Journal of Theoretical and Applied Information Technology
10th September 2014. Vol. 67 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REDUCING DISTRIBUTED URLS CRAWLING TIME : A COMPARISON OF GUIDS AND IDS

[1]I. SHAKIR, [2] S. ABDUL SAMAD, [3]H. BURAIRAH, [4]G. PRAMUDYA ANANTA and [5]S. SUHAILAN

[1-4]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia
[5]Faculty of Informatics and Computing, Sultan Zainal Abidin University, Terengganu, Malaysia
E-mail: [1]shakir.ibrahim@student.utem.edu.my , [2]samad@utem.edu.my , [3]burairah@utem.edu.my , [4]gedepramudya@utem.edu.my , [5]suhailan@unisza.edu.my

ABSTRACT

Web crawler visits websites for the purpose of indexing. The dynamic nature of today's web makes the crawling process harder than before as web contents are continuously updated. In addition, crawling speed is important considering tsunami of big data that need to be indexed among competitive search engines. This research project is aimed to provide survey of current problems in distributed web crawlers. It then investigate the best crawling speed between dynamic globally unique identifiers (GUIDs) and the traditional static identifiers (IDs). Experiment are done by implementing Arachnot.net web crawlers to index up to 20000 locally generated URLs using both techniques. The results shown that URLs crawling time can be reduced up to 7% by using GUIDs technique instead of using IDs.

Keywords: *Distributed systems, Web Crawler, GUID, Search Engine.*

1. INTRODUCTION

Search engine forms some sort of life way in the present internet scenario. Without search engine, many of us would defiantly not able to use the internet in a meaningful way. They help us in getting access to the information that we want to access and also to keeping track for the information that we are interesting in. There are a number of underline technologies that derive search engine. Web crawler is considered as the core of the search engines. It retrieves a massive collection of hyperlinks that contains web pages for the purpose of indexing and retrieving them to users when they ask for [1].

In 1993, the first web crawler invented by Matthew Gray (World Wide Web Wanderer). At that time the crawler used to compile statistics to determine the expansion of the web [2]. A research paper that describe web crawler (the RBSE spider) was written by David Eichmann a year after. A full description for the web crawler architecture given by Burner, his research shaped the base of the Internet Archive crawler. His paper intensively described web scaling challenges. Internet Archive was able to crawl 100 million URLs [3]. Google search engine architecture that uses a distributed system to fetch the pages and connecting with a centralized database is briefly discussed in a paper written by Brin and Page's. In [4] Mercator which happened to be a blueprint for many distributed web crawler is discussed. The UbiCrawler is described in [5], it is a scalable, fault-tolerated and fully distributed web crawler built on Java. UbiCrawler crawler many agents are distributed among web servers, in a given time only one agent is allowed to visit one web server. Each agent is uniquely identified by an ID. UbiCrawler was able to download above 10 000 000 pages per day by using 50 or more threads [6]. Many researchers have tried to make amendment in web crawler architecture in order to improve its performance. In [7], a fully distributed, platform independence and decentralized web crawler is proposed. The proposed crawler had the ability to collaborate with web servers. The results were scalable and fault-tolerance web crawler.

Day by day the internet continues to grow and became larger and larger, with the current number of pages and due to the speed of Web crawlers, crawling the entire set of web pages has come to be a great challenge, considering the dynamic nature of the Web, crawling system should have the capability of building-up its database in a shortest amount of time as possible [1].

Web crawler searches and retrieves the URLs and put them in a repository. Then the URLs

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

extracted from the repository for the purpose of the indexing and parsing, and will be sent to the database. The parsing result (URLs) will be sent to the query server. Subsequently the URLs will be assigned to the crawler agents who will visit the web pages and retrieve new URLs from them. The general architecture of search engines is illustrated in the figure 1.0.

