Implementation of a Distributed Web Crawler

Seint Seint Win, Nyein Nyein Lwin
University of Computer Studies, Yangon
saintsaintwin09@gmail.com

Abstract

Today's search engines are equipped with specialized agents known as Web crawlers (download robots) dedicated to crawling large Web contents on line. Crawlers interact with thousands of Web servers over periods extending from a few weeks to several years. Large scale search engine such as Google use distributed crawler to crawl the entire WWW. The distributed crawler harnesses the excess bandwidth and computing resources of clients to crawl the web. This paper presents design and implemented a scalable distributed crawler by using distributed programming facilities provided by Java RMI. Hash based partitioning is used to partition the urls among the crawlers; communication among crawler is done by Remote Method Invocation. The Crawler can run many crawler instances at the same time.

Keywords: Distributed Crawling, Java RMI

1. Introduction

Search Engine such as Google [9] have become an integral part of the internet. Search engine generally compose of three core components. First, the crawling component crawl the web and downloads the web contents to be cached locally. Second, the indexing component precomputes indexes for the web contents for efficient search. Last, the search component uses the indexes for executing user search queries returning ranked results [2].

Web spider (Crawler or robots) plays an important role as an essential infrastructure of every search engine. It automatically discovers and collects resources, especially web pages, from the Internet [5]. Highly efficient crawling systems are needed in order to download the hundreds millions of web pages indexed by search engines. In fact search engines compete against each other primarily based on the size and currency of their underlying database, in additions to the quality and response time of their ranking function. Even the largest web search engine, such as Google, currently cover only limited part of the web, much of their data is several month out of date[7].

Distributed Web Crawling is a distributed computing technique whereby internet search engines employ many computers to index the internet via web crawling. Each Crawler node in Distributed Crawler downloaded the web page independently and sends the results to central server for later processing. This paper develops a scalable, centralized distribute crawler that can crawls the web pages from internet or intranet.

In centralized system, a single server machine controls other client in order to accomplish the desired task whereas the decentralized system requires no server for coordination of their work.

The rest of this paper is organized as follow, section 2 provides related work. In section 3 we describe theory background, section 4 is system implementation, and section 5 is conclusion.

2. Related Work

There have been many reports on literature on distributed crawling.

CrawlWave is a distributed crawler that used SOAP (Simple Object Access Protocol) as a mechanism for communication; it is used to crawl Greek web pages. It is constructed like a peer to peer system [1]. CrawlWave client communicate with CrawlServer by means of SOAP(Simple Object Access Protocol) message for exchanging web pages and urls.

UbiCrawler is implemented in Java and facilitate the use of JavaRMI and API, which is also fully scalable, distributed and fault tolerant. It used consistent hashing for url partitioning [6].

V. Shkapenyuk et.al presented their work on distributed crawler, the basic crawling strategies used is breadth first crawling. They employ scheduling policies for load balancing between crawlers [7].

K. Koht-arsa et.al constructed a distributed crawler based on the Beowulf clusters. Each crawler is run on the cluster node, urls are partitioned by using the phase swapping, a modified version of url hashing. Synchronization is done by using the NTP (Network Time Protocol) in order to successfully use phase swapping among cluster node. They used compression algorithm to store url in their queue [5].

W. Gao et.al presented geographically focused crawler, their crawler also partition the urls by hash based partitioning method. Crawler nodes in their crawler download web pages that are geographically nearest urls among crawlers [8].

WebRace is also a distributed crawler written in Java. It is part of a more generic system called eRace and described in [3].

Mobile crawlers are described in [4]. B.T. Loo et.al described a distributed crawler that use PIER query engine and Distributed Hash Table for Node partition. Urls are partitioned by their host name, so each crawler downloads only one domain name that can reduces unnecessary network traffic for communication and synchronization and computation of hashing [2].

3. System Overview

One of the main component of a search engine is a web crawler which downloads the web page from the internet. **A web crawler** (also known as a **web spider**, **web robot**) is a program or automated script which browses the World Wide Web in a methodical, automated manner. Other less frequently used names for web crawlers are **ants**, **automatic indexers**, **bots**, and **worms**. This process is called **web crawling** or **spidering**. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a website, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for spam).

A web crawler is one type of bot, or software agent. In general, it starts with a list of URLs to visit, called the **seeds**. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the **crawl frontier**. URLs from the frontier are recursively visited according to a set of policies.

