Design and Implementation of a Scalable Distributed Web Crawler Based on Hadoop

YuLiang Shi and Ti Zhang
School of Beijing University of Technology
BJUT
Beijing, China
e-mail: shiyl@bjut.edu.cn, zhangti_it@163.com

Abstract—**In this article, an efficient and scalable distributed web crawler system based on Hadoop will be design and implement. In the paper, firstly the application of cloud computing in reptile field is introduced briefly, and then according to the current status of the crawler system, the specific use of Hadoop distributed and cloud computing features detailed design of a highly scalable crawler system, and finally the system Data statistics, under the same conditions, compared with the existing mature system, it is clear that the superiority of distributed web crawler. This advantage in the context of large data era of massive data is particularly important to climb.**

Keywords-big data; cloud computing; hadoop; distributed crawler.

I. INTRODUCTION

As a basic component of search engine and network data mining, network crawler plays an important role. With the explosive growth of network scale and complexity, the centralized network crawler that relies on the single machine processing capacity of the computer can not meet the demand of rapidly acquiring large amount of resources. Distributed network crawler is composed of multiple Agents that can acquire resources in parallel. It has good expansibility and can meet the practical needs of resource retrieval. The location and network topology of these agents can be different, so they can be deployed in the same local area network LAN called a single domain network crawler, will be deployed in the WAN different locations called multi-domain network crawler. The size and speed of a single-domain network crawler is affected by the total bandwidth of the LAN network. The crawling size of the multi-domain network is less affected by the network bandwidth, but the communication delay is unreliable, and the retrieval effect is susceptible to the geographical location and network structure.

II. RELATED THEORIES AND TECHNIQUES

This chapter focuses on the relevant technologies used in this paper, providing a technical background for the construction of distributed crawlers[1]. As cloud computing is a continuation and extension of distributed, so the first cloud computing concepts, architecture and other related theories. Then, the Hadoop technology of distributed crawler system is studied deeply, including distributed storage model (HDFS) and distributed computing model (Map / Reduce).

A. Cloud Computing.

Cloud computing is an increase, use, and delivery model of Internet-based services that typically involves providing dynamically scalable and often virtualized resources over the Internet. It is a new computing method of shared computing resources, it proposed the mass of the Internet cloud computing together to provide computing services. Cloud computing proponents believe that cloud computing will be epoch-making groundbreaking computing model, because it makes supercomputing a reality. Enterprises or individuals no longer need to look at the bottleneck of hardware development, do not have to spend a lot of money and financial resources to buy a super server, just at the software level to build a cloud computing framework will be able to complete the massive data supercomputing.

Cloud computing is an improved processing of grid computing, parallel processing and distributed processing, its predecessor is the use of parallel computing to solve large-scale grid computing and computing resources as a measurable service to provide public computing, broadband Internet technology and Virtualization technology after the rapid development of cloud computing sprouted. I use the above basic principles, the use of Internet connectivity and geographically distributed computer clusters exist to provide users with computing, storage, hardware and software and other services. This allows the user to take full advantage of each cluster in the computer to calculate and store their data. Cloud computing is the ordinary personal computers or servers connected together to form a cluster, which is equivalent to access to these computer computing power and storage capacity of the sum, but the cost of spending more inexpensive. This feature can be used to achieve the scalability of my crawler system.

B. Hadoop

Hadoop is an open source Apache project, the project is a reliable, scalable distributed[2] framework. Hadoop's design is fully consistent with google developed Map / Reduce and Google File System. In Hadoop's distributed framework, the computational model is still called Map / Reduce, but the storage structure is called the Hadoop Distributed File System. These two parts can be said to be the core of Hadoop.

(a) Distributed File System (HDFS) is a multi-physical machine on the collaborative organization and management

978-1-5090-3619-6/17/$31.00 ©2017 IEEE

of documents on the system. In Hadoop users manage files and manage files on a machine.

(b) Map / Reduce computing model is a programming model, Hadoop framework in the use of Map / Reduce programming model allows multiple machines to co-calculate a task.

III. Analysis, Design and Implementation of Distributed Web Crawler

A. The Analysis of Distributed Crawler System

Distributed Web crawler is developed from the traditional reptile, so it works and the basic structure is similar to the traditional Web crawler. It can be used as a combination of multiple Web crawler macros. These reptiles can be scattered in different geographical locations, can also be in the same local area network, according to the degree of the crawler system dispersion, distributed reptiles can be based on LAN and WAN crawlers[3]. Because of the high cost and complicated implementation of the crawler system based on wide area network, only large and powerful enterprises adopt it. The system implemented in this paper is not a large-scale distributed system. It is a simple and easy master-slave mode distributed crawler based on LAN. system.