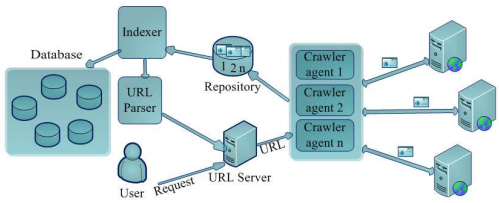

Figure 1: Architecture of Web Search Engine

As the crawling process use a repository, there might be related factors that can influence the process performance. Database speed are influenced by many aspects. One of the aspects is the unique identifier generation which is an important data to index all the information in database. As for network asset management, globally unique identifier (GUID) are used to represent a unique identifier to all assets in this world. This research is focusing on the implementation of GUID in the database for enhancing one of performance factors towards more efficient distributed web crawling process.

2. RESEARCH QUESTIONS AND METHODS

Research is conducted to answer the research questions as the following:

i. Is it database performance is one of major problems in web crawlers?

ii. Can GUID provides advantage to database performance especially in speeding up the storage process?

To answer first question, we have made literature survey from year 2004 until now using ACM/IEEE article journals or proceedings. Keywords such as "web crawler" and "search engine" are queried to seek potential research works. Then, related papers are selected based on identified problems. Problems are then grouped into certain classification. We also study on the usage of GUID on other application. "GUID" is search to query the application of GUID in order to seek its advantages or disadvantages.

In order to see the performance of the proposed solution, we ran the experimental study on locally generated URLs. URL insertion in to the database by using GUIDs is compared by the traditional identifiers or IDs insertion time. The proposed solution implemented on top of an open source DEMO_2.6 version Web crawler called Arachnode.net Web crawler. The crawler is developed in C# programming language. We used Microsoft Visual Studio 2010 Professional and we connected it by Microsoft SQL Server 2008 R2.

3. DISTRIBUTED WEB CRAWLER MAJOR PROBLEMS

Through our survey, we have categorized major problems on distributed Web crawlers into four categories. The problems are scalability, network traffic, crawling strategy, and increasing search engine size.

To solve scalability problem, researchers proposed decentralized architecture to distribute the crawling process among many crawler agents with many crawler managers. Their architectures mainly implemented in P2P architecture [8], [9]. Some researchers proposed to use multi agents web crawler as a solution for the scalability problem [10],[5].

The second problem related to the distributed Web crawler is minimizing crawler network traffic. According to [11], 40 % of internet traffic is made by web crawlers. One of the proposed solutions to overcome this problem was agent migration [12], [13]. In this solution, crawler agents migrate to web servers that hosts web pages and performs the parsing operation for the web pages there so that to reduce the communication between crawler engine and web servers. For example, crawler agent could migrate to a web server and parse a web page and extract the URLs that are contains in the web page and send them to the crawler manager. Query based approach is another useful solution for reducing crawlers' network traffics. In this approach, the updated contents are sent via dynamic web page that contains only the entirely updated pages URLs [14].

The third problem is crawling strategy. There were many algorithms proposed to increase the crawling efficiency like focused crawling, and crawling the hide web [15], [16].

The fourth problem we observed is increasing search engine size which takes a lot of attention by the researchers [1],[17].

Journal of Theoretical and Applied Information Technology
10th September 2014. Vol. 67 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Our paper also comes in this context by trying to solve one sub-problem related to increasing search engine size, which is crawling speed. We try to fill in the gap as we have noticed that not many research papers intensively studied the crawler database. The reason may be because the database does not directly participate in the crawling process. However, we believe that developing a crawler database in such a way that it makes it easy to insert the crawled data will increase the speed of the crawler as a whole, simply because the crawler agents will be free to crawl another pages after they finish their current job. In the figure 2, we illustrated the distributed web crawler knowledge chart and shows where is our contribution is.

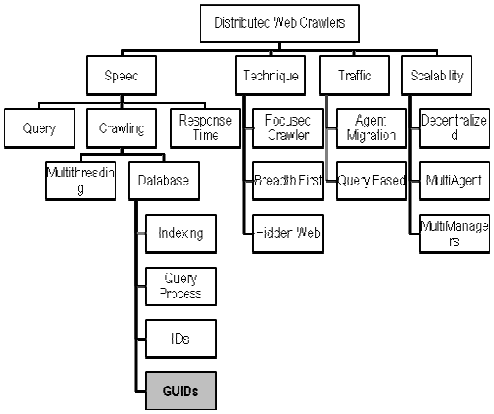

Figure 2: Architecture of Web Search Engine

4. THE USAGE OF GUIDS

In this section we define the GUID and review some applications and systems that use GUIDs. GUID or globally unique identifier is defined in the IETF standard RFC 4122 as universally unique identifier (UUID). It is 16 bytes in length and it provides wide unique values (2 powers 128). Since GUID generates a massive number of identifiers and can guarantee uniqueness, it can be useful in distributed systems. It is also very much easy to generate it on the fly without the need of checking the current value, this is because GUID not concerns about the sequence of its generated values. GUID values are generated randomly and are unique among servers [18].

