International Journal of Security and Its Applications
Vol.9, No.10 (2015), pp.137-146
http://dx.doi.org/10.14257/ijsia.2015.9.10.12

Research on Detection Algorithm of WEB Crawler

Hongyan Zhao

Shandong Yingcai University, Jinan 250014, china,
Zhaohy1127@163.com

Abstract

In the research of Web crawler, the most important things are structure design and solution of the key technologies. Based on the work of other people, we described the structure design of a distribute Web crawler, which including the organization of hardware and module partition of software. In this paper, one PC is utilized as the main node, and other PCs as the common nodes which are connected in LAN. The software architecture included main node design and common node design. Then, we analyzed solutions of the major techniques of the distributed Web crawler, such as how the nodes of the crawler cooperate with each other, how the task is distributed, how to keep the important Web fresh. We have proposed some practicable arithmetic to solve the problems mentioned above. Besides, we implemented a robust, distensible, customized, distributed Web crawler, and anatomized it. At last, we gave the results of two experiments, including common test and a site download test.

Keywords: *Web Crawler, Parallel, Search Engine*

1. Introduction

With rapid development of international network, more and more information is accessible over the Internet [1-3]. So far global webpages have reached 200,000,000, which grow at 73 million each day. Searching out information from such tremendous information ocean is the same difficult as looking for a needle in the ocean. Search engine is one technique developed as to solve the problem. It's an important tool for searching information via the Internet [4-5]. It makes use of various theories and technologies in such fields like information retrieval, artificial intelligence, computer network, distributed processing, database, data mining, digital library, and natural language processing, which pose higher integrity and stronger challenges [6-7]. What's focused here is web crawler, which is key part of search engine [8].

Web crawler, renamed Robots, Spiders and Wanderers appeared almost simultaneously with network. The first web crawler was Wanderer developed by Matthew Gray in 1993. However at that time information scale on the Internet was much smaller than now [9-11]. No papers investigated about the technology for dealing with enormous web information which is encountered at present. In the back-end of each search engine, different web crawlers are working. For the reason of competition, the design of those web crawlers is not open [12].

Some big companies home and abroad have developed accomplished solutions for large-scale web crawlers and have put into use. But, those large-scale search engines can only provide general users with common rather than custom search services, impossible to consider diversified requirements of different users. Standalone web crawlers can hardly meet that in many cases. Owing to flexible customization and incomparable information collection speed and scale against standalone web crawlers, mid-size web crawlers fulfill people's increasing user-oriented demands for web information. Hence in the paper, we concern mainly about one mid-size high quality web crawler, which becomes perfect in every aspect like robustness, scalability and efficiency.

2. Structural Design of Distributed Web Crawler

The distributed web crawler is composed of several PC machines, one of which is controlling node and others are crawling nodes. The former is responsible for maintaining information of all nodes and synchronizing such information, and executing adding and deleting of nodes in controlling node. It is shown in Figure1.

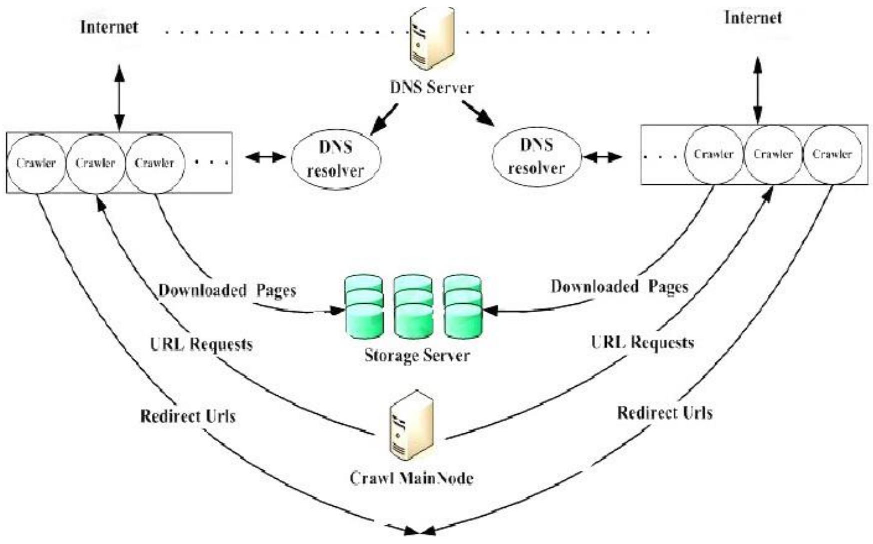

Figure 1. Distribute Web Crawler Frame Design

The design structure of a single node can be divided into five modules as shown in Figure 2.

