An Approach to Agricultural Information System based on Grid Concept

S. Ninomiya\textsuperscript{a}, M. Laurenson\textsuperscript{b} and M. Hirafuji\textsuperscript{c}

\textsuperscript{a} National Agricultural Research Organization, Japan, snino@affrc.go.jp
\textsuperscript{b} Horticulture and Food Research Institute of New Zealand, mlaurenson@hortresearch.co.nz
\textsuperscript{c} National Agricultural Research Organization, Japan, hirafuji@affrc.go.jp

Abstract

Agricultural decisions require several different types of data sets, because of the complexity of agriculture affected by many factors. Those data resources are comparatively small and located in different organizations. Emerging data grid technology offers a powerful mechanism for virtually integrating such requisite data and programs dynamically from multiple autonomous resources of the Internet. One of the challenges in order to realize such data grid for agriculture is to solve data heterogeneity because the formats and structures of databases differ greatly even for the same kind of data available. We suggested an idea of data mediation called data broker that provides client applications consistent accesses to such heterogeneous databases, and practically implemented the idea as MetBroker for meteorological data mediation. The original version of MetBroker has been operational, providing more than 20,000 stations of 23 heterogeneous meteorological data bases of the world that client applications can access with the same interface and proved the usefulness and powerfulness of the idea. The original one, however, has some weak points. In this paper, we first introduce the concept of grid based agricultural decision supports, and outline the mechanism of the original data brokers. Then, we suggest a newly designed data brokers based on EJB and semantic web technologies that gives us the solutions for the issues of the original data brokers.

Key words: data broker, distributed system, web service, semantic web, sensor network

1 Introduction

1.1 Grid for agricultural decisions

The most important advantage of the network is its use in information sharing between distributed resources. Such information sharing can greatly increase the amount of data available to users. In agriculture, several different types of data sets are required to make a decision because agriculture is a complex system affected by many conditions. Those data sets are usually maintained and managed by different organizations such as research institutes, weather networks, farmers’ unions, extension services and private sector organizations, and are located in different places. Therefore, presently, one has to collect data sets one by one for some decision, resulting in ineffective duplications of the same data sets in multiple locations. If these distributed data sets are virtually integrated and become sharable for agricultural decision support, it will surely accelerate effective decisions supports in agriculture.

To realize such data integration, we introduced the idea of grid which is increasingly important for sharing and integrating distributed resources, such as software and hardware, on the Internet. Especially a part of the Grid technology called data grid provides improved access to programs and effective utilization of available databases. The basic idea of data grid which is substantially different from legacy distributed systems is acceptance of heterogeneity and autonomy of distributed resources. This feature fits to the fact that data resources and programs handled in agricultural decision are distributed and managed by several
different organizations. Realizing such a feature of Data Grid, we can expect that data grid enables agricultural decision support applications to virtually integrate the distributed data resources such as weather, soil conditions, crop information, market prices and topology (Ninomiya 2001, 2002). In a grid-based decision support system, the network provides users with the necessary access to dynamically-linked programs and in situ data (Ninomiya 2001, 2002). In this approach, multiple users can share a single executable module, avoiding duplication of software development and maintenance while multiple programs can share the same data set, avoiding duplicated data maintenance and management. And data sets and programs that are managed and updated by their owners are dynamically linked in the Internet, providing diverse functionality to users.

1.2 Data mediation by brokers

One of the major challenges to realizing a practical Data Grid lies in addressing the heterogeneity of databases while retaining their autonomy. For example, typical weather databases accessible through the Internet rarely have the same data format, measured items, resolutions, and interface, so client application programs must be specially adapted for each database. To solve the issue of database heterogeneity, we proposed the use of data brokerage middleware to provide client applications with consistent access to databases (Fig. 1) and practically implemented the idea as MetBroker for meteorological data mediation (Laurenson et al. 2002a, 2002b, http://www.agmodel.org/). The original version of MetBroker has been operational, providing more than 20,000 stations of 23 heterogeneous metrological data bases of the world that client applications can access with the same interface and proved the usefulness and powerfulness of the idea (Table 1). The original one, however, has some weak points. In this paper, we propose the solutions for the weak points, utilizing the idea of web ontology and present a practical implementation of the idea. With this completely new design of the data broker, we become closer to more flexible and ideal data grid for agricultural decision supports. Seamless linkage of data grid to sensor network is also discussed in this paper.

