Validation of Architecture of Migrating Parallel Web Crawler using Finite State Machine

Md. Faizan Farooqui[1], Dr. Md. Rizwan Beg[2] and Dr. Md. Qasim Rafiq[3]

[1]Research Scholar, Department of Computer Application, Integral University, Lucknow, Uttar Pradesh 226026, India

[2]Professor, Department of Computer Science and Engineering, Integral University, Lucknow, Uttar Pradesh 226026, India

[3]Professor, Department of Computer Engineering, Aligarh Muslim University, Aligarh, Uttar Pradesh, India

ABSTRACT

The process of downloading web pages is known as web crawling. In this paper we validate the architecture of Migrating parallel web crawler using finite state machine. The method for Migrating Parallel Web Crawling approach will detect changes in the content and structure. Also Domain specific crawling will yield high quality pages. The crawling process will migrate to host or server with specific domain and start downloading pages within specific domain. Incremental crawling will keep the pages in local database fresh thus increasing the quality of downloaded pages. The crawling strategy makes web crawling system more effective and efficient. Test cases are generated for the validation of the architecture. The approach for generating the test cases through FSM is very reliable and efficient and does not support for the invalid test cases. Valid input strings are generated as test cases.

Keywords: *Web crawling, parallel migrating web crawler, search engine, validation*

1. Introduction

The Internet is a system of interconnected computer networks. World-Wide Web has created challenges for society as well as for the technology used for the Web. Use of the Web has raised important questions in the fields of censorship, privacy, and access to information. On the technological front there is needed to scale the applications to store the resulting large databases and heavy network loads. The solution to above problems lie in the disciplines of distributed systems and information retrieval. In the olden days when a user wants to find information on the Web, he either had to know the exact location of the documents or he had to navigate patiently from one hyperlink to another hyperlink in hopes of finding his desired page. As the size of the Web grew such type of navigation became impossible. Web Crawlers has made the Web easier to use for millions of people. Web Crawlers can be considered as a Web service to assist users in Web navigation. Web Crawler can also be considered as a node in the Web graph that contains links to many sites on the Web. The Migrating Parallel Crawler system consists of Central Crawler, Crawl Frontiers, and Local Database of each Crawl Frontier and Centralized Database.

It is responsibility of central crawler to receiving the URL input from the applications and forwards the URLs to the available migrating crawling process. Crawling process migrated to different machines to in-crease the system performance. Local database of each crawl frontier are buffers that locally collect the data. This data is transferred to the central crawler after compression and filtering which reduces the network bandwidth overhead. The central crawler has a centralized data-base which will contain the documents collected by the crawl frontiers independently.

2. Literature survey

In [13], the author demonstrated an efficient approach to the "download-first process-later" strategy of existing search engines by using mobile crawlers. In [14] author has implemented UbiCrawler, a scalable distributed and fault-tolerant web crawler. In [15] author presented the architecture of PARCAHYD which is an ongoing project aimed at designing of a Parallel Crawler based on Augmented Hypertext Documents. In [16] the author studied how an effective parallel crawler is designed.

As the size of the Web grows, it becomes imperative to parallelize a crawling process. In [17] the author proposes Mercator, which is a scalable, extensible crawler. [18] Presented Google, a prototype of a large scale search engine which makes heavy use of the structure present in hypertext. [19] Aims at designing and implementing a parallel migrating crawler in which the work of a crawler is divided amongst a number of independent. In [20] the author has reviewed the various crawling techniques. [21] is an extended model for effective migrating parallel web crawlers with domain specific and incremental crawling.

3. Issues in Migrating parallel Web Crawlers

According to [1], Web crawlers of big commercial search engines crawl up to 10 million pages per day. Assuming an average page size of 6K [2], the crawling activities of a single commercial search engine adds a daily load of 60GB to the Web. One of the first Web search engines, the World Wide Web Worm [3], was introduced in 1994 and used an index of 110,000 Web pages. The Web is expected to grow further at an exponential speed, doubling its size (in terms of number of pages) in less than a year [4]. Current Web crawlers download all these irrelevant pages because traditional crawling techniques cannot analyze the page content prior to page download [13]. The following issues are important in the study of a migrating parallel crawler interesting are Quality, Communication bandwidth and Overlap.

4. Basics of Finite-State Machine

A finite state automation is represented as (Q, Σ, δ, q_0, F) where Q is finite non empty set of states, Σ is finite non empty set of inputs, δ is function which maps Q x Σ into Q and known as direct transition function, q_0 ($q_0 \in Q$) is initial state and F ($F \subseteq Q$) is the set of one or more final states.

