CS 378 – Big Data Programming

Lecture 2

Map-Reduce

Logistics - Update

- Office Hours – GDC 4.310
  - Tues. 11:00 – 12:00 AM, Thurs 8:15 – 9:15 AM
  - By appointment
  - Email: dfranke@cs.utexas.edu
  - Web page: cs.utexas.edu/~dfranke/courses/cs378-BDP.htm

MapReduce

- Large data sets are not new
- What characterizes a problem suitable for MR?
  - Most or all of the data is processed
    - But viewed in small increments
    - For the most part, map and reduce tasks are stateless
  - Write once, read multiple times
    - Data Warehouse has this intended usage (write once)
  - Unstructured data vs. structured/normalized
- Data pipelines are common
  - Chain of MR jobs, with intermediate results

MapReduce

Table 1-1, Hadoop – The Definitive Guide

	Traditional RDBMS	MapReduce
Data Size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many
Structure	Static schema	Dynamic schema
Integrity	High	Low
Scaling	Nonlinear	Linear

MapReduce

- Tom White, in *Hadoop: The Definitive Guide*

- "*MapReduce works well on unstructured or semistructured data because it is designed to interpret the data at processing time. In other words, the input keys and values for MapReduce are not intrinsic properties of the data, but they are chosen by the persona analyzing the data*."

MapReduce vs. Relational

- RDBMS normalization
  - Goal is to remove redundancy and retain/insure integrity
- MapReduce wants reads to be local
  - Send the code to the data, as it much smaller (Jim Gray)
  - Normalization makes read non-local
- Map function state
  - One input record
- Reduce function state
  - All records (map outputs) with same key

MapReduce

- When writing a MapReduce program …
  - You don’t know the size of the data
  - You don’t know the extent of the parallelism
- MapReduce tries to collocate the data with the compute node
  - Parallelize the I/O
  - Make the I/O local (versus across network)

MapReduce

- This all sounds great. What are the issues?
  - Coordinating the distributed computation
  - Handling partial failures
  - Combining the results of distributed computation

- MapReduce is a programming model that abstracts
  - Disk read and write
  - Parallelization
  - Combining data (keys and values)

Map Function

- Map input is a stream of key/value pairs
  - Web logs: Server name (key), log entry (value)
  - Sensor reading: sensor ID (key), sensed values (value)
  - Record number (key), record (value)
  - Document ID (key), contents (value)
- Map function processes each input pair in turn
- For each input pair, the map function can (but isn't required) to emit one or more key/value pairs
  - Key value pair(s) derived from the input key/value pair
  - Does not need to be the same key or value data type

WordCount Example

- For an input text file of arbitrary size, or
- Multiple test files of arbitrary size, or
- An arbitrary number of documents

- Count the number occurrences of all the words that appear in the input.
- Output:
  – Word1, count
  – Word2, count
  – …

WordCount Example - Map

- Map input is a stream of key/value pairs
  - File position in bytes (key), line in the file (value)
- Map function processes each input pair in turn
  - Extract each word from the line of text
  - Emits a key/value pair for each word: *<the-word, 1>*
- For each input pair, the map function emits multiple key/value pairs
  - Key is a text string (the word), value is a number