A well-designed Web crawler must meet the following characteristics: distributed and scalable, high performance and stability, flexible scalability and load balancing. Based on these principles, I adopt the basic architecture of distributed clusters. I set the local area between the nodes in the cluster to support the addition of node network, which can improve the performance of the cluster. The distributed Web crawler system always runs on the cluster. All nodes operating system using open source operating system Linux operating system. The underlying file system of the cluster is the Hadoop distributed file system. Distributed computing framework using Map / Reduce.

B. The Structure of Distributed Crawler

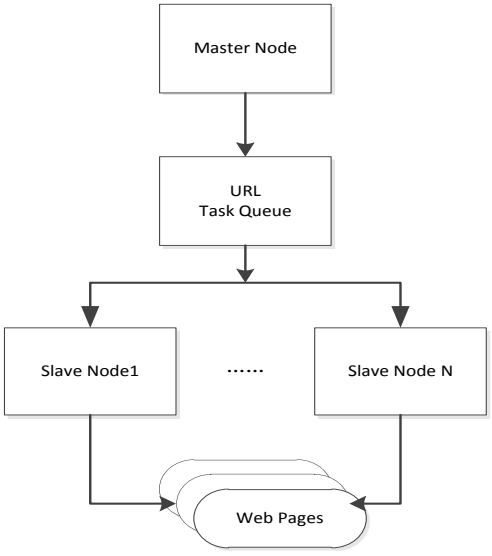

Figure 3-1. The overall architecture of a distributed Web crawler

The overall architecture of the crawler is shown in Fig. 3-1. The distributed crawler consists of a master node and multiple crawling slave nodes.

1) The function of the master node

a) to manage and distribute the URL of the whole system, including:

<1> get the URL from the node to resolve the queue, aggregate all the nodes from the URL, the de-emphasis stored in the master node URL database.

<2> Allocate URLs from nodes to ensure even and efficient crawling of each page from node to node.

b) to ensure smooth communication from the node to maintain the stability of the entire system.

c) the main node does not crawl and parse the page, only scheduling management and system stability maintenance.

2) Function of the slave node

a) URL crawling, which is the main function from the node, the main node assigned to crawl over the URL, regardless of success or failure to record crawl results, a timeout and retry mechanism.

b) page resolution process: from the crawl to resolve the effective URL of the Web page to store the database as a URL crawl further seeds. There are two main URL: (1) search engine URL, this type of page only search engine to return to the effective search links; (2) ordinary web page information, this page information only to retain the page URL and the page at the URL Of the host consistent URL.

c) the node is not responsible for any communication between nodes and URL record maintenance (such as URL conflict excluded), only from the node will resolve the URL stored in the database, waiting for the master node to complete the reading and de-heavy work.

C. The Structure of Distributed Crawler System Design

According to the reptile flow chart in Figure 3-2, the basic flow of the climbing system is described in detail as follows:

a) First select the URL seed file collection from the local file system upload to the Hadoop cluster distributed file system HDFS in the file, in this folder is always stored in the current layer to crawl the URL. At the same time, set the number of grabbed layers to 0.

b) to determine in the folder to be crawled[4] queue is empty. If yes, jump to (g); otherwise, execute (c).

c) to crawl in the folder to be crawled in the queue. The crawling process is done by the CrawlerDriver module, which is a Hadoop-based Map / Reduce process. Later I will detail the CrawlerDriver module Map / Reduce detailed implementation. And finally crawled down the page stored in the HDFS doc folder. The doc folder holds every layer of unprocessed web pages.

d) to resolve the pages have been crawled from the doc folder has been extracted from the web page to extract the

chain out of the link. This parsing process is done by the ParserDriver module, which is also a Map / Reduce calculation process. The specific extraction is achieved in the Map phase through HTML parsing. Later I will detail the ParserDriver module Map / Reduce detailed implementation. After this process will be extracted out of the chain link stored in the HDFS out of the folder. The out folder is always stored in the current layer out of the chain out of the link.