A distributed crawler is a Web crawler that operates simultaneous crawling agents. Each crawling agent runs on a different computer, and in principle some agents can be on different geographical or network locations. On every crawling agent, several processes or execution threads running in parallel keep (typically) several hundred TCP connections open at the same time.

A parallel crawling system requires a policy for assigning the URLs that are discovered by each agent, as the agent that discovers a new URL may not be the one in charge of downloading it. All crawling agents have to agree upon such a policy at the beginning of the computation. To avoid downloading more than one page from each server simultaneously, the same agent is responsible for all the content of a set of Web servers in most distributed crawling systems. This also enables exploiting the locality of links, that is, the fact that most of the links on the Web point to other pages in the same server makes it unnecessary to transfer those URLs to a different agent. An effective assignment function balances the load across the agents such that each crawling agent gets approximately the same work load. A simple url partitioning method is to compute a simple hash and used the hashing value to determine which urls should be downloaded by which crawler agents.

Each Crawler is assigned a unique identifier. Each Crawler maintains its own URL Queue, extracted urls are from URL extractor are stored in the queue if modulo of sum of all of the character in url is equal to the identifier of the crawler ,otherwise the extracted urls are send to communication module, which later send it to other crawler. Downloader downloads the web page by picking up a url from the url queue, the resulting web page is given to the Link Extractor. Link extractor extracts all hyperlinks form the web page, the extracted links are send to the url seen tester to check that the links is already downloaded, if the link is not downloaded already and the crawler must handle that url, the links is added to the url Queue. Communication module is used to communicate with Crawling Manger, communication module interacts with crawling manager to send downloaded web page or to send url for other crawler.

Link Extractor is a module in the crawler which extracts all hyperlink in a web page by using the regular expression.

URL Queue stores all the urls that must be downloaded by the crawler, URL queue is implemented as a FIFO queue, so breadth first crawling can be achieved.

URL seen tester test so that given a url, it check to see that url is already downloaded or not by the crawler.

Downloader downloads the web pages and gives the resulting html pages to the link extractor to extract all hyperlink from that page.

Crawling agents must exchange URLs, and to reduce the overhead of communication, these agents exchange them in batches, i.e., several URLs at a time. Additionally, crawling agents can have as part of their input the most cited URLs in the collection. They can, for example, extract this information from a previous crawl .In this way, agents do not need to exchange URLs found very frequently. Working algorithm of the crawler is showed in Figure 3.

3.1 Java Remote Method Invocation

Java RMI allows one JVM to communicate with another JVM and have it execute an object method. Object can invoke methods on other objects located remotely as easily as if they were on the local host machine. Communication for RMI can be divided into two categories, server and client. Server exposes a service method to be executed by client. Server register with a lookup service to allow client to find them, or they can make available reference to the service in some other fashion. In our distributed crawler, Crawl Manger is an RMI Server, and each Crawler is a RMI Client. Overview of crawler and crawling manager is shown in Figure 1. Execution of the crawler is showed in Figure 2.

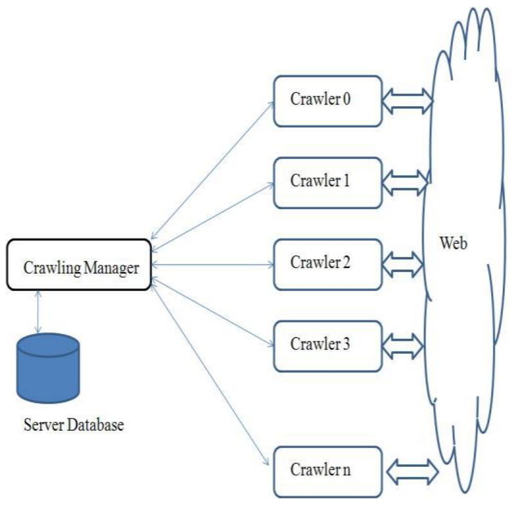


Figure 1. Overview of the System

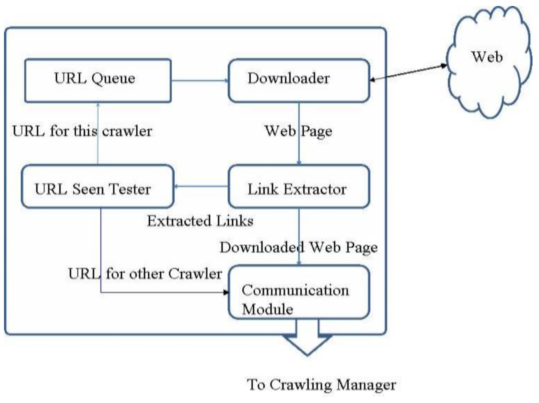


Figure 2. Execution of Crawler

```
Crawler( url_queue)
    while( url_queue not empty)
    url  = pick URL from url_queue
    doc = download url
    new_urls = extract urls from doc
    select urls in new_urls which must be
    downloaded by this crawler by using hash
    based partitiong
    insert urls into queue unless if have been
    downloaded
    Send other urls and web page to Server
```

