Self Ranking and Evaluation Approach for Focused Crawler Based on Multi-Agent System

Hesham Ali
Computers Engineering and Systems Department, Mansoura University, Egypt

Abstract: *The need of better way of retrieving information and dealing with the increasing complexity and volume of information for users is an important research theme. Retrieving information from the www via search engine may be deliberate as the most significant one. Most of the recent efforts that had been done in this area suggest a better solution for general-purpose search engine limitations. That leads to a new generation of search engines called vertical-search engines. However, all current focused crawlers; (crucial component within vertical search engine) crawling strategies introduced high penalties as they consume network bandwidth and resources of hosting servers. This paper introduces a new approach for focused crawling that integrates evidence from both focused crawling and intelligent multi-agent technology. It will help in distributing the computation among data sites, hence overcoming the drawbacks of traditional approaches. We evaluate its performance compared to the client/server-based computation model from perspectives of amount of data transfer and execution time through both simulation and analytical study. The results achieved from the experiments confirm efficiency of the proposed approach. Both of execution time and traffic are reduced in the case of mobile agent compared with client/server model (i.e., eliminating the bottleneck, and overhead problem at crawler site). Furthermore, such combination will also simplify the design and implementation of focused crawlers*.

Keywords: *Search engine, vertical search engine, focused crawler, mobile agents, multi-agents systems.*

Received December 12, 2006; accepted February 27, 200

1. Introduction

Although the internet provides access to a cosmic repository of information, the search and retrieval of specific, useful information and the management of heterogeneous information sources on the Internet is becoming increasingly difficult. Search engines are information retrieval systems that help users to find what they want on the web [1, 15, 2]. The user sends his query to the search engine in a form of keywords. Then, it searches its database and retrieves the pages relevant to the submitted query. Finally, query result is introduced to the user in the form of a ranked list of relevant pages [4, 18]. Most search engines rely on crawlers to traverse the web to collect pages, pass them to the indexer, and then follow links from one page to another web crawlers have the ability to index thousands of pages per day so overcome the limitations of the web portals. They also keep track of changes made to pages visited earlier [7, 23]. For the purpose of illustration, the basic components of a typical search engine are shown in Table 1 and Figure 1. However, search engines suffer from many problems such as, (1) low precision and recall, (2) freshness problem [8], (3) poor retrieval rate [14], (4) long list of result which consumes time and efforts, (5) huge amount of rapidly expanded information which causes a storage problem, and finally, (6) large number of daily hits which makes most search engines not able to provide enough computational power to satisfy each users information need [6]. So, a new strategy of searching is strongly required.

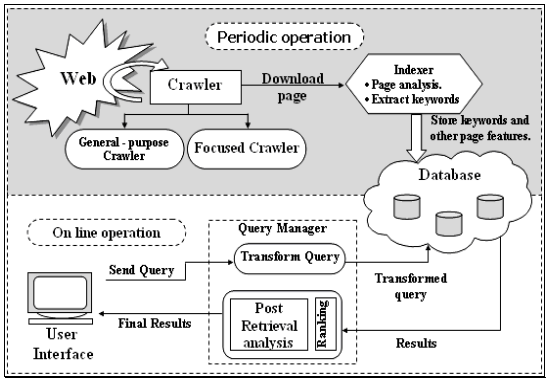

Figure 1. The basic components of a typical search engine.

1.1. Focused Crawler

To overcome those problems, specialized search engines that help users to locate useful information in various domains are proposed. This leads to a new generation of search engines, which are called the domain-specific search (vertical) search engines [26]. Many different vertical search engines are available on the internet, each has its own characteristics, such as law crawler that searches for legal information on the web and building online specialized in searching in the building industry domain on the web.

Table 1. Search engine components.

Component	Description
Crawler	Also called robot, spider or web worm, they are used to retrieve web pages, read them, pass them to the indexer then follow links to the next page [16]
Indexer	It receives the pages retrieved by the crawler, analyzes the various elements of each page like title, headings, body text then extracts the main features of the page, finally dumping the retrieved features into the database [21].
Database	Stores information retrieved from each page the indexer analyze.
Query manager	It has four basic functions: (i) it receives the query and reformulates it into a suitable database query, (ii) retrieve relevant pages from database, (iii) ranking results according to the user query, and (iv) perform analysis on the retrieved pages including categorization and text summarization [19].
User interface	This is the part that the users see. It allows the user to enter his query, sending the query to database via query manager and finally displaying the search result to the user.

In contrast to general-purpose search engines, vertical search engines use a special class of crawlers called focused crawlers [6] as they crawl the web for pages with a specific topic or related to a certain domain. A focused crawling algorithm loads a page, extracts all links inside the page and tries to assign a score for each extracted link then the crawler decides which page to retrieve next by picking the link which was assigned the higher score.

