An Architecture for Distributed Enterprise Data Mining

J. Chattratichat, J. Darlington, Y. Guo, S. Hedvall, M. Köhler, and J. Syed

Data Mining Group
Imperial College Parallel Computing Centre
180 Queen's Gate, London SW7 2BZ, UK
{jc8, jd, yg, dsh1, mk, jas5}@doc.ic.ac.uk

Abstract. The requirements for data mining systems for large organisations and enterprises range from logical and physical distribution of large data and heterogeneous computational resources to the general need for high performance at a level that is sufficient for interactive work. This work categorises the requirements and describes the *Kensington* software architecture that addresses these demands. The system is capable of transparently supporting parallel computation at two levels, and we describe a configuration for trans-atlantic distributed parallel data mining that was demonstrated at the recent Supercomputing conference.

1 Introduction

Data Mining, or *Knowledge Discovery in Databases* is concerned with extracting useful and new information from data, and provides the basis for leveraging the investments in data assets. It combines the fields of databases and data warehousing with algorithms from machine learning and methods from statistics to gain insight in hidden structures within the data. In order to apply the knowledge from the Data Mining process, the results need to be analysed, often with the help of visualisation tools, as well as integrated into the business process.

Data mining systems for enterprises and large organisations have to overcome unique challenges. They need to combine access to diverse and distributed data sources with the large computational power required for many mining tasks. The data mining process, as perceived by the analysts, knowledge workers and end-users of the discovered knowledge, is an interactive one that functions best when a high degree of interactivity is available. The analyses are usually refined during several iterations through the cycle of data selection, pre-processing, model building and model analysis. The best results are usually achieved by combining models from different techniques, which calls for a wide variety of integrated tools within the system, as well as openness for future extensions.

In large organisations, data from numerous sources needs to be accessed and combined to provide comprehensive analyses, and work groups of analysts require access to the same data and results. For this purpose the existing networking infrastructure, typically based on Internet technology, is to be re-used. Confidentiality becomes a key issue, and the system architecture needs to provide security features at all levels of access. The different needs of enterprises require that a system offers a wide range of configuration options, so that it is possible to scale applications from a few client workstations to high-performance server machines.

In this article we will discuss the implications of the above requirements, focusing on the Kensington solution that employs Internet and distributed component technologies for deployment on high-performance servers such as distributed memory and shared-memory parallel machines. The next section will discuss the key functional requirements that have been outlined so far. The following chapter outlines the design and implementation of the

Kensington enterprise data mining system, in particular Java- and CORBA-based networking and component technology. We then describe a scenario for distributed data mining that was demonstrated at SuperComputing'98 as part of the award-winning Terabyte Challenge. The final section concludes and outlines future trends in the field.

2 Enterprise Data Mining Requirements

Data mining system architectures for enterprises have to meet a range of demands from the field of data analysis and the additional needs that arise when handling large amounts of data inside an organisation. Modern data mining applications are expected to provide a high degree of integration while retaining flexibility. In this way they can efficiently support different types of analyses over the organisation's data. Data mining is understood to be an iterative process for the analyst [FPSS96], especially in the initial exploratory phases of the analytical task. Therefore, a high degree of interactivity is required, often combined with the need for visualisation of the data and the analytical results.

The field of data mining is developing rapidly, and the methods applied in a tool today may be superseded by more advanced algorithms in the near future. Furthermore, the convergence with statistical methods has only just started, and will grow in pace over the next few years. The need for enhancement of the existing tool set has to be reflected by a software architecture that enables the straightforward integration of new analytical components. In a similar vein, the results from the analytical functions need to be presented in portable formats, as most analysts will want to use different specialist packages to further refine or report the results.

In large organisations, the amount and the distribution of the data become an additional challenge. The size of the data may make it impractical to move it between sites for individual analytical tasks. Instead, data mining operations are required to execute "close to the database". In the absence of dedicated support for data mining and other analytical algorithms in the database management systems, this can be achieved by setting up high-performance servers in close proximity to the databases. The overall data mining system will then have to manage the distributed execution of the analytical tasks and the combination of the partial results into a meaningful total. Also, this approach can sometimes benefit from the improved generalisation power that often occurs when combining analytical models [CS96].

