A Multi-Tier Architecture for High-Performance Data Mining

Ralf Rantzau **Holger Schwarz**
University of Stuttgart
Institute of Parallel and Distributed High-Performance Systems (IPVR)
{rantzau,schwarz}@informatik.uni-stuttgart.de

Abstract

Data mining has been recognised as an essential element of decision support, which has increasingly become a focus of the database industry. Like all computationally expensive data analysis applications, for example Online Analytical Processing (OLAP), performance is a key factor for usefulness and acceptance in business. In the course of the CRITIKAL[1] project (Client-Server Rule Induction Technology for Industrial Knowledge Acquisition from Large Databases), which is funded by the European Commission, several kinds of architectures for data mining were evaluated with a strong focus on high performance. Specifically, the data mining techniques association rule discovery and decision tree induction were implemented into a prototype. We present the architecture developed by the CRITIKAL consortium and compare it to alternative architectures.

1 Introduction

Today, data mining is a vital research area where innovations quickly find their way into industrial products. Many companies and other organisations have already gained beneficial insights into data which helped them to improve their business. High performance is a key factor for data mining systems in order to allow a sufficient exploration of data and to cope with the huge amounts of data that an organisation has accumulated over the years. High performance of a data mining system is not only a question of choosing the proper algorithms and efficient implementations of them. The architecture has also a remarkable impact on performance. This is particularly important for large business environments where an expensive IT infrastructure is installed that must be fully exploited. The architecture of a data mining system should reflect beneficial features of the infrastructure as much as possible, like multi-processor machines, but it also has to consider restrictions, like low bandwidth connections.

The remainder of this paper is organised as follows: In section 2 we give a short introduction to association rule discovery and decision tree induction. In the CRITIKAL project we concentrated on these data mining techniques. Section 3 presents a couple of architectural alternatives for a data mining system. The architecture and basic components of the CRITIKAL prototype are described in section 4. This also includes the explanation how association rule discovery and decision tree induction is performed in the CRITIKAL three-tier architecture. Finally, we give a conclusion of this paper in section 5.

[1]The consortium of CRITIKAL (Esprit project 22700) consists of: Attar Software Ltd. (UK), www.attar.com; GEHE AG (D); Lloyds TSB Group (UK); Parallel Application Centre, University of Southampton (UK), www.pac.soton.ac.uk; BWI, University of Stuttgart (D), www.planung.bwi.uni-stuttgart.de; IPVR, University of Stuttgart (D), www.informatik.uni-stuttgart.de/ipvr.

2 Data Mining Techniques

A variety of data mining techniques have been developed by researchers in the fields of artificial intelligence and statistics. In this section we focus on two popular techniques, which served as a basis for our development in the CRITIKAL project: association rule discovery and decision tree induction.

2.1 Association Rule Discovery

Association rule discovery has received great attention in the data mining community during the past few years because many real world business and engineering cases have been found where association rule discovery can successfully be applied. Current examples are credit card fraud detection, supermarket layout planning and medical diagnosis. Another reason is that association rules can serve as a basis for other data mining techniques like clustering [HKKM98]. Most papers on association rule focus on efficient algorithms and functional enhancements [AS94, CNFF96, SON95].

We present a short overview of the association rule model to give an idea of the problems that can be solved with this data mining method. Let $\mathcal{I}$ be a set of attributes called *items*. A subset $X \subseteq \mathcal{I}$ is called an *itemset*. Let the *database* $\mathcal{D} = \{T_1, \ldots, T_n\}$ be a set of *transactions*, where each transaction T_k, $k \in \{1, \ldots, n\}$, is an *itemset*. A transaction T *contains* an itemset X if $X \subseteq T$. Each itemset has a certain statistical significance called *support* or *frequency*. The support s of an itemset X is the fraction of transactions in the database $\mathcal{D}$ containing itemset X, i.e. $s(X) = |\{T \in \mathcal{D} \mid X \subseteq T\}|/|\mathcal{D}|$.

