PaDDMAS: Parallel and Distributed Data Mining Application Suite

Omer Rana, David Walker, Maozhen Li, Steven Lynden, and Mike Ward
Department of Computer Science,
University of Wales Cardiff, POBox 916,
Cardiff CF24 3XF, UK

Abstract

Discovering complex associations, anomalies and patterns in distributed data sets is gaining popularity in a range of scientific, medical and business applications. Various algorithms are employed to perform data analysis within a domain, and range from statistical to machine learning and AI based techniques. Several issues need to be addressed however to scale such approaches to large data sets, particularly when these are applied to data distributed at various sites. As new analysis techniques are identified, the core tool set must enable easy integration of such analytical components. Similarly, results from an analysis engines must be sharable, to enable storage, visualisation or further analysis of results.

We describe the architecture of PaDDMAS, a component based system for developing distributed data mining applications. PaDDMAS provides a tool set for combining pre-developed or custom components using a dataflow approach, with components performing analysis, data extraction or data management and translation. Each component is wrapped as a Java/CORBA object, and has an interface defined in XML. Components can be serial or parallel objects, and may be binary or contain a more complex internal structure. We demonstrate a prototype using a neural network analysis algorithm.

1 Introduction

Data mining involves methods for efficient storage, retrieval and computation on geographically dispersed data sets. Such data may either be short lived transactional data, or may be detailed, non-volatile data providing a consistent view of an organisation, or a scientific experiment over a long period of time. To support decision making, various methods have been proposed to analyse patterns in such stored data, involving approaches from artificial intelligence, statistics, databases, optimisation theory and logic programming. The utility of such methods is generally determined by their speed of analysis, the range of problems that can be successfully analysed, and the range of data sets that are supported. These issues become particularly significant when dealing with large data sets, which are a first step in transforming a database system from a mechanism for reliable storage to one whose primary use is in decision support [2]. Associated with large data sets is the concept of OLAP *(On-Line Analytical Processing)* which involves the interactive exploration of data, involving generally computationally intensive queries. Current data mining efforts involve a computer driven exploration of data, often involving data processing where the user does not know how to precisely describe the query. Examples of such problems are present in many business scenarios, such as analysing credit card usage data (to detect fraudulent activity), looking for patterns in telecommunications data (to correlate faults with other regional events), analysing machine usage to predict execution times, and others. Various data visualisation techniques are also needed to support interactive exploration of data, and an area of concern is the ability to deal with data sets with a high dimensionality.

Various data mining toolkits are available today, which range from specialised libraries in general purpose packages such as Matlab and Mathematica, to specialised applications such as ISL's Clementine and SAS's Enterprise Data Miner, a good survey can be found at [11]. What is often missing in these toolkits is the ability to integrate third party tools, or share output generated by analysis tools. Also, most analysis algorithms are restricted to single processor machines, although some parallel tools do exist, such as [14], these often generate output in a proprietary format. Moreover, data mining toolkits are often aimed at statisticians or data analysts, with little support for scientists or business users. A great deal of attention has been placed on algorithms by developers, rather than end users in most data mining tool kits. However, we feel that hiding details of analysis from scientists and business users is as important as providing better and faster analysis algorithms. We describe a component based domain-independent data mining suite called PaDDMAS, where a user can visually construct domain specific data mining applications by plugging together software components written in C, Java or Fortran. Components may be sequential or parallel analysis algorithms making use of parallel libraries such as PVM and MPI. Interfaces

to databases capable of processing SQL queries are provided, with additional support for connecting to remote data sources. At present we only provide neural network based data analysis algorithms, and demonstrate a PaDDMAS prototype. A knowledge based system helps users in selecting components for analysing particular kinds of data, and new rules may be added through a web interface that allows users to add additional sharable components to their local repositories.

