CS 378 – Big Data Programming

Lecture 5

Summarization Patterns

Review

- Assignment 2 – Questions?

- Interested in using Google collection classes?
  - Now called Guava
  - pom.xml for guava dependency available on the assignment 2 description on Canvas

- How to find documentation for Java classes

Multiple Output Values

From Lecture 4

- We can add some methods to the **`LongArrayWritable`** class to make it easier to use.

```
public long[] getValueArray() {
    Writable[] wValues = get();
    long[] values = new long[wValues.length];
    for (int i = 0; i < values.length; i++) {
        values[i] = ((LongWritable)(wValues[i])).get();
    }
    return values;
}
```

Summarization

- Other summarizations of interest
  - Min, max, mean

- Suppose we are interested in these metrics for email length (number of characters)
  - If the length of emails is normally distributed, then median will be very near the mean
  - If the distribution of email lengths is skewed, the mean and median will be very different

Summarization

- Min and max are straightforward
- For each email, output two values
  - Min length (the length of this email)
  - Max length (the length of this email)
  - Key?

- Combiner will get a list of value pairs
  - Select the min, max, output that value pair
  - Key?
- Reducer does the same

Summarization

- Median
  - Get all the values, sort them, then find the middle
- Since our computation is distributed, we done see all values?
- Send them all to one reducer?
  - Not utilizing map-reduce
  - Data sizes likely too large to keep in memory

Summarization

- Median – one approach is to keep the unique email lengths, and the frequency of each length

- Mapper output:
  - Value is one pair on numbers: < email length, 1 >

- Combiner gets a list of these pairs, updates the count for recurring lengths
- Reducer does the same, then identifies the median

Summarization

- Median
  - Hadoop provides the **`SortedMapWritable`** class
  - Associates a frequency count with a length
  - Keeps the lengths in sorted order

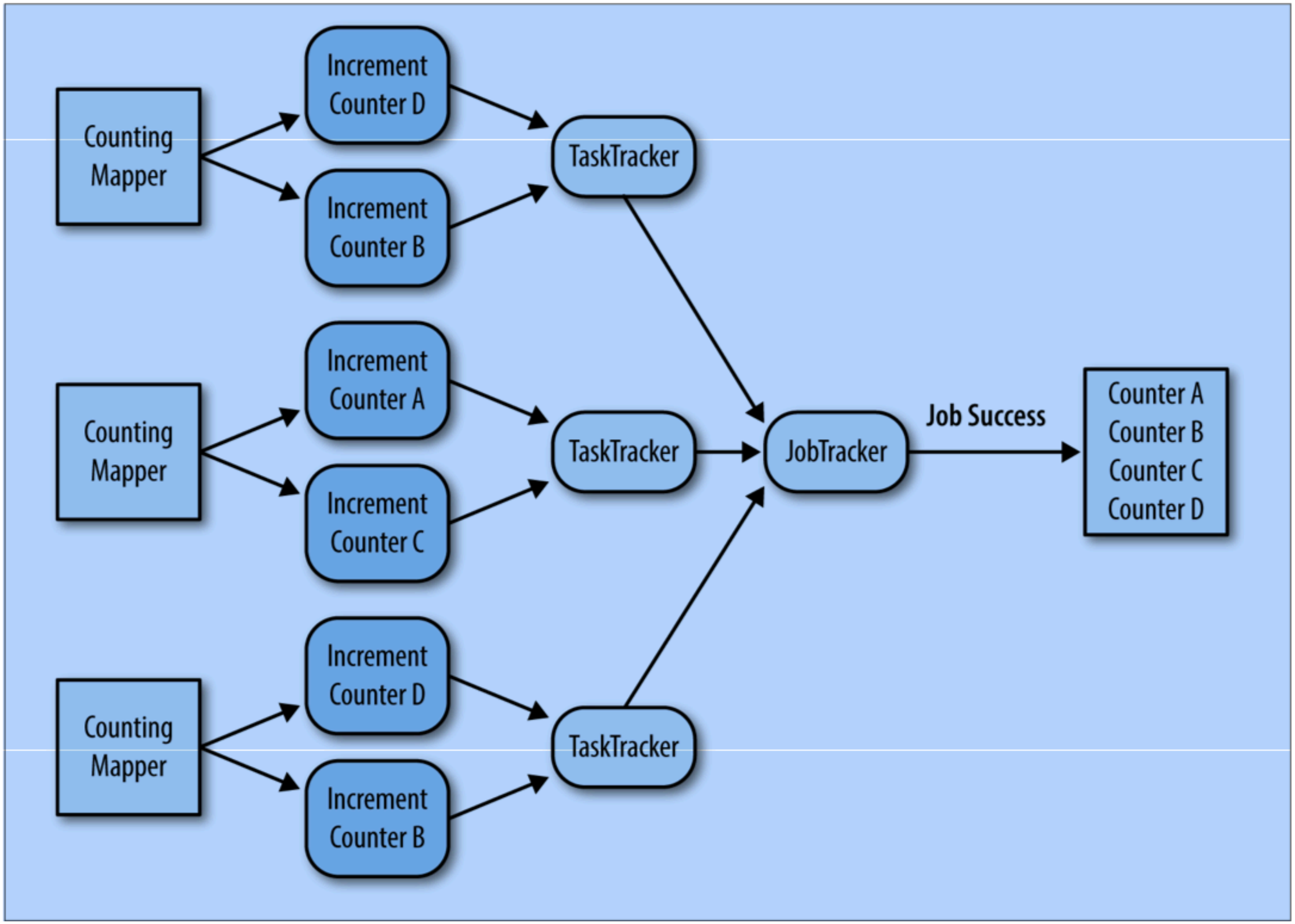
- See the example in Ch. 2 of *Map-Reduce Design Patterns*

