talenta PUBLISHER

DATA SCIENCE
Journal of Computing and Applied Informatics

Improving Data Collection on Article Clustering by Using Distributed Focused Crawler

Dani Gunawan[1]*, Amalia[2], Atras Najwan[3]

[1,3]Department of Information Technology, Universitas Sumatera Utara, Medan, Indonesia
[2]Department of Computer Science, Universitas Sumatera Utara, Medan, Indonesia

Abstract. Collecting or harvesting data from the Internet is often done by using web crawler. General web crawler is developed to be more focus on certain topic. The type of this web crawler called focused crawler. To improve the data collection performance, creating focused crawler is not enough as the focused crawler makes efficient usage of network bandwidth and storage capacity. This research proposes a distributed focused crawler in order to improve the web crawler performance which also efficient in network bandwidth and storage capacity. This distributed focused crawler implements crawling scheduling, site ordering to determine URL queue, and focused crawler by using Naïve Bayes. This research also tests the web crawling performance by conducting multithreaded, then observe the CPU and memory utilization. The conclusion is the web crawling performance will be decrease when too many threads are used. As the consequences, the CPU and memory utilization will be very high, meanwhile performance of the distributed focused crawler will be low.

Keyword: Data collection, CPU utilization, Distributed web crawler, Distributed focused crawler, Focused crawler, Memory utilization, Multithread, Web crawler.

Abstrak. *Pengumpulan data dari Internet sering dilakukan menggunakan web crawler. Web crawler umum dikembangkan untuk menjadi lebih fokus pada topik tertentu. Jenis web crawler ini dinamakan focused crawler. Peningkatan kinerja pengumpulan data tidak cukup hanya dengan menggunakan focused crawler, karena focused crawler membuat penggunaan pita lebar jaringan dan kapasitas penyimpanan data menjadi lebih efisien. Penelitian ini mengusulkan sebuah focused crawler terdistribusi yang ditujukan untuk dapat meningkatkan kinerja web crawler dan juga efisien dalam penggunaan pita lebar jaringan dan kapasitas penyimpanan data. Focused crawler terdistribusi ini mengimplementasikan penjadwalan crawling, urutan situs untuk menentukan antrian URL, dan focused crawler menggunakan Naïve Bayes. Penelitian ini juga menguji kinerja web crawling dengan cara menggunakan multithread yang kemudian dilakukan pengamatan penggunaan CPU dan memori. Kesimpulan penelitian ini adalah kinerja pengumpulan data akan menurun jika terlalu banyak thread yang digunakan. Sebagai konsekuensinya, penggunaan CPU dan memori akan menjadi sangat tinggi, sementara kinerja distributed focused crawler akan menjadi rendah.*

Kata Kunci: *Pengumpulan data, Web crawler terdistribusi, Focused crawler terdistribusi, Focused crawler, Multithread, Utilisasi CPU, Utilisasi memori, Web crawler.*

Received 03 April 2017 | Revised 04 May 2017 | Accepted 13 June 2017

*Corresponding author at: Department of Information technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jalan Alumni No. 9 Kampus USU, Medan 20155, Indonesia
E-mail address: danigunawan@usu.ac.id (Dani Gunawan).

ISSN: 2580-6769 | e-ISSN: 2580-829X | DOI: 10.32734/jocai.v1.i1-82
Journal Homepage: https://talenta.usu.ac.id/JoCAI

1. Introduction

The big picture of the research is summarizing multi-articles in Bahasa Indonesia from online newspaper automatically based on certain category. This research is a part of the big research, which has role to automatically harvest the articles from the online newspaper publishers through their websites. The articles lie on the web pages of each online newspaper publishers' websites. The first research to be conducted is harvesting articles from the Internet. This first research is important to obtain fundamental research data to be used for the next researches.

