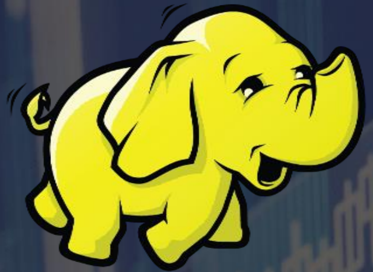
ACADGILD

LEARN. DO. EARN

BIG DATA

DEVELOPMENT



ACADGILD

Getting Started With MapReduce

Session 5

LEARN. DO. EARN

Agenda

ACADGILD

Sl. No.	Agenda Title
1	Building Principles
2	Introduction to MapReduce
3	Some More Real-World Examples
4	Broad Steps
5	Finding Out Maximum Temperature
6	Pseudo Code
7	Mapper Class
8	Reducer Class

Sl. No.	Agenda Title
9	Driver Code
10	Demo: Task 1
11	Exploring Methods of Mapper
12	Exploring Methods of Reducer
13	Demo: Task 2

Building Principles

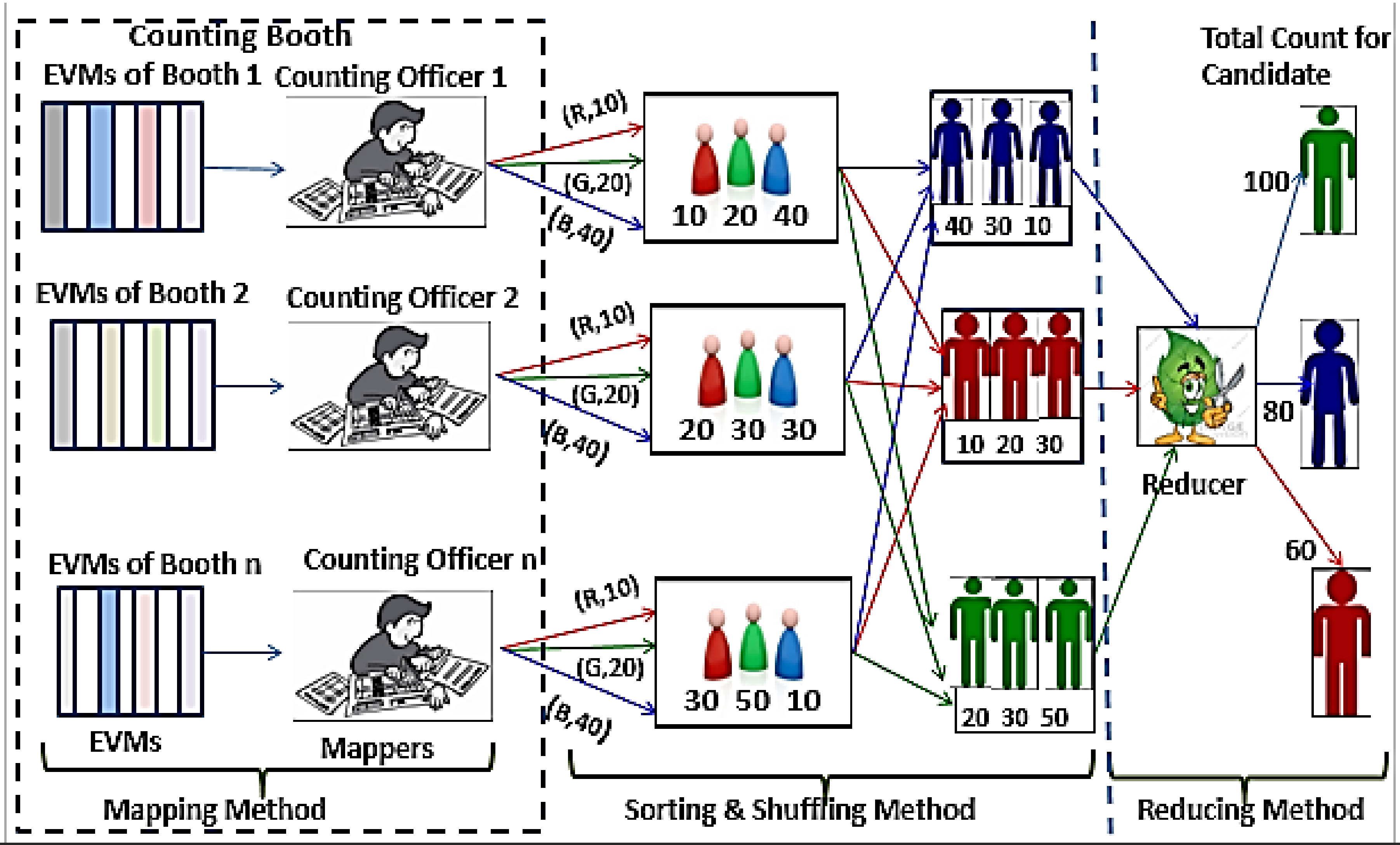
- The individual concepts of functions called map and reduce have been derived from functional programming languages (like C++ & Java) where they were applied to lists of input data.