2 Previous work

Work in parallel data mining is distinguished by the particular method of analysis being employed, such as decision trees based on C4.5 [13], neural networks [5], statistical methods [16], genetic algorithms [7] and others. Pendse [10] gives a good overview of commercial data mining and OLAP tools, dividing commercial tools according to two criteria: location of the processing and location of data storage. Hence, regardless of where the data processing takes places, there are three places where data may be stored: in a Relational DataBase Management System (RDBMS) (with no external storage), in a shared multidimensional database on a server, or as local files on a client PC. Similarly, processing engines may use *multi-pass SQL*, use a dedicated server for handling multiple simultaneous users, or execute on the client PC. Several commercial products work in more than one way, generally making an exact classification difficult. Examples of RDBMS based storage and multi-pass SQL products include the *MicroStrategy DSS Agent*, whereas tools employing RDBMS based storage but other processing techniques include the *IBM DB2 OLAP Server*, the *Microsoft OLAP Server*, *Oracle Express* etc. Some vendors produce desktop OLAP tools that can be integrated into other applications, such as *Cognos PowerPlay*, tools from *Business Objects*, *Brio Technology*, *MicroStategy* and *AppSource*. Albrect and Lehner [1] provide an overview of different OLAP approaches in data warehouses with reference to the *CUBESTAR* system. They contrast an Enterprise Data Warehouse with a Data Mart, where an Enterprise Data Warehouse enables data from different business areas to be integrated, using a single corporate data model, while a Data Mart provides a highly focused version of a data warehouse, to serve a small business group with well defined needs. Various software and hardware vendors also provide tools for parallel data mining, such as *Torrent Systems*, *Tandem*, *Sun Microsystems* and *IBM* [15], supporting either parallel data access from a particular database (*DB2* in the case of IBM), or running a particular application program in parallel.

Our approach is most closely related to the Kensington Enterprise Data Mining System [4] which wraps analysis algorithms as Enterprise JavaBean components. Our system extends their approach to also provide support for third party components. Our work differs from other approaches in that we are not targeting our system to a particular analysis package or algorithm, and are not providing connectivity to a particular database. We leave such decisions to the analyst, and concentrate on developing an infrastructure within which third party tools may be integrated. For instance, the system uses JDBC bridges to connect to databases, and consequently, data extraction tasks are delegated to components capable of processing SQL commands and optimising SQL queries.

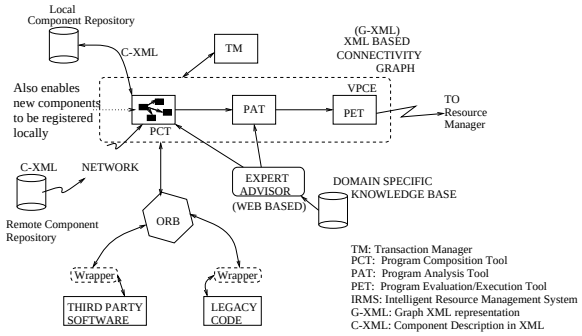

Figure 1: The PADDMAS Architecture

3 PaDDMAS Architecture

The PaDDMAS architecture is illustrated in figure 1. The Visual Program Composition Environment (VPCE) enables visual construction of an application by plugging together components held in two types of repositories. Repositories that are local to the VPCE, which generally contain components generated by a user (or set of users) at a particular site, or repositories that can be at one or more remote sites. Repositories can be shared by users, and this sharing is facilitated by an Expert Advisor, which helps to locate components of interest to a user. Components can provide analysis algorithms, such as a neural network, a decision tree or Bayesian classifier, or can provide data management functions, such as translating schemas between two data sources, data sampling etc. Each component is either a Java or CORBA object, with its interface specified in XML, according to a pre-defined set of XML tags, that must be used for all components within PaDDMAS. Components are stored in the Component Repository using this format, and any binary data associated with a component must also be tagged. XML tags may be used to automatically derive help on particular components already present in the repository, or it may be used to query the availability of particular types of components. User supplied components must also have their interfaces defined in XML. The Program Composition Tool (PCT) enables a user to connect two components only if their interfaces are compatible, performing checks on the

