A Clickstream-based Focused Trend Parallel Web Crawler

F. Ahmadi-Abkenari
Intelligent Software engineering Laboratory, Faculty of Computer Science & Information Systems, University of Technology of Malaysia,
81310 UTM Skudai, Johor, Malaysia,

Ali Selamat
Intelligent Software engineering Laboratory, Faculty of Computer Science & Information System, University of Technology of Malaysia,
81310 UTM Skudai, Johor, Malaysia

ABSTRACT
The immense growing dimension of the World Wide Web induces many obstacles for all-purpose single-process crawlers including the presence of some incorrect answers among search results and the scaling drawbacks. As a result, more enhanced heuristics are needed to provide more accurate search outcomes in an appropriate timely manner. Regarding the fact that employing link dependent Web page importance metrics within a parallel crawler yields a considerable overhead on the overall searching system, and also because such a metric is not able to cover the authorized Web content in dark net and authorized fresh pages, therefore employing these metrics is not an absolute solution within search engines' architecture. This paper proposes the application of a link independent Web page importance metric to govern the priority rule within the crawl frontier through proposing a modest weighted architecture for a focused structured parallel Web crawler (CFP crawler) in which the credit assignment to URLs in crawl frontier is done according to a clickstream-based prioritizing algorithm.

Keywords
Clickstream analysis, Focused crawlers, Parallel crawlers, Web data management, Web page Importance metrics.

1. INTRODUCTION
The dimension of the World Wide Web is being expanded by an unpredictable speed. As a result, search engines encounter many challenges such as yielding accurate and up-to-date results to the users, and responding them in an appropriate timely manner. A centralized single-process crawler is a part of a search engine that traverses the Web graph and fetches any URLs from the initial or seed URLs, keeps them in a priority based queue and then in an iterated manner, according to an importance metric selects the first most important *K* URLs for further processing based on a version of Best-first algorithm. A parallel crawler on the other hand is a multi-processes crawler in which upon partitioning the Web into different segments, each parallel agent is responsible for crawling one of the Web partitions [9]. On the other end of spectrum, all purpose unfocused crawlers attempt to search over the entire Web to construct their index while a focused crawler limits its function upon a semantic Web zone by selectively seeking out the relevant pages to pre-defined topic taxonomy as an effort to maintain a reasonable dimension of the index [8], [18].

The bottleneck in the performance of any crawler is applying an appropriate Web page importance metric in order to prioritize the crawl frontier. Since we are going to employ a clickstream-based metric as a heuristic, our hypothesis is the existence of a standard upon which the authorized crawlers have the right to access the server log files.

In continue, we first review on the literature of parallel crawlers, focused crawlers and the existed link-based and text-based Web page importance metrics by defining the drawbacks of each of them. Then, we briefly discuss our clickstream-based metric since it has been thoroughly discussed in a companion paper. Next, the application of the clickstream-based metric within the architecture of a focused parallel crawler which we call it CFP crawler will be presented.

2. PARALLEL CRAWLRS
An appropriate architecture for a parallel crawler is the one in which the overlap occurrence of download pages among parallel agents is low. Besides, the coverage rate of downloaded pages within each parallel agent's zone of responsibility is high. However, the quality of the overall parallel crawler or its ability to fetch the most important pages should not be less than that of a centralized crawler. For achieving these goals, a measure of information exchange is needed among parallel agents [9]. Although this communication yields an inevitable overhead, a satisfactory trade-off among these objectives should be taken into account for an optimized overall performance.

Selecting an appropriate Web partitioning function is another issue of concern in parallel crawlers. The most dominant partitioning functions of the Web are the URL-hash-based, the site-hash-based and the hierarchical schemes. In URL-hash-based function, assignment of pages to each parallel agent is done according to the hash value of each URL. Under this scheme, different pages in a Web site are crawled by different parallel agents. In site-hash-based function, all pages in a Web site are assigned to one agent based on the hash value of the site name. In hierarchical scheme, partitioning the Web is performed according to the issues such as the geographic zone, language or the type of the URL extension [9]. Based on the definition above, designing a parallel crawler based on the site-hash-based partitioning function is reasonable with regard to the locality retention of the link structure and balanced size of the partitions.

Another issue of concern in the literature of parallel crawlers is the modes of job division among parallel agents. There are different modes of job division which are firewall, cross-over and exchange modes [9]. Under the first mode, each parallel agent only retrieves the pages inside its section and neglects those links pointed to the outside world. Under the second mode, a parallel agent primarily downloads the pages inside its partition and if the pages in its section have been finished, it follows inter-partition links. Under the exchange mode, parallel agents don't follow the

inter-partition links. Instead, each parallel agent communicates with other agents to inform the corresponding agent of the existence of the inter-partition links points to pages inside their sections. Hence, a parallel crawler based on the exchange mode has no overlaps, has an acceptable coverage and has suitable quality, in addition to having a communication overhead for quality optimization.

