RC23362 (W0410-026) October 4, 2004
Mathematics

IBM Research Report

A Grid-Based Approach for Enterprise-Scale Data Mining

Ramesh Natarajan, Radu Sion*, Chid Apte, Inderpal S. Narang*
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

*IBM Reearch Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

A Grid-based Approach for Enterprise-Scale Data Mining

Ramesh Natarajan, Radu Sion, Chid Apte, and Inderpal S. Narang

Abstract— **We describe a grid-based approach for enterprise-scale data mining that leverages database technology for I/O parallelism, and on-demand compute servers for compute parallelism in the statistical computations. By enterprise-scale, we mean the highly-automated use of data mining in vertical business applications, where the data is stored on one or more relational database systems, and where a distributed architecture comprising of high-performance compute servers or a network of low-cost, commodity processors is used to improve application performance and provide the application deployment flexibility for overall workload management.**

The approach relies on an algorithmic decomposition of the data mining kernel on the data and compute grids, which makes it possible to exploit the parallelism on the respective grids in a simple way, while minimizing the data transfer between them. The overall approach is compatible with existing database standards for data mining task specification and results reporting, and hence external applications using these standards-based interfaces do not have to be modified in order to realize the benefits of this grid-based approach.

Index Terms—**Data mining, Grid computing, Predictive modeling, Parallel databases.**

Data-mining technologies that automate the generation and application of statistical models from data are of interest in a variety of applications cutting across industry sectors. These applications include, for example, customer relationship management (Retail, Banking and Finance, Telecom), fraud detection (Banking and Finance, Telecom), lead generation for marketing and sales (Insurance, Retail), clinical data analysis (Health Care), risk modeling and management (Banking and Finance, Insurance), process modeling and quality control (Manufacturing), genomic data and micro-array analysis (Life Sciences), yield management and logistics (Travel and Transportation), text classification and categorization (cross-Industry) among others. Further details, including specific case-studies for some of these applications can be found in [1]. In general, the underlying statistical analysis (for predictive modeling, forecasting, optimization, or exploratory multivariate data analysis) in these business applications is very computationally intensive.

Our grid-based approach is motivated by some of the requirements and challenges for developing enterprise-scale data mining solutions for these applications. By enterprise-scale, we mean the use of data mining as a tightly integrated component in the workflow of vertical business applications, with the relevant data being stored on highly-available, secure, commercial relational database systems. These two aspects of the present problem differentiate the present work from other data-intensive problems studied in the data grid and scientific computing literature (e.g., [2], [3]).

The outline of the remainder of the paper is as follows. Section I considers the current state of database-integrated data mining kernels, and the need for a future evolution towards a grid-based mining architecture. Section II describes the rationale for the algorithmic decomposition of data mining kernels between the data and compute grids, along with a review of related approaches in the literature. Section III illustrates a class of segmentation-based data mining algorithms for which this proposed decomposition will have significant performance benefits. Section IV gives a schematic of the grid architecture, and describes the various components, including the scheduling interface between the data and compute grids. Section V provides the summary and conclusions.

I. Introduction

A. Overview

The traditional statistical data mining approach consists of two steps. The first step, referred to as *modeling,* takes a training data set containing the problem features of interest and uses techniques from multivariate statistics and machine learning to construct models from this training data set, which can be used for variety of application including predictive modeling, exploratory data analysis, and summarization of the data. Once a suitable model has been obtained and validated, the second step, referred to as *scoring,* uses the resulting models for prediction, classification or categorization. In this paper, we are primarily concerned with the modeling step above, although some of the considerations will apply to the highly data-parallel scoring step as well.

We now consider the implications and evolution of this data mining approach from the perspectives of the business application, the data management and the computational requirements respectively.

R. Natarajan is with the IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, N.Y. USA (phone: 914-769-9134, e-mail: nramesh@ us.ibm.com).

R. Sion is with the IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120(e-mail: rsion@us.ibm.com).

C. V. Apte is with the IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, N.Y. USA (e-mail: apte@ us.ibm.com).

I. S. Narang is with the IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120(e-mail: narang@almaden.ibm.com).

From the business application perspective, the modeling step involves specifying the relevant data variables for the business problem of interest, marshalling the training data for these features from a large number of historical cases, and finally invoking the data mining kernel. The scoring step requires collecting the data for the model input features for an individual case (typically the model input features used in scoring are a smaller subset of those in the original training data, as the modeling step will have excluded the irrelevant features from further consideration), and generating model-based predictions or expectations based on these inputs. The results from the scoring step are then used for triggering business actions that optimize the relevant business objectives.