data distribution patterns between components for instance. The Program Analysis Tool (PAT) ensures that data sources are connected and verifies that data gathering tools are online and available for access. The Program Execution Tool (PET) then builds the task graph generated for every application built by the PCT, and passes the graph to a resource management system (IRMS) for scheduling and allocation of the task graph onto the available resources. The Transaction Manager (TM) module sits outside the VPCE, and enables recording of particular events, and provides persistent access to components being used within the VPCE. We use PJama [8] for checkpointing the state of a component, so that it can be re-started for binding at a later stage. PJama enables a reachability tree to be established for each component that is to be made persistent, and all related components connected to it being made consistent in the process.

The XML based component model ensures uniformity across components, and helps to define component structure and interface. Our XML definition enables the division of a component interface into a set of sections, where each section is enclosed within predefined tags. A parser capable of understanding the structure of such a document can identify and match components. The Document Type Definition (DTD) identifying valid tags can be obtained from a URL reference placed in the document header, and identified by the `href` tag. The XML definition can be used to perform information integrity (such as checking the total number of input and output ports), check the suitability of a component for analysing a particular data set (based on the type of data being analysed, the size of the data being analysed, or the location of the data relative to the component) the types of platforms that may support the component and internal component structure when available. The tags are divided into: `context` and `header`, `ports`, `execution` specific detail, such as whether the component contains MPI code, a user specified `help` file for the component, a `configuration` file for initialising a component, a `performance model` identifying costs of executing the component, for the resource manager, and an `event` handler, which enables registering or recording of particular types of events. A component may also contain specialised constraint tags in addition to the mandatory requirements identified above. Constrains can include security or license constraints, where a component is required to run on a particular machine or cluster. The component model for a neural network component can be described as:

```
<?xml version="1.0" href=URL?>

<preface>
  <name alt=DA id=DA01>Neural Network</name>
  <hierarchy id=parent>Tools.Data.Data
     Analyser.Neural_Network</hierarchy>
  <hierarchy id=child></hierarchy>
</preface>

<ports>
 <inportnum>2</inportnum>
 <outportnum>1</outputnum>
  <inportype id=1>float</inportype>
  <inport id=1 type=real/>
  <inport id=2 type=float/>
  <inport id=3..15 type=float/>
 <outportype> real </outportype>
</ports>

<execution id=software>
 <type>parallel</type>
 <type>MPI</type>
 <type>SPMD</type>
 <type>binary</type>
</execution>
...
```

A markup for data mining components has been initiated through the Predictive Model Markup Language (PMML) project [9], which enables users to exchange analysis components such as C4.5, ID3, CART etc, with the structure and intent of the component encoded in PMML. This encoding does not cater for data management components, and is therefore more restrictive than ours. Also, the emphasis is different, as a PMML description encodes data structure, whereas we encode interfaces to sub modules and performance models.

Connectivity to databases is provided through JDBC bridges, requiring the user to provide meta-data for the data source in XML. Hence, data access components make use of the XML Metadata Interchange (XMI) [6] standard, which enables a set of XML DTD production rules for transforming meta-models and meta-data representations between CORBA objects. Correspondences between data models are left for the user to specify, however the PAT tool ensures that constraints are adhered to. Data access components also control the quantity of data transfered across a network, providing a range of data sampling functions, from random to user defined interval sampling. Cache and memory management is relegated to the IRMS, which must also ensure that license and security is not being violated.

Figure 2 demonstrates two use cases by combining data analysis and management components. We assume an acyclic invocation, with connectivity specified by directed links. Data may be streamed through the components, and does not need to be replicated at each point in the flow graph. The output generated may be visualised using a simple graph tool, or a more sophisticated visualisation suite.