3. FOCUSED CRAWLRS

There are two different classes of crawlers known as focused and unfocused. The purpose of unfocused crawlers is to search over the entire Web to construct the index. As a result, they confront the laborious job of creating, refreshing and maintaining a database of great dimensions. While a focused crawler limits its function upon a semantic Web zone by selectively seeking out the relevant pages to predefined topic taxonomy and avoiding irrelevant Web regions as an effort to eliminate the irrelevant items among the search results and maintaining a reasonable dimensions of the index. A focused crawler's notion of limiting the crawl boundary is fascinating because "a recognition that covering a single galaxy can be more practical and useful than trying to cover the entire universe" [8].

The user information demands specification in a focused crawler is via importing exemplary Web documents instead of issuing queries. Therefore, a mapping process is performed by the system to highlight (a) topic(s) in the pre-existing topic tree which can be constructed based on human judgment [8]. The core elements of a traditional focused crawler are a classifier and a distiller sections. While the classifier checks the relevancy of each Web document's content to the topic taxonomy based on the naïve Bayesian algorithm, the distiller finds hub pages inside the relevant Web regions by utilizing a modified version of HITS algorithm. These two components, together determine the priority rule for the existing URLs in a priority based queue of crawl frontier [8], [17].

4. LINK DEPENDENT WEB PAGE IMPORTANCE METRICS

PageRank metric as a modification to Backlink count that simply counts the number of links to a page, calculates the weighted incoming links according to Equation (1) in which the pages t_1 to t_n point to page p and c_i is the number of its outgoing links from the page t_i and d is a damping factor which presents the probability of visiting the next page randomly. So, the PageRank or *IR(p)* is computed as shown in Equation (1) [4], [10];

$$IR(p) = (1-d) + d\left[IR(t_1)/c_1 + \ldots + IR(t_n)/c_n\right] \qquad (1)$$

PageRank suffers from a computation of links per page on the subset of already crawled portion of Web in the index. As a matter of fact, to this time, no crawler could claim to make an index near half of the whole Web. As a result, the crawlers are able to calculate *IR`(p)* instead of the real *IR(p)* [10]. In other words, upon downloading more portion of Web, the computed importance of a page will be affected and so the order of pages according to their importance. Moreover the noisy links such as advertisement related links are not among the links with citation objective [15]. So these types of links could mislead the link dependent crawlers. Besides PageRank does not cover the pages in dark net. Although PageRank improves its functionality by covering one category of pages in dark net as form pages by considering them as an entry point to some highly authorized content but it has no solution for another category of pages in dark net as unlinked pages which are the pages with few or no incoming links [2], [16]. So these pages never achieve a high PageRank score even if they contain authoritative content. Besides due to the fact that pages with high number of in-links mostly are older pages which over the time of existence on the Web they accumulate the links, hence these authoritative fresh Web content are disregarded under the PageRank perspective [15].

The TimedPageRank algorithm adds the temporal dimension to the PageRank as an attempt to pay a heed to the newly uploaded high quality pages into the search result by considering a function of time $f(t)$ $(0 \le f(t) \le 1)$ in lieu of the damping factor d. The notion of TimedPageRank is that a Web surfer at a page i has two options: First randomly choosing an outgoing link with the probability of $f(t_i)$ and second jumping to a random page without following a link with the probability of $1\text{-}f(t_i)$. For a completely new page within a Web site, an average of the TimedPageRank of other pages in the Web site is used [25].

Forward link count metric checks the emanated links from a page with the notion that a page with a high forward link score is a hub page [10]. This metric suffers from this defect as a page creator could simply put links to many destinations from the Web pages to mislead the forward link-based crawlers.

The HITS metric views Web page importance in its hub and authority scores. A Web page with high hub score is a page that points to Web pages with high authority scores and a Web page with high authority score is a page that has been pointed to by Web pages with high hub scores. The mutually relationship between these two scores is shown in Equation (2) and (3). In these equations, *a(i)* is the authority score of page i and *h(i)* is the hub score of page i while E is the set of edges in the Web graph [14]:

$$a(i) = \sum_{(j,i)\in E} h(j) \qquad (2)$$

$$h(i) = \sum_{(i,j)\in E} a(j) \qquad (3)$$

The HITS metric has some drawbacks including the issue of topic drift, its failure in detecting mutually reinforcing relationship between hosts, and its shortcoming to differentiate between the automatically generated links from the citation-based links within the Web environment and its no anti-spamming feature. Due to the fact that the pages to which a hub page points to, are not definitely around the same topic, the problem of topic drift is formed. The second problem occurs when a set of documents in one host points to one document on another host. As a result, the hub score of pages on the first host and the authority score of the page on second host will be increased. But this kind of citation cannot be regarded as coming from different sources. Besides, Web authoring tools generate some links automatically that these links cannot be regarded as citation based links. Furthermore, this metric has no anti-spamming feature since it is easy to put outlinks to authoritative pages to affect the hub score of the pages. Although the literature includes some modifications to HITS algorithm such as the research on detecting micro hubs, neglecting links with the same root, putting weights to links based on some text analysis approach or using a combination of anchor text with

this metric, there is no evidence of a satisfactory success of these attempts [3],[5], [6], [7].

