Data Mining Algorithms as a Service in the Cloud Exploiting Relational Database Systems

Carlos Ordonez
University of Houston
USA

Javier García-García
Instituto Politécnico Nacional
Mexico

Carlos Garcia-Alvarado
Greenplum/EMC
USA

Wellington Cabrera
University of Houston
USA

Veerabhadran Baladandayuthapani
UT MD Anderson C.C.
USA

Mohammed S. Quraishi
University of Houston
USA

ABSTRACT

We present a novel cloud system based on DBMS technology, where data mining algorithms are offered as a service. A local DBMS connects to the cloud and the cloud system returns computed data mining models as small relational tables that are archived and which can be easily transferred, queried and integrated with the client database. Unlike other analytic systems, our solution is not based on MapReduce. Our system avoids exporting large tables outside the local DBMS and thus it avoids transmitting large volumes of data to the cloud. The system offers three processing modes: local, cloud and hybrid, where a linear cost model is used to choose processing mode. In hybrid mode processing is split between the local DBMS and the cloud DBMS. Our system has a job scheduler with FIFO, SJF and RR policies to enhance response time and get partial results early. The cloud DBMS performs dynamic job scheduling, model computation and model archive management. Our system incorporates several optimizations: local data set summarization with sufficient statistics, sampling, caching matrices in RAM and selectively transmitting small matrices, back and forth. We show that in general the most efficient computing mechanism is hybrid processing: summarizing or sampling the data set in the local DBMS, transferring small matrices back and forth, leaving mathematically complex methods as a task for the cloud DBMS.

1. INTRODUCTION

Cloud computing represents a new paradigm in which hardware and software are offered as a service on the Internet, supported by a cluster of computers, providing parallel processing and ample storage [1]. Such approach simplifies IT management and reduces costs. On the other hand, DBMSs supporting SQL remain the dominant data management technology to process transactions (OLTP) and to build large data warehouses (for OLAP, data mining) [5]. Analyzing large databases remains one of the most important challenges, where data mining comes into play. However, in practice, data mining is commonly processed outside the DBMS with statistical or math packages (e.g. Matlab, R). Such programs do not scale to large data sets and require exporting relational tables, which is slow and compromises security. In addition, large normalized tables cannot be directly analyzed to compute a data mining model. This is because most models require variables not available as columns on a table in the database. Therefore, preprocessing tables with joins, aggregations and various mathematical transformations is generally required to build a data set. We should mention that for large scale analytics SQL (including UDFs) and MapReduce are now the main programming alternatives [5]. However, in MapReduce it is necessary to transfer and store the database (or data set) on the cloud, which may not be a practical or acceptable solution. Despite its shortcomings, we demonstrate SQL, extended with UDFs and Stored Procedures, is a promising alternative for database analytics in a cloud service model.

Considering the motivation explained above, we present a cloud computing prototype that offers data mining algorithms as a service. We consider that the cloud service can be external, on the Internet, outside the firewall (possibly a slow connection) or internal, inside the firewall (fast and secure local network). Our system incorporates optimizations to minimize I/O, to reduce data redundancy and to avoid transmitting large volumes of data, which are practical bottlenecks in a cloud environment. There are important differences with other cloud systems. Unlike other cloud systems, our cloud coordinates DBMS processing and it is not based on Hadoop and MapReduce. Another difference is that we show that in certain cases a "small" cloud is feasible, instead of a large cloud with "infinite" and elastic resources. Assuming the user has a local DBMS, but wants to exploit a cloud for data mining, there are several alternatives to split processing between the local DBMS and the cloud, instead of moving the entire database to the cloud. Therefore, the local DBMS participates in cloud processing.

Our system provides important advantages. The cloud system can reduce local workload, it can accelerate processing and it can simplify multiple model management. Our system offers several off the shelf models ready to use; there is no need to install data mining software on the local computers, neither on the DBMS server nor on the local workstation (but such option is available). Since all computations are done in SQL the data set, intermediate tables and data mining models can be queried. Another advantage is that

OLTP databases can be directly analyzed, bypassing a data warehouse. On the other hand, the I/O bottleneck to load large data sets into the cloud can be avoided. When data records remain in the DBMS security is easier to enforce. Our system simplifies database and model management and reduces data redundancy because tables can be preprocessed and summarized locally. Our system extends the DBMS with management of models: the cloud stores a history of computed models, together with their parameters and data set information (columns, provenance) as well as data mining job information. There exist many databases, which do not require massively parallel programming. Moreover, in general, a large transaction table or web data need to be summarized (aggregated) before a data mining model can be computed (where MapReduce is a good complement [5]). All processing is done in standard SQL queries, giving the user the ability to write queries for every processing stage and to integrate diverse DBMSs. Our system supports the full data mining cycle: preprocessing tables to build data set, computing a model on the data set, tuning statistical parameters of such model and deploying the model.