(1)URL distribution module
(2) Node communication module
(3)URL analysis module
(4) Download module
(5) Web analysis module

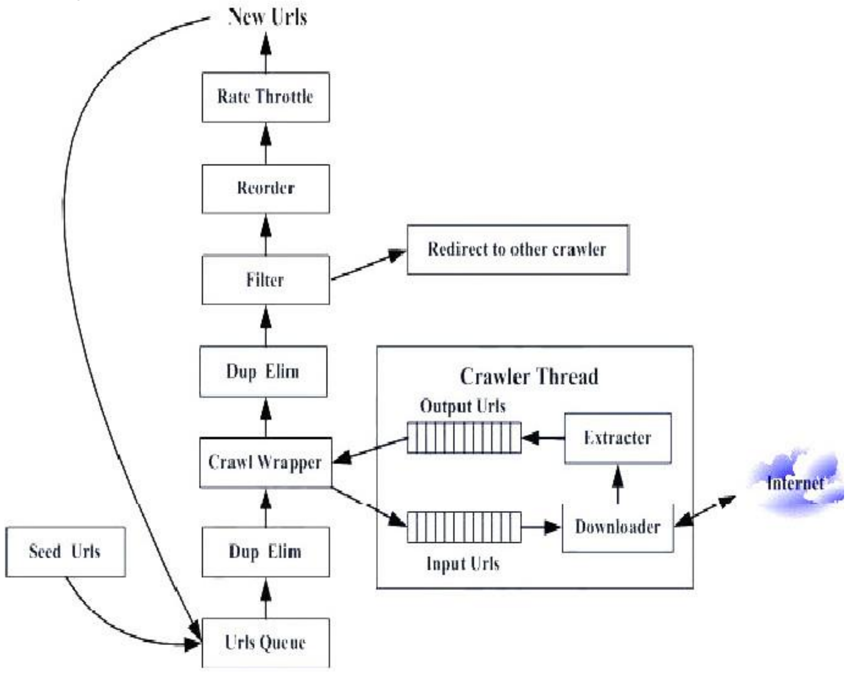

Figure 2. Crawler Node Structure Design

2.1 URL Distribution Module

The main task of this module is to coordinate the work of each node and assign tasks to different nodes, ensure that all nodes of the work did not repeat, and can add, delete nodes.

2.2Node Communication Module

Node communication module is responsible for the communication between nodes, in addition to the collector to collect web pages directly with internet interaction, other time all network communications are to complete by the communicator.
Such design, there are two benefits:
(1) Other modules only need to pay attention to their own strategy, rather than concerned about the details of the specific communication;
(2) The communication module and other modules are loosely coupled, and the system can be constructed on different networks. Communicator runs different protocols.

2.3URL Analysis Module

The task of the URL analysis module is to receive the URL from the distributed module, and determine whether the URL is accessed. If you access, then ignore, or join the URL list.
This module is made up of three sub modules in Figure 3.
(1) Maintenance of the upcoming URL queue module
(2) Maintenance has access to the URL queue module
(3) IP and domain name conversion module

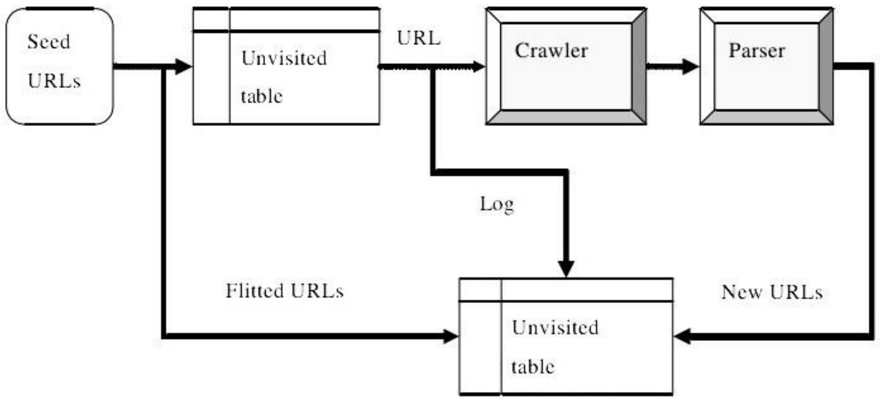

Figure 3. URL Analyze Sketch Map

2.4 Download Module

The task of the download module is to provide the URL with a higher level module, which is downloaded to the URL.
This module has three sub modules:
(1) Download the thread
(2) Node threads control
(3) DNS module

2.5 Web Analysis Module

Web page analysis module is the task of extracting the required URL. The module is simple to implement, but it must be done quickly and to extract as many URL as possible.

