Dispatching mobile Agents for DDM applications

LI Xi-ning[1,2], Guillaume Autran[1]
(1. Department of Computing and Information Science, University of Guelph, Guelph, Ontario, N1G 2W1, Canada; 2. State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing 210093, China)

Abstract: Techniques for mining information from distributed data sources accessible over the Internet are a growing area of research. The mobile Agent paradigm opens a new door for distributed data mining and knowledge discovery applications. In this paper we present the design of a mobile agent system which couples service discovery, using a logical language based application programming interface, and database access. Combining mobility with database access provides a means to create more efficient data mining applications. The processing of data is moved to network wide data locations instead of the traditional approach of bringing huge amount of data to the processing location. Our proposal aims at implementing system tools that will enable intelligent mobile Agents to roam the Internet searching for distributed data services. Agents access the data, discover patterns, extract useful information from facts recorded in the databases, then communicate local results back to the user. The user then generates a global data model through the aggregation of results provided by all Agents. This overcomes barriers posed by network congestion, poor security, and unreliability.

Keywords: mobile Agent; distributed data mining; service discovery; database service
CLC nember: TN253 **Document code:** A **Article ID:** 1673-4785(2008)02-0181-07

Mobile Agent systems bring forward the creative idea of dispatching user defined computations-agents, towards network resources, and provide a whole new architecture for designing distributed systems. Distributed data mining (DDM) is one of the important application areas of deploying intelligent mobile agent paradigm[1-2]. Most existing DDM projects focus on approaches to apply various machine leaning algorithms to compute descriptive models of the physically distributed data sources. Although these approaches provide numerous algorithms, ranging from statistical model to symbolic/logic models, they typically consider homogeneous data sites and require the support of distributed databases. As the number and size of databases and data warehouses grow at phenomenal rates, one of the main challenges in DDM is the design and implementation of system infrastructure that scales up to large, dynamic and remote data sources. The objective of our research is to equip mobile agents with system tools such that those agents can search for data sites, move from hosts to hosts, gather information and access databases carry out complex data mining algorithms, and generate global data model or pattern through the aggregation of the local results.

To deploy mobile agents in DDM, a mobile agent system must provide languages and various programming interfaces for fast and easy development of applications. Different languages, such as C and Java, have been chosen as agent-programming languages for variety of reasons. Among them, logic-programming languages prove to be an alternative tool of building mobile agents. Benefiting from their powerful deductive abilities, complex calculations can often be represented by a set of compact logic predicates, which make agents more suitable to migrate around the network. In addition, mobile agents must interact with their hosts in order to use data services or to negotiate services with service providers. Discovering serv-

收稿日期:2007-07-04.
通讯作者:LI Xi-ning. E-mail: Xli@cis.uoguelph.ca.

ices for mobile agents came form two considerations. First, the agents possess local knowledge of the network and have a limited functionality, since only agents of limited size and complexity can efficiently migrate in a network and have little overhead. Hence specific services are required which aim at deploying mobile agents efficiently in system and network management. Secondly, mobile agents are subject to strong security restrictions, which are enforced by the system security mechanism. Thus, mobile agents should find services that help to complete security-critical tasks, other than execute code, which might jeopardize remote servers. Following this trend, it becomes increasingly important to give agents the ability of finding and making use of data services that are available in a network[3]. A variety of Service Discovery Protocols (SDPs) are currently under development by some companies and research groups. The most well known schemes are Sun's Java based JiniTM[4], Salutation[5], Microsoft's UPP[6], IETF's draft Service Location Protocol[7] and OASIS UDDI[8]. Some of these SDPs are extended and applied by several mobile agent systems to solve the service discovery problem.

In a DDM environment, data may be stored among physically distributed sites and may be artificially partitioned among different sites for better scalability. Therefore endowing mobile agents with the ability of accessing remote databases is the basis for DDM applications. This encourages us to explore the strategies of coupling a mobile agent programming language with database access facilities. In recent years numerous approaches have been made under the topic of designing a coupled system that integrates a relational database and a logic programming language. They all enable programmers to access large amounts of shared data for knowledge processing, manage data efficiently as well as process data intelligently. Generally speaking, coupling a logic programming language with database access facility can be roughly divided into two categories. A loosely coupled system embeds a non-logic language, such as the structure query language (SQL), within the logic-programming context and a closely coupled system provides database query interface as a subset or an extension of the language for featuring dynamic query formulation and view access. For example, SWI-Prolog and KB-Prolog[9] belong to the first and CGW[10] and Quintus[11] fall into the second category.