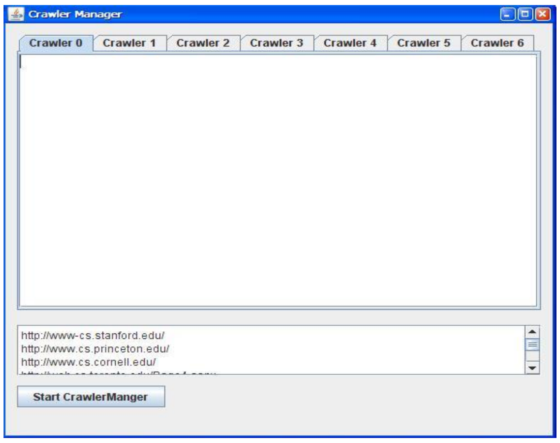
Figure 3. Algorithm for a Crawler

4. System Implementation

In this paper, we implement distributed crawler in a centralized fashion, where CrawlerManger controls all operation of each Crawler.

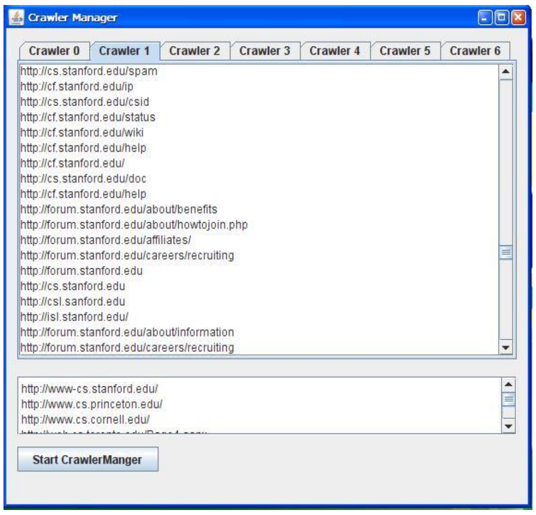
4.1 Crawling Manager

Crawling Manager keeps all instance of the crawler running in the system. Each crawler send the url that is not relevant to them to the Crawling manager, Crawler manager sends the url to the appropriate url to the correct crawler. Crawling manager also accept downloaded web page from each crawler and store it in the server. Crawling manager must be run before any other crawler. When Crawler are started, they are registered in Crawling manager, Crawling manager send crawler id and initial list of url to the corresponding crawler. When sending url list Crawling manager computes hash for each url to determine which crawler must be responsible for downloading that urls. Crawling manager must do this by invoking a remote method on the Crawler instance object.



```
http://www-cs.stanford.edu/
http://www.cs.princeton.edu/
http://www.cs.cornell.edu/
```

Figure 4. Screen Shot for Crawling Manager Intially



```
http://cs.stanford.edu/spam
http://cf.stanford.edu/ip
http://cs.stanford.edu/csid
http://cf.stanford.edu/status
http://cf.stanford.edu/wiki
http://cf.stanford.edu/help
http://cf.stanford.edu/
http://cs.stanford.edu/doc
http://cf.stanford.edu/help
http://forum.stanford.edu/about/benefits
http://forum.stanford.edu/about/howtojoin.php
http://forum.stanford.edu/affiliates/
http://forum.stanford.edu/careers/recruiting
http://forum.stanford.edu
http://cs.stanford.edu
http://csl.sanford.edu
http://isl.stanford.edu/
http://forum.stanford.edu/about/information
http://forum.stanford.edu/careers/recruiting
```