Three kinds of scalability requirements arise in enterprise environments:

1. data sets can be *very large*
2. there may be *many sites* on which data is accumulated
3. there may exist *many users* who need access to the data and the analytical results

An enterprise data mining architecture should be flexible enough to scale well in all these cases. This will require access to high-performance analytical servers (case 1), the ability to distribute the application (case 2) and the capability to provide multiple access points (case 3).

2.1 Vertical structure of Enterprise Data Mining

Any architecture that fulfills these requirements is likely to be a *three-tier client/server architecture*. This structure encapsulates the main application functions inside an application server that can be accessed from clients on the network. The application server is connected to back-end servers that perform tasks on dedicated high-performance machines. The functional specification for the three tiers includes:

- Client
  - interactive creation of data mining tasks
  - visualisation of data and models
  - sample the data, for improved reaction time
- Application server
  - authentication and user session handling
  - persistent storage and access control for users' objects (data, models, tasks)
  - task execution control and coordination
  - data management and pre-processing
- Third-tier servers
  - RDBMS
  - parallel high-performance data mining service

2.2 Horizontal structure of Enterprise Data Mining

Software Component Architectures have been developed to facilitate the creation of flexible distributed systems. Software components are modular building blocks that developers can combine into ensembles. The implementation of the components is encapsulated and exposed only through an application programming interface (API). New functions can be integrated into the application by embedding them into new components and attaching them to the existing system. The flexibility and extensibility required by enterprise data mining environments makes the use of software components mandatory.

A variety of component architectures are available, most notably the Java Enterprise Architecture, CORBA and DCOM. The main differences between these proponents are the range of hardware and operating systems supported, and the options for component deployment.

2.3 The role of high-performance systems

High-performance machines are most likely to be employed in the third tier. Parallel systems have been shown to provide appreciable speed-up for data mining tasks [CDG$^+$97], and they have been in use for parallel database machines for some time.

3 Kensington Data Mining system

The Kensington Enterprise Data Mining system (see Figure 1) has been designed using the Enterprise JavaBeans component architecture and has been implemented in Java. It integrates parallel data mining functions that have been written in C and MPI via a CORBA interface. Databases anywhere on the Internet can be accessed via a JDBC connection. The client is built as a highly interactive Java application using JavaBeans.

