Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 17 (2013) 819 – 827

Procedia Computer Science

Information Technology and Quantitative Management (ITQM2013)

Distributed OSN Crawling System Based On Ajax Simulation

Shan Jixi[a,b,*], Sha Ying[a], Li Yang[b,c], Xu Kai[a], Guo Li[a]

[a] National Engineering Laboratory for Information Security Technologies, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093

[b] Graduate School of Chinese Academy of Sciences, Beijing, 100049

[c] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190

Abstract

In the age of Web2.0, lots of online social networks (OSNs) like Facebook, Twitter, WeiBo become the most popular information transform platform, which catch more and more attention from Information Retrieval (IR). However, traditional web crawling System get into trouble because of the complicated OSN web pages, the rapid message exploding and the heavy using of Asynchronous JavaScript and XML(AJAX). We design and implement a distributed system based on Message Oriented Middleware (MOM) and Ajax simulation, which crawls 70 millions of Twitter detail items in one month. The data Acquisition shows that the crawling with Ajax simulation is able to get items loaded by Ajax without limitations, the distributed system based on MOM and Ajax simulation is able to crawl massive OSN data completely, quickly, frequently and unrestrictedly.

Keywords: OSN, Ajax, Distributed System, Web Spider, Massive data;

1. Introduction

The popularity of online social networks (OSNs) in recent years is continuously increasing. People spend more time on OSNs every day. Twitter, in particular, is one of the most popular OSN, which has 170 million messages and 20 million users totally and 20million new user each year. OSNs play an important role in the spread of media event and emergencies. Therefore, OSNs have attracted extensive attention from research community, including sociology, economics and public security.

The OSNs research is based on the massive data of social network. But the collection of data comes into trouble for the enclosed OSNs, and the OSNs' heavy spreading and using of Ajax make it worse. So there are three ways to reach this goal. The first is using the official Application Programming Interface (API Crawling below), the second is crawling the Ajax data with the web application test framework (WTF Crawling below), and the third is to find the no JavaScript page for crawling (No-JS Crawling below).

* Corresponding author: Shan Jixi.
E-mail address: sdu07@126.com.

doi:10.1016/j.procs.2013.05.105

The above ways can get social network data, but there are the following questions:

- The way of using API limited by crawl times and crawl frequency, can't crawl many times in a large scale;
- The way of using simulation browser is slow, and has more consumption in crawl resource;
- The way of using no JS needs bigger bandwidth, with small valid data scale, long response time and slow crawl speed.

Therefore, those ways can't satisfy the needs of large-scale high-speed crawl.

This paper designs and constructs distributed OSN crawling system based on Ajax simulation. Using message bus to establish control-collection Server-node, the multilayer distributed crawling system realizes dynamic extension of crawl target and system performance. The crawl node through the browser action log and gateway network request record analysis and generate Ajax behavior templates, according to the template to acquire target derived network link request, save the server returns information, and analysis and complete information extraction. Through one month's crawl, the system gets access to 30 million Twitter pushes, the number of push, users and the item of customer relationship reach tens of millions. Compared with Fangweiwei et al.'s [2], Daniele et al.'s[15] existing social network, this system has faster crawl speed, no restricted crawl frequency, lower consumption of calculation resources, and smaller network bandwidth.

This paper is made of the following several parts, the first part present research of social network dynamic content crawl, the second part is crawling based on Ajax simulation, the third part is designing and implementing distributed system based on Ajax simulation, the fourth part is the experimental results and analysis, and the last is summary.

2. Relation Work

OSNs make heavy use of Ajax to load information asynchronously. There are three methods to crawl this Ajax data, the first method is using the official API, the second one is crawling with the web application test framework, the third one is crawling the special page without JavaScript.

The first method is API Crawling. Fangweiwei et al.[2] make a survey of crawling Twitter by the combination of 2 kinds of API and Matko et al. collect Facebook users' relationships with official API[3] [4] too. OSNs develop and supply APIs in order to catch the attention of third-party programmers. We can get the user profiles, user relationship and the users' message easily. Twitter offers Twitter List API and Twitter Lookup API for different condition, so do Facebook and WeiBo. Fangweiwei et al. got 260 thousands messages in 2.6h with Twitter API and Jingyuan Li and Matko Bošnjak collected 5000 messages from Facebook in 22days[4].

The second method is crawling by WTF web application test framework which is used for test website automatically. There are different kinds of framework for different Program Language and browser. For example, Watij (Web Application Testing in Java) [5], Watir (Web Application Testing in Ruby) [6], Selenium [7], HttpUnit [8] are all popular web test framework. WTF makes it possible for us to control browser by program, and we can simulate the way of mouse clicking, form submitting, use input and so on. WTF use the browser to run JS and modifies the web content, which make it possible to collect Ajax information. LIU Shao-bin et al. detect the web by Watij and covers over 98% content [4].