3. Key Technology of Web Crawler

3.1 Selection of Seed Set

If seed set is chosen not well, it will lead to the fact that searched webpage is a very small portion of total amount. To avoid that, one way is after all-round web crawling with manual intervention at the first time, the system will maintain relative URL collection; the subsequent search would base on that collection or choose representative URL from it as the initial seed set for the next search according to one strategy; the other solution is website owners actively submit URL to search engines and that the system will send directionally crawlers to those sites at certain time period.

3.2 Distributed Strategy

3.2.1 Classification of Allocation Strategy. According to collaborative way of node, the distributed web crawlers can be divided to three types:

(1) Independent mode: each node collects separately its own page, without mutual communication;

(2) Dynamic distributed way: for N collecting nodes, based on certain rules, divide web page collection G into M(M>N) subset G1,G2,…,GM, in the meanwhile of meeting first two pre-conditions for the partition of subsets in static allocation mode, no need to meet the third one; then, assign tasks of the first N subsets to N collecting points for processing; every time when one collection program finishes tasks, the system distributes next pending task subset to the node for treatment; each collecting node repeats like that till all subsets are allocated [13];

(3) Static distributed mode: classify URLs beforehand and assign to each collecting node; regarding this mode, there’s a kind of trans-regional link; the difficulty is such link may not be probably found by the collecting node to which it belongs to; if the node acquires it, it may cause repeated collection; otherwise, it may lead to missing collection; recent researches compared three modes: drop mode: not collecting pages that do not belong to one node; cross mode: collecting cross pages; exchange mode: when URL belonging to other node is collected, save those links; after they’re accumulated to a certain quantity, transmit them to the node to which they belong for collection [14].

3.2.2 Selection of Task Distribution Particle Size: Usually the size of the task allocation is the following:

(1) Random division is simple and suitable for any page which can be randomly judged if it’s collected or not;

(2) Select as per web page: rely on hashed values to determine which collector is responsible for the webpage and send its URL to relative node;

(3) Select as per website: results show that 90% links of webpage are connected to own sites, which means communication overheads will be greatly declined; HOST(URL), domain name part or host part of one web address, generally refers to one Web server; so we can see a natural division of URL based on domain, i.e. URL1 and URL2 belong to one “block” if and only if HOST(URL1)=HOST(URL2), mark HOST(URLs), meaning each element of it stands for the collection of all pages on one host; in the case of not causing confusion, the HOST (URLs) element can also be easily seen as a domain name for the host.

Here we select web-based method, whose basic idea is to distribute randomly HOST (URLs), i.e. establish the mapping from HOST(URLs) to Sn. Once a HOST (URLs) is mapped to one searching node, the node will be responsible to collect all pages under

HOST(URLs). Although different webs have big different quantity of webpage, HOST(URLs)>n, and if distribution function is too random, it's thought that web pages will more probably distributed evenly to each node.

3.2.3 Selection of Distribution Function: Distribution function concerns about how to calculate each new URL in the charge of which node; that is, the selection of distribution function is actually choice of hash function.

We mentioned granularity of distribution tasks previously and chose as per web page and HOST(URL). Hence we can easily think of this distribution function:

$$node\_num = hash(new\_url.host)\% node\_sum\_num \tag{1}$$

We improved the method here with logic two-level mapping method. First-level logic nodes can be stored in the form of array, which we make A. The subscript of each array element stands for its logic node number. Store serial number of the logic node.