This approach may be illustrated by the following example scenario. An airline company designing a new loyalty program uses its historical customer purchase and behavioral data from previous promotions to build a response model for the new promotion. This response model is used in conjunction with a profitability model to score and rank customers. This ranking is used to select a group of preferred customers, and to decide on the specific details of the promotion. In this application, the modeling and scoring steps would be typically performed in batch mode (e.g., once a year if the promotion is offered annually). However, evolving business objectives, competitive pressures and technological capabilities might change this scenario. For example, the modeling step may be performed more frequently to accommodate new data or new data features as they become available, particularly if the current model rankings and predictions are likely to significantly change due to changes in the input data distributions or in the modeling assumptions. In addition, the scoring step can even be performed interactively (e.g., the customer may be rescored in response to a transaction event that can potential trigger a profile change, leading to an immediate loyalty program offer at the customer point-of-contact).

Turning to the data perspective, the business may use a central data warehouse for storing the relevant data and schema in a form suitable for mining. This data warehouse, which is loaded with data from other transactional systems or external data sources after various data cleansing, transformation, aggregation and merging operations, is typically implemented on a parallel database system to obtain scalable storage and query performance for the large data tables. For example, many databases support multi-threaded, shared-memory or distributed, shared-nothing modes of parallelism or both (for example, [4], where these two modes are termed as INTRA and INTER PARALLEL respectively). However, in evolving scenarios, the relevant data may also be distributed in multiple, multi-vendor data warehouses across various organizational dimensions, departments and geographies, and across supplier, process and customer databases. In addition, external databases containing frequently-changing industry or economic data, market intelligence, demographics, and psychographics may also be incorporated into the training data for data mining in specific application scenarios. Finally, we consider the scenario where independent entities may collaborate to "virtually" share their data for modeling purposes, without explicitly exporting or exchanging raw data across their organizational boundaries (e.g., a set of hospitals may pool their radiology data to improve the robustness of diagnostic modeling algorithms). The use of federated and data grid technologies, such as [5] which hide the complexity and access permission details of these multiple, multi-vendor databases from the application developer, and which can rely on the query optimizer to minimize excessive data movement and other distributed processing overheads, will also become important for data mining.

From the computational perspective, many statistical modeling techniques for forecasting and optimization are unsuitable for massive data sets, and these techniques often only use a smaller sampled fraction of the data, which increases the variance of the resulting model parameter estimates. Alternatively, they use a variety of heuristics to reduce computational time which have a negative impact on the quality of the model search and optimization. A further limitation is that many data mining algorithms are implemented as standalone or client applications that extract database-resident data into their own memory workspace or disk area for the computational processing. The use of client programs external to the data server incurs high data transfer and storage costs for large data sets. Furthermore, even for smaller or sampled data sets it raises issues of managing multiple data copies and schemas that cannot be easily synchronized to the changing data specifications or content on the database servers. In addition, a set of external processes for data mining with its own proprietary API's and programming requirements cannot be easily integrated into the SQL-based, data-centric framework of business applications.

B. *Implications for Data Mining Architectures*

The evolution of these data mining architectures for modeling is summarized in Figure 1 (we assume that the result of the modeling is an exportable model). In (a), the data mining kernel is implemented as a client application, which uses data that is extracted from the database into its own workspace. In (b), the data mining is implemented as a database-extender consisting of stored procedures and user defined functions installed on the database server. Finally, (c) shows a further evolution to a grid-based architecture with external computational servers implementing high-level mining constructs.

The client-based approach in (a) is useful for carrying out data mining studies in a experimental mode, for preliminary development of new algorithms, and for testing parallel or high-performance implementations of various data mining kernels.

In recent years, the commercial emphasis has been on the approach in (b) where the model generation and scoring algorithms for a set of robust, well-tested data mining kernels are implemented as database extenders. All major database vendors now support integrated mining capabilities on their platforms. The use of accepted or de-facto standards such as SQL/MM, which is a SQL-based API for task and data specification [6], and PMML, which is a XML-based format for results reporting and model exchange [7] enables these integrated mining kernels to be easily incorporated into the production workflow of data-centric business applications. Furthermore, (b) has the advantage over (a) that the data modeling can be triggered based on the state of internal events recorded in the database.

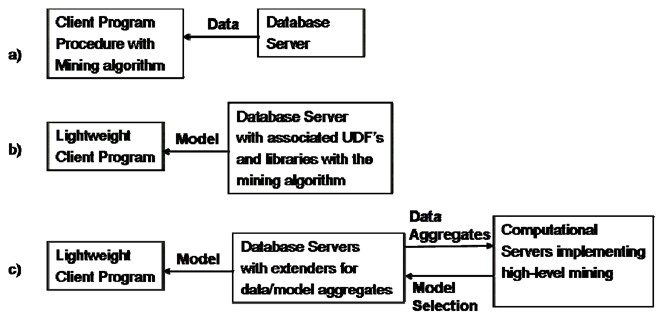
a)
Client Program Procedure with Mining algorithm
Data
Database Server
b)
Lightweight Client Program
Model
Database Server with associated UDF's and libraries with the mining algorithm
c)
Lightweight Client Program
Model
Database Servers with extenders for data/model aggregates
Data Aggregates
Model Selection
Computational Servers implementing high-level mining

Figure 1: Evolution of Data Mining from Databases

.

