

Parallel Web Spiders for Cooperative Information Gathering

Jiewen Luo^{1,2}, Zhongzhi Shi¹, Maoguang Wang^{1,2}, and Wei Wang²

¹ The Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences,
PO Box 2704-28, Beijing, 100080, China

² Graduate School of Chinese Academy of Sciences
luojw@ics.ict.ac.cn

Abstract. Web spider is a widely used approach to obtain information for search engines. As the size of the Web grows, it becomes a natural choice to parallelize the spider's crawling process. This paper presents a parallel web spider model based on multi-agent system for cooperative information gathering. It uses the dynamic assignment mechanism to wipe off redundant web pages caused by parallelization. Experiments show that the parallel spider is effective to improve the information gathering performance within an acceptable interaction efficiency cost for controlling. This approach provides a novel perspective for the next generation advanced search engine.

1 Introduction

As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single spider in the search engine. Therefore, many search engines often run multiple processes in parallel to perform the task. We refer to this type of spider as a parallel spider. This approach can considerably improve the collection efficiency. However, it also takes great challenges in how to control the redundant pages by parallel spiders and minimize the efficiency cost [1,2,4,6].

In order to investigate this problem, we explore to design the parallel spiders as a multi-agent system. A multi-agent system is one in which a number of agents cooperates with each other to achieve a global objective in a complex and distributed environment. With the cooperation, spider agents can coordinate information collection actions, which effectively avoid the page redundancy caused by parallelization. To test the performance of this approach, we implement a parallel spider prototype based on multi-agent platform MAGE [3] and conduct a series of experiments. Experiment results demonstrate it is effective to improve the search performance within acceptable efficiency cost.

2 Parallel Web Spider Model

Based on the framework of MAGE, we construct a parallel model for web information gathering. Figure 1 shows the model's architecture. It contains three kinds of

agents: managing agent (Facilitator), spider agent and index agent. They all inherit the generic Class Agent and can correctly run on the MAGE platform.

As mentioned above, MAGE provides the multi-agent environment support for the whole system. It includes white page service, message transmission service and agent life cycle management service etc. With this software, we need not consider the agent run-time environment and can focus our work on the function implementation of the parallel Spider.

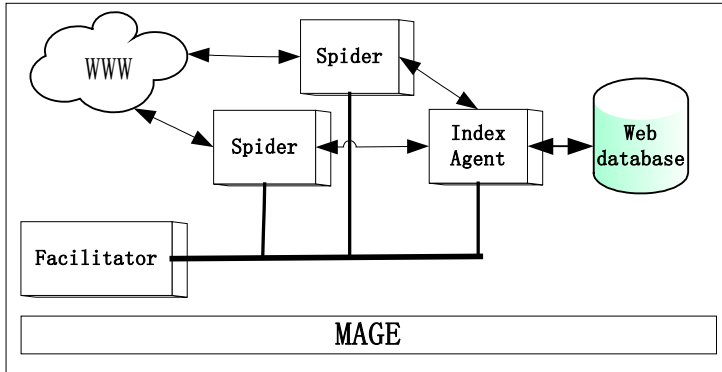


Fig. 1. Architecture of parallel Web Spider

Facilitator is the center of management and communication in the system. It is a special agent that is used to manage the seed URL resources and other agents. Hence, facilitator mainly acts as the server between agents, affording the services including query of addresses and coordination of multi-agent actions. Spider agent starts its work after receiving the allow information from the facilitator. It fetches web pages according to URL and parses their hyperlinks. In our parallel model, we apply the FIPA ACL as the standard communication language for spider agent and facilitator. Through dynamic assignment process of the facilitator, spider agents can effectively avoid the redundant pages.

3 Dynamic Assignment Process

In this section, we discuss how to avoid redundant pages in the parallel spider system. When multiple spiders download pages in parallel, different spiders may download the same pages multiple times. The overlap situation is demonstrated by figure 2.

In order to deal with this problem, we implement the dynamic assignment using multi-agent coordination mechanism. A central facilitator logically divides the Web into different domains (e.g. sina.com and yahoo.com) and dynamically assigns each domain to a spider as the seed URL to download. When MAGE initiates a new spider and assigns it a seed URL, it first checks the facilitator's URL domain to check whether the seed URL has been assigned to other spiders. If not, the new spider adds the seed URL and correctly registers to the facilitator. When facilitator agent receives the registration message, it records the seed URL to its domain in order to avoid redundant allocation in the future.