GUIDs used in numerous applications. For instance, it used for the purpose of identifying network components like ad-hoc in networks. The network participants are also identified through the MAC address. However, every participant provides a number of services that must be uniquely identified as well. Participants can create GUIDs for services without collisions e.g. Bluetooth service discovery protocol uses GUIDs in order to identify services and attributes [19].

As like URIs (Uniform Resources Identifiers) GUIDs know nothing about resource location that it identifies and also it does not know whether the resource is available or not. Likewise, using GUIDs in storage that contains a huge amount of identifiers are significant [19], in [20] JXTA programming environment, GUIDs used to identify entities like an advertisement, peers, etc. GUIDs are also beneficial in object oriented based applications that identify various parts of a system. For instance, in web composition mark-up language all components are uniquely identified by GUIDs. The .NET framework is also identifies its classes and interfaces by GUIDs. With the systems that allow replication of data it is useful to implement GUIDs, because it has the ability to distinguish between data objects if it used as standard definition of data object identity. Accordingly, it allows each participant in such systems to generate its data objects with no risk of data collision with others. Participants can also make references to others data without need to know the location where the data is created and where it is warehoused [21]. Identifying data objects is a problematic task from human-computer interaction (HCI) points of view. To overcome the constraints that related to the contexts of computation like physical, device, and information context, it is significant to identify these contexts in a uniform scheme. Such identification is mainly important for mobile devices and data. GUIDs play a major role here, since it provides such uniform scheme [22]. GUIDs are also used in many distributed databases. For instance, in OceanStore system, GUIDs are used to identify resource like data objects, users' data and host machines. Using GUIDs allows OceanStore to simplify the caching and replication of the resources across several machines, as well as it allows "time travel" which refers to the ability of users to retrieve the old versions of a file or directory [23]. GUIDs used in Visage Information Architecture (VIA) model in MAYA repository to support persistent storage. The abstract data type of VIA is called U-form which is consists from a pair of attribute and value linked with a GUID [24]. VIA simplifies the integration of a new datasets with the existing one. This is because GUIDs can provide permanent identity for all objects [25]. In [26] authors proposed use of GUID in name oriented networking. GUID is used to define the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

communication services provided by the network. They dynamically mapped GUID to many topological network addresses by using a global name resolution services (GNRS). The outcome of mapping leads to hybrid GUID and Network address based routing (HGN). Using GUID is resulted in increasing the scalability and reducing the routing table size. GUIDs have been deployed in many applications. Based on the information that we have at present, using GUIDs with web crawlers have not been investigated in any research papers yet.

5. EXPERIMENTAL SETUP

The basic components of Arachnode.net web crawler architecture are crawler, parser, indexer and the database. When crawler task is start, the crawler instances or agents visit web servers and retrieve the web pages that hosted in those particular web servers, and then the web pages will set to be parsed and indexed and sent to the database. After extracting a hyperlink from a web page, the crawler has to check whether the URL has been encountered before, it does so to avoid adding duplicated URLs to the database. In the case of the URL has not been encountered before, the crawler will add it to undiscovered URL set.

5.1 Processes

In arachanode.net web crawler multi-threading mechanism is implemented to improve the performance by allowing several threads (100 threads by default) to share CPUs resources. Arachnode.net can be configured to run any number of threads and to use as much or as little processor time and memory. When Arachnode.net starts the crawling task, the crawler manager directly starts or issues a number of threads in order to start crawl process. After the requests that are waiting to be processed by the currently running thread are processed, additional requests are assigned from the crawler manager. Only one thread in a given time is allowed to assign crawl requests. However, each crawler agent is allowed to read from its priority queue. The figure 3.0 illustrated how crawl process works in Arachnode.net crawler.

