**Group 9 Members:**

Yiyun Zhang

Michael Karaman

Alexys Lamkin

**Homework 1: Chapter 4 Summary**

There are three conceptually distinct components in the web search problem, crawling the web, inverted indexes and queries response. Chapter 4 focuses on Inverted indexing, which is a large distributed sort and group by operation, most suitable for MapReduce.

The web crawler feature can be achieved in a few hundred lines of code, however, it may encounter many issues such as web servers overloading, pages prioritization, coordination and load-balancing, content changes, duplicate content, multilingual and language identification.

The inverted index is posting lists, one associated with each term that appears in the collection. For each individual posting, it contains a document id and a payload which is the information about occurrences of the term in the document.

In inverted indexing implementation, each document is analyzed and broken down into its component terms first. After this histogram has been built, the mapper then iterates over all terms. For each term, a pair consisting of the document id and the term frequency is created.

The mapper then emits an intermediate key-value pair with the term as the key and the posting as the value. In the shue and sort phase, the MapReduce runtime essentially performs a large, distributed group by of the postings by term. The reducer begins by initializing an empty list and then appends all postings associated with the same key (term) to the list. The postings are then sorted by document id, and the entire postings list is emitted as a value, with the term as the key(Jimmy Lin and Chris Dyer, 2010, p. 76). This baseline implementation of inverted indexing algorithm has scalability bottleneck problems that have to buffer postings in memory.

The value-to-key conversion design pattern addresses the issue by offloading sorting postings by document id task to the MapReduce execution framework. For each key-value pair, a posting can be directly added to the postings list. Since the postings are guaranteed to arrive in sorted order by document id, they can be incrementally coded in compressed form| thus ensuring a small memory footprint(Jimmy Lin and Chris Dyer, 2010, p. 78).

There are several index compression techniques for index compression and storage such as using byte-aligned, word-aligned, bit-aligned codes for integer compression and smaller, faster postings compression. There are two main partitioning strategies for distributed retrieval: document partitioning and term partitioning. The document partitioning broke the entire collection into multiple smaller sub-collections, each of which is assigned to a server. In term partitioning, each server is responsible for a subset of the terms for the entire collection. A server holds the postings for all documents in the collection for a subset of terms. Document partitioning has sharter query latency and term partitioning can theoretically increase a system's throughput. In general, studies have shown that document partitioning is a better strategy overall.

Jimmy Lin and Chris Dyer. 2010. *Data-Intensive Text Processing with MapReduce.* Morgan and Claypool Publishers.