Efficient Multi-threaded Crawling Using In Memory Data Structures

Mohammad A.R. Abdeen,

The Faculty of Computer and Information Systems
The Islamic University of Madinah, Madinah, Saudi Arabia

Summary
Crawling the internet is an important task for any search engine. A crawler is a software program that sends HTTP requests to various webservers available on the world datasphere and downloads their contents. As the size of the internet has gone through a big bang in the last decade, designing efficient parallel crawlers became a necessity. One of the factors that degrades the crawler performance is the disk access every time a file is written. As the process of crawling the web requires the download of tens or hundreds of millions of webpages, much time will be consumed in disk writes due to the seek times. This work presents an efficient multi-threaded crawler that incorporates an in-memory data structure to reduce the overall disk write times. The results show that the proposed technique can increase the throughput by about 50% at selected values of size of the in-memory data structure over the normal multi-threaded crawler with no in-memory data structure. In addition, the results show that this design can achieve an average crawler speed of 22 pages/sec which supersedes previously reported work.

Key words:
Web Crawlers, Distributed Applications, Multi-threading, In-memory Data Structures, Performance Evaluation.

1. Introduction

The size of data in the cyber space has increased dramatically in the last decade by unprecedented amounts. According to an IDC report, the global datasphere is expected to grow from 33 Zettabyte (ZB) in 2018 to mind blowing 175 ZB by 2025 [1]. This is essentially due to the widespread of personal digital devices such as smart phones, iPads, and Laptops and the emergence of ubiquitous computing. Web information mining is an important research area as it addresses the various methods to dig out important information from the massive amount of data existing and increasing by the second.

Web search engines are being used by the majority of the world population. They are now the destiny of just about any person interested in finding information in any knowledge area. Search engines consist of four main components; the crawler, the indexer, the ranking component and the query engine [2]. The crawler is the component responsible for collecting and storing data from the web datasphere, the indexer is the component that creates an index of the collected data for easier and faster future retrieval, the ranking module is responsible for sorting the resulting webpages of the search based on a predefined criteria, and the query engine is the user interface provided to the user to facilitate finding the desired information. A web crawler is the first line of the search engine that interacts with the outside world and is the first stage to process the massive amounts of data from the world datasphere as it collects and stores them in local storage and suitable data structures. The process of collecting the data is called "crawling" and it is the front end of web data collection.

2. Background

2.1 Web Crawlers

A web crawler, also called (ro)Bots, or Spider, is a software program that accesses various available websites in the visible web by sending HTTP requests and collects and stores the response (in the form of a webpage contents) in local disks, data structures or databases. Fig. 1 below shows a simplified structure of a typical web crawler.

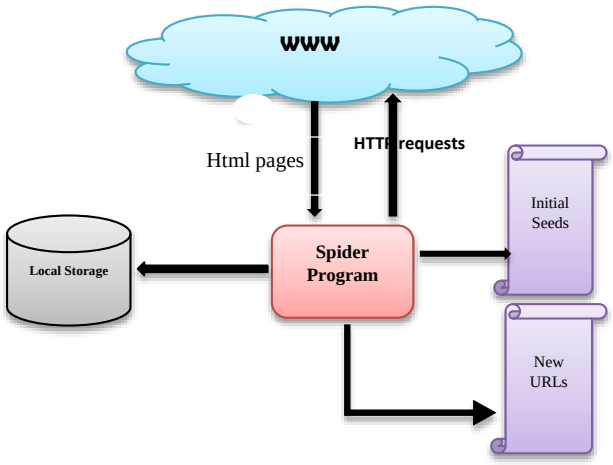

Fig. 1 A simplified web crawler architecture

Manuscript received February 5, 2020
Manuscript revised February 20, 2020

As shown in the figure, a crawler uses initial webpages to start the data collection process. The initial webpages are called the "seed" pages. The crawler program uses those initial pages to collect all available sublinks and expand the list of URLs to be crawled, thereby expanding the data zone to new webpages and new subject areas. There exist two approaches to traverse the web graph and collect its data. In addition, there exists sequential and parallel and multi-threaded versions of the web crawlers.
Web crawlers have been developed with the emergence of the world-wide-web. The first crawler patch date back to early 90s. Early versions of crawlers include RBSE spider [4], World Wide Web Warm [5], and WebCrawler [6]. WebCralwer is considered to be the first parallel available crawler as it is able to download 15 links simultaneously. In 1998, Brin and Page introduced a new large-scale web crawler, called Google. Other versions of web crawlers exit and are listed in [8].

