CS 378 – Big Data Programming

Lecture 7

File Formats

Review

- Assignment 3 – InvertedIndex

- Questions/issues?

File Formats - Review

- What does **TextInputFormat** do?
  - Via its **RecordReader** implementer

- Identifies the next line of input
  - Text through the next newline
- Creates the **Text** object with this content
- Calculates the position of this line in the input split
- Creates the **LongWritable** with this number
- Reports progress via **getProgress()**

File Formats - Review

- What does **TextOutputFormat** do?
    - Via its **RecordWriter** implementer

- Calls **toString()** on the key, writes this string
- Writes a tab character
- Calls **toString()** on the value, writes this string

File Formats

- Suppose we wanted to use the output of WordCount as input to another map-reduce job
  - Maybe we collected word counts for each day's emails
  - Now we want to sum up stats from multiple days

- One approach: Use `TextInputFormat`
  - Map input is `LongWritable`, `Text`
  - We'd have to parse the value in the Text object to separate the key and value (separated by a tab)
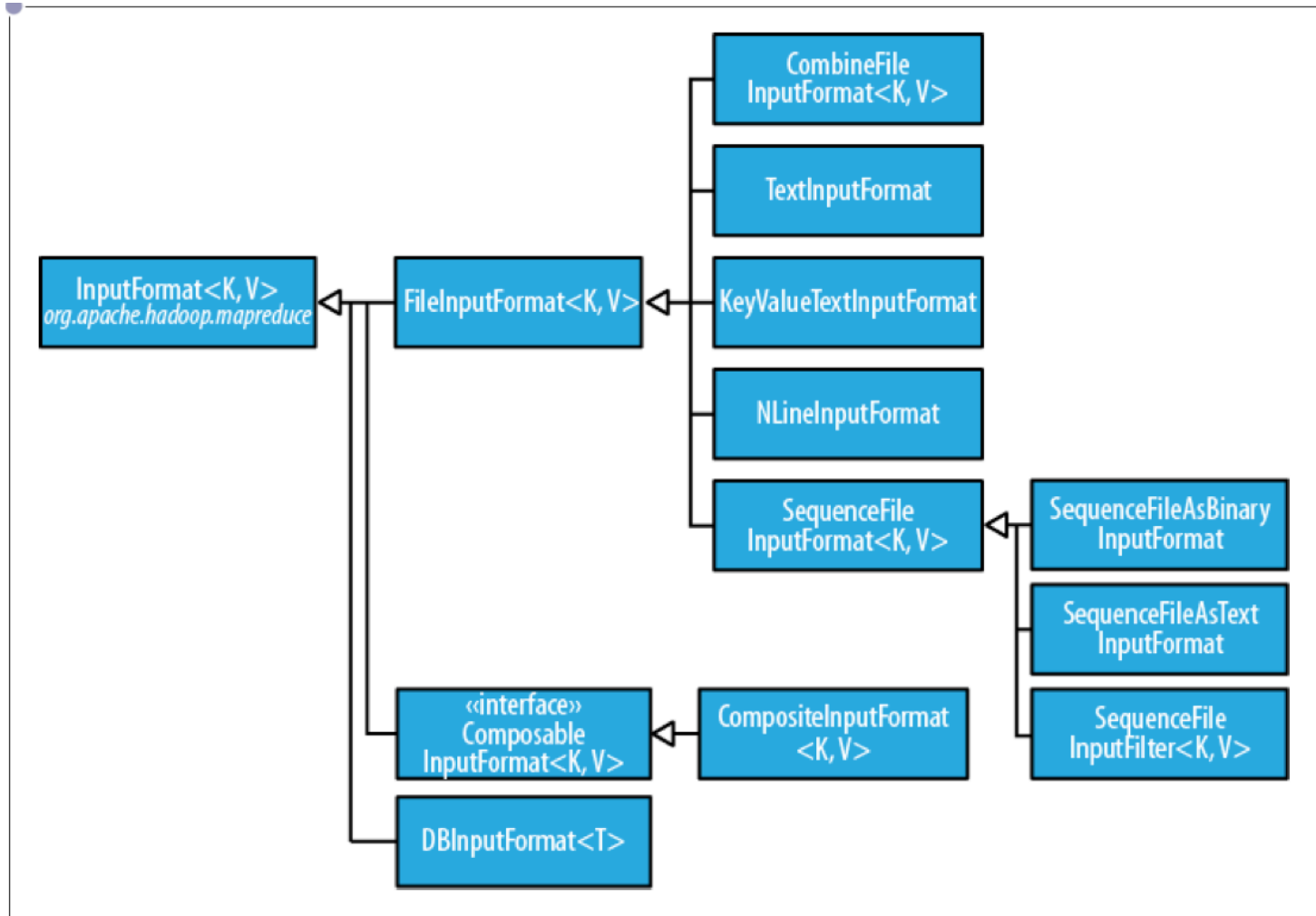
File Formats