The research about crawling web pages for health articles in Bahasa Indonesia has been conducted previously by Amalia [1]. They implement site ordering algorithm, multi-threaded crawler and uses Naïve Bayes algorithm to build the focused crawler engine. However, they harvest the articles from single computer only. Amalia does not specify the performance of their crawler in CPU and memory utilization. They only focused on the performance of the crawler to harvest the specific topic, in their case: health. This kind of crawler is proposed by Chakrabarti [2]. The performance of focused crawler is better to find topic-specific web pages compared to general crawler.

Previously Bal [3] proposed client server architecture based smart distributed crawler to crawl the web pages. In Bal architecture, the server manages the load of a crawler to be distributed to the others by dynamically distributing the URLs. Bal stated that focused crawler makes efficient usage of network bandwidth and storage capacity, meanwhile distributed crawler can enhance the performance.

In 2013, Achsan [4] conducted a research about the usage of multithread web crawler in a computer which will distributed to public proxy server. They observed that if the crawler does not use proxy server, the web crawler will be considered as "cyber-attack" and will be banned by the web server. They concluded that multithreaded web crawler which has been distributed to the public proxy server is the easier and cheaper way than using the distributed web crawler. There is a drawback in this method is lack of coordination among the computers if they want to expand the crawler to be more than one computer.

This paper is organized as follows. In section 2, we the research methodology used to develop distributed focused crawler and the testing method. In section 3, we explain the result and discussion about our findings. Finally, we draw some conclusions in section 4.

2. Research Methodology

Harvesting the web pages from the Internet is commonly known as crawling. A crawler might collect anything which lies in the Internet. One of the important part of the search engine is a crawler. It traverses the hyperlinks in the web pages and download them while traversing [5].

Thus, the web crawling activity is similar with graph traversal. There are several types of web crawlers such as hidden web crawlers, incremental web crawlers, parallel crawlers and focused crawlers/topical crawlers/topic driven crawlers.

In this research, we proposed a method to improve harvesting the articles from the online newspaper publishers. The method called distributed focused crawler. Figure 1 illustrates the architecture of the distributed focused crawler.

