A URL Scheduling Algorithm in Parallel Crawler System

Wang Dazhen[1,2] Wan Fang[1] Peng Yan[1]

(1.Department of Computer Science, Hubei University of Technology, Wuhan, P.R.C 430068)
(2.Department of Information Management, Wuhan University, Wuhan, P.R.C 430074)

Abstract: In this paper, we analysis parallel Crawler fetching Model in the distributed architecture, described function of every component and some rules which crawlers must obey when they fetch the web simultaneously. And, we designed a Hash URL Scheduling based algorithm.

Key Word: Distributed Crawler, Hash Algorithm, URL Scheduling

1 Preface

A distributed Crawler indicates that multiple Crawlers gather the web information in parallel mode from the Internet [1]. In the distributed Crawler, a very important problem is how every Crawler performs its own task according to a certain rule and cooperates each other. That is, how to search the URLs in a range scale and execute the page process in order to make sure that the URLs found by every Crawler couldn't be reduplicated [2][3]. In this paper, we discussed the Hash method and proposed an improved Hash Algorithm that will be applied in the distributed Crawler system to manage the URLs assignment and every Crawler's task and ensure the load balance in the distributed system.

2 Architecture of parallel Crawler

2.1 General Architecture

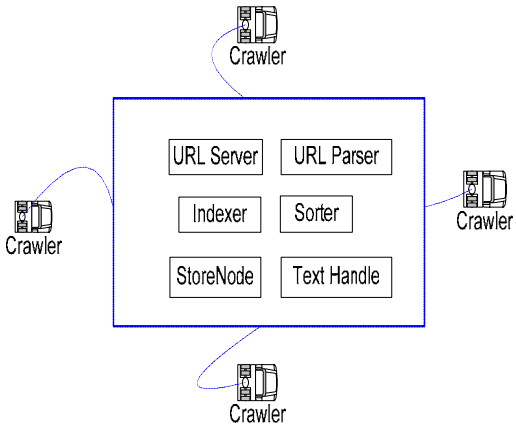

Figure 1: General architecture of parallel crawler

In Figure 1, we illustrate the general architecture of a parallel crawler. The functions for the module are described as follows [4]:

- URL Server: Provide the URLs list to the Crawlers.
- URL Parser:
  a. Read the anchors file and then convert the relative URLs to the absolute URLs and to the FileNum further.
  b. Set the index for the anchors text and creates the relationship between the index and the FileNum, which is aimed by the index.
  c. Establish the link database consisted of the FileNum pairs. Calculate the PageRank value for all the documents. Send the barrels classified by the FileNum to the sorter, and then create the inverted index by classifying the wordID.
- Crawler: The web pages are snatched by several distributed crawlers and then are transferred to the store server: StoreNode. As soon as the URL is parsed out, the URL is assigned a FileNum.
- StoreNode: Compress the web pages and save them as a document into the repository.
- Indexer & Sorter:
  a. The Indexer reads the document from the repository and then decompresses and parses the document.
  b. The Indexer parses all the links in the web pages, and then save the relational and important information into the anchor file that includes enough information. Based on the information, we can know the information of every hypertext link-in or hypertext link-out node for every link and the link text.

c. The Sorter provides the FileNum and offset list and create the converted index.

2.2 Crawler Tasks, Traveling region and Partition Rules

In this system, we introduce the distributed and parallel Crawler fetching model illustrated in Figure 2. As shown in the Figure 2, there is multiple Crawler process units that they are independent in physical but also cooperation. Each process unit performs the basic page downloading and save the downloaded web pages into the local storage. And then, the URL link will be parsed out from the downloaded web pages and will be exchanged in a global range according to the way of the parallel downloading. In this distributed Crawler system, there is a host acted as the Coordinator and it will communicate with all the hosts running the Crawler peer to peer, send the control command (via the RMI protocol) and transfer the URL (via the Socket). It is not allowed that each host running the Crawler communicates with each other directly, the Coordinator must complete all the network communication, collecting the uploaded URL list from each host and performing the URL route, which is the one of the primary tasks of the Coordinator.