In this paper we present the design and implementation of a mobile agent system, which couples a logic language based application programming interface for DDM. Two important system modules, namely, service discovery and database access, have been encapsulated and installed inside of the existing IMAGO (intelligent mobile Agent gliding on-line) system. The IMAGO system is an infrastructure for mobile agent applications. It includes code for the IMAGO server-a Multi-threading Logic Virtual Machine, the IMAGO-Prolog-a Prolog-like programming language extended with a rich API for implementing mobile agent and DDM applications, and the IMAGO IDE, a Java-GUI based program from which users can do editing, compiling, and invoking an agent application. The remainder of the paper is organized as follows. Section 2 gives an overview of the design of service discovery module and integration with the IMAGO system. In Section 3, we discuss definitions of the database predicates and the principle of data driven implementation. In Section 4, we present a simple database access example and briefly explain how to use database interface presented by the IMAGO system. Finally, we give the concluding remarks as well as future work.

1 Data service discovery

The general idea of distributed data services is that a DDM application may be separated from the data sites needed to fulfill a task. These data sites are mostly modeled by local and centralized databases, which are independent of, or sometimes unknown to the application. A commonly used DDM approach is to apply traditional data mining algorithms to an aggregation of data retrieved from physically distributed data sources. However, this approach may be impractical to a large scale of data

sets distributed over the Internet. Deploying mobile agent paradigm in DDM offers a possible solution because the application may decompose data mining problems to scale up to a large distributed data sources and dispatch agents to carry out distributed data processing. This in turn leads us to the data service discovery problem, that is, how to find data sites available to a DDM application.

Clearly, the number of services that will become available in the Internet is expected to grow enormously. Examples are information access via the Internet, multi-media on demand, Web services and services that use computational infrastructure, such as P2P and Grid computing. In general, the service usage model is role-based. An entity providing services that can be utilized by other requesting entities acts as a provider. Conversely, an entity requesting the provision of a service is called a requester. To be able to offer services, a provider in turn can act as a requester making use of other services. In a distributed system, requesters and providers usually live on physically separate hosts. Providers should from time to time advertise services by broadcasting to requesters or registering their services on third party servers.

In the IMAGO system, we have implemented a new data service discovery model DSSEM (Discovery Service via Search Engine Model) for mobile agents[13]. DSSEM is based on a search engine, a global Web search tool with centralized index and fuzzy retrieval. This model especially aims at solving the database service location problem and is integrated with the IMAGO system. Data service providers manually register their services in a service discovery server. A mobile agent locates a specific service by migrating to the service discovery server and subsequently submitting requests with the required data description. The design goal of DSSEM is to provide a flexible and efficient service discovery protocol in a mobile agent based DDM environment.

Before a service can be discovered, it should make itself public. This process is called service advertisement. The work can be done when services are initialized, or every time they change their states via broadcasting to anyone who is listening. A service advertisement should consist of the service identifier, plus a simple string describing what the service is, or a set of strings for specifications and attributes.

The most significant feature of DSSEM is that we enrich the service description by using web page's URL to replace the traditional string-set service description in mobile agent systems. That is, service providers use web pages to advertise their services. Because of the specific characteristics, such as containing rich media information (text, sound, image, etc.), working with the standard HTTP protocol and being able to reference each other, web pages may play a key role acting as the template of the service description. On the other hand, since the search engine is a mature technology and offers an automated indexing tool that can provide a highly efficient ranking mechanism for the collected information, it is useful for acting as the directory server in our model. Of course, DSSEM also benefits from previous service discovery research in selected areas but is endowed with a new concept by combining some special features of mobile agents as well as integrating service discovery tool with agent servers.

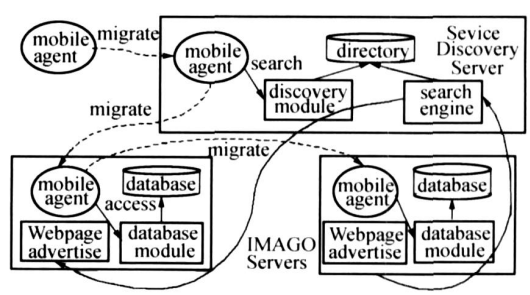

Fig. 1　An example of service discovery and data mining process

In principle, data service providers register the URLs of their websites that advertise all the information concerning services. As a middleware on the service discovery server, the search engine will periodically retrieve web pages indicated in URLs and all their referencing documents, parse tags and words in documents and set up the relationship between the keywords and the host address of these service providers. On the other

hand, a mobile agent can move to a service discovery server, utilize the system interface to access the search engine's database and obtain an itinerary that includes a list of ranked host addresses of the data service providers. Based on the given itinerary, the mobile agent may travel from host to host to carry out a DDM application. Figure 1 gives an example of service discovery and data mining process.

