Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, 10-13 July, 2011

A PEER-TO-PEER BASED PASSIVE WEB CRAWLING SYSTEM

QING-CAI CHEN, XIAO-HONG YANG, XIAO-LONG WANG

Department of Computer Science and Technology
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China
E-MAIL: qingcai.chen@gmail.com, yxh2008@gmail.com, wangxl@insun.hit.edu.cn

Abstract:

Though the commercial success of search engines and large scale web page crawlers, the problems of page refresh, new URL discovering, large file downloading, distributed multimedia content feature extracting and indexing etc. are still open. The independent working behavior of each crawler makes it very hard to seek solutions for all these problems under the classical web crawler architecture. To address these problems, this paper proposes an innovative client/server based web crawling system. This system consists of a crawler server and a crawler client which work in the search engine and website end respectively. The crawler server registers itself to the client and joins into a temporary peer-to-peer network to cooperate and share downloaded data with other crawler servers. Different from the classical crawlers, the data downloading procedure is initialized by a client. So for the crawler server, this is a "passive" web crawling system. The main benefits of this system include the capability of timely management web changes for a crawler, the saving of website bandwidth resources, the capability of downloading large files or multimedia content features, and the capability of protection intellectual properties while indexing and searching the content. Our experiments taken on a simulation system show its efficiency and practicability for the real Internet environments.

Keywords:

Passive web crawler; peer-to-peer network; search engine

1. Introduction

Web crawlers, also known as web spiders or robots, traverse the Internet to discover and collect new web sites, and have been widely used by most search engines. The great demands from industry promote the rapid growth and remarkable attention of the research on web crawler.

The basic communications between crawlers and web sites in classical search engine architecture for web pages download are showed in Figure 1. The crawler first sends an HTTP GET request to a website host with the URL of the webpage. If this URL is valid, then the web page content is transferred to the crawler. The websites don't need to take additional action to cooperate with crawlers except the actions of limiting the visiting frequency from the same crawler to avoid DOS attacks. For the crawler, if it doesn't want to be treated like an attacker, it should follow the protocol provided in "Robots.txt" and limit its visiting frequency on the same website.

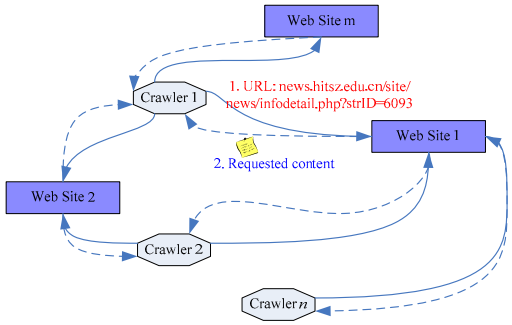

Figure 1. Basic Communications of Classical Web Crawler

Showed in Figure 2, a crawler usually follows the below steps to download web pages: 1) Build a seed URL queue; 2) Extract one URL from the queue and download corresponding web page through HTTP protocol; 3) Analyze downloaded web page, extract URLs contained in this web page and put them into the seed URL queue if they are never appeared in this queue; 4) Repeat steps 2 to 3. The real operation for commercial crawlers is much more complicated. The critical challenges they have to face include how to handle the politeness behavior, URL traps, DNS resolution, URL uniqueness checking, web page duplicate checking, web page refresh and bottleneck of data storage etc. To address these problems, various architectures or techniques have been exploited to design different crawlers such as distributed or scalable architecture [1][2][4], parallel crawler [3], page selection technique [5], URL selection strategy [6], page refresh technique [7][13], URL uniqueness checking and URL list updating algorithm [8], focused crawler [9][10], P2P based

978-1-4577-0308-9/11/$26.00 ©2011 IEEE

crawling method [11] and two-tier architecture [12] etc.