The data mining architecture in (c) is a grid-based data mining approach whose relevance and capabilities for enterprise-scale data mining relative to (a) and (b) is considered below.

First, we note that any client application in (a) can be recast as a grid application, that can be invoked through the database layer using the SQL/MM task and metadata specification (the training data can be exported from the data server as part of the grid task invocation, or a data connection reference can be provided to enable the grid task to connect itself to the data source). Except for the major issue of the data transfer overheads, this approach combines all the remaining advantages of (a) and (b) mentioned earlier.

Second, most stored procedure implementations of common mining kernels are straightforward adaptations of existing client-based programs. Although the stored procedure approach avoids the data transfer costs to external clients, and can also take advantage of the better I/O throughput from the parallel database subsystem to the stored procedure, it ignores the more significant performance gains obtained by reducing the traffic on the database subsystem network itself (for partitioned databases), or by reducing thread synchronization and serialization during the database I/O operations (for multi-threaded databases). The memory and CPU requirements of these stored procedure adaptations, particularly for long-running data mining tasks, can also negatively impact the performance of a multi-purpose, operational database server.

Third, is it difficult to directly adapt existing data-parallel client data mining programs as stored procedures, because the details of the data placement and I/O parallelism are managed by the database administration and system policy, and by the SQL query optimizer, and are not under the control of the application.

Fourth, as data mining applications grow in importance, they will have to compete for CPU cycles and memory on the database server with the more traditional transaction processing, decision support and database maintenance workloads. Here, depending on the service-level requirements for the individual components in this workload, it may be necessary to offload data mining calculations in an efficient way to other computational servers for peak workload management

Fifth, assuming that the associated distributed computing overheads can be kept small, the outsourcing of the data mining workloads to external compute servers is attractive as a computational accelerator, and it provides opportunities to improve the quality of data mining models, through algorithms that perform more extensive model search and optimization,.

II. ALGORITHMIC FORMULATION

A. Functional Decomposition of Mining Kernel

The grid-mining architecture described in this paper is based on reformulating the data mining algorithm into two separate functional phases, viz., a sufficient statistics collection phase implemented in parallel on the data grid, and a model selection and parameter estimation phase implemented in parallel on a compute grid. Successive iterations of these two phases may be used for model refinement and convergence. This functional decomposition is illustrated schematically in Figure 2, where the data grid may be a parallel or federated database, and the compute grid may be high-performance compute-server or a collection of low-cost, commodity processors.

The use of sufficient statistics for model parameter estimation is a consequence of the Neyman-Fisher factorization criterion [8], which states that under the assumption that the data consists of an i.i.d sample $X_1, X_2, \ldots, X_n$, drawn from a probability distribution $f(x \mid \theta)$, where x is a multivariate random variable and θ is a vector of parameters, then the set of functions $S_1(X_1, X_2, \ldots, X_n), \ldots, S_k(X_1, X_2, \ldots, X_n)$ of the data are sufficient statistics for θ, if and only if the likelihood function defined as

$$L(X_1, X_2, \ldots, X_n) = f(X_1 \mid \theta) f(X_2 \mid \theta) \ldots f(X_n \mid \theta),$$

can be factorized in the form,

$$L(X_1, X_2, \ldots, X_n) = g_1(X_1, X_2, \ldots, X_n) g_2(S_1, \ldots, S_k, \theta),$$

where g_1 is independent of θ, and g_2 depends on the data only through the sufficient statistics. A similar argument holds for conditional probability distributions $f(y \mid x, \theta)$, where (x, y) are joint multi-variate random variable (the conditional probability formulation is required for classification and regression applications with y denoting the response variable). The cases for which the Neyman-Fisher factorization criterion holds with small values of k are interesting, since the sufficient statistics $S_1, S_2, \ldots, S_k$, not only gives a compressed representation of the information in the data needed to optimally estimate the model parameters θ using maximum likelihood, but they can also be used to provide a likelihood score for a (hold-out) data set for any given values of the parameters θ (the function g_1 is a multiplicative constant for a given data set that can be ignored for comparing scores). This means that both model parameter estimation and validation can be performed without referring to the original training and validation data.