Counters

- Hadoop Map-Reduce infrastructure provides counters
  - Accessed by group name, counter name
  - Cannot have a large number of counters
    - Can’t use this to do word count
  - A few tens of counters can be used
- Counters are stored in memory on JobTracker

Counters

Figure 2-6, MapReduce Design Patterns



How Hadoop MapReduce Works

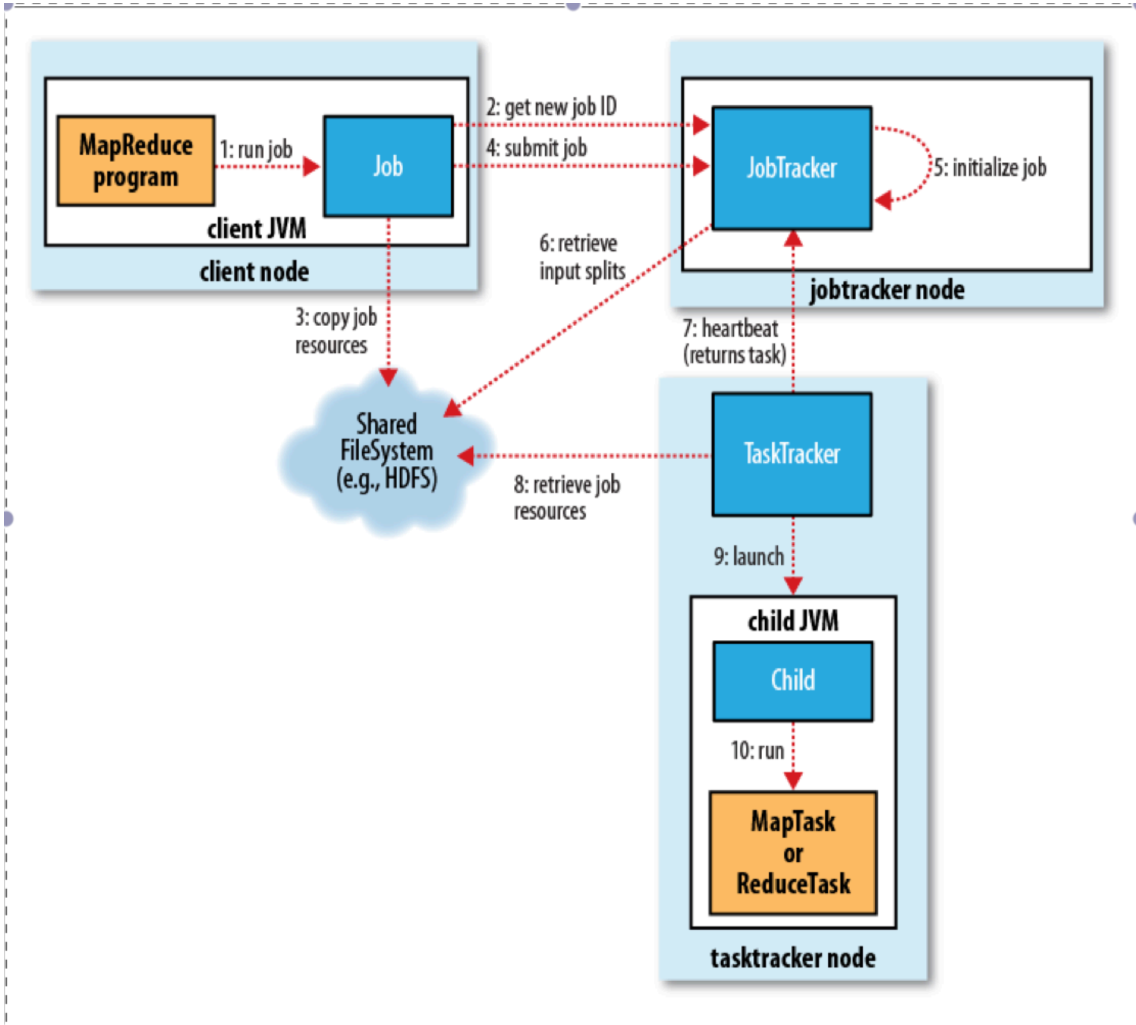
- Since we've seen some terms like
  - Job
  - JobTracker
  - TaskTracker

- Let's understand what they do

- Details from Ch. 6, *Hadoop: The Definitive Guide 3rd Edition*

How Hadoop MapReduce Works

Figure 6-1, Hadoop: The Definitive Guide 3rd Edition



Job Submission

- Job submission
  - Input files exist?
  - Output directory exist?
  - Copy resources to HDFS
    - JAR file
    - Configuration file
    - Computed file splits

Job Tracker

- Creates task (work to be done)
  - Map task for each input split
  - Requested number of reduce tasks
  - Job setup, job cleanup task
- Map tasks are assigned to task trackers that are "close" to the input split location
  - Data local preferred
  - Rack local next
- Reduce task can go anywhere. Why?
- Scheduling algorithm orders the tasks

