Building a Search Engine using Personalized PageRank

Anton Lund, Isaac Rondon, Andreas Cederholm, Muhammad Haky Rufianto

School of Computer Science and Communication, KTH, Stockholm
{antlundh,rondon,ance,rufianto}@kth.se

Abstract. We have studied the Personalized PageRank Algorithm in MapReduce. We created an implementation of a search engine using Apache Nutch to crawl a web site. The data generated by Nutch was turned into link data that was used to run Monte Carlo simulations on the web graph and later processed by Apache Haddoop into PageRank scores. Lastly, the content was indexed using Apache Solr. The approach gave reasonable results but using distributed processing to compute Personalized PageRank is probably excessive for most web sites.

1 Introduction

The growth of the web as well as social media giants such as Facebook and Twitter has created a demand for infrastructure that can run computations on large graphs and data sets. This technical paper revisits the PageRank algorithm and the ubiquitous MapReduce algorithm.

This paper was inspired by a SIGMOD Conference entry, Fast personalized pagerank on mapreduce, that describes how a fast fully personalized PageRank algorithm can be adapted to the MapReduce framework [1]. The goal of this paper was to get acquainted with commonly used frameworks such as Apache Nutch, Solr and Hadoop by crawling a web site and constructing a web site-specific search engine using PageRank computed on a distributed cluster. The paper will start by by introducing the background concepts. This will be followed by a description of the technologies used as well as the implementation. Lastly, the results of the implementation will be discussed.

2 Background

2.1 PageRank

PageRank is an algorithm used for ranking nodes in a directed graph. The ranking of a node is given by the number of nodes linking to it and the rank of the linking nodes [2]. The most widely known use of PageRank is Google Search.

One way to calculate pagerank is using the Monte Carlo approach. With the Monte Carlo approach we simulate a random walk in the graph and count every visited node. In each step of the walk, the algorithm either follows a link from the node or, with a set probability, teleports to a random node. Since walks are independent, several simulations can run in parallell. With enough iterations, the node visit frequency will approximate the distribution of visits in the PageRank model.

When calculated using the Monte Carlo method, Personalized PageRank works the same way as PageRank except when teleported it teleports to a limited set of nodes, often a single node. The rank of each node can be thought of as the probability to reach the node when doing a random walk from the selected start node. In this paper, we use the Perzonalized version of the algorithm and the homepage of a web site as a start node.

2.2 MapReduce

MapReduce is a framework for distributed processing of large data sets. It consists of three operations on data structures in the (key, value) format: Map, Shuffle and Reduce. In the context of Monte Carlo PageRank these could take the form:

1. Map - A number of Monte Carlo iterations are sequenced and link (key) and number of visits (value) are output for each link.
2. Shuffle - Sort the data so that each key goes to a specific node in the cluster.
3. Reduce - Sum over each key and output the total number of visits for each key.

3 Technology

We decided to use software from the Apache Foundation, Nutch, Solr and Hadoop. They are all Open Source Java programs and Nuch and Solr are often used in combination.

Apache Nutch is a web crawler that can save and cache web pages from the internet . Apache Nutch is scalable and extensible, it provides many customization features when parsing or indexing. Also, it provides the possibilities to create a plugin that can be used together with the crawling process to get more defined data from the crawling process. By parsing the crawl data generated by Apache Nutch, we can create a graph representing the link structure for PageRank and the content of each page for search indexing [3].

Apache Solr is a search engine platform from the Apache Lucene project that among many things manage indexing and querying. Solr can be used create search engines of websites, databases and files. Solr is able to achieve fast search responses because, instead of searching the text directly, it searches an index [4].

Apache Hadoop is a Java framework for distributed computing consisting of a storage part (Hadoop Distributed File System (HDFS)) and a processing part (MapReduce). The process is overseen by YARN, which contains a Resource Manager that allocates resources to different applications [5].