An *association rule* is an implication $X \Rightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. X is called the *antecedent* and Y is called the *consequent* of the rule. The rule $X \Rightarrow Y$ holds with *confidence* c if c is the fraction of transactions containing X that also contain Y, i.e. $c(X, Y) = s(X \cup Y)/s(X)$. The confidence denotes a rule's strength. A *confidence threshold* c_{min} is used to exclude rules that are not strong enough to be interesting. Accordingly, there is a *support threshold* s_{min} that excludes all rules whose number of transactions containing the union of the antecedent and the consequent is below a certain amount. Itemsets with minimum support are called *frequent* itemsets.

Age	Income	Risk
28	61k	low
22	46k	high
46	55k	low
25	48k	high
26	38k	high
34	46k	low

Table 1: An example training set.

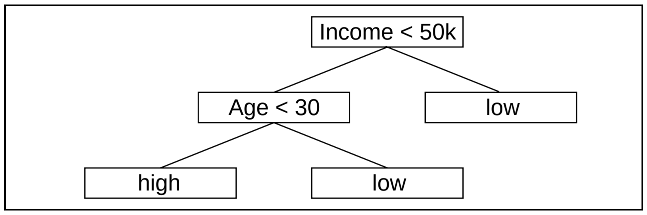

Figure 1: An example decision tree for classification.

The problem of finding association rules can be stated as follows. Given a set $\mathcal{I}$ of items, a database $\mathcal{D}$ of itemsets, a support threshold s_{min}, and a confidence threshold c_{min}, find all association rules $X \Rightarrow Y$ in $\mathcal{D}$ that have support $s(X \cup Y) \geq s_{min}$ and confidence $c(X, Y) \geq c_{min}$. This definition of the problem is sometimes called the *classic* or *boolean association rule problem.* More generalised problems consider item hierarchies [SA95], quantities [SA96], or propose further parameters to describe the interestingness of rules.

2.2 Decision Tree Induction

Classification is widely considered to be an important data mining task. Several classification models have been proposed in the past, among which decision trees are particularly suited for data mining [AIS93, SAM96]. Compared to other classification models the decision tree model is, like the association rule model, easy to interpret and it offers a good performance for classification.

First, we state the general classification problem. Let $\mathcal{A}$ be a set of attributes. Each attribute belongs to either an ordered (continuous attribute) or an unordered (categorical attribute) domain. We are given a *training set* $\mathcal{D}$ of records, where each record contains the same attributes together with values of their respective domains. One of the attributes, called the *classifying* attribute, indicates the class to which each record belongs. The problem of classification is to build a model of the classifying attribute based on the remaining attributes. Table 1 shows an example training set where each row corresponds to a borrower of a bank. The bank is interested in a model that helps to predict which future credit applicants are credit-worthy and which are not.

A decision tree is a class discriminator that recursively partitions the training set until each partition consists entirely or dominantly of records from one class. Each inner node of the tree contains a *split point* which is a test on one or more attributes and determines how the data is partitioned. Figure 1 shows a binary decision tree based on the example training set that can estimate the risk for further customers.

3 Architectures of Data Mining Systems

In this section we describe three alternative architectures for a data mining system. We define a set of basic components of a data mining system and explain where they are located in the different architectures. The evaluation of the approaches is based on several prerequisites for large scale data mining in an enterprise environment.

3.1 Prerequisites for Large Scale Data Mining

During the CRITIKAL project the consortium identified the following list of prerequisites for large scale data mining in an enterprise environment. Based on this list we can derive and evaluate architectures for highly optimised data mining systems.

- No limits on the data set sizes:
  Data mining is especially interesting for large enterprises which have huge datasets. Since they want to derive patterns from all of their data, the architecture should not limit the size of data sets that can be handled, for example to main memory capacity.

- Optimised performance for large data sets:
  A data mining system should incorporate optimisation strategies especially for large data sets in order to enable data mining with acceptable response times. The system architecture should enable a wide range of optimisation strategies like parallelism and caching.

- Flexibility for different data mining techniques:
  Users in an enterprise environment have different business goals in mind when they want to discover hidden knowledge in their data. Hence, the architecture should flexibly support various data mining techniques and algorithms like classification, clustering or association discovery.

