UCYMICRA: Distributed Indexing of the Web Using Migrating Crawlers

Odysseas Papapetrou, Stavros Papastavrou, George Samaras

Computer Science Department, University of Cyprus,
75 Kallipoleos Str., P.O.Box 20537
{cs98po1, stavrosp, cssamara}@cs.ucy.ac.cy

Abstract. Due to the tremendous increase rate and the high change frequency of Web documents, maintaining an up-to-date index for searching purposes (search engines) is becoming a challenge. The traditional crawling methods are no longer able to catch up with the constantly updating and growing Web. Realizing the problem, in this paper we suggest an alternative distributed crawling method with the use of mobile agents. Our goal is a scalable crawling scheme that minimizes network utilization, keeps up with document changes, employs time realization, and is easily upgradeable.

1 Introduction

Indexing the Web has become a challenge due to the Web's growing and dynamic nature. A study released in late 2000 reveals that the static and publicly available Web (also mentioned as surface web) exceeds 2.5 billion documents while the deep Web (dynamically generated documents, intranet pages, web-connected databases etc) is almost three orders of magnitude larger [20]. Another study shows that the Web is growing and changing rapidly [17, 19], while no search engine succeeds coverage of more than 16% of the estimated Web size [19].

Web crawling (or traditional crawling) has been the dominant practice for Web indexing by popular search engines and research organizations since 1993, but despite the vast computational and network resources thrown into it, traditional crawling is no longer able to catch up with the dynamic Web. Consequently, popular search engines require up to 6 weeks in order to complete a crawling cycle for the non-paying sites [1]. Moreover, the centralized gathering nature of traditional crawlers and the lack of cooperation between major commercial search engines are two more reasons toward that.

The absence of a scalable crawling method triggered some significant research in the past few years. For example, Focused Crawling [6] was proposed as an alternative method but it did not introduce any architectural innovations since it relied on the same centralized practices of traditional crawling. As a first attempt to improve the centralized nature of traditional crawling, a number of distributed methods have been proposed (Harvest [4, 5], Grub [14]) and are discussed later on.

In this paper, we propose the UCYMicra, a distributed methodology for crawling the Web with the use of mobile agents. The driving force behind UCYMicra is the employment of mobile agents migrating from the search engine to the Web servers, and remaining there to crawl and monitor Web documents for updates. UCYMicra

utilizes related concepts on Distributed Crawling introduced earlier in [4, 11, 12], to suggest a complete distributed crawling strategy aligned with the characteristics of mobile agents, aiming at (a) minimizing network utilization by avoiding the transmission of unnecessary data between Web servers and the search engine site, (b) keeping up with document changes by performing on-site monitoring that allows fast updates on the search engine database, (c) avoiding unnecessary overloading of the Web servers by employing time realization, and (d) being upgradeable at run time.

This introduction is followed by a description of the shortcomings caused by traditional crawling, and Section 3 describes related work. In Section 4, we present our crawling methodology and in Section 5 evaluate its performance. In Section 6, we discuss the advantages of our method, and we conclude in Section 7 with a reference to our ongoing work.

2 The Problem with the Traditional Crawling

For the last decade, single-sourced (or centralized) Web Crawling has been the driving force behind the most popular commercial search engines. The Google [3] and the AltaVista [15] search engines employ an incremental farm of local running crawlers in order to reach a daily dose of a few hundred millions downloads per day, while the allocation of more computational resources is often announced.

However, with the great expansion of the web, especially in the past four years, the traditional crawling model appears inadequate to adjust to the new facts since crawlers are no longer able to download documents with the daily rate required to maintain an updated index of the web. A relatively recent survey suggests that no search engine succeeds coverage of more than 16% of the estimated web size [19].

More specifically, the traditional crawling model fails for the following reasons:

- The task of processing the crawled data introduces a vast processing bottleneck at the search engine site. Distributed processing of data (at the remote Web servers), which would have alleviated the bottleneck, is not available through the current HTTP protocol used by crawlers. Current practices to alleviate this bottleneck are focused in the addition of more computational resources. However, the latter complicates resources coordination and increases the network traffic and cost.
- The attempt to download thousands of documents per second creates a network and a DNS lookup bottleneck. While the latter bottleneck can be partially removed using a local DNS cache, the former can only be alleviated by constantly adding more network resources.
- Documents are usually downloaded by the crawlers uncompressed increasing in this way the network bottleneck. In general, compression is not under full facilitation since it is independent from the crawling task and cannot be forced by the crawlers[1]. In addition, crawlers download the entire contents of a document, including useless information such as scripting code and comments, which are rarely necessary for the document processing.

