Dynamic estimation for medical data management in a cloud federation

Trung-Dung Le
Univ Rennes
CNRS, IRISA
Lannion, France
trung-dung.le@irisa.fr

Verena Kantere
University of Ottawa
School of Electrical Engineering and Computer Science
Ottawa, Canada
vkantere@uOttawa.ca

Laurent d’Orazio
Univ Rennes
CNRS, IRISA
Lannion, France
laurent.dorazio@irisa.fr

ABSTRACT

Data sharing is important in the medical domain. Sharing data allows large-scale analyses with many data sources to provide more accurate results (especially in the case of rare diseases with small local datasets). Cloud federations consist in a major progress in sharing medical data stored within different cloud platforms, such as Amazon, Microsoft, Google Cloud, etc. It also enables to access distributed data of mobile patients. The pay-as-you-go model in cloud federations raises an important issue in terms of Multi-Objective Query Processing (MOQP) to find a Query Execution Plan according to users preferences, such as response time, money, quality, etc. However, optimizing a query in a cloud federation is complex with increasing heterogeneity and additional variance, especially due to a wide range of communications and pricing models. Indeed, in such a context, it is difficult to provide accurate estimation to make relevant decision. To address this problem, we present Dynamic Regression Algorithm (DREAM), which can provide accurate estimation in a cloud federation with limited historical data. DREAM focuses on reducing the size of historical data while maintaining the estimation accuracy. The proposed algorithm is integrated in Intelligent Resource Scheduler, a solution for heterogeneous databases, to solve MOQP in cloud federations and validate with preliminary experiments on a decision support benchmark (TPC-H benchmark).

1 INTRODUCTION

Medical data sharing is full of promises. It allows large-scale medical data analysis to diagnose diseases more accurately. To reach this goal, the distributed clinics need to optimize queries on shared medical data with data sources in a cloud federation. For instance, in health-care, information of a given patient may be owned by different hospitals that may use various providers. Pay-as-you-go models in cloud federations and elasticity thus raise an important issue in terms of Multi-Objective Query Processing (MOQP) to find a Query Execution Plan (QEP) according to users preferences, such as time, money, quality, etc. However, optimizing queries in a cloud federation raises issues of heterogeneity and variability of cloud environment, such as wide-range communications and pricing models.

In variable environment like a cloud federation with various database systems, we should build a model to estimate the cost values for the MOQP. A cloud federation may rely on various hardware and systems. In addition, it also depends on the variety of physical machines, load evolution and wide-range communications. As a consequence, estimation is complex with the variability of environment. In this context, a challenging problem is how to estimate accurate values for MOQP without precise knowledge of execution environment in a cloud federation consisting of different sites.

Cost modeling can be classified into two classes: without [23, 26, 34] and with machine learning algorithms [11]. However, in a cloud federation with variability and different systems, cost functions may be quite complex. In the first class, cost models introduced to build optimal group of queries [23] are limited to MapReduce [8]. Besides, PostgreSQL cost model [34] aims to predict query execution time for this specific relational Data Base Management system. Moreover, OptEx [26] provides estimated job completion times for Spark [28] with respect to the size of the input dataset, the number of iterations, the number of nodes composing the underlying cloud. However, these papers only mention the estimation of execution time for a job, not for other metrics, such as monetary cost. Meanwhile, various machine learning techniques are applied to estimate execution time in recent researches [1, 15, 30, 35]. They predict the execution time by many machine learning algorithms. They treated the database system as a black box and try to learn a query running time prediction model using the total information for training and testing in the model building process. It may lead to the use of expired information. In addition, most of these solutions solve the optimization problem with a scalar cost value and do not consider multi-objective problems.

In this paper, we introduce a medical system on a cloud federation called Medical Data Management System (MIDAS). It is based on the Intelligent Resource Scheduler (ReS) [11], an open source platform for complex analytics workflows executed over multi-engine environments. In particular, we focus on Dynamic Regression Algorithm (DREAM) to provide accurate estimation with low computational cost. DREAM is then implemented and validated with preliminary experiments on a decision support benchmark (TPC-H benchmark [31]).

The remaining of this paper is organized as follows. Section 2 presents the research background. DREAM is presented in Section 3, while Section 4 presents experiments to validate DREAM. Finally, Section 5 concludes this paper and lists some perspectives.