2.1.1 Depth first crawling

The world wide web is considered a massive graph with webpages representing the nodes and the URLs included in those pages as the outgoing links. One way to travers this graph and collect the data of various webpages is the depth-first search approach (DFS). With this approach an initial webpage is picked from the available seed files. The sublinks (a.k.a. the URLs) of this page are considered new nodes and further traversal is started using one of those sublinks. This process is repeated until the full graph is exhausted. Since the size of world-wide-web is massive and complete exhaustion take a significantly large amount of time and might not be attainable, a threshold depth can be set to simplify the search algorithm and to suit experimentation purposes. This approach is adopted in this work.

2.1.2 Breadth first crawling

In this algorithm (BFS) a web graph is traversed in a way such that adjacent nodes are visited first and their data are collected. As an example, if we start with a seed file, then all URLs listed in this seed file are traversed first and their data are collected. Thereafter, the sublinks of the first URL in the list are traversed and their data are collected, and so on.

2.1.3. Multi-threaded crawling

Multi-threading is a technique used to in software application to enhance the throughput. It has been used in numerous applications such as web browsers, web servers, and computer games.
In [9] the authors proposed a multi-threaded crawler for a large-scale text database-Wikipedia crawling. They combined the depth first and the breadth first algorithms to optimize the crawling process.
In [10] the authors presented the largest publicly available dataset to date. To perform this task, they used 100 machines that run a Hadoop platform and distributed the crawling task among them. With this architecture they achieved a rate of about 7.55 pages/sec of download. This result is below expectations as Hadoop is not efficient due to the nature of patch processing it uses.

2.1.4 Disk drive speeds and In-memory structure

Disk drive speeds are typically measured in milliseconds while memory write times are in nanoseconds. This means that the memory writes are 100,000 folds better than the disk drives. The crawler downloads tens or even hundreds of millions of pages per crawl. For this reason, minimizing the number of disk writes is expected to significantly improve the crawler performance as the hard drive are usually the performance bottleneck is distributed crawler systems. In addition, the hard drive possesses a seek time for every disk write. A typical disk seek time is 9 milliseconds. Therefore, reducing the number of writes associated with every new downloaded file is expected to significantly increase the crawler performance.
Previously, data processing in-memory was prohibitive despite its high speed compared to other storage devices such as hard drives. This was mainly due to the high cost of RAM that prevented data to be loaded directly to memory for processing. Recently, the price of RAM has significantly dropped, and it became economically feasible to load and process data directly from memory [13]. There have been serval recent applications for the use of in-memory data structure in caching and in performance improvement of databases [14].
In this paper, we leverage on the RAM speed and its availability in large size to build a performance efficient parallel crawler and by reducing the number of disc access times and perform patch write processing.

3. Design and Implementation

The crawling algorithm presented in this work is a multi-threaded crawler. The algorithm is implemented on a dual-processor HP ProLiant DL380 Gen 10 server with each processor having 12 cores and with 64 GB of memory. The crawler is implemented in Java language. It is to be noted that this threading technique provides true parallelism as opposed to the concurrency as each crawling instance is assign a separate thread and a separate CPU core. The novelty of this algorithm is that it employs in-memory data structure to reduce the amount of time the disk is to be accessed. Fig. 2 shows the proposed multi-threaded crawler with in-memory data structure. In this work, we have used a linked list as its size can expand as the number of files

increase. The symbols F1, F2, …, Fn denote the memory blocks allocated to store the downloaded files (webpages). The complete data structure is written to memory when the number of files reach a specific threshold TF. The allocated memory is returned to the heap by the automatic garbage collection as soon as the memory is flushed to disk.