A grammar is required for the formation of sentences in any language which is defined as (V_N, Σ, P, S) where V_N is a finite nonempty set whose elements are called variables, Σ is a finite nonempty set whose elements are called terminals and $V_N \cap \Sigma$, S is a special variable called the start symbol and belongs to V_N . In this grammar P is a finite set whose elements are α->β, where α and β are strings on $V_N \cup \Sigma$. Elements of P are called production rules.

5. Validation of Migrating parallel Web Crawler using Finite state Machine

Web content is dynamic in nature [22] and 14% of the links in search engines are broken [25]. At regular interval of time it is very necessary to refresh the data base of web pages. The rate of change varies from site to site. Therefore, managing the local collection afresh becomes a challenging task. The changes that can occur in web pages can be classified into following 4 major categories [23] are Structural Changes, Content level changes, Cosmetic changes and Behavioral changes. Some mechanism [24,26], used for change detection, does use the policy of retaining cached copies of web pages which are later compared with downloaded web document to determine if there has been any change. Domain specific crawling will yield high quality pages. The crawling process will migrate to host or server with specific domain and start downloading pages within specific domain. Incremental crawling will keep the pages in local database fresh thus increasing the quality of downloaded pages [21]. In this paper the architecture of parallel migrating crawler is validated with the help of finite state machine. Then the test cases are generated to validate the architecture.

Given below is the architecture of Migrating Parallel Web Crawler in figure 1. Figure 2 is the finite state machine for the architecture of Migrating parallel Web crawler. Table 1 shows the transition table generated from the finite state machine.

