Faster and Resourceful Multi-core Web Crawling

Arun Kumar Dewangan[1], Prof. Asha Ambhaikar[2]

[1]Rungta College of Engineering & Technology
Bhilai, Chhattisgarh, India
arun.dewangan@gmail.com

[2] Rungta College of Engineering & Technology
Bhilai, Chhattisgarh, India
asha31.a@rediffmail.com

Abstract: ***Due to massive growth of World Wide Web, search engines have become crucial tools for navigation of web pages. In order to provide fast and powerful search facility, search engine maintains indexes for documents and its contents on the web by downloading web pages for processing in iterative manner. Web indexes are created and managed by web crawlers which work as a module of search engines and traverse the web in systematic manner for indexing its contents. A web crawling is a process which fetches data from various servers. It is a time taking process as it gathers data from various servers. Hence, to speed up the searching process in search engines the crawling should be fast enough. The aim of the proposed system is to enhance the speed of crawling process and CPU usage by use of multi-core concept.***

Keywords: Web crawling, multi core, indexing, parallel crawler, CPU, URL.

1. Introduction

The user of World Wide Web is being expanded by an unpredictable speed. To process a query, search engine faces many problems like providing an accurate and updated result to the user. Hence the search engine should respond in appropriate timely manner. A web crawler is a module of search engine that traverse the web and fetches URLs from seed URLs (initial URLs). Fetched URLs are kept in priority based queue and process them in iterative manner. Search engine consists of following modules [1]:

- A user interface and query engine which interacts with database and provide the result as per user query.
- A Crawling module is a system which fetches web pages from web servers, known as web crawler.
- Indexing and parsing module which process the web page and extract data from the page and organize the information.

A multi-core processor is a single computing component having more than one independent actual central processing units (called “cores”), which are the unit that read and execute program instructions. The instructions are normal CPU instructions, but multiple cores can run multiple instructions at the same time and increasing overall speed in parallel computing. With the popularity of the multi-core and many-core architectures there is a great requirement for software frameworks which can support parallel programming methodologies. With the advent of multi-core processors, the importance of parallel computing is significant in the modern computing era since the performance gain of software will mainly depend on the maximum utilization across the cores existing in a system [2].

2. Web Crawler

Web crawler is a module of search engine that fetches web page from web server and extract the information. The main function of a web crawler is to recursively visit web pages, extract all URLs form the page, parse the page for keywords and visit the extracted URLs recursively [3]. Web indexes are created and maintained by Web crawlers which operate on behalf of Web search engines and systematically traverse the Web for the purpose of indexing its contents. Consequently, Web crawlers are information discovery tools which implement certain Web crawling algorithms [6]. A simple crawling algorithm is as follows [1]:

1) Do Forever
2) Begin
3) Read a URL from the set of seed URL’s
4) Determine the IP-address for the Host name
5) Download the Robot.txt file, which carries download information and also includes the files to be excluded by the crawler
6) Determine the protocol of underlying Host like HTTP, FTP, GOPHER
7) Based on this protocol, download the document
8) Check whether the document has already been downloaded or not
9) If the document is a fresh one,
10) Then
11) store it and extract the links and references to other sides from that document
12) Else
13) Abandon the document
14) End

3. Multi-core System

Single-core processors are able to interleave instruction streams, but not execute them simultaneously; the overall performance gains of a multi-threaded application on single-core architectures are limited. On these platforms, threads are generally seen as a useful programming abstraction for hiding latency. This performance restriction is removed on multi-core architectures. On multi-core platforms, threads do not have to wait for any one resource. Instead, threads run independently on separate cores. As an example, consider

two threads that both wanted to execute a shift operation. If a core only had one "shifter unit" they could not run in parallel. On two cores, there would be two "shifter units," and each thread could run without contending for the same resource. Multi-core platforms allow developers to optimize applications by intelligently partitioning different workloads on different processor cores. Application code can be optimized to use multiple processor resources, resulting in faster application performance [11].

4. Literature Review

To improve the crawling speed a number of modifications have been done. Some of them are under consideration.

Distributed Web Crawler [4], in this approach to design a robust and efficient web crawler, it is needed to make the task distribution across multiple machines in concurrent processing. Huge web pages should be independently distributed on the network and they should provide adequate possibility and rationality for concurrent accesses. Meanwhile, concurrent distribution will save network bandwidth resources. Besides, in order to improve the recall ratio, precision and search speed of the whole system, the internal algorithm of the search should boast certain intellectualization. In order to improve the speed, hundreds of distributed crawlers can usually be launched simultaneously. Distributed crawlers simultaneously analyze and dispose of the collected web pages, extract URL links and other relevant information, submit to their respective dispatchers, and their respective dispatchers submit them to the chief dispatcher.

