Cloud Computing Project

Project Goals

Data-intensive Computing and Cloud Computing are two emerging computing paradigms, which are poised to play an increasingly important role in the way Internet services are deployed and provided. Increasingly, large-scale Internet services are being deployed atop multiple geographically distributed data centers. These services must scale across a large number of machines, tolerate failures, and support a large volume of concurrent requests.

The main objective of this project is to design and implement a basic data-intensive application to index and search large documents. More specifically, the goal is to design a simple search engine, referred to as **tiny-Google**, to retrieve documents relevant to simple search queries submitted by users.

This project is designed to provide students with exposure to new programming models of computing and processing of large scale data using **MapReduce** and **Spark**.

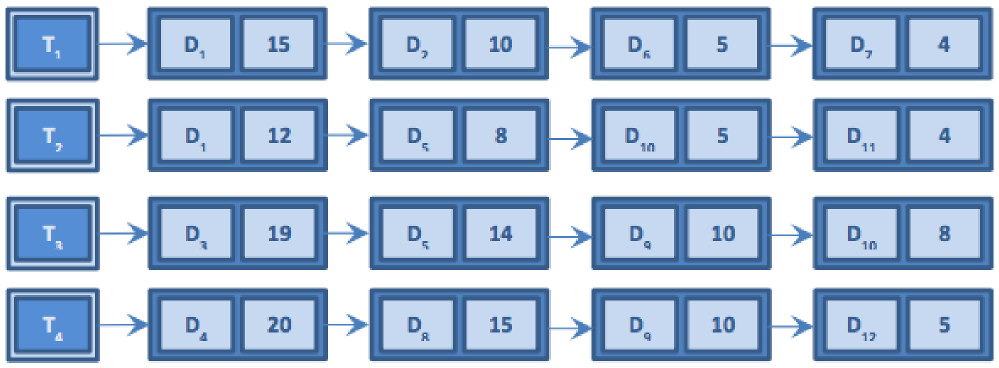
The design and implementation of **tiny-Google** involves three basic components:

1. The first component consists in designing a simple user interface (UI) that
   a. allow users to index a document, and
   b. enter simple search queries and retrieve relevant document objects. The search query might consist of a short list of keywords.
2. The second component consists in developing a data structure, referred to as inverted index (II), to support the full-text search component of an information retrieval engine. In its basic form, an II contains a posting list for each term. The posting list is a linked list of individual postings, each of which consists of a document identifier (id) and a payload. The id value uniquely identifies a document, while the payload contains information about the occurrences.
3. The third component, ranking and retrieval (RaR), consists in retrieving the documents relevant to the query in a ranked order. Given a search query for a pattern of words, RaR returns all the documents that contain the words of the search pattern, rank- ordered in the decreasing order of the word count associated with each document.

Inverted Indexes

Given a collection of documents, **D**, an inverted index is a data structure, which given a term provides access to the list of documents that contain the term. In its basic form, an inverted index consists of: A set of terms, **T**, and A set of posting lists, **L**, each of which is associated with a specific term in **T**.

A posting consists of a document identifier and a payload, for each document in **D**, which contains the term associated with the posting list. Depending on the objective of the application,   the payload filed may be empty, in which case the existence of the posting only indicates the presence of the term in the document, or contain additional information relevant to the frequency of occurrence of the term in the document. A simple illustration of an inverted index is depicted in Figure below. In this example, the inverted index contains four terms and reports on the occurrence of these terms in twelve documents.



Project Implementation

You are to develop the **tiny- Google** search engine, which consist of three components described earlier, using two cloud computing paradigms: **MapReduce** and **Spark**. In each paradigm,

1. Write a program to develop a master inverted index structure to index a collection of documents in the HDFS directory below:
   /user/chatree/CS1699/Books/
   Each term in the inverted index is associated with a posting list, whereby each posting contains the identifier of the document and other information needed for ranking and for location of the term appearing in the document.
2. Write a program that takes one or more keywords and then search the index file and return the postings associated with these keywords, sorted by some criteria, ie. the number of occurrence of these keywords.
3. Measure the performance of each paradigm in term of total time for indexing, and querying
4. When possible, experiment with various techniques to optimize the execution time

The following are the expected deliverables and due dates for this project:

What to submit	Due Date
The design of the programs described above, including the techniques used to optimize the performance of the system, if applicable	December 6
The final report discussing 1. Your implementations, 2. The techniques use to optimize the system, when applicable, and 3. The performance comparison between two implementations.	December 17
Schedule a time to demonstrate your project	December 6 - 17