The application programming interface of data service discovery for mobile agents is a built-in predicate, namely, web_search(Query, Number, Result), where Query is a compound term specifying characteristics of the search, Number is an integer indicating what is the maximum number of results expected, and Result is a variable to hold return values. For example, suppose a food company wants to analyze the customer transaction records for quickly developing successful business strategies, its DDM agent may move to a known I-MAGO service discovery server and then issue a query predicate requesting up to 10 possible food industry database locations:

web_search(locate("food", "customer transaction", "imago data server"), 10, R)

The agent is blocked and control is transferred to the service discovery module of the hosting I-MAGO server. The discovery module will communicate with the searcher, wait for search results, and resume the execution of the blocked agent. Search results will be delivered to the requesting agent in the form of list, where entries of the list are ranked in priorities from high to low.

2 Data mining facilities

Obviously, representing content of a database in logic terms may be achieved by using different methods. As a Prolog program itself is somehow thought of as a database, one natural approach is fact-driven, namely, to append the content of a table as a set of fact clauses to the end of the program. The advantages of such approach are simplicity and ease of use, but the disadvantages are dynamical modification of the original program and creation of garbage code because part of the appended facts might be useless. In IMAGO system, we intend to provide two data representation schemes, i. e., fact-driven and data-driven, which would broaden the range of choices accessible by application programmer. Data-driven approach converts content of a table to a list of rows where each row is a sub-list of fields. In other words, the results obtained from a database query will be automatically transformed to the IMAGO Prolog internal data structures.

Having received a list of database addresses through the service discovery module, an agent may move from host to host to access these databases or clone multiple agents with assigned database addresses to start a DDM application. In order to bridge logic based mobile agents with database systems, the IMAGO system provides a set of database predicates, which enables agents to establish connection with data sources and make requests for desired information.

To be able to access a database, an agent has to find which database drivers are supported by the underlying system. For this purpose, built-in predicate db_drivers(?list) can be invoked to collect a list of all supported database drivers. A driver identifier must be unique and is commonly named based on the database implementation, such as mysql, oracle, and so on.

A database connection is established by issuing a predicate of the form db_connect(-conn, +db_URL). A valid db_URL looks like:

[driver://][username[:password]@]hostname[:port][/database]

where driver indicates the specific database implementation, username and password are used for authentication, hostname gives the network location of the database server, port is used to establish a TCP connection if required, and database specifies the actual database name which the agent wants to access. Upon success, variable conn is instantiated with the database connection handler which will be used in the subsequent database accesses. To disconnect a database, the agent may issue a predicate db_disconnection(+conn), where conn represents a previously established connec-

tion. Once a database has been disconnected, all resources allocated for this connection are released and the connection handler is no longer valid.

To facilitate DDM applications, the IMAGO system provides two different ways of database access operations, namely, the set retrieval and the tuple retrieval. The former returns the entire matching data set to the requesting agent, whereas the latter allows the requesting agent to consume the matching data on the tuple by tuple basis. For example, to issue a database query, a simple predicate db_query(+conn, +sql) can be invoked, where argument sql represents a standard SQL statement. If the query succeeds, user may either issue a subsequent call to db_store_result(+conn, ?rest, +options) to save the entire result set into the agent's memory as a two dimensional list, or to db_use_data(+conn, -rest) to initiate the result set and then issue a sequence of calls to db_next_row(+rest, ?row, +options) for row-wise data processing, and finally invoke a db_free_result(+rest) when all rows have been exhausted.

In a DDM application, agents are not working alone and they need to communicate with each other to cooperate and generate a global data aggregation for further analysis. Most existing mobile agent systems adopt some kind of communication models/protocols from traditional distributed systems. However, the IMAGO system adopts a different strategy to cope with this issue. The idea is to deploy intelligent mobile messengers for inter-agent communication[13]. Messengers are thin agents dedicated to deliver messages. Like normal agents, a messenger can move, clone, and make decisions. Unlike normal agents, a messenger is anonymous and its special task is to track down the receiving agent and reliably deliver messages in a dynamic, changing environment. The IMAGO system provides a set of built-in messengers as a part of its programming interface. A data-mining agent at any remote sites and at any time may dispatch messengers to deliver data to designated receivers. For example, suppose that a mobile agent has completed its data mining work at a remote database server, it can either call move('home') to carry results and migrate back to its stationary server, or invoke dispatch($oneway_messenger, 'home', Results) to create a messenger which is responsible to deliver Results to the home server.