[1] Document compression before transmission is not always available in Web servers. Web server administrators disable compression to save computational resources

- The vast size of the Web makes it impossible for crawlers to keep up with document changes. The re-visiting frequency for non-sponsored documents, in some of the biggest commercial search engines, varies from 4 to 6 weeks. As a result, search engines deliver old content in search queries.

Moreover, the current crawling model has negative consequences for the complete Internet infrastructure:

- The simultaneous execution of many commercial crawlers generates huge non-user related Internet traffic and tends to overload public Web servers, Domain Name servers, and the underlying Internet infrastructure (routers and networks).
- Not every crawler employs time realization. At peak time, web-servers may be "bombed" with HTTP GET requests generated by a Web crawler. Consequences may vary from a simple delay to a complete denial of service.

3 Related Work

In this Section, we discuss previous work done on achieving better Web coverage. First, we discuss Focused Crawling, a methodology where Web Crawlers limit their search on a particular topic. We then elaborate on Parallel and Distributed Crawling work.

3.1 Focused Crawling

Focused crawling [6] was introduced as an alternative to traditional crawling and aimed at (a) alleviating the scalability setback, and (b) improving search results. Driven by a specific topic (namely a set of keywords), focused crawling targets and monitors only a small fraction of the Web and therefore achieves high-quality indexing on a specific subject. More specifically, focused crawlers download and process only the links that their content is expected to be relevant to the topic of interest.

Significant research has been done in focused crawling with several suggestions for algorithms that filter the URLs and drive the crawlers [7, 10]. However, focused crawling suffers from the same scalability problems with traditional crawling since they both target the same expanding and constantly changing Web.

3.2 Parallel and distributed crawling

The concept of parallel and distributed crawling refers to multiple crawling units running at the same time. In parallel crawling [9], the crawling units run within the local environment of a single company, whereas in distributed crawling, the crawling units are geographically spread. Parallel and distributed crawling is a natural evolution toward accelerating crawling.

Mercator [15], an experimental parallel crawling architecture that was used later on in AltaVista's Search Engine 3, was attacking the scalability problem of crawling by adding more parallel crawlers on the local network. In addition, parallel crawling techniques have been used in the academic version of Google, while sufficient re-

search has been done for more efficient task scheduling and distribution of the crawling task in [9].

Despite the falling prices in hardware and lower connectivity rates, however, parallel distribution of the crawling task within local resources fails to keep pace with the growing Web due to its centralized nature. Downloading of documents and DNS lookup share the same network resources. In addition, coordination of the crawling units and resource management generate a significant network overhead.

To resolve the problems of parallel crawling, non-local distribution of the crawling units was introduced. The first well-documented research dates in early 1994 with the Harvest Information Discovery and Access System [4, 5]. The Harvest prototype was running gatherers (brokers) at the information sources sites (namely the Web servers). Gatherers not only collected data through HTTP and FTP, but they also filtered and compressed data before transmitting it to a centralized database. Unfortunately, while similar software is still used for efficient in-house crawling, the Harvest project failed to become accepted due to lack of flexibility, and administrative concerns.

Grub [14], a recently launched project under the Open-source license, implements a distributed crawling methodology in a way similar to the SETI project [22]. Distributed crawlers collect and process data from local and remote sources and send information to the search engine for updates. However, the control of the whole architecture is centralized since a central server defines which URLs are to be crawled by which distributed crawlers.

J. Hammer and J. Fiedler in [11, 12] initiated the concept of Mobile Crawling by proposing the use of mobile agents as the crawling units. According to Mobile Crawling, mobile agents are dispatched to remote Web severs for local crawling and processing of Web documents. After crawling a specific Web server, they dispatch themselves either back at the search engine machine, or at the next Web server for further crawling. While the model offers speed and efficiency compared to current crawling techniques, important issues are to be resolved such as (a) a more efficient resource management (which is recognized from the writers as important future work), and (b) a more decentralized control of the architecture. In addition, this methodology requires from the mobile agents to constantly move through the Internet since there is no notion of the '*parking agent*'. The absence of a parked/stationed agent at the Web server site precludes an immediate way of promptly monitoring the Web server's documents for changes.