PARALLEL CRAWLERS [5], in this paper, a parallel crawler has multiple crawling processes (C-proc's). Each C-proc's performs the basic task that a single process crawler conducts. It downloads pages from the Web, stores the pages locally, extracts URLs from the downloaded pages and follows links. Depending on how the C-proc's split the download task, some of the extracted links may be sent to other C-proc's. The C-proc's performing these tasks may be distributed either on the same local network or at geographically distant locations. The authors described two crawling modes firewall mode and exchange mode. In firewall mode, the overall crawler does not have any overlap in the downloaded pages, because a page can be downloaded by only one C-proc, if ever. However, C-proc's can run quite independently in this mode, because they do not conduct any runtime coordination or URL exchanges. When C-proc's periodically and incrementally exchange inter-partition URLs, they operate in an exchange mode. Processes do not follow inter-partition links. The firewall mode give Cproc's much independence (C-proc's do not need to communicate with each other), but they may download the same page multiple times, or may not download some pages. In contrast, the exchange mode avoids these problems but requires constant URL exchange between C-proc's.

MOBILE CRAWLER [6], an alternative approach to Web crawling is based on mobile crawlers. In this, the author demonstrates more efficient approach to the "download first process later" strategy of search engine by using mobile crawlers. This has advantage that analysis portion of crawler is done locally rather than remotely. This reduces network load and speeds up the indexing phase inside the search engine.

MERCATOR [7] is a scalable and extensible crawler. Mercator's design features a crawler core for handling the main crawling tasks, and extensibility through protocol and processing modules. Users may supply new modules for performing customized crawling tasks. We have used Mercator for a variety of purposes, including performing random walks on the web, crawling our corporate intranet, and collecting statistics about the web at large.

PARCAHYD [8], in this, work it has been proposed that if the links contained within a document become available to the crawler before an instance of crawler starts downloading the documents itself, then downloading of its linked documents can be carried out in parallel by other instances of the crawler. Therefore, it is proposed that meta-information in the form Table Of Links (TOL) consisting of the links contained in a document be provided and stored external to the document in the form of a file with the same name as document but with different extension. This one time extraction of TOL can be done at the time of creation of the document.

Distributed Vertical Crawler [10], in this the authors present a distributed template-customized vertical crawler which is specially used for crawling Internet forums. The Client-Server architecture of the system and the function of every module are described in detail which can be extended to other fields easily. A crawling-period based distribution strategy is also proposed, with which the crawler manager can coordinate the quantity of crawling tasks and the resources of each crawler very well, and the crawler can process websites with different updating frequency flexibly. Authors also define a communication protocol between crawlers and crawler manager and describe how to solve the duplicated crawling problem in the distributed system. The performance of centralized vertical crawler and distributed vertical crawler are compared in the experiment. Experimental results demonstrate that the parallel operation of all the crawlers in the distributed system can greatly enhance the crawling efficiency.

AuToCrawler [12] consist of a user interest specification module that mediates between users and search engines to identify target examples and keywords that together specify the topic of their interest, and a URL ordering strategy that combines features of several previous approaches and achieves significant improvement. It also provides a graphic user interface such that users can evaluate and visualize the crawling results that can be used as feedback to reconfigure the crawler.

Focused Crawlers [13] aim to search only the subset of the web related to a specific topic, and offer a potential solution to the problem. The major problem is how to retrieve the maximal set of relevant and quality pages. The authors propose an architecture that concentrates more over page selection policy and page revisit policy. The three-step algorithm for page refreshment serves the purpose. The first layer contributes to decision of page relevance using two methods. The second layer checks for whether the structure of a web page has been changed or not, the text content has

been altered or whether an image is changed. The third layer helps to update the URL repository.