4 Implementation

4.1 Crawling

With Apache Nutch, we crawled the Harry Potter Wikia. The web site contains a total of approximately $40,000$ pages mostly consisting of articles on different topics. It is relatively small compared to big websites but sufficiently big for our purposes. The process can be described by the following steps:

1. Choose the starting page as seeds for Apache Nutch. It will determine how big data you can get at the end of the crawling process. The starting page for our crawling engine was `http://harrypotter.wikia.com`.
2. Defining constraints. For this project, we only focus on the subdomain of `harrypotter.wikia.com`, Therefore, for the crawling process, we used a Filtering Plugin. This plugin uses regular expressions to check all the outgoing link from the processed page, if the pattern of the outgoing URL matched the Regex, then it will processed on the next iteration, otherwise it will disregard the link.
3. When the crawling has finished, we use Python scripts to convert the dump into easily readable XML documents for the link and content respectively.

When crawling the site, we encountered a few difficulties that made Apache Nutch less effective. Some of those obstacles might deny the crawler to cache all the information in a web page. The first problem is when a webmaster uses a robots.txt that prohibits the crawler to read and cache some or all of the content. The second obstacle is when the author of a web page inserts a META tag to avoid the page being indexed. On the Harry Potter Wikia, we found several pages that had this feature. Lastly, we encountered problems with redirection links. This also becomes a problem when we calculate the PageRank, because the redirection page does not have any content and only has a redirection link.

4.2 Our Search Engine

Figure 1 represents how the differents technologies were tied together in order to achieve the goal of this project. The first technology involved in the process is Nutch. btaining two kind of outputs: a link structure (Crawl Links) with all the links referenced on each page and each link with its respective content (Crawl content). The link structure was used to calculate the PageRank score of each link and the second was used to push all the content to Solr in order to create the Search Engine.

The next step consisted of creating several parsers to transform the results to valid inputs with the corresponding format for the rest of the technologies. A java program was created in order to parse the link structure to create a site graph and generate Monte Carlo simulations on the site graph.

The next step consisted of using Hadoop to transform the aggregated results from each simulation into a PageRank score. Finally, the content data was merged with the PageRank scores and indexed by Solr resulting in a queryable search engine.

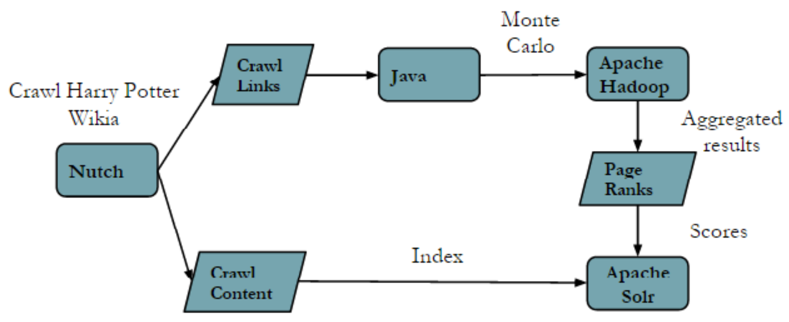

Fig. 1. Schematic representation of how the components tie together.

5 Conclusions

In this paper, we discussed an approach to construct a search engine using a personalized PageRank algorithm computed on a distributed cluster. Achieving this we got familiarized and obtained practical experience with technologies used in industry today, including technologies as Apache Nutch, Solr and Hadoop.

Translating complex websites with redirects and broken links into a coherent data represents a big challenge for this and any similar approach and moreover to handle several encoding practices between different formats requires a big effort when technologies are tied together. The overall approach returned reasonable results but a distributed computing process is probably not needed even for moderately large websites. It is possible that it would be useful for really large websites.

References

1. Bahmani, B., Chakrabarti, K., & Xin, D. (2011, June). Fast personalized pagerank on mapreduce. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 973-984). ACM.
2. Manning, C.D., Raghavan, P, & Schtze, H. 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.
3. Apache Nutch, `http://nutch.apache.org/` [Accessed 24 May 2015]
4. Apache Solr, `http://lucene.apache.org/solr/` [Accessed 25 May 2015]
5. Apache Hadoop, `https://hadoop.apache.org/` [Accessed 24 May 2015]