Distributed data mining on Agent Grid: Issues, platform and development toolkit

Jiewen Luo\textsuperscript{a,b,*}, Maoguang Wang\textsuperscript{c}, Jun Hu\textsuperscript{d}, Zhongzhi Shi\textsuperscript{a}

\textsuperscript{a} Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
\textsuperscript{b} Graduate University of Chinese Academy of Sciences, Beijing 100049, China
\textsuperscript{c} China University of Mining and Technology, School of Computer Science, Xuzhou 221008, China
\textsuperscript{d} NanChang University, Information Engineering School, 330047, China

Available online 5 July 2006

Abstract

Centralized data mining techniques are widely used today for the analysis of large corporate and scientific data stored in databases. However, industry, science, and commerce fields often need to analyze very large datasets maintained over geographically distributed sites by using the computational power of distributed systems. The Grid can play a significant role in providing an effective computational infrastructure support for this kind of data mining. Similarly, the advent of multi-agent systems has brought us a new paradigm for the development of complex distributed applications. During the past decades, there have been several models and systems proposed to apply agent technology building distributed data mining (DDM). Through a combination of these two techniques, we investigated the critical issues to build DDM on Grid infrastructure and design an Agent Grid Intelligent Platform as a testbed. We also implement an integrated toolkit VASTudio for quickly developing agent-based DDM applications and compare its function with other systems.

1. Introduction

In the past, research in data mining has mainly focused on largely homogeneous and localized computing environments. Clearly, these assumptions are no longer met in modern scientific and industrial complex problem-solving environments, which are increasingly relying on the sharing of geographically dispersed computing resources and an ever-increasing amount of digital data, information, and knowledge generated by the underlying activities and processes [3–6]. Future generation computer systems, based on emerging grid and agent technology, promise to provide reliable and secure computing infrastructures facilitating the seamless use of distributed data, knowledge, tools and systems to solve complex problems in science, industry, healthcare, governments and elsewhere. However, the development of such systems also poses new problems and challenges, both in terms of infrastructure and in the development environment. Can we build a platform to expediently connect the heterogeneous data resource and provide a uniform service interface for DDM? Can we create a user-friendly toolkit to aid programmers quickly developing and deploying DDM applications on the platform? These questions are what we are concerned with in this paper. We propose a novel solution from the view of combining the agent and Grid techniques.

As Ian Foster says, the Grid and agent communities both develop concepts and mechanisms for open distributed systems, albeit from different perspectives [1]. The Grid community has historically focused on “brawn”: infrastructure, tools, and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations. In contrast, the agent community has focused on “brain”: autonomous problem solvers that can act flexibly in uncertain and dynamic environments [2]. While the Grid infrastructure has focused on such things as the means for discovering and monitoring dynamic services, managing faults and failures, creating and managing service level agreements, creating and enforcing dynamic policy. To date, only limited progress has...
Fig. 1. The Agent Grid.

been made on creating the higher level reactive behaviors that would enable truly dynamic formation of service composition (e.g., composition of data mining agents). Hence, it is necessary to build an intelligent platform that enables the independent operating entities (agents) to interact with one another with partial knowledge and emerge with a robust desirable behavior to form dynamic services on the Grid.

Specifically, we try to apply the Grid infrastructure to integrate the distributed data resource on different sites for DDM, as Fig. 1 shows. It connects heterogeneous resources and provides a uniform service interface for data mining agents to access and process data. On top of the infrastructure, the middleware provides a run-time environment for agents where multi-agents could communicate and interact to exchange the partial knowledge in the DDM process, which is significant to improve the accuracy of global results. We call this hierarchical system Agent Grid, which is a parallel and distributed software that integrates flexible agent services (e.g., data mining service), agents’ run-time environment and development tool-set on the grid infrastructure. In the Agent Grid, data mining tools are integrated with generic and data grid mechanisms and services. Thus the Agent Grid can be exploited to perform data mining on very large data sets available over grids, to make scientific discoveries, improve industrial processes and organization models, and uncover valuable business information.

