A Study on the XML-based Dynamic Search Engine for Internet Information Retrieval

Yang-Weon Lee, *Member, KIMICS*

Abstract—In this study, a new-concept search agent system for the WWW by using XML-based technology is proposed. The implementation of the prototype of this proposed system, the comparison with traditional search engines, and the evaluation of the prototype system are also

Index Terms—XML, Internet, Search Engine, Database

I. INTRODUCTION

Recent developments of the WWW have changed the paradigm of information retrieval because those developments have explosively increased the total amount of information available on the Internet. People looking for specific data on the Internet face a more difficult task because as the total amount of information increases on the WWW.

In general, most of search engines can be classified into two types: Categorized Directory and Free-text Keyword search. Nowadays, there are few directory-only search engines: most categorized directory search engines also support keyword search. One of common features of these two-type engines is a centralized system with a large database in order to store categorized directory information and/or indexed keywords.

Free-text search engines mainly have three problems. First, those engines often provide tons of garbage search results because the engines cannot distinguish the categories to which the search keywords belong.

Second, those engines could provide out-of date information, dead links, and already disappeared contents because the data in the database is what agents collected in the past. Thus, some information might have been already expired and/or been deleted.

Third, those engines have scalability problems inherent in their systems. As amount of information increases, the centralized database systems cannot be expanded, and data collecting agents cannot collect data within a feasible period of time. Moreover, it is also expensive to maintain a complicated data structure and huge centralized database system.

The major problems of current search engines are garbage hits, out-of date information, dead links, and a huge centralized database. Potential solutions consist of two aspects. One is a dynamic searching approach instead of searching data from database created in advance. In this sense, we develop the kind of "agents", which searches the information on behalf of users. So, In order to improve and solve the problems of current search engines, the proposed system should contain two important concepts, "XML-Based Search" and "Dynamic contents retrieval".

Manuscript received July 20, 2003 .

Yang-Weon Lee, Department of Information and Communication Eng., Honam University, 59-1, Gwangsangu, Gwangju, Korea (Phone: 062-940-5572, Fax: 062-945-9030, E-mail: ywlee@honam.ac.kr)

II. XML-BASED DYNAMIC SEARCH AGENTS

A solution system for the problems needs to have two key concepts: one is XML-based, the other is dynamic contents retrieval. In this section, we develop the basic structure by comparison of various architectures. In this paper, four potential agent models are considered and discussed. The agent is introduced and analysis of its strength and weakness is presented in the following section. The four models are compared and one of them is chosen to implement the developed system.

A. Client-side agent model

Figure 1 is the diagram of the client-side agent model which an agent program runs on each client machine, establishing http connections to the start page, following hyperlinks, and searching XML documents that match the user's specified search condition. The strength of this model is that each agent program will be easily customized for each user. But network bandwidth might not always be sufficient. So the performance will be slow due to many http connections between clients and the web servers. In addition, users must install the agent program into their machines, set up program configurations, and sometimes may need to update it to the latest version.

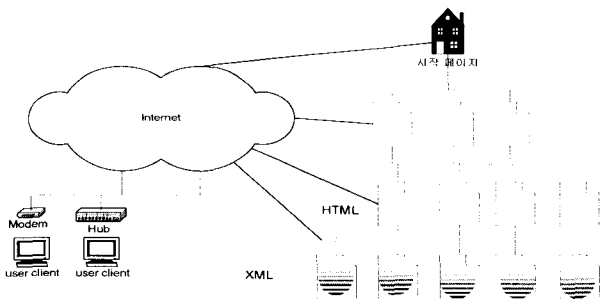

Fig. 1 Client-Side agent model

B. Server-side agent model

Figure 2 is the diagram of the server-side agent model which the agent gets search conditions from user clients, establishes http connections, follows hyperlinks, searches XML documents, and finally sends the search results back to the user clients. Differing from the client-side agent model, most of the HTTP connections are established

between the agent server and the target web site. Only queries and search results are transmitted via the slower links between the agent server and the user clients. Since the agent server connects to the Internet via always-on and wide-bandwidth networks, search performance can be much better than the client-side agent model. The weakness of this model is the concentration of access from many clients.

