

DISTRIBUTED INFORMATION ACCESS FOR E-COMMERCE APPLICATIONS

Nasrullah Memon, Imdad Ali Ismaili*, Mushtaque Ahmed Korai** and Jamal Mustafa Shaikh*

Institute of Information Technology, MUET, Jamshoro

Abstract

In this paper an investigation is made for several key technologies in use to improve the effectiveness and efficiency of information access for e-commerce applications. The mechanisms as well as limitations of three mature technologies CGI, ODBC and Java/JDBC, which greatly enhance the ability of e-commerce applications to access multi-databases, are briefly reviewed. Two emerging technologies, Mobile Agent and XML, are also discussed in detail.

Keywords: CGI, ODBC, Java/JDBC, Mobile Agents, XML

1. Introduction

The rapid growth in the popularity of e-commerce applications in commercial areas significantly increases the necessity as well as the complexity of data retrieval and manipulation from multiple sources (DBMS) that may be heterogeneous in terms of platform, data format and structure.

E-commerce applications are information-business services through Internet, based on HTTP and TCP/IP. The original web services were provided by a number of HTML files stored in one or more Internet node and linked each other. Soon after the appearance of web people realized static HTML files, as data source can no longer

dynamic and diverse online services. Nowadays, an attractive web site usually provides information by more than one type of data, such as HTML documents, text and image files, real time audio and video, structured data in relational databases (RDBs), and even data from legacy systems.

Semi-Structured static HTML files are easy to be opened for direct retrieval, but difficult to be managed and queried based

variability. Unstructured text and multimedia data also have similar characteristics. On the other hand, the RDB as well as the legacy database systems (LDBs) are well structured and managed, suitable for dynamic and automatic update, easy for query. However, they are relatively closed to the open web by their management systems. Extra effort must be made to communicate for data retrieval [2]. Moreover, data for one e-commerce application may be stored distributively on several computers with different operating systems or DBMS. This is especially true for nation-wide and international corporations, and collaborating web services. Thus the ways of communication with DBMS vary.

Different format, structure and platform of data source require different methodologies of access. The question is how to access so many kinds of heterogeneous data sources efficiently. Ideally, users should be able to access and manipulate any data globally completely based on their desire. This transparency of data heterogeneity should also be extended to the front-end e-commerce application developers to achieve software engineering efficiency.

* Institute of Information Technology, University of Sindh

** Postgraduate Student NED UET Karachi

In the past decade the industry and academia put great effort to approach this goal. This survey focuses on two emerging key technologies Mobile Agent and XML/PMML after a brief review of the most popular existing e-commerce application technologies for RDBMS access.

2. Access to Multiple RDBMSs

Sophisticated RDBMSs are the obvious choice to replace HTML files as major data sources in e-commerce applications because of their abundance of useful business data and manageable well-defined structure. Meanwhile the need of running enterprise internal information systems more distributively naturally suggests the use of web architecture, so that the dominance of enterprise RDBMS is reinforced.

2.1 CGI (Common Gateway Interface)

CGI is the first practical solution (called CGI Script) to access RDBMSs. It is a standard interface for external program to interact with web server. CGI script, resides on the web server and is invoked by and run on the server when a client request requiring database access is received. It reads the input parameters in the request and queries the database through its DBMS, sends the result back to the server in certain format (usually HTML). The server then responds the client with that result [4]. This architecture of e-commerce application based on CGI is the original form of multi-tier architecture and a milestone since it significantly extends the traditional client/server architecture.

Theoretically the CGI scripts could be the middle tier to access more than one database on more than one platform, and even LDBS, as long as you can program the corresponding script for each particular type of data source with particular way of

access. And CGI script can be written in any programming languages, most commonly Perl and C++. So it seems to be able to do anything. Unfortunately, that is where the problem comes from. Since CGI scripts are application and platform dependent, any variation of the application and platform, for instance, change of DBMS system or system platform, or just change of the data presentation on a web page, requires development of new CGI program [4]. Large scale of custom coding and the unsafe nature and poor maintainability of the coding languages raise the risk of server system crash, which is vital for commercial online services [5].

Another vital weakness of CGI program is that the web server has to create one process to invoke and run the script for each client request [4]. A typical commercial web service usually has around a thousand users simultaneously online, and there continually are some users going down while new users coming up. Even the web server has only one CGI program to be invoked; it will have to maintain about one thousand concurrent processes and continual creations and shutdowns of processes and database connections, which exhaust CPU time and memory.