The Google Search Appliance [13], a recently launched commercial package by Google, offers local crawling and processing of a company's Web and document servers by plugging into the local network a Linux computer. This hardware/software solution, however, comes at a great cost and installation overhead. Furthermore, a closed Linux box is an approach that could not serve more that one search engine simultaneously, meaning that a similar package from another search engine would demand another similar box plugged in the local network.

4 The UCYMicra Crawling System

We introduce UCYMicra, a crawling system that utilizes concepts similar to those found in Mobile and distributed Crawling introduced in [11, 12, 4]. UCYMicra extends these concepts and introduces new ones in order to build a more efficient model for distributed Web crawling, capable of keeping up with Web document changes in real time. UCYMicra proposes a complete distributed crawling strategy by utilizing the mobile agents technology. The goals of UCYMicra are (a) to minimize network utilization (b) to keep up with document changes by performing on-site monitoring, (c) to avoid unnecessary overloading of the Web servers by employing time realization, and (d) to be upgradeable at run time.

The driving force behind UCYMicra is the utilization of mobile agents that migrate from the search engine to the Web servers, and remain there to crawl, process, and monitor Web documents for updates. Since UCYMicra requires that a specific mobile agents platform be running at the Web Server to be crawled, it is currently running under a distributed voluntary academic environment spanning in a number of continents. In the rest of this Section, we examine the architecture and functionality of UCYMicra.

4.1 Architecture of UCYMicra

UCYMicra consists of three subsystems, (a) the Coordinator subsystem, (b) the Mobile Agents subsystem, and (c) a public Search Engine that executes user queries on the database maintained by the Coordinator subsystem (the discussion of the public Search Engine is not in the scope of this paper).

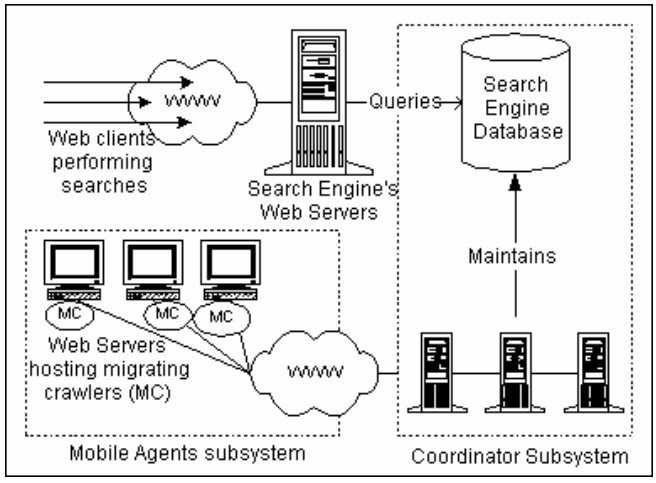

Fig. 1. Architecture of UCYMicra

The Coordinator subsystem resides at the Search Engine site and is responsible of maintaining the search database. In addition, it administers the Mobile Agents subsystem, which is responsible for crawling the Web. The Mobile Agents subsystem is divided into two categories of mobile agents namely the Migrating Crawlers (or Mobile Crawlers) and the Data Carries. The former are responsible for on-site crawling and monitoring of remote Web servers, and the latter for transferring the processed and compressed information from the Migrating Crawlers back to the Coordinator subsystem. Figure 1 illustrates the high-level architecture of UCYMicra.

4.2 The Coordinator Subsystem

Running locally to the search engine database, the Coordinator subsystem is primarily responsible of keeping it up-to-date by integrating fresh data received from the Mobile Agents subsystem. Second, but not less important, the Coordinator monitors the Mobile Agents subsystem in order to ensure its flawless operation. More specifically, the Coordinator subsystem:

1. Provides a publicly available Web-based interface through which Web server administrators can register their Web servers for participating in UCYMicra.
2. Creates and dispatches one Migrating Crawler for a newly registered Web server.
3. Monitors the lifecycle of the Migrating Crawlers in order to ensure their flawless execution.
4. Receives the Data Carriers in order to process their payload and integrate the result in the search engine database.

To avoid a potential processing bottleneck, the Coordinator is implemented to run distributed on several machines that collaborate with a simplified tuplespaces model (similar with Linda's distributed model described in [2]).