Reduce Function

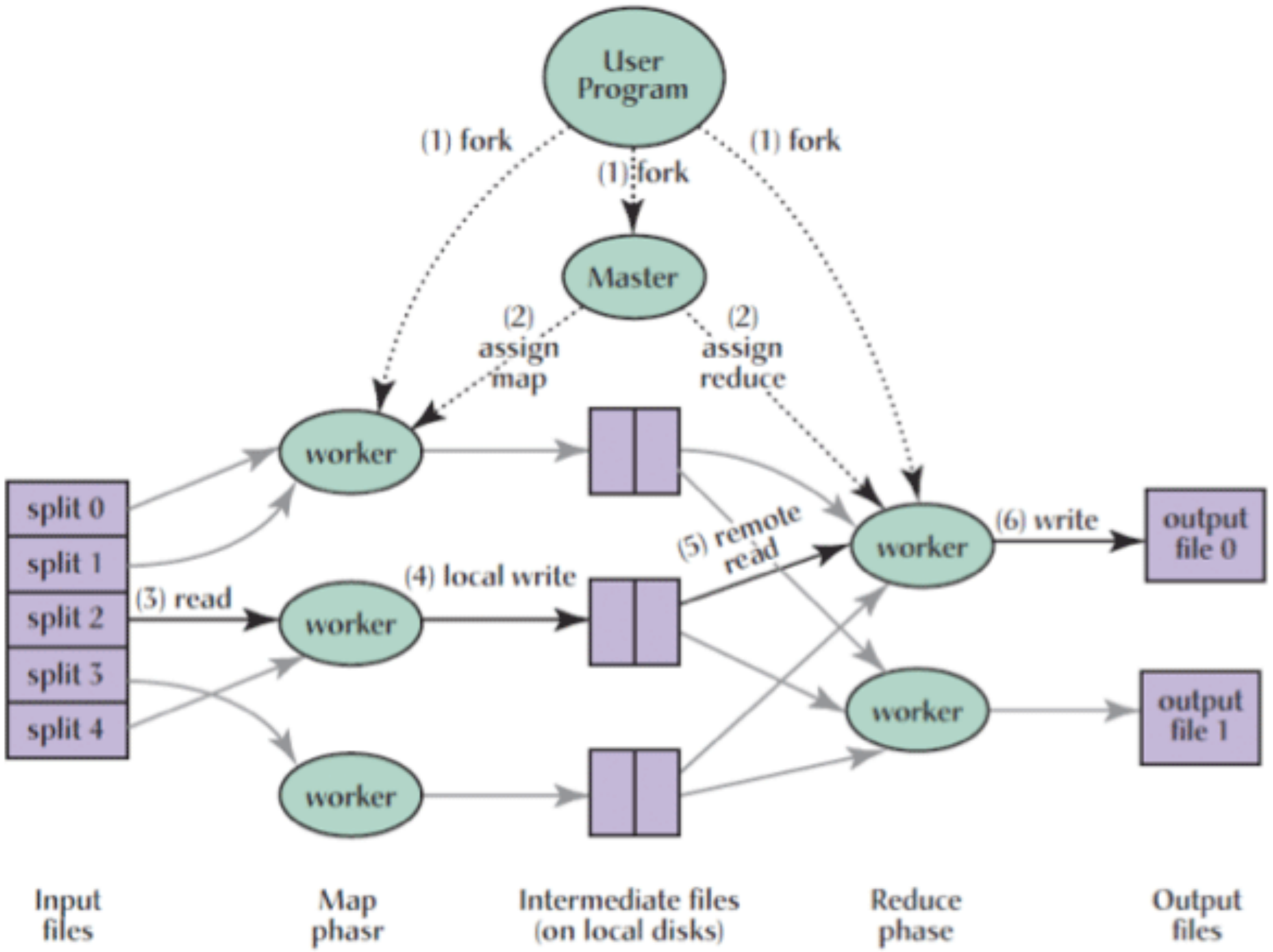
- Reduce input is a stream of key/value-list pairs
  - These are the key value pairs emitted by the map function
- Reduce function processes each input pair in turn
- For each input pair, the reduce function can (but isn't required) to emit a key/value pair
  - Key value pair derived from the input key/value-list pair
  - Does not need to be the same key or value data type

WordCount Example - Reduce

- Reduce input is a stream of key/value-list pairs
  - These are the key value pairs emitted by the map function
  - Key is a text string (the word), value is a list of some number of the value "1"
- Reduce function processes each input pair in turn
  - Sums the values in the value-list
- For each input pair, the reduce function emits a key/value pair
  - Key is a text string (the word), value is the count for that word

