Towards Distributed Web Mining in Net-Enabled Enterprises

Yihua Sheng
Southern Illinois University Carbondale

Follow this and additional works at: http://aisel.aisnet.org/amcis2004

Recommended Citation
http://aisel.aisnet.org/amcis2004/238

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Towards Distributed Web Mining in Net-Enabled Enterprises

Yihua Philip Sheng
Department of Management and
Department of Computer Science
Southern Illinois University Carbondale
phsan@cs.siu.edu

ABSTRACT
In today’s information age, web sites have become an important source for business information collection and analysis. They provide a company abundant information for competitor analysis and business intelligence. Also, web mining on a firm’s intranet can greatly assist a firm’s endeavor in knowledge management of a firm. However, web mining is a complex and resource-consuming process that consists of crawling a large number of web pages and automatically extracting needed information from retrieved pages. Traditionally, web mining systems use dedicated systems with a client-server architecture. This makes it very expensive for an organization, a community or an individual to have their own web mining system. We propose a light-weight distributed web mining system model that makes use of a firm’s available computers in a peer to peer network. The peer to peer network excludes the use of dedicated systems, and the mobile agents here make use of the systems whose resources are readily available to balance the load. A prototype of the distributed web crawling part of the system is implemented to illustrate the feasibility of the concept.

Keywords
Web Mining, Mobile Agents, Peer to Peer Network, P2P, Distributed Computing

INTRODUCTION
Web mining is the use of data mining techniques to discover and extract information or knowledge from documents on web sites, and can be divided into three categories: web content mining, web structure mining and web usage mining (Etzioni 1996). The most interesting one of these three is probably the web content mining part. The World Wide Web (WWW) has abundant valuable information nowadays, but it is still quite difficult for individuals to find information or knowledge they need. It is partly due to the way the information is organized on the web, and also partly due to lack of good information mining tools.

Today, the majority of web users rely on general-purpose search engines like Google.com to search needed information on the WWW, which often turns out to be a rather tedious process. What makes these general-purpose search engines less effective is that they do not provide users ways to customize a search, other than trying different keyword combination. Companies and individuals might have their specific search needs for special purposes. For example, a company might only want to search its competitors’ websites, not the whole WWW, for information about certain competing products. Searching the whole WWW, instead of a number of websites of the competitors, could bring too much information with equivalent, or often times low, relevancy that 1) buries relevant and valuable information, and therefore 2) causes the user much more time to browse through the whole list of returned information and possibly even to miss the real valuable information. Under such circumstances, highly customizable web mining systems could greatly help an information collector to conduct specialized information search on the web.

Another use of a web content mining system would be to mine for certain information on the corporate intranet. The corporate intranet has become a significant place for a firm to publish internal information. Information such as new features of the products under development, customer services records, meeting memos, internal job postings, etc., is often seen on many companies’ intranet, each of which is getting bigger and bigger. The rapid growth of information on corporate intranets has caused a great deal of difficulty for employees to find needed information. Many firms are buying search technologies from companies like Google.com to build their internal search engines to search their own intranet. However, these search engines, as described above, lack flexibility in performing searches and are usually very costly because they use a set of a dedicated high-end computer and network equipment.
We propose a distributed web mining system which utilizes a firm’s existing information technology infrastructure, such as computers, network, etc. The peer-to-peer (P2P) technology pools unutilized resources to construct a powerful computing platform, and mobile software agent technology makes use of this computing platform and increases the effectiveness, flexibility and speed of searches.

BACKGROUND

Web mining involves many steps, such as web crawling, page ranking, information extraction, information codification, information presentation, etc. Building a search engine - let alone a web mining system, which is more complicated than a web search engine, could be expensive (Lawrence and Giles 1999). Reliable and durable software and hardware, a large number of computers for web crawling and page ranking, high speed network connection and databases with large capacity are the minimum requirements for building a good web mining system. Any of these can stop a company making efforts to build its own web mining system.

However, there are actually many computers in a company’s network that have much idle CPU time, disk storage, and under-utilized network bandwidth. These resources certainly can be gathered to make a distributed web mining system. In this paper, we propose a working model of a distributed web mining system based on mobile agent technology and a P2P computing platform. We have so far implemented the distributed web crawling part of the system, which is also reported in this paper.

