PaDDMAS: Parallel and Distributed Data Mining Application Suite

Omer Rana, David Walker, Maozhen Li and Steven Lynden

Department of Computer Science,
University of Wales Cardiff, POBox 916,
Cardiff CF2 3XF, UK.
O.F.Rana@cs.cf.ac.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 toolset 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 framework for developing distributed data mining applications. PaDDMAS provides a toolset 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 and written in Java.

Keywords: *Data Mining, Java/CORBA, MPI, Neural Networks*

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 business problems that can be successfully analysed, and the range of data sets that are supported. Such 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 [3]. 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. Whereas traditional OLAP involves a human driven exploration of data, data mining involves a computer driven exploration, in many cases 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, from analysing credit card usage data (to detect fraudulent activity) to looking for patterns in telecommunications data (to correlate faults with other regional events), for instance. 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.

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 which are independent of location, programming languages or platforms. Hence, components may be sequential or parallel analysis algorithms written in C or Fortran, or they may be components developed in Java or Java-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.

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 [8] and others. *SLIQ* (Supervised Learning In Quest) developed by IBM's Quest project team, is a decision tree classifier designed to classify large training data [1]. *SLIQ* pre-sorts attributes when building a decision tree, and maintains separate lists for each continuous attribute and a memory resident data structure called the *class list.* Although *SLIQ* can handle disk-resident data efficiently, it still has to maintain the *class list* in memory, which grows in direct proportion to the total number of records in the training set. Shafer *et. al.* present a decision-tree based parallel classification algorithm called *SPRINT* [14], which overcomes some of the restrictions in *SLIQ*.

Pendse [12] 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 an 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 server employing multi-user capability 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*, OEM based tools from *Business Objects*, *Brio Technology*, *MicroStategy* and *AppSource*. Albrect and Lehner [2] provide an overview of different OLAP approaches in data warehouses with reference to the *CUBESTAR* system. They contrast an *Enterprise Data Warehouse* providing integrated data from different business areas stored with a single corporate data model, with *Data Marts* providing a high-focused version of a data warehouse, and developed 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 UDB* in the case of IBM), or running a particular application program in parallel. The emphasis is generally on particular hardware architectures, database systems or particular analysis packages such as *SAS* or *Matlab*. Parallel database access requires the vendor to provide a means of extracting and merging data obtained in parallel from a database, generally by partitioning database tables across multiple nodes of the processing engine, and is again dependent on the analysis application being used.

Many products now also employ web based interfaces, however ease of use and utility of this approach is limited by the absence of secure communication protocols and a session state within web interaction. According to Flohr [7], not all applications may be web enabled efficiently, such as function-intensive applications for specialised users. Also, it is often difficult to match visualisation capability available in a standard GUI environment with that available within a browser. However, web based OLAP could provide a cheap approach for delivering sophisticated visualisation for intra- and extra-nets in the future.

Our approach is most closely related to the Kensington Enterprise Data Mining System [4] which wraps analysis algorithms as Enterprise JavaBean components. Our system is however more extensible, and provides a designer with the option of including third party components. Our work differs from other approaches in that we are not targeting our framework 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 framework 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.

2.1 Requirements of High Performance Data Mining

Mining large data sets in parallel involves the following consideration:

- *A deep memory hierarchy* to maintain the data physically closer to the processing elements, in cache rather than main memory, for instance. The question of data migration also becomes important as selecting a data distribution across multiple processors becomes significant for improving application performance. The developer should have the option of utilising different cache layers, and also be able to partition data across the cache, main memory or disk on a single host machine. The framework should provide support for allowing the user to maintain data at various levels in the memory hierarchy, and provide primitives to position data at particular points in the processor pool.

  As large data sets are involved, processing should be maintained closer to the data, requiring minimal data migration. Support for data distribution through specialised data structures, such as multidimensional distributed arrays often encountered in scientific computing, may be used to determine different distribution patterns. Such data structures occur frequently in many data mining algorithms, such as weight matrices in neural networks and chromosome structures in genetic algorithms, for instance.
- *Load balancing across multiple processors* becomes particularly significant when there is a high diversity in the range of available machines, such as in a metacomputing environment. Migration of service needs to be performed transparently, and is aided by the presence of a concurrent process model (threads, for instance) in the underlying language or operating system.
- *Computational complexity* of data mining algorithms can be excessive, and being able to implement such algorithms in parallel is of particular benefit when an algorithm can be divided into a set of independent tasks. Neural networks are particularly suitable candidates, due to the explicit parallelism that exists within most neural learning rules based on local updates of processing units, and the neural architecture. Other analysis algorithms, such as decision trees may also be divided into separate tasks, which can be distributed across processors.
- *Quality* of results is particularly important for decision support, and the framework should support visualisation to detect anomalies, and enable measurement of output for calculating the quality of the output generated.

3 PaDDMAS Architecture

The PaDDMAS architecture is illustrated in figure 1 and addresses concerns in section 2.1.