1.3 Other data brokers

In addition to MetBroker, we have already developed SoilBroker, DEMBroker (Laurenson and Ninomiya 2002) and ChizuBroker. SoilBroker, DEMBroker and ChizuBroker mediate soil databases, digital elevation databases and map databases respectively (http://www.agmodel.org/). These types of the databases also important for agricultural decision support are now available on the Internet but heterogenous. Those brokers were implemented, succeeding the same architecture of MetBroker and utilizing the core part of the codes.

Fig. 1 Concept of data broker. Database drivers absorb heterogeneity of databases.
2 MetBroker and its new architecture

2.1 Original MetBroker

Based on the concept of the data brokerage middleware, we first implemented middleware for weather databases called MetBroker, originally using Java RMI (http://java.sun.com/products/jdk/rmi/reference/whitepapers/javarmi.html).

<table>
<thead>
<tr>
<th>Database</th>
<th>Number of Stations</th>
<th>Access Year</th>
<th>Access Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMeDAS Data (MAFFIN)</td>
<td>1479</td>
<td>1976</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>1989</td>
<td></td>
</tr>
<tr>
<td></td>
<td>137</td>
<td>2001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>1998</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1986</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1986</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1986</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2002</td>
<td>10 min</td>
</tr>
<tr>
<td>HortPlus Ltd</td>
<td>20&lt;</td>
<td>2002</td>
<td>10 min</td>
</tr>
<tr>
<td></td>
<td>12000&lt;</td>
<td>1994</td>
<td></td>
</tr>
<tr>
<td></td>
<td>152</td>
<td>1996</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1964</td>
<td></td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>1997</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>1996</td>
<td>15 min</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1987</td>
<td>15 min</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>1987</td>
<td>15 min</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1919</td>
<td></td>
</tr>
<tr>
<td>HortPlus Ltd</td>
<td>39</td>
<td>1996</td>
<td></td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>1997</td>
<td></td>
</tr>
<tr>
<td>HortPlus Ltd</td>
<td>6547</td>
<td>1853</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>1997</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>1993</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1995</td>
<td></td>
</tr>
</tbody>
</table>

MetBroker does not require any modification of the original database, allowing the autonomy of the database. MetBroker includes a meta-database to contain the access information of the weather databases available through MetBroker and the plug-in drivers which handle the heterogeneity of the databases to provide the consistent access to clients (Fig. 1). Once a new driver for a new database is plugged into MetBroker, all the client applications programmed to access the weather databases through MetBroker, instantly become able to access the new database without any modification.

MetBroker provides clients applications with details of available data, receives requests from the client applications specifying the elements, resolution and period required, queries remote databases and returns results (weather data requested) to the client applications. All the data communications between clients and MetBroker are encapsulated as objects. Requests can be either for a single station or the geographical area of interest. In the latter case, MetBroker returns results from multiple databases to the client, which is unprecedented functionality. The results of geographical requests can be used for spatially interpolation, for example. MetBroker utilizes a powerful metadata structure to provide catalogues of available data to client applications, and identify which databases should service geographical requests.
2.2 Defects of original MetBroker

Though we realized the usefulness and powerfulness of MetBroker, it still carries some issues;

- MetBroker, itself was published as an open source application but it needs commercial object orientated database as the meta-base.
- We need to restart MetBroker when a driver for a database is installed, modified or removed and it must be kept stopped when the meta-database is being updated.
- People need to know JavaRMI to develop client applications of MetBroker.
- To add new metrological elements to the MetBroker functionality is highly difficult because the elements are hard-coded in MetBroker.