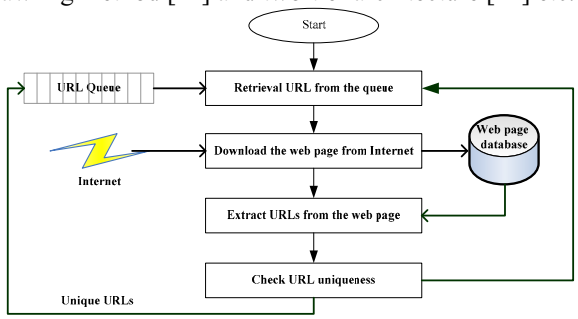

Figure 2. The basic principle of a crawler

Without doubt, many problems mentioned above of classical crawler have been realized, which promotes the remarkable commercial success of classical crawler in search engines. However, it is very difficult to seek the solution to solve all the problems, and enhance the cooperation between crawlers and websites and the cooperation among crawlers under current architecture. To achieve the above goal, we propose an innovative pear-to-pear based passive web crawling system composed of the sever end subsystem running in search engines and the client end subsystem running in a website servers to further enhance the role of websites.

Besides the basic function of downloading web pages from websites, the crawler server is also responsible for registering itself to the client and joining into a temporary peer-to-peer network to cooperate and share downloaded data with other crawler servers. After being successfully registered on a client, the server is waiting for the content updating notification sent from clients. Different from the classical crawlers, the data downloading procedure is initialized by a client. So for the crawler server, this is a "passive" web crawling system. The responsibilities of a crawler client include maintaining the list of registered servers, extracting features for local multimedia contents, monitoring and packing updated contents or their features into formatted files, initialize the content downloading task etc. The main benefits of this system include the capability of timely management web changes for a crawler, the saving of website bandwidth resources, the capability of downloading large files or multimedia content features, and the capability of protection intellectual properties while indexing and searching the content. Our experiments taken on a simulation system show its efficiency and practicability for the real Internet environments.

The remains of this paper are organized as follows. Section 2 describes the basic principle and the detailed architecture of our crawling system. Section 3 analyzes the efficiency and cost of crawler server and crawler client. The simulation and performance evaluation of this passive crawling system is provided in Section 4. Section 5 provides the conclusions and the discussion of future work.

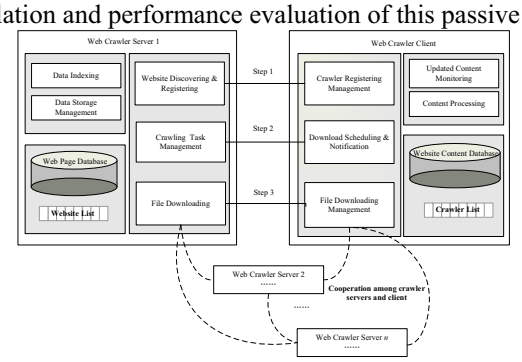

Figure 3. Architecture of P2P based passive web crawling system

2. P2P based passive web crawling system

2.1. Architecture of passive crawling system

Figure 3 shows the top level architecture of the P2P based passive crawling system where the key components include the crawler server running on the search engine end and the crawler client that runs on the website end. Under this passive crawling framework, website masters need to configure and run the client system on their site servers at first. Then the client is waiting for the register of crawler server and maintains an information list for all registered servers. The client also monitors the changing of website contents and packing updated contents through the Updated Content Monitoring and Content Processing components. In fact, by well designed Content Processing components, the client is able to take any necessary and complex tasks on the behalf of webmasters. The only limitation is that all of the client behaviors should follow a standard way, so that the servers can understand the output of their actions.

Table 1 lists the responsibilities of server and client under the passive web crawling framework. The first step of a crawler server is finding websites and registering itself on discovered websites, which is assigned to Website Discovering and Registering component of the server. The second step of web crawling is initiated by crawler client. After the package on a website reaches the preconfigured minimum size, crawler client sends notification message to all registered servers with the downloading time. As shown in Figure 3, File Downloading component of crawler server will communicate with the File Downloading Management component of client and a temporary P2P network is constructed at that time. According to the number and network status of participated servers, crawler client will

segment the file and arrange the file segments for each server to download. The full copy of the file is finally transferred to each server following a P2P protocol.