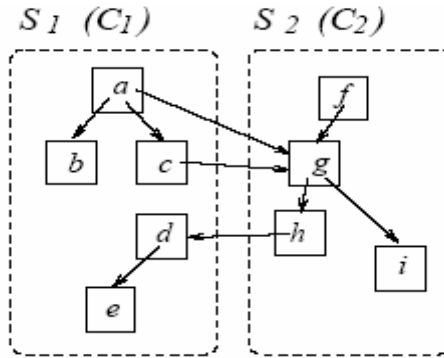


Fig. 2. Site S_1 is crawled by C_1 and site S_2 is crawled by C_2

Dynamic Assignment Procedure

Procedure *FacilitatorDynamicAssignment* ;

Begin

Facilitator ();

Message=Facilitator.WaitForMessage ();

If (Message.type= "Registration")

Record (SpiderName, IP, seedURL);

Put (seedURL, DomainLibrary);

Else If (Message.type= "CheckSeedURL")

url=Message.url; //get the spider's seed url from the message

flag=Check (url); //check whether the url is existent in domain library

If (flag= true)

SendMessage (Spider, Refuse," The seed URL is existent");

Else SendMessage (Spider, Allow," You can use the seed URL");

Wakeup (Spider);

End

Procedure *SpiderDynamicAssignment* ;

Begin

Spider (SpiderName, IP)

seedURL=Spider.getSeedURL ();// Get the seed URL from user Input

SendMessage (Facilitator, "CheckSeedURL");

Message=WaitForMessage (); //Get the response from facilitator

If (Message.type= Refuse)

System.out.println (Message.Content);

Else SendMessage (Facilitator, "Registration") ;

Do

url=Search (seedURL);

If (url is not consistent with seedURL)

SendMessage (Facilitator,"CheckSeedURL");

Message=WaitForMessage ();

If (Message.type= Refuse)

seedURL=FindBrotherNode (url); // Call back and Find the Brother Node

Else New Spider (SpiderName, IP) // Generate a spider for the new seedURL

SendMessage (Facilitator, "Registration");

Until (NumOfSite>MaxOfVisitedSite || SearchDepth> MaxOfSearchDepth)

End

Fig. 3. Pseudocode of Dynamic Assignment

4 Implementation and Experiments

4.1 Implementation

We have implemented the parallel spider system based on our multi-agent platform MAGE. All programs are written in Java to be independent with operating system. Every spider inherits **Agent Class** in MAGE and has the basic functions such as communication using FIPA ACL message and life cycle.

Figure 4 demonstrates the spider agent screenshot. A Spider thread runs in a separate thread to conduct the Web search. Separate threads are used so that the main GUI frame can continually update the search tree and process the stop search request. As the Spider runs, it continually adds hyperlink nodes to the display tree. When the search completes, users can view the site's vital statistics, including the keywords present, total text characters, total images, and total links.

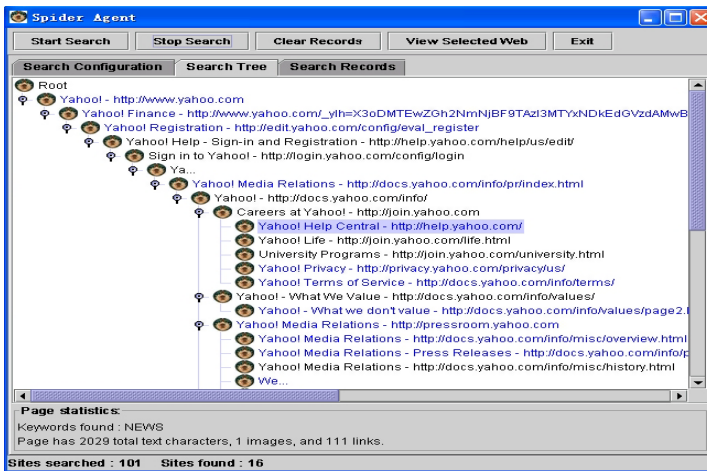


Fig. 4. Screenshot of Spider Agent

4.2 Experiments

In order to test the performance, we conduct a series of experiments. In the first experiment, the three spiders run independently. That means we permit the redundant web pages and the facilitator agent does not work as the coordinator. We record the number of search pages and redundant pages every 30 seconds. The experiment results are showed in figure 5. The average redundant rate of the parallel web paper is 4.1 %. Because most search engine has to collect millions pages in its tasks, this redundant rate should be seriously considered in order to save a mount of storing space.