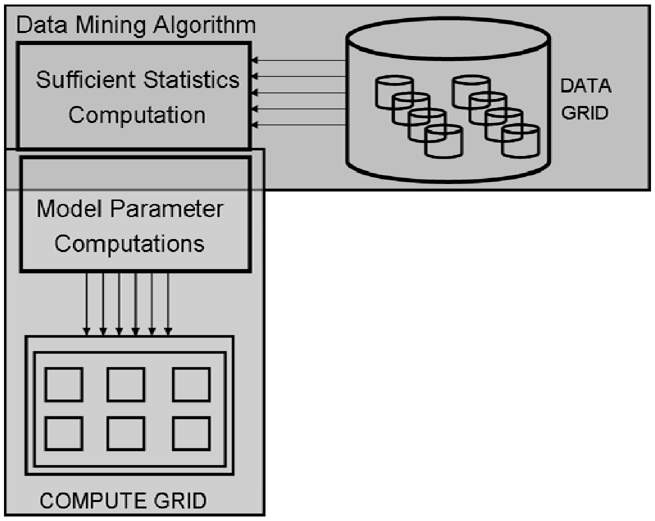
Data Mining Algorithm
Sufficient Statistics Computation
DATA GRID
Model Parameter Computations
COMPUTE GRID

Figure 2: Functional decomposition of data mining algorithm

In summary, the functional decomposition of the mining kernel can be shown to have several advantages for a grid-based implementation.

First, many interesting data mining kernels can be adapted to take advantage of this algorithmic reformulation for grid computing, which is a consequence of the fact that there is a large class of distributions for which the Neyman-Pearson factorization criterion holds with a compact set of sufficient statistics (for example, these include many of the distributions in the exponential family such as Normal, Poisson, Log-Normal, Gamma, etc.).

Second, for these many of these kernels, the size of the sufficient statistics is not only significantly smaller than the entire data set which reduces the data transfer between the data and compute grids, but the sufficient statistics can also be computed efficiently in parallel with minimal communication overheads on the data-grid subsystem.

Third, the benefits of parallelism for these new algorithms can be obtained without any specialized parallel libraries on either the data or compute grid (e.g., message passing or synchronization libraries). In most cases, the parallelism is obtained by leveraging the existing data partitioning and query optimizer on the data grid, and by using straightforward, non-interacting parallel tasks on the compute grid.

B. *Related Work*

The proposed algorithmic decomposition described above can accommodate many of the data mining formulations in the literature as special cases. For example, as a trivial case the entire data set is a sufficient statistic for any modeling algorithm (although not a very useful one from the data compression point of view), and therefore, sending the entire data set is analogous to the usual grid-service enabled client application on the compute grid. Another example is obtained by matching each partition of a row-partitioned database table to a compute node on a one-to-one basis, which leads to distributed algorithms where the individual models computed from each separate data partitions are combined using weighted ensemble averaging to get the final model [9]. Yet another example is bagging [10], where copies of the original data set obtained by random sampling with replacement from the full data set, are used by distinct nodes on the compute grid to construct independent models which are then averaged to obtain the final model. The use of competitive mining algorithms provides another example, in which identical copies of the entire data set are used on each compute node to perform parallel independent searches for the best model in a large model search space [11]. All these algorithms fit into the present framework, and are efficient if the sufficient statistics can be passed instead of the full data.

There is also a considerable literature on the implementation of well-known mining algorithms such as association rules, K-means clustering and decision trees directly with database resident data. Some of these algorithms are client application or stored procedures that are structured so that rather than copying over the full data or using a cursor interface to the data, they directly issue database queries to obtain the relevant sufficient statistics. For example, [12] considers a decision tree algorithm in which for each step in the decision tree refinement, a database query is used to return the relevant sufficient statistics required for that step (these sufficient statistics are of the set of all bi-variate contingency tables involving the target feature at each node of the current decision tree). They show how this query can be formulated so that the desired results can be obtained in a single database scan. Also, [13] considers the same issue of obtaining the sufficient statistics for decision tree refinement, in the distributed case when the data tables are partitioned by rows and by columns respectively. These approaches do not focus on the computational requirements in the stored procedure, which are relatively quite small for decision tree refinement, and offer little scope for the use of computational parallelism

There has been some related work on pre-computation or caching of the sufficient statistics from data tables for eventual amortized use in data mining. For example, [14] describe a sparse data structure for compactly storing and retrieving all possible contingency tables that can be constructed from a database table, and they show that this data structure can be used by many statistical algorithms, including log-linear response modeling. A related method is squashing [15], where a small number of pseudo data points and corresponding weights are obtained, so that the low-order multivariate moments of the pseudo data set and the original large data set are equivalent; many modeling algorithms such as logistic regression can use these weighted pseudo data points, which can be regarded as an approximation to the sufficient statistics of the original large data set, as a computationally-efficient substitute for modeling purposes. The idea of using approximate rather than exact sufficient statistics to reduce the computational and data access costs of modeling algorithms has also been considered for feature selection in linear models [16], and for structure identification in Bayesian networks [17].