Task distribution algorithm is described in details as follows:

Algorithm1 Task distribution arithmetic1

```
Input: await one URL sent from other node or returned by its competent crawling program and relative web pages
Output: download URL or pass over to other nodes
If one URL from other node is got
Check if URL is in visited_table
If one URL returned by the crawling program is collected, parse hyper link from URL-related web page
2.1 Assign one new URL from unvisited_table to the crawling program and return URL to visited_table
2.2 Perform module n hash for HOST of each newly acquired URL to get one integer i (0<i<logic node number); then get node num from one mapping table maintained on each node
2.3 Regarding each hyperlink LINK and its relative integer num
    2.3.1 If the node number is num, execute Step 1.1
    2.3.2 Otherwise, send LINK to node num.
3 Return
```

3.2.4 Implementation of Task Allocation of Web Crawlers Downloaded as Websites: The system here is used for crawling web pages of medium-size. For some big websites like Sina and 163, total-site downloading is what's implemented by the system. However, since one website's host is similar, it's necessary to make modifications on the basis of common task distribution. Firstly, task particle size cannot be host; otherwise, tasks can only be allocated to one node because the value of Hash function of pages with the same site is only; it is not workable. But if the allocation is done as per granularity of URL, it turns out too small. Hence after analysis of URLs of different big sites and conclusions, we proposed multi-computer collaborative task allocation algorithm for website's all-station downloading.

Below to see some 163 web site:

http://co.163.com:80/forum/content/94_369717_1.htm
http://quote.stock.163.com:80/s/ZNH.html
http://fund.163.com:80/05/1019/08/20DPAQNV001317NC.html
http://gameweb.163.com:80/2004/zh /boss/page_3_11_2.htm
http://www8.blog.163.com:80/rss/-dkc6.xml
http://photo.163.com:80/messages/xxo740529/

Look at the site of the Sina website:

http://finance.sina.com.cn/money/lczx/20060614/03022649081.shtml
http://blog.2006.sina.com.cn/m/panyueming
http://bj.house.sina.com.cn/decor/pretywoman/index.html

Structures of big websites are shown in Fig. 2. It's seen that they have many sub-sites. The URL of a similar sub-site web page has one point in common: the first two letters of their URLs are identical, indicating that the web page is one of the major website's sub-sites. So we have reasons to assume that those sub-sites' web pages have very close interconnection relationship; but the reference of web pages between different sub-sites are not quite frequent, with great differences. It's exactly the case after statistics. Thus we define the granularity of task distribution as sub-site.

To analyze these URL, and found the first "." the first word is a good representation of the web site. So the hash function of this article can be changed:

$$node\_num = hash(new\_url\_firstword)\%node\_sum\_num \tag{2}$$

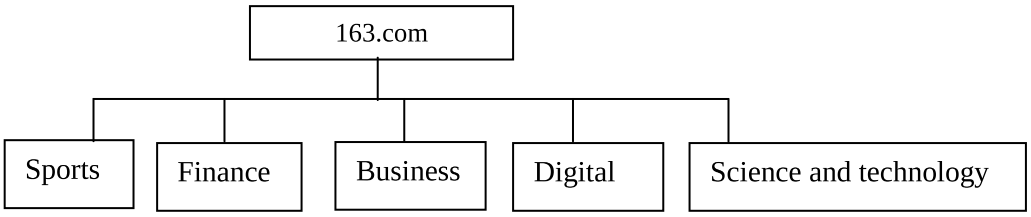

Figure 2. 163 Site Organization Sketch Map

The task allocation algorithm is modified by following steps:

Algorithm2 Task distribution arithmetic 2

```
Input: wait for one URL sent from other node or returned by its competent crawling program and relative web pages;
Output: download URL or send to other nodes;
1 If one URL from other node is got
   1.1 Check if URL is in visited_table;
2 If one URL returned by the crawling program is collected, parse hyper link from URL-related web page;
   2.1 Assign from unvisited_table one new URL to the capturing program and put the returned URL in visited_table;
   2.2 If crawlers for common crawling, go to 2.2.1; if for web crawling, skip to 2.2.2;
      2.2.1 For each newly acquired URL, according to formula 1, calculate Hash value i (0<i<maximum number of logic nodes); get specific node num from one mapping table maintained on each node;
      2.2.2 For each newly collected URL, according to formula 2, compute Hash value i (0<i<maximum quantity of logic nodes); get specific node num from one mapping table maintained on each node;
   2.3 As for every hyperlink LINK and its relative integer num:
      2.3.1 If the node number is num, do like Step 1.1;
      2.3.2 Otherwise, send LINK to node num.
3 Return
```

4. Experiment Implementation and Evaluation

4.1 Implementation of the System

This paper describes the system is a PC cluster based distributed web crawler, environment, Realized environment: 7 PC, Fedora4 operating system, the configuration of the machine: 3GHz CPU, 2GB of ram, hard disk 500g.