Table 1. Responsibilities Assignment for Crawler Server and Crawler Client

WIC Server	WIC Client
1. Search new website, 2. Register on the website, 3. Wait and receive the downloading notification, 4. Join into a temporary P2P network composed of by other servers and the client to download content package at the appointed time, 5. Download and share data following a P2P protocol, 6. Maintain downloaded data	1. Receive registering request of servers, 2. Maintain an information list of all registered servers, 3. Monitor the changes of the website contents, 4. Collect and pack updated contents, if necessary, extract features of contents before packing, 5. Send downloading notifications to all registered servers, 6. Construct a temporary P2P network for servers at the arranged time, 7. Provide one copy of the package for updated contents as the seed file for downloading, 8. Segment the seed file and arrange the downloading tasks for all servers

2.2. Crawler server registering

To let a crawler server directly communicate with one or more clients to finish the server registering and other tasks, the website needs to provide crawler servers the information of clients running on it. Just like the "robots.txt" for the classical crawlers, we put a XML file named ``robots.xml'' in the root directory of each website. Information contained in "robots.xml" includes the listening port of each client and its IP address. Crawler servers will directly contact with clients through these information. Other optional but also important information includes the content subjects ("commercial", "education", "information technology" etc.) and types ("audio", "video", "image" etc.) managed by each client. The webmaster can also put any additional server-understandable information in this file.

When a website is discovered by a server, "robots.xml" is downloaded through root directory of this website and is parsed by a standard "robots.xml" parser. If the server is interested in some topics or some types of contents contained in the website, it sends a registering message to the respected clients through the given IP address and listening port. A registering message contains necessary description like the name of the search engine, the domain name, the unique identification number of the crawler server, its IP address and monitoring port. Here, UDP protocol is suggested for the registering. Since UDP is not a reliable way to transfer data, the client should respond to the server as soon as it receives the registering message. The responds should be sent at least twice to increase the possibility of reaching the server end. The responds message will keep as short as possible and just contain the unique identification of the client and registering result of "accepted" or "rejected". A successful register means that the client had put the server into its server list and any update notification will be sent to the server. Otherwise, the server cannot accept update message from the client. If there is no respond received within a given period, the server should send the same registering request again or give up after several times of trying.

2.3. Content updated notification

Before notifying servers about updated content, there are still several problems need to be resolved by the client. The first problem is the arrangement of downloading time. For a simple solution, the webmaster can set a specific time according to the visit status of the website when configure the client. However, to avoid the network traffic, a random delay within half hours is recommended for the client. To keep things simple, the appointed time contained in the notification for all servers are the same. But for the same reason to avoid network traffic on the website, a random delay of several seconds for each server is necessary before they visit the website to get information about the downloading procedure. When the client segments the package and assigns the downloading tasks for servers, the latter kind of delay must be taken into considered.

Another problem that needs to be discussed here is the downloading of history data. Usually the history data of a website is tens or hundreds times of the latest updated contents. By increment packing, the storage and management of history data are not critical problems for the client. However, downloading large amount of data is a resource consuming task. A simple solution for this problem is setting a relative long period for history data downloading. So for a new registered server, it has to wait for several days or weeks to be notified to download all history contents on this website. The waiting time is determined by three factors: the size L of history data, the total number N of new registered servers and the longest waiting time T. If L is smaller, then N and T should be also smaller.