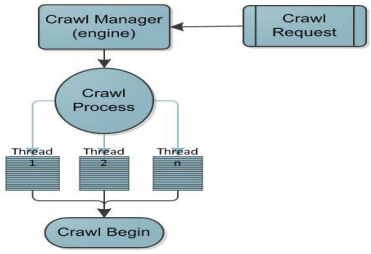

Figure 3: Arachnode.net Crawler

5.2 Naming

The naming approach used in Arachnode.net crawler is attribute-based naming. To resolve a name it executes an SQL query string to Directory database, and then the AbsoluteUri which placed on Directory database will take care of resolving process. Whenever a crawler agent (server) wants to resolve a domain name it sends extract Domain request to the base Directory database, the directory first asks AbsoluteUri to extract the IP address of the requested domain, if the IP address does not exist it asks to extract host address, if the host address is available, AbsoluteUri will parse the URL and returns the IP address.

5.3 Synchronization

Synchronization in Arachnode.net web crawler application is done by implementing logical clocks for each event as shown in Figure 4. For each process or event, an identifier, local date and time stamp is attached to recognize whether there is a change is happened or not. PropertyChangeEventHandler which has two parameters (propertyChange and propertyChanged) receives the changed value and time the change is occur and compare the received information with its local time stamp and adjust its time, after that it distribute the information among other components. For example, when the lastDiscovered time is changed for a particular web page, these changes (time and value) will shortly be recognized by PropertyChangeEventHandler by using onLastDiscoveredChanged method. Continuously it will send the updated information to other system components.

Journal of Theoretical and Applied Information Technology
10th September 2014. Vol. 67 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

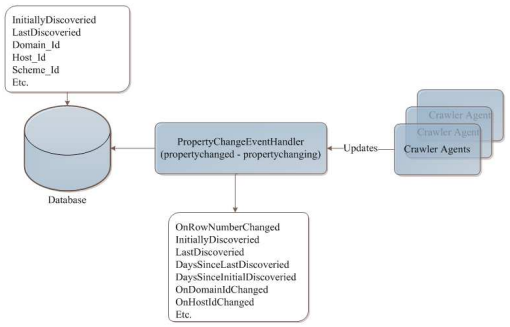

Figure 4: Synchronization Mechanism

5.4 Replication and Consistency

The replication in Arachnode.net web crawler is implemented by using cache mechanism as shown in Figure 5. If a crawler does not find a Discovery (URL, PPT, PDF, etc.) in the local cache, then the crawler checks neighbour crawlers asking for the Discovery, if it is not found there, then the crawler checks the database. If the Discovery is not found in the database, the Discovery request is recorded for future benefit of itself and other connected crawlers. To ensure replication consistency of the cached data with the data that saved in the database, Arachnode.net web crawler implement cache coherence protocol. In which the crawler agents have to check the cached data (e.g. lastCrawlRequest for a particular URL) in Discoveries table before they perform crawl action. It compares the lastCrawlRequest date and time of its copy with the database copy, if it found out that the database copy is newer that its copy it will update the copy. This is important for the crawling performance and efficiency in not duplicating efforts.

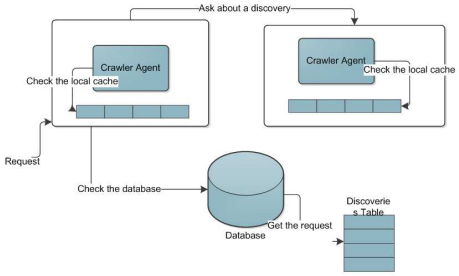

Figure 5: Replication and Consistency

5.5 Fault Tolerance

Aracnode.net web crawler is fault tolerance, meaning that when one crawler (instance) agent goes down, it can still perform crawl actions. When a fault is happened, the URLs that to be executed will re-distributed among the remaining agents. Arachnode.net uses TIMERS to detect crashes agent. If the crawler sent number of request to an agent and in case of that agent did not respond to that request (CrawlRequestTimeoutInMinutes), the crawler will assume that the agent is down. The figure below shows how the fault tolerance is implemented in Arachnode.net Web crawler.