2. SYSTEM OVERVIEW

2.1 System Architecture

We assume there is a client DBMS server with a large data set that requires analysis. We assume the user needs the following services: (1) data mining algorithms provided by the cloud, (2) reducing the local DBMS workload for heavy data mining processing, or (3) model management provided by the cloud. The cloud system has a powerful DBMS server, prepared to analyze data set summaries, data set samples or a combination. There is an additional front-end cloud management server responsible for handling data mining job requests, monitoring server progress, estimating cost for three alternative processing modes, and managing jobs. The minimum "system" requirement are that both DBMS servers (local and cloud) can evaluate standard SQL queries and aggregate UDFs. The cloud DBMS has SQL stored procedures (SPs) and UDFs ready to use. Alternatively, UDFs and SPs can be installed on the local DBMS. From a "systems" perspective, database processing happens in the DBMS server with SQL queries (possibly calling UDFs). To connect to the cloud system on the Internet, there is a web application responsible for communicating the local computer and the cloud. Our system considers network speed, data set size and iterative versus non-iterative method to decide how to process a data mining job.

Our system offers data mining algorithms for large data sets as a DBMS service on the Internet. Our system splits work between the cloud and the local DBMS to minimize I/O and data transmission. After a model is computed, the cloud sends back SQL queries for scoring, which are used to tune the model, evaluate (test) accuracy on an independent data set or deploy the model to a production environment. Finally, the cloud offers a service to manage and query statistical models. The model database includes input data sets, selected columns, model input parameters, model quality, job parameters and job processing time, among others.

2.2 Data Sets and Statistical Models

The data set $X = \{x_1, x_2, \ldots, x_n\}$ has n records and d attributes (numeric or categorical). In several models all

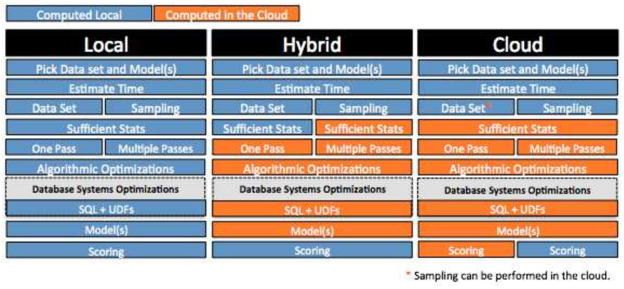

Figure 1: Processing Mode: Local, Hybrid, Cloud.

attributes are numeric (i.e. d is dimensionality), where categorical attributes are mapped to binary dimensions. Such data set is stored as a relational table with d columns and an additional primary key column (we call it i). We assume the data set was built before by preprocessing (joining, aggregating, transforming) tables in the local database. The user is responsible for providing the data set table, where the primary key will be automatically excluded from analysis.

All our statistical models share a common matrix foundation, which enables efficient summarization and generalized optimizations [3, 2]. Our system only offers statistical models that can be computed with sufficient statistics and which can produce accurate results with samples. Unsupervised models include PCA (to reduce dimensionality; solved with Singular Value Decomposition (SVD)) and K-means clustering (to group similar points; iterative). On the other hand, supervised (predictive) models include: linear regression (solved by SVD decomposition), variable selection (in linear regression, a computationally intensive problem, solved with a Bayesian MCMC method) and the famous Naïve Bayes (NB) classifier. In supervised models, X has an extra "target" attribute: a dependent variable Y (numeric in regression) or a discrete "class" attribute G (for Naïve Bayes, generally 0/1). Such extra column translates into an extra column in the input table. In summary, the system offers a full spectrum of widely used models.

2.3 Job Processing

A data mining job can be processed in three modes: (1) Local, exploiting the local DBMS, (2) Cloud, transferring the data set (if small and fast network) or samples (if large) to the cloud, (3) Hybrid, combining local and cloud processing. Local processing makes sense when the data set is relatively small, the DBMS is fast and the model can be obtained in a few runs. That is, there is a good idea about the final model. Cloud and hybrid processing are useful when the data set is large, when the local DBMS has a heavy workload, when multiple models are combined or when many trial/error runs are required (e.g., a new project, exploratory analysis). Figure 1 compares the three processing modes, step by step.

Our cloud system offers two kinds of algorithms from an I/O perspective: a fixed number of passes (one or two) or multiple passes (iterative). Naïve Bayes and PCA models can be computed in a single pass. Linear regression requires two passes: one to find coefficients and a second one to evaluate fit (squared error). Other algorithms require multiple passes. K-means takes many iterations to converge. The system also provides an incremental K-means version that gives an approximate solution in a few passes. Lastly, a Bayesian method used for variable selection needs only one pass on the data set, but takes thousands of iterations performed in RAM.