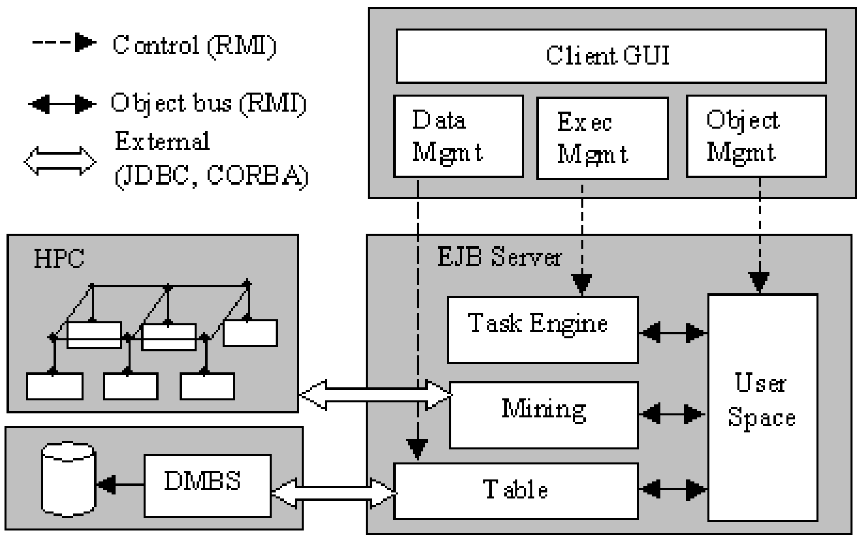


Fig. 1. Kensington Client and Server Components

3.1 Application server

Design The application logic consists of four Enterprise JavaBeans (EJB) classes, which execute in an EJB server environment. The EJB server is a generic implementation of Sun's EJB 1.0 standard, with additional capabilities for handling user login and authentication. The EJB server provides container functions for all EJB classes according to deployment options that are set as part of the application design. Each EJB class provides services for one aspect of the application, namely user object management, task execution, mining component management and database access and data storage. The EJB server provides *container-based* persistence, but allows EJBeans to manage their persistence themselves (*bean-managed* persistence).

A key advantage of the EJB architecture is the ability to run multiple cooperating servers at different sites. The Kensington EJB server provides two global services for such a set-up: user authentication and secure communication between EJB server incarnations. EJBean classes can then be flexibly deployed across these servers. An example of a distributed configuration is described in section 4. The flexible deployment options enable the use of multiple points of entry for user sessions, thereby providing scalability in the number of supported clients.

Implementation The four EJB classes that constitute the application server functionality are:

UserSpace EJB manages persistent objects for the user; the functionality of this EJBean includes a directory structure for the objects with user and group access control. The UserSpace EJB manages the persistence of user's data mining tasks and results.

TaskEngine EJB is the contact point for the client to control the execution of data mining tasks, including preprocessing, model generation and model evaluation; the execution can be distributed between different sites; results are returned to the client.

Mining EJB handles the interface with the analytical components; high-performance parallel components are accessed via a range of protocols, including CORBA and JNI.

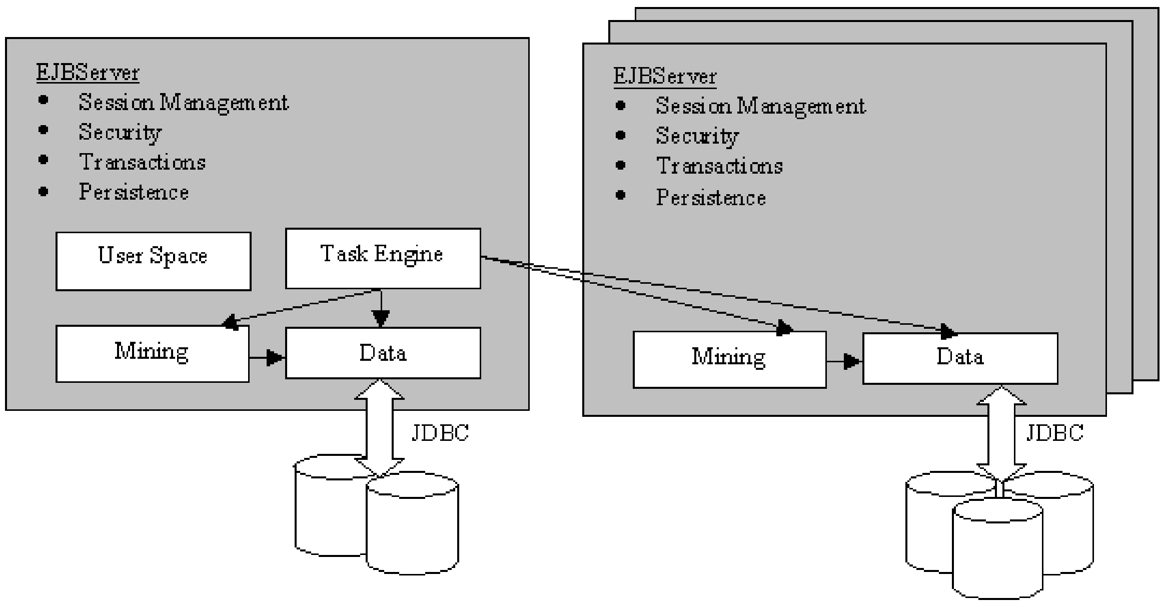


Fig. 2. A Scenario with Distributed Kensington EJB Servers

Table EJB is responsible for the import and storage of data bases, via JDBC; it supports browsing the external databases from the client. As of this writing, the Table EJB utilises the container-based persistence for the user's datasets, but a bean-managed version is under construction for improved performance.