4.3 The Mobile Agents Subsystem

The Mobile Agents subsystem is the distributed crawling engine of our methodology and it is divided into two categories of Java mobile agents, (a) the Migrating Crawlers, and (b) the Data Carriers (the initial code of both agents resides at the Coordinator System where it is accessible for instantiation and dynamic installation).

In addition to its inherent mobile capabilities [8], a Migrating Crawler is capable of performing the following tasks at the remote site:

1. *Crawling*: A Migrating Crawler can perform a complete local crawling either though HTTP or using the file system in order to gather the entire contents of the Web server.
2. *Processing*: Crawled Web documents are stripped down into keywords, and keywords are ranked to locally create a keyword index of the Web server contents.
3. *Compression*: Using Java compression libraries, the index of the Web server contents is locally compressed to minimize transmission time between the Migrating Crawler and the Coordinator subsystem.

4. *Monitoring*: The Migrating Crawler can detect changes or additions in the Web server contents. Detected changes are processed, compressed and transmitted to the Coordinator subsystem.

A Data Carrier is a mobile agent dedicated in transferring processed and compressed data from a Migrating Crawler to the Coordinator subsystem for updating the search database. The choice of using mobile agents for data transmission over other network APIs (such as RMI, CORBA or sockets) is the utilization of their asynchronisity, flexibility and intelligence in order to ensure the faultless transmission of the data.

4.4 Deployment of UCYMicra

Figure 2 illustrates how UCYMicra works. A registration procedure is required for a Web server to participate under the UCYMicra crawling system. The Coordinator subsystem provides the interface through which Web server administrators can define the registration parameters. Those parameters are divided into (a) the basic network information of the Web server to be crawled, and (b) the configuration options of the Migrating Crawler and the Data Carriers.

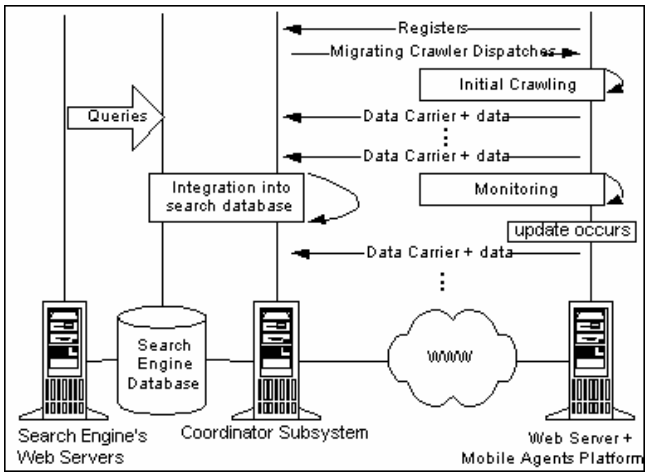

Fig. 2. UCYMicra at Work

The configuration options provide the Web server administrator with the ability to customize the behavior of the Migrating Crawler prior to its creation and relocation to the Web server site. More specifically, the Web server administrator defines[2]:

[2] The Web server administrator can also modify the configuration options at run time, in order to change the behavior of the Migrating Crawlers

1. The time spans in which the Migrating Crawler is allowed to execute. Web server administrators might wish that the crawling and processing of data be executed during off-peak hours.
2. The sleep time between successive crawls used in monitoring the Web server contents for changes or additions. Web servers with frequent changes might consider having a smaller sleep time.
3. The maximum packet size allowed of processed data for the Migrating Crawler to hold before dispatching it to the Coordinator subsystem for integration into the search database.
4. The maximum time-to-hold the processed data (packets) before dispatching it to the Coordinator subsystem. This parameter was introduced in order to avoid the case where processed data may become outdated before dispatching it to the Coordinator subsystem. Data may stall at the Migrating Crawler until the maximum packet size allowed is reached, in which case a data carrier is created, assigned the data, and dispatched.
5. Whether the Migrating Crawler will perform the crawling directly on the file system or through HTTP. For the former method, the absolute path(s) of the Web documents is provided. This method is recommended for Web servers with static Web content. For the second method, the Web server's URL(s) is provided and it is recommended for Web servers with dynamic content. Moreover, a combination of the above two can be used.

Following a successful registration of a Web server, the Coordinator subsystem creates a new instance of a Migrating Crawler and updates it with the registration options. The Migrating Crawler is then allowed to dispatch itself to the Web server.