2. State of the art

While data mining has its roots in the traditional fields of machine learning and statistics, the sheer volume of data today poses the most serious problem. For example, many companies already have data warehouses in the terabyte range (e.g., FedEx, UPS). Similarly, scientific data is reaching gigantic proportions (e.g., NASA space missions, Human Genome Project) [7]. Traditional methods typically made the assumption that the data is memory resident. This assumption is no longer tenable. Implementation of data mining ideas in high-performance distributed computing environments is becoming crucial for ensuring system scalability and interactivity as data continues to grow inexorably in size and complexity.

In distributed data mining [3–5], one of the most widely used approaches in business applications is to apply traditional data mining techniques to data which have been retrieved from different sources and stored in a central data warehouse, i.e., a collection of integrated data from distributed data sources in a single repository. However, despite its commercial success, such a solution may be impractical or even impossible for some business settings in distributed environments because data may be inherently distributed and cannot be localized on any one host for a variety of reasons including security and fault tolerance etc.

In order to solve the problems above, during the past decades, there have been several models and approaches proposed to use multi-agent paradigm building distributed data mining applications, such as JAM [9], PADMA [8], BODHI [11] and Papyrus [10]. Common to all approaches is that they aim at integrating the knowledge, which is discovered out of data at different geographically distributed network sites, while every system focuses on special and representative techniques for combing the local models.

3. Issues

However, none of those systems above makes use of the grid infrastructure for the implementation of the agent service, data access and communication since it is a more complicated process. Before building an Agent Grid platform for DDM, one should seriously consider the following issues since they play a significant role in the system’s feasibility, reliability and extensibility.

3.1. Platform architecture

Is the Agent Grid platform a kind of super agent or agent system itself? Do grids and agent systems both provide services within platform [19]? Are agents fully autonomous including being independent of the grid? How are agents and grid services related? For instance, do agents implement the long list of services that the grid provides or is that the underlying component software? Does each agent contain a planner or is a planning service global to a collection of agents?

3.2. Service architecture

How can we define an integrated service architecture providing a robust foundation for autonomous behaviors? This architecture should define baseline interfaces and behaviors supporting dynamic services, and a suite of higher-level interfaces and services codifying important negotiation, monitoring, and management patterns. The definition of an appropriate set of such architectural elements is an important research goal in its own right, and, in addition, can facilitate the creation, reuse, and composition of components.

3.3. Service composition

The realization of a specific or complex task may require the dynamic composition of multiple services. Web service technologies define conventions for describing service
interfaces and workflows, and WSRF\(^1\) provides mechanisms for inspecting service state and organizing service collections. Yet we need far more powerful techniques for describing, discovering, composing, monitoring, managing, and adapting such service collections [2]. This work can be realized by agent services, which have more flexible control and adaptable capability.

3.4. Data management

Grid technologies make it feasible to access large numbers of resources securely, reliably, and uniformly. However, the coordinated management of these resources requires new abstractions, mechanisms, and standards for the management of the ensemble despite multiple, perhaps competing, objectives from different parties, and complex failure scenarios. This requirement will motivate the development of robust and secure agent services for data access and processing. And we also require advances in the summarization and explanation (e.g., visualization) of large-scale distributed systems.

3.5. Development toolkit

Although there are several agent development tools for DDM to date, almost all of them focus on specific data mining techniques and applications. Can we develop a more generic toolkit, which has a powerful algorithm library and a user-friendly GUI environment? How to make the toolkit more extensible for users to add their own data mining algorithms? Can we build a toolkit to improve the accuracy of global results? These questions are our interests, also our goal for a novel toolkit for distributed data mining.

4. Agent Grid platform

In order to investigate the above issues, we built a platform prototype AGriP,\(^2\) which is compliant to Agent Grid architecture and provides a testbed for DDM on Grid.

4.1. Platform architecture

We propose a four-layer model for AGriP platform from the implementation point of view, as illustrated in Fig. 2:

- **Common resources**: consisting of various resources distributed in Grid environment, such as workstation, personal computer, computer cluster, storage equipment, databases or datasets, or others, which run on Unix, NT and other operating systems.
- **Agent environment**: it is the kernel of Grid computing which is responsible for resources location and allocation, authentication, unified information access, communication, task assignment, agent library and others.
- **Developing toolkit**: providing development environment, containing agent creation, information retrieval, distributed data mining, to let users effectively use grid resources.
- **Application service**: organizing certain agents automatically for specific application purposes, such as e-science, e-business, decision support and bio-information.