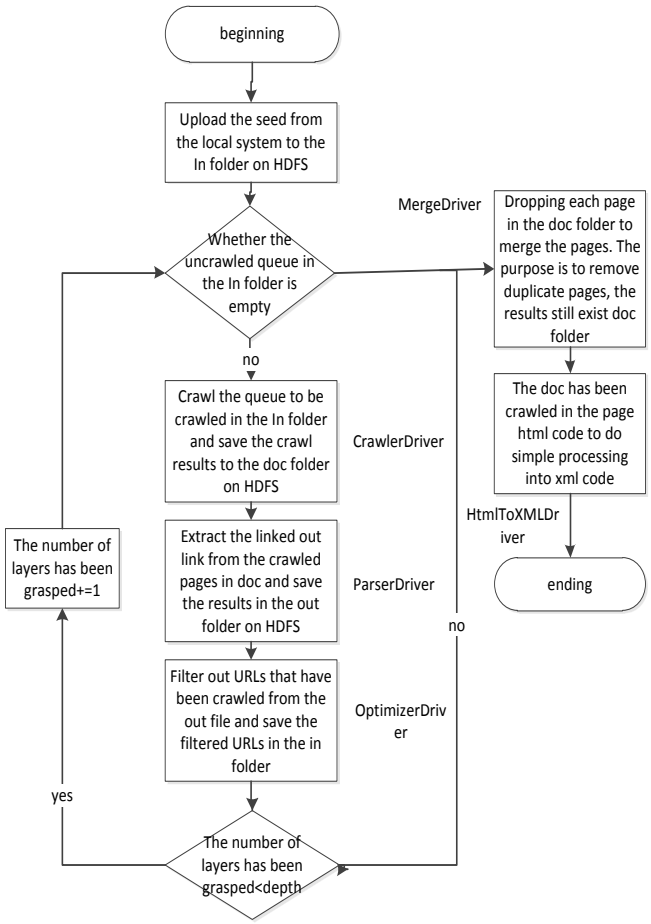

Figure 3-2. Reptile flow chart

e) to optimize the link out of the link, filter has been crawling the URL, that is out from the folder has been parsed out of the link to the URL link in the filter has been captured URL. This optimization process is done by the OptimizerDriver module, which is also a Map / Reduce process. Later I will detail how to complete the OptimizerDriver module based on Hadoop Map / Reduce implementation. After optimization, the optimized URL set will be saved in the in folder to wait for the next round of crawling.

f) determine whether the number of layers has been grasped less than depth. If the number is less than 1, the number of grabbed layers is incremented by 1; otherwise, it returns to (g).

g) merge to heavy, each page will be merged at the same time to remove the crawl of the pages removed. This work is done by the MergeDriver module, the module is also a Hadoop-based development of Map / Reduce process. Later I will detail how to complete the OptimizerDriver module based on Hadoop Map / Reduce implementation. After merging, the results are still stored in the doc folder on the distributed file system HDFS.

h) on the crawl of the page to do a simple pretreatment. Html code will be converted to xml. This work is done by HtmlToXmlDriver module, the same, this module is also a Hadoop-based development of Map / Reduce process. Will handle the xml file stored in the xml folder in HDFS.

i) over.

From the above process, I can see that the implementation of the entire crawler system can be divided into five parts, each part is an independent module to complete the corresponding function, each module corresponds to a Map / Reduce process. The following describes the function of these five modules:

- CrawlerDriver module: parallel download the queue to be crawled, the text file in the folder as a URL to be crawled seed collection, the text file in the first round of the user is given the initial crawl, from the first The beginning of the second round is extracted from the chain out of the link. The module is based on Hadoop developed a Map / Reduce process, Map and Reduce were completed different functions, the specific download is completed in the Reduce stage, and the use of multi-threaded download, download part of the use of Java network programming completed. The downloaded web page is saved in the doc folder on HDFS.
- ParserDriver module: parallel analysis has been downloaded from the Web page, link extraction link. According to the doc folder has been downloaded from the web page analysis out of each page in the link to the link that is linked to the link. The module is also based on Hadoop developed Map / Reduce process, but only need a Map phase to complete the goal. In the Map phase of the main work is to use the HTML parser to parse out the chain of links, in addition, but also by the rules limit the type of URL, to prevent the chain of links extracted to other sites[5]. Finally, these chain links are stored in HDFS on the out folder.
- OptimizerDriver module: parallel optimization link chain, filter out the duplicate link. According to out of the folder link has been extracted out of links to optimize the remaining URL for the next layer to deal with the crawl. As the relationship between the site layer and the link layer is a diagram of the structure, so the work of the module can be understood as a loop to find the problem, will constitute the loop URL filter out. This module is also a Map / Reduce process based on Hadoop development. Optimized URL will be stored in the HDFS in the in the folder.

- MergeDriver module: parallel merge layers crawled web pages. According to the doc folder in each layer to crawl the web page, merge, remove the layer may overlap between the pages. This part is also a Map / Reduce process based on Hadoop development. Finally, the results are still stored in the doc folder.
- HtmlToXMLDriver module: parallel to the HTML into XML. According to doc folder in the crawl of the page, complete the conversion pretreatment. This is done in the DOM tree. It is also a Map / Reduce process. The converted xml is saved in the xml folder on the HDFS.

In this way, the five functional modules constitute a distributed crawler system based on Hadoop. The CrawlerDriver, ParserDriver and OptimizerDriver are executed cyclically from the generated queue to be crawled to complete the crawling of the web pages. After the loops, the MergeDriver and HtmlToXMLDriver preprocessing are performed[6]. Among them, the number of loops is controlled by the pre-set parameters "crawl depth" and "whether the queue to be fetched is empty".