Upon arrival at the Web server, the Migrating Crawler parks and suspends itself until the beginning of a pre-configured time span for execution arrives. For the first crawl, the Migrating Crawler will load, and process all the available Web documents at the Web server. Processing may include removal of useless html comments, scripting code, or even local extraction of keywords and ranking for each keyword (based on its occurrence frequency and other properties i.e. position on the document, font size and color). The processed data is being compressed and stored in memory until a data packet with the maximum allowed size can be assembled.

A Data Carrier agent is created for every packet assembled, or when the time-to-hold of a packet expires. Data carriers will then dispatch themselves to the Coordinator subsystem where they deliver their payload.

After the first crawl is completed and the Coordinator subsystem has updated the search database, the parked Migrating Crawler monitors the contents of the Web server for changes or additions of new documents. For documents retrieved through the file system, the last update time is used whereas for documents retrieved through HTTP, a conditional HTTP GET is used (using the If-Modified-Since header [16]). In either case, when the Migrating Crawler detects an update or an addition, it crawls and processes the affected documents, and transmits the new documents' index to the Coordinator subsystem. The transmission follows the rules defined in the pre-mentioned parameters (3) and (4), set by the Web server administrator.

4.5 Security in UCYMicra

Security is always an important aspect in every distributed system. In UCYMicra, security concerns are twofold:

- First, a secure runtime environment must be provided to Web server that will host the Migrating Crawlers. Such a runtime must guarantee that the migrating crawlers have access only to the necessary resources of their host, both software and hardware.
- Second, the runtime must also guarantee that the host machine has no access to the Migrating Crawler's processing code and data, in this way preventing malicious altering of the processing results.

Since the scope of this paper is to provide proof of concept for our crawling methodology, UCYMicra still does not employ security. Mobile Agents Security [23, 25], as well as the protection of the mobile agents themselves [18, 21] is a well-researched topic. It should be straightforward to embed security mechanisms in UCYMicra and this is part of our ongoing work.

5 Evaluating UCYMicra

5.1 Initial Experiments

We have performed an initial set of experiments in order to evaluate the performance of the UCYMicra crawling system against traditional crawling. We measure performance in terms of (a) *size of data transmitted* across the Internet, and (b) *total time required*, for the complete crawling of a given set of documents. For the time being, we study only this two obvious metrics and do not experiment with parameters such as document change frequency or update latency.

As opposed to the *size of data transmitted* metric that can be easily calculated in both approaches, the *total time required* metric was difficult to calculate, and depended of the experimental setup. This was mainly due to the following parameters that are beyond our control: (a) the network condition, under which our experiments were performed, and (b) the processing power and the load of the participating web servers during the experiments. We include, however, our time measurements in this paper to provide an additional insight on comparing the two approaches.

For the traditional crawling, the total time required metric stands for the time elapsed from the moment we feed the crawler's URL queue with the URLs to be crawled until all URLs have been successfully crawled and integrated into the search database. For UCYMicra, this is the time elapsed from the moment the Migrating Crawlers are dispatched from the Coordinator subsystem to the web servers that host the URLs to be crawled, until all the crawled contents of the URLs are carried by the Data Carriers back to the Coordinator subsystem and integrated into the search database.

Since it was not feasible to include commercial Web servers in our experiments, we have employed a set of ten Web servers within our voluntary distributed academic environment that span in several continents. Each Web server was hosting an average

of 200 Web documents with an average document size of 25 Kbytes. The previous numbers yield 46.2 Mbytes of document data to be crawled by both the traditional crawling and the UCYMicra approach.

For the traditional crawling experiment, we have developed our own Java-based, multi-threaded crawling system and installed it on our own server machine. To trigger the crawler, we fed its URL queue with the addresses of the participating Web servers, and immediately the crawler started executing. Please note that in this experiment, the documents were downloaded uncompressed. In general, compression between a traditional crawler and a Web server is independent from the crawling task since a traditional crawler cannot enforce it.

For the UCYMicra evaluation, we have developed and installed the Coordinator Subsystem on our server machine. The Migrating Crawlers and Data Carriers were developed using the Voyager Platform [24] and their bytecode was installed on the server machine. Additionally, we have installed the Voyager Runtime on each available Web server machine to be crawled. For practical reasons, we did not require from an administrator to register every participating Web server. Instead, we automatically registered all the participating Web servers under reasonable registration parameters.

After the experiment was initiated, Migrating Crawlers were initialized and dispatched to every registered Web server to locally perform crawling, processing and compression of documents (one Crawler per Web server). Each Migrating Crawler initiated a number of Data Carriers that transferred the processed and compressed data back to the Coordinator Subsystem for integration within the search database.