- Support for multiple users and concurrency:
  In an enterprise environment a couple of users concurrently start data mining sessions on overlapping datasets. The data mining system should therefore support specific user priorities and user groups as well as the management of concurrent mining sessions. The system architecture should reflect these multi-user and multi-session capabilities.

- Full control of system resources:
  A data mining system is part of an enterprise IT infrastructure in which a couple of different applications run concurrently. Hence, the system needs full control of bandwidth and CPU cycles consumed by a user. This allows to start data mining activities in parallel to other applications without impairing these services. The architecture should particularly consider limited or low bandwidth connections to clients.

- Full control of access to data:
  In most enterprises data mining is based on data of a central data warehouse. Because this incorporates all important enterprise data, not all users should have unrestricted access. The data mining system has to offer the ability to restrict the access in addition to the data warehouse security system. This provides data mining specific access control. The architecture has to be designed in a way that prohibits users from bypassing this access control.

- Remote administration and maintenance:
  In a distributed enterprise environment there are many clients of the data mining system at different locations. Depending on the architecture the system might also incorporate several servers. Remote administration and maintenance is vital and should include installation and updating of software components.

This collection of requirements is a base for objective evaluation of system architectures that support large scale data mining.

3.2 Basic Components of a Data Mining System

The basic components of a data mining system are the user interface, data mining services, data access services and the data itself. The user interface allows the user to select and prepare data sets and apply data mining techniques to them. Formatting and presenting the results of a data mining session is also an important task of the user interface. Data mining services consist of all components of the system that process a special data mining algorithm, for example association rule discovery. These components access data through data access services. Access services can be optimised for special database management systems or can offer a standard interface like ODBC. The data itself constitute the fourth component of a data mining system. Since data mining is particularly interesting in large scale enterprise environments we assume the data to reside in a data warehouse.

These four basic components are present in all data mining systems. In the following we describe three different architectures where these components are appropriately distributed over the various tiers. For each approach we check with which of the prerequisites of section 3.1 it complies. Figure 2 outlines the architectures discussed below.

3.3 One-tier Architecture

The classical architecture of data mining systems is a one-tier architecture. Such a system is completely client based. Basically all data mining systems of the first generation are based on this architecture. The user has to select a small subset of data warehouse data and load it on the client in order to make it accessible to the data mining tool. This tool may offer several data mining techniques. The most obvious drawback of the one-tier approach is the size of the data set that can be mined and the speed of the mining process. This is often overcome by selecting a random sample from the data. A truly random (unbiased) sample is needed to ensure the accuracy of mined patterns, and even then patterns relating to small segments of the data can be lost. The data resides in raw files of the client's file system. Another disadvantage is the absence of a multi-user functionality. Each user has to define his own subset of the data warehouse and load it separately onto the client machine. Since each user runs his own client-based data mining software, there is no way for data mining specific access control and control of system resources. Optimisation of the data mining process is restricted to choosing more efficient implementations of the data mining techniques.

3.4 Two-tier Architecture

In a two-tier architecture the data mining tool completely resides on the client but there is no need to copy data to it in advance. The data mining application may choose to load parts of the data during different stages of the mining computation. There are several alternatives for running data mining algorithms in this architecture.

Download Approach

The connection to the data warehouse can be used to load data to the client and make it accessible for data mining. This can be done dynamically, therefore avoiding the problems with storing huge data sets on the client. Even if data is loaded in advance, this approach is superior compared to the one-tier architecture. The automatic loading of data by the client enables it to store pre-processed data depending on the user's needs. Pre-processed data may be of reduced size and stored in a way that supports the data mining algorithm. Hence, better performance of the discovery process and less space consumption is achieved.

Query Approach

For some data mining techniques it is possible to formulate parts of the algorithm in a query language like SQL. The client sends SQL statements to the data warehouse and uses the results for the data mining process. One advantage compared to the download approach is that only data which is really needed is sent to the client because filtering and aggregation is already carried out by the database system. Since parts of the application logic are formulated in SQL, query processing capabilities of the data warehouse system can be exploited [Att98].

Database Approach

In this approach the complete data mining algorithm is processed by the database system. This can be realized by stored procedures and user defined functions. Only the data mining results have to be sent to the client which is responsible for displaying them. The data mining process is able to exploit the efficient processing capabilities offered by the data warehouse.

