CS1699: Cloud Computing

Project 1: Tiny-Google Design Document

Jacob Hershey - [jah215@pitt.edu](mailto:jah215@pitt.edu)

Nick Taglianetti - [nct11@pitt.edu](mailto:nct11@pitt.edu)

**Technology**

* We will be implementing the MapReduce jobs of the Tiny-Google inverted index project in Java.
* The MapReduce jobs will be run and tested on a single-node cluster using the cloudera/quickstart image in a Docker container.

**Project Approach (subject to change slightly)**

Component 1: User Interface

We will implement a simple command-line menu interface in a driver class (tiny\_google.java) with three options:

1. Index a document, specifying the path of the input. This will run a MapReduce job on the input in which the different words of a document are counted, such as in WordCount.java from the second assignment. The output of the MapReduce will be added to the inverted index data structure.
2. Search indexed documents based on query, one or more keywords (ie. Strings). This will run another MapReduce job on the inverted index to search all documents in the data structure for the queried keyword(s).
3. Quit the program.

Component 2: Inverted Index Data Structure

When a user specifies a document to index, assuming the file is already in HDFS, perform indexing via MapReduce.

In the Mapper function of this MapReduce job, parse the document, index the words of the document (such as in WordCount.java from Assignment 2) using the word itself and number of occurrences in the document as the key (eg. “theWord-12”), and the context around the word as the value (eg. “4wordsBefore-theWord-4wordsAfter”). The context words can be delimited by a different delimiter (eg. “\t”) than the indexed word itself (eg. “-”).

In the Reducer function, build the inverted index from the formatted results of the mappers, where they will be sorted in order of word (alphabetically), number of occurrences of that word (descending), and lastly the document name (alphabetically).

Third Component: Rank and Retrieval

With the Inverted Index in HDFS, perform another MapReduce job to rank and retrieve the keyword term, the number of times that term occurs in the document, the context of that term in the document, and the document number. More specifically, the mapper will take the Inverted Index from the previous component as input and make a key of the document number, number of times the term occurred in that document, and the value (the context) of the term in that document. These key-value pairs will be sorted by Document, so as to have all documents with their values for the key terms together. This data can then go into the reducer which will rank the documents by adding together the number of times the term occurred in a document, until it reaches a new document, at which point it can check to see if this sum total is larger than any of the other documents in our list of top documents (similarly to how max temperature was returned in the MapReduce job of Assignment 4).