Since a server usually faces to a large amount of websites, providing it the estimated longest waiting time of downloading history data is very helpful for the server end performance optimization. It is also a factor for the server

to judge the status of the website. The left task for the client is to plan the schedule for downloading updated package and history data according to the estimated longest time of each

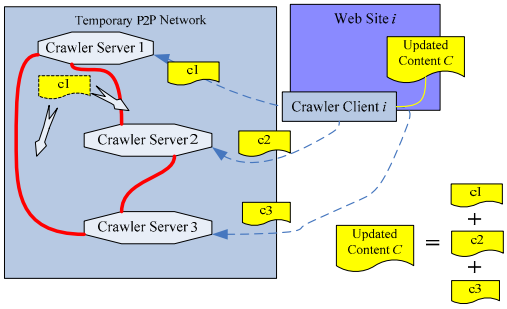

Figure 4. Download updated package via a P2P network

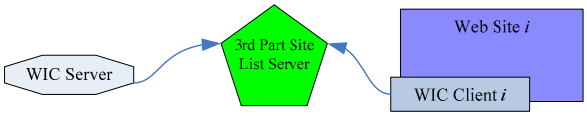

Figure 5. Website Discovering Based on 3rd Part Website List

server and the current status of the website. The notification messages are sent out according to the schedule.

2.4. Download updated content by P2P network

In this paper, the Bittorrent P2P protocol is introduced not only for increasing the average downloading efficiency of crawler server, but also for saving bandwidth resource of websites. Figure 4 shows the procedure of downloading updated content from a website. Under the P2P protocol, the client just provides one copy of the packed file and segments this file into multiple pieces. Each server downloads one or several pieces from the client, and share downloaded pieces with other servers. To increase the data transferring efficiency, all servers that are downloading data are required to share their parts of downloaded data to others. Though it seems to be the most difficult part for promoting the passive crawling architecture, the advantages mentioned above may attract the participation of search engines. Our simulation given in experiment section also shows attractive performance for this technique.

2.5. Website discovering

Considering the relatively fewer amount and the static of websites, to discover a website is easier than that of new webpage. The classical approaches used to discover new webpage or websites through the analysis of downloaded web pages are also applied under the passive crawling architecture. To maximize the efficiency gain of the crawling system, a third part website list service is also an efficient way. As shown in Figure 5, each website is able to register itself on the 3rd part site list server and then the crawler server can easily maintain its website list by communicating with the list server.

Though lots of such websites can provide website address list, most of them lack an automatic way to interact with websites and web crawlers. It causes another problem to existing 3rd part site list servers, i.e., no one provides a relatively complete list for all websites in the Internet since most of the list are manually built. Introducing an automatic site address registering method is not difficult in technique aspect. The most difficult part is the lack of standard interaction protocol, which is out the scale of this paper.

3. Efficiency and costs analysis

In the passive crawling system, most of the computing and bandwidth resource of the servers are consumed by downloading updated content via P2P network. According to the properties of Bittorrent protocol, a server that downloads 1GB data may upload 1 to 10GB data. Compared with the amount of downloaded data, other bandwidth consuming such as the registering information, the data updated notification etc. are ignorable. Except the uploading bandwidth consumed by transferring the copies of content to other servers, there is no more bandwidth consumed than classical crawlers. On the other hand, these bandwidth consuming are saved for websites since they just provide one copy of data to multiple servers rather than one copy for each server. Though the payment for upload bandwidth, the computing resources spent on building connections for each URL, checking update status of each web page, checking uniqueness of each URL and the feature extraction for multimedia contents etc. are all be saved. So it is really a valuable tradeoff for the crawler servers. Our experiments taken on a simulation system also shows the high performance of the passive system.

The costs for website ends are mainly the storage space for packing updated contents and history data and the computing time of clients running on website servers. The former costs are not a big problem since GB or TB scale of storage space is available and enough to store their history web pages or features of multimedia files. Though depending on the implement techniques, the costs of computing resource for monitoring and packing updated contents in the local website servers are usually no more than 5% of the total computing time of the system. But for multimedia contents, feature extraction algorithm may become very complex to increase the searching precision.

So the implementation of the passive crawling system must keep enough flexible to webmasters.