IV. Analysis of Experimental Data

A. Scalability

The test environment is as follows: The network bandwidth is 2MB per second. The initial site is 20 news portal. The crawl depth of the URL link is set to 2, and I test to download about 11,000 pages in 3 different experiments. The search results are shown in Table I.

TABLE I. Extensibility Test Table

Node	Visited Pages	Successfully download pages	Take time(second)	Rates (pages/second)
2	11812	11236	846	13.96
3	11887	11243	753	15.79
4	11965	11196	621	19.27

When the cluster is two nodes, it takes 846 seconds to complete the crawl. When I add a third node, it takes 753 seconds. When I add the fourth node, it takes 621 seconds. It shows that the crawler system is scalable[7], that is, when the node increases, the system performance is improved.

B. Crawl Efficiency

In this section, the system uses 4 nodes and Heritix (a good open-source single-machine network crawler system) contrast. The initial creep[8] was almost the same, but the differences began to appear gradually after about 35 minutes. The results of the crawl[9] were shown in Table II. As you can see, MyCrawler is more efficient than Heritrix.

TABLE II. Crawl Efficiency Comparison Table

	Heritrix	MyCrawler
Take time(second)	2311	2204
Download pages	50009	96532
Rates(pages/second)	21.64	43.80

Since the number of crawled pages per second has changed over time, My tests start at 35 minutes and are counted every 5 minutes until the 60th minute. Then draw the crawling efficiency[10] trend, as shown in Figure 4-1. The X coordinate represents time, and the Y coordinate represents the number of pages crawled per second.

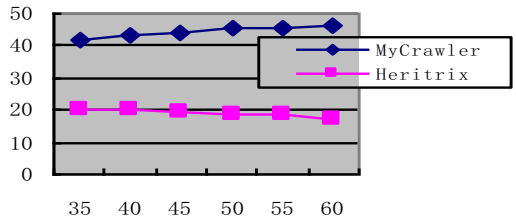

Figure 4-1. Comparison of crawl efficiency

MyCrawler is more efficient than Heritrix by comparison. Limited to the impact of network bandwidth, CPU resources, memory and other factors, stand-alone network crawler is difficult to play high-performance and crawling large-scale data, it will become more and more powerless.

V. Summary

In this paper, I design and implement a highly efficient and scalable distributed Web crawler system based on Hadoop technology. Through multi-machine efficiency together, each node can work through multi-threaded efficiency. In contrast to Heritrix and Nutch, My creep speed advantage is very obvious. Due to limited time and ability, it is difficult to balance all aspects, and there are many places that need to be improved and improved. With the rapid development of the Internet, new technologies are emerging. Web crawler requirements are also increasing. It is necessary to optimize existing solutions and research new solutions to meet new challenges.

On the other hand, addressing the problem is also a key factor affecting performance. Nutch is based on the Distributed Search Engine Web Crawler, which has some advantages in terms of search. However, it crawls slowly. The number of pages crawled per second is only 5, when it runs on 3 nodes. In contrast, MyCrawler's crawling speed advantage is obvious, I realize the functional focus is different. Therefore, it needs to be selected according to actual needs, in order to achieve a more reasonable situation.

References

[1] M. Marin and C. Bonacic. Bulk-Synchronous On-Line Crawling on Clusters of Computers. In 16th Euromicro Intl. Conf. on Parallel, Distributed and Network-Based Processing (PDP 2008), pages 414-421. IEEE CS, 2008.

[2] H T Y Achsan, W C Wibowo, "A Fast Distributed Focused-web Crawling [J]", Procedia Engineering, vol. 69, no. 1, pp. 492-499, 2014.

[3] M Wu, J. Lai, "The Research and Implementation of Parallel Web Crawler in Cluster [J]", International Conference on Computational & Information Sciences, pp. 704-708, 2010.

[4] Y Ye, F Ma, Y Lu et al., "iSurfer: A Focused Web Crawler Based on Incremental Learning from Positive Samples [J]", Lecture Notes in Computer Science, 2004.

[5] R Ferreira, P Brito, J Melo et al., "An architecture-centered framework for developing blog crawlers [J]", Expert Systems with Applications, vol. 40, no. 4, pp. 1177-1195, 2013.

[6] H Zhan, Y. Yang, "Research and Optimization of Nutch Distributed Crawler [J]", Journal of Frontiers of Computer Science & Technology, 2011.

[7] Y. Gu and R. L. Grossman, "Sector and Sphere: the design and implementation of a high-performance data cloud," Philosophical transactions. Series A, Mathematical, physical,and engineering sciences, vol. 367, pp. 2429--2445, 2009.

[8] D. Borthakur. The Hadoop distributed file system: Architecture and design. http://lucene.apache.org/hadoop/hdfs_design.pdf, 2007.

[9] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[10] Rajaraman A, Ullman J D. Mining of massive datasets[M]. Cambridge University Press, 2012.