The VPCE enables visual construction of an application by plugging together components held in two repositories. 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 components etc. Each component is either a Java or CORBA object, with its interface specified in XML, according to a standard data model applied to all components within PaDDMAS. Components are stored in the Component Repository using this format, and any binary data

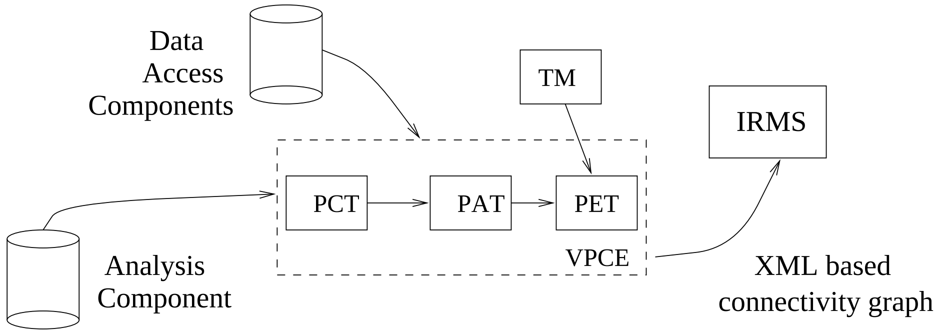

Fig. 1. A heterogeneous environment

associated with a component must also be tagged appropriately. 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 on-line 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 recorded of particular events, and persistent access of objects recorded within the VPCE. The TM enables objects to be saved with their current state, so they can be invoked at a later stage. We use PJama [9] for checkpointing the state of an object, so that it can be re-started for binding at a later stage. PJama enables a reachability tree to be established for each object that is to be made persistent, and all related objects connected to it being made consistent in the process, including threads and transitive persistence.

The XML based component model ensures uniformity across components, and helps to abstract component structure and implementation from component 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 which meet this interface. The Document Type Definition (DTD) identifying valid tags does not need to be placed with each interface, as it 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 the total number of inports and outports), check the suitability of a 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 licence 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 [11], 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, where as we encode interfaces to sub modules and performance models.

Connectivity to databases is provided through JDBC bridges, requiring the user to provide metadata 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 metamodels and metadata 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 licence 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.

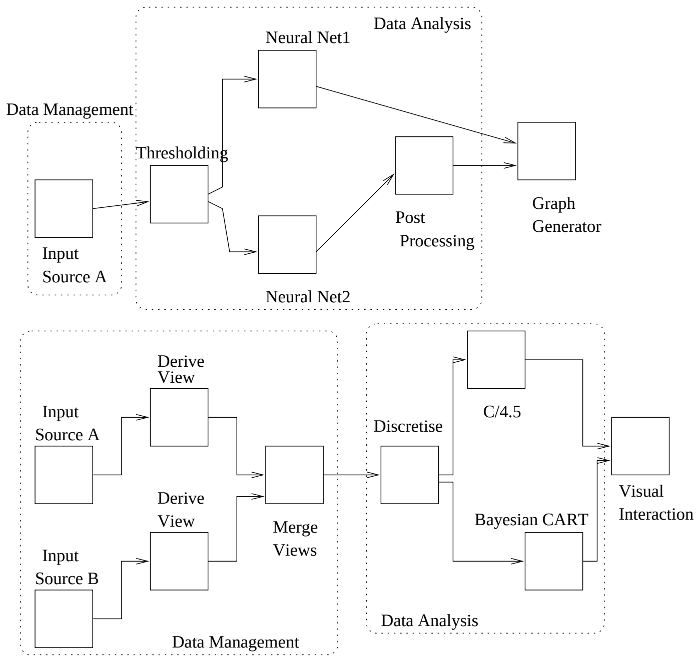

Fig. 2. Use Cases – Constructing Applications in PaDDMAS

4 A Neural Network Prototype

A neural network application was developed to demonstrate the PaDDMAS architecture. Figure 3 illustrates the GUI for a user, where output is plotted using a graph generator illustrated in figure 4.