Figure 4 is the overall layout of the system, which is consisted of several PC machines; of that, one is controlling node and others are crawling nodes; the former is responsible for maintaining information of all nodes and synchronizing it; a few threads run on crawling nodes and complete downloading tasks.

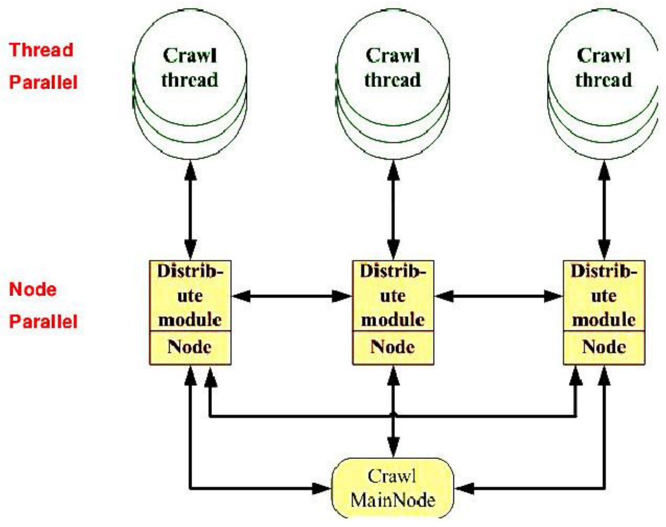

Figure 4. System Architecture Design

4.2 Realization of Distributed Task Allocation

First of all, we define one option in configuration file of program, i.e. crawlers used for web crawling or common crawling. In the initializing stage of Main function, we analyze configuration files. For every newly parsed URL, according to formula 1, we calculate Hash value i (0<i<maximum quantity of logic nodes); then we get the specific node_num from one mapping table maintained on each node. As seen from Fig.5, when one URL is got, it's passed to function check_node(). In it, invoke Hash function; then Hash() function returns one integer i, which stands for the quantity of logic nodes; then in check_node() function, inquire second-level list of second-level mapping as to get the actual node number before final judgment. If the node number is equal to local_node_num, the URL should be put in wait_list; otherwise, it's added to send_wait_list; each send_wait_list is corresponding to each node except itself.

Send_check() function performs the function: when in all send_wait_lists, the length is over SEND_LIST_MAX (macro definition), it will trigger send() function, which is implemented within one thread.

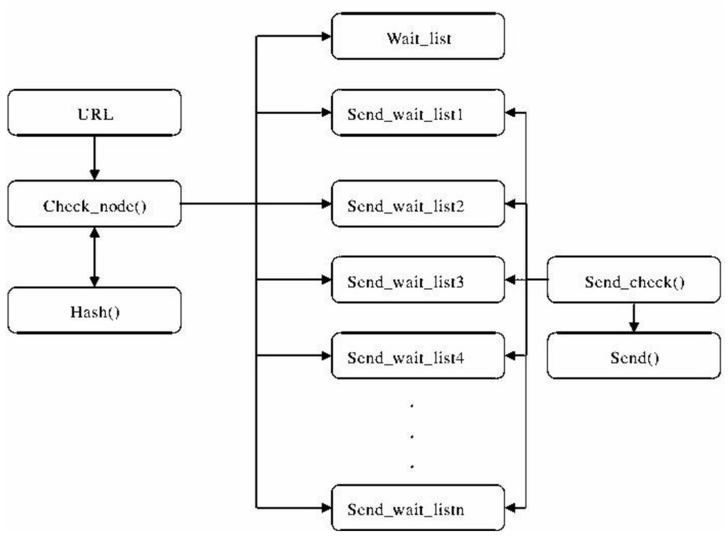

Figure 5. Task Distribute Function Map

Each node runs two threads in the beginning, as found in Figure 6.

a) Listen() thread: responsible for monitoring SERVER_PORT, receiving URLs sent from other nodes and controling synchronization information of nodes; after each node receives data packet, recall unpacking function and add URL-type objects to the wait queue based on the priority; the unpacking function aims at re-generating URL-type objects by recalling URL-type construction function after receiving URL character string and its depth.
b) Send() thread: in charge of send_wait_list; scan over the length of each queue at regular intervals; if the length is over SEND_LIST_MAX, package them and deliver; packaging function doesn't transmit the whole URL-type object but URL character string and its depth, in order to reduce communications between nodes; before delivery, it's required to extract from URL-type object the URL string and depth of it.

