Project Report

5130309515 litianchu

1. Introduction

One purpose of my project this term is to design a website for data searching from DeepDive, a database of our group. And another is to collect the academic papers data as many as possible.

2. Project

2.1 Website for DeepDive

2.1.1 Content

The first part of my project is the website of DeepDive construction.

This is a relatively easy project. I just use HTML, CSS to realize the layout of the whole web. Then, Javascript and Jquery are used to achieve the certain function like search by author, title or catalog, the paper sorted by relevance, author, or alphabet. My website mainly contain two subsite, homepage and result page. To make it easy to find the result, I divide the result page into two part. One is made up with all results relevant to search content, and every paper item contains its title, author and abstract hiden in scroll block which you can put the mouse on to expand. Another is paper website contain the specific paper of PDF.

2.1.2 Tools

Tools	Function
HTML, CSS	for static page
Javascript, Jquery	for dynamic page
Web.py	for web backend

2.2 Crawler

2.2.1 Crawler of Nutch

At the begin of this term, I choose Nutch, a general crawler frame written by JAVA, to crawl considering that Nutch is designed for search engine, which is just the main task of my group and easy to combine nutch with Hadoop, hbase to realize distribute crawler.

The principle of Nutch is as the figure 1 shows.

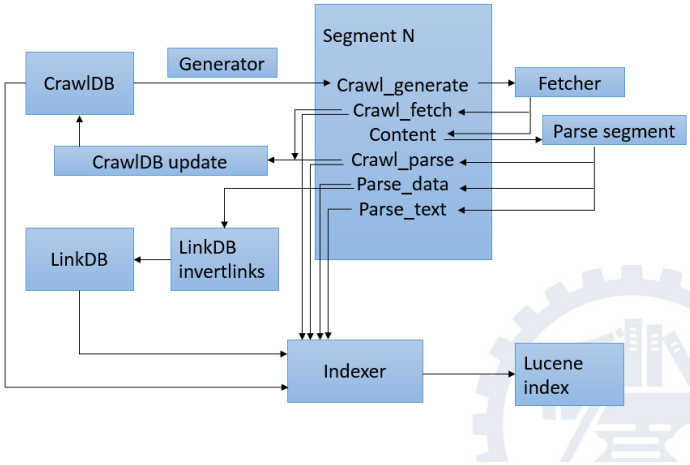

Figure 1 the principle of Nutch

However, during experiment, we find Nutch doesn't work as ideal as expected. Because Nutch focus on distribute crawler with Hadoop (Hadoop will take a lot of time), it will be slower than single crawler when there are few slave machines. Besides, it has no API for accurate extraction plug-in and total API for plug-in is only six, which meaning that my crawler has little flexibility

2.2.2 Crawler of Python

So to improve the data quality, I then use python to write crawler instead of Nutch. In this case, I choose to write a crawler by myself.

My crawler contain three basic functions specially for the website I crawl, which are getting the url of journals, getting the url of papers and parse the data from the paper page. The data I crawl contains the title, author, author affiliation, abstract, and doi (a unique number of every paper) of paper, and all of them are saved of json. BFS can be used in collecting urls.

Of course, just according to the basic principle to realize a crawler is bound to be at a low speed. The key of crawler is how to speed crawler up. To accelerate my crawl, first I use gzip, a method of compression to reduce the response of the website I crawl by 70%, meaning that it will cost 30% of origin time to receive the response.

Then multithreading and asynchronous IO are used into my crawler. As the name shows, asynchronous IO can switch to other coroutine named greenlet when the current greenlet meet IO congestion to gurantee that there are always greenlet running, and through experiments, it's proved that these two ways are of the maximum efficiency for single machine.

The final way I use is distribute crawler. In brief, the simplest distribute crawler is just crawl url in master, then distribute these urls to slaves to parse and store data in database. To realize the distribute crawler, I use redis to store urls and postagedb to store the data. What's more, I use bloom filter for url deduplication. It bases on hash and can find whether url already in set with time complexity of 1.

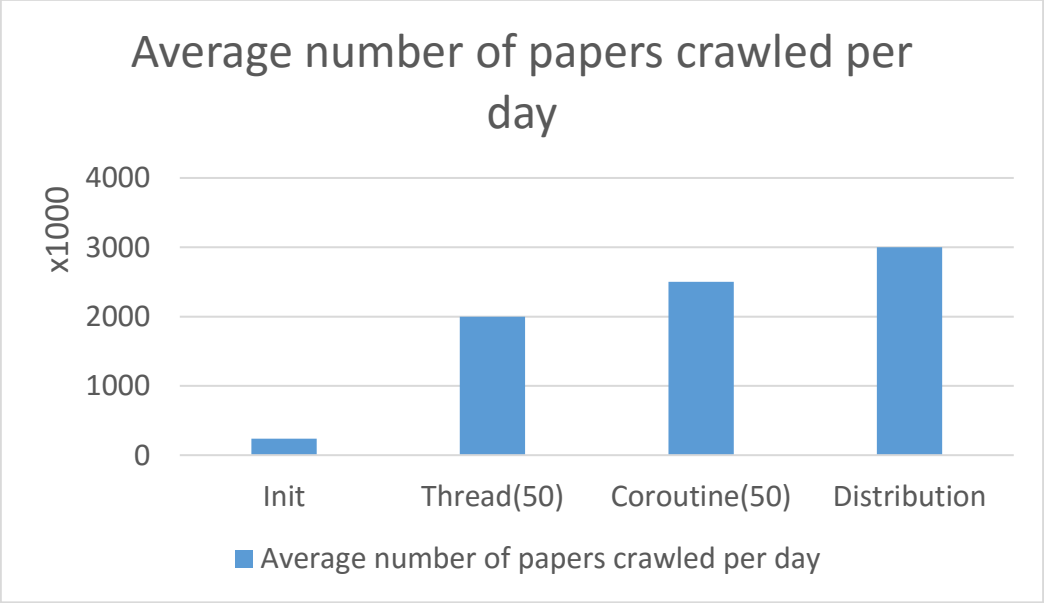

Figure 2 efficiency of different ways used

3. Problem and Solution

The main problem I met when crawling is that my ip is always banned by website for the huge visit times in a short time. To solve this problem, I first disguise my crawler as a browser by changing

the request headers it sends to website to be the same as a real browser performs. And then use procxy to cover my ip. In this case, I crawl a procxy website specially to collect different procxy ip and then put them into my ip pool.

4. Further work

Though there are some way to avoid my ip to be banned, the result is not as good as expected showed by experiments. So I will keep on solving the ban from the website and optimize the distributed crawler system. If all above problems are solved, the next step is to realize a crawler as general as possible.

5. Conclusion

Crawler is seem to be a easy project, but it is not enough to just realize it. The key of crawler is how to speed crawler up while avoid ip blocked. The project I achieve this term is just the begin of the academic data collection, there are more problem waiting for my group and I to solve such as filter the useful information and find the relative of these data.

Finally, thanks Mr. Wang and Mr. Tian for their teaching and guidance. Thanks assistant for a patience to answer and students for their wonderful lectures.