- Another approach: implement a custom file format

- What do we need to do?

- In our custom input file format class …
  - Define a `RecordReader` interface implementer to:
  - Grab one line of input from the input split
  - Find the key/value separator
  - Return the key (the word) as a `Text` object
  - Return the value (the count) as a `LongWritable` object

- Seems like a convenient class to have around

File Formats

- Hadoop provides (almost) this class for us:
- **`KeyValueTextInputFormat`**
  - You can set the separator character (by default, tab)
  - Key and value types are **`Text`**

- Other file formats and readers provided by Hadoop
  - Reading from a database
  - Each mapper receives exactly N lines
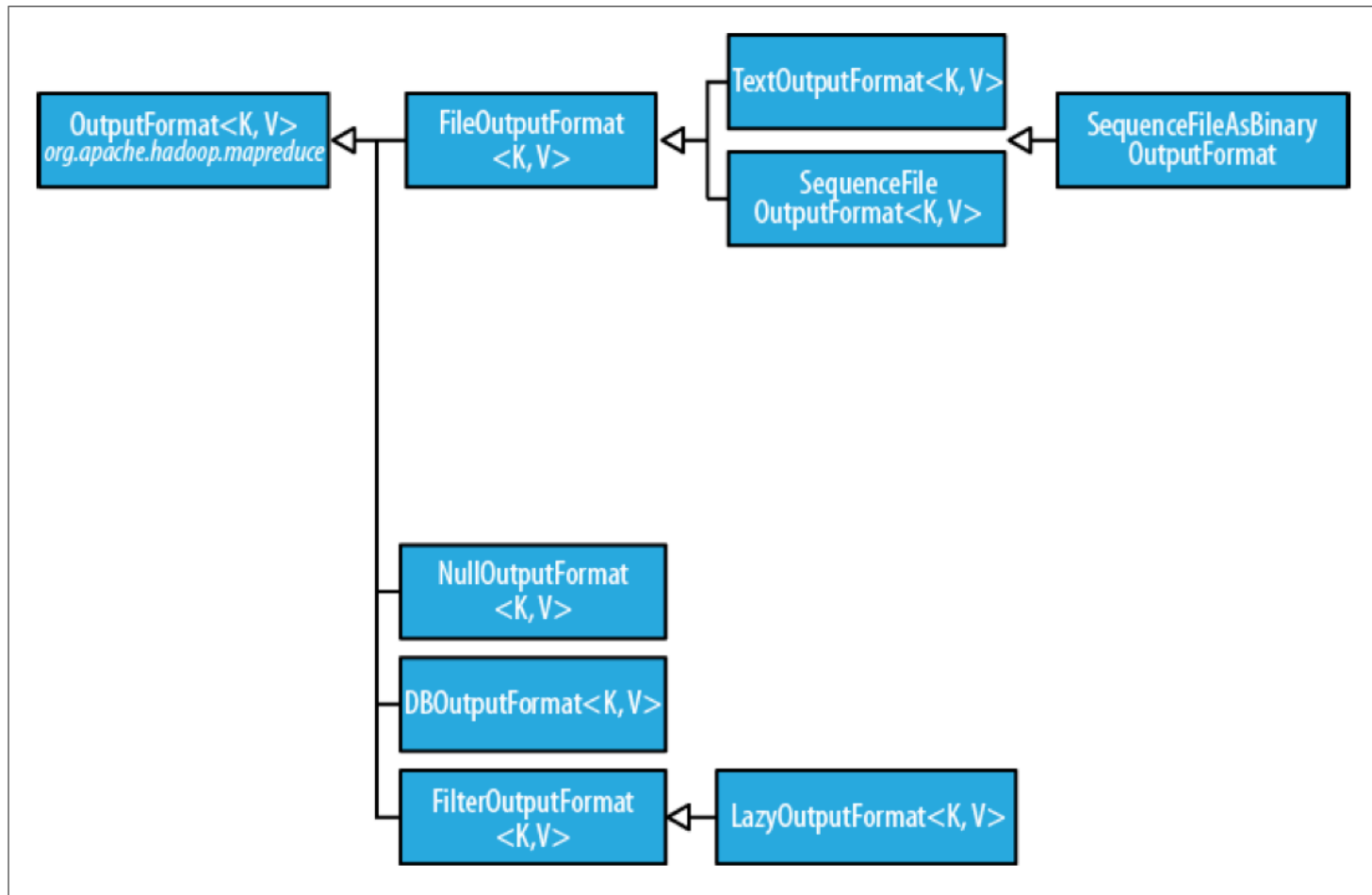 - XML stream processing
  - Sequence files (binary)

