15-440 Lab #4: Map-Reduce
Wikipedia Corpus w/Cloud^9

Vital Stats

Partners: Yes
Due Date: April 28th
Handin procedure: To be described

Introduction

The Map-Reduce paradigm is really good at processing data – huge stores of data, in particular. There are a few sources of large-scale public available data: The Web, *Wikipedia*, and various high-resolution image sets immediately come to mind. Until recently *NetFlix* also provided a free data set as part of their competition, but not anymore.

Wikipedia has become very popular in the education community for use in teaching about the Map-Reduce paradigm and *Hadoop*. This is because it is (a) huge, (b) record-oriented, (c) has all sorts of interesting inter-relationships, which make it huger in complexity, and (d) enough people are using it that some of them are developing great tools (thanks, Maryland!). *Medline* is another increasingly popular example, but seems to have less sizzle for computer scientists. In this lab, you'll be analyzing the *Wikipedia* corpus, in particular, the full text of all of their articles and pages. It is about 27GB large and contains nearly 6 million independent articles.

The Question

We'd like you to ask, and use *Hadoop* to answer, some question about the *Wikipedia* corpus. These questions are designed to require more than one pass, e.g. more than one *Map*-Reduce phase. You are most welcome to ask and answer any question you'd like with our approval (just ask), but a few good questions you might ask include:

- If *Google*'s published *PageRank* algorithm is applied to the articles, what are the top-ranked articles?

- Determine the distance from one article to all articles within some distance (links) of it, or, alternately, determine the distance between any pair of articles (One semester, students had fun with this and an article about a person of religious significance to many).

- Determine how related some set of articles is to each other, e.g. cluster the articles. *Hint:* Although it might be tempting to throw K-Means clustering or Canopy directly at the articles, and judge how related they are based on distance, this may not prove to be a great idea – that requires chasing around the graph. It might be better to preprocess some characteristic, e.g. key words, and then find the relationship among those.

- Determine which authors' have had the greatest impact upon *Wikipedia* by performing a citation analysis, e.g. how many people cited them directly? How many people cited those who cited them?

- Determine which authors are working in the same field through a co-citation analysis. In other words, look to see which authors tend to be cited together in papers, even if they do not happen to cite each other.

In choosing among these questions, you've got to pick your own adventure. The easiest thing to do is probably *PageRank*, because it has been done by everyone and their brother and sister. You can easily use your favorite search engine to find *Google*'s original paper, which applied to the Web, and all sorts of information about doing it with *Hadoop*. You're most welcome to read any descriptive text you can find, and you are most welcome to read any *PageRank* code you can find that doesn't process *Wikipedia* articles. But, please don't look at other code out there related to *Wikipedia*. The same principle applies to the other areas of inquiry, as well. Descriptions of the algorithm can be found, among other places, here:
http://en.wikipedia.org/wiki/PageRank
http://infolab.stanford.edu/~backrub/google.html

And, if you do this first, for practice, *Wikipedia* should become somewhat more straight-forward:
http://www.umiacs.umd.edu/~jimmylin/cloud9/docs/exercises/pagerank.html

The clustering algorithms, especial *K-Means* tend to be fairly straight-forward and well discussed on the Web, including *Hadoop* implementations to solve various problems. The real trick there is just wrapping your brain around the idea of clustering, if it is unfamiliar to you. One of our doctoral students, a sometimes-Googler, once produced this tutorial for the *NetFlix* dataset:

http://kheafield.com/professional/google/more.pdf

Finding distances turns out to be as much brute force as anything. It is slow going. The Web provides a few good discussions, including the one below. But, remember, you'll only want to go a few to a small several steps from the starting point, or things might blow up on you. You'll need to experiment to find the right distances.

http://www.johnandcailin.com/blog/cailin/breadth-first-graph-search-using-iterative-map-reduce-algorithm

If you are doing some type of citation or co-citation analysis, you can basically roll your own. What you'll probably find yourself doing is counting the number of direct citations or parental co-citations, and the number that you find with each step away from the original. But, you'll probably dramatically devalue the weight of the indirect citations as you get farther away. None-the-less, how you do this is up to you. In some ways, like shortest path, it is a BFS problem.

The Tools

You'll use the *ghc* cluster to answer one of several suggested questions about the dataset – or, with the permission of the course staff, one of your own. In order to make your lives easier, we've preloaded the data onto */wiki* on the cluster, and preprocessed it for you into a block-compressed *SequenceFiles* which are readily processable using the *Cloud*^{9} toolkit, which we've also installed into */usr/local/cs/hadop-0.20.1/* on the cluster machines. You are set to go.

A really good starting place is the *Cloud*^{9} page:

http://www.umiacs.umd.edu/~jimmylin/cloud9/docs/index.html

And, remember, in addition to their documentation, you can see the source code form their examples. It gives you a really good idea how things work. Look especially at the code relating to the *Wikipedia* corpus:

http://www.umiacs.umd.edu/~jimmylin/cloud9/src/dist/edu/umd/cloud9/
http://www.umiacs.umd.edu/~jimmylin/cloud9/src/dist/edu/umd/cloud9/collection/wikipedia/

If you'd like to start on something smaller, than the full set, feel free to download the first few blocks of the file from the Web. Just remember that the last record won't be complete, so don't let your program die when it gets that exception, just proceed with those records you do have.

The Deliverables

- Your source code
- The relevant portion of the raw output of your program
- A paper, in .pdf form, written in English prose describing (a) what you set out to do, (b) what you did, and (c) the meaning you derived from your raw output, e.g. the answer to the question you asked. Be sure to explain the question you asked and the answer you got in terms fit for the laity – not in a mathematic way.