A focused crawler [6, 26] can be considered as a special type of crawlers that seeks out pages about a specific topic and guides the search based on both the content [5] and link structure [19]. It has a priority queue that initialized with a number of seed pages [16]. As illustrated in Figure 2, a focused crawler fetches the page that located on the head of its queue, analyze the page (using parsers to extract keywords and links) and assign a score for each link found in the processed page [24]. The links are sorted according to the scores and inserted into the queue. So, it will organize itself in order to place links with higher scores in the queue head so that they will be processed first. This approach ensures that the crawler moves towards the relevant pages with the assumption that relevant pages tend to be neighbours to each other [26]. The crawler will continue operating as well as its queue has URLs for processing.

The main contribution of this paper arises from the formulation of a new approach for incorporating multi-agents into the area of focused crawling. Hence, it can effectively utilize the discriminative features of the mobile agent as a new network-computing paradigm to solve the problems of the current focused crawling strategies. Such combination will also simplify the design and implementation of focused crawlers. On the other hand, more instances can be constructed using the introduced model.

2. Agent-Based Systems Technology

Agent-based systems technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software systems [25]. This promise is particularly attractive for creating software that operates in environments that are distributed and open,

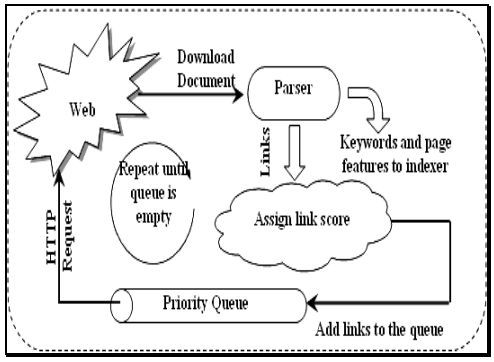

Figure ٢. The structure of a typical focused crawler.

such as the Internet. Currently, the great majority of agent-based systems consist of a single agent. However, as the technology matures and addresses increasingly complex applications, the need for systems that consist of multiple agents that communicate in a peer-to-peer fashion is becoming apparent. Central to the design and effective operation of such Multi-Agent Systems (MASs) are a core set of issues and research questions that have been studied over the years by the many research communities [9, 10].

2.1. Multi-Agent Systems

Sometimes we need several agents to perform a task. Having multiple agents could speed up a system's operation by providing a method for parallel computation. In the case of application that is easily broken into individual tasks -that can be handled by separate agents could benefit from MAS. That is mean the parallelism of MAS can help deal with limitations imposed by time-bounded reasoning requirements. While parallelism is achieved by assigning different tasks or abilities to different agents, robustness is a benefit of MAS that have redundant agents. If control and responsibilities are sufficiently shared among different agents, the system can tolerate failures by one or more of the agents. Another benefit of MAS is their scalability. Since they are inherently modular, it should be easier to add new agents to MAS than it is to add new capabilities to a monolithic system. From a programmer's perspective the modularity of MAS can lead to simpler programming [21]. Rather than tackling the whole task with a centralized agent, programmers

can identify sub-processes and assign control of those subtasks to different agents [17]. To achieve common goals agents need coordination [12]. On the other hand, effective coordination requires cooperation, which in turn can be achieved through communication and organization. Many popular software agents bill themselves as intelligent when in fact they are basic software agents. Intelligent agent technology is a sub-field of Distributed Artificial Intelligence (DAI) [3, 13]. DAI is concerned with issues arising from problem solving by a collection of smart entities/systems and concurrency of computation at different levels. Some of the basic tenets of problem solving by intelligent agents are: (1) each agent may have different knowledge, capabilities, reliability, resources, responsibilities or authority, (2) different agents may perceive the same event or object differently, (3) agents may specialize in or focus on different problems and sub-problems, (4) an important goal is convergence on solutions despite incomplete or inconsistent knowledge or data [21, 22, 17].

2.2. Mobile Agent Technology

Mobile Agents (MA) are programs that are able to migrate from node to another in the computer network (under their own control) to perform the user specified tasks [11, 27]. They can choose when and where to migrate. MA may also interrupt their execution and continue else where in the network. They provide a powerful paradigm for network computing. To put this claim into perspective, consider Figures 3 and 4.

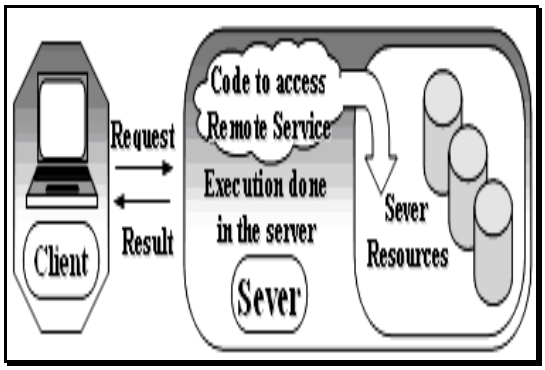

Figure 3. Client server paradigm.