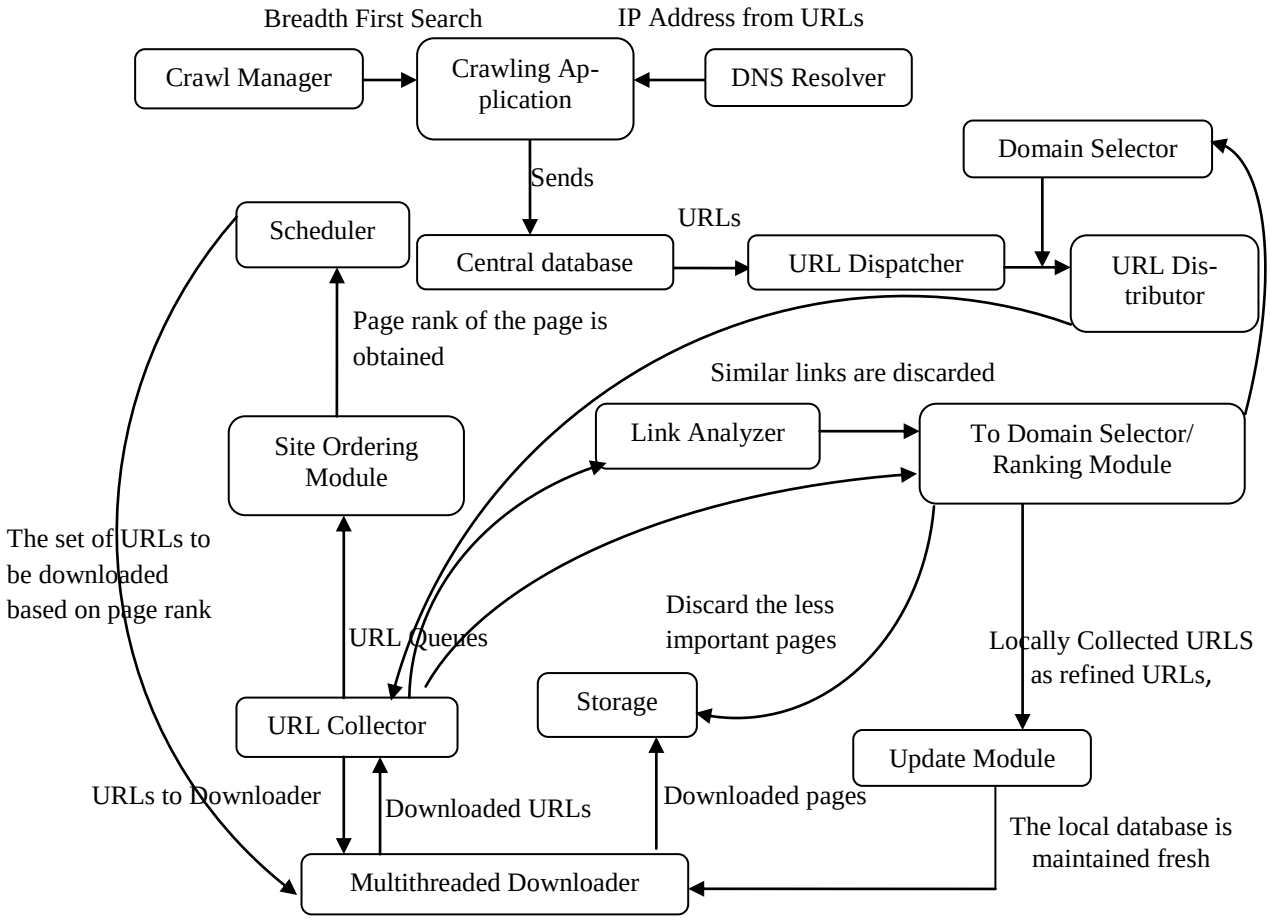

Fig. 1 Architecture of Migrating Parallel Web Crawler

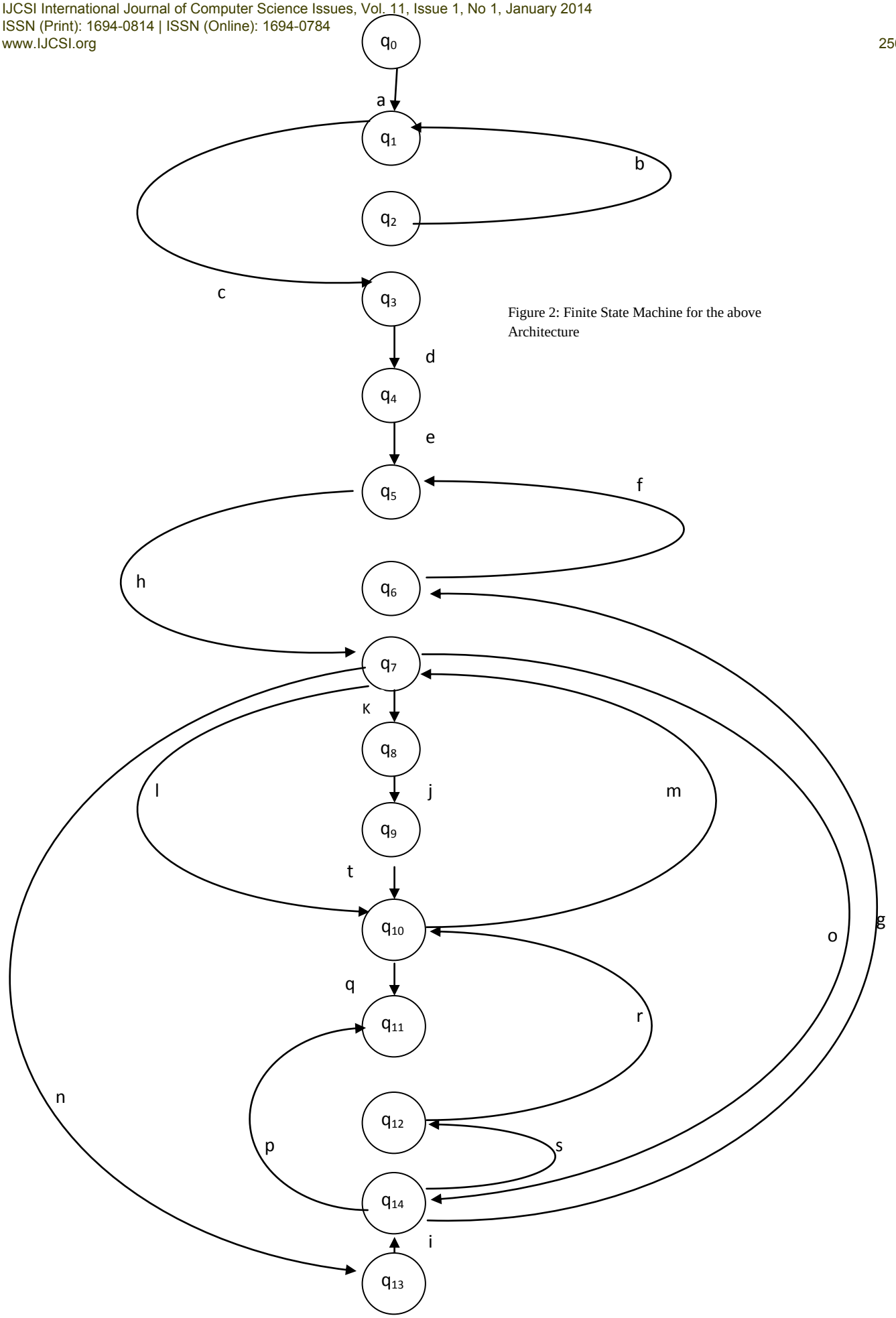

Figure 2: Finite State Machine for the above Architecture

Table 1: Transition Table

States/ Inputs	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'	'k'	'l'	'm'	'n'	'o'	'p'	'q'	'r'	's'	't'
q_0	q_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_1	-	-	q_3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_2	-	q_1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_3	-	-	-	q_4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_4	-	-	-	-	q_5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_5	-	-	-	-	-	-	-	q_7	-	-	-	-	-	-	-	-	-	-	-	-
q_6	-	-	-	-	-	q_5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_7	-	-	-	-	-	-	-	-	-	-	q_8	q_{10}	q_{13}	-	q_{14}	-	-	-	-	-
q_8	-	-	-	-	-	-	-	-	-	q_9	-	-	-	-	-	-	-	-	-	-
q_9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	q_{10}
q_{10}	-	-	-	-	-	-	-	-	-	-	-	-	q_7	-	-	-	q_{11}	-	-	-
q_{11}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
q_{12}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	q_{10}	-	-
q_{13}	-	-	-	-	-	-	-	-	q_{14}	-	-	-	-	-	-	-	-	-	-	-
q_{14}	-	-	-	-	-	-	q_6	-	-	-	-	-	-	-	-	q_{11}	-	-	q_{12}	-

The various productions and Transition functions can be induced for the above finite state machine:-

$\delta(q_0,a)=q_1 \Rightarrow q_0 \rightarrow aq_1$
$\delta(q_2,b)=q_1 \Rightarrow q_2 \rightarrow bq_1$
$\delta(q_1,c)=q_3 \Rightarrow q_1 \rightarrow cq_3$
$\delta(q_3,d)=q_4 \Rightarrow q_3 \rightarrow dq_4$