- Another key underlying concept is that of "divide and conquer", where a single problem is broken into multiple individual subtasks. This approach becomes even more powerful when the subtasks are executed in parallel.

- The developer focuses on expressing the transformation between source and result data sets, and the Hadoop framework manages all aspects of job execution, parallelization, and coordination.

- **MapReduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks.**

Introduction to MapReduce



Some More Real-World Examples

ACADGILD

- An address book relates a name (key) and to contact information (value).
- A bank account uses an account number (key) to associate with the account details (value).
- The index of a book relates a word (key) to the pages on which it occurs (value).
- On a computer file system, filenames (keys) allow access to any sort of data,

such as text, images, and sound (values).

Introduction to MapReduce (Contd.)

- MapReduce sends code to distributed data, instead of bringing data to the actual code.

- MapReduce works by breaking the processing into two phases:
  - The map phase
  - The reduce phase

- Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer.

- The programmer also specifies two functions:
  - The map function
  - The reduce function

- There is a shuffle and sort phase in between.

Broad Steps

ACADGILD

- Map phase takes input in key-value pairs
- It produces output in the form of key-value pair.
- Output from various Map tasks are grouped together on the basis of key.
- Key and its associated set of values are sent to the Reduce phase.
- Reduce method operates on key and associated list of values.
- Output of Reduce is written to HDFS.

Finding Out Maximum Temperature

ACADGILD

- To visualize the way the map works, consider the following sample lines of input data
- (Dataset contains date in mm-dd-yyyy, zipcode and temperature)

- 10-01-1990,123112,10
- 14-02-1991,283901,11
- 10-03-1990,381920,15
- 10-01-1991,302918,22
- 12-02-1990,384902,9

Finding Out Maximum Temperature (Contd.)

ACADGILD

- These lines are presented to the map function as the key-value pairs:

- (0, 10-01-1990,123112,10)
- (22, 14-02-1991,283901,11)
- (44, 10-03-1990,381920,15)
- (66, 10-01-1991,302918,22)
- (88, 12-02-1990,384902,9)

- The keys are the line offsets within the file, which we ignore in our map function.
- The values are the whole line itself.

Finding Out Maximum Temperature (Contd.)

- The map function merely extracts the year and the temperature, and displays them as its output

- (1990, 10)
- (1991, 11)
- (1990, 15)
- (1991, 22)
- (1990, 9)

Finding Out Maximum Temperature (Contd.)