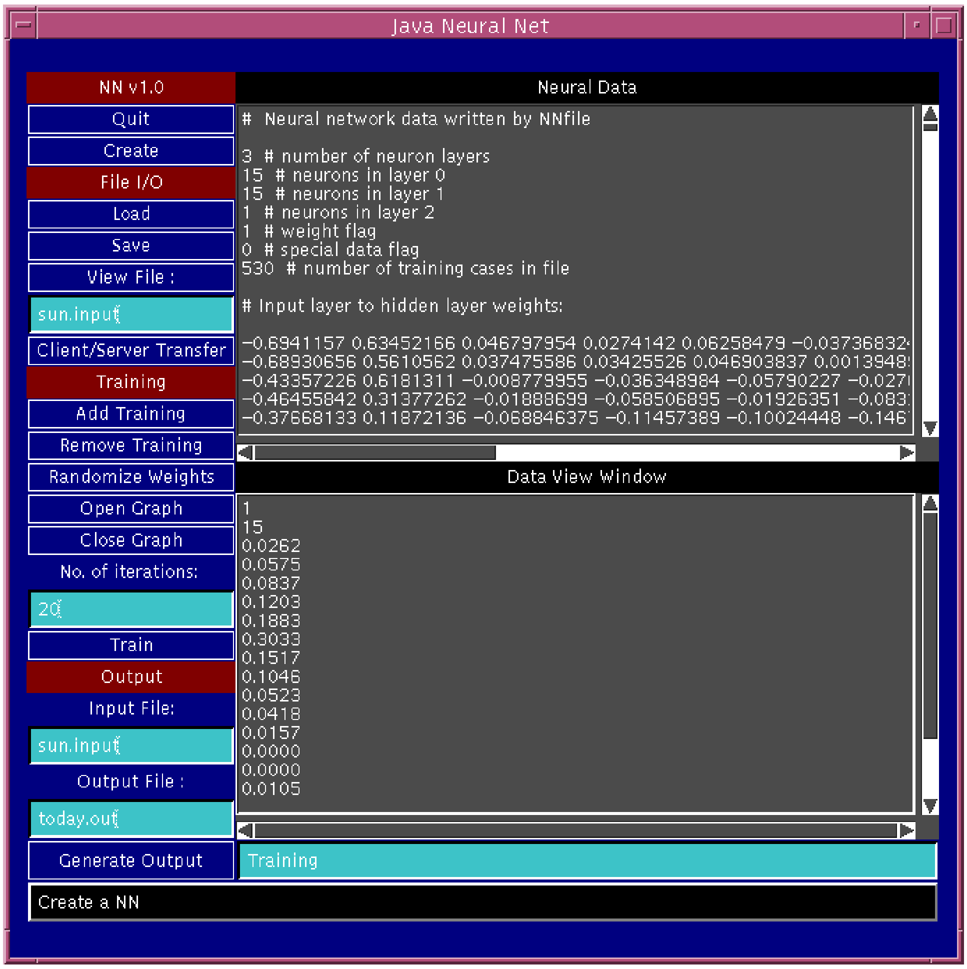

Fig. 3. GUI for neural network application

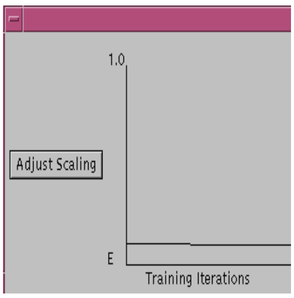

Fig. 4. Graph output

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 downloading an entire data set, or to send a filename of a data source to the remote host. Interactions between the GUI 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 immediately serialised and sent to the remote neural network for analysis.

The client GUI also offers a data browser for inspecting the contents of a remote file, and performing simple checks for outliers on data sets. Further data intergrity is not performed at present, and left to the user or the underlying DBMS to handle. The data source may be a flat file or a structured database. The use of 'View Derivation' components enables a part of a database to be extracted reducing the size of data transfer to the analysis tool. All data does not need to be replicated at the input of each component, as data selection is based on component functionality and component position.

5 Conclusion

We propose a framework for parallel and distributed data mining. The framework 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. The framework is currently under development as part of a larger multi-disciplinary Problem Solving Environment (M-PSE) project. We show how the framework may be used for a neural network application across a number of workstations as a *proof-of-concept* application. A developer may implement various data mining algorithms on the framework, 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 framework. Previous tools have stressed analysis components, but not data management functions.

The framework is quite general, and may be used in various applications, that can be divided into subtasks and are computationally expensive. Further experiments are underway in utilising this framework in a parallel agent environment, using the *Aglets* [10] mobile agent workbench.

References

1. R. Agrawal, A. Arning, T. Bollinger, M. Mehta, J. Shafer, and R. Srikant. The Quest Data Mining System. In *Proc. of the 2nd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Portland, Oregon.* 1996.
2. J. Albrecht and W. Lehner. On-Line Analytical Processing in Distributed Data Warehouses. In *IDEAS proceedings.* IEEE Comput. Soc. Press, July 1998.

3. P. Bradley, U. Fayyad, and O. Mangasarian. Data Mining: Overview and Optimization Opportunities. In *INFORMS: Journal of Computing*, 1998.
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. U. Flohr. OLAP by Web. *BYTE*, September 1997.
8. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989. ISBN: 0201157675.
9. M. Jordan and M. Atkinson. Orthogonal Persistence for Java. *Third International Workshop on Persistence and Java, Tiburon, CA, Sep 1-3*, 1998.
10. Danny B. Lange and Mitsuru Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.
11. UIUC National Center for Data Mining. Predictive Model Markup Language, 1999. See Web site at `http:///www.ncdm.uic.edu/`.
12. N. Pendse. OLAP Omnipresent. *BYTE*, February 1998.
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.