The EJB server implemented for Kensington provides all the container functions specified by the EJB-1.0 standard, functions for container-managed persistence from the EJB-2.0 standard and additional services for user session management. The latter have been designed to provide system-wide authentication for multiple cooperating EJB servers that can be distributed. In this scenario a common JNDI service (Java Networking and Directory Interface) provides global naming services. In addition, the EJB server facilitates the sharing, or *pooling* of critical resources such as network connections to databases and CORBA connections to external components.

All EJB classes can be flexibly configured within any number of EJB servers under a common JNDI umbrella. This implies that *a single application server* configuration can contain

- multiple points of login for user sessions
- multiple sites for storing data
- multiple sites for mining data

A single user's or group's data can be distributed across all the sites that contain a Table EJBean, and the user can initiate a mining task from any of the login sites to execute mining functions on any of the EJB server sites that contain a Mining EJBean. Of course, all EJBean classes can also be configured to run within a single EJB server.

The services provided by the EJB classes are exposed to the client via control interfaces (cf. the *Control* connections in Fig. 1), which are realised in RMI over secure sockets (SSL).

All interaction between EJB classes uses the same interfaces, thus allowing for transparently distributed instances of the EJBeans. The UserSpace EJBean serves as repository for persistent objects which interacts with the other EJB classes via an interface that functions as an *object bus*.

External interfaces have been defined for the Table and Mining EJBean. They are realised via JDBC for database access and CORBA, respectively, thus allowing maximal portability and adaptability.