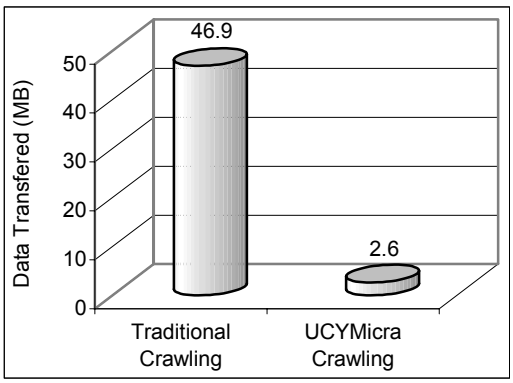

Fig. 3. Size of transferred data. UCYMicra outperforms traditional crawling by approximately generating 20 times less data

Figure 3 displays the size of transferred data by both the approaches in the experiment. Please note that in the 2.6 MB of data that UCYMicra generates, the code of the Migrating Crawlers and the Data Carriers that were dispatched to move the data is included, while in the traditional crawling, we include the data size of the HTTP re-

quests and HTTP response headers. UCYMicra outperforms traditional crawling by generating 20 times less data. This is due to the following reasons:

- The Migrating Crawlers of UCYMicra crawl and process (with the same processing algorithm that the traditional crawler uses) the Web documents locally to the Web server. In this way, only the ranked keyword index of the Web server contents is transmitted to the Coordinator subsystem. In the traditional crawling approach, the crawler downloads the complete contents of a Web server.
- Before transmission to the Coordinator subsystem, the index of the Web server contents can always be compressed. A traditional crawler may request but cannot force a Web server to compress its contents before download. Unless the Web server has compression modules installed and enabled, the documents are delivered uncompressed.

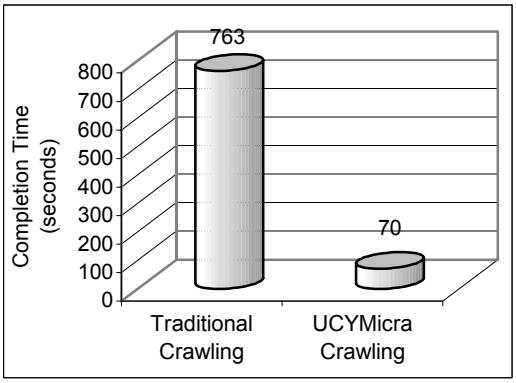

Fig. 4. Total time required for crawling. UCYMicra outperforms traditional crawling by an order of magnitude

Figure 4 displays the total time required by both the approaches to perform crawling. Similar to the previous results, UCYMicra requires one order of magnitude less time than the traditional approach to complete crawling. This is due to the following reasons:

- UCYMicra moves less data than the traditional approach and, therefore, requires less time.
- Processing of the Web documents is done in parallel by the Migrating Crawlers.

5.2 Analytical Experiments

To better understand the meaning of the results of our initial experiments, we performed an analytical second set of experiments by running one more variation of the traditional crawling approach, and three more variations of UCYMicra. The new traditional crawling variation downloads the Web documents compressed. As mentioned before, traditional crawlers cannot force the Web servers to compress documents, however, we were able to perform this experiment since we had control over the participating Web servers.

In the three new UCYMicra variations, the Migrating Crawlers differ in performing:

1. *Neither data processing nor compression*. In this variation, the Migrating Crawlers transmit to the Coordinator subsystem the entire contents of a Web server unprocessed (no keyword index generated) and uncompressed. This variation *emulates* traditional crawling by using the Data Carriers to move the data instead of the remote HTTP GET requests and HTTP response headers. In this case, none of the capabilities of the Migrating Crawlers is exploited (other than continuous monitoring).
2. *Data processing but not compression*. In this variation, the Migrating Crawlers transmit to the Coordinator subsystem the uncompressed keyword index.
3. *Compression but no data processing*. In this variation, the Migrating Crawlers transmit to the Coordinator subsystem the entire contents of a Web server compressed but not processed.