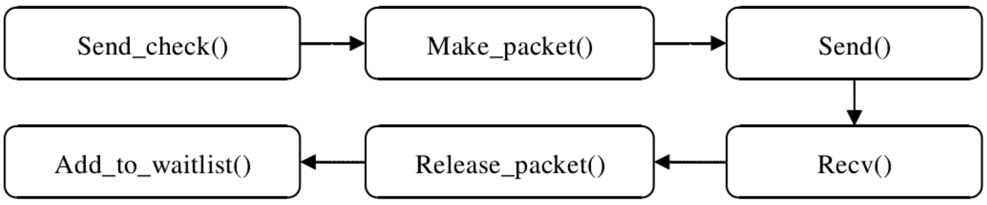

Figure 6. Node Communication Sketch Map

4.3 Realization of Single-node Downloading Tasks

Now we talk about the processing flow of single nodes, which is implemented as depicted in Figure 7: after program is initiated, recall initialization function, which needs to complete the application for data structure space, parameter initialization and invoke configuration file analytic function; then, add initial URL to WaitList.

Next, go to one cycle; at the outset of circulation, call firstly listRead() function, which should realize the function: manipulating waiting queues. Because of reasons mentioned previously, waiting queues can't totally exist in the memory; so this function needs to invoke many other waiting queues to process the function, e.g. split those queues and keep in the memory those that won't be used recently or of lower priority

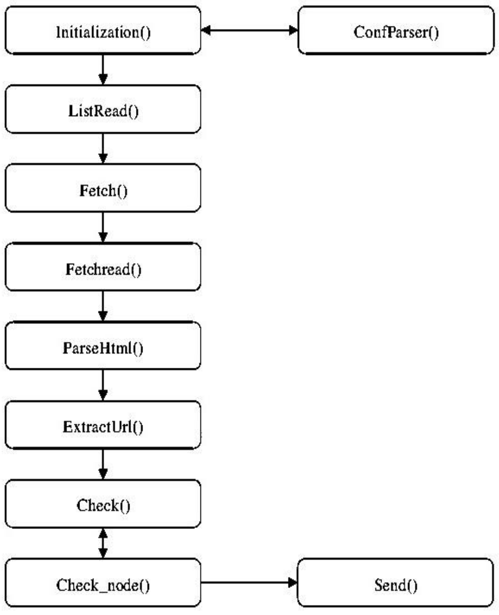

Figure 7. Overview Flow Chart

4.4 System Evaluation

The crawler can be evaluated from these:

(1) Load balancing, communication load, single node's downloading speed and comparison of speed of single nodes in distributed crawlers;

(2) Testing all-station crawling of the crawler used as one application of itself across one web page, and test algorithm 2.

4.4.1. Single Crawl Test: Parameter setting, it is shown in Table 1.

Table 1. Parameter Table

Parameter	Values
Download thread count	200
Dns thread number	5
Crawl depth	15
Same server access interval	35s
Initial URL	www.163.com

Crawl Results: 10 hours, Download page :1176804, the size of 26.16GB

4.4.2 Parallel Crawling Test for 2 Nodes: Parameter setting, it is shown in Table 2.

Table 2. Parameter Table

Parameter	Values
Download thread count	200
Dns thread number	5
Crawl depth	15
Same server access interval	35s
Initial URL	www.163.com
SEND_LIST_MAX	500
Send_check query time	0.1s

Crawl result:
Crawling time: 10 hours
Node 1: download the web 901355, the size of 21.34GB
Node 2: download the web 681645, the size of 15.94GB
Node 1 to node2 transmitted 4645344 URL
Node 2 to node 1 transmitted 3887462 URL.

The testing started from 10 P.M. to next morning. Single nodes reached desirable results. In the parallel testing, the efficiency of single nodes declined partially versus standalone ones, which is however inevitable and in the normal range, because two nodes crawling simultaneously share outbound network port and the network bandwidth becomes the main bottleneck to concurrent crawling. The running efficiency of double nodes increased by around 40% is than single node. We believe improved networking environment will boost greatly the working efficiency of concurrent network crawlers. It is shown in Figure8.