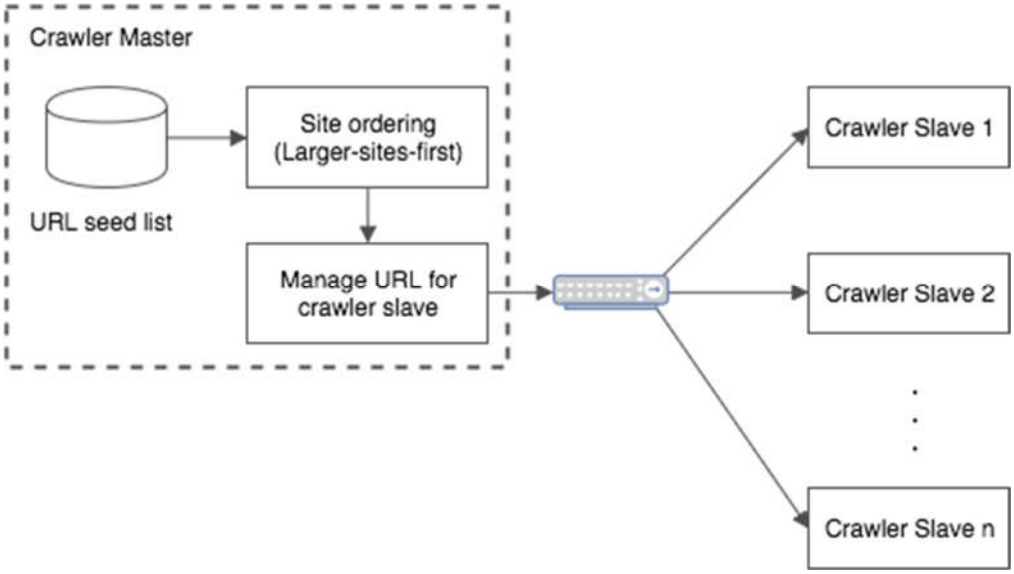

Figure 1. Distributed Focused Crawler Architecture

2.1 Distributed Focused Crawler Configuration

As shown in Figure 1, a distributed focused crawler is divided into two parts. The first part is crawler master and the second part is crawler slaves. The crawler master is used to configure database such as hostname, password, database name and so on. All the database-related operation is executed in crawler master. The crawler master also configures the distributed mode such as adding the crawler slaves IP addresses. Furthermore, crawler master also takes responsibility to provide storage to save downloaded web pages. It also configures URL depth to limit link traversing. Crawler master can also use proxy to access the web pages. It should check robot.txt in every website to make sure the crawler slave subjects to the website rules. The crawler master also implements larger-site-first algorithm to crawl websites based on site ordering. It determines the number of threads and specific topic to be downloaded. It does filtering URL, contents and minimum term to be visited and determine crawler lifetime in a web page.

The second part of the distributed crawler is the crawler slaves. This crawler slaves are set of several computers which have a main task to download web pages based on provided URLs by crawler master. We use the same type and specification for the crawler slaves. This is not mandatory as the distributed system might use any type of the computer. We use the same type and specification for the crawler slaves to obtain ideal crawling result.

2.2 Topology

To provide networking among the computers we set our distributed crawler environment in star topology. Although the real distributed-system does not need the computers to be in the same network, we set the computers in the same network and controlled environment. This setting is used to provide the most ideal results. However, the clients are able to directly connect to the Internet without having obligation to pass the request or response via the crawler master. This will ensure that the crawler slaves have full access to the Internet. By implementing this configuration, the result is expected to be as ideal as possible.

2.3 Crawling Scheduling

As shown in Figure 1, the distributed crawler use URL seed list to keep the address of the URLs that will be crawled. At the initial condition, distributed crawler will start the process with small number of URLs. The number of the URL seed list will grow as more links extracted from the web pages.

According to Figure 1, distributed crawler use a crawler master to help managing URL seed list. Crawler master is a dedicated computer which act as a coordinator to schedule crawling activity. This coordinator can communicate with other computers called crawler slave (shown in Figure 1). One of the coordinator functions is to manage the URL seed list which will be passed to the crawler slave. For each crawler slave, there is also a coordinator to manage the URL seed list for its available thread. This means the coordinator does not directly assign a URL to be crawled by a crawler slave's thread. The crawler slave also has its own coordinator to schedule the crawling activity.

There are some crawler strategies to determine page ordering to download web page according to the URL in the queue list. This activity is known also as crawling scheduling. Those page ordering algorithms are Breadth First Search, Depth Focused Crawler and Larger Sites First Algorithms. Breadth First Search algorithm generally focus on the objective lies in the depthless parts in a deeper tree [6]. Depth Focused Crawler start the crawling activity at the root URL and traverse depth through child URL. The third is larger sites first algorithm which is used to determine the crawling sequence based on the number of pending pages. It will crawl the site which has the larger number of pending pages first [7].

Ricardo Baeza-Yates et. al [7] conduct a research about the most effective algorithm to be used as page ordering in web crawler. According to them, Breadth-first algorithm has worse performance compared to other algorithms. Another algorithm, On-Line Page Importance Computation (OPIC), requires more calculating time than larger-sites-first. It does not need the weight in the link like the OPIC does. As the consequences, larger-sites-first algorithm sorts websites which will be crawled based on number pages in descending order.

As the larger-sites-first algorithm has better performance in computation time than similar algorithm, this research will util ize larger-sites-first algorithm to handle the URL seed list scheduling for both single-computer and distributed-computer crawler. Besides, this method will avoid many pending-page in the websites [8]. This strategy is aiming to download the important page first. The measurement of importance level of a web page is done with the Pagerank algorithm [9].

