The Design and Implementation of Distributed Crawler System

Jinfeng Cao[1, a], Taizhi Lv[1, b] and Yu Zhuang[1, c]

[1] School of Information Technology, Jiangsu Maritime Institute, Jiangsu Nanjing 211170, China

[a]1439187163@163.com, [b]lvtaizhi@qq.com, [c]2200307645@qq.com

Abstract

Big data has penetrated into every industry and business, and it become an important factor of production. The mining and application of massive data is becoming more and more widespreadly. It indicates the importance of the crawler system. A distributed crawler system is designed and implemented in order to improve the efficiency of reptiles. This system can not only collect Internet information by search engine, but also be used as a directional information collector to collect specific information under certain websites, such as recruitment, rental information and so on. This crawler system is based on Java language. Redis is used as a repository to store temporary page URLs, HttpClient is used to download pages, HtmlCleaner is used to parse pages, MySQL is used to store information, ZooKeeper is used to monitor distributed crawlers.

Keywords

Distributed Crawler System; Java; Zookeeper; MySQL.

1. Introduction

With the rapid development of the network, the World Wide Web has become the carrier of a large number of information. How to extract and utilize Internet information effectively has become a huge challenge. Search Engine, such as Baidu，Yahoo! And Google, is used as a tool to help people retrieve information and becomes an entry and guide for users to access the Internet [1]. In order to solve the above problems, focused crawlers emerge as the times require grabbing relevant web resources. Focus crawler is a system that automatically downloads Web pages. It selectively accesses web pages and related links on the World Wide Web to obtain the required information according to the established crawling target. Unlike general purpose web crawler, focused crawler does not pursue large coverage. Instead, it aims to crawl web pages related to a specific topic and prepare data resources for topic-oriented user queries.

Web crawler is a program that automatically extracts web pages. It downloads Web pages from the World Wide Web for search engines. It is an important component of search engines. The traditional crawler starts with one or several URLs of the initial web page and obtains the URLs of the initial web page. In the process of crawling the web page, it constantly extracts new URLs from the current page and puts them into the queue until it meets certain stopping conditions of the system. The workflow of focused crawler is complex. It is necessary to filter topic-independent links according to a certain web page analysis algorithm, retain useful links and put them in the waiting URL queue. Then, it will select the next page URL from the queue according to a certain search strategy, and repeat the process until it reaches a certain condition of the system. In addition, all web pages captured by crawlers will be stored by the system, analyzed, filtered, and indexed for subsequent query and retrieval; for focused crawlers, the analysis results obtained in this process may also provide feedback and guidance for future crawling process.

2. Crawler Classification

According to the system structure and implementation technology, web crawlers can be roughly divided into the following types: general purpose web crawler, focused web crawler, incremental web crawler and deep web Crawler [2-4]. Universal Web Crawler, also known as Scalable Web Crawler,

extends the crawling objects from some seed URLs to the whole Web, mainly collecting data for portal search engines and large Web service providers. Focused Crawler, also known as Topical Crawler, refers to the selective crawling of web crawlers related to pre-defined topics. Incremental Web Crawler refers to a crawler that updates downloaded pages incrementally and only crawlers newly generated or changed pages. To some extent, it can ensure that the crawled pages are as new as possible. Deep Web data is more professional and more data-intensive than the information provided in static pages of other websites, and it is more valuable for users to make use of. The general search engine cannot crawl deep web data when crawling network information. For search engine users, the valuable information they can get is limited.

3. System Design

A distributed crawler system is designed and implemented in Java language, which can store information in different places. The architecture of the system is shown in Figure 1.

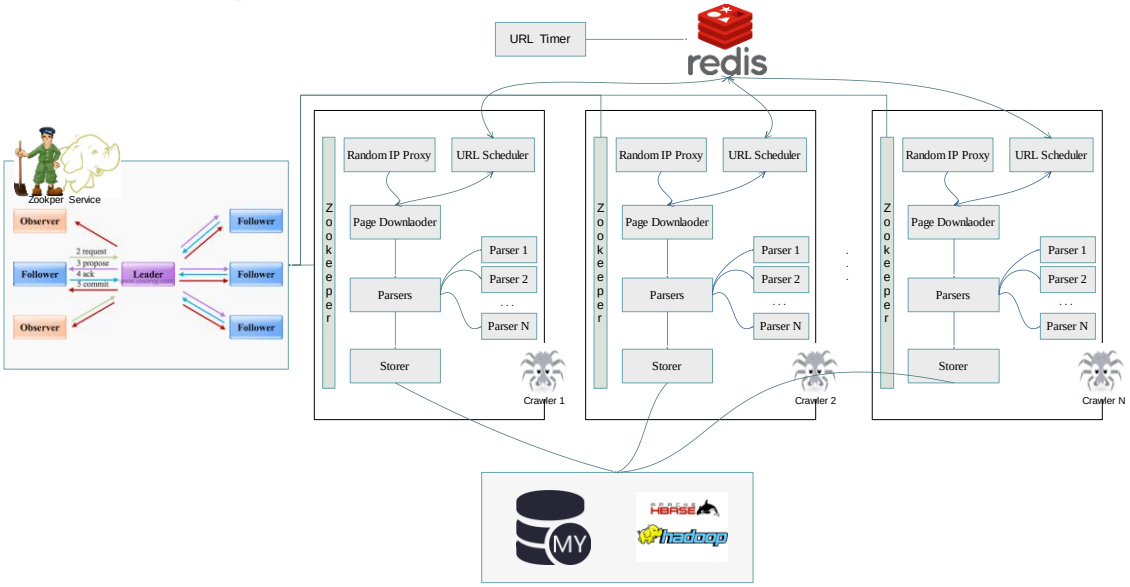

Figure 1. The architecture diagram

Redis is an open source, network-enabled, memory-based and persistent log-based, Key-Value database [5] and Redis stores URL list data in this system. Zookeeper is a distributed, open source distributed application coordination service [6]. In this system, Zookeeper is used to monitor and alarm the crawler system. HBase is a distributed, column-oriented open source database [7]. It uses crawled web page information in this system.