5. TEXT-BASED METRICS

The metric of content-query similarity checking measures the textual similarity between the query *q* and page *p* according to the vector space model. In this computation the factor of Inverse Document Frequency (*idf*) is needed which describes the number of times the word appears in the entire collection [10]. Since in this approach, the terms need to carry an absolute measure across the whole collection and indexing the whole Web is still an open issue so the precise calculation of *idf* is impossible [15], [23]. Also due to the noisy environment of the Web and the hurdles to realize semantically related terms, utilizing the mere text dependent approach could be really challenging. Although Latent Semantic Indexing (*LSI*) and the usage of Singular Value Decomposition (*SVD*) is proposed to discover the documents containing the semantically related terms to the query, employing SVD within Web is not an absolute solution with regard to its huge time complexity [11], [15].

6. CLICKSTREAM-BASED WEB PAGE IMPORTANCE METRICS

The literature on clickstream analysis includes the research on this dataset for e-commerce objectives and the usage of clickstream dataset as a Web page importance metric is roughly ignored [13], [15]. In a companion paper we proposed the clickstream analysis for Web page importance determination with a thorough discussion on the challenges of using this dataset as an importance metric and our heuristics to conquer the problems [1]. In another companion paper we propose the application of this metric in parallel crawlers [20]. Here we shortly review this metric. Clickstream-based importance metric is computed according to the total duration of all visits per a Web page during the time period of observation. In other words, the log ranking (LR_{dj}) of a Web page of d_j is the total duration of server sessions per that page (D_{sdj}) as shown in Equation (4) [1]:

$$LR_{dj} = D_{sdj} \tag{4}$$

Regarding the fact that clickstream analysis has no relation with the page content, in order to cover the authorized newly uploaded Web content and authorized Web pages with few or no incoming links (unlinked pages in dark net) in result set, we combine it with the context analysis approach of *Okapi* to put weight to page context. Besides, due to the fact that some Web pages with high LR_{dj} scores may contain news and be of low descriptive nature for user information demands, therefore, the combination of clickstream-based metric with a text analysis approach is vital in order to have a robust decision on Web page importance. Our final importance calculation of each Web page of d_j is shown in Equation (5);

$$I(d_j) = \alpha \times E \times Okapi(d_j, c*) + \beta \times LR_{dj} \tag{5}$$

In Equation (5), two factors of α and β is applied in order to normalize two measures of LR_{dj} and $Okapi(d_j,c^*)$ and make the results of these two measures balanced with each other. Moreover the emphasis factor of E empowers the $Okapi(d_j,c^*)$ in order to consider the importance of Web page content. Here the text analysis approach of *Okapi* measures the similarity of each Web page content to a node in topic taxonomy tree (c^*) instead of a user issued query that will be discussed in detail in section 8.

Upon employing this metric in crawlers, the calculated importance of each page is precise and independent from the downloaded portion of the Web [1]. In our approach, we will go beyond noticing the page importance in its connection pattern and instead, the page credit computation is performed according to an algorithm which worked based on a simple textual log file in lieu of working with matrixes of high dimensions. As a result the time complexity of this algorithm is not high. Moreover due to the fact that pages from the same domain are adjacent URLs in crawl frontier, upon one server log file accessing the importance of these pages will be calculated. Hence, the importance calculation burden will be shared among the pages in the same Web site.

7. PROBLEM STATEMENT

Upon considering the literature on link dependent Web page importance metrics, it can be concluded that applying such metrics within a parallel crawler is not an absolute solution because:

- The calculated importance of each page is not permanent in an observation period since this value is dependent to the downloaded portion of the Web [10].
- Detecting the authoritative recently uploaded Web pages on Web is an obstacle with link-based metrics [15].
- Detecting the authoritative Web pages with few or no numbers of incoming links is another drawback with link-based metrics [15].

Additionally, the major problem in a traditional focused crawler is a suitable credit assignment to the items in crawl frontier [12], [18]. So our different credit allocation approach is employed in this framework to address the mentioned barrier. On the other hand, there is no proposed structure for a focused crawler which works in a parallel mode. Moreover, the presence or absence of a central coordinator within the parallel crawler architecture is an issue of concern in the architecture. Due to the fact that the presence of the central coordinator section organizes the order of fetching the parallel agents' queue, causes an inevitable communication overhead on the overall crawler due to the persistent information exchange among parallel agents and the coordinator component especially when these sections are implemented on physically disperse nodes over the Web graph [9].

So our problem statement is how can we improve the searching process regarding the coverage and total searching time issues meanwhile reduce the overhead on the overall parallel crawler by considering the presence of authoritative pages in dark side of the Web and authoritative recently uploaded pages among search results?

Thus the objective of this paper is proposing an architecture for a focused-based parallel Web crawler (CFP crawler) in which prioritizing the crawl frontier is based on a combination of clickstream analysis and text analysis approaches. Besides, the suggested architecture answers this question that in the absence of a central coordinator, in which order, the overall crawler reads the ordered queue of the parallel agents to achieve the most important

discovered pages by parallel agents at the earliest time of result organizing.