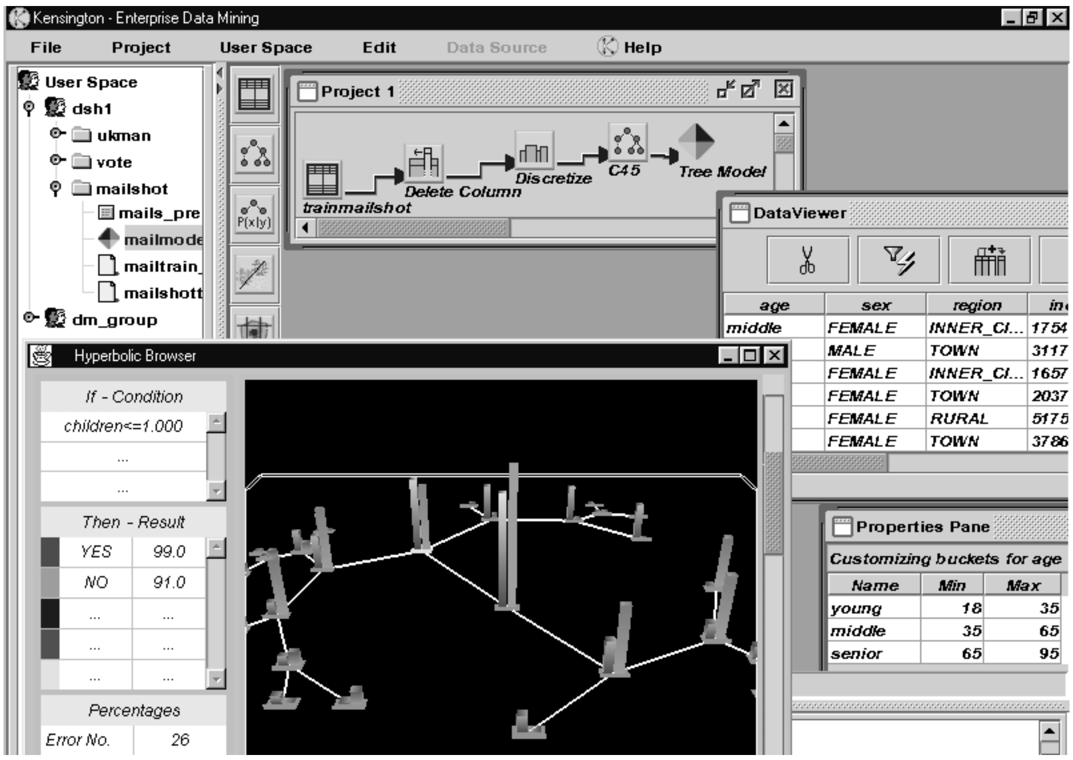
3.2 Client

Design The main purpose of the client is to provide an interactive GUI with two main capabilities: interactive visual programming of data mining tasks, and three-dimensional visualisation of data and analytical models. The client/server connection makes conservative assumptions about the performance of network connections and tries to reduce the amount of data transferred between the two tiers. Kensington uses a sampling method to provide the user with enough data for interactive pre-processing and preliminary analysis without overloading either the network, or the memory of the client machine. However, the full data set can be accessed from the client on demand. The data mining tasks programmed by the users generally execute over the full data set, thus providing the most comprehensive analysis possible.

The client offers a data browser for inspecting the contents of remote databases, prior to import into the Kensington server. The user only needs read access permissions to the database and tables in question. Once entered, the details of the connection are stored as *database bookmark* for future use. It is possible to reduce the data imported from the source data table by interactively formulating selection queries. Upon completion of the selection, the Table EJBean of the attached EJB server imports the data into the Kensington EJB server.

Implementation The client is a Java application that is connected to one or more EJB servers via the RMI protocol. The graphical user interface has been implemented with JavaBeans to allow flexible extension of the visual programming framework. Every component in the user's data mining task is represented by a JavaBean element. The data mining process consists of data sources (tables), pre-processing operations, analytical functions (data mining and statistical methods) and application nodes (model assessors and predictor nodes), cf. Fig. 3. When the user connects task nodes, the client checks the metadata definitions of the data set to check the semantic conditions of the operations.

The GUI on the client interacts with the EJB server via three components. The *Data Manager* is responsible for accessing the services of the Table EJBean and provides the elementary functions for browsing remote data bases. The *Exec Manager* is invoked on the client when the user requests the execution of a data mining task. The *Object Manager* provides a view of the user's objects held on the Kensington server. These components reflect the functions of the EJBeans on the server, while also encapsulate caching and network connection management.



age	sex	region	in
middle	FEMALE	INNER_CI...	1754
	MALE	TOWN	3117
	FEMALE	INNER_CI...	1657
	FEMALE	TOWN	2037
	FEMALE	RURAL	5175
	FEMALE	TOWN	3786

Customizing buckets for age

Name	Min	Max
young	18	35
middle	35	65
senior	65	95

If - Condition
children<=1.000
...
...

Then - Result		
	YES	99.0
	NO	91.0
	...	...
	...	...
	...	...

Percentages	
Error No.	26

Fig. 3. The Kensington Client GUI

3.3 Third-tier components: high-performance mining

Design The analytical components of Kensington are parallel programs that have been encapsulated as objects via the Mining EJB class. This class controls data conversion and parameter passing to each of the algorithms. Any component that executes on dedicated separate systems, such as parallel machines, can be integrated via CORBA, RMI (Java Remote Method Invocation) or JNI (Java Native Interface). The interfaces allow the introduction of further mining components with very little overhead, providing flexibility similar to that in the client. The predictive models that are computed by the data mining modules are internally represented in the emerging PMML (Predictive Model Markup Language) standard, which is developed within the Terabyte Challenge project, cf. section 4.

Implementation The current range of analytical functions includes classification (C4.5 [Qui93], backpropagation neural networks [Pen97] and naive Bayesian methods), clustering (self-organising maps [Rue97] and k-means methods), association rule discovery (a-priori method [Toi96]). In addition, an interface for calling functions in an S-plus statistical server has been created. This allows feeding any data from Kensington to any function available in S-plus, with results reported on the client.