As illustrated in Figure 3, in the client server paradigm, the server provides a set of services that may be accessed remotely by the clients. Such paradigm requires permanent connection between the client and the server to establish the required service. On the other hand, MA (as illustrated in Figure 4) offers more flexible paradigm as the agent is allowed to migrate from one host to another to establish the required task. MA have salient properties that client-server paradigm does not have such as; (1) they reduce the network traffic, (2) they overcome the network latency, and (3) they are executed asynchronously and autonomously since MA are usually independent of the creating process and can operate asynchronously (do not need a permanent connection).

Using MA technologies provides potential benefits to many distributed applications. As MA can be integrated with any distributed application, they found their way to enter many research areas. In [9], a policy-based architecture of a secure mobile agent platform (SECMAP) has been introduced, which provides an isolated, secure execution environment for MA. It also presents a policy-based framework to protect system-level resources and agents against unauthorized access, as well.

In [12], a MA Intrusion Detection System (MAIDS) has been proposed. MAIDS is an agent based distributed Intrusion Detection System (IDS) that detects the attacks against the computer systems. MA are also used in "distributed job coordination" [10], where a set of users who wish to remotely and mutually offer their own computing resources for their time-critical needs. Other recent applications of mobile agents include e-commerce [10, 20], network management [2], grid-computing [21], fault-tolerance [20], and distributed information retrieval [13, 11].

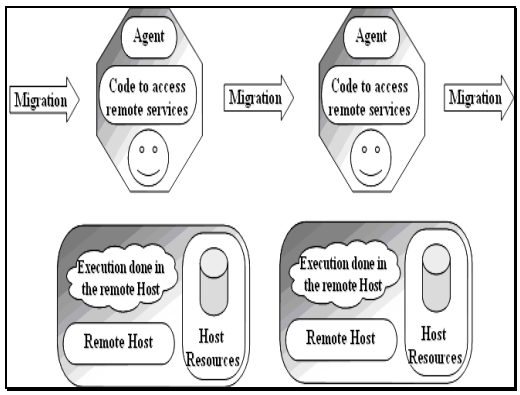

Figure 4. MA paradigm.

3. Problem Definition

Most of the previous focused crawling strategies rely on a client/server computation model. In such model, resources are available at remote servers while execution is done locally. Hence, web pages reside at remote servers, while the crawler resides at the local (client) machine. The crawler is continuously asking remote servers for new pages to process. Precision [15, 21] (Number or relevant web page retrieved divided by number of document retrieved) can be considered as the most important matrices to measure the retrieval effectiveness of IR systems. As the number of document retrieved increased, the efficiency of the search engine will increase (assuming good strategy is used of searching and classification).

For achieving such goal (increase the number of received web pages), the focused crawling is estimated to cover large number of web pages. To perform an exhaustive search, it is require to performing several

recursive crawling, which will lead to accessing huge number of web pages. So, focused crawling strategies in this case will introduce high penalties as they consume network bandwidth (the page must be downloaded even if it is irrelevant). That means bottleneck phenomena will happen within the network. Also, performing the different tasks that constitute the focused crawling processes, require exhaustive memory and storage capabilities, and usually present heavy load to their hosting sites. Hence, it is needed to remove the bottleneck phenomena, and overcome the problem of overload at the client sites (original site where the crawler start), the indexing and classification needs to be decentralized (distributed) in the same manner that the raw information is.

3.1. Plan of Solution

According to the basic client/server computation model, a lot of research and effort had been done to improve the overall performance of the system. This improvement can be done by one of the following directions; (1) partitioning the data or replicating it at cooperating servers (2) caching of data by proxy server (3) the use of mobile code and mobile agent (4) the requirement to add and remove mobile devices in convenient manner [22, 2, 27]. In this work we select the 3rd solution, by employing mobile agent for distributing the computations (pages classification and analysis), which would be performed at the client (crawling site) among the various sites that contain raw data (web page). As a distributed classification is not held at a central point, the indexing and classification are not concentrated in one portion of the web and thus the problem of bottlenecks can be avoided. Since the re-classification process is local, it can be performed under the control of the site creator as often as required.

This means that the currency of the classification can be much better. Shifting the onus of the classification task to the information provider should present no problem on the basis that having bothered to place the information on the web; the site creator presumably wants to ensure that it can be found. In distributed classification scheme, each website is responsible for maintaining its own index i.e., a list of URL's against the specific subject. Thus each web server maintains a small part of the overall index at the focused crawler site.