4.2. MAGE

MAGE\(^3\) is located at the second layer, which is a multi-agent environment with a collection of tools supporting the entire process of agent-oriented software engineering and programming. It is designed to facilitate the rapid design and development of new multi-agent applications by abstracting into a toolkit the common principles and components underlying many multi-agent systems. The idea was to create a relatively general and customizable toolkit that could be used by software users with only basic competence in agent technology to analyze, design, implement and deploy multi-agent systems [20]. Fig. 3 illustrates the architecture of MAGE.

It mainly consists of four subsystems: Agent Management System, Directory Facilitator, Agent, and Message Transport System.

---

\(^1\) http://www.globus.org/wsrf/.

\(^2\) http://www.intsci.ac.cn/intro/agrip.html.

\(^3\) http://www.intsci.ac.cn/en/research/mage.html.
Agent management system is a mandatory component of MAGE. It maintains a directory of AIDs (Agent Identifiers), which contain transport addresses for agents registered in MAGE and offer white pages services to other agents.

Directory facilitator (DF) is an indispensable component of MAGE. It provides yellow page services to other agents. Yellow page services allow agents to publish one or more services they provide so that other agents can find and successfully exploit them. Agents may register their services with the DF or query the DF to find out which services are offered by other agents.

Message transport service (MTS) is the default communication approach between agents on different FIPA-Compliant agent platforms. It uses FIPA ACL\(^4\) as the standard communication language.

Agent is the fundamental actor in MAGE, which combines one or more service capabilities into a unified and integrated execution model that may include access to external software, human users and communications facilities.

Software Fig. 3 is not a internal part of MAGE. It represents all non-agent, external components accessible to an agent. For example, agents may add new services or acquire new communication/negotiation protocols, etc.

4.3. Agent interface services

From the platform architecture discussed in Section 4.1, we can see that how to provide an Agent Environment in the second layer is the key to implementing the DDM on the Agent Grid platform because of the following reasons:

Agent Environment integrates the components of Common Resources and makes these resources thus available for data mining agents to access and process.

Agent Environment provides different kinds of Agent Grid common services or composition for the upper layer to complete complex DDM tasks.

Hence, we need a powerful agent environment to provider interfaces for the hierarchical architecture of the AGriP platform. As referred to in Section 4.2, MAGE is a distributed agent environment and has many advantageous features. Accordingly, we built the agent interface services based on the MAGE environment.

4.3.1. Directory service

Grid applications often involve large amounts of data and computing and are not easily handled by today’s Internet and web infrastructures. Grid technologies enable large-scale sharing of resources within groups of individuals and institutions [18]. In these settings, the discovery, characterization, and monitoring of resources, services, and computations are challenging problems due to the considerable diversity, large numbers, dynamic behavior, and geographical distribution on the entities in which a user might be interested.

Consequently, directory services are a vital part of any grid software or infrastructure, providing fundamental mechanisms for discovery and monitoring, and hence for planning and adapting application behavior. In AGriP, there are two types of directory service. Correspondingly, there are two types of agents: DF (Directory Facilitator) agents and GISA (Grid Information Service Agent) agents.

DF is a mandatory component that provides a yellow pages directory service to agents. It is the trusted, benign custodian of the agent directory. MAGE may support any number of DFs and DFs may register with each other to form federations. Every agent that wishes to publicize its services to other agents should register its service description with DF. Also an agent can deregister itself from DF, which has the consequence that there is no longer a commitment on behalf of the DF to broker information relating to this agent. At any time, and for any reason, the agent may request the DF to modify its service description. An agent may search in order to request information from a DF.

We extend DF with a more abstract interface so that you can query: which agent can do classifying/cluster work? Which agent is available at hand? Since we add a reasoning machine embedded in DF, it can “think” by itself. If it thinks that a single agent cannot do a specific job, it may return more than one agent whom together can do that job, which often happens in distributed data mining. Moreover, if it cannot find agents that are capable to finish the work, it may resort to other DFs for requesting help.