2.4 Web Crawling

Article in a web page usually consists of several media such as text coded with HyperText Markup Language (HTML), JavaScript code for client-side scripting, cascade style sheet (CSS) for web page styling, images, sounds, and video. As we only need the text of the articles, the crawling only focused on the text in the web pages. The other media will be removed. To filter unwanted media in a downloaded web page, we did several steps such as fetching, parsing and filtering.

A. Fetching

Fetching is an activity to download the whole document in a web page based on URL seed list. In this step, the downloaded web page usually only consists of text-based page in HTML format, JavaScript, CSS and image files (Both JavaScript an CSS in external files). However, as we do not deal with the other media other than text-based HTML page, we exclude JavaScript and CSS external files, common image media file such as JPG/JPEG, GIF, PNG and so on. As the result, the downloaded page is in HTML format.

B. Parsing

A web page usually has hyperlinks to connect its page with the others. We need hyperlinks in a web pages to be kept in our URL seed list in order to provide another page to be crawled. There hyperlinks buried in the downloaded page. The result of downloaded page will consist of many HTML tags if opened with a text editor. We only need to get all links from the page. To crawl the links, we can parse the link tag “a” from the downloaded page. The result of this action is a list of URLs from a web page.

C. Filtering

As explained before, there are tags that mark the text in order to make the text viewed properly in the browser. As well as automatic text summarization [10], to cluster the articles, the markup tags are useless because the required input is only the plain text without any markup. The other things to concern is the structure of a web page. A web page usually consists of a few areas, such as header, navigation, sidebar (left and right), content and footer. As we only need the main article, the other area should be removed. To achieve the main articles, we implement boilerplate algorithm [11] which detect the main content of a web page. The boilerplate algorithm does not

require any training or inter-document knowledge such as frequency of text block, common page, layout and so on.

D. Content Extraction

After obtaining the main article from a web page, the next step is extracting the content. The main article extracted by using the boilerplate algorithm only separating it to the others area of a web page. It still contains the HTML tags. As the boilerplate does not clean the HTML tags, we need to do further step to extract the plain text from the main article. This step is mandatory because the clustering input requires plain text.

E. Focused Crawling

To avoid harvesting web pages in various topics, the crawler is designed to crawl only certain topic. This kind of crawler called focused crawler. This condition can be achieved by classifying the crawled articles by using some algorithms such as Naïve Bayes, Support Vector Machine, and K-Nearest Neighbors [12]. In this research we utilize Naïve Bayes algorithm to provide the classification function.

F. Crawler Data

In multi-threaded single computer, each threads saved the crawled data in the local database. Meanwhile, we adopt the thin-client architecture for the multi-threaded distributed computer. Thin-client architecture put the load in the server to keep the client as thin as possible. This means threads in each crawler slave did not save the crawled data in the local database. The data are passed directly to the coordinator computer. The crawler slave does not keep any data. This will keep the crawler slave thin because the data are saved in the crawler master computer.

G. Data Source

A crawler requires initial URL seed list to begin crawling the web pages. This list usually is small in the beginning. Later after the crawler traverses the web pages, the URL seed list will grow as much as the available links in the web pages. Initially we use 32 URLs as our URL seed list. The example of the first ten URLs is shown in Table 2.

H. Testing

We observed the crawling result conducted by the distributed crawler in certain parameters. The web pages will be crawled in several controlled conditions such as bandwidth, number of threads, and page ordering algorithm. The page ordering algorithm used is larger-sites-first (LSF). The crawling time is done for 60 minutes for every testing condition. By testing the crawling activity in several conditions, we observed the most suitable condition for the maximum result. Table 1 is the controlled conditions as the scenario for testing the distributed focused crawling.