4. The Proposed Approach

The key idea of the proposed approach is to use MA to distribute the computations (i.e., carry the computations to the data rather than the data to the computations) among the available sites which have a resident agent achieving the task and use mobile agent for returning the results. In other word, instead of moving large volume of data (retrieved page) across www for analysis (moving data to code), MA can be dispatched to remote information sources (moving code to data), data (pages) are then accessed and processed locally, then links extracted with the corresponding scores are send back to the origin. As interactions can take place locally, MA reduces the flow of raw data across the network. Such concept is very compatible with the web environment where very large volumes of data are stored at remote hosts. This will certainly reduce the network overheads and overcome the network latency. These data should be processed locally rather than transferred over the network. Formulating the web crawling as a multi-agent system, and how to combine between focused crawler and multi-agent system to overcome the problem associated to focus crawler is depicted in Figure 5. It represents the architecture model (the main components and their relationships) of the proposed approach. As depicted is this Figure, five type of agents are cooperating to achieve our goal when the communication is held between two different sites. The first site is original site where the crawler starts its work. The second is the remote site where the computation and analysis are accomplished. The employed agents are as follow: Original Site Resident Agent (OSRA), Computations Mobile Agent (CMA), Remote Site Resident Agent (RSRA), Returned Link Mobile Agent (RLMA), and Returned Pages Mobile Agent (RPMA).

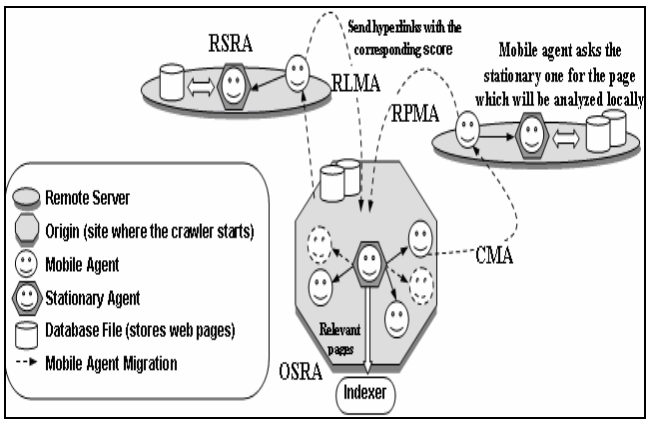

Figure 5. Architectural model of the proposed approach of the focused crawler based on a multi-agent system.

Figure 6 illustrates the different steps of the proposed approach (based on multi-agent system). Figure 7, depicts the fundamental model of the proposed approach. It represents the functional properties of the components represented in Figure 5. The operation within the proposed approach starts by resident agent at the host that receives a query. The responsibility of that agent (OSRA) is (1) maintaining of the queue (extracting the link to be processed and rearrange the exist and incoming link according to its

Table 2. Various types of the agent in the proposed approach.

Agent	Functions	Mobility	Position
OSRA	(1) Maintaining of the queue (extracting the link to be processed and rearrange the exist and incoming link according to its weights. (2) Creating number of mobile agent, and (3) Providing the generated agents with the required domain knowledge.	Stationary	Site that use the crawler
CMA	This agent will make negotiation and transaction with another agent, Moving to the specified URL and carry the information to the resident agent at remote host	Mobile	
RSRA	(1) Page Preparation, prepares (tiding and parsing). The aim of page parsing is to represent the web page in machine readable and easy to process form. The most popular form is the Document Object Model (DOM Tree), (2) Page Analysis, extract the main features of the page so that it is possible to decide whether the page is related to the domain of interest or not, (3) Page Importance: for estimating the importance of the currently processed page by the occurrence of a number of domain concepts inside such page, (4) page link analysis. all links found in the relevant page are identified. LIM performs three different tasks for each identified link. These tasks are; (i) extraction of the link URL, (ii) weight Calculation for each extracted link, (iii)injecting all extracted links with the corresponding weights to the queue, and finally, (5) pass the result to the original site to be added with focus crawler queue.	Stationary	Site correspondi ng to the URL in the Crawlin-g Queue
RPMA	Carrying the relevant page form the remote site to the DB of the original site	Mobile	
RLMA	Carrying the relevant URL form the remote site to the crawling queue at the original site	Mobile	

weights, (2) creating number of MA CMA (this number can be tuned to control the performance), (3) providing CMA with the required domain knowledge (keywords and URL). It recursively does this process until the total number of pages it crawled or the number of links it collected crosses the target specified.

The crawler queue (Q) contains pages that are ready-to-process $Q = \{l_1, l_2, l_3 \ldots l_n\}$, where $l_1, \ldots, l_n$ is the Uniform Resource Locators (URLs), or simply links, for the ready-to-process pages. Q is a best first queue in which the links are ordered according to the link weight. The dispatched mobile agents will carry these parameters to the resident agent at the host that associated to pick URLs.