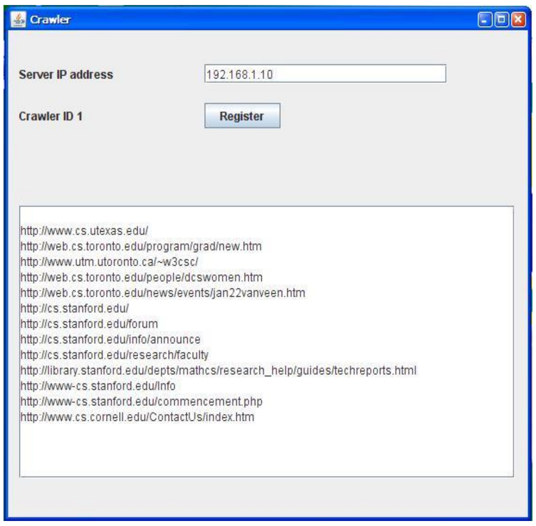
```
http://www-cs.stanford.edu/
http://www.cs.princeton.edu/
http://www.cs.cornell.edu/
```

Figure 5. Screen Shot for Crawling Manager

Initial state of the crawling manager is showed in Figure 4. Screen shot for crawling manager, a list of crawler registered in crawling manager, the urls send by each crawler is showed in Figure 5.

4.2 Crawler

When the system startup, the crawling manger register itself in the RMI registry. Several crawlers can be added to the crawling system. When a crawler want to add to the crawling system, it must register in the crawling manager, crawler register themselves at the crawling manger by connecting the server and looking up the crawling manager server object by means of Java Remote Method Invocation. As a result, crawling manager issues an internal crawler id that is used for url partitioning. Crawling manager stores all instances of the crawlers in the system. When the crawling manager want to initiate crawling, it broadcasts the initial url ,seed given by the user to all crawlers, crawlers received urls from the crawling manager and start crawling. Each crawler download the page from its own URL queue, extracts hyperlink from that pages, and then it compute hashing on the new urls. Hashing is simply done by summing all ASCII characters in the url and picking the modulo of this sum by their internal crawler id. If the remainder is equal to their crawler id ,that url is stored on their own queue, otherwise that url is send to the crawling manager, and the crawling manager keeps all urls sended by each crawler in a temporal queues, when temporal queue of the crawling manager exceeds more than 10 urls , server computes hash on each url in its temporal queue, and send the corresponding urls to each crawler. The screen shot of crawler, urls downloaded by the crawler is shown in Figure 6.



Server IP address 192.168.1.10

Crawler ID 1

```
http://www.cs.utexas.edu/
http://web.cs.toronto.edu/program/grad/new.htm
http://www.utm.utoronto.ca/~w3csc/
http://web.cs.toronto.edu/people/dcswomen.htm
http://web.cs.toronto.edu/news/events/jan22vanveen.htm
http://cs.stanford.edu/
http://cs.stanford.edu/forum
http://cs.stanford.edu/info/announce
http://cs.stanford.edu/research/faculty
http://library.stanford.edu/depts/mathcs/research_help/guides/techreports.html
http://www-cs.stanford.edu/Info
http://www-cs.stanford.edu/commencement.php
http://www.cs.cornell.edu/ContactUs/index.htm
```

Figure 6. Screen Shot for Crawler

5. Conclusion

This paper presents an implementation of a distributed crawler that is fully scalable to many crawler, crawler can download web pages simultaneously from the web sites and send it to the crawling manager. This system used Java RMI for distributed computing facilities; Java RMI and network API are suitable for a distributed crawler.

6. References

[1]. A. Kritikopoulos, M.Sideri, K. Stroggilos, "**CrawlWave: A Distributed Crawler**",Technical Report, Athens University of Economics and Business, 2004.

[2]. B.T. Loo, O. Cooper , S.Krishnamurthy, "**Distributed Web Crawling Over DHT**'", Techical Report, University of California, Berkely,

[3]. D. Zeinalipour-Yazti, M.Dikaiakos, "**Deisgn and Implementation of a Distributed Crawler and**

Filtering Processor", Technical Report, University of California, USA, 200..

[4]. J Fiedler, J. Hammer, "**Using The Web Efficiently: Mobile Crawlers**", UF Technical Report ,University of Florida, June 1998.

[5]. K. Koht-arsa, S. Sanguanpong, "**High Performance Large Scale Web Spider Architecture**", Kasetsart University, Thailand.

[6]. P. Boldi, B. Codenotti,M. Santini, S. Vigna, "**UbiCrawler: A Scalable Fully Distributed Crawler**"

[7] V. Shkapenyuk, T.Suel, "**Design and Implementation of a High Performance Distributed Web Crawler**", Technical Report, Polytechnic University, 2001

[8] W. Gao, H.C. Lee, Y.Miao, "**Geographically Focused Collaboartive Crawling**", Proceeding of the ACM International World Wide Web Conference, 2006.

[9] http://www.google.com