GISA contains static and dynamic information about computing resources, as well as static and dynamic information about the network performance between computing resources. It provides an information directory service. One can query GISA to discover the properties of the machines, computers and networks or other common resources that you want to use: what is the state of the computational grid? Which resources are available? How many processors are available at this moment? How much bandwidth is provided? Is the storage on tape or disk? GISA provides middle-ware information in a common interface to put a unifying picture on top of disparate equipment. The GISA uses the LDAP (Lightweight Directory Access Protocol) as a uniform interface to such information.

\(^4\) http://www.fipa.org/specs/fipa00061/.
4.3.2. Resources management

We build the GRMA (Grid Resource Management Agent) in a MAGE platform, which is an important agent that provides capabilities to do remote-submission job start up. GRMA unites Common Resources and services, providing a common user interface so that one can finish a job with any common resource or service. GRMA is a general, ubiquitous service, with specific application toolkit commands built on top of it.

The GRMA processes the requests for resources for remote application execution, allocates the required resources, and manages the active jobs. It can also return updated information regarding the capabilities and availability of the computing resources to GISA and DF.

Furthermore, GRMA provides an API for submitting and cancelling a job request, as well as checking the status of a submitted job. We extended Globus Resource Specification Language (RSL) to describe requests. Users write requests in ERSL and then the request is processed by GRAM as part of the job request.

For example, suppose in the ics-domain (ics.ict.ac.cn) [17], which connects hundreds of machines to form a Grid. If one wants to start a MAGE platform in the domain, however, his machine is busy to the moment. He could first query DF to ask which agents have the capabilities to provide information services. After DF tells him a GISA agent can do it, he then queries GISA which machine is free in ics-domain with a command: “GISA-query-machine ics-domain”. Once he gets an answer that “machine-name.ics.ict.ac.cn” is free, he can start a MAGE platform on host “machine-name.ics.ict.ac.cn” with a request for AGrIP to run “java machine-name.ics.ict.ac.cn mage.Boot -gui”.

4.3.3. Data management

In an increasing number of scientific disciplines, large data collections are emerging as important community resources. In domains as diverse as global climate change, high-energy physics, and computational genomics, the volume of interesting data is already measured in terabytes and will soon total petabytes. The communities of researchers that need to access and mining these data (often using sophisticated and computationally expensive techniques) are often large and are almost always geographically distributed, as are the computing and storage resources that these communities rely upon to store and analyze their data.

DMA (Data Management Agent) is built in MAGE as an agent mainly aiming at accessing remote data and avoiding useless data transfer. Specifically, when a client submits only one distributed data mining task, it is possible to optimize the overall computation time when several servers can perform the computation or when data needed are already stored on storage elements. However, considering from the data side, if data needed by the computation are not already present, there is no possibility to optimize. Now, if the client submits a sequence of DDM tasks, which share data, then some data may be transferred more than once. If the result of a task is used by the next task in the sequence, then it is useless to transfer this result to the client and back to the platform.

A data management service, added to the platform, will register and store this middle result to avoid these useless transfers. Using this service, clients will conduct their data mining more efficiently.

5. Development toolkit

Although having grid platform AGrIP and several interface agents, it is useless if we have not the corresponding data mining agent development environment. Hence, we built a visual agent studio named VASTudio to make up this gap. VASTudio is designed to provide the agent software developers with an integrated environment for quickly constructing intelligent agents and agent-based software. Because of integrating powerful algorithm libraries, it is very convenient for developers to create industry data mining agents.

5.1. Hierarchical structure of VASTudio

From a software engineering perspective, the reusable classes are the main components of new software and programmers prefer to integrate the existing modules for rapid development. Compliant to this view, VASTudio adopts a hierarchical structure, which is composed with four different layers: the algorithms library layer, behavior layer, agent layer and society layer, shown in Fig. 4. The lower layers supply the fundamental functions or interfaces for the upper ones to construct more complex applications.