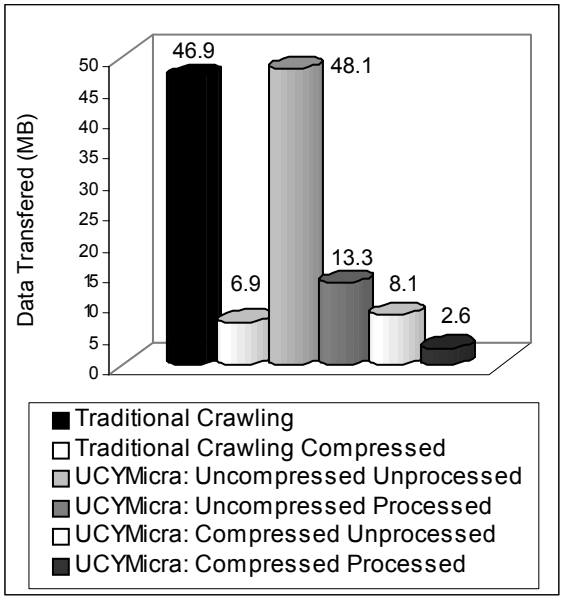

Fig. 5. Analytical performance results for size of transferred data

Figure 5 displays the analytical performance results for size of transferred data along with the initial performance results of Figure 3. Column 1 displays the initial results for the traditional crawling (with no compression). Column 6 displays the initial results for UCYMicra (with both processing and compression).

As seen in the results, it is clear that UCYMicra has to perform either processing (Column 4) or compression (Column 5) in order to outperform traditional crawling. However, both must be performed (Column 6: the original UCYMicra) in order to outperform the variation of the traditional crawling that downloads the Web documents compressed (Column 2).

The UCYMicra unprocessed-uncompressed variation (Column 3) that emulates traditional crawling is outperformed by traditional crawling (Column 1). This is because the execution code of all the Data Carriers that are dispatched to move the data is added to the total data size, which finally exceeds that of the traditional crawling.

Figure 6 displays the analytical performance results for the total time required for crawling.

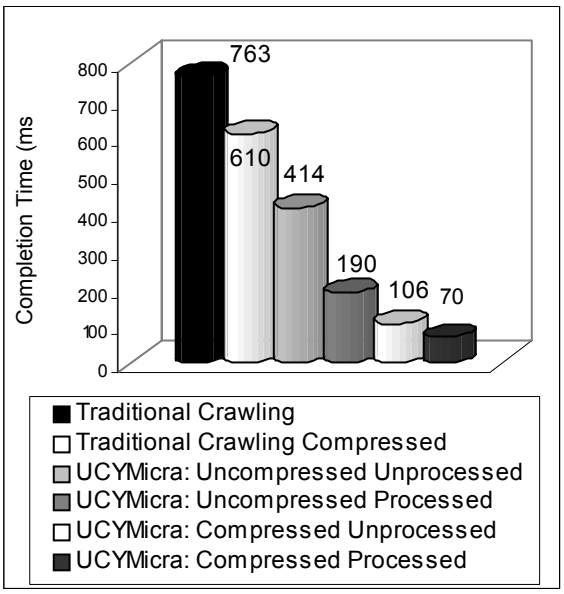

Fig. 6. Analytical performance results for total time required for crawling

In spite of the small size of transferred data that the traditional crawling with compression variation generates, our analytical experiments still prove its inefficiency in downloading Web documents with subsequent HTTP GET requests (Column 2). Even in the absence of remote compression and processing, UCYMicra outperforms the traditional crawling variation (with compression) since it transmits Web documents in a batched manner using the Data Carriers (Column 3).

6 Advantages of UCYMicra

As seen in the previous Section, UCYMicra outperforms traditional crawling without compression by a factor of ten in network utilization and total time required to perform crawling. Besides the performance gains, UCYMicra is a scalable distributed crawling architecture based on the mobile agents technology. More specifically, by delegating the crawling task to the mobile agent units, UCYMicra:

- Eliminates the enormous processing bottleneck from the search engine site. Traditional crawling requires that additional machines be plugged in to increase process-

ing power, a method that comes at a great expense and cannot keep up with the current Web expansion rate and the document update frequency.

- Reduces the use of the Internet infrastructure and subsequently downgrades the network bottleneck by an order of magnitude. By crawling and processing Web documents at their source, we avoid transmission of data such as HTTP header information, useless HTML comments, and JavaScript code. In addition, data is compressed before transmission, which is an option that cannot be forced by traditional crawlers.
- Keeps up with document changes. Migrating Crawlers monitor Web document for changes at their source and updates are immediately dispatched to the Coordinator subsystem. With traditional crawling, several weeks may pass before a Web site is revisited.
- Employs Time Realization. Migrating Crawlers do not operate on their hosts during peak time. As mentioned earlier, our approach requires from the administrator to define the permissible time spans for the crawlers to execute.
- Is upgradeable at real time. Newly developed crawling, processing, or compression code can be deployed over the entire UCYMicra since its crawling architecture is based on Java mobile agents.