Communication among agents takes place by means of an Agent Communication Language (ACL). The essence of an ACL is to make data mining agents understanding the purpose and meaning of their exchanged data. In general, a message consists of two aspects, namely, performative and content. The performative specifies the purpose of a message and the content gives a concrete description for achieving the purpose. In order to facilitate open standards of ACL's, the IMAGO agent-based communication model is in compliance with the FIPA ACL message structure specification[14]. Of course, the performative and content of a message often determine the reaction of the receiver. In addition to the various types of system built-in messengers for sending agents, the IMAGO system provides a set of predicates for receiving agents. The predicate which is similar to an unblocking receive is accept(Sender, Msg). An invocation to this procedure succeeds if a matching messenger is found, or fails if either the caller's messenger queue is empty or there is no matching messenger in the queue. Likewise, the predicate which implements the concept of blocking receive is wait_accept(Sender, Msg). A call to this procedure succeeds immediately if a matching messenger is found. However, it will cause the caller to be blocked if either the caller's messenger queue is empty, or no matching messenger can be found. In this case, it will be automatically re-executed when a new messenger attaches to the caller's messenger queue. Pragmatically, the semantics of matching messengers is implemented by logic unification.

3 A simple example of DDM application

In order to illustrate the usage of the IMAGO database facilities, we show a very simple example in this section. Although not very practical, this example provides a typical sample of how to deploy mobile agents in DDM applications. As a reference

framework, more complex DDM algorithms can be easily coded.

```
:-home_agent(a_DDM_agent).
 a_DDM_agent(_) :-
 create ("worker1", worker1, location("perseus.imago
      _lab. cis. uoguelph. ca", "mysql://localhost/
      ImagoDB1")),
 create ("worker2", worker2, location("orion. imago_
      lab. cis. uoguelph. ca","mysql://localhost/ Ima-
      goDB2")),
 wait_accept(worker1, Table1),
 wait_accept(worker2, Table2),
 result_compare(Table1, Table2),
 terminate.
result_compare(T1, T2) :- write("Two tables are i-
dentical."), nl.
result_compare(_, _) :- write("Two tables are not i-
dentical."), nl.
:-end_home_agent(a_DDM_agent).

:-mobile_agent(worker1).
  worker1(location(Database_Server, DB_URL)) :-
     move(Database_Server),
     db_connect(C, DB_URL),
     db_query(C,"SELECT * FROM LOG"),
     db_store_result(C, Table, [name(true), type
(true), length(true)]),
     db_disconnect(C),
     dispatch($oneway_messenger, home, Table).,
     dispose.
:-end_mobile_agent(worker1).

:-mobile_agent(worker2).
  worker2(location(Database_Server, DB_URL)) :-
     move(Database_Server),
     db_connect(C, DB_URL),
     db_query(C,"SELECT * FROM LOG"),
     db_use_result(C, R),
     get_rows(R, Table),
     db_disconnect(C,),
     dispatch($oneway_messenger, home, Table),
     dispose.
     get_rows(R, [Row|T]) :-
     db_next_row(R, Row, [name(true), type(true),
length(true)]),
     get_rows(R, T).
     get_rows(R, []) :-
     db_free_result(R).
:-end_mobile_agent(worker2).
```

The code of a_DDM_agent defines the home agent resided at the stationary server. When the home agent starts execution, it creates two mobile agents called worker1 and worker2 alone with initial arguments and then waits for results. When a worker is loaded, it migrates to the IMAGO database server specified in its argument. Having arrived to the database server, the worker continues execution by connecting to the given local database and making a SQL query for data access. Two worker agents take different way to retrieve data, where worker1 uses set retrieval and worker2 adopts tuple retrieval. Having collected all required data, both workers separately dispatch a $ oneway_messenger to deliver results back to the home agent for further data mining analysis.