The third method is No-JS Crawling. Gianluca Stringhini finds that all the OSN offer special website for mobile devices which is weak in JS analyzing, and over 80% OSN spiders collect data with these mobile pages [9]. Web spider can get whole page one time without analyzing JS. So crawling the page without JS makes it easier to crawl all OSN information.

The above ways can get social network data, but there are kinds of limitation in the crawling speed, the crawling frequency and the faction of coverage.

For the crawling speed and frequency：Official API is designed for programmer to create useful applications like tools and games in OSN, so the API provider limit calling frequency, result numbers and so on. In Twitter API, the Oauth user can call the API only 60 times per an Hour, stricter for user without Oauth [10]. Crawling

frequencies through the No-JS page is limited by server for the lack of webserver and poor bandwidth for mobile devices.

For the crawling's range. In order to protect users' privacy, OSNs offer little information for API crawler. In Facebook, you can't get the friends list of your friends [11].The page without JS is very simple and crude containing less information than page for PC and we can't get the message time and details in the page for mobile devices.

For the resource consumption, the WTF spends lots of time and resource on page render, advisements information, pictures and UI. It takes Gianluca Stringhini 5 hours to collect 6000 users, which show the low speed for WTF [9].

All in all, the Ajax used in OSNs caused problems for information crawling. Tradition methods like API Crawling, WTF Crawling and No-JS Crawling has limitations such as low speed, strict request frequency, page incomplete and so on. These methods are not suitable for mass OSN data. How to crawl massive OSN data quickly, completely and unrestrictedly is this paper's goal.

3. Crawling Based on Ajax Simulation

Distributed SNS crawling system based on AJAX simulation mainly consists of Distributed crawling system and simulation Ajax behavior crawler.

3.1. Ajax behavior simulation

Ajax is the way that browser request to the http server asynchronously. Ajax simulation is to analysis the browser Ajax behavior, generate activates templates, visit by simulation browser, analyze Ajax requests result and finally simulate the Ajax.

Ajax behavior analysis is the most important step in Ajax simulation. But it contains great difficulty for Ajax behavior analysis in complex social websites. To realize the analysis of Ajax behavior simulation, a traditional method is to check the corresponding web site source codes, analyze JS script action, and find the Ajax request links. This is effective and accurate for simple Ajax web. However, Ajax's behaviors are very complicated in OSNs due to the complexity of social networks. JS file of Twitter Ajax is 112 KB, and Ajax JS file of Facebook is168 KB. What's worse, JS codes are compressed and confused in order to make others analyze difficult, which increases the difficulty of JS analysis.

In order to analyze the Ajax activities accurately, fast, briefly, we propose an analysis method based on combination of browser behavior logging and network gateway request recording. On one hand we use the Log module supplied by Chrome browser developer tools, record XML HttpRequest action, and filter the Ajax request behavior; On the other hand, the whole Http link request is recorded by the local proxy gateway. Then the above two aspects are compared and analyzed and we can find the different between each similar Ajax request with regular expressions. Finally we get the template of Ajax behavior shown as expression 1 below. The working flow is shown by Figure 1.

According to the flow chart we can get Ajax behavior templates easily, and then we use the simulation browser to request to the OSN server with the templates above. Receive and save the response from server, and get the html source code by JSON parser and get strut data with Html analyzer, finally the Ajax behavior simulation is realized.

Ajax behavior simulation is different from API Crawling, WTF Crawling and No-JS page crawling. Ajax simulation can request to the server without frequencies limited and avoid a waste of resources occupation by the heavyweight browser test components in WTF Crawling way. Moreover, simulation Ajax request can get the least size of response without invalid information like recommend information and advertising information, so the Ajax simulation can collect part of OSNs' data frequently and unrestrictedly.

Express 1: the timeline Ajax template

/i/profiles/show/%s/timeline? include_available_features=1&include_entities=1&max_id=%s.

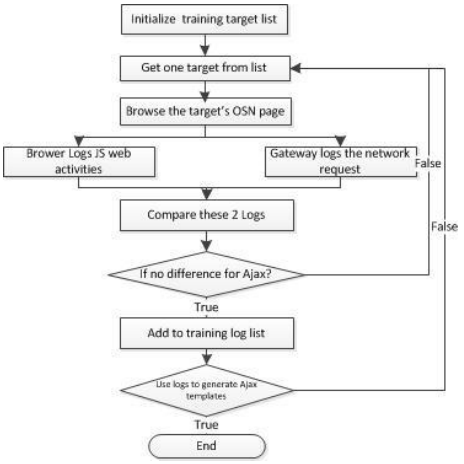

Figure 1: Ajax activates simulate work flow

3.2. Single node crawling algorithm

Input: OSN key user id list

Output: User messages, user relationships, user profiles

Initialize:

1: Initialize the web browser client

2: Initialize the Jason parser for messages, relationships and profiles.