2.2 ODBC (Open Data Base Connectivity)

ODBC, a common interface between application programs and RDBMSs, is specified by Microsoft. It is a set of standardized APIs for the applications to invoke for issuing standard SQL queries against RDBMS. Different ODBC drivers, which both implement the ODBC API specifications and are provided by vendors of different types of RDBMSs, take care of the particular ways of access to their corresponding DBMSs. Calling ODBC API in CGI programs for RDBMS access

database vendor and platform. It is of

course up to the vendors of DBMSs whether to implement the ODBC drivers [6]. But it is obvious that they would desire to do so to keep their shares of market. ODBC provides CGI programs the practical ability to open connections to multiple RDBMSs (of different types).

The ASP technology and tools substantially achieve automation and modularity of server-side program development, greatly shorten the application development cycle, and obtain multi-database access ability due to ODBC, hence gain wide acceptance rapidly. But the performance problem is still left open because of the retaining of CGI mechanism.

Furthermore, the MS-platform based technology and tools bring a significant limitation to the choice of server platform. The C-based ODBC API also has drawbacks to interface with programming languages other than C.

2.3 JDBC (Java DataBase Connectivity)

JDBC, developed by JavaSoft, specifies a set of standardized Java abstract classes for Java applications to query RDBMS using SQL. The basic idea of providing standard API between applications and RDBMSs is the same as ODBC. There is no fundamental difference between JDBC and ODBC in terms of multiple database access. Java application calling JDBC to access a particular type of RDBMS also requires a JDBC driver provided by that particular vendor implementing the JDBC APIs and acting as a SQL engine, as for ODBC [6]. They both achieve independence on DBMS vendor and DBMS platform, and contribute to the efficiency on application development and maintenance. The advantages of using JDBC over using ODBC come from the difference of their context:

- (a) The programming language using JDBC is Java, which is platform independent, fully object-oriented and portable. Any Java e-commerce application using JDBC is independent of the database platform as the server platform. This provides greater flexibility to web development and maintenance. Meanwhile, programming in fully object-oriented Java (OOJ) eliminates the potential risk that would arise when otherwise using C++/ODBC due to some cumbersome features of C/C++.
 - (b) In Java, clients, web server, application program, JDBC driver and even the DBMS are all objects. Once loaded for running, they may coordinate with each other by efficient Java method calls and be shared by multiple requests, rather than keep creating and closing processes from time to time. This may solve the runtime performance problem of CGI [5].
- server-side Java program cooperating with web server to serve data requests, comes into play.
- (c) Java EJB technology takes advantage of JDBC and presents the data in RDBMSs into reusable objects of data models. It eases the programming complexity for e-commerce application to deal with large scale distributed databases.
 - (d) The downloadable Java applets also provide alternative for client interface implementation and flexibility of distribution of data handling tasks in e-commerce application development.

Therefore, as large scale RDBs become more predominant in web-based applications, the Java technology suite based on JDBC is deemed as the best choice for application implementation.

However, standardization and modularization always means extra runtime overhead while gaining adaptability, maintainability and scalability. In the pattern of client/web-server/servlet/JDBC/DBMS, multiple JDBC drivers for different DBMSs must be installed and run on the web server, so must the stationary servlet programs and HTTP service. Heavy burden is concentrated on the web server, so that the hardware performance of the server will become a bottleneck. Moreover, custom coding is still required for access of data sources other than RDBs. And Java-style API has problem when interfacing with programs in other languages.

3. Mobile Agent

The world wide distributed computing environment on Internet, the inflexibility of stationary e-commerce application software, the platform-independent portable nature of Java programs and data/function encapsulation methodology in object-oriented computing have naturally inspired movement of programs among hosts within a network. That is the emerging mobile agent technology. Derived from software agent technology, a mobile agent is an active program that can roam a network by transporting itself from host to host, and carry out autonomous operations to fulfill a user goal [7]. From OOP, a mobile agent is an object that contains its state and some result of its task, and most importantly its intelligent behavior including self-governing and self-dispatching.

3.1 Basic ideas and advantages to e-commerce applications

A mobile agent will at least have the following characteristics [8]:

(a) Mobility: transport itself from host to host within a network then resume execution

- (b) Autonomy: settle, execute and move independently without supervision
- (c) Loyalty: performance computation on behalf of the user who creates it
- (d) Intelligence: learn from the information it gets and make decisions, e.g. where to move next
- (e) Recursion: create child agents for subtasks if necessary.

The mobile agent has maximum flexibility and is applicable to the whole variety of network-distributed applications, yet is hard to implement based on the current technologies. There is a long way to go before we can realize a generic mobile agent and reach a level of widespread adoption. Applications are still on experimental level and concentrate on some specific but most valuable areas, including web databases, cooperative environments, information gathering systems and E-commerce systems [9].