2 BACKGROUND

Our work is a part of the MIDAS project, which aims to provide a data management system for cloud federation. In this section, we introduce an architecture of the system, concepts and techniques, allowing us to implement the proposed medical data management on a cloud federation.

First of all, an overview of MIDAS and the benefits of cloud federation where our system is built on are introduced. After that, an open source platform, which helps our system managing and executing workflows over multi-engine environments is described.
The concept of Pareto plan set related to MOQP in MIDAS is then defined. In addition, Multiple Linear Regression is also introduced as the basic foundation of our proposed algorithm for MOQP.

2.1 MIDAS

MIDAS is a medical data management system for cloud federation. The proposal aims to provide query processing strategies to integrate existing information systems (with their associated cloud provider and data management system) for clinics and hospitals. Figure 1 presents an overview of the system. Integrating the system within a cloud federation allows to choose the best strategy for MOQP. The different cloud resource pools allow the system to run in the most appropriate infrastructure environments. The system can optimize workflows between different data sources on different clouds, such as Amazon Web Services [3], Microsoft Azure [4] and Google Cloud Platform [16]. The proposed system is developed based on the Intelligent Resource Scheduler (iReS) for complex analytics workflows executed over multi-engine environments on a cloud federation.

2.2 Cloud federation

A cloud federation enables to interconnect different cloud computing environments. Cloud computing [2] allows to access on demand and configurable resources, which can be quickly made available with minimal maintenance. According to the pay-as-you-go pricing model, customers only pay for resources (storage and computing) that they use. Cloud Service Providers (CSP) supply a pool of resources, development platforms and services. There are many CSPs on the market, such as Amazon, Google and Microsoft, etc., with different services and pricing models. For example, Table 1 shows the pricing of instances in two cloud providers. The price of Microsoft instances is lower than the price of Amazon instances, but the price of Amazon is without storage. Hence, depending on the demand of a query, the monetary cost is lower or higher at a specific provider.

In medical domain, cloud federation may lead to query data on different clouds. For example, mobile patient data can be analyzed with many distributed sources of data to provide more accurate results. Data management in a cloud federation is thus a critical issue in terms of multi-engine environment and Multi-Objective Query Processing.

2.3 Pareto plan set

Let a query \( q \) be an information request from databases, presented by a set \( Q \) of tables. A Query Execution Plan (QEP) includes an ordered set of operators (select, project, join, etc.). The set of QEPs \( p \) of \( q \) is denoted by symbol \( P \). The set of operators is denoted by \( O \). A plan \( p \) can be divided into two sub-plans \( p_1 \) and \( p_2 \) if \( p \) is the result of function \( \text{Combine}(p_1, p_2, o) \), where \( o \in O \).

The execution cost of a QEP depends on parameters, which values are not known at the optimization time. A vector \( x \) denotes parameters value and the parameter space \( X \) is the set of all possible parameter vectors \( x \). In MOQP, \( N \) is denoted as the set of \( n \) cost metrics. We can compare QEPs according to \( n \) cost metrics which are processed with respect to the parameter vector \( x \) and cost functions \( c^i(x) \). Let denote \( C \) as the set of cost function.

Let \( p_1, p_2 \in P \) and \( p_1 \) dominates \( p_2 \) if the costs according to each cost metric of plan \( p_1 \) is less than or equal to the corresponding values of plan \( p_2 \) in all the space of parameter \( X \). That is to say:

\[
C(p_1, X) \leq C(p_2, X) \quad \forall n \in N, \forall x \in X : c^n(p_1, x) \leq c^n(p_2, x).
\]

The function \( \text{Dom}(p_1, p_2) \subseteq X \) yields the parameter space region where \( p_1 \) dominates \( p_2 \) [32]:

\[
\text{Dom}(p_1, p_2) = \{x \in X \mid \forall n \in N : c^n(p_1, x) \leq c^n(p_2, x)\}.
\]

Assume that in the area \( x \in A, A \subseteq X \), \( p_1 \) dominates \( p_2 \) if \( C(p_1, A) \leq C(p_2, A) \). Let \( \text{Dom}(p_1, p_2) = A \subseteq X \). \( p_1 \) strictly dominates \( p_2 \) if all values for the cost functions of \( p_1 \) are less than the corresponding values for \( p_2 \) [32], i.e.