8. OPERATION SYNOPSIS OF THE CFP WEB CRAWLER

Within a focused structure crawler, let's consider G as a Web graph, C as a tree-shaped topic directory, $c \in C$ as each topic node, $D(c)$ as exemplary document set associated with topic c that was defined manually as the best descriptive Web documents for each topic node, c^* as the highlighted mapped node and d_j as a document. Since it is not very easy for users to issue an effective search request, the user imports the Web pages of his/her interest to the searching system (D_u). Upon analyzing the imported documents, CFP crawler highlights (a) node(s) (c^*) in the existed topic taxonomy tree through the mapping process by machine Z as the *Mapper* as shown in figure 1. This process will be done by using an *Inverted List* shown as J in figure 1 through *TF-IDF* scheme. So given a set of user documents of $D=\{d_1,...,d_j,..., d_n\}$, there is a vocabulary V contains all the distinct terms in the semantic region in which the focused crawler is specialized. The Inverted list version is $<id_j, w_i, [o_1,..., o_k]>$ in which the id_j is the document unique identifier, the w_i is the weight of each term i and the rest is the offset of the term i in the document j. For using less memory space for the Inverted List, a method of compression like *Elias Delta coding* [15] could be used for representing the document unique identifier since it is the most space consuming section of an Inverted List.

Furthermore, the CFP crawler will add any distinct term from the corpus which is not included in the vocabulary V into a temporary index of V`. In this step, the section of *Semantic Checker* shown as $S$ in figure 1 will be used to determine the semantically related terms from the $V$ ` with the terms in the main vocabulary of V. Then the not semantically related words with high importance rate will be used by system to edit the topic taxonomy of T and the vocabulary of V. Figure 1 depicts the mapping function of mapper machine (Z) from the imported documents (D_u) to (a) node(s) in topic taxonomy tree of T. After node detection, the pre-existed Web pages associated with the node(s) will be added to the D_u to form the crawling list (CL) as shows in Equation (6) [1]. The CL list is considered as the second level seed URLs and should be divided among parallel agents by the site hash-based partitioning function.

$$CL = D_u + D(c^*) \qquad (6)$$

Figure 2 shows the operational framework of our CFP crawler. The overall framework has been divided to a *central agent* (A), *parallel agents* and the *index* sections. The central agent (A) includes the *topic taxonomy tree* (T), *mapper machine* (Z), *Seed partitionner* (B), *crawl history* (C), and the user imported documents (D_u) as input and the exemplary documents ($D(c)$) for each node. Each parallel agent has the elements of *crawler* (D), *distiller* (G), *classifier* (H), *crawl frontier* (F), *duplication detector* (E) and *coordinator* (I) but because of the space limitation, only the details of one parallel agent are shown. The list below describes each section in the CFP crawler architecture;

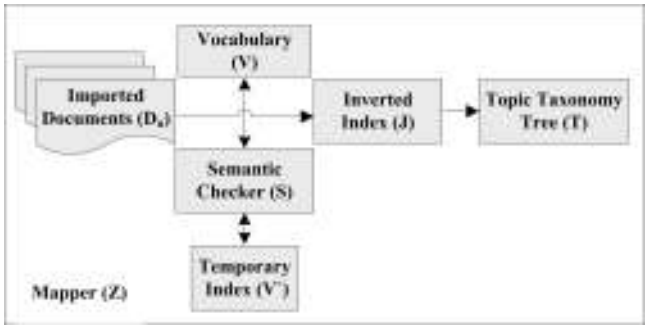

Figure 1. Mapping process by Mapper machine (Z)

- The *topic taxonomy tree* (T) is a tree-shaped topic tree that is constructed by human judgment on one semantic zone like the area of computer science in which each node represents a specific phrase.
- The *seed partitionner* (B) is responsible to partition the second level seed URLs among parallel agents according to the site-hash based partitioning function.
- The *crawler* (D) element is responsible for the process of fetching any unvisited URL from the allocated second level seed URLs in an iterated manner and populates them in the crawl frontier.
- The crawl frontier (F) is a priority-based list corresponding to the Best-First crawling that will be checked by the sections of *crawl history* (C) and *duplication detector* (E) before further processing as described below.
- A *crawl history* (C) or a time-stamp list of visited URLs is maintained by the *central machine* (A) to keep those URLs that their pages have been fetched, as a way to decrease the overlap among different parallel agents.
- A section of *duplication detector* (E) is maintained by each parallel agent to prevent the duplicate URLs in the crawl frontier through maintaining a separate hash-table.
- The decision on the importance of each Web page in crawl frontier is made by *distiller* (G) and *classifier* (H) sections after the crawl frontier is checked by section C and E. The *distiller* is responsible to calculate the page importance based on the clickstream analysis.
- Intermittently, *classifier* (H) measures the importance of each URL in crawl frontier based on the content relevancy of page d_j to the mapped topic(s) of c^* according to *Okapi*(d_j ,c^*)(shown in Equation (7)). Due to the fact that our topic taxonomy tree includes short terms/phrases, we choose the Okapi content relevancy ranking method since it is well suited for the systems where the search query is concise in comparison to other similarity methods like Cosine or Pivoted Normalized Weighting (PNW) [15], [19], [22]. In Equation (7) t_i is a term, f_{ij} is the number of occurrences of the term t_i in document d_j, f_{ic^*} is the number of occurrences of the term t_i in topic node of c^*, N is the total number of documents in the collection, df_i is the number of documents that contain term t_i, dl_j is the length of the document d_j in byte, $avdl$ is the average document length of the collection, k_1 as a parameter is between 1.0 and 2.0, b is a parameter usually set to 0.75 and k_2 is a parameter between 1 and 1000.