File Formats

Figure 8-2, Hadoop: The Definitive Guide 4th Edition

File Formats

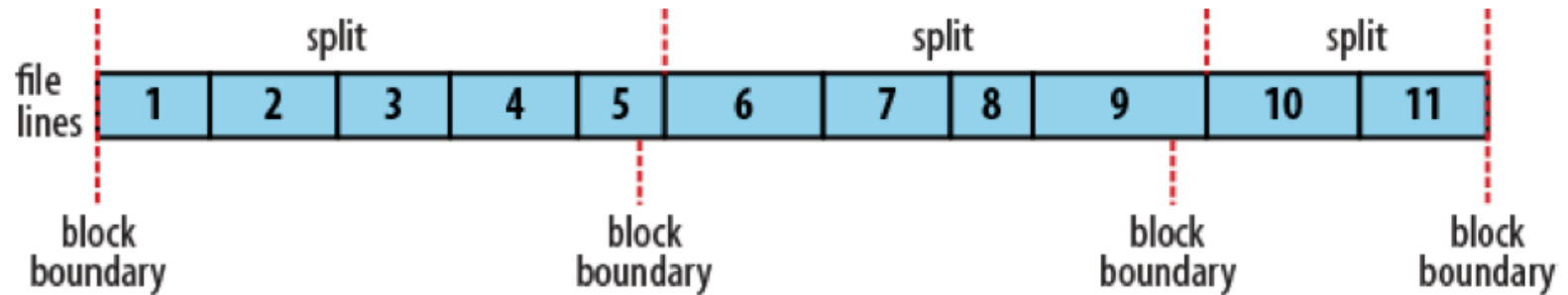Figure 8-4, Hadoop: The Definitive Guide 4th Edition

Input Format

- What does an `InputFormat` do?
  - Validate input configuration (is the data there?)
  - Split input blocks/files into logical chunks
    - Logical chunks are of type `InputSplit`
    - Each is assigned to a mapper
  - Create the `RecordReader` that generates key/value pairs from the `InputSplit`
- `RecordReader` also has to fix up records that span splits
- We've used `TextInputFormat`

Processing Splits

Figure 8-3, Hadoop: The Definitive Guide 4th Edition

Input Format

- **TextInputFormat** uses **LineRecordReader**
  - Reads an input split to get the next input line (`'\n'`)
  - At the beginning of an input split, find first newline
  - Reads past the split boundary until it finds an end-of-line
  - Key returned: position in the input split
  - Value: the input line

- **KeyValueTextInputFormat**
  - How is it different from **TextInputFormat**?

Generating Random Data

- Random data can be used for testing when:
  - Real data does not yet exist, and/or
  - You want to control the "shape" of the data

- We can create a custom input format to generate random data
  - No actual input is read
  - The `RecordReader` will generate random values as "input"

InputFormat Interface

- Two methods to implement:

- **getRecordReader()**
- **getSplits()**

- **InputSplit** methods:

- **getLength()**
- **getLocations()**

RecordReader Interface

- Methods to implement:

- **initalize()**
- **getCurrentKey(), getCurrentValue()**
- **nextkeyValue()**
- **getProgress()**
- **close()**