PROJECT PROPOSAL

WENJING YU, PENG WAN, YULING DENG, ZHEN LU

1. Background

Modern information retrieval becomes an important role in daily life. Many research topics are based on this area search engines are one of them. Many of us have built search engines before using tools such as *Apache Lucene*[1].

The technologies for search engines are well-developed by now. They all use classic algorithms and there are lots of tools such as *Lucene* are built for creating search engines. So we can build a search engine easily using these frameworks.

However, these tools provide implemented parallel algorithms for building a search engine thus we are not very familiar with the parallelism inside search engines. So for this course project, we decide to implement the main algorithms and parts for search engine to understand the parallelism of search engine better.

2. Problem Description

For modern search engines, there are 4 main parts, including crawling web pages, indexing documents, ranking and querying.

Web crawler handles obtaining data from the web. Giving some starting URLs, a crawler downloads the pages and parse the page to extract the URLs inside the page.

The most common method for indexing is to represent the documents using vector space model. This part mainly focuses on the representation and building an inverted index for documents.

When it comes to rank, PageRank is the most popular algorithm to measure the importance of a page.

At last, when a query is given, we can use the inverted index and calculated PageRank for each page to get the result matches the query best and then returns a ranked list to user as the final result.

[1]For more details, see http://lucene.apache.org

3. Proposed Work

Based on the main parts of search engines. This project can be divide into 4 parts as well. Details for each part will described below.

- **Crawling:** implement a parallel web crawler for obtaining data from some seed URLs.
- **Indexing:** use vector space model for document representation, then build the inverted document index for the dataset.
- **Ranking:** implement a parallel version for PageRank algorithm.
- **Querying:** build the parallel query system to get the search result for a specific query.

The techniques we decide to use to implement this project is MapReduce. We will use Hadoop as the MapReduce implementation. So the language would be JAVA.

The reason why we choose MapReduce is that MapReduce is an elegant framework for parallel programming. And applying MapReduce would make this project more interesting to work with.

The main focus for this project will be on Ranking algorithm and Querying method. Crawling and indexing part does not have much dependent data for each single document, that would make this easy to implement and less interesting. However, calculating PageRank and search the related document for a query requires many steps and they share lots of data. For example, when calculating PageRank, we need to use the result from previous iteration to calculation the value in current iteration. That requires more techniques to handle data dependency.