In e-commerce application, the idea of mobile agent may be refined a bit further:

- i. It holds the request and its parameters,
- ii. It traverses one or more servers to satisfy the request, and goes back to deliver the result when finish.
- iii. It requests web service as a representative of the user, collecting information, negotiating, purchasing, and paying money.
- iv. It determines whether traverse by itself or creates and sends out child agents for the rest of the task according to some pre-set strategy.

The benefits of applying mobile agents over traditional client/server architecture

are obvious. The computation burden can be flexibly distributed among the client system, the web server and other service providers that host the mobile agent [9]. The need of continuous network connection can be reduced. Thus the performance and usability of both the e-commerce applications and networks increase, especially in case of slow or expensive networks such as wireless [8]. The web client can access any online data source regardless its vendor, platform, structure and storage format, provided some mechanism interfacing between mobile agent resident and the data source application exists locally on that host, say, a local stationary agent associated with the data source as mobile agent residence manager. The only care the client must take is the user's input to be passed as request. Isn't that our dream of heterogeneity transparency?

3.2 Infrastructure of web-based mobile agent applications

The domain of mobile agents is typically a heterogeneous network consisting of not only various host platforms and data sources in different structures, but also different types of network connections and protocols. For the mobile agents, standards of application framework and communication protocol are definitely needed [11].

Infrastructures proposed by different groups of researchers up to now are slightly different due to the implementation languages, and the level of abstraction, they are essentially similar in terms of agent runtime environment. A specialized server program usually run on each node that will host mobile agents and provide them requested data-related services. This so-called agent server is responsible for accepting and authenticating agents, assigning proper resources and access authorizations for the

communication and transfer and helping the agent to request local services [11]. For the last responsibility it requires a translator which is just a JDBC driver that would convert the SQL queries into proper internal commands of the local applications, send those commands to applications, get the corresponding results back and respond to mobile agent through the API.

The role of mobile agent varies a little in different proposals due to the level of abstraction, although mobility and intelligent autonomy are the common characteristics. An agent contains the code that defines its behavior, the state that represents its current activities and status and some result of its tasks, the attributes identifying itself including origin and owner, tasks and resource requirements, authentication ID, etc. The attributes should not be changed by agent itself and is accessible to the infrastructure [11]. The behavior generally includes all of the autonomous functionality. In some cases, agents are refined into categories of mobile agents with different roles, for example, DBMS agent takes care of creation and maintenance of database connection and information query, while messenger agent is responsible for message transfer between client, web server and data sources, and coordinator agent resides on web server as parent agent in charge of the parallel work of its child agents with subtasks. Stationary agent also introduced as intermediate helper between mobile agent and local data sources [8].

The role of client system is somewhat ambiguous, partially because the traditional form of web client, i.e. HTML browser, client-side script, downloadable program and HTTP connection with server, seems still applicable to current researches. Two opposite trends due to level of mobile agents may become clearer. Client may fade out and be

replaced by a similar agent environment with some stationary agent interfacing with user, or may remain thin and only take care of user interface while the direct server provides agent creation service. The architecture presented in [11] is close to the latter case.

3.3 Mobile Agent communication and mobility

A mobile agent infrastructure provides runtime platform which could be implemented in different ways by different vendors. Furthermore, the mobile agents themselves could be heterogeneous as well, if implemented in different languages. A protocol is needed for them to communicate with each other, with any agent platforms, and to be transferred between platforms. The only available protocol draft for standardization that covers both mobile agent communication and mobility and most possibly becomes standard is ATP/0.1 [13], which is based on HTTP and submitted by IBM. A language called KQML could be an alternative for agent communication [14].

3.4 Problems and future work

The immediate and hard problem is security. The foundation of the mobile agent technology is to enable foreign code to run on local host, which is also foreign to that code. Two security aspects arise: the security of the data sources and that of the mobile agent [12]. The first issue could be solved by existing technologies, for instance, using interpreted languages to implement mobile agent, or using sandbox model similar to Java security manager. The other issue, to protect mobile agents from malicious attack from either some host or other mobile agent, is the hardest task and becomes the most salient problem when the mobile agent represents the user to perform E-commerce transactions or

carries some precious and vital data. Most ongoing researches can only provide partial solutions. One compromising but guaranteeing solution is that mobile agents only migrate to trusted hosts, so that the openness of e-commerce applications is seriously restricted. Another technique on the way is called "code obfuscation", which is like a black-box holding the agent code and defers the attack since the attacker must analyze the black-box code first [12].

