Lecture 1: MapReduce: Counting at Scale
Modeling Social Data, Spring 2017
Columbia University

Samuel Meshoyrer

February 10, 2017

1 What is Hadoop?

- An abstraction for parallel data analysis
- consists of many subprojects
  - we will mostly focus on Map/Reduce
- deals with distributed data
- born out of open source web indexing, crawling, and searching software
- Map/Reduce revealed by Google in 2004
  - added to Hadoop, which is adopted by Yahoo! in 2006

2 Why do we need Map/Reduce?

- There is still a lot of latency when dealing with a lot of data
- Read speed of commodity hard disk is about 1 TB/4hrs
- Using Hadoop, 1PB can be sorted in 16.5 hours! petabytesort

3 Map/Reduce

- break into parts
- process in parallel
- combine results

For example, if we wanted to count the number of occurences of each word in a book:

1. For every word on every page
2. Map to (word, count) e.g. ("cat", 1)
3. Shuffle to collect all records with the same key (word)

   hash(val) = hashVal mod number of reducers
4. Reduce results by adding count values for each word

To use Map/Reduce, you must specify the Map and Reduce steps

4 Principles

- move code to data
- allow programs to scale transparently

5 Strengths of Map/Reduce

- batch, offline
- write-once read-many
- simple computations
- I/O bound by disk or network bandwith

6 Weaknesses

- does not work well for high performance parallel computing applications
- low latency on random access
- not always the right solution
- difficult to find points of failure

Before you use Map/Reduce, you should make sure it's the right tool for the job.

7 Beware the Curse!

- often there is an assumption that all machines in the Map/Reduce pipeline end up with approximately the same amount of work
- this is not always true
    - not the case when working with skewed data
    - can do clever shuffles to avoid this if there is prior knowledge of skewness in the data

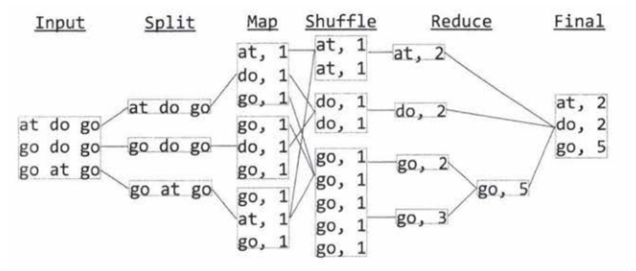

Figure 1: In this example, we see a graphical representation of a word count program using Map/Reduce. The Map step emits tuples of (word, count). The Shuffle step brings all corresponding words together. The Reduce step adds the counts of all the corresponding words. In the final step we have a list of tuples, each with its own word and the count of that word in the original input.

8 Examples

Below are two different examples of MapReduce, taken from "Cracking the Coding Interview, 6th Ed." by G. McDowell.

8.1 WordCount

See above.

8.2 Average Temperature

Say you are given a list of data in the form (City, Temperature, Date). You wish to find the average temperature for each city in a given year. How would you set up the Map and Reduce steps to perform this calculation?

- **Map**: Given a data point, the Map step will emit a (Key, Value) tuple of the form (CityYear, [Temperature, N]). The "Key" will be a particular city and year. The "Value" will be a tuple of Temperature and N, where N indicates the associated temperature is the average of N data points. This will be important in the Reduce step.
- **Reduce**: Given a list of emitted values, the Reduce step will take the weighted average of temperatures for each CityYear, where the weights will be N.