In the second experiment, we adopt the dynamic assignment mechanism and make the spiders can coordinate with each other to wipe off the redundant pages phenomenon. We can completely avoid the web page repetition through the facilitator's coordination. However, although avoiding redundancy, it takes efficiency cost at the interaction process. This effect is reflected by the decrease of search number at a fixed time scope.

The experiment result is showed in the figure 6. From the result ,we find in most time area, the efficiency cost are between 1% and 5%. That means we have to sacrifice some efficiency if we want to wipe off the redudant pages in the parallization process. However, it is worthy to adopt the dynamic assignment since we can control the efficiency cost in a proper scope, which will be made up by the upgrade of hardware.

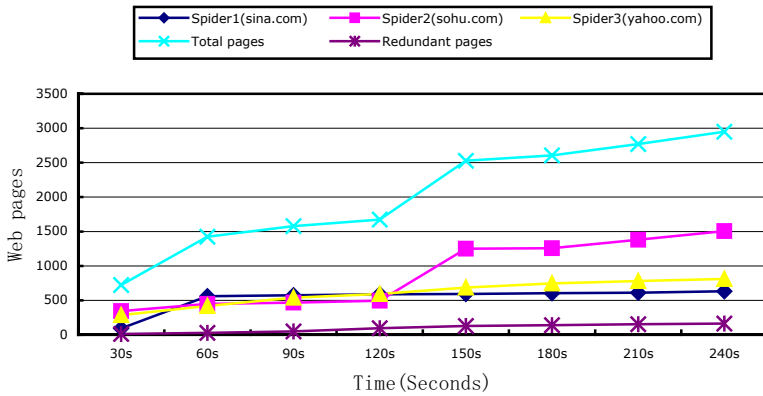


Fig. 5. The figure represents the number of pages when three spiders run independently

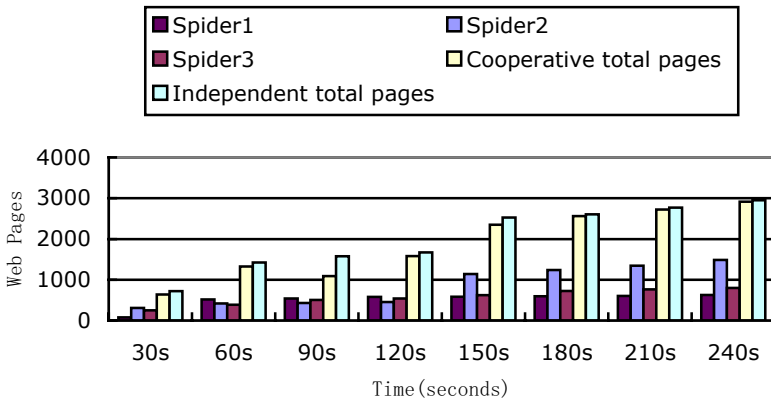


Fig. 6. Comparison of cooperative run and independent run

5 Conclusions and Future Work

Experiment shows that it costs about 4%-6% storing space for redundant web pages caused by parallelization if we do not adopt some effective measures to avoid it. In

this paper, we construct a parallel spider based on multi-agent paradigm. Experiment result shows the spider model is effective in eliminating redundancy and improving the search although it takes some cost. In future work, we plan to integrate post-processing in the parallel spider system. In this case, mature data mining technology can be applied to acquire the useful information hid in the tremendous web pages.

Acknowledgements

The research work in this paper is supported by the National Basic Research Priorities Programme (No.2003CB317004), the 863 Project (2003AA115220) and National Natural Science Foundation of China (No. 60435010).

References

1. J. Michael Chau. Design and evaluation of a multi-agent collaborative Web mining system Decision Support Systems 35 (2003) 167– 183
2. Junghoo Cho, Hector Garcia-Molina. Parallel Crawlers Proc. of the 11th International World--Wide Web Conference
3. Zhongzhi Shi MAGE: An Agent-Oriented Programming Environment Third IEEE International Conference on Cognitive Informatics (ICCI'04) pp. 250-257
4. Zhuge, China's E-Science Knowledge Grid Environment, IEEE Intelligent Systems, 19(1) (2004) 13-17
5. Marian Nodine, Jerry Fowler, Tomasz Ksiezyk, Brad Perry, Malcolm Taylor and Amy Unruh. Active Information Gathering in InfoSleuth In International Journal of Cooperative Information Systems 9:1/2, 2000, pp. 3-28.
6. Dong, Shaohui Liu, Haijun Zhang, Zhongzhi Shi, Parallel Web Spider Based on Intelligent Agent, In: Proceedings of The 5th Pacific Rim International Workshop on Multi-Agents (PRIMA2002), Tokyo, 2002