Another open issue is transaction management. New transaction model is required to meet its mobile and asynchronous nature. For example, A mobile agent may send out some child agents to other data sources for subtasks while it is performing query as well. The mechanism must be in place to avoid and detect deadlock among agents. Moreover, the transactions between mobile agent and data application update not only the data source but also the state of the mobile agent. The consistency of the state of mobile agent is as important as that of data source, so that efforts must be taken to extend the existing transaction management technology for problems like how to recover the state of mobile agent from aborted transaction [12].

The reliability of communication becomes a problem for agents with high mobility. What flow on the network are the mobile agents in migration? However, message passing may be needed for agent management, which includes instructions from parent agent and orphan detection by child agent, and remote communication between agents or between agents and remote data sources. The traffic at each communication node of the network must consist of a mixture of agents and messages. To guarantee reliable message delivery to agents that move very frequently, especially in the presence of

link or node faults, requires advanced communication mechanism [15].

In the early days mobile agents, were implemented by languages such as safe-Tcl and Telescript for security concern of agent hosts [11]. Later researchers tend to use Java as uniform implementation tool for both mobile agent and agent platform. The trend is natural due to Java's superior characteristics including platform independence, host security mechanism, fully object-orientation and network awareness. Network awareness releases developers from the tedious task of taking close care of the network communication details. The object serialization mechanism in RMI (remote method invocation) package using Java offers an easy way to migrate objects. IBM has developed a fundamental set of Java classes as a Java extension package for mobile agent development, specially "aglet", and implemented an aglet server as an agent platform to be installed on any agent-hosting server. It seems to be a good starting point. Most ingredients for transparency of RDBMS heterogeneity seem to be ready. However, only Java is not sufficient for the wide applications of mobile agents. For example, JDBC is defined for Java programs to access various RDBMSs. But mobile agent will be facing much higher heterogeneity. Numerous non-relational data sources (NRDS) are seeking ways to the open Internet world. If Java is used to implement the local agent interface on those sources, more features have to be added to Java. A more general architecture CORBA, namely Common Object Request Broker Architecture that is defined by Object Management Group (OMG), standardizes program components implemented in different languages as objects, which APIs are specified in a neutral language - IDL. Integrating CORBA with Java technology will greatly enhance the platform independency

and interoperability of mobile agent technology in a highly heterogeneous world.

4. XML/PMML

Some researchers are pushing forward the uniformity of data format to achieve platform-independence of data. The main stream among these efforts is XML, Extensible Markup Language that is derived from a general markup language SGML (Standard Generic Markup Language), which is a standard system for defining document formats and allows documents to describe their own grammar. It is also the ascendant of the familiar HTML. Unfortunately, SGML is too general to be practically implemented. Its developers move onto a simplified subset of SGML, relaxing its generality meanwhile retaining the advantages of extensibility and structure, i.e. XML. According to the official XML documentation from the development organizer W3C, XML "is an extremely simple dialect of SGML ... The goal is to enable generic SGML to be served, received and processed on the Web ... XML has been designed for ease of implementation, and for interoperability with both SGML and HTML" [16].

The basic idea behind XML is converting data into structured and format uniformed text files, in the price of lower processing efficiency and security. Ordinarily, data in particular application is in particular format that only that application program can understand, and usually in binary format. Typical examples are data in RDBMS and engineering design by CAD software. Text files, on the other hand, are the simplest format for computers to read, understand and transform, and easy to reach extensibility, interoperability and platform independence [16]. HTML is a good example of pure-text formatted data in widely used application.

It is far from enough to describe data structurally. The tags in HTML only define how data will be displayed on the browser, rather than what the data is. It does not support specification of data structure, does not allow specification of different tags needed by diversity of data. From this point of view we can say XML is a data-oriented markup language. It uses tags to delimit data described in text format like HTML, but leaves the interpretation of the tags completely to the programs that want to use it. So any application has options to use any tag specification or define new tags to satisfy the complexity of its data. Tags can be nested to reflect the structure of any data. The most important point is, the XML file can contain the description of its grammar / tags, so called Document Type Declaration (DTD), so that interested programs know how to interpret it. "...XML describes a class of data objects stored on computers and partially describes the behavior of programs which process these objects" [16]. At this point we again see the hope of heterogeneity transparency. Due to its strong data description capability, XML has the high adaptability to any type of data. We can expect that different data sources, structured, semi-structured or non-structured, use XML as external data format so that applications can exchange information freely without any barrier. XML could even be used as internal storage format. XML query language (XQL) is under developed to query data sources in XML format. Since XML is originally tailored for e-commerce applications, it tends to play an important role in commercial B2B and B2C applications, as global format of data storage and interchange and document delivery technology.