3: Initialize the Html Analyzer.

Begin：

1: Random access the key users' page for messages, relationships and profiles.

2: The browser logger logs the JS activities.

3: The gateway server logs the network request information.

4: If log is enough to generate the Ajax activities template, then go to 5; else go to 1.

5: Get one user from the key user list, and generate the request activities.

6: Request to the OSNs server through the browser client with Ajax templates.

7: Receive and save the response from server, and get the html source code by JSON parser and get strut data with Html analyzer.

8: Put the new user into the new user list and remove the reduplicative ones.

9: If the size of new user list is not increase then end; else go to 5.

4. Design and Implement Distributed System

With the Ajax simulation described by last section, we can detect the OSNs frequently, and freely. However, single-node crawler computing power, storage capacity, and network's bandwidth is limited. At the same time, with the OSNs growing rapidly, the Twitter has 170 billion messages and 500 million more each day. The distribute system based on Ajax simulation is designed and implemented to crawl massive data quickly, completely and unrestrictedly.

The distributed system can make full use of Ajax simulation and overcome the single-node's limitations. To improve efficiency and performance, more nodes will be added in to distributed system dynamically. Message bus will contact them together and dispatch message, and crawler server will schedule the crawling tasks so as to enlarge storage capacity. We use the distributed system to build Mulity-Nodes web cache server used to save the original html code, expand the web bandwidth and avoid IP being banned. Furthermore, we make a proxy-server-pool which is used by nodes as proxy to access the OSN server.

4.1. Design Distributed Crawling System

As surveyed by Oung-Woo et al., there are five kinds of architectures to build distributed system [13]. They are Message Oriented Middleware (MOM), Open Service Gateway Initiative (OSGi), Remote Procedure Calls (RPC), Re-mote Batch Invocation RBI, Socket. OSGI, RPC, RBI，Socket are all Synchronous, poor reliable and peer-to-peer commutation model. However, OSNs can widen varieties, update rapidly and develop complicatedly.

Distributed Crawling system with Mom has three advantages:

- High available, Mom contains error handling module and abnormal recovery module, which improve the availability of OSNs crawling system;
- Asynchronous communication, senders can send message without waiting until it is received;
- Supporting point-to-point communication model and point-to-multipoint communication model, crawler server sends control command with point-to multipoint model and nodes exchange information with point-to-point model. In summary, Mom is suitable for OSN distributed system.

Build 3-level architecture distributed system. We implement a node-crawlServer-control architecture for crawling. With this design, each node are independent from each other which make it easy to expand crawl ability and each crawl-server are independent form each other which makes it easy to add new OSN targets.

Make full use of Mom type. For OSN crawling, control message are transferred by point-to multipoint message bus, and result message are transferred by point-to-point message bus. In Java message services (JMS) [14], crawling server sends messages to Topic message bus and one or more receivers get the message; node sends messages to Queue message bus and only one receiver get the message. Table 1 shows the using of Mom Type in JMS.

Table1. Topic/Queue in MOM distributed system

ID	Type	Description
ControlN2S	Topic	Node sends self-condition information to crawl server.
ControlS2N	Topic	Crawl server sends control information to nodes.
Task	Queue	Crawl server dispatches crawl tasks to nodes.
Statistics.Destination	Queue	Nodes and crawling server can receive some queues status
ControlC2S	Queue	Crawler server sends message to control center;
ControlS2C	Topic	Control center sends messages to crawl servers.

Build the crawling support subsystem. Distributed database, distributed web cache server and proxy-server-pool are all the key points for high speed and effective crawling. We designed different database for different OSN and store them in different server. MongoDB [15], the distributed NoSql [16] document database, is used for the cache. We combine many proxy server with different IP into a proxy-pool which is used for node to avoid being banned and improved the network bandwidth.

Make the distributed system high available. Mass information of OSNs required the distributed crawling system to be stable all the time and the message bus is guaranteed for high speed crawling and large quantities of information. User connects to Mom with the address like this:

Failover :(TCP: //broker1:616161, TCP: //broker2:61616, TCP: //broker3:61616)

Broker1 is the master broker and the broker2, broker3 are slavers. When the master broker is broken down, broker2 or broker 3 get the lock of database and become the master broker. If the broker is recovered, it will work as a slave broker.

