# Tiny Google Design

Nick Risinger

Description:

Tiny Google will be two separate applications. One for the MapReduce framework and one for the Spark framework. The UI will be a command line interface with three instructions: index, search, and quit. Index will index all the books in the Books directory. Search will search for the search query terms and return files, occurrence counts for each individual keyword, and line numbers for each individual keyword. Results will be ranked in inverse order by the total number of occurrences of all keywords in the file. With both index and search, the time to complete the respective indexing and searching will be displayed to the nearest tenth of a millisecond.

For indexing, each document name will be passed to a mapper (or parallelized in spark). The mapper will open the document with the given name and produce an InvertedIndex with the posting lists. Posting list entries will have document IDs, occurrence counts, and line numbers of occurrences. The mapper will return this InvertedIndex with the document filename as the key. The reducer will combine InvertedIndexes into a single InvertedIndex, combining the posting lists for each word and sorting them in decreasing order of occurences.

For ranking and retrieval, each query term will be passed to a mapper (or parallelized in spark). The mapper will return the query term as the key and a list of documents with the query term, occurrence count, and line numbers of occurrences. The reducer will then combine each document list by summing the occurrence count and appending the keyword and associated occurrences and line numbers.

Performance will be tracked by starting a StopWatch at the beginning of an index or search command and stopping it once the result has been obtained. The time elapsed will then be displayed along with the result of the query.

# Outline

## UI

Command-line interface with three options:

Index: indexes the folder of interest using the selected framework

Ex: Input: “index spark”

Output:

“Indexed. Ready to search.

Time to index: 1509.3ms”

Search: searches for a single query term or list of query terms (separately) using the selected framework

Ex: Input: “search mapreduce pillow monkey”

Output: “FileQ:

total occurrences: 12

line numbers: 1,3,8,23,65,88,101,122,143,156,167,184

FileC: …”

## MapReduce

## Indexing

Map each document to a mapper.

Within mapper:

Construct hashtable with stemmed word as index and LinkedList as value.

LinkedList entries will contain 1 entry with document ID and occurrence count.

Within reducer:

Combine hashtables of stemmed words into a single hashtable.

When entries exist for a particular word in each hashtable, add all LinkedList entries to the LinkedList for that word in the new hashtable.

When completed, indicate that it’s finished to the user.

## Ranking and Retrieval

Stem each query term.

Map each stemmed query word to a mapper.

Within mapper:

Construct linked list of documents containing terms in decreasing order of occurrence.

LinkedList entries will contain document IDs and occurrence count.

Within reducer:

Combine LL into a single LL of documents in decreasing order of occurrence.

If document exists in both search term's LL, sum occurrence count.

When completed, display results to user

## Spark

## Indexing

Parallelize a list of document names.

Map a function to open document and return hashtable of stemmed words with posting lists with the document ID and occurrence count.

Reduce hashtable via a combining function.

When completed, indicate finished to user.

## Ranking and Retrieval

Stem the list of query terms.

Parallelize the list of query terms.

Map a function over the list to search over the InvertedIndex Hashtable and return sorted linked list of documents.

Reduce the sorted linked list of documents with a combining function that adds together occurrences of keywords for each document.

When completed, display results to user.

## Performance Tracking

A stopwatch will be used to time the indexing and ranking-and-retrieval. The time taken to finish indexing and RaR will be at the end of the output.

## Optimization

Could try different data structures for the InvertedIndex and Results

Could try different number of mappers/reducers