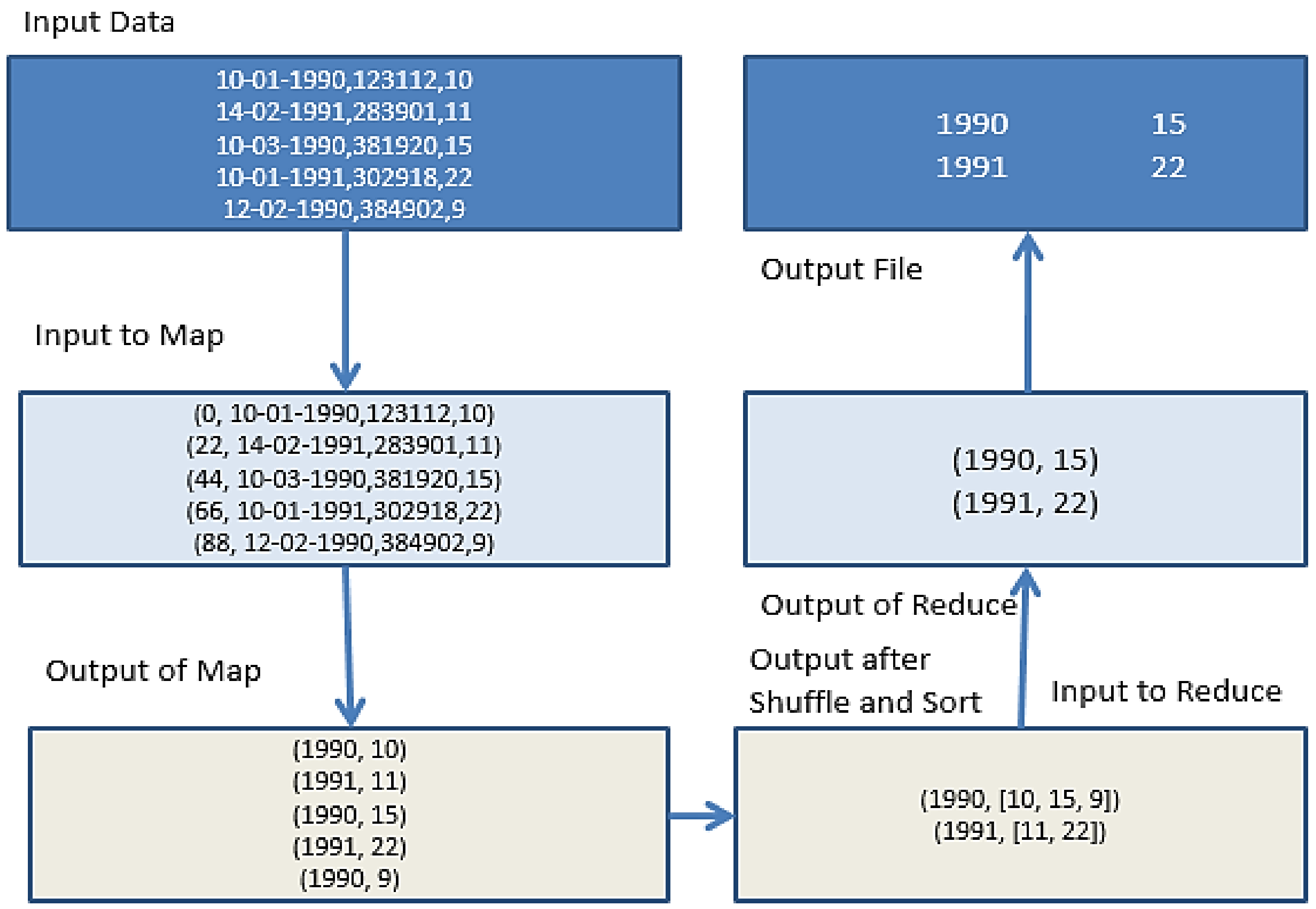
ACADGILD

- The output from the map function is processed by the MapReduce framework before being sent to the reduce function.
- This processing sorts and groups the key-value pairs by key.
- So, continuing the example, our reduce function sees the following input:
- (1990, [10, 15, 9])
- (1991, [11, 22])

Finding Out Maximum Temperature (Contd.)

- All that the reduce function has to do now is iterate through the list and pick up the maximum reading:
- (1990, 15)
- (1991, 22)

Finding Out Maximum Temperature (Contd.)



