CS 378 – Big Data Programming

Lecture 5

Summarization Patterns

Review

- Assignment 2 - WordStatistics

- We'll look at implementation details of:
  - Mapper
  - Combiner
  - Reducer
  - Supporting classes

- Other questions/issues?

File Formats

- In assignments 1 and 2, we used
  - **`TextInputFormat`**
  - **`TextOutputFormat`**
- Key value pairs:
  - Input: **`LongWritable`**/**`Text`**
  - Output: **`Text`**/**`DoubleArrayWritable`**
- The input file is just lines of text
  - How does the **`LongWritable`** get generated?

File Formats

- Input formats provide an instance that extends Hadoop class **`RecordReader`**
- **`RecordReader`** methods
  - **`initialize(InputSplit, TaskAttemptContext)`**
  - **`nextKeyValue()`**
  - **`getCurrentKey()`**
  - **`getCurrentValue()`**
  - **`getProgress()`**
  - **`close()`**

File Formats

- 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

- Key value pairs:
  - Output: **`Text`**/**`DoubleArrayWritable`**
- The output file is just lines of text
  - How does this text get generated?
- Similar to input formats, output is controlled by instances that extend **`RecordWriter`**
- **`RecordWriter`** methods
  - **`write(key, value)`**
  - **`close()`**

File Formats

- 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

- How do we control the format of our results for WordStatistics?

Summarization

- Another summarization of interest
  - Inverted index

- Suppose we are interested indexing the emails by individual email addresses
  - For a given email address, which emails contain it
  - Indices are built for search engines to quickly identify which documents are relevant
  - Interesting for anyone investigating an email corpus

Inverted Index

- For an inverted index that represents which emails an individual email address appears in:
- What is the final output?
  - Key: email address
  - Value: list of emails the address appears in
- Given our data set of emails
  - What should the mapper do?
  - What should the reducer do?
  - Can we use a combiner?

Inverted Index

- Some additional functionality
- Can we partition the references into:
  - Emails where the address is in the From field
  - Emails where the address is in the To: field
  - Emails where the address is in the Cc: or Bcc: field
- How would we do this?

Inverted Index

- Email example
  - Message-ID: <23426663.1075857497542.JavaMail.evans@thyme> Date: Mon, 23 Apr 2001 03:05:00 -0700 (PDT) From: jane.tholt@enron.com To: elizabeth.hernandez@enron.com Subject: Re: Mar 2001 Price Mime-Version: 1.0 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit X-From: Jane M TholtX-To: Elizabeth L Hernandez X-cc: X-bcc: X-Folder: \Jane_Tholt_Jun2001\Notes Folders\Sent X-Origin: Tholt-J X-FileName: jtholt.nsf CHANGED PRICE ON 3/8 TO 16.00

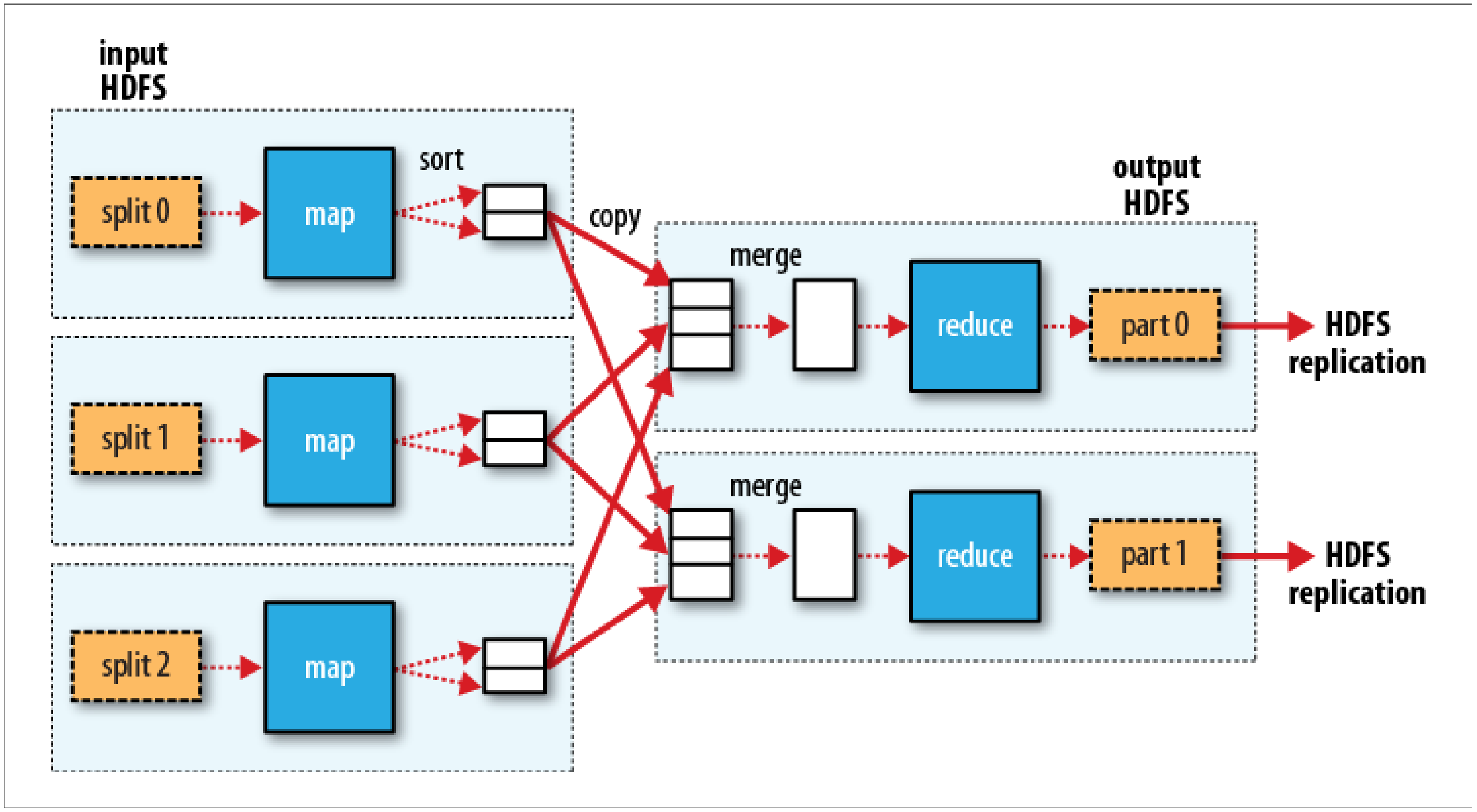
- What should the keys be?
- What should the value(s) be?

Inverted Index

- Email parsing
  - Message-ID:
  - Date:
  - From:
  - To:
  - Subject:
  - Cc: [optional]
  - Mime-Version:
  - Content-Type:
  - Content-Transfer-Encoding
  - Bcc: [optional]
  - X-From:
  - …
- Address fields: From: To: Cc: Bcc:

MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide


input HDFS
split 0
split 1
split 2
map
map
map
sort
copy
merge
merge
reduce
reduce
output HDFS
part 0
part 1
HDFS replication
HDFS replication