$$Okapi(d_j, c^*) = \sum_{ti \in c^*, dj} A \times B \times C \qquad (7)$$

$$A = \ln\frac{N - df_i + 0.5}{df_i + 0.5} \quad B = \frac{(k_1+1)f_{ij}}{k_1(1-b+b\frac{dl_j}{avdl}) + f_{ij}} \quad C = \frac{(k_2+1)f_{ic}^*}{k_2 + f_{ic}^*}$$

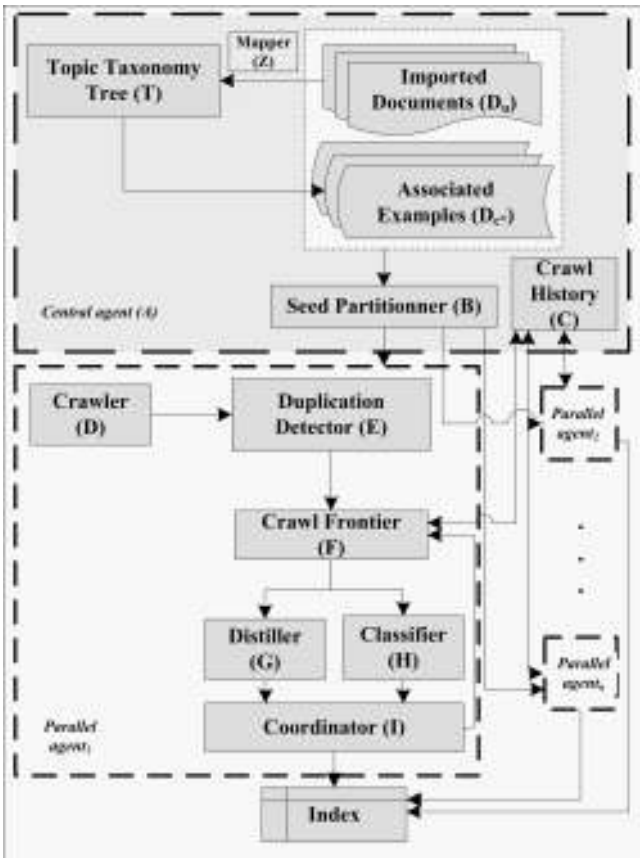

Figure 2. Architecture of the CFP crawler

- The *coordinator* section (*I*) of each parallel agent is responsible to apply a combined importance metric of clickstream and Okapi approaches for credit assignments to the items in crawl frontier. As discussed earlier, the presence of a central coordinator causes an inevitable load on the network due to the unavoidable communication between this component and parallel agents especially when these sections are implemented on disperse nodes over the Web graph and also causes the maintenance complication regarding the dependency of the whole system to this element on the upgrade occasions. Hence we eliminate the presence of this section and instead, there is a smaller part inside each parallel agent to harmonize the results with those of other parallel agents. Therefore the coordinator of each parallel agent is responsible for combining the two measures of LR_{dj} and $Okapi(d_j,c^*)$ to yield an importance measure of $I(d_j)$ as shown In Equation (5). Our experiments will determine a range or an exact value for each of the factors of α, β and E. Upon computing of $I(d_j)$, the queue of each parallel agent is reordered based on the descending value of $I(d_j)$.

The raised question in the absence of a central coordinator is that in which order the overall crawler fetches the ordered queue of each parallel agent. To address this issue, a matter of communication transfer is vital among parallel agents to substitute the role of the central coordinator and to yield an organized and integrated result by the overall CFP crawler. To achieve this objective, let's consider K as the size of the crawl frontiers. As a part of communication, parallel agents should inform each other of the importance of the pages in their queue. If during the communication, the average of $I(d_j)$ for all the K URLs is transferred, a problematic issue is that a queue with few number of very high important pages and many low important pages may have the same average as a queue with many medium important pages. While missing the very high important pages will be a crucial inaccuracy and misjudgment of the overall crawler. To prevent this problem, the communication among parallel agents in our approach, consists sending the average of $I(d_j)$ for a fraction of the pages or the K/L size of the frontier in lieu of transmitting the average of $I(d_j)$ for K URLs. Therefore if m is the number of parallel agents in the CFP crawler and L is the number of frontier divisions and n is the size of each information nugget which is due to transfer, the notification overhead is calculated as shown in Equation (8);

$$Notification\ overhead = n \times m \times (m-1) \times L \qquad (8)$$

As depicted in figure 3, the format of each nugget is in a way that the first position from the left is a flag bit. It is set to one if the average of $I(d_j)$ belongs to the last K/L set of that parallel agent's queue and zero vice versa. The rest of the nugget consists of the correspondent parallel agent identification number, the K/L identification number and the last part is the average of $I(d_j)$ for the K/L set respectively. Equation (9) shows the formulation for n in which λ is the size of I_{avg} value in bits.

