URL Assignment Algorithm of Crawler in Distributed System Based on Hash

Yuan Wan Hengqing Tong

Department of Mathematics, Wuhan University of Technology,

wanwanyuan@sina.com

Abstract

Web crawlers are the key component of services running on Internet and providing searching and indexing support for the entire Web, for corporate Intranets and large portal sites. More recently, crawlers have also been used as tools to conduct focused Web searches and to gather data about the characteristics of the WWW.

In this paper, we research on the gathering model of crawler in the distributed circumstance. We describe the function of every module and establish some rules which crawlers must follow to maintain the equilibrium load and robustness of system when they are searching on the web simultaneously. Then we design and implement a new URL assignment algorithm based on hash for partitioning the domain to crawl, and more in general discuss the complete decentralization of every task.

Key Words: distributed crawler; URL assignment; gathering model; Hash algorithm

1. Introduction

A crawler is a program that traverses the hypertext structure of the Web automatically, starting from an initial hyper-document and recursively retrieving all documents accessible from that document. Web crawlers are also referred to as robots, or wanderers. Due to the explosive growth in web pages, web search engines are becoming increasingly important as the primary means of locating relevant information [1,3,4]. We have realized that centralized crawlers are not any longer sufficient to crawl meaningful portions of the web. Indeed, it has been recognized that 'as the size of the web grows, it becomes imperative to parallelize the crawling process, in order to finish downloading pages in a reasonable amount of time [2] '.

The distributed crawler means several crawlers fetchweb information simultaneously. A very important problem is how to create some rules which every crawler follows to do their own work, that is, how to search URL in a certain range and after being done page setup, the URLs fetched by each crawler are not repeated.

There are some algorithms for crawling URL, however, they are not under the distributed circumstance. In this paper, we discuss Hash method and present an improved Hash method used by the distributed crawler system to assign URL and decentralize the different work of crawler to guarantee the balanced load of this system. Experiment shows that this method effectively increased the performance of the web crawler.

The remaining of the paper is organized as follows: Section 2 presents distributed crawler system architecture. Section 3 describes the common URL assignment algorithm and section 4 presents the design of URL assignment algorithm in distributed circumstance based on Hash. Section 5 presents preliminary experimental results and finally, section 6 offers summary and our conclusions.

2. Distributed crawler system architecture

In this section we first give a detailed description of the architecture of distributed crawler system. Then we describe the mission of every crawler and partition rules for traverse region.

2.1 Architecture of distributed crawler system

We now give the architecture of distributed crawler system, as figure 1 shown. The system contains 6 main components: URL Server, URL Parser, crawlers, StoreNode, Indexer and Sorter. All of these components, plus the crawling application, can run on different machines (and operating systems) and can be replicated in order to obtain better system performance. Functions of these components are explained as follows:

- URL Server: The users requests are sent to URL sever, then URL server affords URL queue to Crawlers.
- URL Parser: This is an analysis module of this system responsible for analyzing and transforming URL, it has several functions.

  Func1: Read and link descriptive anchors file, then transform the relative URL to absolute URL and then to FileNum.

  Func2: Compile index for linking the descriptive text and create the association to FileNum the index points to.

Func3: Establish the link database composed by FileNum pairs to calculate the values of PageRank of all files. Send the sorter barrels which are classified by FileNum, then create inverted index by wordID.

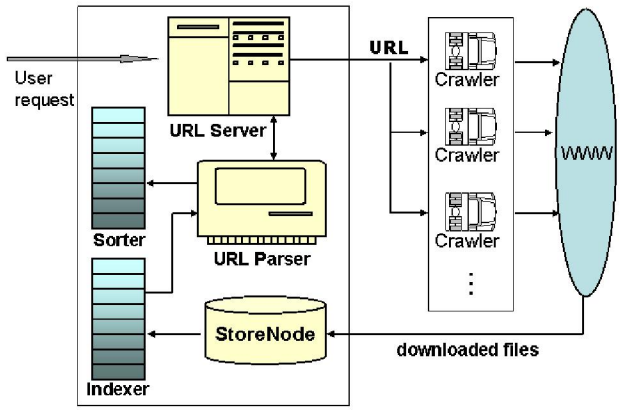

Figure 1: Architecture of the distributed Crawler

- Crawler: There are several distributed crawlers fetching the requested web pages. The fetched pages are sent to memory server, StoreNode. When URL is extracted out from web page, it will be assigned a FileNum.
- StoreNode: It takes charge of compressing the downloaded web pages and saving them to the knowledge warehouse, repository.
- Indexer and Sorter:

  Func1: After web pages are downloaded in repository, indexer reads these files, then decompresses and analyzes them.

  Func2: Indexer also analyzes all the links in the web pages and save the important information to link descriptive files—anchors, which contains so much information to judge the link-in and link-out node of every link and linked file.

  Func3: Sorter presents FileNum and the offset table and creates inverted index.