In the algorithms library layer, classification, clustering and associations algorithms are main components since numerous practical algorithms are successfully developed in the past decades. Especially, the field of machine learning has made substantial progress, which generates many sophisticated integrating techniques for distributed data mining. Under this situation, it is a natural idea to integrate these algorithms into a toolkit for software reuse. VASTudio builds an extensible algorithm library and provides the interaction mechanism for them to run in parallel and collaborate with each other. In order
Table 1
Algorithms layer description

<table>
<thead>
<tr>
<th>Link Type</th>
<th>Entity Component</th>
<th>Procedure Call</th>
<th>Entity Name</th>
<th>Entity Type</th>
<th>Link Name</th>
<th>Link Type</th>
<th>Entity Component</th>
<th>Entity Resource</th>
<th>Precondition</th>
<th>Post condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe</td>
<td>Entity Component</td>
<td>Procedure Call</td>
<td>Entity Name</td>
<td>Entity Type</td>
<td>Link Name</td>
<td>Link Type</td>
<td>Entity Component</td>
<td>Entity Resource</td>
<td>Precondition</td>
<td>Post condition</td>
</tr>
<tr>
<td>Cache</td>
<td>Entity Component</td>
<td>Procedure Call</td>
<td>Entity Name</td>
<td>Entity Type</td>
<td>Link Name</td>
<td>Link Type</td>
<td>Entity Component</td>
<td>Entity Resource</td>
<td>Precondition</td>
<td>Post condition</td>
</tr>
<tr>
<td>Message Passing</td>
<td>Entity Component</td>
<td>Procedure Call</td>
<td>Entity Name</td>
<td>Entity Type</td>
<td>Link Name</td>
<td>Link Type</td>
<td>Entity Component</td>
<td>Entity Resource</td>
<td>Precondition</td>
<td>Post condition</td>
</tr>
</tbody>
</table>

Besides mining algorithms, another challenge is how to integrate the local results to a global one. The integration is not simply putting together results from all sites because an interesting pattern in a local database may not be an interesting pattern globally. In VAStudio, we employ the ideas from meta-learning and ensemble learning such as voting, weighted voting and stacking etc. All integrating algorithms are stored in the integration algorithms library, which can be reused in the behavior layer.

Behaviors are the basic components to build agents. In VAStudio, every behavior has at least one action. We provide the definition of the base Behavior Class and every child Behavior (i.e. ID3_Behavior, C4.5_Behavior) is generated as a subclass of it, which implements common interfaces for subclasses to inherit.

Although behaviors are components having concrete actions, they cannot be dispatched to remote hosts and executed as independent entities. They should be used by autonomous and mobile agents, which hold runtime environment and lifecycle. In VAStudio, agents wrapping concrete behaviors can be moved to distributed data sites for independent run and collaborate with their peers for complex DDM applications. Like behavior, we have implemented the base class Agent for Child-Agent to inherit.

The highest layer of the structure is agent society that provides a platform for simulating multi-agent interaction in distributed data mining. In this layer, we can monitor the message transmission, collaboration process and move route of mobile agents.

5.2. Implementation

Considering a heterogeneous environment in distributed process, VAStudio is developed in the Java language, which has superior characteristics on cross platforms and databases.

In earlier versions, we do not emphasize the data mining function and aim to provide a general multi-agent development platform. Accordingly, the toolkit learns many experiences and lessons from several mature multi-agent systems and development platforms, such as JADE [12], ZEUS [13] and Aglets [14] etc. However, different from these systems, we adopt plug-in techniques to make the toolkit extensible like MATLAB [16], which can continually update and add new functions according to practical requirements. In the recent version, we specially built the DDM algorithms library and aim to make it a powerful toolkit for constructing agent based DDM.

As Fig. 5 shows, VAStudio is composed of six main parts: the visual coding environment, algorithms library, debug frame, hierarchical Behavior–Agent–Society frame, menu bar and tool bar. Visual coding environment offers an ideal window for Java-oriented programmers to implement agent-based applications. Considering some programmers are not familiar with agent theory, we provide Behavior, Agent and Society building Wizards like most business software to help them quickly develop DDM systems. Moreover, the toolkit provides an integrated debug frame in the bottom, which makes it very convenient for users to correct the programs in real time.

In order to conduct the following experiments, ID3, C4.5, Cart and Ripper algorithms [15] are added to build relative sub-behavior classes, showed in Fig. 5. And then, through the Agent Wizard, we construct MobileAgent-ID3, MobileAgent-C4.5 that will be used for local data mining. In order to integrate the local models, a weighted voting algorithm is chosen from the Integrated Library.