$$n = 1 + \lambda + Log2^{mL} \qquad (9)$$

Figure 3. The structure of the notification nuggets

Upon receiving the notification nuggets from other parallel agents, the coordinator section of each parallel agent compares the $I(d_j)$ average in the received nuggets with that of itself. The URLs of the correspondent K/L set to the best $I(d_j)$ average will be sent to the index section and a new nugget will be produced for the next K/L set. The production of the nuggets is done in a synchronous manner in each t seconds intervals. When the time of t arrives, only a parallel agent which sent a set of URLs to the index is eligible to produce a new nugget. This is a normal run of the procedure and any other variation of this rule produces an erroneous condition. Figure 4 depicts an example of a CFP crawler with four parallel agents in which L=4. For the matter of simplicity it only depicts the nuggets received by parallel agent$_1$ and the first transmitted nugget from parallel agent$_1$ in the first iteration of notification transmission.

According to Equation (8), to decrease the notification overhead, the number of parallel agents should not be high. Google employs a few numbers of parallel agents of power two [4]. Other researches also shows considering a few numbers of parallel agents in order to decrease the overhead on the overall systems [21]. Moreover the size of each nugget should be maintained as small as possible. Besides, considering the L as a big number causes more numbers of notification nugget transfers among parallel agents resulting more overhead on the overall system.

Upon sending the URLs with best $I(d_j)$ average in each K/L set to index, the CFP crawler is capable to announce the most important pages fetched from different parallel agents in the earliest time of the result organizing. Besides, the index will receive the partially sorted most important URLs. As a result the sorting time will be decreased dramatically due to the large number of URLs to be indexed. Figure 7 illustrates the algorithm of the CFP crawler.

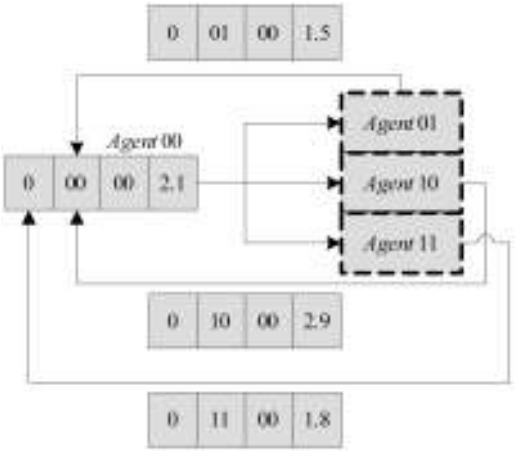

Figure 4. The transmitted and received notification nuggets by parallel agent$_1$ in the first transmission

Figure 5 and 6 illustrates the effect of m and L on the notification overhead. We consider λ =4 for both diagrams, L=4 for showing the impact of the number of parallel agents (m) on the overhead and m=4 for illustrating the impact of frontier division (L) on the notification overhead. In figure 5, the horizontal axis (m) is set to 1, 2, 4, 8 and 16 and the overhead is shown in vertical axis and in figure 6, the horizontal axis (L) is set to 1, 2, 4, 8, 16, 32, 64, 128 and 256 and the overhead is shown in vertical axis. According to the diagram, notification overhead is more affected by m rather than L, since considering m=16 and L=4 produces about the same overhead when L=64 and m=4. Due to the fact that implementing higher L value leads to having more precision on the CFP crawler's performance, thus finding an acceptable tradeoff between the appropriate m and L values and the cost of overall searching system should be taken into account.

9. CONCLUSION

In this paper we proposed an architecture for a focused structured parallel crawler (CFP Crawler) which employs a clickstream-based Web page importance metric. Besides in our approach, parallel agents collaborate with each other in the absence of a central coordinator in order to minimize the inevitable communication overhead.

Our future work consists of more research to minimize the notification overhead to speed up the whole process and to run the crawler on the UTM University's Web site. We intend to determine the precise value for the emphasis factor of E and the two balancing factors of α and β. Moreover, our research on the clickstream-based Web page importance metric is not finished since we are trying to make the metric more robust.

Since the associated examples for each node (D_{c^*}) in topic taxonomy tree is selected based on the higher PageRank score instead of selecting them randomly or considering the pages with high number of outgoing links [24], so in order to evaluate our CFP crawler, we will combine all second level seed URLs into one document in order to have a highly relevant Web source and then calculate the context relevancy of Web pages in result set to this document. Besides, our CFP crawler performance could be evaluated by using precision and harvest rate factors too.