Our system has the capability to estimate running time based on a simple linear cost model that takes into account local I/O speed, transmission speed, cloud I/O speed, data set sample size, physical database operator and specific model for each processing mode, explained below. We basically fit a linear regression model where each parameter has a weight that gets periodically adjusted based on a batch of jobs. Since most data mining algorithms require reading the entire data set a table scan is the basic physical operator. Therefore, the user can make an informed decision about processing mode. Model computation with sufficient statistics and samples takes a few seconds.

We now explain our system job scheduler, which aims to provide interactive response time. Our system provides FIFO job scheduling by default. There is a wait time threshold ψ. If the job queue grows and wait time of an individual job goes beyond ψ then the system switches to Shortest Job First (SJF), giving priority to jobs working on sufficient statistics. Finally, if the system is in SJF mode the queue is long, most jobs take long time to compute and wait time goes beyond ψ then the system switches to Round Robin (RR) where users can visualize "early" results, based on incremental model computation or samples. In RR mode processing large data sets in the cloud DBMS is postponed. The system backtracks from RR to SJF and from SJF to FIFO as the workload decreases.

2.4 Data Mining Algorithmic Optimizations

We now provide a technical explanation of optimizations. Matrices with sufficient statistics (multidimensional data set summaries) are exploited to accelerate data mining algorithms. In an orthogonal manner, sufficient statistics can be computed on samples, providing fast model approximation, whose accuracy can be controlled [4]. Alternatively, they can be computed on the entire data set (if small). The sufficient statistics we exploit are n (a count of points), the linear sum of points L (a d-dimensional vector) and their quadratic (with cross-products) sum of points Q (a $d \times d$ matrix): $L = \sum_{i=1}^{n} x_i, Q = \sum_{i=1}^{n} x_i x_i^T$. Sufficient statistics matrices are mathematically constrained for each model (full matrix, diagonal matrix, multiple sets). These matrices allow deriving multiple means, variance and correlation matrices that are ubiquitous in our models. Therefore, depending on the model, one or multiple sets of L and Q are computed, where matrix Q can be constrained diagonal or full. For NB there are $k = 2$ sets of sufficient statistics: two point counts (i.e. n becomes a 2-d vector), two L d-vectors and two diagonal matrices Q. In K-means L, Q are computed on k data subsets and each Q is assumed diagonal. For PCA, variable selection and linear regression there is a single set of L, Q and Q is non-diagonal (a full, dense matrix). Sufficient statistics can be computed with SQL queries (aggregation GROUP-BY queries, medium speed), or they can be computed faster with an aggregate UDF (much faster, not portable), which requires setup (downloading the UDF, compiling it and releasing it to the DBMS). Issues about UDFs include: lack of portability, limited memory manipulation capabilities, a main memory threshold and constrained I/O functionality inside UDF code. For large data sets sampling is an important mechanism to accelerate computation, but which introduces approximation error [4]. Sampling is exploited for very large data sets or iterative algorithms like K-means. We offer two sampling methods to approximate models [4]: geometric sampling and bootstrapping. Sampling is iterated until error in sufficient statistics is sufficiently small. By exploiting native DBMS sampling mechanisms, samples on large data sets can be collected in time an order of magnitude smaller than a table scan. Caching tables in RAM is applied whenever tables are small enough. This is the case for sufficient statistics matrices, model matrices and sample data subsets. A key optimization is to maintain a large sample in RAM to run the scoring script returned by the cloud. In summary, our system exploits both local and remote server CPU and server RAM, minimizing I/O locally and externally (in the cloud).

3. SYSTEM DEMONSTRATION

Our demonstration will illustrate our DBMS-based cloud service, showing how the local DBMS with large data sets can exploit the remote cloud DBMS server for fast processing and statistical model management. The local DBMS will be running on a portable computer and the cloud DBMS will be remotely available. All algorithms (summarization, sampling, model computation) will be available as UDFs and SPs. Both local DBMS and the cloud DBMS server will run Microsoft SQL Server on computers with 32 bit CPUs, 4 GB RAM and 1 TB on disk. Tables will be transferred with bulk utilities and DBMS query connections will use ODBC. Our system is programmed in C# in Microsoft SQL Server.

The local database will contain real data sets, coming from the UCI Machine Learning repository, replicating them (both on d and n) to get larger data sets. Specifically, the user will analyze large, high dimensional data sets, where n= 100k, 1M, 10M and $d = 10, 50, 100, 200$.