4. Simulation and performance comparison

Since the innovative architecture has not been deployed in real website, this paper constructed a simulation environment to conduct the performance evaluation experiments. The environment is composed of 56 IBM blade servers connected by 10Gbps InfiniBand network and the maximum of just 1Gbps bandwidth is available by IPoIB techniques. The configuration of each server is as follows: dual Intel Xeon E5310 CPUs with 1.6GHz/1066MHz/8MB L2/Quad Core, 4GB Memory and 292GB SAS hard disk. The main objective of this simulation is to evaluate the computational and bandwidth consuming efficiency for both crawler client and crawler server.

The BitTorrent-4.4.0 implemented by Bram Cohen is applied with minor changes. One change is that the memory based pieces storage method is added to increase the overall performance of the data transferring. Another change is the adjustment of the default piece size from 16KB to 128KB. The performance is greatly increased by this adjustment, which shows the importance of piece size optimization for different situations. Considering the real situation of the passive crawling system, we limit the upload bandwidth of each crawler client to be 2MBps (or 16Mbps), but half of total available bandwidth for each crawler server. The download bandwidth is not limited for any crawler server. 46 of these 56 blade servers are running as crawler clients and the remaining 10 as crawler servers. To avoid the effect of hard disk accessing on the overall downloading performance, the experiments of first type store the downloaded pieces of each file in the memory until the full copy of the package is downloaded. Since the maximum memory is 4GB, each package size is limited to be 59MB. In the second type of experiments, the disk accessing is taken into consideration. However, since each piece of a package will be accessed for several times during downloading, the overall performance of the system is restricted by the accessing rate of hard disk.

The result of our first experiment is shown in Figure 6 which reveals that with the increase of clients, the total data transfer rate of servers increases before closing the maximum limitation of bandwidth. Since the upload rate of each client is 16Mbps, the maximum bandwidth of each server is 1000Mbps. The maximum data transfer rate 758.7Mbps is reached when using 36 clients, the efficiency of overall bandwidth is about 0.76 at server end. Figure 7 shows the average CPU occupation rate servers under the same situations as Figure 6. Figure 7 demonstrates that through the Bittrorent protocol, the high rate of data transferring does not require too much CPU resources. This encouraging result points out the latent capabilities of managing large amount of websites by a single crawler server.

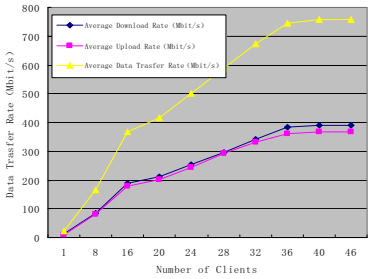

Figure 6. The change of average data transfer rates of servers with the increase of client number

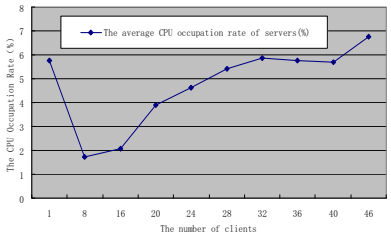

Figure 7. The average CPU occupation rate of servers with the increase of client number

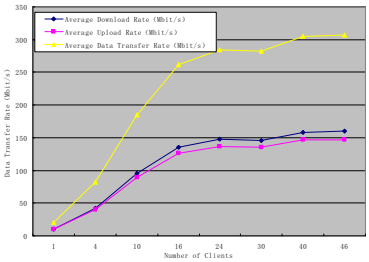

Figure 8. The change of average data transfer rates of servers with the increase of client number (hard disk method)

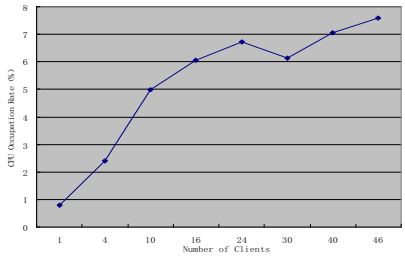

Figure 9. The average CPU occupation rate of servers with the increase of client number (hard disk method)

Based on the hard disk method, Figure 8 shows the same data transfer rate as Figure 6. Figure 9 is the CPU occupation rate under the same configuration as Figure 7 for the hard disk based method. These two figures show that the maximum data transfer rates are greatly reduced while the
CPU occupation rates are not greatly affected by this method. Though this limitation, the hard disk based method are also useful for the crawler servers with no more than 100Mbps of bandwidth to manage a large amount of websites concurrently.