The two-tier architecture has evident advantages over purely client-based data mining. It enables direct access to the data warehouse. No data extraction is necessary as a prerequisite for data mining. The two-tier architecture does not limit the size of a database that can be mined. New information can be discovered in the masses of data stored in the data warehouse. Additionally, the data mining process can take advantage of the query processing capabilities provided by special data warehousing hardware and software.

Besides the advantages of this approach some problems still remain. One problem is the limited access to the data warehouse system. Data warehouse systems are often in a "dark" site with restricted access. It is not allowed to install and configure applications on this system. Only the download approach and the query approach are applicable. Additionally, there is limited control over system resources. When all users directly access the data warehouse it is not possible to control the bandwidth and the CPU cycles each user needs for its data mining application. Many users concurrently access the data warehouse for data mining purposes. In the two-tier environment there is no way to control this access by data mining specific user priorities and user groups. The last drawback we want to mention here is the limited scope for optimisations. There are only two strategies to make the data mining process more efficient: the exploitation of the query processing capabilities of the

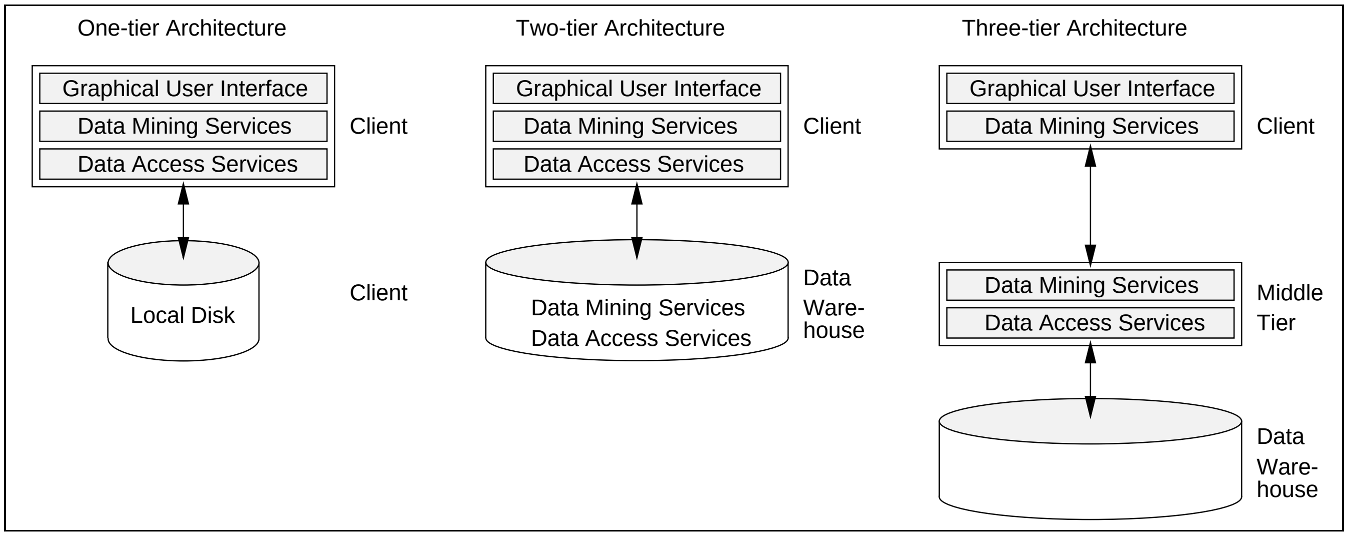

Figure 2: Architectures for data mining systems.

data warehouse and the enhancement of the data mining algorithm. There is limited scope for parallel algorithms and reuse of results by different clients.

3.5 Three-tier Architecture

A three-tier architecture addresses the problems remaining with a two-tier architecture. This is achieved by an additional layer holding the data access services and parts of the data mining services. Data mining services may also be present on the client. Which part of the data mining services should be client based depends on data mining techniques and algorithms.

