CS 378 – Big Data Programming

Lecture 10

Complex “Writable” Types

AVRO

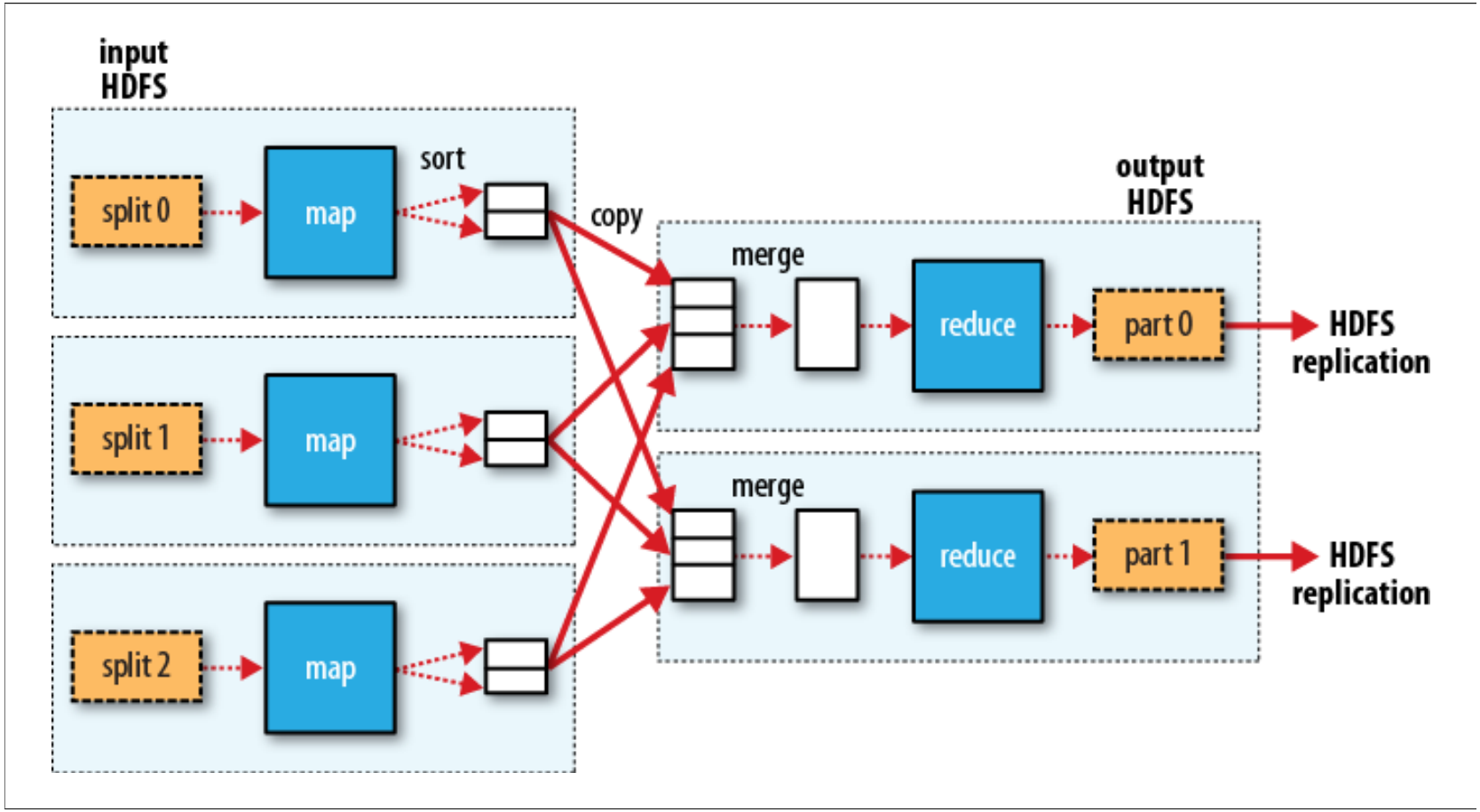
Review

- Assignment 4 – CustomWritable

- We'll look at implementation details of:
  - Mapper
  - Combiner
  - Reducer
  - Supporting classes
- What's being called where?
  - `write(),readFields()`
  - `toString()`

MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



Custom Writables

- Last time we discussed custom `Writables`
- Provided by Hadoop
  - Coded for us in Java
- Google's protocol buffers
- AVRO
  - Language bindings generated by a compiler
  - Uses your definition of the data

Custom Writables

- For our custom `Writable`

- We had to implement `Writable` interface
  - `readFields()`
  - `write()`
- We had to implement `toString()` for text output
- We had to be able to parse in the text representation

- AVRO will implement these things for us

AVRO Example

```
{“namespace”: “com.refactorlabs.cs378.assign5”,
 “type”: “record”,
 “name”: “WordCountData”,
 “fields”: [
     {“name”: “word_count”, “type”: “long”} ]
}
```

- How does this get transformed to Java code?
  - Add the schema file to your project (*filename*.avsc)
  - Run maven to force AVRO compile
    - Or run maven target in your IDE

AVRO Generated Code

- Accessors for the internal data
  - Has methods
    - `hasWordCount()`
    - …
  - Get methods
    - `getWordCount()`
    - …
- Builder class for constructing instances
  - Above methods
  - Plus set and clear methods

AVRO – Builder Classes

- Why construct instances using the Builder class?
- You AVRO schema contains constraints
  - Value types: enforced by accessors
  - Required vs. optional values (union): checked by build
- Incremental construction
  - For arrays and maps, data can be added incrementally

AVRO I/O

- Text output
  - AVRO text representation is JSON
- Avro container files
  - Binary representation that we can read as input
- The particular format is determined by
  - The types of objects we output
  - The file output format

Assignment 5

- Bootstrap script (control classpath order)
  - We want a specific version of AVRO
  - This script will place your JAR file at the start of the classpath
  - Add this as a bootstrap "custom action" in your cluster
- pom.xml provided
  - Use this one, as AVRO with Hadoop is version sensitive
  - Select AMI version 3.10.0 when defining your cluster
- Example use of AVRO: WordCountA.java
- All files on Canvas / Files / Assignment 5

Assignment 5

- Implement an AVRO object for WordStatistics data
  - Call it `WordStatisticsData`
  - Mapper output:
    - `Text, AvroValue<WordStatisticsData>`
  - Reducer output:
    - `Text, AvroValue<WordStatisticsData>`

- See code in WordCountA
  - Output file format: `TextOutputFormat`
  - Set JAR to beginning of classpath
    - conf.setBoolean(MRJobConfig.*MAPREDUCE_JOB_USER_CLASSPATH_FIRST*, **true**);
  - Calls using `AvroJob`