$\delta(q_4,e)=q_5 \Rightarrow q_4 \rightarrow eq_5$
$\delta(q_5,h)=q_7 \Rightarrow q_5 \rightarrow hq_7$
$\delta(q_6,f)=q_5 \Rightarrow q_6 \rightarrow fq_5$
$\delta(q_7,k)=q_8 \Rightarrow q_7 \rightarrow kq_8$
$\delta(q_8,j)=q_9 \Rightarrow q_8 \rightarrow jq_9$
$\delta(q_9,t)=q_{10} \Rightarrow q_9 \rightarrow tq_{10}$
$\delta(q_{10},q)=q_{11} \Rightarrow q_{10} \rightarrow qq_{11}$
$\delta(q_7,l)=q_{10} \Rightarrow q_7 \rightarrow lq_{10}$
$\delta(q_7,n)=q_{13} \Rightarrow q_7 \rightarrow nq_{13}$
$\delta(q_7,o)=q_{14} \Rightarrow q_7 \rightarrow oq_{14}$
$\delta(q_{10},m)=q_7 \Rightarrow q_{10} \rightarrow mq_7$
$\delta(q_{14},p)=q_{11} \Rightarrow q_{14} \rightarrow pq_{11}$
$\delta(q_{12},r)=q_{10} \Rightarrow q_{12} \rightarrow rq_{10}$
$\delta(q_{14},g)=q_6 \Rightarrow q_{14} \rightarrow gq_6$
$\delta(q_{13},i)=q_{14} \Rightarrow q_{13} \rightarrow iq_{14}$
$\delta(q_{14},s)=q_{12} \Rightarrow q_{14} \rightarrow sq_{12}$

From the above grammar, test cases are generated with the help of grammar and it can be verified from the production rules.

Valid Test Case 1: Similar links are discarded

URL collector transfers the sets of URLs to link analyzer. Link Analyzer performs the link analysis where similar links are discarded. Link analyzer then transfer the unique sets of links to domain selector. For this equivalent grammar is given below:

$q_7 \rightarrow nq_{13}$
$q_{13} \rightarrow iq_{14}$

By replacing the non terminals on RHS of productions, the following conclusions are drawn which shows that similar links are discarded.