III. Segmentation based modeling

A. Motivation

In commercial applications of data mining, the primary interest is often in extending, automating and scaling up the existing and traditional predictive modeling methodology. One difficulty that is frequently encountered with this approach is the need to deal with heterogeneous data populations (i.e., data that is drawn from a mixture of distributions), each of which exhibits the same general model characteristics but with different values for the model parameters. A general class of methods that is very useful in this context is segmentation-based predictive modeling [18]. Here the space of the explanatory variables in the training data is partitioned into mutually-exclusive, non-overlapping segments, and individual predictive models are constructed for each segment using multi-variate probability models that are standard practice in the relevant application domain.

The overall model naturally takes the form of "*if-then*" rules, where the "*if*" part defines the condition for segment membership, and the "*then*" part defines the corresponding segment predictive model. The segment definitions are Boolean combinations of univariate tests on each explanatory variable, including range membership tests for continuous variables, and subset membership tests for nominal variables (note that these segment definitions can be easily translated into the *where*-clause of an SQL query for retrieving all the data records in the corresponding segment from the database).

The determination of the appropriate segments and the estimation of the model parameters in the corresponding segment models can be carried out by jointly optimizing the likelihood function of the overall model for the training data, with validation or hold-out data being used to prevent model overfitting. This is a complex optimization problem involving search and numerical computation, and a variety of heuristics including top-down segment partitioning, bottom-up segment agglomeration, and combinations of these two approaches are used in order to determine the best segmentation/segment-model combination. The segment models that have been studied include a bi-variate Poisson-Lognormal model for insurance risk modeling [19], and multivariate linear and logistic regression models for retail response modeling [20]. These algorithms have also been used to generate feature transformations for other predictive modeling methods [21], and they are closely related to model-based clustering techniques (e.g., [22]).

B. Representative Performance Analysis

The potential benefits of the proposed formulation for segmentation-based modeling can be examined using the following architectural model. The data grid and compute grids are assumed to consist of P_1 and P_2 processors respectively, with the corresponding time for 1 floating point operation (flop) on each node being denoted by α_1 and α_2 respectively, and the time for accessing a single data field on the data node being denoted by β_1. Similarly, the cost of invoking a remote method on a compute grid node is denoted by $\gamma_1 + \gamma_2 w$, where γ_1 is the latency for remote method invocation, γ_2 is the cost per word for moving data over the network, and w is the size (in words) of the data parameters that are transmitted. Finally, we assume that the database table used for the segmentation-based modeling consists of n rows and m columns, and is perfectly row-partitioned so that each data grid partition has n / P_1 rows (we ignore the small effects when n is not perfectly divisible by P_1).

Using this model, we consider one pass of a multi-pass a segmented predictive model evaluation, in which a linear regression model with feature selection is computed in each segment (e.g., using the algorithms described for evaluating the sufficient statistics as described in [20]). Since several potential segmentations can be evaluated in parallel, we assume that there are N segments, which may be non-overlapping or overlapping (with $N >> P_1, P_2$ in general). The sufficient statistics for each potential segment are a pair of covariance matrices (training + evaluation) for the data in each segment, which can evaluated for all N segments in a single parallel scan over the data table. The overall time for this aggregation step T_D, which can be shown to be given by $(\beta_1 nm + 0.5\alpha_1 knm^2) / P_1 + \alpha_1 NP_1 m^2$. Here $k < N$ is an overlap factor which denotes the number of segments that each data record on average contributes to, with $k = 1$ in the case of non-overlapping segments. The three terms in T_D respectively correspond to the time for reading the data from the database, the time for updating the covariance matrices locally, and the time for aggregating the local covariance matrix contributions for each segment at the end of a data scan. These aggregates are then dispatched to a compute node, for which the scheduling time T_S can be estimated as $\gamma_1 + \gamma_2 Nm^2 / P_2$ (where we assume that these independent scheduling tasks can be performed in parallel using a multi-threaded scheduler). On the compute nodes, the algorithm used to perform stepwise feature selection proceeds by successively adding features to the regression model, based on first computing the regression model parameters from the training data sufficient statistics, and then examining the degree-of-fit of these models using the evaluation data sufficient statistics. The time T_C for the parameter computation and model selection step can be estimated as $\frac{1}{12}\alpha_2 Nm^4 / P_2$, where only leading order terms for large m have been retained. We note that the usual algorithms for the solution of the normal equations by the Choleski factorization algorithm are $O(m^3)$ (e.g., as described in [20]), but incorporating the more rigorous feature selection algorithm based on the degree-of-fit with the evaluation data as proposed above, raises the algorithm complexity to $O(m^4)$. The total time T is thus given by

$$\begin{aligned} T &= T_D + T_S + T_C \\ &= (\beta_1 nm + 0.5\alpha_1 knm^2)/P_1 + \alpha_1 NP_1 m^2 . \\ &+ \gamma_1 + \gamma_2 Nm^2 / P_2 + \tfrac{1}{12}\alpha_2 Nm^4 / P_2 . \end{aligned}$$

We consider some data sets that are representative of retail customer applications, with $n = 10^5, m = 500$ and take the nominal values $k = 15$, $N = 500$ for the algorithm parameters. For the architectural parameters we set $\alpha_1 = \alpha_2 = 2 \times 10^{-8}$ sec, $\beta_1 = 5 \times 10^{-6}$ sec/ *word*, $\gamma_1 = 4 \times 10^{-2}$ sec and $\gamma_2 = 2 \times 10^5$ sec/ *word*.