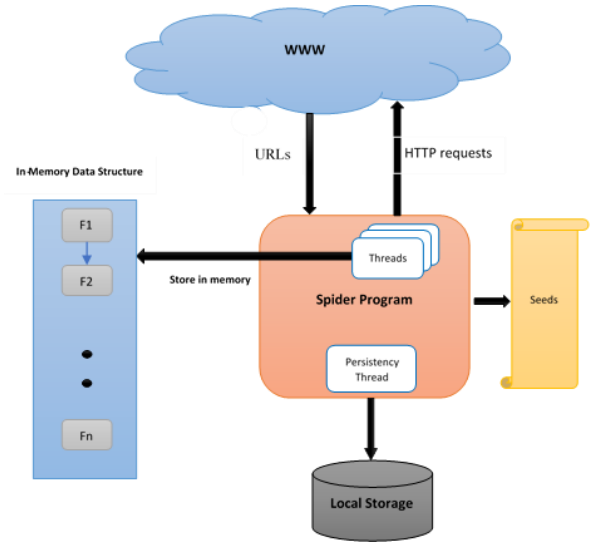

Fig. 2 The Multi-threaded Crawler with In-Memory Data structures

The following listing shows the algorithm of the multi-threaded crawler that uses in-memory data structure. We have used a linked list data structure as it can expand as much as webpages we download. We setup our crawler to stop traversal of the graph when the depth reaches a certain threshold depending on the amount of information we desire to accumulate. For the purpose of this study, this threshold is set to 10,000 URLs.

1. Create a linked list MT_LL
2. Initialize Depth to zero
3. Initialize $T_F = n$
4. Initialize Threshold to 10,000
5. For each item in the seed file list:
6. Read next webpage:
7. Extract HTML contents of the page
8. Create a Node in MT_LL
9. Store page content in new Node
10. Extract URLs of this page
11. Update Depth by the number of URLs
12. collected
13. If length of MT_LL = = T_F
14. Then flush MT_LL to disk
15. If time = 8 hours
16. END
17. Else If number of URLs equals Threshold,
18. Go-to 4.

4. Experiments and Results

As this paper is part of a larger project to mine and classify the contents of the Arabic web [3], the experimentation we performed here is to crawl the Arabic web and store its contents in various categories. We have used 10 categories. These include, tourism, religion, sports, education, IT, automobile, economics, health, culture, and recruitment. For each category we prepared a carefully selected group of webpages to serve as seeds for each category. For the multi-threaded crawler version, we ran 20 threads, each thread on one core (since we have a total of 24 cores). In this case, each category of the 10 categories has two working cores that collect the relevant information and is assigned two different seed files for each category to guarantee mutual exclusion.

We ran 10 experiments to measure the system throughput. The throughput measure we used in this work is the average total download size per minute. The measurements are taken over an eight-hour period. The performance measurements are done using the Microsoft Windows performance monitoring tool that is available for MS-Windows server installed on the machine being used in this work. When the crawl process starts, collected files are stored in a linked list data structure instead of directly writing to the hard drive. This process continues until the size of the linked list (and consequently the number of downloaded files) reaches a given threshold n. We tested the system was values of n (in thousands) = 10, 20, 30, 50, 75, 100, 150, 200, and 300. We also tested the system without the use of the in-memory data structure, i.e., directly storing the downloaded files to the hard-drive. Fig. 3, Fig. 4, and Fig. 5 show sample results of one of the performance parameters (Average Disc Writes/Sec) for three cases; no in-memory data structure, with linked list of sizes 10,000 and 100,000. The last two figures show that the sick drive is being used in time separate intervals depending on the size of the linked list. The percentage write times were 26% for the “no in-memory” data structure case while it was as low as 19% for the in-memory data structure usage with linked list of length 100,000 nodes.