Pseudo Code

ACADGILD

Map Phase

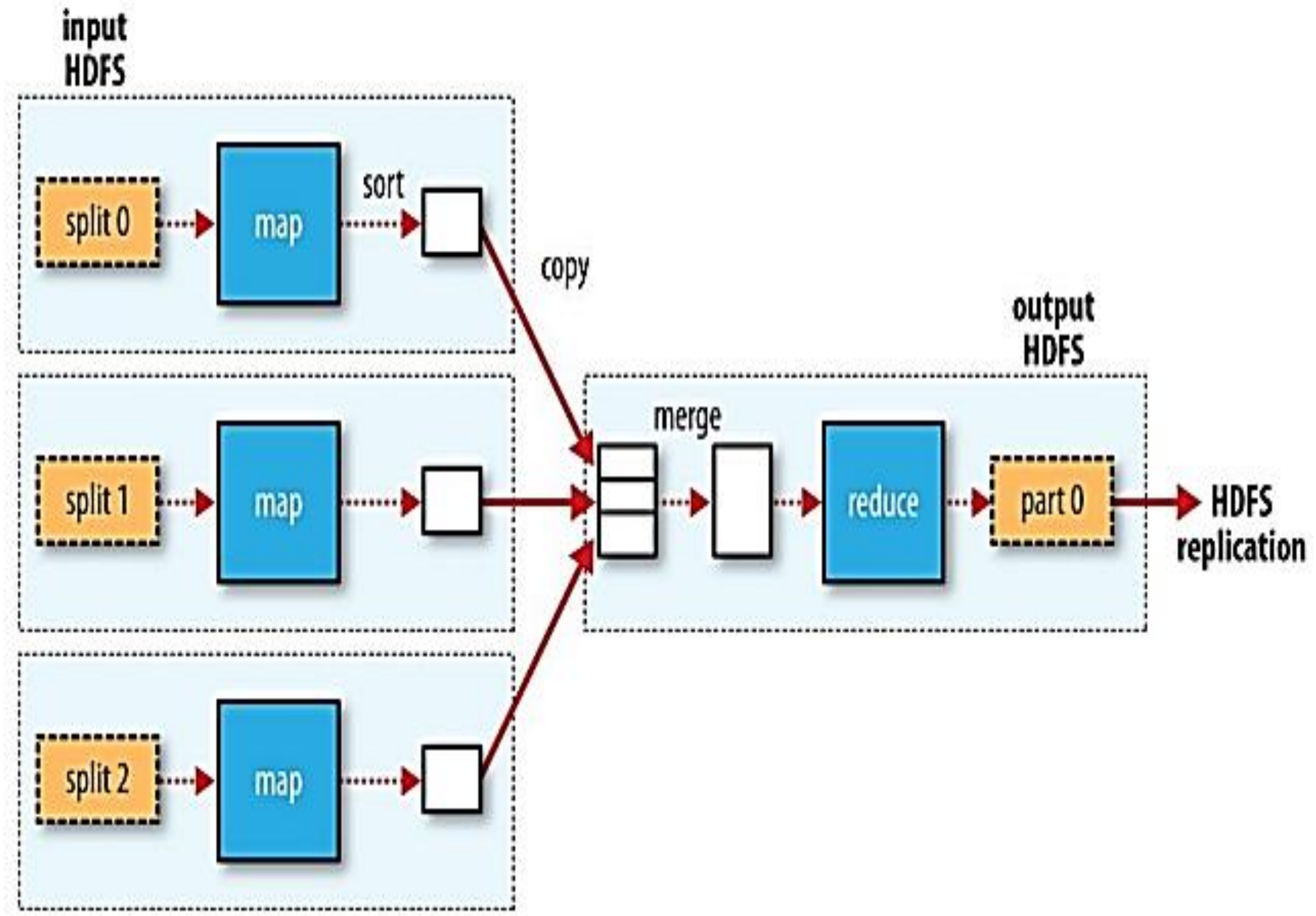
- Read K, V (By default, K is byte offset of the line, V is the whole line)
- Extract year and temperature from V
- Emit (year, temperature)

Reduce Phase

- Read K, list<V> (Here K is the year, as produced from output of map method)
- Iterate through list<V> to get maximum element
- Emit (K, max<V>)

Pseudo Code (Contd.)

ACADGILD



Quiz

Which phase reads input data?

1. Map
2. Reduce
3. Shuffle and Sort
4. None

Quiz (Contd.)

Which phase reads input data?

1. Map
2. Reduce
3. Shuffle and Sort
4. None

Ans: 1. Map

Quiz (Contd.)

ACADGILD

Which phase performs user-defined aggregation?

1. Map
2. Reduce
3. Shuffle and Sort
4. None

Quiz (Contd.)

Which phase performs user-defined aggregation?

1. Map
2. Reduce
3. Shuffle and Sort
4. None

Ans: 2. Reduce

Quiz (Contd.)

Which phase makes sure that one key and all of its associated values go to a single reducer?

1. Map
2. Reduce
3. Shuffle and Sort
4. None

Quiz (Contd.)

Which phase makes sure that one key and all of its associated values go to a single reducer?

1. Map
2. Reduce
3. Shuffle and Sort
4. None

Ans: 3. Shuffle and Sort

Quiz (Contd.)

Identify incorrect statement(s):

1. One reducer may contain multiple keys from map output.
2. One key (from map output) can some values in one reducer and some in the other reducer.
3. We can have zero reducer.
4. One key (from map output) goes to a single reducer along with all of its associated values.

Quiz (Contd.)

Identify incorrect statement(s):

1. One reducer may contain multiple keys from map output.
2. One key (from map output) can some values in one reducer and some in the other reducer.
3. We can have zero reducer.
4. One key (from map output) goes to a single reducer along with all of its associated values.

Ans: 2. One key (from map output) can some values in one reducer and some in the other reducer.

Mapper Class

ACADGILD

- carries out the activity of the mapper

For class, Mapper<K1, V1, K2, V2>

- K1 and V1 are the types of input key and value
- K2 and V2 are the types of output key and value

Methods:

void setup(org.apache.hadoop.mapreduce.Mapper.Context context)

- Called once at the beginning of the task.

void map(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Mapper.Context context)

- Called once for each key/value pair in the input split.

cleanup(org.apache.hadoop.mapreduce.Mapper.Context context)

- Called once at the end of the task.