Component selection is achieved through an expert advisor, illustrated in figure 3, where components are first divided based on the kind of data they can be used for – textual/symbolic data, numeric data or a combination of the two. Constraint propagation is used to reduce the number of possible components that may be used in a particular instance. If no components match the needs of a user, or if a new component is to be added, the associated condition un-

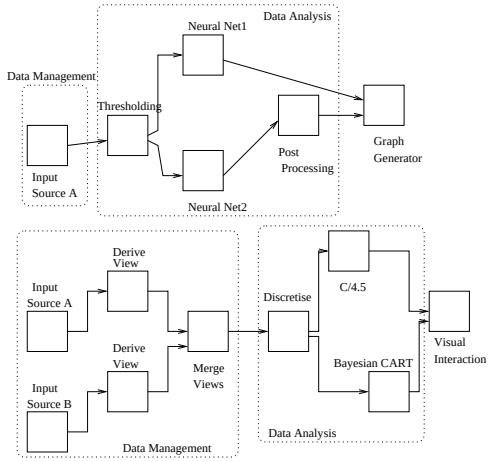

Figure 2: Use Cases – Constructing Applications in PaDDMAS

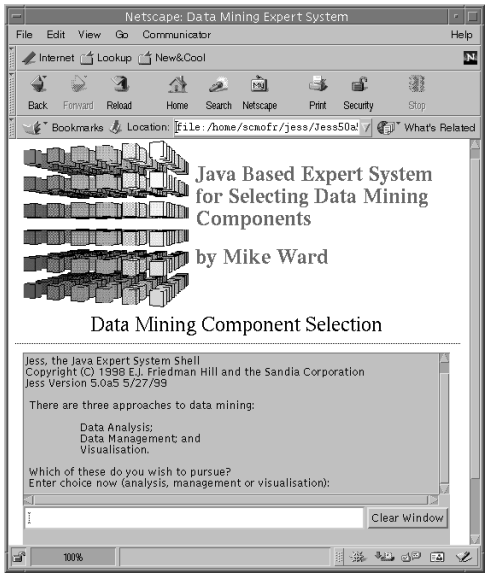

Figure 3: Web Interface for the Expert Advisor

der which this component may be used needs to be defined by the user. Rule files are co-located with the component at the same site. Constraint checking is through string matching with the XML definition of a component, which can include type of an algorithm being used, the size of data sets the algorithm can handle, whether the algorithm requires pre-processing etc.

Three types of components are provided within PADDMAS:

- Data Management Components: these provide predefined routines for intergrating schemas in different databases, providing alternative views of data within a structured database, and parsing a flat file with comma or space delimited data vales. A specialised component is provided for each task associated with extracting data. In the future, we will add components that can also support data quality management, such as filling in missing values, thresholding to identify outliers etc.

  To obtain a column of numeric data from a table of experimental results stored in a structured database, a component can generate the SQL query:

```
SELECT Results_Table.ColumnX1 FROM Results_Table
  WHERE result > 10
```

  The general form of the query can be:

```
SELECT ColumnX1, ..., ColumnX2
  FROM Relation1, ..., RelationN
 WHERE Condition1, ..., ConditionM
 ORDER BY AttributeX
```

  A standard component is created that can be instatiated with values identifying relations and conditions, as described in the above code segment. These correspond to inport/outport defintions in XML. Hence, data from multiple tables can be modified based on the WHERE and ORDER clauses above. The ORDER clause is one instance of other functions that may be introduced to retrieve data from a structured source.

  The data management components also enable sampling of a data set, when the complete data may be too large to migrate to the analysis component. For neural network learning, this is performed by extracting one or more columns of numeric data, ordering these, and then randomly selecting records from the ordered list.

- Data Analysis Components: these wrap particular analysis algorithms, such as a neural network or a decision tree generator, for instance. Each data analysis component can be treated as a black-box by the user, or a user can provide their own components, which may be developed by either combining existing ones, or by implementing an analysis algorithm directly. In both instances, the user will need to construct the XML interface for the component, and if it is to be made sharable, to publish it to the expert advisor.

