CSCI1380 Spring 2024

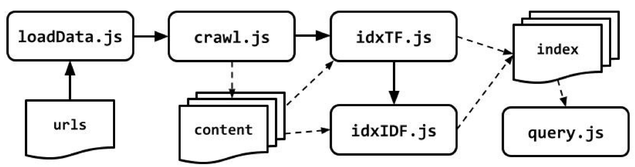
GitGitGo: A Distributed Repository Search Engine

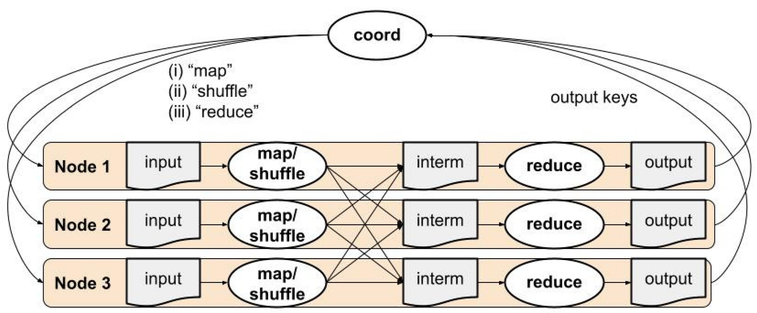
Alex Ding · Megan Frisella · Sophia Liu · Tianran Zhang
GitHub: https://github.com/meganfrisella/gitgitgo

Introduction

Our project is a distributed and scalable GitHub Readme search engine with crawling and indexing workflows on a custom distributed MapReduce implementation. The distributed system runs on 16 AWS EC2 servers over 300k GitHub Readmes.

Methodology





Libraries
Our code adapts the following node.js libraries: natural, fs, yargs, crypto, and @supercharge/promise-pool.

Acknowledgments
We want to thank Professor Vasilakis and the CS1380 TAs.

Design

Distributed File Store

- Sharded key-value store organized by collections
- Efficient appends via custom encoding and OS file appends
- Optimized store.get returns only first **k** entries in a file

TF-IDF Index

- Splits documents into words and stems them using the Porter stemmer.
- Evaluates word importance by comparing frequency in a document to the entire corpus.
- Indexes are distributed over files in a store, with keys representing terms and values listing matching documents sorted by score.

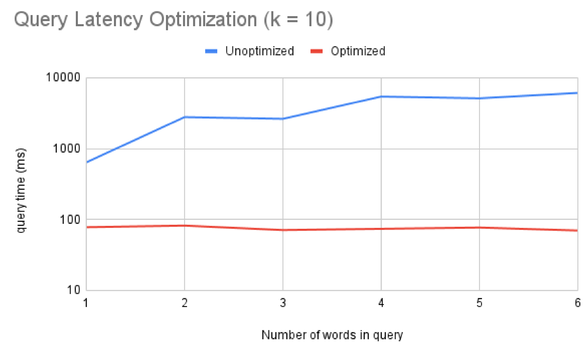
$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Query

- Querying term **t** is just store.get(t) in distributed index
- Use optimized store.get to only return first **k** entries
- Query split into terms; per-document TF-IDF scores summed for each term and sorted

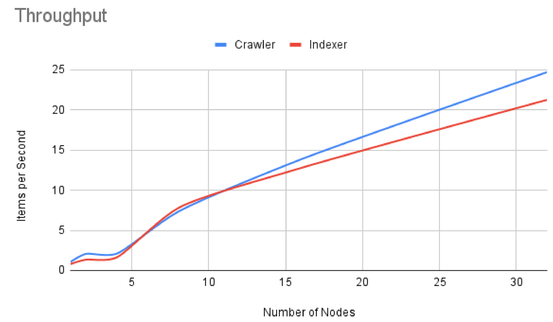


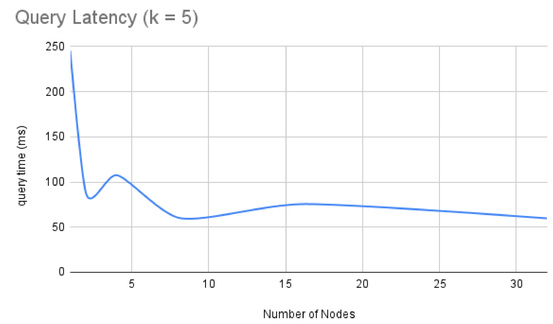
Distributed MapReduce

- Custom distributed MapReduce implementation built on top of a distributed store.
- Crawler and indexer implemented as MapReduce workflows.
- Optimized for scalability, successfully handling up to 5 million keys.

Results

We tested our search engine's **scalability** on different node counts with a 1000-repository sample. Throughput increases significantly with more nodes, while query latency remains roughly constant due to no computation during querying.





The full system **throughput** and **query latency** performance are revealed by the following table.

Component	Runtime (s)	Throughput (items/s)
URL loader	101	2998
Crawler	17367	17
Indexer	7334	41

Table 1: System throughput (16 nodes on $N = 302824$ repositories)

Further Work

- Replace the CLI with a web interface
- Enhance the crawling workflow to dynamically scrape new repositories