For the case $P_1 = 1, P_2 = 0$, when all the computation must be performed on the data server itself (this is then equivalent to the case (b) described in Figure 1), the overall execution time is T = 7.8 hours (we note that in this case there is no scheduling cost as all the computation is performed on the data server itself). For the case, $P_1 = 1, P_2 = 1$, the execution time increases to $T = 8.91$ hours (which consists of $T_D = 0.57$ hours, $T_S = 1.11$ hours and $T_C = 7.23$ hours), which is an overhead of 14% over the serial case, although with 80% of the overall time being off-loaded from the data server. However, by increasing the number of processors on the data and compute grids to $P_1 = 16, P_2 = 128$, the execution time comes down to $T = 0.106$ hours (which consists of $T_D = 0.041$ hours, $T_S = 0.009$ hours and $T_C = 0.057$ hours), and this represents a speedup of about 74 over the base case when all the computation is performed on the data server itself.

We have intentionally used modest values for n, m in the analysis given above, and many retail data sets can have millions of customer records, and the number of features can increase dramatically if interaction terms involving the primary features that are incorporated in the segment models. For example, quadratic and even higher-order interaction terms in the segment regression models in order to have a final predictive model with fewer overall segments, but with more complicated nonlinear models within each segment). The analysis suggests that an increasing the number of segment modeling features makes the use of a separate compute grid even more compelling for this application.

We believe that segmentation-based predictive data modeling problems are ideally suited for the proposed grid architecture, with significant costs requiring parallelism in the data access and sufficient statistics computations on the data grid, as well as in the model computation and search on the compute grid. These modeling runs are estimated to require several hours of computational time running serially on current data mining architectures for typical data sets.

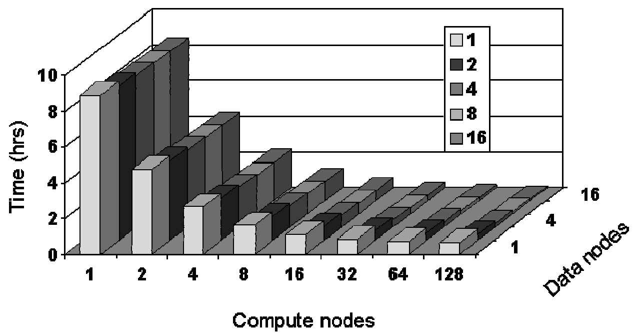

Figure 3: Execution time for varying numbers of compute and data nodes

IV. GRID ARCHITECTURE SCHEMATIC

A. Overall Schematic

The overall schematic for grid-based data mining is shown in Figure 4. It consists of a parallel or federated database, a web service engine for task scheduling and monitoring, and a compute grid. A functional schematic describing the various components is shown in Figure 5.

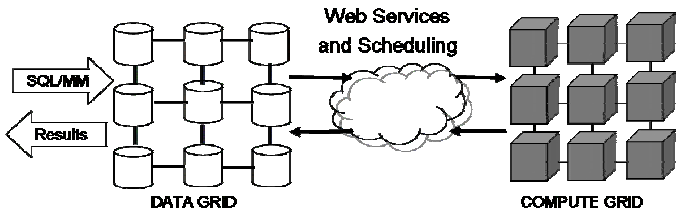

Figure 4: Grid data mining architecture schematic

B. Detailed Description of Individual Components

Our description for the data grid layer will refer to the DB2 family of products [4], although the details are quite generic and can be ported to other relational databases as well. The data grid layer implements the SQL/MM interface for data mining task specification and submission. A stored procedure performs various control-flow and book-keeping tasks, such as for example, issuing parallel queries for sufficient statistics collection, invoking the web service scheduler for parallel task assignment to the compute grid, aggregating and processing the results from the compute grid, managing the control flow for model refinement, and exporting the final model.