On the other hand, the responsibility of resident agent at the remote site (RSRA) is to analyze such page to judge on its relevancy to the domain of interest. Five functions are embedded within RSRA, which are: (1) page preparation, (2) page analysis, (3) page importance. Irrelevant pages will be discarded while relevant pages will be returned pack to the original site at the same time it will processed by the last function of this agent (4) page link analysis, and finally, (5) pass the result to the original site via RPMA.

According to topical locality (pages in the same topic tends to be closed to each others in the web graph), as the currently processed page is relevant to the domain of interest, it certainly contains links that point to other relevant pages. In order to compensate the crawler queue with new links, the links found in relevant pages will be extracted, weighted (ranked) according to the domain of interest, and then added to the crawler queue for future analysis.

Hence, the tasks performed within this function (4th) of the RSRA are: (1) page links identification, (2) weight calculation for each extracted link, (3) injecting all extracted links with the corresponding weights to RLMA. Table 2 summarizes the functions and position of the different types of the used agents in the proposed approach.

When an agent arrives in an agent server it talks to a (static) space manager agent to find out what other agents are present in that space. It then interacts with each of those other agents finding out their interests. If it meets with an agent with similar interests, it can exchange details and amend its itinerary accordingly. The details exchanged can be addresses of web pages it has found and the addresses of sites where it is likely to meet more like-minded agents or the agent handle of other agents it should attempt to contact. An agent can contact another agent directly (using the Agent Space messaging service) once it has the target agent's handle.

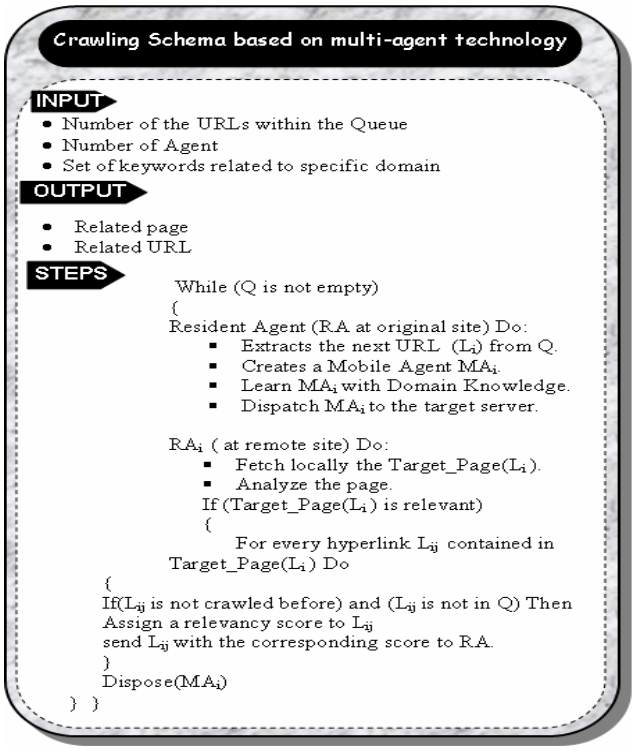

Figure 6. Different steps of the proposed approach.

5. Evaluation and Analysis

Validation of the proposed approach has been done via simulation. This simulation aims to analyze the impact of using multi-agent system on reducing the amount of

data have to be transferred and minimizing the processing at the crawler site.

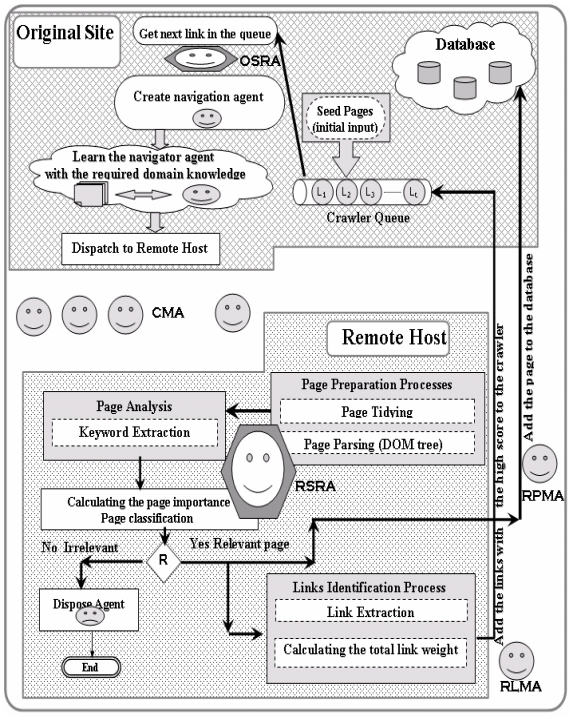

Figure 7: Different steps of the proposed approach.