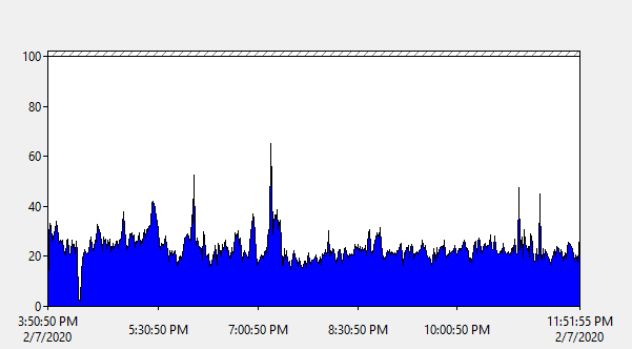

Fig. 3 Disc Writes/sec with no usage of in-memory data structure

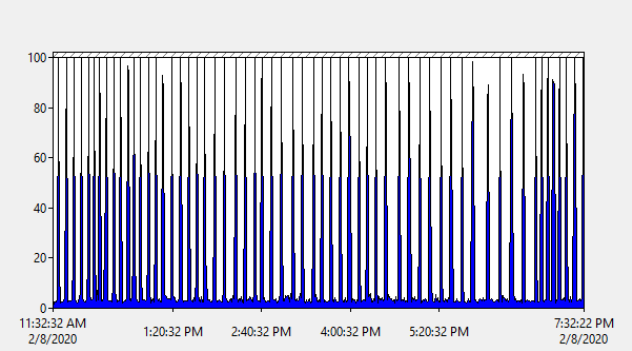

Fig. 4 Disc Writes/Sec with in-memory data structure with linked-list length of 10,000 nodes

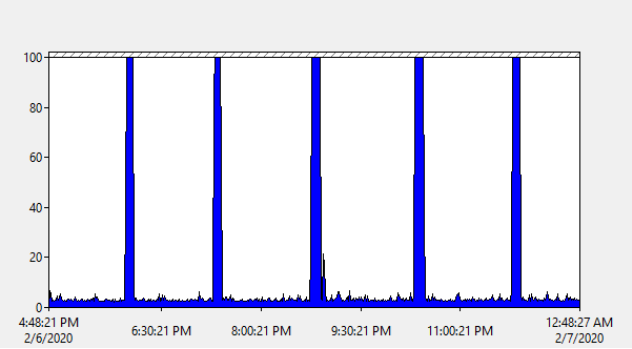

Fig. 5 Disc Writes/Sec with in-memory data structure with linked-list length of 100,000 Nodes

The system throughput represented by the total downloaded data in Mega Bytes per minute (MB/min) is shown in Fig. 6. The graph shows a somewhat oscillating characteristic of the throughput with a maximum value obtained around a linked list of size 100,000 nodes. This can be due to the alignment of hard disk sectors and the total data size to be stored. A match between the two values servers to produce a local optimal system throughput.

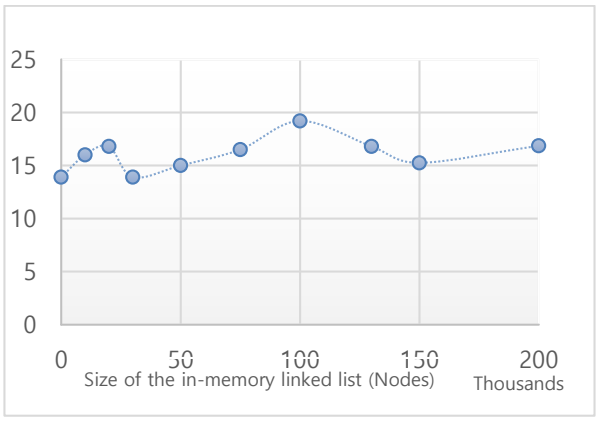

Fig. 6 A graph showing the number of nodes (n) in the linked list (x-axis) and the throughput in MB/min (y-axis)

Another performance measure has been considered which is the average number of files downloaded for various values of linked-list size. A previous research [10], has reported an average of 7.75 pages per second (pages/sec). Table 1 below shows the achieved pages/sec for some selected values of ***n*** (10, 20, 75, 100 and 150 thousands).