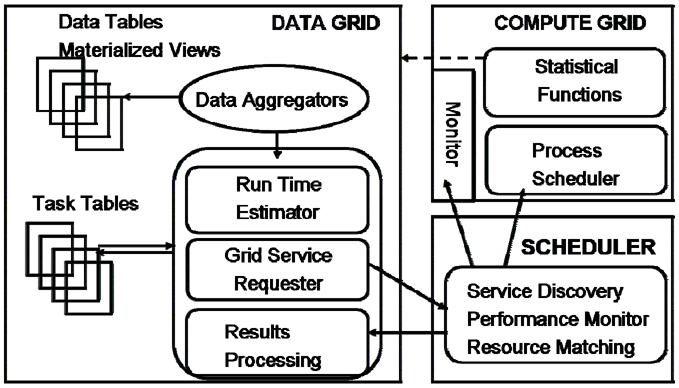

Figure 5: Functional schematic of data mining components

Many parallel databases provide built-in parallel column functions like MAX, MIN, AVG, SUM and other common associative-commutative functions, but do not yet provide an API for application programmers to implement general-purpose multi-column parallel aggregation operators [23]. Nevertheless, these parallel data aggregation operators for accumulating the sufficient statistics can be implemented using scratchpad user defined functions, which on parallel databases leverage the parallelism in the SQL query processor (for both INTRA and INTER parallel modes) by using independent scratchpads for each partition or thread. The results in the scratchpads are then aggregated at the end of the data scan using the shared memory or shared disk regions for communication.

For federated databases, these data aggregation operators would be based on the federated data view, but would leverage the technologies developed for the underlying federated query processor and its cost model in order to optimize the trade-offs between function shipping, data copying, materialization of intermediate views, and work scheduling and synchronization on the components of the federated view to compute the sufficient statistics in the most efficient way ([24], [25], [26]).

The task scheduler, which is implemented as a web service for full generality, can be invoked from SQL queries issued from the database stored procedure (in the case of DB2, using the SOAP messaging capabilities provided by a set of user defined functions for invoking remote web services with database objects as parameters, as provided in the XML extender [27]). This invocation of the scheduler is asynchronous, and the parameters that are passed to the scheduler include the task metadata and the relevant task data aggregate. It also includes a runtime estimate for the task, parameterized by CPU speed and memory requirements. In the special case when the compute grid is co-located within the same administrative domain as the data grid, rather than passing the data aggregate as an in-line task parameter, a connection reference to this data is instead passed to the scheduler. This connection reference is used by the corresponding remote task on the compute grid to retrieve the relevant data aggregate, thereby avoiding the small but serial overhead of processing a large in-line parameter in the invocation of the scheduler.

The task scheduler, which shields the data layer from the details of the compute grid, has modules for automatic discovery of compute resources with the relevant compute task libraries, built-in scheduling algorithms for load balancing, task-to-hardware matching based on the processor and memory requirements, polling mechanisms for monitoring task processes, and task life-cycle management including mechanisms for resubmitting incomplete tasks. The parameterized runtime estimates for the tasks are combined with the server performance statistics for task matching, load balancing and task cycle management (which includes the diagnosis of task failures or improper execution on the compute grid). The scheduler can also implement various back-end optimizations to reduce task dispatching overheads on trusted compute-grid resources. Our experiments on a 4 node Linux cluster located on the same LAN as the data server, shows an average task scheduling overhead of about 40 milliseconds through the web-service scheduler, suggesting that the granularity of the scheduled tasks must be somewhat larger in order for the linear speedup regime to apply on the compute grid.

The compute grid layer contains the code base for high-level mining services including numerically-robust algorithms for parameter estimation and feature selection from the input data or from the sufficient statistics of the data where applicable. The compute grid nodes also contain the resource discovery and resource monitoring components of the task scheduler, which are used for task matching by the scheduler as described above. We are currently experimenting with a range of hardware platforms for the compute grid including commodity processors and high-performance compute servers on a LAN, as well as multi-site remote compute servers.

V. Summary

As business applications of data mining become widespread, it will be necessary to improve the performance and quality of the embedded data mining kernels, as well as minimize their impact on other aspects of the data processing operational workload.

The approach to enterprise-scale data-mining described in this paper is an evolutionary approach that will meet these requirements. It is based on a functional decomposition of the data mining kernel to exploit the parallelism on data and compute grids with minimal use of explicit parallel programming software constructs, and with minimal inter-grid communication. It leverages existing or evolving data mining standards such as SQL/MM for job specification and submission, and PMML for model specification and export. It can provide better scalable performance or an improved data mining model quality by using parallelism, when compared to existing implementations of database-integrated mining kernels. Finally, the present approach can lead to greater flexibility in the deployment of a data mining application, leading to better overall workload management on enterprise data servers that support a complex mix of transactional, decision support, data management and data mining applications.

REFERENCES

[1] C. Apte, B. Liu, E. P. D. Pednault and P. Smyth, “Business Applications of Data Mining,” *Communications of the ACM*, Vol. 45, No. 8, August 2002.

[2] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, ”Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets,” *Journal of Network and Computer Application,* Vol. 23. pp. 187-200, 2001.