\[
\text{StriDom}(p_1, p_2) = \{x \in X \mid \forall n \in N : c^n(p_1, x) < c^n(p_2, x)\}.
\]

A Pareto region of a plan is a space of parameters where there is no alternative plan has lower cost than it [32]:

\[
\text{PaReg}(p) = X \setminus \left( \bigcup_{p^* \in P} \text{StriDom}(p^*, p) \right)
\]

2.4 IReS

Intelligent Multi-Engine Resource Scheduler (iReS) [11] is an open source platform for managing, executing and monitoring complex analytics workflows. IReS provides a method of optimizing cost-based workflows and customizable resource management of diverse execution and various storage engines. Interface is the first module which is designed to receive information on data and operators, as shown in Figure 1. The second module is Modelling, as shown in Figure 1, is used to predict the execution time by a model chosen by comparing machine learning algorithms. For example, Least squared regression [25], Bagging predictors [6], Multilayer Perceptron in WEKA framework [33] are used to build the cost model in Modelling module. The module tests many algorithms and the best model with the smallest error is selected. It guarantees the predicted values as the best one for estimating process. Next module, Multi-Objective Optimizer, optimizes MOQP and generates a Pareto QEP set. In Multi-Objective problem, the objectives are the cost functions user concerned, such as the execution time, monetary, intermediate data, etc. Multi-Objective Optimization algorithms can be applied to the Multi-Objective Optimizer. For instance, the algorithms based on Pareto dominance techniques [7, 9, 10, 19, 21, 22, 29, 36, 37]
are solutions for Multi-objective Optimization problems. Finally, the system selects the best QEP based on user query policy and Pareto set. The final query plan is run on multiple engines, as shown in Figure 1.

2.5 Multiple Linear Regression

A cost function of Multiple Linear Regression (MLR) model [27] is defined as:

\[ c = \beta_0 + \beta_1 x_1 + \ldots + \beta_L x_L + \epsilon, \]  

where \( \beta_l, l = 0, \ldots, L \), are unknown coefficients, \( x_l, l = 1, \ldots, L \), are the independent variables, e.g., size of data, computer configuration, etc., \( c \) is cost function values and \( \epsilon \) is random error following normal distribution \( N(0, \sigma^2) \) with zero mean and variance \( \sigma^2 \). The fitted equation is defined by:

\[ \hat{c} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_L x_L. \]  

**Example 2.1.** A query \( Q \) could be expressed as follows:

```sql
SELECT p.PatientSex, i.GeneralNames
FROM Patient p, GeneralInfo i
WHERE p.UID = i.UID
```

where Patient table is stored in cloud A and uses Hive database engine [18], while GeneralInfo table is in cloud B with PostgreSQL database engine [24]. This scenario leads to concern two metrics of monetary cost and execution time cost. We can use the cost functions which depend on the size of tables of Patient and GeneralInfo. Besides, the configuration and pricing of virtual machines in cloud A and B are different. Hence, the cost functions depend on the size of tables and the number of virtual machines in cloud A and B.

\[ \hat{c}^t = \hat{\beta}_0 + \hat{\beta}_1 x_{Pa} + \hat{\beta}_2 x_{Ge} + \hat{\beta}_3 x_{nodeA} + \hat{\beta}_4 x_{nodeB} \]

\[ \hat{c}^m = \hat{\beta}_m0 + \hat{\beta}_m1 x_{Pa} + \hat{\beta}_m2 x_{Ge} + \hat{\beta}_m3 x_{nodeA} + \hat{\beta}_m4 x_{nodeB} \]

where \( \hat{c}^t, \hat{c}^m \) are execution time and monetary cost function; \( x_{Pa}, x_{Ge} \) are the size of Patient and GeneralInfo tables, respectively, and \( x_{nodeA}, x_{nodeB} \) are the number of virtual machines created to run query \( Q \).