$q_7 \rightarrow niq_{14}$ ------- (1)

Derived production (1) shows that domain selector will proceeds with unique set of URLs similar links will be discarded.

Valid Test Case 2: Sets of URLs that are downloaded are based on page rank.

URL collector transfers the URLs to site ordering module. The page rank is calculated at the site ordering module. The pages with high page rank are transferred to the scheduler. Scheduler then transfer the set of URLs to the multi threaded downloader. For this equivalent grammar is given below:

$q_7 \rightarrow kq_8$
$q_8 \rightarrow jq_9$
$q_9 \rightarrow tq_{10}$

By replacing the non terminals on RHS of productions, the following conclusions are drawn which shows that set of URLs that are downloaded are based on page rank.

$q_7 \rightarrow kjtq_{10}$ ------- (2)

Derived production (2) shows that set of URLs that are downloaded by multithread downloader are based on page rank.

Valid Test Case 3: The pages that are less important are discarded.

URL collector transfers the sets of URLs to link analyzer. Link Analyzer performs the link analysis where similar links are discarded. Link analyzer then transfer the unique sets of links to Ranking module. The ranking module discards the less important links. For this equivalent grammar is given below:

$q_7 \rightarrow nq_{13}$
$q_{13} \rightarrow iq_{14}$
$q_{14} \rightarrow pq_{11}$

By replacing the non terminals on RHS of productions, the following conclusions are drawn which shows that similar links are discarded.

$q_7 \rightarrow nipq_{11}$ ------- (3)

Derived production (3) shows that less important links are discarded.

Valid Test Case 4: Maintaining Local Database Fresh

URL collector transfers the sets of URLs to Ranking Module. Ranking module refines the locally collected URLs. Refined URLs are then transferred to update module. Update module checks for updated URLs. Only those URLs that represents then updated pages are sent to the multi threaded downloader for downloading. For this equivalent grammar is given below:

$q_7 \rightarrow oq_{14}$
$q_{14} \rightarrow sq_{12}$
$q_{12} \rightarrow rq_{10}$

By replacing the non terminals on RHS of productions, the following conclusions are drawn

which shows that local database is maintained fresh.

q_7->$osrq_{10}$ -------(4)

Derived production (4) shows that the multi threaded downloader downloads the pages that are fresh and the database is maintained fresh with the help of update module.

6. Conclusions

In this paper we validated the architecture of Migrating parallel Web crawler with the help of finite state machines. The approach for generating the test cases through FSM is very reliable and efficient and does not support for the invalid test cases. Valid input strings are generated as test cases.

The research directions in migrating parallel crawler include:

• Security could be introduced in migrating parallel crawlers

• Migrating parallel crawler could be made polite

• Location awareness could be introduced in migrating parallel crawlers

This future work will deal with the problem of quick searching and downloading the data. The data will be collected and analyzed with the help of tables and graphs.

Acknowledgments

First and foremost, our sincere thanks goes to Prof. Syed Wasim Akhtar, Honorable Vice Chancellor, Integral University, Lucknow. Prof Akhtar has given us unconditional support in many aspects, which enabled us to work on the exciting and challenging field of Engineering. We would also like to give our special thanks to Prof. T. Usmani, Pro Vice Chancellor, Integral University. His encouragement, help, and care were remarkable during the past few years. We are also grateful to Prof S. M. Iqbal Chief Academic Consultant, Integral University. Prof Iqbal provided us with valuable thoughts for our research work. My gratitude also goes to Dr. Irfan Ali Khan, Registrar, Integral University for his constructive comments and shared experience.

References

[1] D. Sullivan, "Search Engine Watch," Mecklermedia, 1998.

[2] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Stanford University, Stanford, CA, Technical Report, 1997.

[3] O. A. McBryan, "GENVL and WWW: Tools for Taming the Web," in Proceedings of the First International Conference on the World Wide Web, Geneva, Switzerland, 1994.

[4] B. Kahle, "Archiving the Internet," Scientific American, 1996.

[5] J. Gosling and H. McGilton, "The Java Language Environment," Sun Microsystems, Mountain View, CA, White Paper, April 1996.

[6] J. E. White, Mobile Agents, MIT Press, Cambridge, MA, 1996.