6. Related work

Despite its relative infancy compared with centralized data mining, distributed data mining has already achieved important research fruits in the past years. Those systems operate on clusters of computers or use agent technology. Common to
all approaches is that they aim at integrating the knowledge, which is discovered out of data at different geographically distributed network sites. However, none of those, to the best of our knowledge, make use of the grid infrastructure for the implementation of the agent service, data access and communication.

**BODHI** is an agent-based distributed data mining system that offers an environment capable of handling heterogeneous distributed data mining. It has been designed according to a framework for collective data mining on heterogeneous data sites such as supervised inductive distributed function learning and regression [11].

**JAM** is an agent-based meta-learning system for DDM. It is implemented as a collection of distributed learning and classification programs linked together through a network of data sites. Each local agent builds a classification model and different agents build classifiers using different techniques. After local data mining, JAM provides a set of meta-learning agents for combining multiple models learnt at different sites into a meta-classifier that in many cases improves the overall predictive accuracy [9].

**Papyrus** uses Java aglets for supporting move data, models, results or mixed strategies. It supports different task and predictive model strategies. It is a specialized system for clusters, meta-clusters, and super-clusters. Each cluster has one distinguished node, which acts as its cluster access and control point for the agents. Coordination of the overall clustering task is either done by a central root site or distributed to the (peer-to-peer) network of cluster access points [10].

**PADMA** is an agent-based architecture for parallel and distributed data mining, which deals with the DDM problems from homogeneous data sites. Partial data cluster models are first computed by stationary agents locally at distributed sites. All local models are collected to a central site that performs a second-level clustering algorithm to generate the global cluster model [8].

**Kensington** is a PDKD system based on a three-tier client/server architecture in which the three tiers include: client, application server and third-tier servers (RDBMS and parallel data mining service). The Kensington system has been implemented in Java and uses the Enterprise JavaBeans component architecture [21].

7. Conclusion and future work

Grid is growing up very quickly and is going to be more and more complete and complex both in the number of tools and in the variety of supported applications. Distributed data mining on Grid is becoming a trend with business data which are prone to be geographically dispersed on different sites all over the world. Hence, how to build the DDM on Grid becomes an interesting topic in recent years. In this paper, we summarize our past work on applying agent technology to develop DDM application on Grid in detail. The contribution of this paper is that we systematically analyzed the issues of Agent Grid and implemented an Agent Grid platform AGriP which provides infrastructure for agent based DDM on Grid environment. Furthermore, a user-friendly and extensible development toolkit VASStudio is presented to aid users developing real DDM applications.

In future work, on the one hand, we plan to investigate the use of policy-based means to establish an agent service self-reconfiguration mechanism. Through the policy control, the platform can automatically adjust its agent services according to environment requirements. On the other hand, we plan to enrich the VASStudio’s data mining algorithms library, which aims to provide a more powerful development environment for programmers.

**Acknowledgements**

The first author’s work is supported by the National Science Foundation of China No. 60435010, National Basic Research Priorities Programme No. 2003CB317004 and the Nature Science Foundation of Beijing No. 4052025.

**References**

[21] J. Chatterjichat, Anarchitecture for distributed enterprise data mining,
Jiewen Luo, born in 1980. He received his B.S. in Computer Science Department from Jilin University and now is a Ph.D. candidate in Graduate University of Chinese Academy of Sciences (GUCAS). He is also a Research Assistant in Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include distributed artificial intelligence, agent grid and data mining. In addition, he is a member of China Computer Federation (CCF) and the Chairman of CCF Young Computer Scientists & Engineers Forum For Graduate Students (YOCSEF-GS).

Zhongzhi Shi, born in 1941. He is a professor and doctoral supervisor in Institute of Computing Technology, Chinese Academy of Sciences. His current research areas include artificial intelligence, multi-agent system and data mining. He is the Chairman of Working Group 12.3 of IFIP, vice president of Chinese Artificial Intelligence Society, and member of council of Chinese Computer Federation. He serves as an Academic Committee member of Institute of Computing Technology, Academia Sinica, and also a member of the Technical Committee of National Artificial Intelligence Laboratory. In addition, he is vice president of Chinese Society of Machine Learning and vice president of Chinese Society of Knowledge Engineering.