The data mining process works as follows in this architecture. First, the user defines the parameters for data mining by the graphical user interface. The data mining services on the client perform some pre-processing prior to calling the data mining services on the middle tier. The first task on the middle-tier is authentication and authorisation of the users. Then the data mining services queue and execute the tasks of several clients and send back the results. These are used in the post-processing of the client, which computes the final outcome and presents it to the user. A client may start several data mining tasks in one session. Each of them includes a number of calls to the middle tier. Data mining services use the data access services on the middle tier in order to read from different types of data sources.

This three-tier approach has several advantages compared to the two-tier architecture. First, the data mining services can fully control bandwidth and CPU cycles for each user because there is a centralised service that manages users' tasks and resources. This enables the system to guarantee a maximum usage of system resources for data mining purposes. Second, the system can service users according to their priority and to their membership in user groups. This includes restricted access to data mining tables as well as user specific response behaviour. Third, a wide range of optimisation strategies can be realised. The tasks of the data mining services can be distributed over the client and the middle tier. The middle tier can exploit parallelism by parallel processing on the middle tier hardware and parallel connections to the database layer. Additionally, the data mining services can reuse the outcome of data mining sessions and pre-compute common intermediate results.

4 The CRITIKAL Prototype

This section describes the prototype that was developed within the CRITIKAL project as successive enhancements to a commercial two-tier decision tree induction package [Att98]. The prototype is based on a three-tier architecture as illustrated in figure 3. Data mining services are part of the client and of the middle tier. In section 4.2 and section 4.3 we describe how association rule discovery and decision tree induction are integrated in the CRITIKAL architecture. The data mining services also include algorithm independent general services which are summarised in the section below. Data access services in the CRITIKAL prototype offer access to data in files and in several relational DBMS. This access is provided by proprietary access modules with specific RDBMS optimisations as well as by a generic ODBC module. Further implementation details are given in section 4.4.

4.1 General Services

General services in the CRITIKAL prototype include all tasks that are independent of specific data mining techniques and algorithms used for their implementation. These tasks fall into three groups: connection and access services, remote administration services and the work management.

The connection and access services enables clients to connect to the middle tier. In this process a listener is involved that starts dedicated connection servers. Each connection server handles the communication with one client. There may be several connections with different clients at the same time. The connection servers are also responsible for access control. They check that a client is entitled to be logged on to the middle tier. Additionally, they check with the security mechanism of the data source which tables a user may access. Connection servers enable the user to interrupt and resume sessions in order to support long running mining activities.

The remote administration services are accessed by a remote administration client. This allows the middle tier software to be configured and controlled from a computer that is physically separated from the middle tier. This includes a system to monitor and shut down the middle tier. The remote administration client allows error messages and other

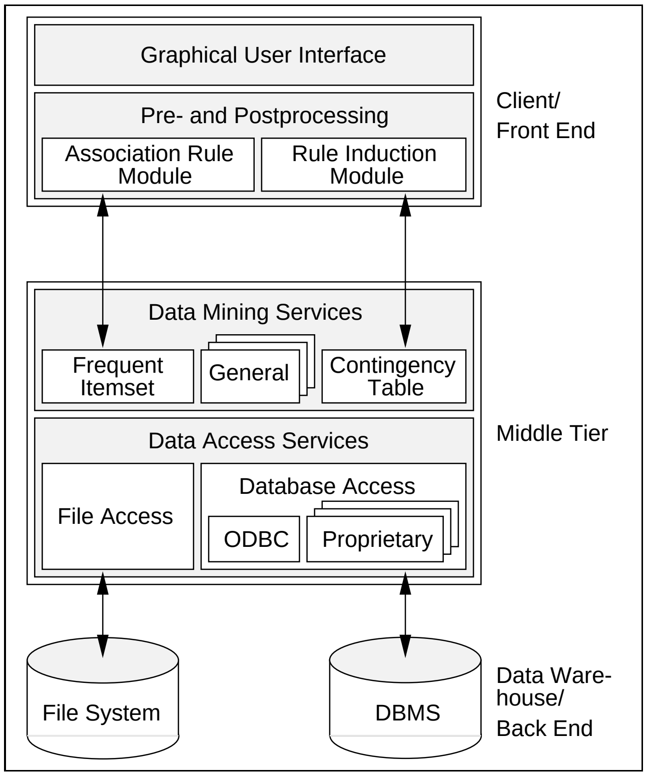

Figure 3: The three-tier architecture of the CRITIKAL prototype.

information to be displayed remotely. It also enables the administrators to control access to the middle tier for clients and to define the data sources.