The requirement of distributed crawler for deployment is high in hardware memory. It needs to run cloud server, database server and Web server. On the wave server of system service deployment, enabling these services will occupy more memory, so the requirement for hardware and memory is relatively high. Because of the extensibility of OpenStack platform, more servers can be added in future use.

Table 1. Environment Configuration

	Hardware	Software
Server environment	CPU: Xeon E5-2407 Memory: 16GB Disk: 1TB	OS: Center OS 6.0 Cloud: OpenStack Datbase: MySQL Web Server: Tomcat
Development Environment	CPU: Intel Core 3 Memory: 8GB	OS: Window 10 JDK 1.8 Eclipse for Java EE

Distributed system is composed of a group of computer nodes that communicate through the network and coordinate their work in order to accomplish common tasks. The emergence of distributed systems is to use inexpensive, ordinary machines to complete computing and storage tasks that a single computer can not complete. The goal is to use more machines to process more data.

4. Implementation

4.1 IP Proxy

The main purpose of adding random IP proxy is anti-crawler, so if there is an IP proxy library and different proxies can be used randomly when building http client, it will be very helpful for us to anti-crawler. Then in the tool class that builds the HTTP client, when the tool class is first used, these proxy IP will be loaded into memory and into a HashMap in Java. The implementation of IP proxy is shown in Figure 3-7.

```
// IP地址代理库Map
private static Map<String, Integer> IPProxyRepository = new HashMap<>();
private static String[] keysArray = null;   // keysArray是为了方便生成随机的代理对象

/**
     * 初次使用时使用静态代码块将IP代理库加载进set中
     */
static {
    InputStream in = HttpUtil.class.getClassLoader().getResourceAsStream("IPProxyRepository
    // 构建缓冲流对象
    InputStreamReader isr = new InputStreamReader(in);
    BufferedReader bfr = new BufferedReader(isr);
    String line = null;
    try {
        // 循环读每一行，添加进map中
        while ((line = bfr.readLine()) != null) {
            String[] split = line.split(":");   // 以:作为分隔符，即文本中的数据格式应为192.16
            String host = split[0];
            int port = Integer.valueOf(split[1]);
            IPProxyRepository.put(host, port);
        }
        Set<String> keys = IPProxyRepository.keySet();
        keysArray = keys.toArray(new String[keys.size()]);  // keysArray是为了方便生成随机的代
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

Figure 2. IP proxy

4.2 URL Scheduler

The URL scheduling system is the bridge and key to realize the distributed crawler system. It is through the use of the URL scheduling system that the whole crawler system can obtain the URL more efficiently (Redis as storage) randomly and realize the distributed of the whole system.

4.3 Web Downloader

The web Downloader is used to download data in web pages, mainly based on the following interface development: as shown in Figure 3.

```
/**
 * 网页数据下载
 */
public interface IDownload {
    /**
     * 下载给定url的网页数据
     * @param url
     * @return
     */
    public Page download(String url);
}
```

Figure 3. The interface of web Downloader

4.4 Page Parser

The web parser is to parse the data we are interested in from the downloaded Web pages and save it to an object for further processing by data storage to save it in different persistent warehouses.

5. Conclusion

With the rapid development of the Internet and explosive growth of data, web crawler technology, as the data source support of other applications such as search engine and public opinion analysis, has attracted more and more attention. In order to improve the efficiency of large-scale data capture, this paper designs and implements a distributed crawler system. Through IP proxy library, adding random IP can deal with anti-crawler. Through efficient Redis storage, the efficiency of URL scheduling is

improved. When the crawler system is started, the program will start a Zookeeper client to register its node information with Zookeeper, and realize the dynamic scheduling of the crawler through Zookeeper. Using Hbase and MySQL to store data ensures data integrity.

Acknowledgements

This work was financially supported by practical innovation project for college students of Jiangsu Maritime Institute, the higher vocational scientific research subject of computer national computer basic education institute (2018-AFCEC-265), the funding of Jiangsu QingLan outstanding young teacher project and the funding of professional leader high level study project for Jiangsu higher vocational institute teachers .

References

[1] Ahmadi-Abkenari, Fatemeh. "An architecture for a focused trend parallel Web crawler with the application of clickstream analysis."Information Sciences 184.1(2012):266-281.
[2] Liu, H., and E. Milios. "PROBABILISTIC MODELS FOR FOCUSED WEB CRAWLING." Computational Intelligence 28.3(2012):16-22.
[3] Feng, Zhao, et al. "SmartCrawler: A Two-Stage Crawler for Efficiently Harvesting Deep-Web Interfaces." IEEE Transactions on Services Computing 9.4(2016):608-620.
[4] Tan, Qingzhao, and P. Mitra. "Clustering-based incremental web crawling." Acm Transactions on Information Systems 28.4(2010):1-27.
[5] Zhu, Jin, et al. "Research of Lightweight Vector Geographic Data Management Based on Main Memory Database Redis." Journal of Geo-Information Science 16.2(2014):165-172.
[6] Goel, Lipika Bose, and R. Majumdar. "Handling mutual exclusion in a distributed application through Zookeeper." Computer Engineering & Applications 2015.
[7] Pandagale, Ashwini A., and A. R. Surve. "Hadoop-HBase for finding association rules using Apriori MapReduce algorithm." IEEE International Conference on Recent Trends in Electronics 2017.