4 Conclusion

In this paper, we discussed the scheme of deploying mobile agents in DDM applications. The advantage of adopting mobile agents for DDM is to scale up to large, dynamic and remote data sources, such as various databases distributed over the Internet. We presented the design of data service discovery module and database management module. The programming interface of these modules is a set system built-in predicates capable to couple a logic programming language with functionalities of locating data services and accessing remote databases. Equipped with those system tools, mobile agents may search for suitable data sites, roam the Internet to collect useful information, and communicate with each other to generate a global view of data through the aggregation of distributed computations. In order to verify the feasibility and efficiency of the mobile agent based DDM proposal, experimental service discovery module and database management module have been implemented and integrated with the IMAGO system. The service discovery module is based on the search engine technology and concentrates on locating database services. It uses web pages as a medium to advertise services, and runs an independent search engine to gather and index service provider's information, such as service types, da-

tabase names, URLs, access modes, as well as possible verification information. The database management module not only provides flexible interface for accessing data, but also manipulates database connections efficiently. At the current stage, the database model in the IMAGO system is MySQL, the most popular open source DBMS system in the world[15].

Research on the agent based DDM involves further extensions of the IMAGO system. First, the current implementation of service discovery module deals with only a limited number of logical relationships. To be able to offer more precise discovery service, this module could be enhanced to parse some complex search criteria, such as conditional expressions and sub-string matching. Secondly, since databases may contain multi-dimensional data, retrieving such kind of information from flat web pages is a pending problem. We are looking to use XML meta-data to solve the database dimensional problem. In addition, we are making investigations on adding more programming languages to the IMAGO system, as well as introducing more flexible and efficient communication tools, such as mobile socket, to facilitate DDM applications.

Acknowledgments. We would like to express our appreciation to the Natural Science and Engineering Council of Canada and State Key Laboratory of Novel Software Technology (Nanjing University) for supporting this research.

References:

[1] KLUSCH M, LODI S, MORO G. The role of Agents in distributed data mining: issues and benefits[C]// IEEE WIC International Conference on Intelligent Agent Technology. Halifax, Canada, 2003: 211-217.

[2] PARK B, KARGUPTA H. Distributed data mining: algorithms, systems, and applications[M]. [S. l.]: Lawrence Erlbaum Associates, 2003: 341-358.

[3] BETTSTETTER C, RENNER C. A comparison of service discovery protocols and implementation of the service location protocol[C]// Proc of EUNICE 2000. Twente, Netherlands, 2000: 13-15.

[4] HASHMAN S, KNUDSEN S. The application of jini technology to enhance the delivery of mobile services. [2001-12-01]. http://www.sun.com.

[5] Salutation consortium, salutation architecture overview [EB/OL]. [1999-06-23]. http:// www.salutation.org/whitepaper.

[6] Universal plug and play forum, UPnP? device architecture, version 1.0.1 [EB/OL]. [2006-9-11]. http://www.upnp-ic.org/resources/UPnP_device_architecture_docs/.

[7] GUTTMAN E, PERKINS C, VEIZADES J, et al. Service Location Protocol, Version 2 [EB/OL]. [1999-07-21]. http://www.ietf.org/rfc/rfc2608.txt.

[8] OASIS [EB/OL]. [2005-10-27]. http://www.uddi.org.

[9] BOCCA J, DAHMENM M, FREESTON M, MACARTNEY G, PEARSON P J. A Prolog for very large knowledge bases [C]// Proceedings of the Seventh British National Conference on Databases. Heriot-Watt University, England, 1990: 163-184.

[10] CERI S, GOTTLOB G, WIEDERHOLD G. Efficient database access from prolog[J]. IEEE Transactions on Software Engineering, 1989, 15(2): 153-164.

[11] Quintus Inc. ProDBI, ODBC Interface for Quintus Prolog Database, V. 4.0 [EB/OL]. [1997-06-18]. http://www.sics.se/.

[12] SONG L, LI X, NI J. A database service discovery model for mobile Agents[J]. International Journal of Intelligent Information Technologies, 2006, 2(2) 16-29.

[13] LI X, AUTRAN G. Inter-agent communication in I-MAGO prolog[J]. Lecture Notes in Artificial Intelligence, 2005: 3346, 163-180.

[14] FIPA ACL, Agent Communication Language Specifications, FIPA, 2005: http://www.fipa.org,.

[15] MySQL AB: MySQL 5.0 Reference Manual, MySQL Documentation Library [EB/OL]. [2008-01-03]. http://dev.mysql.com/doc/mysql/en/.

作者简介:

LI Xi-ning was born in 1952. He is a professor of the Department of Computing and Information Science at the University of Guelph and the director of the IMAGO Lab. His research interests include mobile agent system, logic programming, and virtual machine implementation. He received a PhD in computer science from the University of Calgary in 1989. He has published 1 book, 2 book chapters, and 70 journal and conference papers.