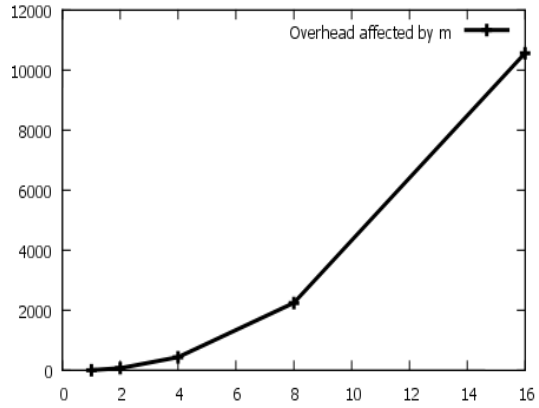

Figure 5. The impact of the number of parallel agents (m) on the notification overhead

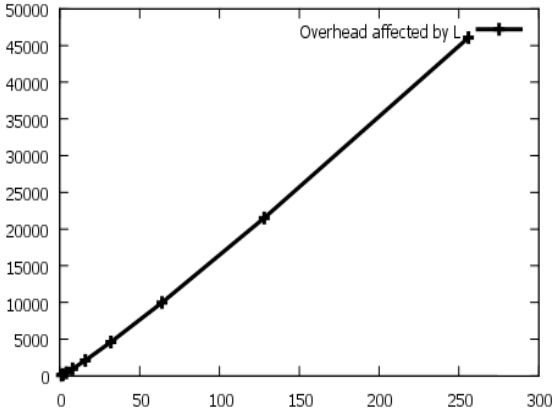

Figure 6. The impact of the number of frontier divisions (L) on the notification overhead

10. REFERENCES

[1] Ahmadi-Abkenari, F., and Selamat, A. July 2010. Application of Clickstream Analysis in a Tailored Focused Web Crawler. International Journal of Systemics and Informatics World Network. Volume 10. pp: 137-144.

[2] Barbosa, L., and Freire, J. 2007. An adaptive Crawler for Locating Hidden-Web Entry Points. In Proceedings of WWW 2007, ACM 978-1-59593-654-7/07/005. Alberta, Canada.

[3] Bharat, K., and Henzinger, MR. 1998. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval. pp: 104-111.

[4] Brin, S., and Page, L. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks, 30(1-7), pp: 107-117.

[5] Chackrabarti, S. 2001. Integrating Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information Extraction. In Proceedings of the 13th International World Wide Web Conference (WWW'01). pp: 211-220.

[6] Chackrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. 1999. Mining the Link Structure of the World Wide Web. IEEE Computer. 32(8): 60-67.

[7] Chackrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., and Kleinberg, J. 1998. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In Proceedings of the 7th International World Wide Web Conference (WWW'7).

[8] Chakrabarti, S., Van den Berg, M., and Dom, B. 1999. Focused Crawling: A New Approach to Topic Specific Web Resource Discovery. Computer Networks, 31(11-16), pp: 1623-1640.

[9] Cho, J., and Garcia-Molina, H. Parallel Crawlers. 2002. In Proceedings of 11th International Conference on World Wide Web. ACM Press.

[10] Cho, J., Garcia-Molina, H., and Page, L. 1998. Efficient Crawling through URL Ordering. In Proceedings of 7th International Conference on World Wide Web.

[11] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Arsman, R. 1990. Indexing by Latent Semantic Analysis. Journal of American Society for Information Science 41. pp: 391-407.

[12] Diligenti, M., Coetzee, F.M., Lawrence, S., Giles, C.L., and Gori, M. 2000. Focused Crawling using Context Graph. In Proceedings of the 26th VLDB Conference. Cairo, Egypt. pp: 527-534.

[13] Giudici, P. 2003. Applied Data Mining, Chapter 8, Web Clickstream Analysis. ISBN: 0-470-84678-X. pp: 229-253. Wiley Press.

[14] Kleinberg, J. 1999. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM 46(5), pp: 604-632.

[15] Liu, B. 2007. Web Data Mining, Chapter 6, Information Retrieval and Web Search. ISBN: 3-540-37881-2. pp: 183-215. Springer Press.

[16] Madhavan, J., Ko, D., Kot, L., Ganapathy, V., Rasmussen, A., and Halevy, A. 2008. Google's Deep Web Crawl. In Proceedings of VLDB'08, Auckland, New Zealand.

[17] McCallum, A., and Nigam, K. 1998. A Comparison of Event Models for Naïve Bays Text Classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization.

[18] Menczer, F., Pant, G., and Srinivasan, P. 2004. Topical Web Crawlers: Evaluating Adaptive Algorithms. ACM Transactions on Internet Technology 4(4), pp: 378-419.

[19] Robertson, S.E., Walker, S., and Beaulieu, M. 1999. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Filtering Tracks. In Proceedings of the 7th Text Retrieval Conference (TREC-7), pp: 253-264.