Mobile Agents

A mobile agent is a process that can actively migrate from one host to another host and, based on locally computed decisions, can actively migrate to a third host (Rothermel and Hohl 1998). It is a software agent that has the additional property that it is not bound to operate only in the system in which it started. A mobile agent has the unique property that during its lifetime it can be halted, its state and code can be moved to another computer on the same network and then it continues executing from where it stopped. A mobile agent is a persistent agent, i.e. it can outlive the application it originates from, and typically is limited in size. Agents are said to be strongly mobile if their entire code and execution state can be moved when they move from one computer to another. The foundation of distributed systems and applications were laid in 1984 when Birrell and Nelson proposed a mechanism which allows programs to call sub-programs residing on a different machine (Birrel and Nelson 1984).

Use of mobile agents in a system will have a distributed architecture rather than a centralized architecture. It utilizes a pool of computers to fulfill a designated task. Mobile agents are scattered among these computers. The number of participating computers usually increases along with the complexity of the task. This implies that the owner of the system somehow needs to own a large computer pool in order to have enough computing resources to perform a big task, like web crawling and web page ranking, within a reasonable time period. However, this is often not the case in the real world. It is not quite practical for a company to buy a large number of computers and a large amount of network bandwidth just for building a web mining system. A P2P computing network is a possible answer for this issue.

P2P Computing

A P2P computing platform consists of a pool of networked computers that take advantage of underutilized resources, such as storage, processing, etc. in a distributed way (Talia and Trunfio 2003). P2P computing technology can be defined as the sharing of computer resources and services by direct exchange (Moore and Hebeler 2001; Oram 2001). P2P network is a type of network where each computer has equivalent power. In such an environment, servers, desktops and notebook computers in a network become peers that contribute all or part of their resources. This type of architecture transforms client computers from mere consumers of services to service providers as well.

Figure 1 illustrates two P2P models: a pure P2P computing model and a hybrid P2P computing model (Edwards 2001). In a pure P2P computing model, all peers are considered equal. Individual computers in a P2P pool have access to a real-time index of other active computers in the pool. The index itself is scattered across the pool. A computer can directly communicate with any other computer in the pool using the address from the index. In a hybrid P2P computing model, a central server in the pool exists and holds such information as the index directory, computer profiles, user profiles, etc. Napster uses this model to store its centralized index directory (Edwards 2001; Parameswaran, Susala and Whinston 2001).

The SETI@home project has demonstrated convincingly the scalability and feasibility of distributed computing with spare CPU cycles. In spring 2002, there were 500,000 active users; a total of almost 1 million CPU years had been donated (Anderson et al. 2002). Gnutella, KaZaa, and Morpheus have generated ample interests in P2P networks and have shown

Proceedings of the Tenth Americas Conference on Information Systems, New York, New York, August 2004
their utility. Combining mobile agents with P2P computing seems to be an ideal tool for building a web mining system, because (a) they can react flexibly on the availability of resources, and (b) they allow for new computational tasks to be created without the need for installing new software on multiple hosts.

Figure 1 Two P2P Models (Source: IDC)

(a) Pure P2P
(b) Hybrid P2P

P2P computing provides an alternative to the traditional client/server architecture. While employing the existing network, servers, and client infrastructure, P2P offers a computing model that is orthogonal to the client/server model. The two models coexist, intersect, and complement each other. In a client/server model, the client makes requests of the server with which it is networked. The server responds to the requests and acts on them. With P2P computing, each participating computer, referred to as a peer, functions as a client with a layer of server functionality. This allows the peer to act both as a client and as a server within the context of a given application.

A DISTRIBUTED WEB MINING SYSTEM

Figure 2 shows the conceptual layout of the proposed distributed web mining system. Computers in the P2P pool are equal peers. Each computer accepts and runs intelligent agents with different tasks. Some of the agents are mobile, and some are stationary for specific purposes such as human-computer interactions. As with SETI@home, an execution environment for mobile agents, called agency, will be voluntarily installed on each participating computer in the P2P computing pool. Grasshopper1 (Breugst, Busse, Covaci and Magedanz 1998) is a good candidate because of its excellence in providing a platform for developing and executing mobile agents. The function of the agency is to accept, run, and move mobile agents.