  A Data Analysis component must expose its internal data distribution to the VPCE, enabling the program analysis tool discussed in section 3 to perform consistency checks when components are connected. This is particularly true for SPMD components, which can have `cyclic` or `block` distribution amongst process groups. For a neural network component implemented in HPJava [3] we use the representation:

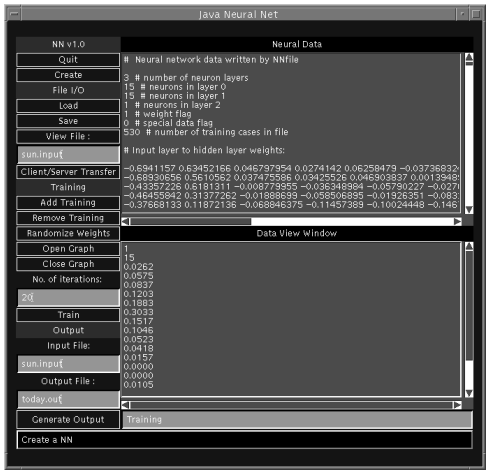

Figure 4: GUI for neural network application

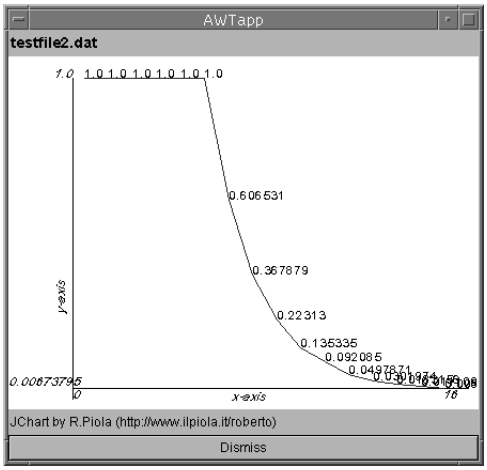

Figure 5: Graph output

```
Procs1 p = new Procs1(4,4) ;
 on(p) {
   Dimension d = p.dim(0) ;
   Range u = new BlockRange(N, d) ;
   Range v = new BlockRange(L, d) ;
   Range t = new BlockRange(M, d) ; ...
```

which indicates that vectors `u,v` and `t` are `Block` distributed over dimension (0) of a 4 by 4 process array. If the data for these vectors is retrieved from a database using SQL queries, then the column of numeric data obtained must be distributed in this way. Standard components are created to provide this type of data distribution. Use of standard analysis components also facilitates load balancing, as component granularity can be modified, based on available resources.

- Data Visualisation Components: these can be simple graph generators, as in figure 5, or they can be more complex interactive visualisation tools, that enable navigation through a data set, or 3D visualisation.

4 A Neural Network Prototype

An application employing a neural network component was developed to demonstrate the PaDDMAS architecture. Figure 4 illustrates the user interface where output is displayed using a graph generator illustrated in figure 5. The graph generator is a public domain package [12] that has been integrated within a data visualisation component.

A typical session would involve the user identifying the data source, loading the complete or partial data set, running one or more analysis algorithms on the data set, and generating a graphical output from the system. The user is given the option of down-loading an entire data set, or to send a filename of a data source to the remote host. Interactions between the user interface and the server implementing the neural network are via Java RMI. Data distribution at the remote site is managed by a local run time system, and sent as a parameter to the remote site by the client. The user can change parameters locally, such as the number of iterations or the state of the neural network at any part of the training phase, these parameters are serialised and sent to the remote neural network for analysis.

The client user interface also offers a data browser for inspecting the contents of a remote file, and performing simple checks for outliers on data sets. Further data integrity is not performed at present, and left to the user or the underlying database to handle. The data source may be a flat file or a structured database. The use of a ‘view derivation’ component enables a part of a database to be extracted reducing the size of data transfer to the analysis tool.