The work manager is responsible for the execution of data mining tasks. It receives a work specification from the client which contains information on the data mining task and the required data sources. Based on this information it determines the engine that is able to perform the specified data mining task. The work specification then becomes part of a work package. A work package describes in detail what the client wants, how this information can be generated and where the raw data can be found. This work package is brought to a queueing system. Each work package is taken off the queue by the work manager and executed by one of several threads of execution. The results are packaged up and sent to the client.

The work management adds flexibility to the CRITIKAL prototype. The middle tier may be configured with multiple engines that allow different implementations of the same data mining task. For example, one engine could be optimised for SQL whereas another is optimised for in-memory execution. The work management also enables the control of resource consumption of the middle tier software. Therefore, the number of execution threads is controlled and the rate at which jobs are processed is limited. Parallel execution threads enable multiple parallel connections to the database.

4.2 Association Rule Discovery in a Three-tier Architecture

Mining for association rules involves two activities. The first one finds all frequent itemsets whose support is greater than a given minimum support threshold. The second one generates the desired rules from the frequent itemsets found in the first step. They are required to have a confidence that is greater than a given minimum confidence threshold.

The first step is computationally more expensive than the second one. This is because in all but the most trivial cases the amount of data to be processed is much larger in the first step (a database of transactions) than in the second one (frequent subsets of transactions). Furthermore, the first step often involves multiple passes over the database. This is the reason why most of the algorithms in the literature for mining association rules deal with the first step only. Our prototype reflects this functionality split. The middle tier is responsible for generating frequent itemsets which are sent to the client. The association rules are computed on the client based on the frequent itemsets and a minimum confidence parameter. If the user varies the minimum confidence the rules have to be recomputed, but no communication with the middle tier is necessary. Of course, a variation of the minimum support or surprise parameters requires the middle tier to run a new frequent itemset discovery. Frequent itemset discovery is very well located on the middle tier for several reasons. First, the middle tier is likely to be based on a more powerful machine compared to the client. Second, the middle tier is designed for high performance, incorporating parallelism. Third, it can exploit the query processing capabilities of the data warehouse.

Based on an extensive analysis of the most promising algorithms [Sch97], we decided to design and implement an algorithm using the ideas established in the Dynamic Itemset Counting algorithm [BMUT97]. This algorithm reduces the number of passes for frequent itemset discovery. Hence, the amount of data transferred to the middle tier is much smaller than for other algorithms. The data transfer can further be reduced by tokenisation and filtering. The idea of tokenisation is to map long multi-attribute identifiers on short integer values. This reduces the number of bytes that are sent to the middle tier for each identifier. Filters can be applied in two ways. First, items are selected according to the work specification. Second, all items that already turned out not to be frequent are discarded. In our implementation, the data partitions are processed by parallel threads that insert new frequent itemsets into a shared hash-tree data structure.

The association rule discovery facility integrated into the prototype has several features which are considered important by the end user project partners of CRITIKAL. Among them are the ability to include and exclude certain items prior to the association rule discovery. Another feature in the CRITIKAL prototype design is the support of item hierarchies. For example, many organisations maintain a hierarchy of product groups in their data warehouse. This allows for more general rules like "beverages" $\Rightarrow$ "baby care" in addition to specific rules like "beer" $\Rightarrow$ "nappies". Both features not only help the user to concentrate on interesting aspects of the data, but they also reduce the amount of data to be processed, yielding faster response times.

4.3 Decision Tree Induction in a Three-tier Architecture

The technique for building decision trees uses statistical information on data held in a single data mining table. The user specifies one field of the table to be the outcome. The data of the table is then classified with respect to this outcome by applying binary splits to the decision tree.