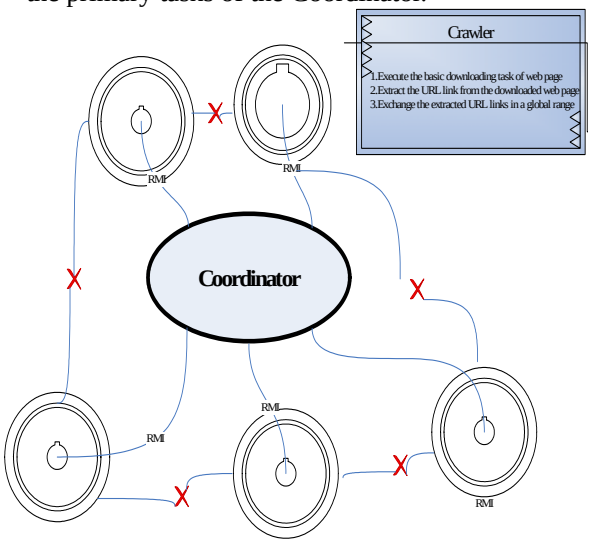

Figure 2: Distributed and parallel crawler fetching model

The distributed web crawler divided the downloading job for a whole web into some sub regions. Each node in this distributed system performs the whole search and downloading jobs for one of the sub regions. Supposed that the P represents the whole web-downloading region (we can see it as all the URLs in the web site). Therefore, based on the distributed system, we need to find a mapped function named f, which will meet:

$$p_i = f(P); P = \sum_{i=1}^{n} p_i \qquad (1)$$

$$p = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \overline{p})^2} \approx 0 \qquad (2)$$

The formula (1) indicates that the mapped function can divided the W into some sub regions and the mix set of these sub regions is just the whole web region. The mapped function also should ensure that there is not any intersection among any sub regions, which means that any one URL in the web can and just can be processed by one of the nodes of the distributed crawler. Single-value map ensures that there is not any downloading and index repeated, which will save the network bandwidth and storage resource. The formula (2) requires that the standard deviation among each sub regions of the web downloading should be close to zero. That is, any multiple accesses are impossible.

3 Basic methods for URL assignment

The common algorithm for the URL assignment is based on a traversing graph algorithm with deep first search method, which will settle some relational websites for the searching for each crawler. If the searching is out of the range, the URL will be ignored. The algorithm is used when searching the assigned website: W, so all sub URLs belong to website W will be found- first, by calling the host function to get the host name, and then make sure whether the host is same with the host where the website W resides or not. By this way, we can know whether the URL belongs to the website W or not.

The algorithm is described as follows:

```
Proc1:   name=GetHostNameOfUrl()
Proc2:   if(name=Host_name)
             If(URL exist in the ListUrl)
                 Save the Page
             Else
                {
                 Save the URL
                 Save the Page
                }
         Else
             Abandon the URL
```

By using the above algorithm, we can filter the URLs out and only extract the URLs belonged to the special website and then save and parse them. But, this

way has not already been fit in the distributed crawler architecture because the assignment for each page should perform the URL parsing prior to its arrival to the computer where the crawler is running. Otherwise, the rules specified by the formulas (1) and (2) will not be met.

4 A URL scheduling algorithm based on Hash

The main idea of Hash method is to decide the store address according to the key code value of the node. That is, referring to the key code value K as a independent variable, figuring out the corresponding value of the function via a definite function relationship h(K) (named Hash function). And then we can interpret the value as the store address for the node and save the node into this store unit. When retrieving, we can use the same way to figure out the address and then read the node information from the corresponding store unit.

We can convert a URL to an integer by using the Hash algorithm and then locate the integer on some crawler via some user-defined method. So we will perform the URL assignment.

As a sample, we put forward the following algorithm based on a parallel system with n crawler nodes.

$$Key_i = \sum_{i=1}^{t} Transfer(host(URL_t)) \bmod n$$

The host function is used to get the host address for each URL. For example, we will get http://www.hbut.edu.cn/ from http://jsjxy.hbut.edu.cn by the host function. The host function retains the host name of the URL, which makes sure that all documents in a host computer will just be downloaded and saved by a crawler. We also will keep a mapped table between the common characters and numbers. Furthermore, the design way of the mapped table can be expanded further for the purpose of being able to execute some simple type conversions just like that illustrated in table 1:

Character	Number
a	1
b	2
c	3
d	4
……	……

Table 1: Characters-numbers mapped table used in the transfer function

The transfer function gets the corresponding integer value for each character in the string from the host function and calculates the sum for all the integer values. We can get the key of the URL using modular arithmetic with modulus n that is the count of the crawlers. That is, the key is equal to the result that the sum value are divided by n. Suppose that there are 20 sub nodes, the key will only be a natural number from 0 to 19. Before running the system, each crawler will register to the coordinator and be assigned an ID that is an integer and will increase together with the registered crawler increasing. During the system is running. The integer is just used as the parameter that will match to the result of the hash function. If both are same, the URL will be downloaded. Or else, the URL will be saved and sent to the coordinator via the Internet. The match way is just like that illustrated below:

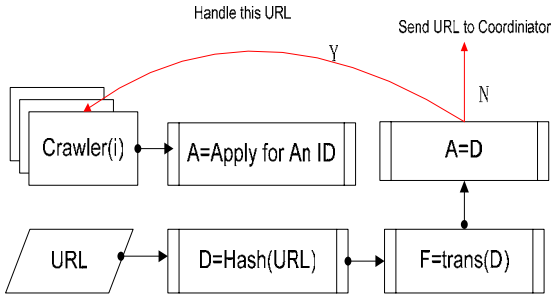

Figure 3: Hash dynamic match URL flow diagram

The hash algorithm accords with the two conditions mentioned above well enough, specially, the formula (2). By the experiment below, it will prove that the hash algorithm can ensure that each crawler can collaborate each other well and keep the system load balance persistence and uniformity, so that the executing ability of the system will be improved in whole hog.

5 Experiment

For a distributed crawler system, when we are in the testing, the performance factor we have to think mainly is the frequency of error searching when fetching. That is, how many times the same web page in the same website is found by different crawlers. The algorithm for URL assignment in the Nutch is based on the distributed system. Therefore, based on the APIs of Lucene and the system architecture of Nutch, we improved the algorithm for URL assignment. Suppose that the initial Nutch algorithm is named as original algorithm, our improved algorithm as improved algorithm, we can create a experiment platform to carry through the comparison – mainly for the frequency comparison of error searching when fetching. We use five computers that run under the

Linux operation system as the crawlers, and then we select a new computer as the coordinator again. The experiment result is illustrated in the figure 4. We can draw a conclusion from the figure, in a non-large-scale data testing, the frequency of error searching based on the improved algorithm is obviously less than that based on the original algorithm [5].

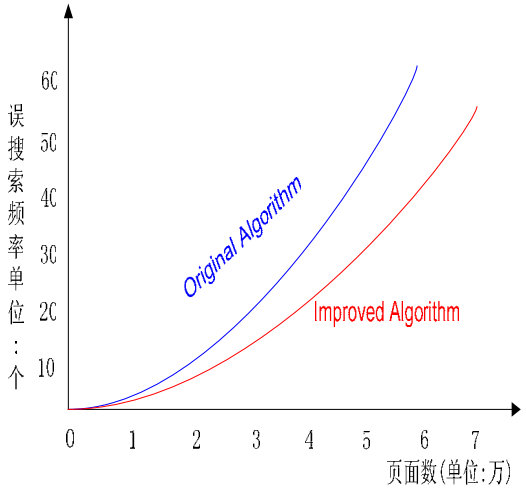

Figure 4: Performance comparison based on the distributed crawler algorithm

6 Conclusion

In this paper we study the parallel crawler-fetching model in the distributed architecture and the basic rules that have to follow in order to proportion the system load balance when each crawler is executing the parallel searching. Finally, we bring forward the URL scheduling algorithm based on the hash method. By tested in the experiment, the improved algorithm can improve the system parallel performance. But, there is still a long way to walk if we want to improve the performance on a large-scale data testing.

References:

1. Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S.Roselli, and Randolph Y. Wang. Serverless network file systems. In *Proc. of SOSP*, 1995.

2. Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proc. of VLDB Conf.*, 2000.

3. Pingfan Yan, Changshui Zhang. “Artificial Neural Networks and Evolutionary Computation”. Beijing, Tsinghua University Press 2002.

4. LI Xiao-Ming, FENG Wang-Sen. Two Effective Functions on Hashing URL. Journal of Software,2004,15(2): 1792184.

5. Ming Zhang and Xiao-dan Liu. Data Structures and Algorithm Analysis. Beijing, Publishing House of Electronics Industry, 1995.