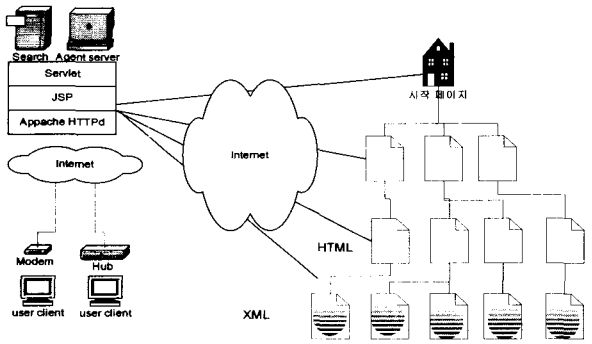

Fig. 2 Server-side agent model

C. Distributed agent model

Figure 3 is the diagram of the distributed agent model which the agent server is deployed and installed to every target website. Similar to the server-side agent model, search queries and search results are only transmitted via slower links between the agent server and the user clients. In addition, since there are only query requests and search responses being transmitted via the Internet, network traffic can be reduced to a very low level. Search time can also be reduced because the distributed servers can search simultaneously, and the searching is performed on local hard drives or LAN that tends to have smaller latency. The weakness is that this model has expensive costs because each web site needs to install the agent server.

D. Peer-to-Peer agent model

Figure 4 is the diagram of the peer-to-peer agent model which the centralized server manages and provides the information of files located in different distributed server. Recently, free-text search engines have faced several problems such as scalability, efficiency, and data freshness problem. Therefore, peer-to-peer based search model has attracted researchers' attention to be a solution to the limitations of current search engine system.

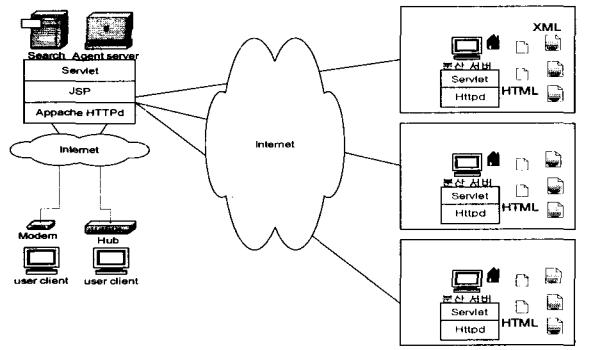

Fig. 3 Distributed agent model

E. Recommended model

Among four proposed models, the recommended model is server-side agent model because the model has strength in both the performance and the feasibility. the following table shows the comparisons among the four models in terms of performance, cost, network usage, and maintenance.

Table 1 Comparison among the four models

	Client side	Server side	Distributed	Peer-to-Peer
Performance	Poor	Good	Best	Better
Cost	Best	Better	Poor	Good
Network usage	Poor	Good	Best	Poor
Maintenance	Poor	Good	Poor	Poor

III. IMPLEMENTATION

In this section, first, the system design concepts are stated. Second, several packages used by the proposed program are explained in detail. Third, the detailed implementation skills and system algorithm are both described and illustrated for clarification.

A. System design

In order to facilitate the uses of the system, N-tier architecture is applied on the proposed system. Client side users use web browsers to communicate with it, Nowadays, web browser is a very common application and built in most popular operating systems. In figure 1, model manages the behavior and data of the application domain, responds to requests for information about its state, and responds to instructions to change state. View manages the graphical and/or textual outputs of the application. Controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate. The MVC(Model/View/ Controller) behavior is necessary to provide a flexible and powerful system.

As illustrated in figure 2, the system architecture of proposed system has followed the MVC model. "XMLRetriever" and "ShowMatch" are the Java Servlet program which are responsible for receiving the requests from the clients (Controller). All the classes under agent program and the agent program itself are the Java program that take responsibility for retrieving the data from the web site(Model). At last, all the search results will be forwarded to JSP(view) for representing.