In our comparative study, we choose three metrics, the Execution Time (ET), Data Transmission Time (DTT), and Data Transmission DTR (traffic), to evaluate the performance of the client/server-based and mobile-agent-based computing models in collaborative processing. At the same time, where the performance of the crawler is depend on its ability to retrieve large number of pages relevant to specific domain (relevancy accuracy is beyond of our scope of this paper), that is mean increasing the number of the used mobile agent. So, the number of the mobile agent is considered as an additional factor in the comparison.

5.1. Performance Metrics

The ET is the time spent to finish a processing task. In the mobile-agent-based model, it starts from the time a mobile agent is created to the time the mobile agent returns with results. In the client/server-based model, it is from the time the clients send out data to the time the data processing is finished and results are generated at the server. The ET consists of three components, T_{trans} (represent the time spent in transferring the migration unit from one node to the other), T_{proc} (represents the processing time), and $T_{overhead}$ (represents the overhead time). In the case of client/server-based model, it is the time spent on file access; and in the case of mobile-agent-based model, it is the time used to create, dispatch, and receive the mobile agent, and at the same time, a few factors that can affect the execution time include the network transfer rate Net_{rat}, the data file size S_f (the size of web page at each sit), the MA size (S_a), the overhead of file access O_F (the time used to read and write a data file), the overhead of mobile agent O_a, the number of agents *m*, and the number of remote sites n that each agent migrates. The amount of transferred data within the network in the client/server and mobile agent model (DTR_{CS}, DTR_{MA}) can be calculated as follow:

$$(DTR)_{CS} = \sum_{i=0}^{m} (S_f)_i \quad (1)$$

while $$(DTR)_{MA} = \sum_{i=0}^{m} (S_a)_i \quad (2)$$

Then DTT, can calculated from the following equation:

$$\textit{Data transmission time} = \textit{latency} + \textit{Transmission delay} \quad (3)$$

where *Transmission delay*: determined by the amount of transferred (DTR_{CS} or DTR_{MA}) divided by the (Net_{rat}) *data transfer rate*, and *Latency*: is the delay incurred from the time the message is sent until it starts to arrive at the destination. It is *composed of two components*, the overhead time in the client/server model is ($2nO_f$; assuming the time used to read and write the data file is the same); and the data processing time. So; the time for data transmission for the client /server model.

$$(DTT)_{CS} = DTR_{CS}/Net_{rat} + 2nO_f \quad (4)$$

Therefore, the total execution time using the client/server based model is

$$(ET)_{CS} = (DTR)_{CS} + 2nO_f + \sum_{i=0}^{m} (T_{proc})_i \quad (5)$$

For the mobile-agent-based computing, the DTT for the mobile agent model:

$$(DTT)_{MA} = DTR_{ma} / Net_{net} + 2(m+n)\, O_a \quad (6)$$

The agent overhead time is = $2(m+n)\, O_a$ as it takes $2mO_a$ for the cluster head to dispatch and receive *m* MA and $2nO_a$ for all the local nodes to send and receive each MA. Therefore, the total execution time using the MA based model is:

$$ET_{MA} = (DTT)_{MA} + 2\, mnO_a + Max\,(T_{proc})_i \quad (7)$$

Based on the previous analytical discussion, the investigated simulation had been implemented in two distinct versions in order to evaluate the comparative performance of two approaches client/server and multi-agent. Both of the number of MA and the size of the data file at remote site (web page) are generated in random fashion that will help in studying the impact of increasing of number of MA, and achieving the reality of simulating the real web pages.

The above simulation results show that the MA based model always perform better than the client/server-based model and in different scenarios. The execution time remains constant for MA model, while it varies according to the page size for client/ server model; it is proportional to the page size. At the

same time the traffic in the MA model will be less than client server model. This is to be expected, as the entire page has to be downloaded; the MA, which is quite small in size, has a constant execution time.

- *Effect of the number of mobile agents on the performance metrics:* we can see from Figures 8, 9 that the data transmission and execution time of the MA model is always less than that of the client/server model because of distributing the computation among the different sites, where the size of agent always less than the size data file. Interestingly, the execution and data transmission time of the MA model decreases as the number of MA increases. Then the execution time begins to climb. This is because more MA will increase the number of cooperating sites to finish the computations, thus reducing the execution time.

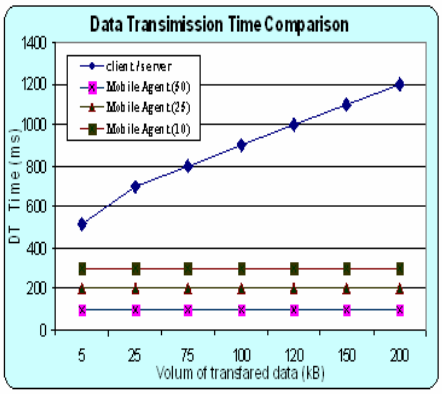

Figure 8. Data transmission comparison.