Table 1. Testing Scenario

Number of Threads	Bandwidth (Mbps)	With LSF	Without LSF
100, 200, 500, 1000, 2000	*2*	*Yes*	*Yes*
100, 200, 500, 1000, 2000	*3*	*Yes*	*Yes*
100, 200, 500, 1000, 2000	*5*	*Yes*	*Yes*

3. Result and Discussion

In this section we discuss about the result of crawling with using Larger-sites-first as page ordering algorithm and without using page ordering algorithm. The result also shows the influence of the number of thread, memory and bandwidth usage for distributed crawler to download web pages. Furthermore, we also discuss about CPU and memory utilization for all scenarios.

3.1 Hardware Specification

We use one computer for crawler master and four computers which have the same specification for the crawler slave. The hardware specification for crawler master and crawler slaves are as follow:

A. Crawler Master

- Processor Intel Core i5-3450 3.10 GHz
- RAM 4 GB
- Storage 250 GB
- MySQL database version 5.6.24

B. Crawler Slave

- Processor Intel Core i5-4150 3.5 GHz
- RAM 2 GB

3.2 Site Ordering

This section shows the result of site ordering using larger-sites-first by crawler master by calculating the number of links in a web page. Then the number of links in crawled web pages will be sorted from the in descending order. If the URL cannot be visited or the request for the URL reaches timeout, then the URL will be marked as link=0 or does not has link. As the consequences, the URL will be put in the bottom of the URL seed list. After sorting the initial URL seed list, the new list will be kept in URL seed list, as shown in Table 2.

Table 2. The Snippet Result of Site Ordering using Larger-Site-First Algorithm

Before LSF	After LSF
http://www.informasikesehatan.my.id/	*http://health.detik.com/*
https://anakbayibalita.wordpress.com/	*http://www.vemale.com/tags/kesehatan-anak/*
http://www.posyandu.org/	*http://bidanku.com/*
https://www.klikdokter.com/	*http://www.depkes.go.id/*
http://bidanku.com/	*https://www.klikdokter.com/*
http://www.vemale.com/tags/kesehatan-anak/	*http://dechacare.com/*
http://www.depkes.go.id/	*http://www.posyandu.org/*
http://dechacare.com/	*http://duniaanak.org/*
http://duniaanak.org/	*http://www.informasikesehatan.my.id/*
http://health.detik.com/	*https://anakbayibalita.wordpress.com/*

3.3 Crawling Performance

We conducted several testing scenario including the usage of site ordering, different number of threads and bandwidth. As shown in the Table 1, the crawling was tested several times including testing with different number of threads for each crawler slave. We also tested the difference between using larger-sites-first (LSF) and no site (None) ordering implementation. The result in the Table 3 shows the number of files downloaded from the web pages.

Table 3. Summary of Crawling Result

Threads	2 Mbps		3 Mbps		5 Mbps	
	LSF	None	LSF	None	LSF	None
100	*26,749*	*25,192*	*28,884*	*27,452*	*28,414*	*27,452*
200	*26,771*	*26,886*	*29,168*	*27,460*	*29,448*	*27,460*
500	*27,198*	*27,288*	*29,512*	*28,434*	*29,955*	*28,434*
1000	*22,008*	*22,735*	*26,967*	*26,274*	*24,663*	*26,274*
2000	*13,575*	*13,420*	*10,183*	*8,284*	*8,553*	*8,284*

As shown in Figure 2, sites ordering does not influence the data harvesting significantly. In several testing conducted, there is no significant changes between the crawlers which implement larger-site-first algorithm or not. This condition occurred in all scenario of crawling method.