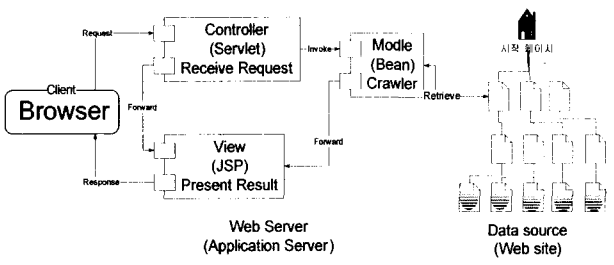

Fig. 4 MVC design paradigm

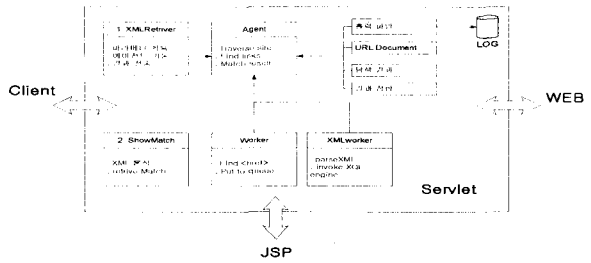

Fig. 5 System architecture

The agent program is multi-threaded to improve the system performance. If the agent visits a web page, it finds the links inside the web page and then visits them one by one. This algorithm is not efficient. It can be improved by using the multi-thread programming as illustrated in Figure 3. If there are many agents visiting all the links simultaneously, for example, if one agent found five links, then it will create another five agents visiting these five links simultaneously. Retrieving the documents simultaneously could benefit the system performance.

B. Flowchart of the algorithm

In order to clarify the algorithm of the agent program, these flowcharts describe some major classed within the agent program. In the case of controller(Fig. 7), first it reads the initial parameters from configuration files and waits for the inputs from the clients. Once the controller receives the request form the clients, it will generate the XML query strings and hand it to the agent program, the crawler as "MODEL" in fig. 4, then when the agent program finish crawling, the controller gets the search results from the agent program and puts them into JSP files for presenting.

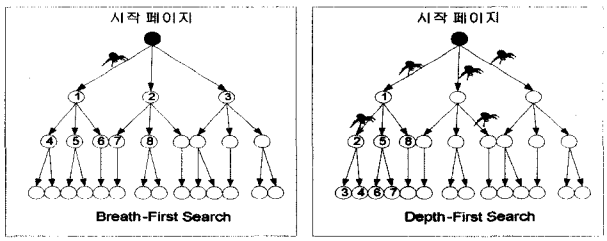

Fig.6 Multi-threaded agent program

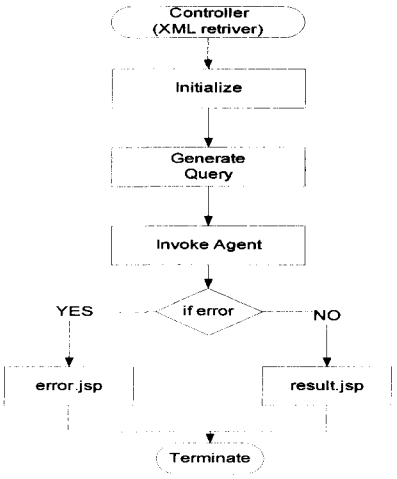

Fig.7 Flowchart of controller program

In the case of model(Fig.5), the agent program)(crawler) receives the XML query strings from the controller and initialization itself. Then it will invoke threads to find more HTML hyperlinks or XML documents. All these links will be put in a queue. The agent program will be terminated only if there are no more threads working and the waiting queue is empty. Figure 6 show the simple algorithm of the threads contained in the agent program. Basically, the Worker threads is used to parse the XML documents and run the XQL Engine to generate the search results. It deserves to be mentioned that all the links will be examined only if they are under the same domain. This idea is applied in order to avoid endless linkage which can often occurs because all the web pages are all hyper-linked with each other as a net.

IV. EXPERIMENTS

A. Specification of the experimental system environment

The following table shows the specification of the experimental system environment the agent server and the proxy cache server.