Above experimental results show the high efficiency of the P2P based data transferring method. Since it is one of the key components in the passive web crawling system, these results encourage us to practice this innovative system. Assume that the average change interval of websites in the Internet is one week and the average size of updated package is 50MB for each website, then for the computer servers with the same hardware and bandwidth as the experiment computers, more than 40 websites are able to update concurrently within about 1 minute. That means the crawler server is able to manage the updating of 57,600 websites in one day. In other words, it can manage the changes of about 465,000 websites under the given change rate.

5. Conclusions

This paper proposes an innovative P2P based passive web crawling system which consists of two main components: the servers and the clients. By introducing the client component as a distribution extension of the classical web crawlers, many responsibilities that have been taken by the classical crawlers including webpage discovering, update checking, web page packing and multimedia content feature extraction are now be able to assign to the clients. The experiments taken on the simulation system show that introducing P2P protocol into the passive web crawling system is very efficient and practicable for real situation.

Since the implementation aspects of the passive crawling system are not detail discussed in this paper, here provides some directions for the future research and implementation: 1) the first and also most important task is the standardization of communication messages between servers and clients; 2) the optimization algorithms of downloading updated and history contents for the server and client; 3) optimization of the BitTorrent protocol for downloading large scale data; 4) a flexible and extendable implementation patterns for the client system. Since the clients are running on various kinds of website servers, the context sensitivity capability for the client system is also required. A possible implementation way of this client is introducing multi-agent techniques.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 60703015).

References

[1] S. Brin and L. Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, Proc. of WWW, pp. 107-117, 1998.

[2] Heydon and Najork, “Mercator: A scalable, extensible Web crawler”, World Wide Web, Vol. 2, No. 4, pp. 219-229, Dec. 1999.

[3] Kasom Koht-arsa and Surasak Sanguanpong, “High Performance Large Scale Web Spider Architecture”, Proc. of the Internataional Symposium on Communications and Information Technology, 2002.

[4] Hafri and Djeraba, “High performance crawling system”, Proc. of ACM MIR, pp. 299-306, Oct. 2004.

[5] A. Arasu, J. Cho, H. Garcia-Molina, et al. “Searching the Web”, ACMTrans. on Internet Technology, Vol. 1, No. 1, pp. 2-43, 2001.

[6] Najork and Wiener, “Breadth-first crawling yields high-quality pages”, Proc. of the 10th international conference on World Wide Web Conference, 2001.

[7] J. Cho, H. Garcia-Molina, “The Evolution of the Web and Implications for an Incremental Crawler”, VLDB, pp. 200-209, 2000.

[8] H. T. Lee, D. Leonard, X. M. Wang, and D. Loguinov, “IRLbot: Scaling to 6 Billion Pages and Beyond”, Proc. of WWW, pp. 427-436, 2008.

[9] I. Altingovde, O. ULUSOY, “Exploiting interclass rules for focused crawling”, IEEE Intelligent Systems, Vol.19, No. 6, pp. 66-73, 2004.

[10] P. Srinivasan, F. Menczer, G. Pant, “A General Evaluation Framework for Topical Crawlers”, Information Retrieval, Vol. 8, No. 3, pp. 417-447, 2005.

[11] A. Singh, M. Srivatsa, L. Liu and T Miller, “Apoidea: A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web”, Proc. of SIGIR Workshop on Distributed Information Retrieval, pp. 126-142, Aug. 2003.

[12] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long and K. Shanmugasundaram, “ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval”, Proc. of WebDB, pp. 67-72, Jun. 2003.

[13] J. Cho, H. Garcia-Molina, “Effective page refresh policies for Web crawlers”, ACM Trans. on Database Systems, Vol. 28, No. 4, pp. 390-426, 2003.