Task Tracker

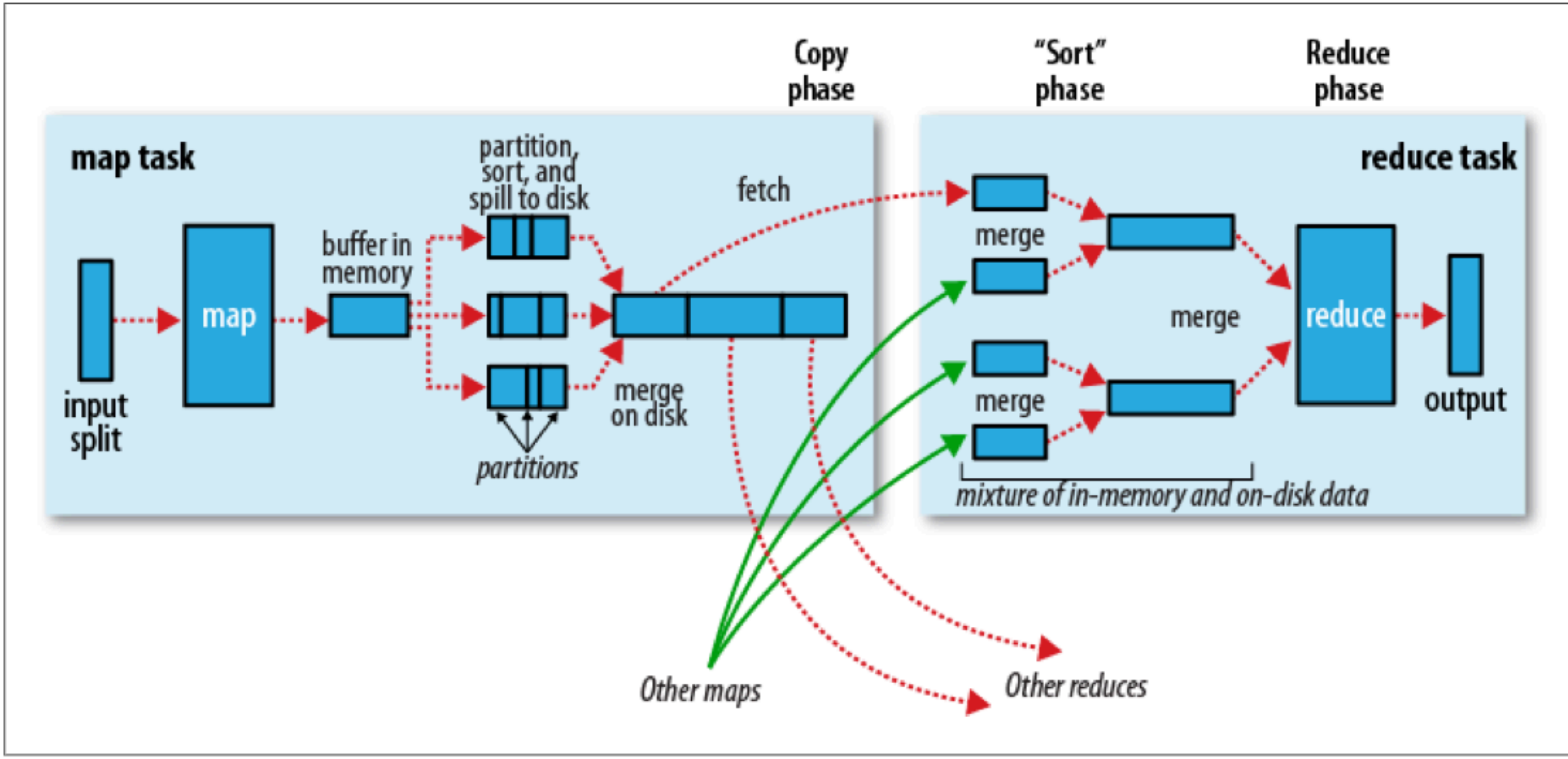
- Configured for a # of map and reduce tasks
- Periodically sends a "heartbeat" to job tracker
  - "I'm still alive"
  - "Ready for new task"
- For a new task:
  - Copy files to local file system (JAR, configuration)
  - Launch a new JVM (**`TaskRunner`**)
  - Load the mapper/reducer class and call its method
  - Update the task tracker of progress

Task Progress

- Mapper
  - What proportion of the input has been processed
- Reducer – more complicated
  - Sort, shuffle, and reduce are considered here
  - Progress is an estimate of how much of the total work has been done

Shuffle

Figure 6-6, Hadoop: The Definitive Guide 3rd Edition



MapReduce in Hadoop

Figure 2.4, Hadoop - The Definitive Guide