CS 378 – Big Data Programming

Lecture 11

AVRO Formats

Review

- Assignment 5 – AVRO Objects
- Questions/Issues?
  - `datum()`
- Using the latest pom.xml

AVRO File Formats

- We're going to look at the combination of different key/value types and file formats
  - What does the output look like (text versus binary)

- Key and value type declaration for mapper/reducer
  - `AvroKey<>`, `AvroValue<>`
- Specifying the key/value type for the job
  - `AvroJob.setMapOutputKeySchema()`
  - `AvroJob.setMapOutputValuyeSchema()`
  - `AvroJob.setOutputKeySchema()`
  - `AvroJob.setOutputValueSchema()`

AVRO File Formats

- `TextOutputFormat`
  - How are various key and value types handled?
  - Recall that `TextOutputFormat` will cause `toString()` to be called
- `AvroKey<CharSequence>`
  - Acts like `Text`, so it just returns its string value
- `AvroValue<WordStatisticsData>`
  - Returns the value created by the `toString()` method

AVRO File Formats

- `TextOutputFormat`
- `Text, AvroValue<WordStatisticsData>>`
  - The key is output as a string
  - tab character
  - `toString()` called on `WordStatisticsData`
- Result (see WordCountA example code)
  - `Brown { "document_count": … }`

AVRO File Formats

- `TextOutputFormat`
- `AvroKey<CharSequence>, AvroValue<WordStatisticsData>>`
  - The key is output as a string
  - tab character
  - `toString()` called on `WordStatisticsData`
- Result (see WordCountB example code)
  - `Brown { "document_count": … }`

AVRO File Formats

- `AvroKeyValueOutputFormat`
  - Creates a generic AVRO record with a “key” field and a “value” field
    - Types: `AvroKey<Pair< K, V >>, NullWritable`
    - `K: CharSequence`
    - `V: WordStatisticsData`
  - Avro container file (binary)
  - Can be read in using: `AvroKeyValueInputFormat`
  - See WordCountC example code

AVRO File Formats

- `TextOutputFormat`

- `AvroKey<Pair<CharSequence, WordStatisticsData>>, NullWritable`
  - Used as the key to the `write()` method, with value `NullWritable`
  - Generates a string representation in the `toString()` method of `Pair`
  - In this form: `{"key": `*theKey*`, "value": `*theValue*` }`
  - *theKey* comes from `CharSequence`, so just a string
  - *theValue* comes from `WordStatisticsData`, so an AVRO text representation is generated (calls `toString()`)
  - `{ "document_count": ….. }`
- See WordCountD example code

AVRO File Formats

- `AvroKeyOutputFormat<T>`
  - Extends
  - `AvroOutputFormatBase(AvroKey<T>, NullWritable>)`
  - Only the key is output, value is ignored
  - AVRO container file (binary format)

  - Can be read in using: `AvroKeyInputFormat`

  - See WordCountE example code

AVRO File Formats

- `AvroSequenceFileOutputFormat`
  - Sequence file output format that can handle `AvroKey` and `AvroValue` in addition to `Writable`
  - Binary format
  - Can be read with: `AvroSequenceFileInputFormat`

AVRO Input File Formats

- `AvroKeyValueInputFormat`
  - Reads generic Avro records with a "key" field and a "value" field
  - Reads an AVRO container file (binary format)
  - Data written with: `AvroKeyValueOutputFormat`
- `AvroKeyInputFormat`
  - Extends: `FileInputFormat(AvroKey<T>, NullWritable>)`
  - Only the key is read, value is ignored
  - Reads a AVRO container file (binary format)
  - Data written with: `AvroKeyOutputFormat`

AVRO File Formats

- `AvroSequenceFileInputFormat`
  - Input format that can read sequence files that support Avro types
  - Data written with: `AvroSequenceFileOutputFormat`

Unit Tests with AVRO

- MRUnit understands serialization in Hadoop …
  - Writable interface (`readFields()` and `write()`)
- We need tell MRUnit to use AVRO serialization
- And we need to construct our expected outputs
  - For `map()` and `reduce()` expected output
- And we need to construct our inputs
  - For `reduce()` input

Design Pattern

- Structured to hierarchical design pattern

- Data sources linked by some foreign key
- Data is structured and row based
  - For example, from databases
- Data is semi-structured and event based
  - Web logs

Design Pattern

- Structured to hierarchical design pattern
- `MultipleInputs`
  - Able to accept data inputs from different formats
  - Mappers load and parse the input into a cohesive format
  - Prepared for work in the reducer
  - Map output key will be the unifying element of the hierarchical record
- Combiners don't help, as they don't "reduce" the data (make it smaller)