There are \( M \) historical data, each of them associates with a response \( e_m \), which can be predicted by a fitted value \( \hat{e}_m \) calculated from corresponding \( x_{Im} \) as follows:

\[ \hat{e}_m = \hat{\beta}_0 + \hat{\beta}_1 x_{I1} + \ldots + \hat{\beta}_L x_{IL}; m = 1, \ldots, M. \]  

Let denote

\[ A = \begin{bmatrix} x_{11} & x_{21} & \ldots & x_{1L} \\ x_{12} & x_{22} & \ldots & x_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1M} & x_{2M} & \ldots & x_{LM} \end{bmatrix}, \]

\[ C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix}, \]

\[ B = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_L \end{bmatrix} \]

To minimize the Sum Square Error (SSE), defined by:

\[ SSE = \sum_{m=1}^{M} (e_m - \hat{e}_m)^2, \]

the solution for \( B \) is retrieved by:

\[ B = (A^T A)^{-1} A^T C. \]

### 2.6 Motivation

Our proposed method is integrated into **Modelling** module to predict the cost values with low computational cost in MOQP of a cloud environment. However, the machine learning algorithms in **Modelling** module of IReS need entire of training datasets. It may lead to use expired information. Hence, the proposal algorithm aims to improve the accuracy of estimated values with low computational cost.

In addition, MOQP could be solved by Multi-objective Optimization algorithms or the Weighted Sum Model (WSM) [17]. However, Multi-objective Optimization algorithms may be selected thanks to their advantages when comparing with WSM. The optimal solution of WSM could be not acceptable, because of an inappropriate setting of the coefficients [13]. Furthermore, the research in [20] proves that a small change in weights may result in significant changes in the objective vectors and significantly different weights may produce nearly similar objective vectors. Moreover, if WSM changes, a new optimization process will be required. Hence, our system applies a Multi-objective Optimization algorithm to the Multi-Objective Optimizer to find a Pareto-optimal solution.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Machine</th>
<th>vCPU</th>
<th>Memory (GiB)</th>
<th>Storage (GiB)</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>a1.medium</td>
<td>1</td>
<td>2</td>
<td>EBS-Only</td>
<td>$0.0049/hour</td>
</tr>
<tr>
<td></td>
<td>a1.large</td>
<td>2</td>
<td>4</td>
<td>EBS-Only</td>
<td>$0.0098/hour</td>
</tr>
<tr>
<td></td>
<td>a1.xlarge</td>
<td>4</td>
<td>8</td>
<td>EBS-Only</td>
<td>$0.0197/hour</td>
</tr>
<tr>
<td></td>
<td>a1.2xlarge</td>
<td>8</td>
<td>16</td>
<td>EBS-Only</td>
<td>$0.0394/hour</td>
</tr>
<tr>
<td></td>
<td>a1.4xlarge</td>
<td>16</td>
<td>32</td>
<td>EBS-Only</td>
<td>$0.0788/hour</td>
</tr>
<tr>
<td>Microsoft</td>
<td>B1S</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>$0.011/hour</td>
</tr>
<tr>
<td></td>
<td>B1MS</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>$0.021/hour</td>
</tr>
<tr>
<td></td>
<td>B2S</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>$0.042/hour</td>
</tr>
<tr>
<td></td>
<td>B2MS</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>$0.084/hour</td>
</tr>
<tr>
<td></td>
<td>B4MS</td>
<td>4</td>
<td>16</td>
<td>32</td>
<td>$0.166/hour</td>
</tr>
<tr>
<td></td>
<td>B8MS</td>
<td>8</td>
<td>32</td>
<td>64</td>
<td>$0.333/hour</td>
</tr>
</tbody>
</table>
In conclusion, our solution aims to improve the accuracy of cost value prediction with low computational cost and to solve MOQP by Multi-objective Optimization algorithm in a cloud federation environment. To provide accurate estimation while reducing the number of previous measures, our algorithm is proposed based on Multiple Linear Regression.

3 DYNAMIC REGRESSION ALGORITHM

Most of cost models [12, 23, 34] depend on the size of data. Hence, our cost functions are functions of the size of data. In particular, cost function and fitted value of Multiple Linear Regression model are previously defined in Section 2.5. The bigger \( M \) for sets \( \{x_m, x_{lm}\} \) is, the more accurate MLR model usually is. However, the computers is slowing down when \( M \) is too big.