Parallel Migrating Web Crawler [1], in order to keep the database of a search engine up to date, crawlers must constantly retrieve/update Web pages as fast as possible. To do this, decentralize and perform site-based distribution of work among the machines and simultaneously crawl as many domains as possible. The crawling function is logically migrated to different machines which send back the filtered and compressed data to the central machine which saves time and bandwidth. The crawler system itself consists of several specialized components, in particular a central crawler, one or more crawl frontiers, and a local database of each crawl frontier. This data is transferred to the central crawler after compression and filtering which reduces the network bandwidth overhead. These crawl frontiers, are logically migrated on different machines to increase the system performance. The central crawler is responsible for receiving the URL input stream from the applications and forwarding it to the available crawl frontiers. The crawler system is composed of a central crawler and a number of crawl frontiers which perform the task of crawling.

5. Problem Identification

With growing size of web, it is difficult to retrieve the whole or important portion of web by single process. Hence, many search engines run multiple processes in parallel to perform the task and to maximize the download rate. In serial execution of processes, the CPU is simply idle while process is waiting for I/O process to complete. In multi-threaded environment the processes are executed concurrently. The concurrent processes are handled by the operating system using context switching. In both the cases no two processes are executed in parallel (at the same time) in single CPU. Since, multi-core systems have been developed, to make CPU efficient and speed up the process, we should make use of all the cores available in the single CPU. Web crawling is a time taking process. To speed up the crawling process we should utilize the multi-core systems by executing the crawling process in parallel.

6. Proposed Solution

The proposed solution to the above problem is to distribute the crawling process among the different cores available in the CPU. In order to approach towards the implementation of multi-core utilization of CPU and speed up the process, some sample html files are processed. We processed 6 sample html files (named temp1.html, temp2.html, ---, temp6.html) to count the number of characters in each html files. We processed the html files in parallel in multi-core system by distributing the processing of files into different cores. The output is taken with respect to their start time, number of characters read, time taken for its execution.

7. Observation

The following observations are taken for serial execution, multi threading execution and multi-core execution:

Serial Execution:
Started on: 10:51:25
temp1.html contains 52594 char
Execution time 9.5 secs

Started on: 10:51:35
temp2.html contains 90760 char
Execution time 28.828 secs

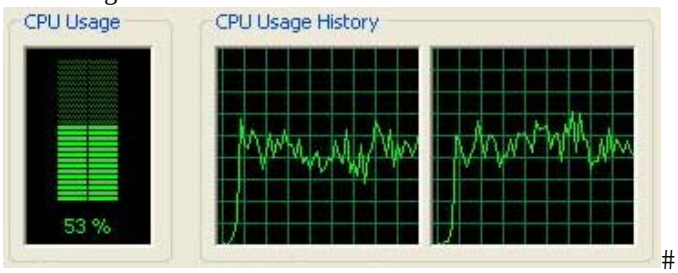
Started on: 10:52:3
temp3.html contains 25060 char
Execution time 2.344 secs

Started on: 10:52:6
temp4.html contains 50534 char
Execution time 9.296 secs

Started on: 10:52:15
temp5.html contains 61543 char
Execution time 13.75 secs

Started on: 10:52:29
temp6.html contains 61745 char
Execution time 13.469 secs
Total run time 77.21900000000001 secs

CPU Usage is 49% - 55%



#

Figure 1. Serial Execution#

Multi-threading Execution:
Started on: 10:59:55
temp1.html contains 52594 char
Execution time 9.594 secs

Started on : 11:0:5
temp2.html contains 90760 char
Execution time 27.765 secs

Started on: 11:0:33
temp3.html contains 25060 char
Execution time 2.094 secs

Started on: 11:0:35
temp4.html contains 50534 char
Execution time 9.266 secs

Started on: 11:0:44
temp5.html contains 61543 char
Execution time 13.391 secs

Started on: 11:0:58
temp6.html contains 61745 char

Execution time 13.141 secs
Total run time 75.297 secs

CPU Usage is 50% - 60%

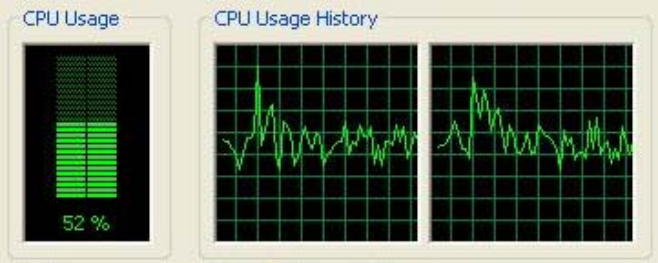


Figure 2. Multi-threading Execution#

Multi-core Execution:
Started on: 11:4:55

Started on : 11:4:55
temp1.html contains 52594 char
Execution time 14.079 secs

Started on : 11:5:9
temp3.html contains 25060 char
Execution time 3.2960000000000003 secs