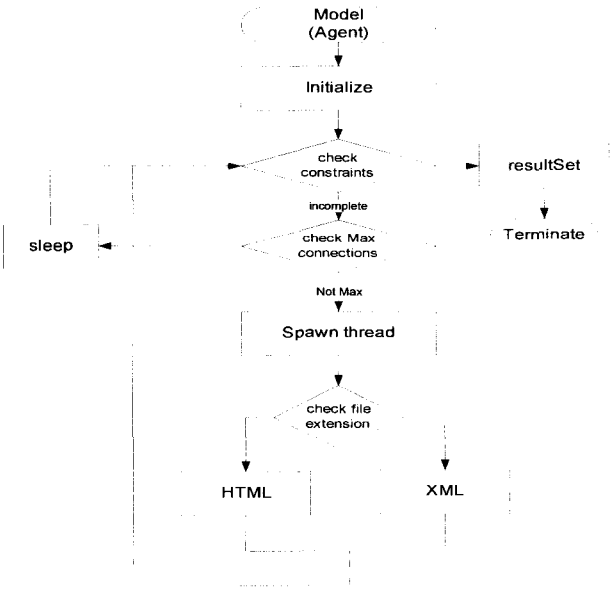

Fig.8 Flowchart of model program

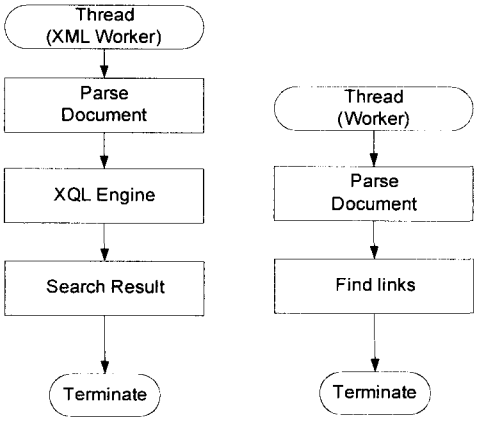

Fig. 9 Flowchart of model program

Table 2 Hardware specifications

	Agent server	Proxy cache server
CPU	Pentium IV	Pentium IV
RAM	128Mb	256 Mb
Network I/F	10Base T	10Base T
Operating System	windows 2000	windows 2000

B. Performance

We present the results of three test cases that illustrate the performance. The first case was a repository created inside the same network. Therefore, the target website and the agent server were connected by Ethernet, 10Mbps. The contents of the repository were small HTML files and XML files, less than 4Kbytes each. The second case was a real XML repository (http://www.honam.ac.kr/), which has the conclusive resource for University Markup Language which is used to exchange school information and data via the Internet. The third case was also a real-world case: the Honam library Metadata site (http://library.honam.ac.kr/) which is a specification for describing library materials.

Table 3 shows the search time for the cases mentioned above. The search time results clearly depend on erogenous variables like network conditions, PC hardware specifications, and load status. In order to eliminate at least some variation in these factors, we tested each case five times, and present the mean values in Table 3.

Table 3 Comparison among experiment cases

	Local experimental Environment	Honam University	Honam Library
Number / Average size of HTML	16 / 458 [bytes]	1 / 10490 [bytes]	34 / 16966 [bytes]
Number / Average size of XML	16 / 2756 [bytes]	173 / 10314 [bytes]	27 / 974 [bytes]
Direct access	2545[ms]	19291[ms]	44637[ms]
Via cache server	1772[ms]	14577[ms]	40305[ms]
Ratio of (w/ Cache) / (w/o Cache)	0.696	0.756	0.903

V. CONCLUSION AND FUTURE RESEARCH

In this paper we analyze possible search mechanisms for large XML repositories, and present a dynamic search agent. We describe its functionality and its architecture, analyze its performance and compare it to other search mechanisms. For future work, we plan on a) expanding agent program to search multiple repositories in parallel, and collate the results, b) explore a peer-to-peer architecture for agent program, wherein each repository would host its own agent and c) explore new ways to improve the performance of agent program.

REFERENCES

[1] R. Khare and A. Rifkin, "XML: A door to automated web applications", IEEE Internet Computing, vol. 1, pp. 78-87, 1997.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", in Scientific American, vol. September, 2001.

[3] N. J. Belkin and W. B. Croft, "Retrieval Techniques", Annual Review of Information Science and Technology, vol. 22, pp. 109-145, 1987.

[4] D. Billsus and M. Pazzani, "Learning Collaborative Information Filters", presented at Machine Learning: Proceedings of the Fifteenth International Conference, 1998.

[5] B. B. Anderson, A. Bajaj, and W. Gorr, "An Estimation of the Relative Effects of External Software Quality Factors on Senior IS Managers' Evaluation of Computing Architectures", Journal of Systems and Software, To Appear.

[6] F. Menczer, "Life-like agents: Internalizing local cues for reinforcement learning and evolution", in Computer Science and Engineering. San Diego: University of California, 1998.

[7] K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability", White Paper, SCS, Carnegie Mellon University, Pittsburgh, 2001.

[8] M. Foley and M. McCulley, JFC Unleashed: SAMS Publishing, 1998.

Yang-Weon Lee

He Received B.S degree in the Department of Electrical Engineering from Chungang University, Korea, in 1982, and the M.S. degree in the Department of Control and Measurement Engineering from Seoul National University, in 1991. In 1998, he received Ph. D. degrees in the Department of Electrical Engineering from Pohang University of Science and Technology (POSTECH), Korea. During 1982-1996, he worked for Agency for Defense Development as senior researcher. Since 1996, he has worked in the Department of Information and Communication Engineering at the Honam University, where he now works as a associate professor. His current research interests include radar signal processing, target estimation and tracking filter design, communication signal processing, distance education using satellite and its Internet application.