Table 1: The Average collected pages/sec against the linked-list size (n)

Linked-list size(n)	***Total pages collected***	Average pages/sec
-	544,320	18.9
10,000	601,920	20.9
20,000	619,200	21.5
75,000	564,705	19.6
100,000	633,600	22
150,000	555,840	19.3

It can be shown from the table that there is an optimal value of 22 pages/sec at n = 100,000 nodes. This value is consistent with the result shown in Fig. 6 as the maximum number of downloaded pages/sec is expected to relate to the overall system throughput that occurred in a similar value of n.

5. Conclusions

In this work, the author presented a distributed and multi-threaded crawler design using the in-memory data structure. This system is motivated by the fact that frequent disc writes of downloaded individual files causes performance degradation and a lower throughput due to the disk seek times. Patch writing of data to disc after being stored in in-memory data structures is a viable alternative especially due to the availability of larger memory with affordable prices. The results show that the proposed technique shows about 50% performance improvement for some selected values of the size of the in-memory data structure over the multi-threaded technique with no in-memory data structure. The system also achieved a crawling rate of 22 pages/sec which is superior to previously reported values.

6. Future Work

The authors are already extending the current system to employ a graphic processing unit (GPU) with over 1500 cores to increase the throughput as well as experimenting with significantly larger in-memory data structure to reach one million nodes.

Acknowledgments

This research is supported by the First Tamayoz initiative project number 23/40 by the research deanship of the Islamic University of Madinah, Saudi Arabia. The author would like to thank Mr. Ali Domlo for his help during the development of this work.

References

[1] D. Reinsel, J. Gantz, J. Rydning, "he Digitization of the World From Edge to Core", IDC white paper, Available From: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf [Accessed: February 2020].

[2] S. Büttcher, C. L. A. Clarke, G. V. Cormack, Information Retrieval: Implementing and Evaluating Search Engines, MIT Press – July 23, 2010

[3] M. A.R. Abdeen et. al, "A Closer Look at Arabic Text Classification", The International Journal of Advances in Computer Science Applications (IJACSA), Nov. 2019.

[4] D. Eichmann, The RBSE spider: Balancing effective search against web load. In Proceedings of the 1st World Wide. Web Conference, 1994

[5] O. A. McBryan, GENVL and WWWW: Tools for taming the web. In Proceedings of the 1st World Wide Web Conference. 1994, pp. 79–90.

[6] B. Pinkerton, Finding what people want: Experiences with the webcrawler. In Proceedings of the 2nd International World Wide Web (Online & CDROM review: the international journal of), Anonymous (Ed.), Vol. 18(6). Learned Information, Medford, NJ, 1994.

[7] S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30, 1998, pp. 107–117.

[8] S. M. Mirtaheri et al. "A brief history of web crawlers." ArXiv abs/1405.0749, 2013.

[9] G. Sun, H. Xiang, S. Li, On Multi-Thread Crawler Optimization for Scalable Text Searching. Journal on Big Data, Vol. 1, 2019, p. 89–106.

[10] The ClueWeb09 Dataset. 2009. Homepage. Retrieved May 16, 2018 Available From http://lemurproject.org/clueweb09/ [Accessed Feb. 2020].

[11] L. Meegahapola, et. al. "Adaptive technique for web page change detection using multi-threaded crawlers." In The IEEE Seventh International Conference on Innovative Computing Technology (INTECH), 2017, pp. 120-125

[12] P. Boldi, et. al. "BUbiNG: Massive Crawling for the Masses". ACM Trans. Web 12, 2, Article 12, June 2018.

[13] R. S. Maso, "In-Memory, Distributed Data Structures for the masses", July, 2018, Available From: https://medium.com/sharenowtech/in-memory-distributed-data-structures-for-the-masses-c6627664474a [Accessed Feb. 2020].

[14] A. Chatzistergiou, M. Cintra, S. Viglas, "REWIND: Recovery Write-Ahead System for In-Memory Non-Volatile Data-Structures", PVLDB, 8, 2018, pp. 497-508

Mohammad A. R. Abdeen received the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Victoria, and the University of Ottawa, Canada respectively. He assumed numerous academic and industrial positions in international academic institutions and industrial organizations. He is currently an Adjunct Professor at the University of Ottawa, Canada and an Associate Professor at the Islamic University of Madinah, Saudi Arabia.