Started on : 11:5:12
temp4.html contains 50534 char
Execution time 13.579 secs

Started on : 11:5:26
temp2.html contains 90760 char
Execution time 42.438 secs

Started on : 11:5:37
temp5.html contains 61543 char
Execution time 19.937 secs
temp6.html contains 61745 char
Execution time 15.844000000000001 secs
Total run time 58.391 secs

CPU Usage is 85% - 93%

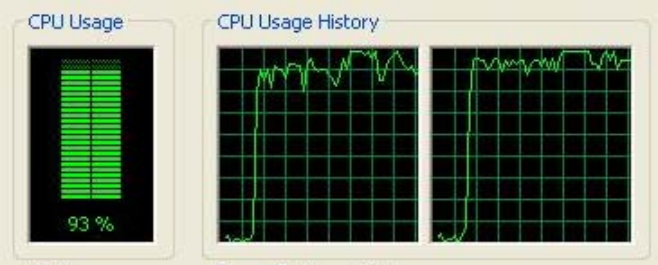


Figure 3. Multi-core Execution

8. Conclusion

In this paper, we proposed a solution to maximize the CPU usage and speed up the execution power of multi-core CPU. As per the graph and observation recorded, it is concluded that in multi-core execution, CPU utilization and speed of execution is more as compared to serial and multi-threaded execution. The same concept (multi-core execution) will be implemented in "Faster and Resourceful Multi-core Web Crawling".

References

[1] Akansha Singh, Krishna Kant Singh, "Faster and Efficient Web Crawling with Parallel Migrating Web Crawler," International Journal of Computer Science Issues, Vol. 7, Issue 3, No 11, pages 28-32, May 2010.

[2] G.A.C.P. Ganegoda, D.M.A. Samaranayake, L.S. Bandara, K.A.D.N.K. Wimalawarne , "JConqurr - A Multi-Core Programming Toolkit for Java", International Journal of Computer and Information Engineering 3:4, pages 223-230, 2009.

[3] Anup A Garje, Prof. Bhavesh Patel, Dr. B. B. Meshram, "Realizing Peer-to-Peer and Distributed Web Crawler", International Journal of Advanced Research in Computer Engineering & Technology, Volume 1, Issue 4, pages 353-357, June 2012.

[4] Shaojun Zhong, Zhijuan Deng, "A Web Crawler System Design Based on Distributed Technology", JOURNAL OF NETWORKS, VOL. 6, NO. 12, pages 1682-1689, DECEMBER 2011.

[5] Junghoo Cho, Hector Garcia Molina, "Parallel Crawlers". In Proceedings of the Eleventh International World Wide Web Conference, 2002.

[6] Joachim Hammer, Jan Fiedler, "Using Mobile Crawlers to Search the Web Efficiently", International Journal of Computer and Information Science, 1:1, pages 36-58, 2000.

[7] A. Heydon, M. Najork, "Mercator: A scalable, extensible web crawler", World Wide Web, vol. 2, no. 4, 1999.

[8] A. K. Sharma, J. P. Gupta, D. P. Agarwal, "PARCAHYD An Architecture of a Parallel Crawler based on Augmented Hypertext Documents", International Journal of Advancements in Technology, Vol 1, No 2, pages 270-283, October 2010.

[9] P. Boldi, B. Codenotti, M. Santini, S. Vigna, "Ubicrawler: A scalablefully distributed web crawler", In Proceedings of AusWeb02 - The Eighth Australian World Wide Web Conference, Queensland, Australia, pages 1-14, 2002.

[10] Bing Zhou, Bo Xiao, Zhiqing Lin, Chuang Zhang, "A Distributed Vertical Crawler Using Crawling-Period Based Strategy", 2nd International Conference on Future Computer and Communication, Volume 1, pages 306-311, 2010

[11] Shameem Akhter, Jason Roberts, "Multi-Core Programming", Intel Corporation, April 2006.

[12] Jyh-Jong Tsay, Chen-Yang Shih, Bo-Liang Wu , "AuToCrawler: An Integrated System for Automatic Topical Crawler", Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, pages 1-6, 2005.

[13] Swati Mali, B. B. Meshram, "Focused Web Crawler with Page Change Detection Policy", 2nd International Conference and workshop on Emerging Trends in Technology, pages 51-57, 2011.