[20] Selamat, A., and Ahmadi-Abkenari, F. 2010. Application of Clickstream Analysis as Web Page Importance Metric in Parallel Crawlers. In Proceedings of the International Symposium on Information Technology (ITSIM'10). Kuala Lumpur, Malaysia.

[21] Selamat, A., and Selamat, H. 2005. Analysis on the Performance of Mobile Agents for Query Retrieval from Internet. International journal of Information Sciences, Volume 172, Issues 3-4, pp: 281-307.

[22] Singhal, A. 2001. Modern Information Retrieval: A Brief Overview. IEEE Data Engineering Bulletin 24(4), pp: 35-43.

[23] Srivastava, A.N., and Sahami, M. 2009. Text Mining, Classification, Clustering and Applications. ISBN: 978-1-4200-5940-3. CRC Press.

[24] SH. Zhengh, P. Dmitriev, C. and Giles. 2009. Graph based Crawler Seed Selection. WWW 2009, Madrid, Spain, ACM 978-1-60558-487-4/09/04.

[25] Yu, P.S., Li, X., and Liu, B. 2005. Adding the Temporal Dimension to Search- A Case Study in Publication Search. In Proceedings of Web Intelligence (WI'05). pp: 543-549.

1. **INPUT:**
2. D_u is the set of documents imported by user
3. **PARAMETERS:**
4. c^*, is each topic node in topic tree. D_{c^*}, is predefined exemplary documents with each topic node. *CL*, is the second level seed urls. *cl*, is partitioned seed urls. m, is the number of parallel agents. *CF*, is the queue of crawl frontier. *p*, each Web page. *CHistory*, is crawl history. *flag*, is the flag bit in each information nugget. I_{avg}, average of importance for the links in each frontier division. *devisionID*, is the identification of each frontier division. *dID*, is the identification of each frontier division. *agentID*, is the identification of each parallel agent. *aID*, is the identification of each parallel agent. *NuggetTransfer*, is a method of transferring information nuggets among parallel agents. *NuggetInfo*, is a data structure to save the nugget information. *Transfer*, is a method that transfers the best URLs to the index section along with their ranking and the division ID.
5. **OUTPUT:**
6. *LR[]*, is log ranking of array. $T_p[]$, is the context similarity ranking array.

I[], is the final importance array. index[] is a partially sorted array of the most important links.

7. **BEGIN:**
8. $c^* \leftarrow D_u$; // map the imported documents to the topic node
9. $CL = D_u + D_{c^*}$; // formation of second level seed links
10. Partition (*CL, m*) ; // partitioning of seed links among parallel agents
11. **for each** $i \in m$ **do**
12. $CF \leftarrow$ Crawl(*cl*); // crawling process within each parallel agent
13. **for each** $p \in CF$ **do**
14. duplicationDetetion(*CF*) ; // detecting repeated link in crawl frontier
15. **if** $p \in CF$ already exists in *CF*
16. **delete(*p*)**; // remove the repeated link
17. **else**
18. CHistory. **append(*p*)**; // add a fetch page to crawl history
19. CHistory. **refresh()**; // refresh crawl history
20. **continue**; // proceed to next link
21. **repeat** the process from line 13 to 20
22. **for each** $p \in CF$ **do**
23. read(*robot.txt*) ; // respect to politeness policy
24. $LR[] \leftarrow$ Distill(*p*); // calculation of log ranking for each link
25. $T_p[] \leftarrow \sum_{ti \in c^*, dj} \ln \frac{N - df_i + 0.5}{df_i + 0.5} \times \frac{(k_1+1) f_{ij}}{k_1(1 - b + b\frac{dl_j}{avdl}) + f_{ij}} \times \frac{(k_2+1) f_{i\,c^*}}{k_2 + f_{i\,c^*}}$; // context similarity
26. $I[].value \leftarrow \alpha \times LR + E \times \beta \times Tp$; // final page importance calculation
27. $I[].url \leftarrow p$;
28. **repeat** until the end of *CF*
29. **for each** $i \in I$ **do**
30. **sort** *I[]* from the highest *I[].value* to the lowest;
31. **for each** $j \le m$ **do**
32. **for each** $i \le NuggetInfo[].length$ **do**
33. NuggetTransfer(); NuggetInfo[i].*flag*← *flag*; NuggetInfo[i].*I*← I_{avg};
34. NuggetInfo[i].*dID*← *devisionID*; NuggetInfo[i].*aID*← *agentID*;
35. **repeat** until all nuggets is saved
36. **max**(NuggetInfo[i].I_{avg}) ; // the best importance average for set of pages
37. index[].**append ()**← Transfer (*url, I, devisionID*);
38. **Do**
39. NuggetTransfer(); NuggetInfo[i].*flag*← *flag*; NuggetInfo[i].*I*← I_{avg};
40. NuggetInfo[i].*dID*← *devisionID* +1; NuggetInfo[i].*aID*← *agentID*;
41. **Until** (flag=1)
42. **repeat** the process from line 32 to 41
43. **END**

Figure 7. Algorithm of the CFP crawler