Figure 2 A Distributed Web Mining System in a Peer-to-Peer Computing Pool.
(Each computers in the pool accepts and runs intelligent agents with different tasks)

1 http://www.grasshopper.de
The distributed web mining system has three major components: Web Crawling, Content Mining, and Human-Computer Interface, as shown in Figure 3. The agent repository is used to store different kinds of agents used for different purposes, such as different web crawling agents, different page ranking agents using different rules, etc. Agents in this system follow the thinking of open agent architecture, so that any users with agent programming skills can write their own agents to do different things or do the same thing but in different ways or using different algorithms. In the open agent architecture, one of the important issues is how agents, probably programmed by different people, can communicate with each other. It is an issue of knowledge representation and communication of agents. The most popular agent communication languages (ACLs) are the Knowledge Query Manipulation Language (KQML) (Finin and Labrou 1997) and the ACL developed by the Foundation for Intelligent Physical Agents (FIPA) (Labrou, Finin and Peng, 1999; O’Brien and Nicol 1998). For content representation language, the Knowledge Interchange Format (KIF) and FIPA’s Semantic Language (SL) of are the two most commonly used languages. These standards have laid certain foundation, though not enough yet, to establish an open agent architecture. The open architecture allows the system to evolve and grow continuously without any centralized control (José, Moreira, Meneses and Coulson 2001). This could be an extremely beneficial factor of such a publicly available system.

A web mining task starts at the Human-Computer Interface part. A user inputs a mining request through an interface. Then, the request is divided into smaller tasks such as 1) web page crawling to retrieve related web pages, 2) page content mining for extracting information or knowledge, 3) information codification and organization to formulate the information in the way the user wants, and 4) finally the information is presented to the user through the Human-Computer Interface part.

The whole task usually starts with a bunch of web search activities, which involve launching many web crawling agents to search for web pages containing certain keywords derived from the web mining task specified by the user. Once the related web pages are all retrieved from a range of web sites specified by the user, the content mining sub-system takes over to mine for information needed by the user. Because the whole task is a collaborative process among many different intelligent agents, a management mechanism needs to be in place to coordinate the activities among these agents. Manager agents are designed to do so. A Manager Agent analyzes the search requests from users and then decides how many initial web crawling agents need to be created. Because no one knows in advance how deep and broad a website is, the web crawling work is actually a dynamic process. The Manager Agents watch the size of the page crawling work and create more agents as needed.

The content mining sub-system can be fairly complex and it determines, to a large degree, the precision and therefore the usefulness of the system. Information search and filtering is implemented in this part. Also, this part can go one step further to do more complicated content mining based on the information gathered. For example, it could mine for association rules or do classification and prediction based on the retrieved web contents (Han and Kamber 2001), all depending on how the agents in this part and the Human-Computer interface part are designed and implemented. Our project has not gone to this stage yet. This paper only designs and implements the distributed web crawling part of the system to study and demonstrate the feasibility of using mobile agents and a P2P computing network to build a distributed web mining system.
Prototype Design of the Distributed Web Crawling Sub-System

The function of the prototype of the distributed web crawling system is to get keywords and a list of web sites as input, and then search for web pages containing the keywords within the range of domain (web sites) specified in the input. This current web crawling system consists of Java based agents. The agents that we have used in this system are:

- Console Agent (CA)
- Database Agent (DA)
- Manager Agent (MA)
- Resource Agent (RA)
- Scheduling Agent (SA)
- Web Crawling Agent (WCA)

Along with these we have also used a link database with the following tables:

- Inverted Index Table (keyword, links)
- Waiting List Table (keyword, domains, query_count)

**Console Agent (CA)**
A Console Agent is an application present on all the participating computers in the pool. It has a GUI with provisions for entering the keywords and the domains in which the keywords are to be searched. The user can also set priority for the search on the scale of 1 to 5. The CAs interact with the DAs and display the links in which the keywords are present.

**Database Agent (DA)**
A Database Agent is a stationary agent present on the computer where the link database resides, which could be the initiating user’s own computer. A DA takes care of querying the database, maintaining the tables, updating the results returned by WCAs and sending the jobs to SAs.

The DA gets keywords and the domains from CAs. It looks for the presence of the keywords in the Inverted Index Table (IIT) and, if found, returns the links corresponding to the keywords. If the keywords were not found then it would add the keywords and the corresponding domains to the Waiting List Table (WLT). If the keywords have already been entered into the WLT, the DA increases the query_count of the keywords by the priority specified by the CA. When the query_count of any keyword reaches a predefined value, then the DA sends the keyword and the domains to a WCA as a search job. The DA also receives the links for the keywords from MAs and updates the link database.