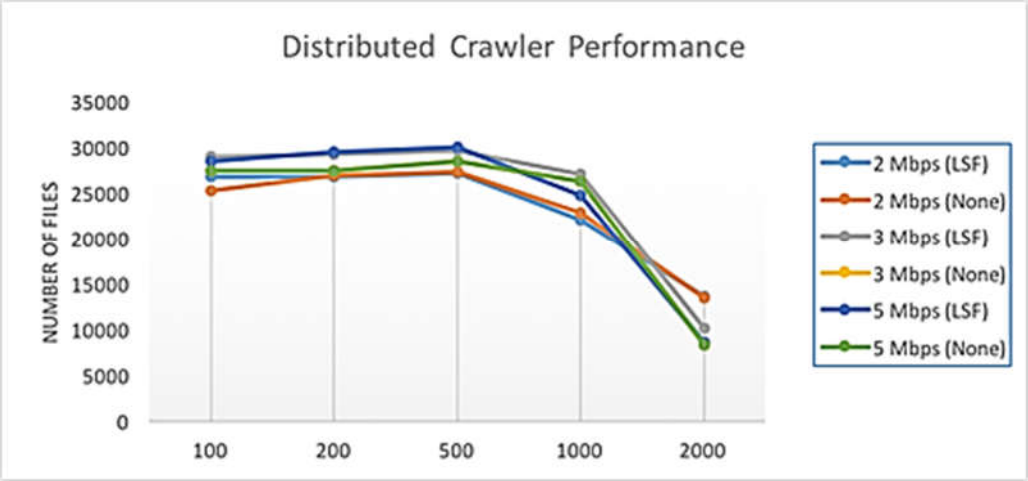

Figure 2. Distributed Crawler Performance

Furthermore, another fact that we found is the bandwidth for all scenario did not have any significant effect to the result of the downloaded web pages. According to Figure 2, the crawling result shows the increment from the 2 Mbps to 3 Mbps. The result is not moving too far for bandwidth 5 Mbps.

Moreover, although the theory that tells having many parallel task will increase the speed the execution time, our research tells the opposite. The crawling result will continue to raise until the number of threads reach five hundred tasks. The performance of the crawler which uses 1000 thread and above will be decreasing. Thus, having as many as possible parallel task does not mean the performance will be raise. We also notice that the increasing of the number of threads from 100 to 200 will not double the performance. This might happen because of the limited number of internet connection allowed to a computer. Because we have tried to raise the assigned dedicated bandwidth to the network and the result did not raise significantly as we mentioned before.

3.4 CPU and Memory Utilization

Heap memory and CPU usage are calculated every minutes. Based on our observation, for crawler slaves almost have the same utilization value for both CPU and memory. Thus, for every thread scenario, we calculate the median (of CPU and memory utilization) among four crawler slaves which have the same thread.

As shown in Figure 3, the CPU utilization is below 10% for the crawler with 100 and 200 threads. For the crawler with 500 threads, CPU utilization raised between 10-20% for about 30 minutes. Then it goes down below 10% in the same level with crawler with 100 and 200 threads. Meanwhile, the crawlers with 1000 and 2000 threads use about 50-60% CPU utilization. According to the previous result in Figure 2, the performance for the crawler with 1000 and 2000 threads are below the performance of the crawlers with 100, 200 and 500 threads. Although the Figure 3 and Figure 4 only shows the CPU and memory utilization of the crawlers with LSF and

2 Mbps bandwidth, this pattern also occurred in all scenario of our distributed crawler. Thus, the implementation of 1000 and 2000 threads requires a high cost of CPU utilization.

