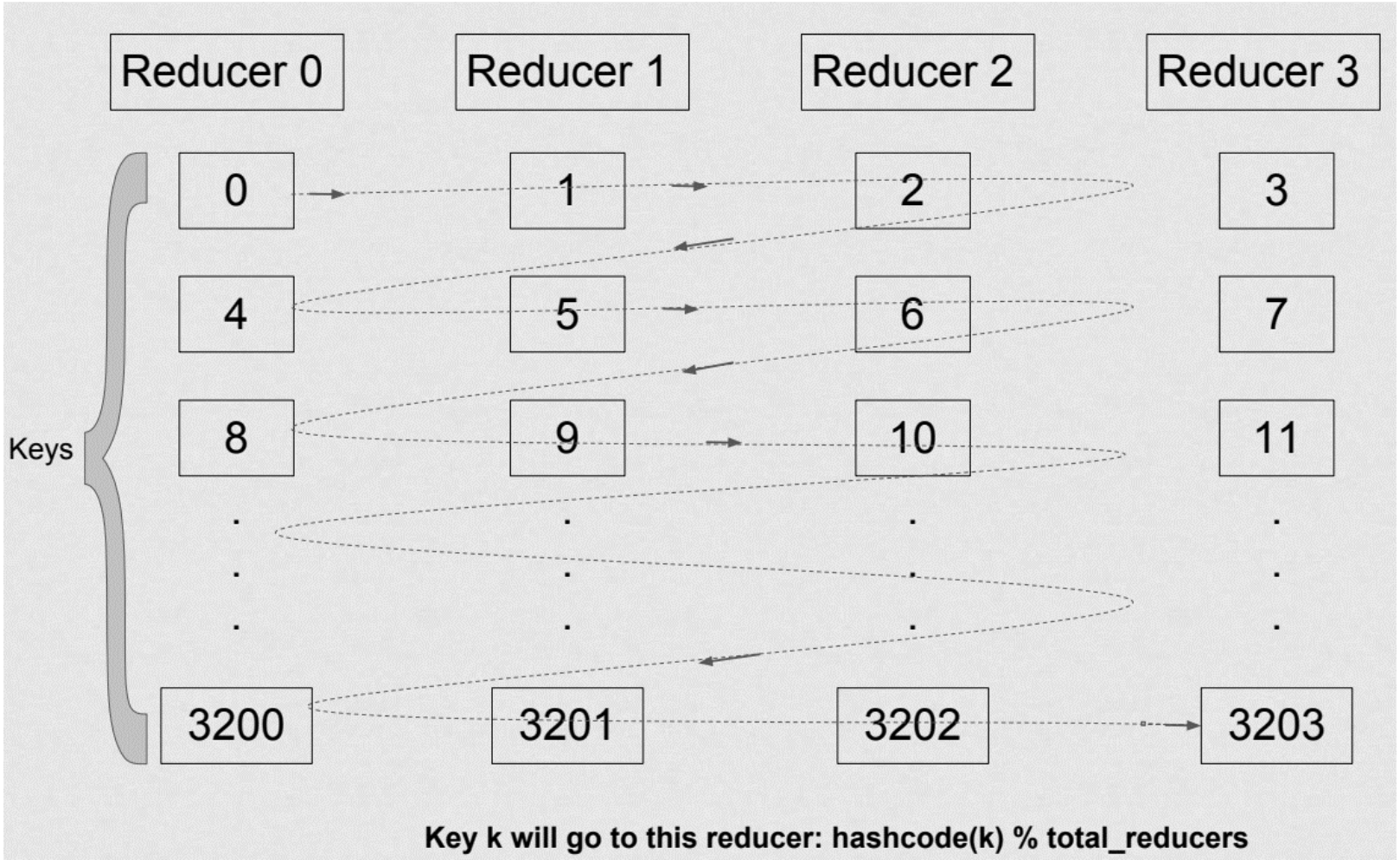
- The logical records that FileInputFormats define usually do not fit neatly into HDFS blocks
- Some remote read might be needed



COMBINERS

- Hadoop's Sorting and Shuffling takes considerable amount of time.
  - It's preferable to limit the amount of data that should be shuffled as needed.
  - Leveraging combiners is one way to optimize the process.
  - Combiners can be used to implement Local Aggregation.
    - Local aggregation aims to reduce the amount of intermediate key,value pair to **improve efficiency**

QUESTION: HOW DO REDUCERS PICK MAPPER RESULTS?
ANSWER: PARTITIONERS



SORTING — ORGANIZE INPUT AND OUTPUT FOR OPTIMIZATION

Customize sorting in each stage

- Map stage
  - Default is sorting by text key
  - Change key type to IntWritable for integer sorting
  - Define new text key, ie. composite key
  - Rewrite how keys are compared
- Shuffle stage
  - Rewrite how keys are partitioned
- Merge stage (right before the reduction-phase)
  - Rewrite how to group data to pass to the reduce method

MAPREDUCE SORTING PROBLEM – SORTING BY VALUES

- The MapReduce framework sorts the records by key before they reach the reducers.
  - For any particular key, however, the values are not sorted.
  - The order that the values appear is not even stable from one run to the next, since they come from different map tasks, which may finish at different times from run to run.
- Generally speaking, most MapReduce programs are written so as not to depend on the order that the values appear to the reduce function.
  - However, it's possible to impose an order on the values by sorting and grouping the keys in a particular way.
  - The answer is Secondary Sorting Algorithm!
  - We will discuss Secondary Sorting algorithm in the MapReduce design patterns.

REFERENCES

- Hadoop The Definitive Guide 4th Edition, Tom White.
- Data-Intensive Text Processing with MapReduce, Jimmy Lin and Chris Dyer.