[7] C. G. Harrison, D. M. Chess, and A. Kershenbaum, "Mobile Agents: Are they a good idea?," IBM Research Division, T.J. Watson Research Center, White Plains, NY, Research Report, September 1996.

[8] H. S. Nwana, "Software Agents: An Overview," Knowledge Engineering Review, Cambridge University Press, 11:3, pp. , 1996.

[9] M. Wooldridge, "Intelligent Agents: Theory and Practice," Knowledge Engineering Review, Cambridge University Press, 10:2, pp. , 1995.

[10] P. Maes, "Modeling Adaptive Autonomous Agents," MIT Media Laboratory, Cambridge, MA, Research Report, May 1994.

[11] P. Maes, "Intelligent Software," Scientific American, 273:3, pp. , 1995.

[12] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language," University of Maryland Baltimore County, Baltimore, MD, September 1994.

[13] Joachim Hammer , Jan Fiedler "Using Mobile Crawlers to Search the Web Efficiently" in 2000

[14] Paolo Boldi, Bruno Codenotti, Massimo Santini, Sebastiano Vigna, "UbiCrawler: A Scalable Fully Distributed Web Crawler" in 2002

[15] A.K. Sharma, J.P. Gupta, D. P. Aggarwal, "PARCAHYDE: An Architecture of a Parallel Crawler based on Augmented HypertextDocuments" in 2010

[16] J. Cho and H.Garcia-Molina, "Parallel crawlers". In Proceedings of the Eleventh International World Wide Web Conference, 2002, pp. 124 – 135

[17] A. Heydon and M. Najork, "Mercator: A scalable, extensible web crawler". World Wide Web, vol. 2, no. 4, pp. 219 -229, 1999.

[18] Akansha Singh , Krishna Kant Singh , "Faster and Efficient Web Crawling with Parallel Migrating Web Crawler" in 2010

[19] Min Wu, Junliang Lai, "The Research and Implementation of parallel web crawler in cluster" in International Conference on Computational and Information Sciences 2010

[20] Md. Faizan Farooqui, Md. Rizwan Beg, Md. Qasim Rafiq, "A Critical Review of Migrating Parallel Web Crawler", Advances in Computing and Information Technology,Advances in Intelligent Systems and Computing Volume 177, 2013, pp 631-637

[21] Md. Faizan Farooqui, Dr. Md. Rizwan Beg and Dr. Md. Qasim Rafiq, "An Extended Model For Effective Migrating Parallel Web Crawling

With Domain Specific And Incremental Crawling", International Journal on Web Service Computing (IJWSC), Vol.3, No.3, September 2012, DOI : 10.5121/ijwsc.2012.3308 85

[22] Cho, Junghoo, Angeles, Los, and Garcia-Molina, Hector, "Effective Page Refresh Policies for Web Crawlers", ACM Transactions on Database Systems, Volume 28, Issue 4, pp. 390 – 426, December 2003.

[23] Francisco-Revilla, L., Shipman, F., Furuta, R., Karadkar, U. and Arora, A., "Managing Change on the Web", In Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, pp. 67 – 76, 2001.

[24] Manku, Gurmeet Singh, Jain, Arvind, and Sarma, Anish Das, "Detecting near-duplicates for web crawling", In Proceedings of the 16th international conference on World Wide Web, pp. 141 - 150 , May 2007.

[25] Lawrence, S., and Giles, C. L., "Accessibility of information on the web", Nature, 400:107-109, 1999.

[26] Schleimer, S., Wilkerson D. S., and Aiken, A., "Winnowing: Local algorithms for document fingerprinting", Proceedings of the ACM SIGMOD international conference on Management of data, pp. 76-85, June 2003.

First Author Mohammed Faizan Farooqui obtained his Bachelor's degree in Computer Application from the University of Lucknow in 2000 and his M.C.A. degree from Uttar Pradesh Technical University, Lucknow in 2003. Md Faizan Farooqui works since 2005 at Integral University, Lucknow as a full time Associate Professor (Jr Scale). Currently is giving courses on programming Computer Graphics and animation in MCA program at IU, Lucknow

Second Author Dr. Md. Rizwan Beg is Professor in the Department of Computer Science & Engineering in Integral University, Lucknow. He has published over 70 research paper in the International Journal of Repute. He has published many papers in International Conferences. . He is the editor and Chief Editor of many reputed International Journals. He is the member of program committee of International conferences.

Third Author Professor (Dr.) Md. Qasim Rafeeq was the chairman of Computer Engineering Department in Aligarh Muslim University.