3.1 Demonstration Goals

We will emphasize these main points: (1) Providing data mining algorithms as a service in the cloud that is fast and easy to use (no complicated setup, no external packages needed). (2) Combining local and external processing; illustrating how to split processing between the local DBMS and the cloud, to overcome export I/O and network communication bottlenecks. (3) Understanding our linear cost model to decide processing mode. Comparing efficiency of three processing modes. Showing that external processing of large data sets in the cloud is a bottleneck due to transmission, uploading and I/O time. (4) Understanding interaction between data mining jobs requirements and job scheduler. The user will understand when processing goes from FIFO to SJF, and from SJF to RR, to provide good quality of service (QoS). (5) Understanding impact of optimizations, especially sampling and sufficient statistics acting as data compression mechanisms. (6) Emphasizing our system is different from most cloud proposals: we exploit DBMS technology instead of Hadoop. Moreover, we avoid transmitting large volumes of data to the cloud and there is a small time to start a job. (7) Model archive management, going beyond data management, enabled by keeping a history of past models and jobs, indexed by data set, columns, statistical model, model quality metrics and job parameters.

3.2 Demonstration Scenarios

The user will open a GUI to connect to a local DBMS and the cloud DBMS. The GUI will guide the user to navigate the data mining process step by step.

After connecting to the local DBMS and the cloud via ODBC the system will determine internal parameters for the cost model: CPU speed, RAM size, data set size and data mining model size. The user will initially pick a data set, input columns and a data mining model. The system will then measure local DBMS I/O speed, network speed and cloud DBMS I/O speed to read and transmit data. Based on these parameters the system will estimate cost and processing time for the three processing modes.

The user will compare the three processing modes: (a) Local processing, computing model locally, but storing the model in the cloud at the end. The user will understand how to quickly download/install SQL stored procedures and UDFs. However, all programs will be preinstalled. We will show sampling and caching in RAM matrices accelerate local processing. The user will observe local processing is best for small data sets and it does not exploit hardware from the cloud. (b) Cloud processing. Here the user will experience this processing mode is best for iterative algorithms, an overloaded local DBMS or to perform trial/error runs. The user will observe a slow upload operation to send a large data set to the cloud and then sending back the scored data set. That is, network transmission (Internet) and upload into the cloud are two bottlenecks, even if the data set is compressed. Thus the user will understand sampling on the local DBMS is needed (next scenario). Also, we will explain security issues and encryption overhead when the data set goes outside the firewall. Additionally, the user can compare the time it takes to load the data set or to load samples to MapReduce. (c) Hybrid processing, where the user will see an interesting interaction, step by step, sharing processing between the local DBMS and the cloud. The user will experience how sampling and caching in RAM accelerate local processing. We will show hybrid processing is best for algorithms with a fixed number of passes, and a low workload DBMS, and slow for iterative algorithms. The user will see sufficient statistics are locally computed quite fast, then they are quickly sent to the cloud, where models are computed. Finally, models and SQL queries for scoring are sent back, to evaluate accuracy. The user will understand the most time-consuming part is computing sufficient statistics and thus it makes sense to compute them locally. We will emphasize that since sufficient statistics are small matrices they can be transmitted fast. This processing mode will be shown to be competitive (marginally slower) with local processing, but simpler to manage.

This data mining job processing demonstration will conclude allowing the user to submit several demanding jobs. The user will set several ψ values going from 5 to 20 seconds, to understand interactive processing. The GUI will show how the overloaded cloud DBMS server switches from FIFO to SJF. The user will request data mining without sampling (accurate, but slow). Such longer jobs will make the system scheduling switch from SJF to RR, where several users can see progress and partial results on the data mining computation. To conclude the demonstration, we will demonstrate how our system extends data management to statistical model management. The user will try several simple queries to retrieve models based on data set, quality (assuming familiarity with data mining), date run, elapsed time and parameters. We will illustrate practical situations where the user needs to compare models, select best models, go back to a previous successful model or export it to an external statistical tool (e.g., the R package) for further processing and visualization.

Acknowledgments

This research work was partially supported by National Science Foundation grants IIS 0914861 and IIS 0915196. We thank Rogelio Montero-Campos, Julio Vega and José Luis Alvarez, from UNAM University, for helping in the development of our system.

4. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[2] M. Navas, C. Ordonez, and V. Baladandayuthapani. On the computation of stochastic search variable selection in linear regression with UDFs. In *Proc. IEEE ICDM Conference*, pages 941 – 946, 2010.

[3] C. Ordonez. Statistical model computation with UDFs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(12):1752–1765, 2010.

[4] C. Ordonez and S.K. Pitchaimalai. Fast UDFs to compute sufficient statistics on large data sets exploiting caching and sampling. *Data and Knowledge Engineering*, 69(4):383–398, 2010.

[5] M. Stonebraker, D. Abadi, D.J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin. MapReduce and parallel DBMSs: friends or foes? *Commun. ACM*, 53(1):64–71, 2010.