A CORBA interface was designed to handle the transfer of data and control between the Mining EJBean and the external mining components. This allows the immediate integration of CORBA-capable systems. Furthermore, any data mining application can be *wrapped* by an object layer that provides the interface methods and translates the data and control parameters [CDG+98]. The open interface of the Mining EJBean allows quick migration of

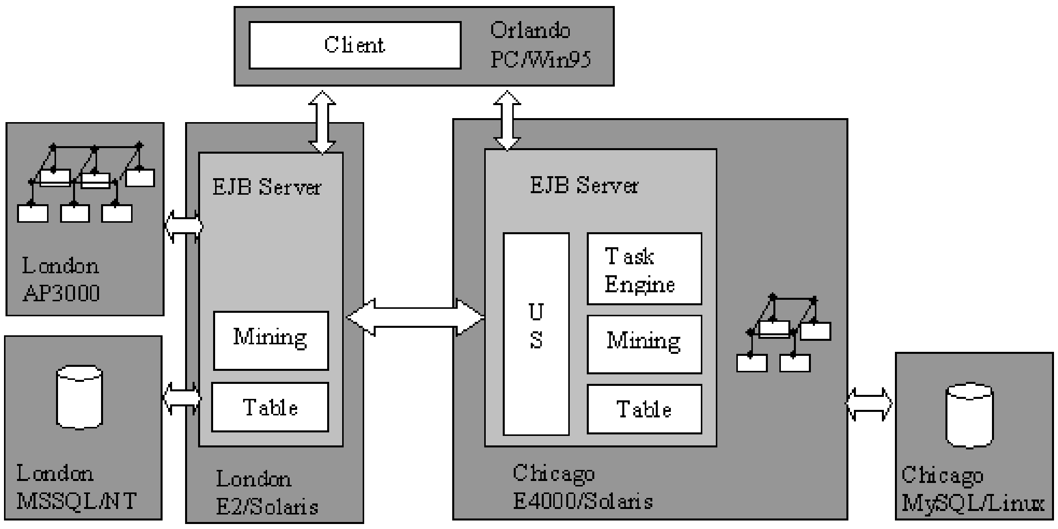


Fig. 4. The Trans-Atlantic Kensington Configuration for Supercomputing'98

separately programmed components into Kensington, thus providing rapid access to the other features of the system for the data mining developer.