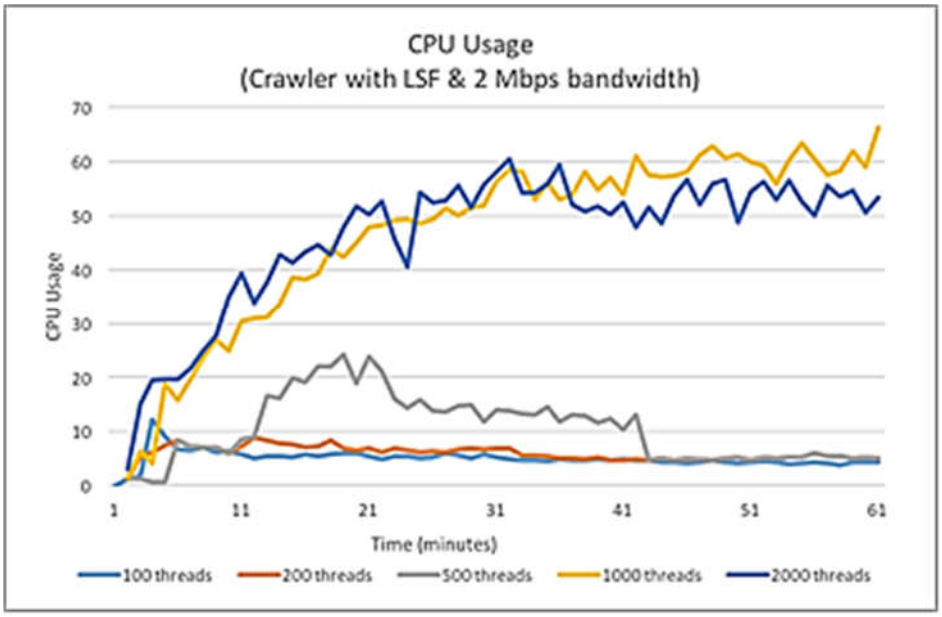

Figure 3. CPU Usage

Figure 4 shows the memory utilization of the distributed crawler with LSF and 2 Mbps bandwidth. The graph shows that the crawler with 100 to 1000 threads almost use the same amount of memory. They used between 300-400 MB memory. Meanwhile, the crawler with 2000 threads spent higher amount of memory allocation. It spent between 400-500 MB memory. This pattern also occurred in all scenario of our distributed crawler. Thus the usage of 2000 threads requires a high cost of memory utilization.