**Resource Agent (RA)**
In the P2P computing platform, we use the computers in the network whose CPU usage is less than a particular percentage. In order to do this, we implement a Resource Agent. Upon request for a resource, i.e. a computer, from either a SA or a MA, the RA searches in the network for a relatively less used system and gives its location to the corresponding agent.

**Scheduling Agent (SA)**
Because different search jobs have different priorities as specified by users, a SA is designed to implement the priorities. A SA is also a stationary agent residing on the computer of the user who starts the search job. The SA receives jobs from the DA and then applies resources from the RA; it then creates a MA and moves it to the system allocated by the RA.

A SA maintains a tree structure for storing the jobs and a queue to fire it. It works in such a way that the jobs having higher priority are more often carried out than the jobs having lower priority. In order to avoid starvation, the SA sees that all types of jobs are allotted resources. Upon having a job to be done and a resource address at hand, it creates a MA and dispatches it into the system.

**Manager Agent (MA)**
Manager Agents create WCAs to carry out the jobs handed over by the SA. A WCA may some time inform its MA that the load on that computer has crossed the limit it was supposed to handle, in which case the MA should create more WCAs on different computers in the pool. Moreover, the MA would create more sub-MAs to share the load when the number of WCAs it handles crosses the number it could manage. Like the SA, MAs look to the RA for resources. When WCAs finish their jobs and send back the results to their MAs, the MAs pass the results to the DA.

**Web Crawling Agents (WCA)**
A web crawling agent gets a list of URL addresses, visits these web pages and delivers the page content to the page filtering and ranking agents for further analysis. In this simplified prototype system, the page filtering and ranking function, which is greatly simplified as well, is assigned to the web crawling agents too, i.e. instead of delivering the web pages to page
filtering and ranking agents that are absent in the current system, the crawling agents filter the web pages by themselves. It simply looks for the presence of the keyword in the pages. If the agent finds the keyword, it retains the reference to the page. In addition, the agent collects all the outgoing links in the pages and adds them to a list. The agent continues this process until all the links in the list are visited. At the end of the process, it would send the keyword and the URLs it finds to the MA, which will further deliver to the DA for storage.

**Implementation of the Prototype Design**

The Grasshopper agent environment is used for the implementation of the prototype system. The selection of the Grasshopper platform is based on a comparative study of existing tools and environments. Grasshopper is completely implemented in Java and has been designed in conformance with the Object Management Group’s Mobile Agent System Interoperability Facility (MASIF). The platform can furthermore be enhanced with an add-on, which is compliant with the FIPA specification.

So far, we have implemented a prototype version of the distributed web crawling part of the whole system. Through a CA, a user gives the keywords, the domains in which the keywords need to be searched, and the priority of the search. The CA then contacts the DA that then looks for the keywords in the IIT. If the DA finds the keywords, it sends back the links that belong to the specified domain. If the keywords are not present in the IIT, the DA returns null result to the CA. Then, the DA looks for the keywords in the WLT. If the keywords are present, the DA adds the priority specified by the user to the keyword of the keywords. If the keywords are not present in the WLT, the DA places the keywords and the domain in the WLT and initializes the `query_count` with the search’s priority.

The DA also checks if the `query_count` of the tuples in the WLT has crossed the threshold value. If so, then the DA sends the SA the information for searching that keyword in the associated domains. The SA looks for the resources from the RA. The RA sends the SA the IP address of the participating computer whose CPU usage is less than the predefined value. The SA then creates a MA and dispatches it to that computer. The MA gets the links and the keyword from the SA and creates a WCA with the keyword and starting URL address. The WCA recursively crawls the web pages and returns the URLs in which the keyword was found to the MA. The MA passes this information to the DA, which places it in the database. If the number of pages to be crawled by one WCA exceeds a preset limit, the MA will create more WCAs to balance the load. Figure 4 is an illustration of the relationship among different agents.