7 Conclusions and Future Work

In this paper, we have presented a distributed crawling approach based on mobile agents. Our approach is streamlined not only in improving crawling performance, but also in handling Web document updates by performing onsite monitoring. UCYMicra presents a complete distributed crawling architecture that is efficient, flexible and quite scalable.

Our ongoing work focuses in extending UCYMicra to support a hybrid crawling mechanism that borrows technology from both the traditional and the fully distributed crawling system. Such a hybrid crawling system will support a hierarchical management structure that considers network locality, and will be able to perform traditional crawling in a completely distributed manner. Efficient algorithms for work delegation, administration, and result integration are currently under development.

8 References

1. Altavista Search Engine, Basic submit, Available at http://addurl.altavista.com/addurl/new
2. S. Ahuja, N. Carriero and D. Gelernter: Linda and Friends. IEEE Computer 19 (8), 1986, pp. 26-34.
3. S. Brin, and L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In WWW7, Brisbaib, April 1998.
4. C. M. Brown, B. B. Danzig, D. Hardy, U. Manber, and M. F. Schwartz: The harvest information discovery and access system. In WWW2, Chicago, October 1994.
5. C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz: Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, August 1995.

6. S. Chakrabarti, M. van den Berg, B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. WWW8 / Computer Networks 31(11-16): 1623-1640 (1999).
7. S. Chakrabarti, K. Punera, and M. Subramanyam: Accelerated Focused Crawling through Online Relevance Feedback. In WWW2002, Hawaii, May 2002.
8. D. Chess, C. Harrison, and A. Kershenbaum: Mobile Agents: Are They A Good Idea? IBM research.
9. J. Cho, H. Garcia-Molina, Parallel Crawlers: In WWW2002, Hawaii, May 2002.
10. M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori: Focused Crawling Using Context Graphs. VLDB 2000: 527-534.
11. J. Fiedler and J. Hammer: Using the Web Efficiently: Mobile Crawling. In Proc. of the Seventeenth Annual International Conference of the Association of Management (AoM/IAoM) on Computer Science, Maximilian Press Publishers, pp. 324-329. San Diego, CA, August 1999.
12. J. Fiedler and J. Hammer: Using Mobile Crawlers to Search the Web Efficiently. International Journal of Computer and Information Science, **1**:1, pp. 36-58, 2000.
13. Google Search Appliance. Available at http://www.google.com/services/.
14. Grub Distributed Internet Crawler. Available at www.grub.org.
15. A. Heydon, M. Najork: Mercator: A Scalable, Extensible Web Crawler. Compaq Systems Research Center. In WWW9, Amsterdam, May 2000.
16. Hypertext Transfer Protocol – HTTP/1.0, specification. Available at http://www.w3.org/.
17. B. Kahle: Achieving the Internet. Scientific American, 1996.
18. G. Karjoth, N. Asokan, C. Gulcu: Protecting the Computation Results of Free Roaming Agents. Second International Workshop on Mobile Agents, MA'98, LNCS-1477, 1998.
19. S. Lawrence, C. Lee Giles: Accessibility of information on the web. Nature, 400(6740):107 –109, July 1999.
20. P. Lyman, H. Varian, J. Dunn, A. Strygin, and K. Swearingen: How much information? Available at http://www.sims.berkeley.edu/how-much-info.
21. T. Sander and C. F. Tschudin: Towards Mobile Cryptography. In Proc. of the 1998 IEEE Symposium on Research in Security and Privacy, USA.
22. SETI: Search for Extraterrestrial Intelligence. Available at http://setiathome.ssl.berkeley.edu/.
23. V. Varadharajan. Security enhanced mobile agents: ACM Conference on Computer and Communications Security 2000: 200-209
24. Voyager Web site, by ObjectSpace. Available at http://www.recursionsw.com/products/voyager/voyager.asp.
25. N. Yoshioka, Y. Tahara, A. Ohsuga, S. Honiden: Security for Mobile Agents. AOSE 2000: 223-234.