The drawbacks of XML are, due to its pure text format, requiring more storage space and longer downloading delay, and

reading data from XML requires more processing time to parse the file and analyse the tags. Insecurity is another disadvantage of text XML documents. However, the continual improvement of processing power and compression techniques and the decrease of the storage expense partially compensate these disadvantages. And they should be deemed as the price of heterogeneity instead of that of XML itself.

Because the power of XML comes from that it allows user/application to define the grammar and attaches the DTD with the data, one can imagine that there will be enormous data type definitions for variety of data. PMML (Predictive Model Markup Language) is such a typical example. PMML v1.0 is recently proposed by Data Mining Group aiming at provision of data to data mining tools and other online analytic applications in a standard way so that the applications don't have to deal with the individual differences between data sources [17]. It defines a set of tags to describe data dictionary, tree model and regression model. They can be used to "specify the entities and attributes for documenting decision tree and multinomial logistic regression models" [17]. Something like PMML is an efficient way for companies to easily define data models and share them among compliant vendors' applications.

5. Conclusion

As computer application continually expands in all aspects of the society, the scale of data and data sources increase accordingly together with the data heterogeneity. On the other hand, the development of computer network, especially Internet, opens the door to provide and obtain data services globally. Data heterogeneity is one of the main obstacles to the openness of network and Internet applications. CGI, ODBC and

JDBC were the most suitable solutions for the data scale, hardware/software ability and commercial demands at those particular phases. Nowadays, we have come to the situation that our economic and social life largely relies on Internet and e-commerce applications. Complete openness on Internet becomes an unprecedented strong demand and necessity for human beings. Data heterogeneity as transparency becomes the biggest enemy of the virtual society. Among all efforts of attacking this enemy, there are two most promising directions. One is mobile agent framework based on some code platform-independency technologies such as Java and CORBA. The other is data platform-independency technology such as XML.

References

- [1] Florescu, Daniela, Alon Levy, and Alberto Mendelzon. "Database Techniques for the World Wide Web: A Survey." 1998.
- [2] Bouguettaya, Athman, et al. "World Wide Database – Integrating the Web, CORBA and Databases." 1999.
- [3] Bouguettaya, Athman, et al. "Using Java and CORBA for Implementing Internet Databases." Proceedings of the 15th Int. Conf. on Data Eng. 1999.
- [4] Lu, Hongjun, and Quan Xia. "JdWeb: A Proxy Server for Intranet/Internet Database Application Development." 1998.
- [5] Kuhlins, S., and Korthaus. A. "Java Servlets versus CGI." Decker, H., and W. Gaul, eds. Classification and Information Processing at the Turn of the Millennium. 1999.
- [6] Friberg, Torbjorn. "Evaluation of the commercial status of Java database connectivity (JDBC)." 2001.
- [7] Piszcz, Alan. "A Brief Overview of Software Agent Technology." 1998.
- [8] Papastavrou, Stavros, George Samaras, and Evaggelia Pitoura. "Mobile Agents for WWW Distributed Database Access." Proceedings of the 5th Int. Conf. On Data Engineering. 1999:228-237.
- [9] Samaras, George, et al. "Mobile Agent Platforms for Web Databases: A Qualitative and Quantitative Assessment." Proceedings of ASAMA'99. 1999:50-64. White, Jim. "Prospectus for an Open Simple Agent Transfer Protocol." General Magic, Inc. Online. 200.
- [10] Pogodaev Gleb et al, Access to Databases Over Web: Overview of Technologies", Proceedings of the 2nd International Workshop on Computer Science and Information Technologies CSIT'2000, Ufa, Russia, 2000
- [11] Lingnau, Oswald Drobniak, and Peter Domel. "An HTTP-based Infrastructure For Mobile Agents." WWW Journal - 4th Int. WWW Conf. Proc.. 1995.
- [12] Funfrocken, Stefan. "Integrating Java-based Mobile Agents into Web Servers under Security Concerns." 1997.
- [13] Lange, Danny B., and Yariv Aridor. "Agent Transfer Protocol – ATP/0.1." 1997.

- [14] Murphy, Amy L., and Gian Pietro Picco. "Reliable Communication for Highly Mobile Agents." 1999.
- [15] Finin, Tim, Yannis Labrou, and James Mayfield. "KQML as an agent communication language." Proc. 3rd Int. Conf. Info. And Knowledge Mgmt., ACM Press. 1994.
- [16] Bray, Tim, ed. "Extensible Markup Language (XML)." W3C. Online. 2001.
- [17] Cover, Robin. "Predictive Model Markup Language (PMML)." OASIS. Online. 2000.
- [18] Bos, Bert. "XML in 10 points." W3C. Online. 2001.