5. Conclusion

We studied about how to establish a distributed web crawler and introduced key technologies of such crawler. The infrastructure used here is N common PC machines, of which one is controlling node and the others are crawling nodes; the former is responsible for monitoring the work of an entire crawler; each crawling node collaboratively downloads web page. One system was designed with distributed web crawler. It can flexibly configure parameters and has good robustness and scalability. With the application of the system, all-station downloading became possible. Experiments proved that concurrent web crawler's downloading speed was increased much than standalone crawler. It can be used as one executable medium-size web crawler to put in practice.

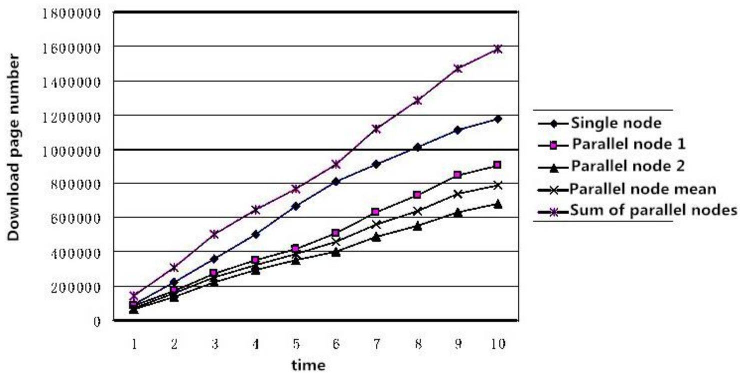

Figure 8. The Performance of Stand-alone and Parallel

References

[1] Yangrui. Hu Hongsi, Zhang Wenbo, Yao Tianfang. Design and implementation of a distributed web crawler. Journal of Jiangxi Normal University (NATURAL SCIENCE EDITION), 2013,04:382-386.
[2] Wang Yan. Development of web crawler technology in search engines, telecommunications express, 2008,10:20-22.
[3] Zhou da. in the cloud environment Web application scanning in the network of web crawler technology research. Information network security, 2012,05:20-23.
[4] Zhan Hengfei, Yang Yuexiang, Fang Hong. Research and optimization of Nutch distributed crawler. Computer science and technology, 2011,01:68-74.
[5] Wu Libing, Ke Yalin, He Yanxiang, Liu Nan. Design and implementation of distributed web crawler. Computer application and software, 2011,11:176-179.
[6] Wang Yanhong, Zhou Jun. Research on the technology of web crawler based on Hadoop. Journal of Jilin Institute of engineering and technology, 2014,08:87-89.
[7] Sun Yibing, Liu Hongbo. A model based on the price index of the web crawler technology. Statistical research, 2014,10:74-80.
[8] Guo Xianghao, Li Lei. Chinese intelligent search engine. Http://www.ccidnet.com2
[9] D. Eichmann. The RBSE Spider-Balancing Effective Search Against Web Load. In Proceedings of the First International World Wide Web Conference, 1994:113~120
[10] O. A McBryan. GENV and WWW: Tools for Taming the Web. In Proceedings of the First International World Wide Web Conference 1994:79~90
[11] B. Pinkerton. Finding What People Want: Experiences with the Web Crawler. In Proceedings of the First International World Wide Web Conference 1994:54~65
[12] Pan Chunhua, Chang Min et al. Design and development of information collection tool for Web. Computer application research, 2002, (6): 144~148
[13] Demetrios Zeinalipour-Yazti, Marios D. Dikaiakos. Design and implementation of a distributed crawler and filtering processor. In Proceedings of the fifth Next Generation Information Technologies and Systems (NGITS), volume 2382 of Lecture Notes in Computer Science, Caesarea, Israel, 2002: 58~74
[14] Junghoo Cho, Hector Garcia-Molina. Parallel crawlers. In Proceedings of theeleventh international conference on World Wide Web, Honolulu, Hawaii, USA, ACM Press, 2002: 124~135

Author

Hongyan Zhao, She received her B.S degree from Shandong University of Finance. She received her M.S degree from Shandong Normal University. She has been ph.D student in Thailand Dhurakijpundit University. She is an Associate professor in Shandong Yingcai University. Her research interests include Data Mining, Information Processing.