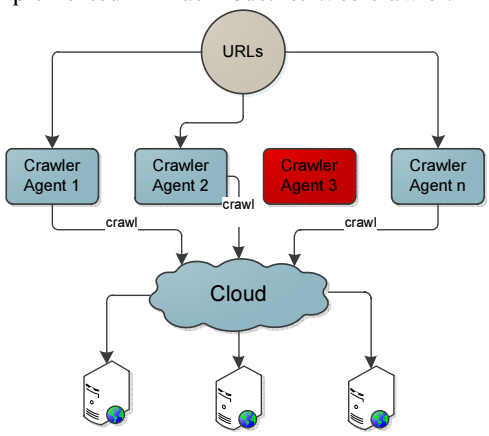

Figure 6: Fault Tolerance

6. IMPLEMENTATION

In this section, we discuss the major changes that we have made in the implementation of arachanode.net distributed web crawler in order to apply GUIDs on it. In the database side, we have implemented GUIDs in the tables that directly participate in URLs crawling process such as webpages, and Disallow Absolute URIs Discoveries tables. However, because there are some tables have foreign key relationships with those particular tables we therefore applied GUIDs in them as well as in some stored procedures. In the crawler engine side we have created new variables and methods that have GUID data type.

The tables that we have applied GUIDs on top of them are the Webpages table which is responsible for storing the URLs that the crawler engine assigns them later to a number of threads. In addition, it also stores the initial and last discovered pages time. GUID column is the primary key of the webpages table, that has an association with webpage ID in such a way that whenever an update is happened in any table that has a relationship with the webpages table, the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

update will set to be send to other tables. The webpage ID is a foreign key of webpages table in all other tables that have relationship with Webpages table. We further applied GUIDs on Disallow Absolute URIs table that responsible for discoveries and content types, and it stores the URIs that we want to run crawling process against them. For example, when it discovers a file it will send the files to the Files table, or when it discovers an image it will direct it to the Images table. Moreover, Disallow Absolute URIs table, stores the URIs that have not been discovered during the crawling process and it shows the reason why crawler could not discovered them. For example, the reason could be, the remote name could not be resolved, or the remote server returned an error 404 (page not found). Additionally, we have added a new variable named @GUID inside a stored procedure called "arachnode_omsp_WebPages_INSERT" for sending the input and the output values that the crawler will store in the Webpage table.

In the crawler (C#) side, we have created a new private variable _lastWebPageGUID that has a GUID data type. Also we have created a new attribute called LastWebPageGUID, the value of this attribute is passed by the variable _lastWebPageGUID by using get and set methods. These methods are responsible for passing the GUID value of the last web page. Moreover, we have overridden the insertWebpage method and we added the variable we have created (_lastWebPageGUID) to this method. This method is responsible for the insertion of the discovered web pages into the GUID database.

7. RESULT AND DISCUSSION

In this section we are going to evaluate the proposed solution for reducing URLs crawling time. We also present the equipment that used during the evaluation process and shows the evaluation results as shown in Figure 7.

In order to evaluate the crawling time of the proposed solution we run the crawling process by using GUIDs and IDs three times. The URLs to be crawled are selected randomly. Each time we have selected 40000/ 60000/ 80000 inserted URLs on the database to evaluate the speed. Then, the time of URLs that inserted in to the database is calculated for the purpose of measurement of the insertion time. Continuously, a bar chat that shows the time difference between crawling by using GUIDs and IDs is drawn.

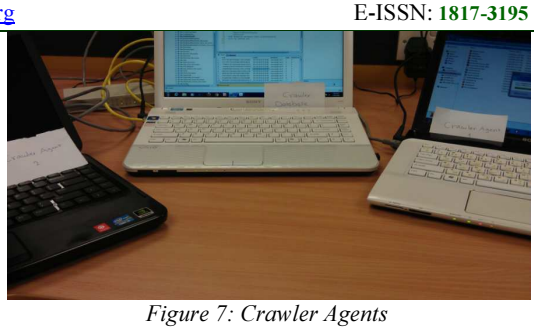

Figure 7: Crawler Agents

The experiments are continued on four laptops computers and their specifications are as follows: two computers have Intel CORE i3 processor 2.20 GHz with 4GB of RAM and one of them have Intel CORE i5 processor 2.6 GHz with 6GB of RAM and the last one is supported with Intel CORE i5 processor 2.1 GHz with 4GB of RAM. The network device that we used in order to connect these laptops is ST Lab 8 Port 10/100M Nway Switch Hub.