Reducer Class

ACADGILD

- carries out the activity of the reducer
- For class, Reducer<K1, V1, K2, V2>
  - K1 and V1 are the types of input key and value
  - K2 and V2 are the types of output key and value

Methods:

void setup(org.apache.hadoop.mapreduce.Reducer.Context context)

- Called once at the start of the task.

void reduce(KEYIN key, Iterable<VALUEIN> values,
  org.apache.hadoop.mapreduce.Reducer.Context context)

- This method is called once for each key.

cleanup(org.apache.hadoop.mapreduce.Reducer.Context context)

- Called once at the end of the task.

Driver Code

Driver code controls following parameters:

- Input path
- Output directory
- Input format
- Output format
- Job name
- Output key-value types etc.

Job object is used to define the above parameters.

Driver Code (Contd.)

Job job = Job.getInstance(new Configuration());

- //getInstance is a static method defined in class Job.
- //Job object controls the attributes of the Map-Reduce job.
- //We can also initialize a Job object using other methods as well:
- //Job job = Job.getInstance();
- //Job job = new Job(); //this is deprecated.

- job.setJobName(“<Name of the map-reduce job>");
- job.setInputFormatClass(<class name for Input Format>**);**
- job.setOutputFormatClass(<class name for Output Format>**);**

Driver Code (Contd.)

- job.setMapperClass(<Mapper Class Name>);
- job.setReducerClass(<Reducer Class Name>);

- job.setOutputKeyClass(<class type of key from reducer output>);
- job.setOutputValueClass(<class type of value from reducer output>);

- job.setMapOutputKeyClass(<class type of key from mapper output>);
- job.setMapOutputValueClass (<class type of value from mapper output>);

- FileInputFormat.setInputPath(job, <path object for input file/directory>);
- FileOutputFormat.setOutputPath(job, <path object for output directory>);

- job.waitForCompletion(**true);**

This method will submit the MapReduce job and will wait for its completion

Driver Code (Contd.)

ACADGILD

Configuration

- Provides access to configuration parameters contained in core-site.xml

InputFormat

- Validate the input-specification of the job.
- Split-up the input file(s) into logical InputSplits. InputSplits determine number of mappers.
- Provide the RecordReader implementation to be used to read input records from the logical InputSplit for processing by the Mapper. RecordReader takes care of the remote read.

Context

- It is used to store intermediate key-value pair emitted from mapper or to write final output in key-value pairs from reducers.

Driver Code (Contd.)

ACADGILD

OutputFormat

- Validate the output-specification of the job. For e.g. check that the output directory doesn't already exist.
- Provide the RecordWriter implementation to be used to write out the output files of the job.

Writable and WritableComparable

- Serialization interfaces used by Hadoop.
- WritableComparable can be compared with each other. WritableComparable interface extends both Writable and Comparable.

Demo: Task 1

Finding out Maximum Temperature for every year

Quiz

ACADGILD

- Is it advisable to create new objects in map() or reduce() method?

1. Yes, it speeds up the process.
2. No, we must avoid creating new objects in map() method as far as possible.

Quiz (Contd.)

- Is it advisable to create new objects in map() or reduce() method?

1. Yes, it speeds up the process.
2. No, we must avoid creating new objects in map() method as far as possible.

Ans: No, we must avoid creating new objects in map() method as far as possible.

Exploring Methods of Mapper

ACADGILD

void setup(org.apache.hadoop.mapreduce.Mapper.Context context)

- Called once at the beginning of the task.

void map(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Mapper.Context context)

- Called once for each key/value pair in the input split.

cleanup(org.apache.hadoop.mapreduce.Mapper.Context context)

- Called once at the end of the task.

- Since map() is called for every line of input, it is always good to do these sets of operations in setup() method:

1. Creation of immutable objects
2. Connection to a database (if it is required)

Exploring Methods of Reducer

ACADGILD

void setup(org.apache.hadoop.mapreduce.Reducer.Context context)

- Called once at the start of the task.

void reduce(KEYIN key, Iterable<VALUEIN> values, org.apache.hadoop.mapreduce.Reducer.Context context)

- This method is called once for each key.

cleanup(org.apache.hadoop.mapreduce.Reducer.Context context)

- Called once at the end of the task.

- Since reduce() is called for every input key, it is always good to do these sets of operations in setup() method:

1. Creation of immutable objects
2. Connection to a database (if it is required)

Demo: Task 2

Finding out Maximum Temperature for every month

ACADGILD

THANK YOU

Email us at- support@acadgild.com

LEARN. DO. EARN