[3] D. Arnold, S. Vadhiyar and J. Dongarra, “On the Convergence of Computational and Data Grids,” *Parallel Processing Letters*, Vol. 11, pp 187-202, 2001.

[4] The IBM DB2 Universal Database V8.1, http://www.ibm.com/software/data/db2, 2004.

[5] The IBM DB2 Information Integrator, http://www.ibm.com/software/integration, 2004.

[6] ISO/IEC 13249 Final Committee Draft. Information Technology – Database Languages –SQL Multimedia and Application Packages. http://www.iso.org, 2002

[7] Predictive Modeling Markup Language, http://www.dmg.org, 2002.

[8] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*, Third Edition, Addison Wesley, 2002.

[9] A. Prodromides, P. Chan and S. Stolfo, “Meta learning in distributed data systems – Issues and Approaches,” *Advances in Distributed Data Mining*, (eds. H. Kargupta and P. Chan), AAAI Press, 2000.

[10] L. Breiman, “Bagging Predictors,” *Machine Learning*, Vol. 24, No. 2, pp. 123-140, 1996.

[11] P. Giudici and R. Castelo, “Improving Markov Chain Monte Carlo Model Search for Data Mining,” *Machine Learning,* Vol. 50, pp 127-158, 2003.

[12] Graefe, G.; U. Fayyad and S. Chaudhuri, "On the efficient gathering of sufficient statistics for classification from large SQL databases," *Proceedings Fourth International Conference on Knowledge Discovery and Data Mining,"* AAAI Press, Menlo Park, pp.204-208, 1998.

[13] D. Caragea, A. Silvescu and V. Honavar, ‟A Framework for Learning from Distributed Data Using Sufficient Statistics and its Application to Learning Decision Trees,″ *Int. J. Hybrid Intell. Syst*., Vol. 1, pp. 80-89, 2004.

[14] A. Moore and Mary Soon Lee, “Cached Sufficient Statistics for Efficient Machine Learning with Massive DataSets,” *Journal of Artificial Intelligence Research*, Vol. 8, pp. 67-91, 1998.

[15] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes and D. Pregibon, “Squashing Flat Files Flatter,” *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining,* pp. 6-15, 1999.

[16] R. Natarajan and E. P. D. Pednault, “Using Simulated Pseudo Data to Speed Up Statistical Predictive Modeling from Large Data Sets,” *Proc. First SIAM Conference on Data Mining*, Chicago IL, 2000.

[17] N. Friedman and L. Getoor,, “Efficient learning using constrained sufficient statistics,” *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistic, 1999.*

[18] C. Apte, R. Natarajan, E. Pednault, F. Tipu, A Probabilistic Framework for Predictive Modeling Analytics, *IBM Systems Journal*, V. 41(3), 2002.

[19] C. Apte, E. Grossman, E. Pednault, B. Rosen, F. Tipu, and B. White, “Probabilistic Estimation Based Data Mining for Discovering Insurance Risks ,“ *IEEE Intelligent Systems*, Vol. 14(6), 1999.

[20] R. Natarajan and E. P. D. Pednault, “Segmented Regression Estimators for Massive Data Sets,” *Proc. Second SIAM Conference on Data Mining,* Crystal City VA, 2002.

[21] E. Pednault, “Transform Regression and the Kolmogorov Superposition Theorem,” IBM Research Report RC 23227, IBM Research Division, Yorktown Heights, NY 10598, 2004.

[22] C. Fraley, “Algorithms for Model-Based Gaussian Hierarchical Clustering,” *SIAM J. Sci. Comput*., V. 20, No. 1, pp. 270-281 1988.

[23] M. Jaedicke and B. Mitschang, “On Parallel Processing of Aggregate and Scalar Function in Object-Relational DBMS,” *Proc. ACM SIGMOD Int. Conf. on Management of Data,* Seattle WA, 1998.

[24] M. Atkinson, A. L. Chervenak, P. Kunszt, I. Narang, N. W. Paton, D. Pearson, A. Shoshani, and P. Wilson, “Data Access, Integration and Management,” Chapter 22, *The Grid: Blueprint for a New Computing Infrastructure*, Second Edition” (eds., I. Foster and C. Kesselman), Morgan Kaufman, 2003.

[25] M. Rodriguez-Martinez and N. Roussopoulos, “MOCHA : A Self-Extensible Database Middleware System for Distributed Data Sources,” *Proc.s ACM SIGMOD International Conference for Distributed Data Sources*, Dallas TX, pp. 213-224, 2000.

[26] D. Kossmann, Franklin, M. J. and Drasch G., "Cache investment: integrating query optimization and distributed data placement,” *ACM Transactions on Database Systems*, Vol. 25, pp. 517-558, 2000.

[27] The IBM DB2 XML Extender, http://www.ibm.com/software/data/db2/extender/xmlext, 2004.