![Figure 4 Agents involved in a Search](image-url)

Figures 5 and 6 show the screen outputs of a search job. The search job is to find all the pages containing a keyword called “computer”, the starting web site is http://www.cs.siu.edu. Figure 5 shows there are one MA and two WCAs running on two different computers (NT7 and NT8). NT7 is also the computer running the “region” of Grasshopper that holds all the information about mobiles agents. The agency for running mobile agents was installed on NT7 and NT8, and was registered...
in the “region” as well. Figure 6 shows the results returned from these web crawling agents. All the links contain the keyword “computer.”

![Figure 5 Region Information](image1)

![Figure 6 Results Returned from Web Crawling Agents with a Simple Filtering Function](image2)

**BENEFITS OF THE PROPOSED SYSTEM**

Building a distributed web content mining system on a P2P computing platform using mobile agents can obtain great benefits, as compared to building such a system on a traditional client-server architecture.
Cost Saving. A web content mining system is quite costly. It is quite costly for an average company to buy a set of dedicated equipment to build such a system. Using a P2P architecture akin to SETI@home and Napster allows a company to exploit its internal under-utilized computing resources to build such a powerful tool.

Highly Customizable. The customizability of this system is manifested in two ways. 1) The system itself could offer many different ways and different algorithms for doing web mining, such as only conducting web mining on the websites a user specifies. 2) the system is expandable through programming new agents, or assembling available agents in different ways to do things differently.

Balanced Network Bandwidth. In the traditional client-server architecture, all the data retrieved from the Internet needs to be transferred back to the server for processing, such as information filtering or ranking, or data mining, so the network bandwidth for the server is very high. In this distributed system, only the final results are sent back to the user originating the task, all the intermediate data can be distributed in the pool through a dynamically distributed database architecture. The dynamic feature of the database architecture is needed because, in this public pool, a computer holding part of the intermediate data might become unavailable at any time for any reason. When it happens, the loss data needs to be reconstructed.

Conservation of Network Bandwidth. Due to the mobile nature of the agents in this system, when it comes to analyzing or mining a large amount of data, the code, i.e. a mobile agent, is sent to the computer holding the data, not the other way around. Because the size of code is tiny when compared to the size of the data, i.e. retrieved web pages, sending code to data, instead of sending data to code, can greatly conserve valuable bandwidth.

High Reliability. The whole system is built on a big pool of computers company-wide. A single failure of any computer would not affect the operation of this system. When a computer in the pool become unavailable, its monitoring peer(s) will know and the task will be moved to a new computer to continue.

High Scalability. The company-wide P2P computing pool makes this system highly scalable to different web mining tasks. Any number of the computers, up to the size of the pool, could participate in any given task.

Low Maintenance Effort. A traditional client-server system is closely maintained, especially the server part, because it is heart of the whole system. However, this distributed system is not owned by any user. Participating users do not need make any effort to maintain the whole system. If some parts of the system stop working, the mobile agents will move to the other parts that work.

Stealth. BargainFinder reported that a third of the online CD merchants it accessed blocked all of its price requests (Maes et al. 1999). This kind of blockade could cause a serious problem when the web crawling job is targeted at a specific company’s website for the purpose of in-depth analysis of this company. If the web crawling activities are initiated from a fixed location, the target companies can easily cut the network traffic from this location through IP address blocking. However under this system, because the web crawling agents are distributed all across the Intranet, which can be geographically dispersed, such as worldwide, in a large corporation, it is difficult for a website to block web crawling activities from this system. Mobile agents can always move to different computers if they find the computers in which they reside are blocked by the target website.

CONCLUSIONS AND FUTURE WORK

In this paper, we have propose a distributed web content mining system using mobile agents in a P2P computing pool. The prototype implementation of the distributed web crawling sub-system indicates that the mobile agent technology works very well in the P2P computing pool. Building a web mining system in the traditional client/serve architecture is expensive, but a distributed web mining system based on publicly available computers is not. A low-cost web content mining system could greatly help a company in its efforts of doing competitor analysis and internal knowledge management.

So far, what we have designed and implemented is a prototype system of the distributed web crawling part. A lot of work needs to be done to make a full fledged system. The content mining sub-system needs to be designed and implemented. This part serves as a key part of the system. The human-computer interaction part needs to be designed with this part too because how web mining is conducted is partly decided by what kinds of questions or conditions or requirements a user can submit.
ACKNOWLEDGMENTS

I would like to thank Prof. Zhong Chen, Prof. Shahram Rahimi and Mr. Srinivas Mandalapu for their help and contribution in this research

REFERENCES