5 Conclusion

We define a system architecture for parallel and distributed data mining, which enables third party components to be integrated to create an application. The architecture is based on the Java programming language, with components being Java or CORBA objects containing XML interfaces, and with support for parallelism provided via MPI libraries. Hence, components can be internally parallel, and are wrapped as a Java or CORBA object. The ORB is not used to communicate between active processes, this is still achieved using the MPI-Communicator, however, the IDL interface is used to invoke operations on an object, some of which launch the MPI-Runtime. Each parallel object can have its own version of MPI (i.e. its own runtime environment), hence this approach also provides a way to link different MPI implementations together. The system is currently under development as part of a larger multi-disciplinary Problem Solving Environment (M-PSE) project. We show how the system may be used for a neu-

ral network application across a number of workstations as a *proof-of-concept* application. A developer may implement various data mining algorithms on the system, and may link such algorithms to parallel and sequential data sources via specialised data management components. Third party tools may also be integrated into the system. Previous tools have stressed analysis components, but not data management functions. The Expert Advisor is specific to an application domain, and supports two user categories: (1) domain users, such as physicists, chemists or biologists who are not interested in creating new components and therefore do not directly alter the rule base; (2) domain developers, mainly data analysts or statisticians, who create new components and therefore could potentially make alterations to the rule base.

The XML component model is central to PADDMAS, as both the expert advisor and the composition tool (VPCE) are based on this. An XML based interface also enables us to hide component implementation, enabling components to wrap executables in Java, C or Fortran. The particular contribution of our work is the ability to construct data mining applications from pre-packaged components, enabling components to be located at distributed sites based on their properties, and providing specialised components which differentiate between data management and integration, data analysis and visualisation.

References

[1] J. Albrecht and W. Lehner. On-Line Analytical Processing in Distributed Data Warehouses. In *IDEAS proceedings*. IEEE Comput. Soc. Press, July 1998.

[2] P. Bradley, U. Fayyad, and O. Mangasarian. Data Mining: Overview and Optimization Opportunities. In *INFORMS: Journal of Computing*, 1998.

[3] B. Carpenter, G. Fox, D. Leskiw, X. Li, and Y. Wen. Language Bindings for a Data-Parallel Runtime. *NPAC - Syracuse University, Syracuse, New York 13244*, 1997.

[4] J. Chattratichat, J. Darlington, Y. Guo, S. Hedvall, M. Kohler, and J. Syed. An Architecture for Distributed Enterprise Data Mining. *HPCN, Amsterdam*, 1999.

[5] M. Craven and J. Shavlik. Using Neural Networks for Data Mining. *Future Generation Computer Systems*, 1997.

[6] DSTC. XML Metadata Interchange, 1999. See Web site at `http://www.dstc.edu.au/MOF/`.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989. ISBN: 0201157675.

[8] M. Jordan and M. Atkinson. Orthogonal Persistence for Java. *Third International Workshop on Persistence and Java, Tiburon, CA, Sep 1-3*, 1998.

[9] UIUC National Center for Data Mining. Predictive Model Markup Language, 1999. See Web site at `http:///www.ncdm.uic.edu/`.

[10] N. Pendse. OLAP Omnipresent. *BYTE*, February 1998.

[11] G. Piatetsky-Shapiro. KDNuggets Directory: Data Mining and Knowledge Discovery Resource, 1999. See Web site at `http://www.kdnuggets.com/`.

[12] Roberto Piola. JChart, 1997. See Web site at `http://www.ilpiola.it/roberto/jchart/`.

[13] R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, 1997. ISBN: 558602380.

[14] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A Scalable Parallel Classifier for Data Mining. In 22^{nd} *VLDB Proceedings, Bombay, India*. 1996.

[15] Torrent Systems and IBM Corporation. Whitepaper: Achieving Scalable Performance for Large SAS Applications. 1997.

[16] B. Tabachnick and L. Fidell. *Using Multivariate Statistics*. Addison Wesley, 1996. ISBN: 0673994147.