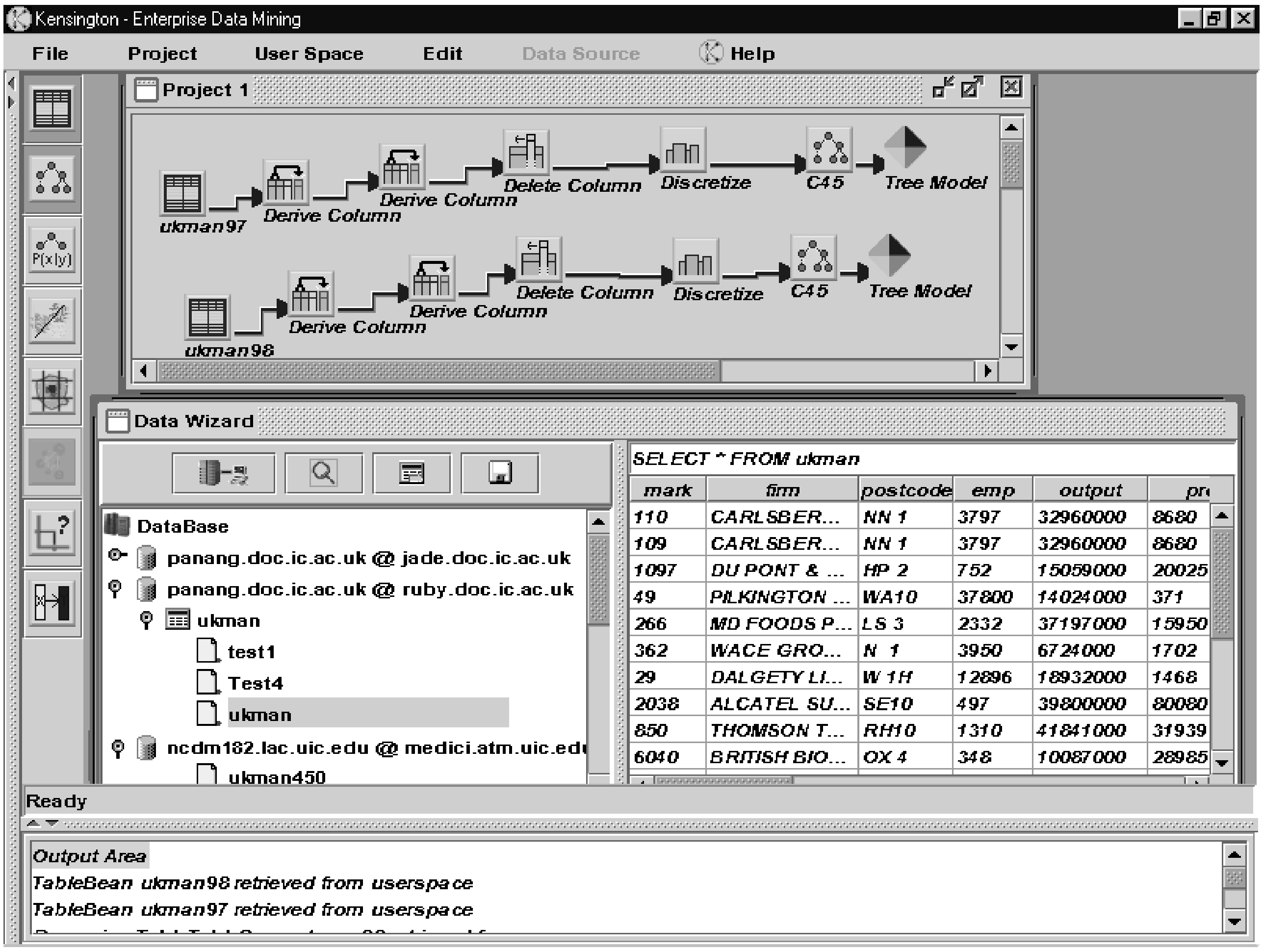
4 Distributed Parallel Data Mining for the Terabyte Challenge

The Terabyte Challenge project explores the technology to tackle data analysis projects of the largest imaginable size. The scope of the project encompasses distributed and parallel computing software technology. The Kensington system was demonstrated in a distributed setting with two host sites and a client running at the SuperComputing'98 site. Each host site consisted of a machine that ran an instance of the Kensington EJBserver. One of the sites, at the University of Illinois in Chicago, was equipped with a full complement of EJBeans and acted as the coordinating site for the whole installation. The second site, at Imperial College, ran an EJB server with only Table and Mining EJBeans (see Figure 4). The network connections were standard IP connections over Ethernet.

The client was configured with access to both server sites, which enabled the users to access and mine data concurrently across the network. It is important to note that the coordinating EJB server, and in particular the TaskEngine EJBean, act as arbitrator for control purposes only: the data that was distributed over databases at the host sites was loaded and mined locally.

In a typical client session (Fig. 5), the user browses remote databases until the required data has been identified. The browsing and loading is handled by a Table EJBean that is associated with each database at request time[1]. The user then proceeds to edit a data mining task on the client. A mining task starts with a data source, may contain some preprocessing operations and terminates with an analytical function. Separate tasks can be defined for any data source that has been loaded, including data loaded from different databases and

[1] For the SuperComputing demonstration, a static mapping from databases to EJB servers was employed. Dynamic mappings are possible as well.