As described above, the distributed system is designed and implemented as the figure2 below:

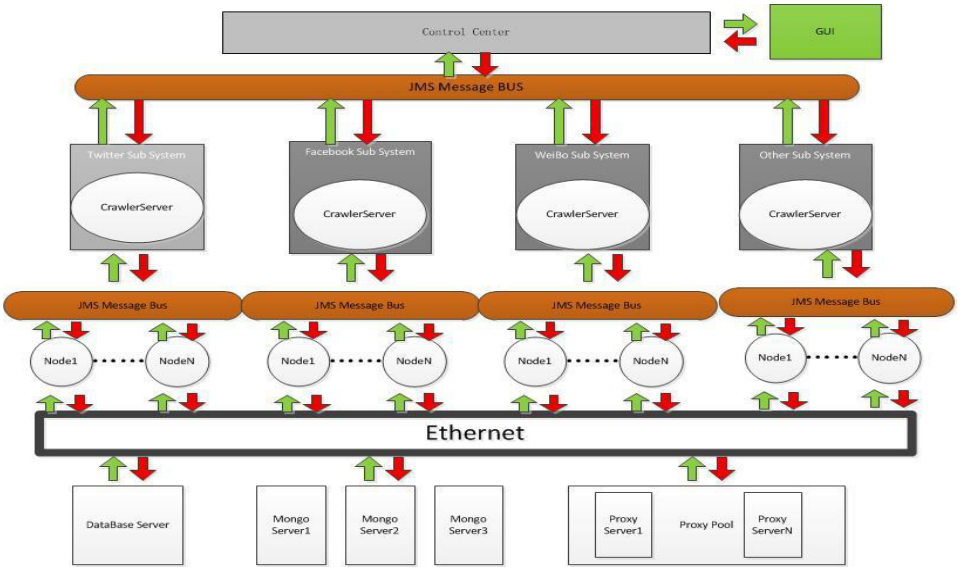

Figure 2: the architecture of OSN crawling system

4.2. Distributed OSN Crawling Algorithm

Input:

List: OSN key user id list

MaxDepth: BFS Depth

Output: User messages, user relationships, user profiles

Initialize:

1: Initialize the web browser client.

2: Initialize the Jason parser for message, relationships and profiles.

3: Initialize the Html Analyzer.

4: Initialize the Task Message Bus and User Message bus.

5: Initialize the variable: *CurrentDepth*.

Begin：

1: Random access the key users' page for message, relationships and profiles.

2: The browser logger logs the JS activities.

3: The gateway server logs the network request information.

4: If log is enough to generate the Ajax activities template, then go to 5; else go to 1.

5: Crawler server generates tasks to Task Message Bus.

6: Nodes get tasks from the message bus.

7: Nodes generate the request activities, request to the OSN server through the browser client with Ajax templates.

8: Nodes Receive and save the response from server, and get the html source code by JSON parser and get strut data with Html analyzer.

9: Nodes send new user to User Message Bus.

10: Server receives the new users

11: If the Task Message Bus is empty and *CurrentDepth* equals *MaxDepth* then go to 12; else if Task Message Bus is empty then go to 5; else go to 11.

12: End.

5. Data Acquisition and Analysis

Distributed OSN crawling system based on Ajax simulation can reach a high crawling speed with low network bandwidth. We design and implement a distributed system with Ajax simulation for Twitter, collect all the information of target users, and reach tens of millions of data. Compare to Daniele Quercia et al.'s study on crawl of Twitter [17], this System shows that Ajax simulation is significantly greater than API crawling. Details are shown as Table2 below.

Table 2. Comparison Ajax crawling system to Daaniel crawling system on the crawl efficiency

	Ajax distributed system 15th Oct-15th Dec		Daaniel 27th Sep-30th	
	Sum(pieces)	Speed（Item/H）	Sum (pieces)	Speed（Item/H）
TimeLine	34381661	47752	31,565,708	14142
User	11169997	15513	240,982	107.
User Relation	22,311,521	30988	10,254,969	4594
Total	67863179	94253	42061695	18844

Comparing distributed crawling system based on Ajax simulation to Daaniel crawling system, it can be known that Ajax crawling system has great advantages in crawl speed and content range. Average crawl speed of Ajax crawling system is five times of Daaniel API's. The reason for such great difference is that distributed crawl of simulation Ajax is totally different from the API crawling in crawl content, speed and frequency. Table3 show these differences among API, No-JS Crawling and Ajax simulation.