The algorithm for decision tree induction implemented in the CRITIKAL prototype [Att98] requires information on the data in the form of *contingency tables* (sometimes called *cross tabulations* or *pivot tables*). A contingency table is a two-dimensional matrix. One axis represents the values of an attribute and the other axis represents the values of the outcome. The cells of the matrix contain counters for the occurrences of attribute/outcome combinations. For example, an outcome attribute could be a text field with the three values "yes", "no" and "maybe".

Once the user has specified the table to be used for decision tree induction, the client requests statistical information to describe the attributes within that table. This information allows users to decide which attributes they will use for building decision trees. Decision tree construction involves repeated requests for contingency tables. A contingency table is requested per attribute per node of a tree.

Decision tree induction can also be decomposed into two steps. In the first step contingency tables are computed based on raw data. This task is performed on the middle tier. The contingency tables are created from the subset of data that is represented by the node to be split. In a second step the client applies statistics based on a high level algorithm to these contingency tables in order to induce a split of the node. The result of the split are two child nodes which represent two disjoint subsets of the parent node. For each of them a filter can be defined. The high level information travelling between the client and the middle tier is designed to satisfy any bandwidth constraints between these two tiers.

It can be seen that determining a split point in a decision tree consists of two parts. One is defining a subset of the raw data using a filter. The other part is aggregation in order to create contingency tables. It should be noted that the order of these two operations is an implementation decision. It is possible to build suitably indexed contingency tables for the full data set and then extract subsets for a given node as defined by a filter. The CRITIKAL prototype uses SQL for aggregation but constructs the contingency tables on the middle tier as the data is streamed across from the database.

4.4 Implementation Details

Part of the CRITIKAL prototype software was implemented in Java because of its well-known merits like readability and portability, thereby facilitating code development and testing. Other parts of the CRITIKAL software required special optimisations and are implemented in C/C++. These programming languages were also necessary to access interfaces like ODBC and other database specific APIs.

The client is able to run on any of Microsoft's Windows operating systems, while the middle tier software is currently restricted to Windows NT. The database access services incorporate optimised code for several database systems like Oracle8 or NCR's Teradata. Any database system implementing ODBC can be accessed.

5 Conclusion

This paper presented a powerful and flexible three-tier architecture for data mining that was developed as part of the CRITIKAL project. The focus of this architecture is to allow a high-performance implementation of data mining techniques like association rule discovery and decision tree induction. The prototype is able to mine patterns in very large datasets that may reside in databases or files.

Some of the ongoing work in CRITIKAL is an optimised multi-tier support of data preparation prior to the data mining tasks. This reflects today's need for data mining systems that offer high performance across every phase of the entire knowledge discovery process.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database Mining: A Performance Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, December 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases, Santiago, Chile*, pages 487–499, September 1994.

[Att98] Attar Software. *XpertRule Profiler Reference Manual*, 1996–1998.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proceedings of the ACM SIGMOD International Conference, Tucson, Arizona, USA*, pages 255–264, May 1997.

[CNFF96] David Wai-Lok Cheung, Vincent T. Ng, Ada W. Fu, and Yongjian Fu. Efficient Mining of Association Rules in Distributed Databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):911–922, December 1996.

[HKKM98] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Hypergraph Based Clustering in High-Dimensional Data Sets: A summary of Results. *Bulletin of the Technical Committee on Data Engineering*, 21(1):15–22, March 1998.

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. Mining Generalized Association Rules. In *Proceedings of the 21st International Conference on Very Large Databases, Zürich, Switzerland*, pages 407–419, September 1995.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In *Proceedings of the ACM SIGMOD International Conference, Montreal, Canada*, pages 1–12, June 1996.

[SAM96] John Shafer, Rakesh Agrawal, and Manish Mehta. SPRINT: A Scalable Parallel Classifier for Data Mining. In *Proceedings of the 22nd International Conference on Very Large Databases, Bombay, India*, pages 544–555, September 1996.

[Sch97] Holger Schwarz. Survey of State-of-Art Association Rules Discovery. Deliverable No. D4.1, European Commission, ESPRIT Project No. 22700, Brussels, Belgium, May 1997.

[SON95] Ashok Savasere, Edward Omiencinski, and Shamkant Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the 21st International Conference on Very Large Databases, Zürich, Switzerland*, pages 432–444, September 1995.