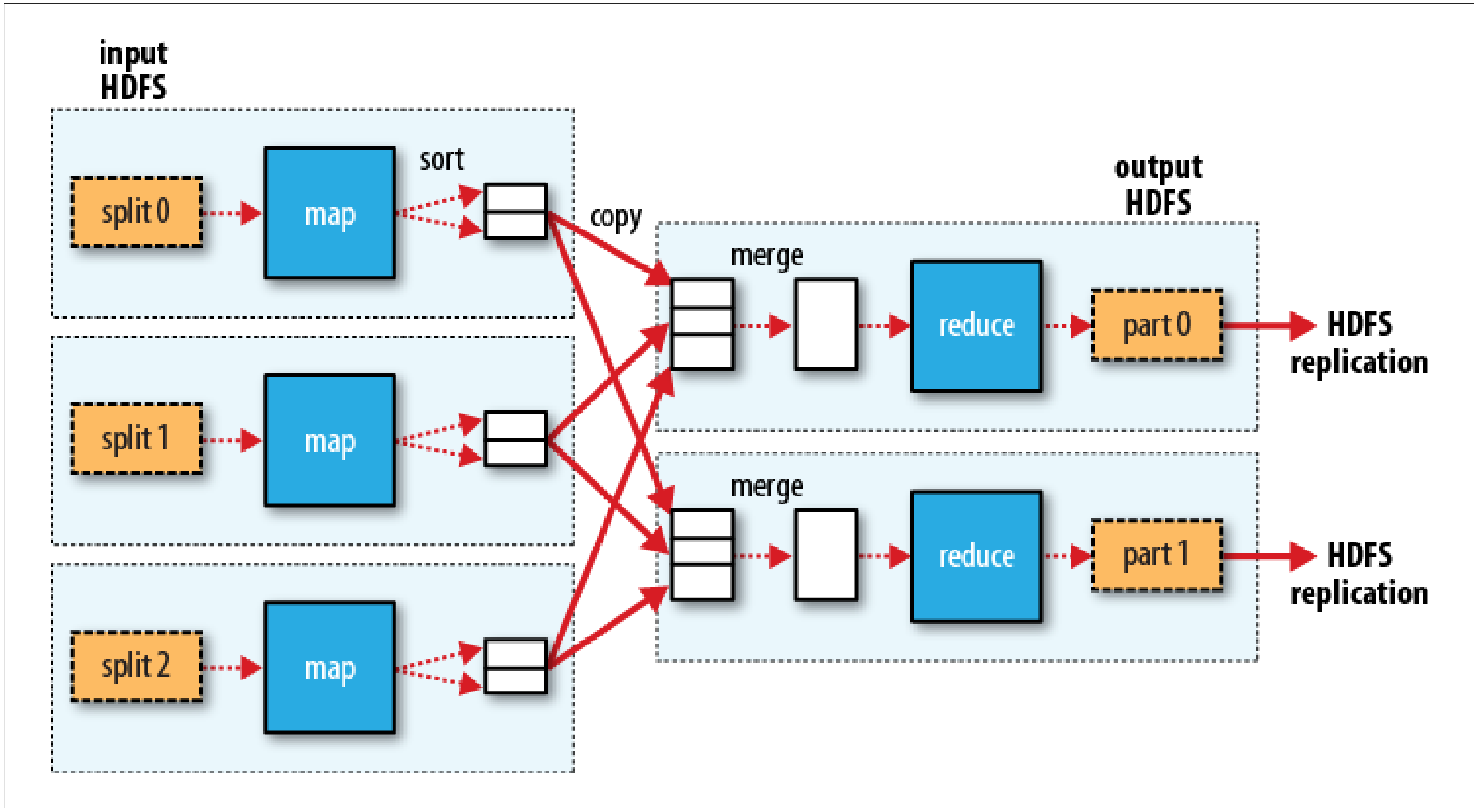
MapReduce

(from cubrid.org)



MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



Java and Maven Review

- Directory structure expected by maven (supported in IDEs):
  - Project directory (example name: bdp)
  - Source code directory: bdp/src/main/java
  - The Java package structure appears in the “java” directory
  - Ex: bdp/src/main/java/com/refactorlabs/cs378
  - A class defined in the com.refactorlabs.cs378 package is placed here
  - Ex: bdp/src/main/java/com/refactorlabs/cs378/WordCount.java

- Easy setup - Create you project directory
  - Place pom.xml in this directory
  - Place WordCount.java as shown above
  - Import the maven pom.xml into your IDE.