The issue of the commercial software was solved by implementing a modified version of MetBroker that uses a free OOD, OZONE (http://www.ozone-db.org/frames/home/what.html) in stead of the commercial object orientated database (OOD), Object Store (http://www.objectstore.net/index.jsp) originally used. For the issue of the MetBroker operationability, we totally redesign MetBroker based on EJB (Enterprise Java Beans), using the J2EE (http://java.sun.com/j2ee/index.jsp) platform, so that all the drivers became independent of MetBroker (Fig. 2). This EJB based MetBroker is called MetBrokerEJB. By this modification, the issue was completely solved. The issue of JavaRMI was solved by providing a SOAP/XML interface as described in the next section.

2.3 MetSOAP and XML Web Services

As mentioned above, MetBroker was developed using Java RMI. Java RMI may be the best solution if we consider only the performance of the brokers or the grid system we are developing. Using Java RMI, however, brings us two disadvantages. One is that the client applications must be developed by Java, which limits the number of the programmers who can develop the clients because Java is still difficult especially among those involved in agricultural DSS development. The other is that Java RMI is not firewall friendly so that the accesses to or from the brokers frequently are blocked by the firewall. We first solved the second issue adopting HTML wrapping so that all the RMI accesses become equivalent to ordinal web accesses. This approach, however, cannot solve the first issue.

To solve both of the issues all together, we first developed the XML web service interface to the brokers, using the Java Web Services Development Pack (Sun Microsystems, http://java.sun.com/webservices/jwsdp/index.jsp). For example, MetSOAP, the nickname of the SOAP interface for MetBroker, receives a SOAP request that it invokes a corresponding MetBroker function through MetBroker’s RMI-based interface. MetSOAP transforms the function’s output into a SOAP object, which it then relays back to the client application. Most of the brokers functionality can be accessed through this interface. Using this interface, one can use any program languages such as Visual Basic, C, Perl, PHP and Pascal that provide SOAP interfaces in order to develop firewall friendly client applications of the brokers. We also redesigned this service based on EJB and implemented it so that it becomes easy to add new interfaces to MetBroker (Fig. 3).
3 Intelligent data broker based on semantic web

The last issue left is that the broker is not flexible to handle newly added metrological items. In this study, we attempted to develop a more flexible intelligent data broker that has inference capability to handle database heterogeneity based on ontology, again targeting metrological databases as the first step. An existing solution for this issue is to create some wrapper application for each database and to make it

Fig. 3 Three types of the accesses to the data brokers.

Fig. 4 Difference of the architectures between MetBroker (top) and MetBrokerEJB (bottom).

Fig. 4 show the difference between the original MetBroker and MetBrokerEJB. The last issue of inflexibility of MetBroker will be discussed in the next section.
communicate in SOAP/XML format. This technology possibly hides heterogeneity of access methods to
databases, but there is still an issue that SOAP/XML cannot hide semantic heterogeneity. Therefore, each
wrapper application needs to translate its local vocabulary into the global one. This translation is conducted
statically. This static link prevents clients from flexibly obtaining data. We employed RDF/OWL
(http://www.w3.org/2001/sw/WebOnt/) technology to solve this issue. This technology helps local
metadata connect to other metadata outside and realize a semantically integrated database. Intelligent data
broker is a challenge to provide clients with access to this virtually integrated database.

3.1 Implementation

The system implementation is still in the process of developing. This prototype version is coded in Java
language, and the main part of the inference engine utilizes Jena2 (http://jena.sourceforge.net/index.html)
provided by Hewlett-Packard Development Company, L.P. The intelligent data broker is composed of one
basic vocabulary file and as many item definition files and station metadata files as targeted databases. The
basic vocabulary file is an OWL file which has all standard weather items and other metadata description
vocabulary to describe weather stations. Each of the item definition file is an OWL file which has the local
vocabulary that is used in each database. And each of the station metadata file is an RDF file which
describes all the weather stations included in a particular database.