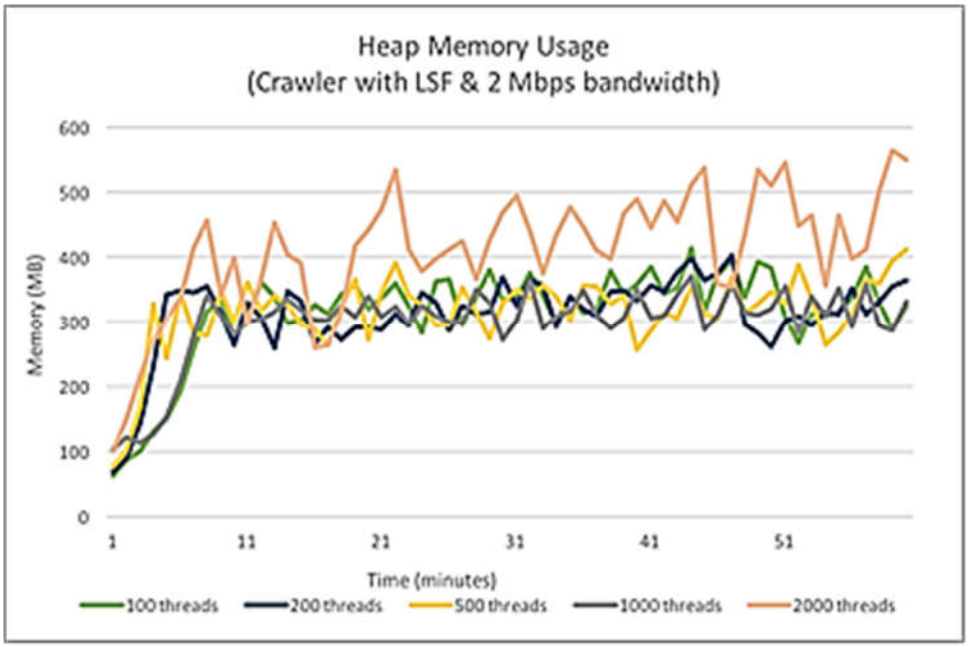

Figure 4. Heap Memory Usage

4. Conclusion

Crawling is a way to harvest data from the Internet. The plethora of data available in the Internet will force to think how to harvest most of them in minimum time required. A single computer will not able to do the job efficiently. Although a computer might utilize parallel processing in harvesting the data, we need another way to improve more crawling result. This research utilizes a computer as a crawler master and four crawler slaves with the same specification. Each crawler slave utilizes parallel processing to crawl the web page. We conduct several scenarios such as 100, 200, 500, 1000 and 2000 threads. We also combined them with the usage of site ordering algorithm, larger-site-first (LSF). To observe the bandwidth influence in our distributed system, we put the distributed crawler in dedicated network with 2, 3 and 5 Mbps of Internet connection. The result shows that the number of crawled web pages did not increase significantly if we double the number of thread. The maximum number of thread before the crawler performance drops is 500 threads. Crawlers with more than 1000 threads show unexpected result, where the number of downloaded web pages is very low, although the bandwidth has been set to 5 Mbps.

Furthermore, crawlers with more than 1000 threads are highly cost for CPU utilization. They use about 50-60% CPU utilization. This waste computer resource because there is no significant result as the return. As well as CPU utilization, memory usage also shares the same pattern. Although crawlers with 1000 threads spent the same amount of memory with the 100-500 threads crawlers, but the performance is below average. Moreover, crawlers with 2000 threads spent about 400-500 MB with insignificant result. Thus, the number of crawlers do not guarantee the result will directly proportional. To conclude, according to our observation and scenario, the optimum number of thread is 500 threads.

Acknowledgments

The authors gratefully acknowledge that the present research is supported by Ministry of Research and Technology and Higher Education Republic of Indonesia. The support is under the research grant TALENTA USU scheme PENELITIAN DOSEN MUDA of Year 2017 Contract Number 313/UN5.2.3.1/PPM/KP-TALENTA USU/2017.

REFERENCES

[1] A. Amalia, D. Gunawan, A. Najwan and F. Meirina, "Focused crawler for the acquisition of health articles," in *2016 International Conference on Data and Software Engineering (ICoDSE)*, Denpasar, 2016.

[2] S. Chakrabarti, M. v. d. Berg and B. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," 1999.

[3] S. K. Bal and G. Geetha, "Smart distributed web crawler," in *Information Communication and Embedded Systems (ICICES), 2016 International Conference on*, Chennai, 2016.

[4] H. T. Y. Achsan dan W. C. Wibowo, "A Fast Distributed Focused-Web Crawling," *24th DAAAM International Symposium on Intelligent Manufacturing and Automation,* p. 492 – 499, 2013 .

[5] S. Sharma and P. Gupta, "The anatomy of web crawlers," in *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, Noida, 2015.

[6] C. Catillo, M. Marin, A. Rodriguez and R. Baeza - Yates , "Schedulling Algorithm for Web Crawling," Riberao Preto, 2004.

[7] R. Baeza-Yates, C. Castillo, M. Marin and A. Rodriguez, "Crawling a Country: Better Strategies Than Breadth-first for Web Page Ordering," in *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, Chiba, 2005.

[8] W. Wang, X. Chen, Y. Zou, H. Wang and Z. Dai, "A Focused Crawler Based on Naive Bayes Classifier," in *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, Jinggangshan, 2010.

[9] L. Page, S. Brin, R. Motwani and T. Winograd, "The Pagerank citation algorithm: bringing order to the web," Stanford Digital Library Technologies Project, 1998.

[10] D. Gunawan, A. Pasaribu, R. F. Rahmat and R. Budiarto, "Automatic Text Summarization for Indonesian Language Using TextTeaser," *IOP Conference Series: Materials Science and Engineering,* vol. 190, no. 1, 2017.

[11] C. Kohlschütter, P. Fankhauser and W. Nejdl, "Boilerplate Detection using Shallow Text Features," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, New York, 2010.

[12] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.