Figure 8: Switch Hub

We started the crawling process on four machines or crawler agents. For each crawler agent, we have assigned 10,000 URLs to crawl. We have chosen four URLs to crawl. We started crawling by using IDs and then we started crawling by GUIDs. The result of the first test showed great result. The outcome indicated that using GUIDs are faster than using IDs. Crawling by using IDs takes 5.88 minutes and GUIDs takes 5.30 minutes. In the second experiment, we also started crawling process using four crawler agents. Each agent crawled 15,000 URLs. Continuously, crawling process results have showed that comparing with ID, using GUID reduced the crawling time by 0.33 minutes. We further started another crawl experiment. Similarly this time we used four crawler agents, each one of them was responsible for crawling 20,000 URLs. The results are also proved that using GUID in distributed web crawler can reduce the crawling time up to a minute. Figure 9 shows the reduced time in all three experiments. Based on the three experiments, the average of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

reduced time is up to 0.68 minutes or 7% of reduction rate.

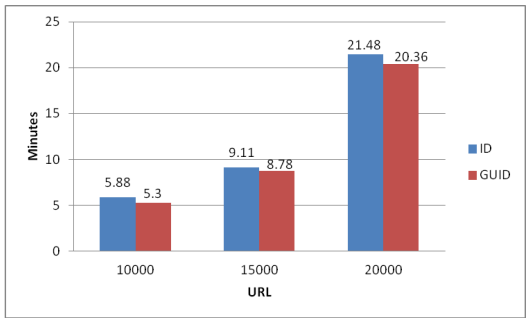

Figure 9: Evaluation of URLs Crawling Time

The results have shown that using GUIDs as an alternate of IDs reduced the crawling time. We claim that this result is because, unlike GUID, ID has to check with the server its next value, which slows down the process of adding new URLs into the database. Whereas GUID does not have to check with the server, instead it is directly inserted into the database, simply because it can assure uniqueness globally. Of course the result with few numbers of URLs is not considered time. However, when we talk about Web crawlers that retrieve billions of Web pages it becomes an issue.

7. CONCLUSION

In this paper we have highlighted the major problems that related to distributed web crawlers. We also study the uses of GUIDs on varieties of application and systems. Then GUIDs is implemented on Arachnode.net web crawler to evaluate its performance in solving query speed of distributed web crawlers. The evaluation results have shown that GUIDs is better compared to ID in distributed Web crawlers' environment as it usage could reduce the crawling time. However, we would recommend using GUIDs only in large distributed web crawlers as ID can fit in small ones, because the reduction of crawling time is no longer exists or disappears in small ones. Furthermore, considering the size of GUID (128 bit) a trade-off should be made between storage spaces and crawling speed.

REFRENCES

[1] M. A. Qureshi, 2010 "Analyzing the Web Crawler as a Feed Forward Engine for an Efficient Solution to the Search Problem in the Minimum Amount of Time through a Distributed," Information Science and Applications (ICISA), 2010 International Conference on, pp. 1,8, 21–23.

[2] M. Gray, 1993 "Internet Growth and Statistics Credits and Background," [Online]. Available: http://www.mit.edu/people/mkgray/net/background.html

[3] M. Burner, 1997 "Crawling towards Eternit, Building An Archive of The World Wide Web," Web Techniques Magazine.

[4] A. Heydon, M. Najork, L. Ave, and P. Alto, 1999 "Mercator : A Scalable , Extensible Web Crawler Architecture of a Scalable Web Crawler," World Wide Web, vol. 2, no. 4, pp. 219–229.