- *Traffic comparison:* execution time is not the sole criterion of the comparative evaluation of such application of IR. Traffic is highly critical for application used intensively (Google processed 160 million queries daily 2004). As illustrated in Figure 10 as the amount of data increase the generated traffic in client/server will increase gradually, on the other hand such increasing does not have the same effect with MA model. That is because the traffic generated in the MA model will be only the summation of the size of the agents. Traffic for client/server is logically proportional to the page length, while it remains constant for the mobile computing model. Notice that traffic comparison and execution time are very similar this is because the execution time mainly reflects the traffic generated in the cases of MA paradigm.

6. Conclusion

A general-purpose web crawler tries to collect all accessible web pages. Such an approach runs into a scalability problem. Focused crawler tries to solve such problem by restricting itself to search within a specific domain. However, focused crawlers are a greedy entity that consumes both network and local host resources as they depend in a semi-client server paradigm. The goal of our study in this paper is to address issues in focused search engine as information retrieval system based on multiple agent technology. Experimental results show that significant improvements in the performance of the proposed approach can be realised. Where the use of agent technology in the process of searching for information on the Internet could significantly decrease the amount of time it takes users to find relevant information. The proposed approach seems to be flexible, efficient, and easy to be implemented. It also speeds up the crawling process as the resident agent can create more navigational ones. Hence, it simplifies the implementation of distributed and parallel crawling approach. On the other hand it can be considered as an abstract framework that can be utilized to construct multiple crawling instances. Our experimental results showed that, using of multiple-agent technology significantly improved the average execution time compared to the previous client/server computation model.

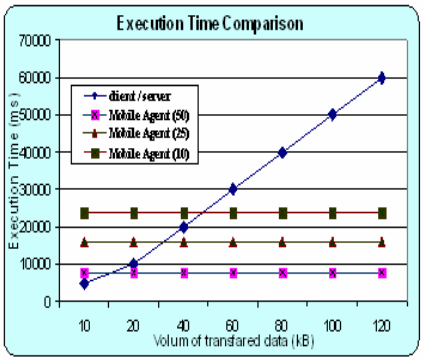

Figure 9. Execution time comparison.

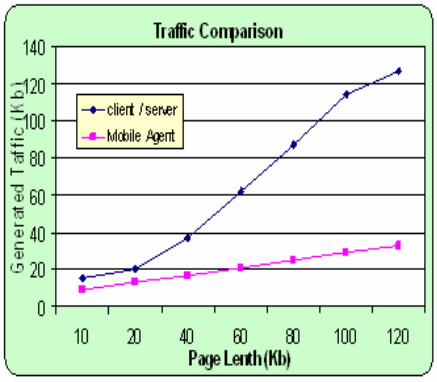

Figure 10. Traffic comparison.

References

[1] Albert B., Carlos C., Alexandru P., and Weber I., "An Analysis of Factors Used in a Search Engine's Ranking," *in Proceedings of the Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Chiba, Japan, pp. 207-305, 2005.

[2] Andronico P., Buzzi M., Castillo C., and Barbara L., "Improving Search Engine Interfaces for

Blind Users: A Case Study," *Journal of Universal Access in the Information Society*, Special Issue on Information Systems Accessibility, Springer, vol. 3, no. 1, pp. 278-319, 2006.

[3] Bieszczad A., Pagurek B., and White T., "Mobile Agents for Network Management," *IEEE Communications Surveys*, vol. 1, no. 1, 1998.

[4] Castillo C. and Ricardo B., "WIRE: An Open-Source Web Information Retrieval Environment," *in Proceedings of the International Workshop on Open Source Web Information Retrieval (OSWIR)*, Compiegne, France, pp. 27-30, 2005.

[5] Chakrabarti S., Joshi M., and Tawde V., "Enhanced Topic Distillation Using Text, Markup Tags, and Hyperlinks," *in Proceedings of the 24th ACM-SIGIR Conference on Research and Development in Information Retrieval,* USA, pp. 92-97, 2001.

[6] Chau M. and Chen H., "Using Content-Based and Link-Based Analysis in Building Vertical Search Engines," *in Proceedings of the 7th International Conference on Asian Digital Libraries (ICADL '04)*, Shanghai, China, pp. 515-518, 2004.

[7] Chen J., David J., Tian F., and Wang Y., "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," *in Proceedings of the ACM International Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, USA, pp. 379-390, 2000.

[8] Cho J. and Garcia-Molina H., "Synchronizing Database to Improve Freshness," *in Proceedings of the ACM International Conference on Management of Data (SIGMOD '00)*, Dallas, Texas, USA, pp. 179-190, 2000.

[9] Fukuda M., Kashiwagi K., and Kobayashi S., "The Design Concept and Initial Implementation of Agent Teamwork Grid Computing Middleware," *in Proceedings of the international IEEE Pacific Rim Conference on Communications, Computers and signal Processing (PACRIM '05)*, pp. 225-228, 2005.