Design Pattern

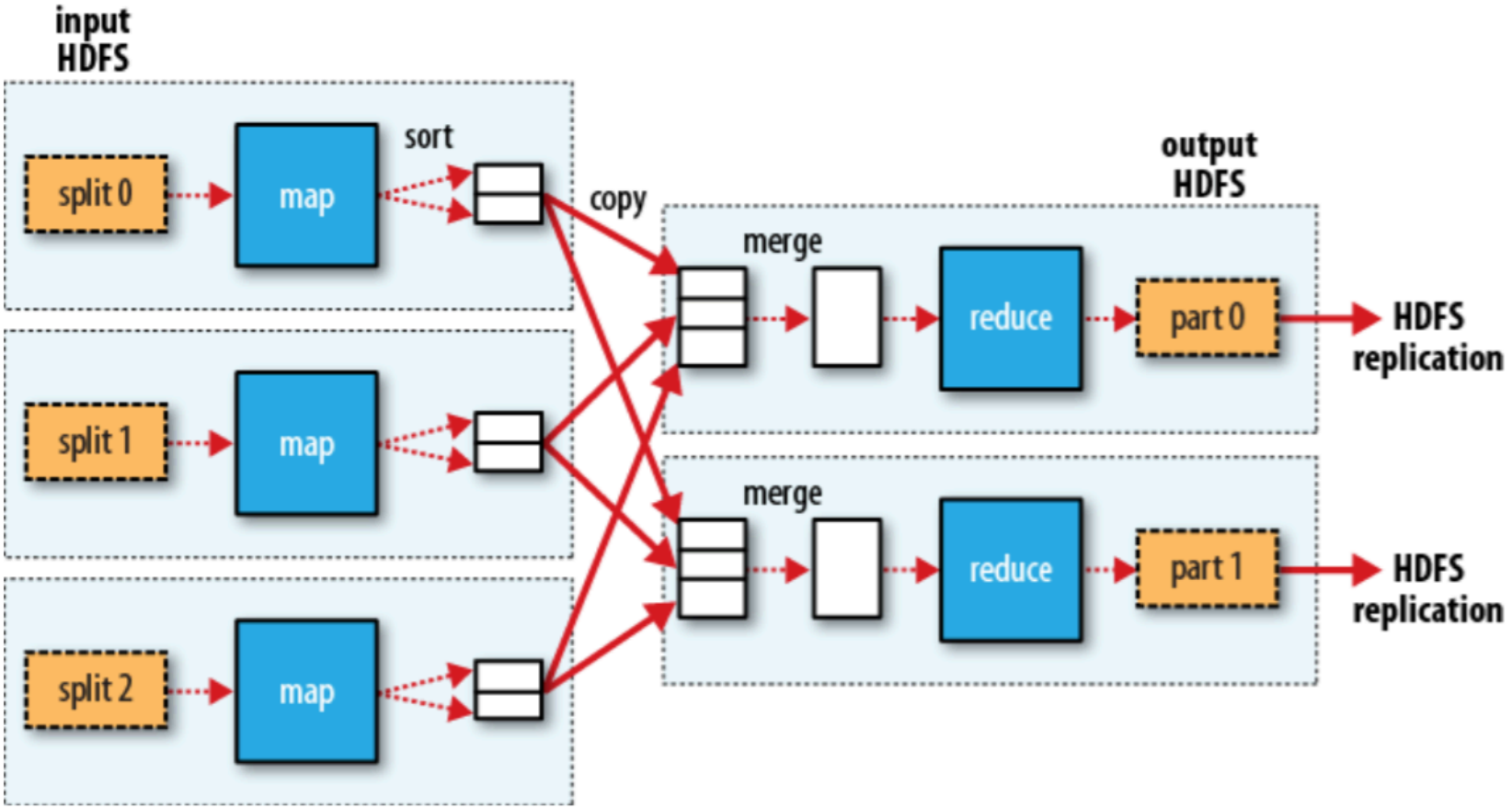
- Structured to hierarchical design pattern

- Reducer takes all the data associated with a key
- Builds the structure to be output

- Example:
  - User session contains info about the user (IP, browser, …)
  - An array of actions (page views, clicks, …)

MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



Sessionizing Web Logs

- Create user sessions from individual web log entries

- Represents all the actions by a user
- Allows later analysis to “replay” the user actions

- Collect measures and metrics about user behavior
  - Pages viewed, time on page, clicks
  - Path through the site, entry to the site (from a search engine?)