2.2 Mission of crawler and partition rules for traverse region

In this distributed system, we adopted gathering model of parallel crawler, as the figure 2 shown. There are several processing units which are physically independent but cooperative to each others [6,7]. They carry out elementary downloading of web pages and save the downloaded web pages to local memory. Then they extract URL links from the downloaded web pages and make exchanges of the URL links wholly-scoped in the way of parallel download. This distributed crawler system has a host computer as Coordinator, taking charge of peer-to-peer communication between all other host computers, sending control demand (RMI protocol) and transmitting URL(Socket protocol)[8]. Every crawler, however, can not communicate to others directly, that is, all the communication must depend on Coordinator. Getting together the URL list which is uploaded by each host and achieving routing URL are the main tasks of Coordinator.

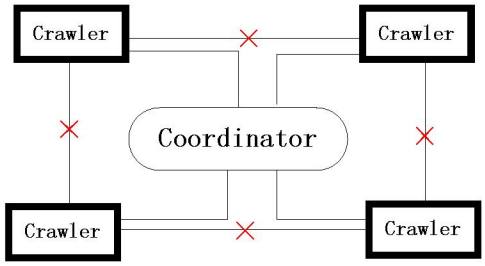

Fig.2 Gathering model of distributed parallel Crawler

Distributed web crawler divides work of downloading the whole web into several subsets. Each node in this system takes charge of all the traversing and downloading of one subset. We use P to denote the whole download region—all the URLs on web.

Reasonable partition of the region P is the assurance of high performance of the distributed system. Each partition corresponds to a mapping function f, and then which function is the best one? Here, we establish two rules that mapping function f should satisfy:

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

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

Formula (1) shows that the mapping function can divide P into several subregions which union is just the whole web region. Mapping function should ensure that there are no cross unions between any subregion, and this means any URL on web can be only processed by one node on distributed crawler. Single-valued mapping ensures nonexistence of repeated download and index, thus saving web bandwidth and storage recourse.

Formula (2) explains that standard deviation d between each subregion of the region P is very close to 0, thus eliminating the possibility of repeated accessing.

3. Basic URL assignment algorithm

General URL assignment algorithms are graph traverse algorithm based on Depth-First algorithm [11]. Such algorithms assign an appointed website for each crawler to search URL. Via getting the function of host name, crawler can judge whether this host is same to that of website W, then it can judge whether this URL belongs to the appointed website. If one URL exceeds the range of

this appointed website, then this URL would be discarded. After searching on the appointed website, each crawler can find out all subURLs on this website.

Algorithm is as follows:

```
Step 1: name=GetHostNameofUrl( )
Step 2: if(name=Host_name)
            If(URL exist in the ListUrl)
              Save the Page
            Else
              {
              Save the URL
              Save the Page
              }
        Else
        Abandon the URL
```

This algorithm helps us filter useless URLs, and only save and analyze those URLs belong to this website. However, this method is not suitable for the distributed crawler system because the assignation of website should be done URL analysis before the crawlers' searching, or it cannot satisfy the rules (1) and (2) that defined in section 2.2.

4. URL assignment algorithm based on hash

In this section, we present an improved URL assignment algorithm. We introduce hash method first, and then the URL assignment method based on hash.

The main idea of hash method is using the key value of each node to determine its saving address. The key value K is considered as an independent variable. By using a function $h(K)$(called hash function), each K corresponds to a value which is explained the saving address of the node. Then save node in this saving unit. When searching for node, we just use the same method to calculate address and then fetch the needing node in the corresponding unit.

We use hash algorithm to transform one URL to an integer, then locate this integer onto one crawler in a customized way, thus the URL assignment is accomplished.

Take a parallel system with n crawlers as the example, the designed algorithm is

$$Key_i = \left(\sum_{i=1}^{l} \text{Transfer}(\text{host}(URL_i)) \right) \bmod n$$

Function host extracts the host address of each URL. For example, the web page address http://jsjxy.whut.edu.cn corresponds to http://www.whut.com. Function host reserves the host name, and then all files in one host are downloaded and saved by only one crawler. We reserve a corresponding table of common-used characters and numbers. This table can be expanded further to do simple type transform, as Table 1 shown.

Character	Number
a	1
b	2
c	3
d	4
.....	.....
z	26
.....	...

Table1: Transform example of characters to numbers by transfer function

Transfer function accumulates the integers corresponding to each character of character cluster got from host function. The accumulation takes *mod* to the number of crawler n, then we get the key of this URL. If there are 20 subnodes, then the values of key can be any integer from 0 to 19.

Before running the system, each crawler should register on Coordinator to get an ID—an integer that increases as the increasing number of registered crawler. In the process of running, this integer is the parameter to match the result from hash function. If they are equal, then download this URL, or this URL will be saved and sent to Coordinator. The matching way is shown as figure 3:

This Hash algorithm satisfies the two conditions we presented in Section 2, especially the formula (2). Besides this, the experiment in Section 5 will demonstrate that this algorithm ensures preferable cooperation and decentralization of work of each crawler, thus it makes the persistent equilibrium loading of the distributed system with low cost and high performance.