SELECT * FROM ukman

mark	firm	postcode	emp	output	pr
110	CARLSBER...	NN 1	3797	32960000	8680
109	CARLSBER...	NN 1	3797	32960000	8680
1097	DU PONT & ...	HP 2	752	15059000	20025
49	PILKINGTON ...	WA10	37800	14024000	371
266	MD FOODS P...	LS 3	2332	37197000	15950
362	WACE GRO...	N 1	3950	6724000	1702
29	DALGETY LI...	W 1H	12896	18932000	1468
2038	ALCATEL SU...	SE10	497	39800000	80080
850	THOMSON T...	RH10	1310	41841000	31939
6040	BRITISH BIO...	OX 4	348	10087000	28985

```
Output Area
TableBean ukman98 retrieved from userspace
TableBean ukman97 retrieved from userspace
```

Fig. 5. Mining Data simultaneously in the UK and US

held at different EJB server sites. Upon request, tasks are handed to the TaskEngine on the coordinating EJB server. It selects executing Mining EJBeans according to a resource allocation policy[2] and initiates the computations. Results are sent back to the coordinating TaskEngine which forwards them to the client.

This configuration effectively provides parallelism at two levels. The mining tasks are parallel components that run on two different multi-processor architectures (distributed and shared memory). The overall mining task consists of mulitple mining activities which are scheduled to happen concurrently on the two different machines.

5 Conclusion and outlook

The Kensington system architecture has been designed to address the demands of enterprise data mining. Its flexibility results from the use of distributed components, built with Java

[2] The default policy is to execute mining operations in the EJB server that contains the data.

Enterprise Beans technology. The EJB server developed within the project provides global binding, authentication and security services. The deployment of components for the application can be customised to match the computational and data resources at hand. The Kensington server has been built entirely in Java, which provides the additional benefit of platform independence.

The interfaces between components and to the environment are open to facilitate the integration of new components at all levels, including analytical functions (data mining and statistics), data pre-processing operations and visualisation and report JavaBeans. A CORBA interface provides connections to high-performance data mining services, and an interface to the S-plus package integrates numerous statistical functions.

All network connections can operate over standard TCP/IP Internet or Intranets or virtual private networks, and the system incorporates connection management and secure communication.

Data mining architectures have rapidly evolved from stand-alone applications over flat files to systems with integrated data management. Systems such as Kensington belong to the class of third generation architectures which are characterised by distribution of data and computation.

Emerging technologies for *dynamic networking*, such as Sun's Jini, will provide a layer above the current interfaces for distributed components. The main addition is the capablity to dynamically add computational resources to a network, thus enabling truly *networked applications*. These capabilities, combined with the mobility of code that has been pioneered by the Java language, will allow future enterprise data mining systems to deliver comparatively compact analytical functions to any database site on the network. The full performance potential will be leveraged when the integration of Java into the database management systems allows the mobile mining components to execute within the databases themselves.

References

[CDG$^+$97] J. Chattratichat, J. Darlington, M. Ghanem, Y. Guo, H. Hüning, M. Köhler, J. Sutiwaraphun, H. W. To, and D. Yang. Large scale data mining: Challenges and responses. In *Proceedings of Third International Conference on Knowledge Discovery and Data Mining*, pages 143–146, 1997.

[CDG$^+$98] J. Chattratichat, J. Darlington, Y. Guo, S. Hedvall, M. Köhler, A. Saleem, J. Sutiwaraphun, and D. Yang. A software architecture for deploying high-performance solutions on the internet. In *High-Performance Computing and Networking*, 1998.

[CS96] P. K. Chan and S. J. Stolfo. Sharing learned models among remote database partitions by local meta-learning. In E. Simoudis, J. Han, and U. Fayyad, editors, *The Second International Conference on Knowledge Discovery and Data Mining*, pages 2–7. AAAI Press, 1996.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetesky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

[Pen97] Wanida Pensuwon. Parallel neural networks. Msc. thesis, Imperial College, University of London, 1997.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman Publishers, 1993.

[Rue97] Stefan Rueger. Parallel self-organising maps. In *Proceedings of the Seventh Parallel Computing Workshop, Australian National University, Canberra, September 25–26*, 1997.

[Toi96] Hannu Toivonen. *Discovery of Frequent Patterns in Large Data Collections.* PhD thesis, Department of Computer Science, University of Finland, 1996.