# CS 378 – Big Data Programming

Lecture 3

Anatomy of a Hadoop

Map-Reduce Program

# Assignment 1 Update

- JAR file build issues?
- What's in pom.xml

- Running the example on AWS
  - The cluster and job monitor page
  - Log files: controller, syslog

- Questions?

# Map-Reduce Code

- main() method
- Job object - Collects up all the specs for the job
  - Where is the JAR file to distribute?
  - Type of the output pair
  - Mapper and Reducer classes
  - Input and output file formats
  - Input file(s), output directory
- Configuration object – forwarded to map(), reduce()
  - Job level parameters communicated via this object

# Map-Reduce Code

- MapClass
  - Extends Mapper, declaring the input and output pair types for the map() method

- map() method
  - Arguments: input pair, and the Context
  - Output done via the context object

# Map-Reduce Code

- ReduceClass
  - Extends Reducer, declaring the input and output pair types for the reduce() method

- reduce() method
  - Arguments: input pair, and the Context
  - Output done via the context object

# Map-Reduce Code

- map() and reduce() input pair and output pair types
- Derived from Writable
  - readFields(DataInput in)
  - write(DataOutput out)

- Text, IntWritable, LongWritable all implement Writable
  - As do many other types, some of which we will use

- Possible to design your own class that implements Writable

# Map-Reduce Code

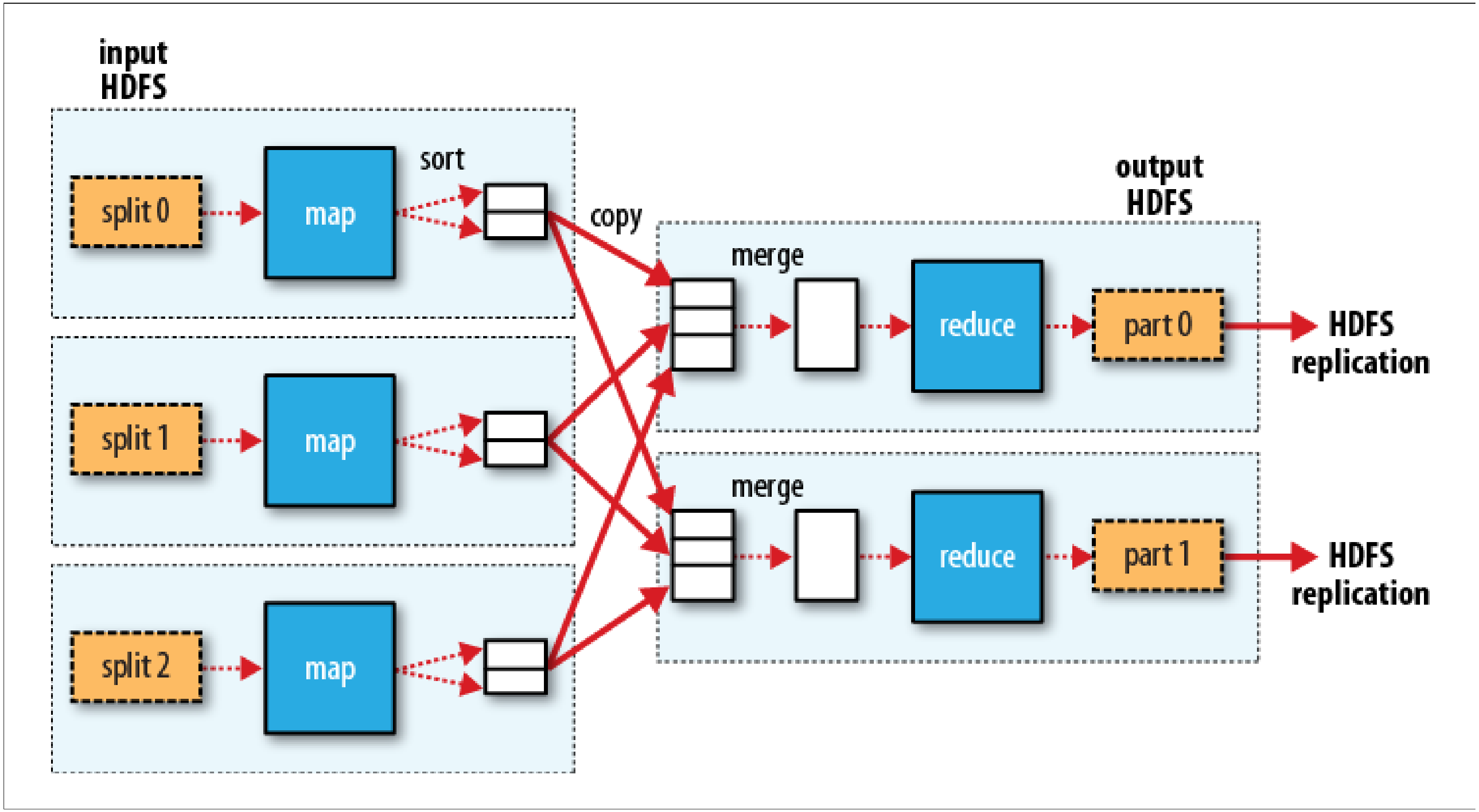
- Combiner – combines multiple outputs from a Mapper before shuffle

- Input and output pair types must be the same.
  - Why?

- When can a combiner be used?
  - Map output can be processed ("combined") even through we do not see all values associated with the key
  - Combiner output can be interpreted by reducer
  - Word count, and many other counting applications can use a combiner.

# MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide



# Map-Reduce Code

- For WordCount, suppose we used a hash table to collect word counts over multiple input records.
- Why wouldn't this work?