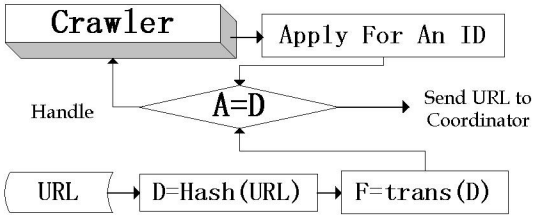

Fig.3: Flow chart of Hash matching URL

5. Experimental results

We now finally present some preliminary experimental results. A detailed analysis of performance and scaling behavior is beyond the scope of this paper, and would require a fast simulation test bed, since it would not be possible to do such a study with our current Internet connection or on real web sites.

In our experiment to test this distributed crawler system, the main-considered performance index is the

frequency of mistake fetching, that is, the frequency of the web pages in the same website which are fetched by several different crawlers. Nutch's URL assignment algorithm [12]is also based on distributed system, so we choose API of Lucene and the structure of Nutch system to compare with the approached URL assignment algorithm. We call Nutch algorithm original algorithm. We use 5 Linux machines as crawlers and one as coordinator to establish experiment platform by which to compare these two algorithms, mainly about the frequency of fetching mistake searching. Result of the experiment is as figure 4 shown. We can see, when the scale is not so large, the frequency of mistake searching of approached algorithm is obviously lower than that of the original algorithm.

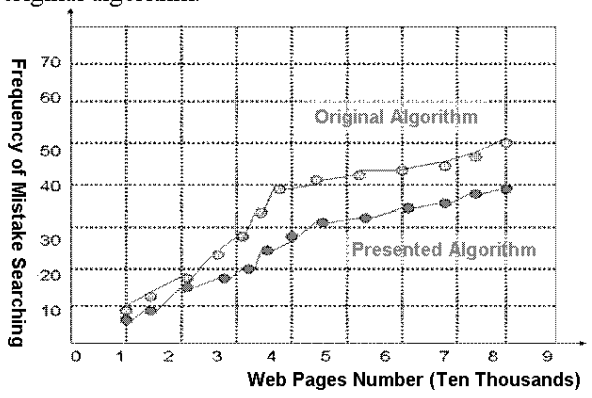

Fig.4: Performance comparison of distributed crawler algorithm

6. Conclusion and summery

In this paper, we have described the architecture and gathering model of the distributed system and the basic rules that make the system balanced in loading when crawlers are searching parallelly. We then presented URL assignment algorithm of our distributed crawling system, and carried out some preliminary experiments.

Experiment has shown that this algorithm has preferable parallel operative performance.

There are obviously many improvements to the system that can be made. A major open issue for future work is a detailed study of the scalability of the system and the behavior of its components. This could probably be best done by setting up a simulation testbed, consisting of several workstations, that simulates the web using either artificially generated pages or a stored partial snapshot of the web.

7. References

[1] Paolo Boldi, Bruno Codenotti,, "UbiCrawler: a scalable fully distributed Web crawler", *SOFTWARE—PRACTICE AND EXPERIENCE* , 2004 34:711–726

[2] WU Lihui, WANG Bin, and YU Zhihua,"Design and Realization of a General Web Crawler, *Computer Engineering February* 2005.2 123-124

[3] D.Zeinalipour-Yazti and M. Dikaiakos, "Design and Implementation of a Distributed Crawler and Filtering Processor"A. Halevy and A. Gal (Eds.): NGITS 2002, LNCS 2382, pp. 58 – 74, 2002.

[4] A. Arasu, J. Cho, and H. Garcia-Molina, "Searching the web". *ACM Transactions on Internet Technologies*,1(1), June 2001.

[5] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, andD.Weitz. "Approximating aggregate queries about web pages via random walks". *Proc. of 26th Int. Conf. on Very Large Data Bases*, September 2000.

[6] Boldi P, Codenotti B, and Santini M, "Ubicrawler: Scalability and fault-tolerance issues", Poster Proceedings of the11th International World Wide Web Conference, Honolulu, HI, 2002. ACM Press: New York, 2002.

[7] Najork M, Wiener JL. "Breadth-first search crawling yields high-quality pages". *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, China, 2001. ACM Press: New York, 2001.

[8] Cho J, Garcia and Molina H. "Parallel crawlers." *Proceedings of the 11th International World Wide Web Conference*, 2002. ACM Press: New York, 2002

[9] Junghoo Cho and Hector Garcia-Molina. "The evolution of the web and implications for an incremental crawler". I*n Proc. of VLDB Conf*, 2000.

[10] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. "On near-uniform URL sampling ". In *Proc. of the 9th Int. World Wide Web Conference*, May 2000

[11] S. Raghavan and H. Garcia-Molina, "Crawling the hidden Web". In Proc. of 27th Int. Conf. on Very Large Data Bases, Sept. 2001

[12] LI Xiao-Ming, and FENG Wang-Sen. "Two Effective Functions on Hashing URL" *Journal of Software*, 2004,15(2) 179-184