[10] Ganzha M. and Paprzycki M., "Mobile Agents in a Multi-Agent e-Commerce System," *in Proceedings of the 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*, Timisoara, Romania, pp. 207-214, 2005.

[11] Gray R., "Mobile-Agent Versus Client/Server Performance: Scalability in an Information-Retrieval Task," *in Proceedings of International Conference Mobile Agents (MA'2001),* LNCS, vol. 2240, pp. 198-212, Springer, 2001.

[12] Kona M. and ChengXu X., "Framework for Network Management Using Mobile Agents," *in Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS '02)*, pp. 225-229, 2002.

[13] Kosch H. and Mario D., "Content-Based Indexing and Retrieval Supported by Mobile Agent Technology," *in Proceedings of the International Workshop on Multimedia Databases and Image Communications (MDIC '01)*, Amalfi, Italy, pp. 152-165, 2001.

[14] Lawrance S. and Giles C., "Accessibility of Information on the Web," *in Proceedings of the Nature Magazine*, vol. 400, pp. 107-109, 1999.

[15] Marios D., Athena S., Loizos P., "An Investigation of Web Crawler Behaviour: Characterization and Metrics," *Computer Communications*, vol. 28, pp. 880-897, 2005.

[16] Menczer F., Paint G., and Srinivansan P., "Topical Web Crawlers: Evaluating Adaptive Algorithms," *ACM Transactions on Internet Technology*, vol. 4, no. 4, pp. 378-419, 2004.

[17] Michael R., Chen X., and Wong T., "Design and Evaluation of a Fault-Tolerant Mobile-Agent System," *IEEE Intelligent Systems Journal*, vol. 19, no. 5, pp. 32-38, 2004.

[18] Pandey S. and Olston O., "User-Centric Web Crawling," *in Proceedings of the 14th international World Wide Web Conference (WWW '05)*, Chiba, Japan, pp. 114-117, 2005.

[19] Pant G. and Srinivasan P., "Link Contexts in Classifier-Guided Topical Crawlers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 107-122, 2006.

[20] Pham V. and Karmouch A., "Mobile Software Agents: An Overview," *IEEE Communications Magazine*, vo1. 34, no. 1, pp. 26-37, 1998.

[21] Rafael F. and Silva J., "Migration Transparency in a Mobile Agent Based Computational Grid," *in Proceedings of the 5th International Conference of World Scientific and Engineering Academy and Society (WSEAS '05) on Simulation, Modelling and Optimization*, Corfu, Greece, pp. 31-36, 2005.

[22] Satoh I., "A Testing Framework for Mobile Computing Software," *IEEE Transactions on Software Engineering*, vol. 29, no. 12, pp. 564-571, 2003.

[23] Shao Q. and Zatsman M., "Index Structures for Querying the Deep Web," *in Proceedings of the International Workshop on the Web and Databases (WebDB),* San Diego, California, pp. 41-47, 2003.

[24] Srinivasan P., Menczer F., and Pant G., "A General Evaluation Framework for Topical Crawlers," *International Journal of Information Retrieval*, vol. 8, no. 3, pp. 417-447, 2005.

[25] Suat U. and Erdogan N., "A Flexible Policy Architecture for Mobile Agents," *in Proceedings of the 32nd International Conference on Current*

Trends in Theory and Practice of Computer Science (SOFSEM '06), pp. 538-547, 2006.

[26] Thanh T., David H., Nick C., and Kathleen G., "Focused Crawling for Both Topical Relevance and Quality of Medical Information," *in Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM '05)*, Bremen, Germany, pp. 147-154, 2005.

[27] Wang Y., Ranjan S., and Wong J., "Towards the Automatic Generation of Mobile Agents for Distributed Intrusion Detection System," *International Journal of Systems and Software*, vol. 29, no. 1, pp. 1-14, 2006.

Hesham Ali received a BSc in electrical engineering, and MSc and PhD in computer engineering and automatic control from the Faculty of Engineering, Mansoura University, in 1987, 1991 and 1997, respectively. He was assistant professor at the University of Mansoura, Faculty of Computer Science in 1997 to 1999. From January 2000 to September 2001, he was joined as post doctor to the Department of Computer Science, University of Connecticut, Storrs. From 2002 to 2004, he was a vice dean for student affair, the Faculty of Computer Science and Information, University of Mansoura. Since 2004, he has been an associate professor at the Computer Engineering Department, Faculty of Engineering, University of Mansoura. His interests are in the areas of distributed systems, network security, mobile agents, pattern recognition, databases, and performance analysis. He supervises many PhD and Msc students in computer engineering and science. He also leads and teaches modules at both BSc and MSc levels in computer science and software engineering.