We can see the difference from the table3 below. Twitter API limits the request frequency for friends list in 60 times per hour, which means the API crawler can get only get 1200 users in an hour. The Facebook's API is stricter than Twitter's and the crawler can't get the user's friends' friend list. Crawling with No-JS page gets less information than API crawling and Ajax simulation in message detail and follower\following detail.

We compare the Ajax simulation with No-JS crawling and API crawling and find that the Ajax simulation can crawl quickly, frequently and unrestrictedly. Ajax simulation show great advantages in crawl frequency, crawl amount, message detail, and user detail. The more important of all is that the Ajax simulation will get more detailed information in a higher speed without limitation. Obviously the Ajax simulation is the most suitable method for the massive OSN data crawling. Because of Ajax behavior analyzing process, the Ajax simulation is a little more complex than No-JS crawling and API crawling. Perhaps, analyzing through machine learning will make Ajax simulation easier and simpler.

Table 3: Differences among API, No-JS Crawling and Ajax simulation

Method Detail	API	Crawl no-js page	Ajax simulation
Message count per User	Newest 3200	Newest 3200	All the messages of user
Message detail	Most detailed	Very simple information	Detail enough for analysis
Crawl Frequency	720 times per hour	Dynamic limitation	No frequency limitation
Message detect speed	200 pieces per request.	20 pieces peer request	20 pieces peer request
Follower\following count	All the friends in list	Part of friends in list	All the friends in list
Follower\following frequency	60 times per hour	Dynamic limitation	No frequency limitation
Follower\following speed	20 pieces peer request	20 pieces per request	20 pieces per request

6. Conclusion

Comparing with API crawling, WTF crawling and No-JS crawling, the Ajax simulation shows great advantages in OSN detection. The more important of all is that the Ajax simulation can get more detailed information in a higher speed without limitation. We design and implement distributed system to make full use of MOM and Ajax simulation, which crawls 70 millions of Twitter detail items in 30 days. The experiment shows that this Ajax simulation is a new and powerful method for OSNs data crawling. The crawling system based on Ajax simulation can collect the massive OSNs data completely, quickly, frequently and unrestrictedly.

The future research includes: analyze the Ajax behavior by machine learning, extend kinds of crawl method in OSNs and analyze the billions of items of OSNs data.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (61070184), the "Strategic Priority Research Program" of the Chinese Academy of Sciences Grant No.XDA06030200, and National Key Technology R&D Program (No: 2012BAH46B03).

References

[1] Erin Allen Update on the Twitter Archive at the Library of Congress Library of Congress; 2013-01-04.
[2] Fangweiwei, Jingyuan Li. The Research On Twitter Crawl Method.0061 Journal of Shandong University, 1671 9352(2012) 05 00730(in Chinese).
[3] Facebook Developer. https://developers.facebook.com/docs/reference/api
[4] Matko Bošnjak. Twitter Echo - A Distributed Focused Crawler to Support Open Research with Twitter Data; ACM 978-1-4503-1230-1/12/04
[5] LIU Shao-bin, ZHANG Zu-ping, LONG Jun. Research on Watij-based Spider for Deep Web Computer Engineering Feb; 2011
[6] Jiangang Wang. An Empirical Study of Dangerous Behaviors in Firefox Extensions Springer-Verlag Berlin Heidelberg; 2012

[7] Selenium. http://seleniumhq.org/
[8] HttpUnit. http://httpunit.sourceforge.net/
[9] Gianluca Stringhini. Christopher Krueger. Detecting Spammers on Social Networks. ACSAC '10 Dec. 6-10, 2010, Austin, Texas USA; 2010
[10] REST API V1.1 Limits per Window by Resource. Twitter
[11] Zhefeng Xiao, Bo Liu, Huaping Hu, Tian Zhang. Design and Implementation of Facebook Crawler Based on Interaction Simulation. 2012 IEEE DOI 10.1109/TrustCom; 2012.121
[12] Minas Gjoka, Maciej Kurant. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. IEEE INFOCOM; 2010
[13] Oung-Woo Kwon, Eli Tilevich, William RCook. Which Middleware Platform Should You Choose for Your Next Remote Service? Service Oriented Computing and Applications Volume 5, Issue 2, pp 61-70; 2010
[14] ActiveMQ: http://activemq.apache.org/how-does-a-queue-compare-to-a-topic.html.
[15] MongoDB. www.mongodb.org/
[16] Haihong, Guan Le, Jian Du. Survey on NoSQL database. Pervasive Computing and Applications (ICPCA); 2011
[17] Daniele Quercia .The Social World of Twitter: Topics, Geography, and Emotions, Sixth International AAAI Conference on Weblogs and Social Media; 2012