[5] C. Dimou, A. Batzios, A. L. Symeonidis, P. A. Mitkas, and A. W. Spidering, 2006 "A Multi-Agent Simulation Framework for Spiders Traversing the Semantic Web," Web Intelligence, WI IEEE/WIC/ACM International Conference on, vol. pp.736,739, pp. 736 – 739.

[6] P. Boldi, B. Codenotti, M. Santini, and S. Vigna, 2004 "UbiCrawler : A Scalable Fully Distributed Web Crawler," pp. 1–14.

[7] M. S. Kumar, 2011 "Design and Implementation of Scalable , Fully Distributed Web Crawler for a Web Search Engine," vol. 15, no. 7, pp. 8–13.

[8] A. Singh, M. Srivatsa, L. Liu, and T. Miller, 2004 "Apoidea : A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web," Distributed Multimedia Information Retrieval, pp. 126–142

[9] Q. Chen, X. Yang, and X. Wang, 2011, "A PEER-TO-PEER BASED PASSIVE WEB CRAWLING SYSTEM," pp. 10–13,.

[10] J. Bahru, 2007, "Multi-Agent Crawling System (MACS) Architecture for Effective Web Retrieval Siti Nurkhadijah Aishah Ibrahim and Ali Selamat," vol., no. July, pp. 1–4, 2007.

[11] S. S. Vishwakarma, A. Jain, and A. K. Sachan, 2011 "A Novel Web Crawler Algorithm on Query based Approach with Increases Efficiency," vol. 46, no. 1, pp. 34–37, 2012.

[12] N. Singhal and R. P. Agarwal, 2011 "Information Retrieval from the Web and Application of Migrating Crawler," International Conference on Computational

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Intelligence and Communication Systems, p. pp.476,480

[13] N. Singhal, A. Dixit, R. P. Agarwal, and A. K. Sharma, 2012 "Regulating Frequency of a Migrating Web Crawler based on Users Interest," vol. 4, no. 4, pp. 246–253.

[14] S. Mishra, 2011 "A Query based Approach to Reduce the Web Crawler Traffic using HTTP Get Request and Dynamic Web Page," vol. 14, no. 3, pp. 8–14.

[15] B. Zhou, B. Xiao, Z. Lin, and C. Zhang, 2010 "A Distributed Vertical Crawler Using Crawling-Period Based Strategy," IEEE 2nd International Conference on Future Computer and Communication, vol. V1–306, pp. 306–311.

[16] V. Shkapenyuk, 2002 "Design and Implementation of a High-Performance Distributed Web Crawler," Data Engineering, 2002. Proceedings. 18th International Conference on, p. pp.357 – 368.

[17] J. Akilandeswari, 2008 "An Architectural Framework of a Crawler for Locating Deep Web Repositories using Learning Multi-agent Systems," Internet and Web Applications and Services, 2008. ICIW '08. Third International Conference on, pp. 558–562.

[18] B. S. Hoberman, 2008 "Is GUID Good ?," no. September.

[19] C. Lutteroth and G. Weber, 2008 "Efficient Use of GUIDs," Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 115–120.

[20] I. Report, 2001 "JXTA : A Network Programming Environment," IEEE INTERNET COMPUTING, no. June, pp. 88–95.

[21] P. S. et al. 2003 (Eds.), UML 2003 The Unified Modeling Language. Modeling languages and Applications. Berlin: Springer-Verlag, , pp. 2–3.

[22] S. Avancha, A. Joshi, and T. Finin, 2002 "Enhanced Service Discovery in Bluetooth," Computer, vol. 35, no. 6, pp. 96–99.

[23] S. Francisco, S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, 2003 "FAST ' 03 : 2nd USENIX Conference on File and Storage Technologies," 2nd USENIX Conference on File and Storage Technologies.

[24] P. Lucas and J. Senn, 2002 "Toward the Universal Database : U-forms and the VIA Repository.

[25] P. Lucas, D. Widdows, J. Hughes, and W. Lucas, 2005 "Roles in the Universal Database: Data and Metadata in a Distributed Semantic Network.

[26] A. Baid, T. Vu, and D. Raychaudhuri, 2012 "Comparing Alternative Approaches for Networking of Named Objects in the Future Internet *," In Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference on, pp. 298–303.