3.2 Single station access and spatial access

The simplest case is that clients request some weather items from a specified weather station. In this case,
after the intelligent data broker receives the request, the inference engine attempts to find the equivalent
items of the weather station. After finding the local items, it queries for data to the database by using the
local item and returns the data to clients.

This is the case that clients request some weather items for the weather stations located in a specified
gеographical area. In this case, after the intelligent data broker receives the request, it refers to the station
metadata files to find weather stations in the requested area. Then, the inference engine attempts to find the
equivalent items of each weather station. After finding the weather items, the process goes likewise.

4 Applications of data brokers

Conventional software applications rely upon querying a particular database structure. In contrast,
broker-based software applications request data from an intermediate broker rather than directly from a
particular database. The broker transforms the client application’s data request into database specific
queries, hiding the details of a particular database from the client application. The broker can also hold
metadata about each database; in the MetBroker case these metadata include details of which weather
elements are available from each station and at what resolution. Until now, several applications that utilize
the data brokers have been developed (Tanaka et al. 2000, 2001, Rafoss et al. 2004). Some of the
applications utilize more than two brokers at the same time.

Fig. 5 shows one of the simplest Java client applications of MetBroker that obtains the weather data of a
station based on the query about the database name, station name, weather items, duration etc. The
application is compatible with Resource Server (Laurenson et al. 2002) so that this single application can show the labels in several different languages automatically identifying the locale of the web browser. We also provided a servlet version of the simple application so that we can use non-java browsers on mobile phones to access the weather database through MetBroker (Laurenson et al. 2002). Tanaka et al. (2000,
2001) ported several legacy crop models to the MetBroker applications such as rice growth prediction
models and pest prediction models. For example, a rice blast prediction model ported for MetBroker can be
used anywhere if the necessary weather data for the particular place is available through MetBroker.

Fig. 6 left shows a client application that uses both MetBroker and ChizuBroker. The weather conditions of
all the weather stations within the rectangle region requested by the client are displayed on a corresponding
map. Because this application uses the spatial query function of MetBroker, the stations from more than
two different weather databases are automatically shown if the region is covered by multiple databases. Fig.
6 right shows another example of the combination of MetBroker and ChizuBroker which calculate and
display the risk of climatic extremes across a region by showing pie graphs of probabilities on the location
of the weather stations (Laurenson et al. 2001). It retrieves twenty years of daily temperatures (max and min) from all the stations in the nominated region.

As a simple demonstration of the utility of Web services, we wrote Excel spreadsheet Visual Basic macros that let a user select one of more than 10000 weather stations linked to MetBroker and import weather data from that station directly into a spreadsheet. The macros connect to MetSOAP via the Microsoft SOAP Toolkit and retrieve metadata to guide the user’s selection, and then retrieve the actual weather data from the selected station. Other MetSOAP client applications have been implemented using the Microsoft Toolkit approach in Visual Basic 6, Active Server Pages, and VB Script. Clients applications have also been developed in Borland Delphi (using the Borland SOAP Toolkit) and in Visual Studio .NET.

Imada et al. (2004) developed another Web service to interpolate meteorological data. This service utilizes MetBroker and DEMBroker’s Web services. A client program that needs interpolated weather variables can be developed rather easily by utilizing the Web service instead of accessing the two brokers separately and running an interpolation module in it.

5 Discussion

Through the development and operation of the original MetBroker, we were convinced that the approach based on the grid concept is powerful and suitable for agricultural decision supports where several heterogeneous data sets are needed to be integrated. Our conviction has become more strengthened through evolving MetBroker and other brokers to solve existing issues. Although meteorological data are most important in agricultural decision, they are not sufficient in most of the cases, indicating that the success with MetBroker is just a start. The heterogeneity among the metrological databases is comparatively
moderate and the design and the implementation of MetBroker was just a good exercise to learn the best approach to realize the targeted agricultural grid system. We have just launched the initial step to real data broker generally and flexibly adaptable to wide range of data resources by adopting web ontology. Flexible fusion of data grid with sensor network represented by Field Server (Fukatsu and Hirafuji 2003) is also a good direction that data brokers should follow.

6 References