Furthermore, the target of Multi-Objective Query Processing is the Multi-Objective Optimization Problem [36], which is defined by:

\[
\text{minimize}(F(x) = (f_1(x), f_2(x), \ldots, f_K(x))^T),
\]

where \( x = (x_1, \ldots, x_L)^T \in \Omega \subseteq \mathbb{R}^L \) is an \( L \)-dimensional vector of decision variables, \( \Omega \) is the decision (variable) space and \( F \) is the objective vector function, which contains \( K \) real value functions.

In general, there is no point in \( \Omega \) that minimizes all the objectives together. Pareto optimality is defined by trade-offs among the objectives. If there is no point \( x \in \Omega \) such that \( F(x) \) dominates \( F(x^*) \), \( x^* \in \Omega \), \( x^* \) is called Pareto optimal and \( F(x^*) \) is called a Pareto optimal vector. Set of all Pareto optimal points is the Pareto set. A Pareto front is a set of all Pareto optimal objective vectors. Generating the Pareto-optimal front can be computationally expensive [5]. In cloud environment, the number of equivalent query execution plans is multiplied.

**EXAMPLE 3.1.** Assuming that a query is processed on Amazon EC2. If the pool of resources includes 70 vCPU and 260GB of memory, the number of different configurations to execute this query is thus around 70 \( \times \) 260 = 18,200. Hence, the system can generate 18,200 equivalent QEPs from a given execution plan.

Example 3.1 shows that a query execution plan can generate multiple equivalent QEPs in cloud environment. The smaller \( M \) for sets \( \{c_m, x_{lm}\} \) is, the faster speed for the estimation cost process of Multi-Objective Query Processing for a QEP is. In the system of computationally expensiveness in cloud environment as in Example 3.1, a small reduction of computation for an equivalent QEP estimation will become significant for a large number of equivalent QEPs estimation.

The most important idea is to estimate MLR quality by using the coefficient of determination. The coefficient of determination [27] is defined by:

\[
R^2 = 1 - \frac{\text{SSE}}{\text{SST}},
\]

where \( \text{SSE} \) is the sum of squared errors and \( \text{SST} \) represents the amount of total variation corresponding to the predictor variable \( X \). Hence, \( R^2 \) shows the proportion of variation in cost given by the Multiple Linear Regression model of variable \( X \). For example, the model gives \( R^2 = 0.75 \) of time response cost, it can be concluded that 3/4 of the variation in time response values can be explained by the linear relationship between the input variables and time response cost. Table 2 presents an example of MLR with different number of measures. The smallest dataset is \( M = L + 2 = 4 \) [27], where \( M \) is the size of previous data and \( L \) is the number of variables in Equation (5). In general, \( R^2 \) increases in parallel with \( M \). In particular, \( R^2 \) should be greater than 0.8 to provide a sufficient quality of service level. As a consequence, \( M \) should be greater than 5 to provide enough accuracy. Hence, when the system requires the minimum values of \( R^2 \) is equal to 0.8, \( M > 6 \) is not recommended. In general, \( R^2 \) still rises up when \( M \) goes up. Therefore, we need to determine the model which is sufficient suitable by the coefficient of determination.

Our motivation is to provide accurate estimation while reducing the number of previous measures based on \( R^2 \). We thus propose DREAM as a solution for cloud federation and their inherent variance, as shown in Table 2. DREAM uses the training set to test the size of new training dataset. It depends on the predefined coefficient of determination. The new training set is generated in order to have the updated value and avoid using the expired information. With the new training set, **Modelling** uses less data in building model process than the original approach.

Cost modeling without machine learning [23, 26, 34] often uses the size of data to estimate the execution time for the specific system. Besides, the machine learning approach [11, 33] can use any information to estimate the cost value. Hence, our algorithm uses the size of data as variables of DREAM. In (6), \( \hat{c} \) is the cost value, which needs to be estimated in MOQP, and \( x_1, x_2, \ldots \) are the information of system, such as size of input data, the number of nodes, the type of virtual machines. If \( R^2 \geq R^2_{\text{require}} \), where \( R^2_{\text{require}} \) is predefined by users, the model is reliable. In contrast, it is necessary to increase the number of set value. Algorithm 1 shows a scheme as an example of increasing value set: \( m = m + 1 \).

In this paper, we focus on the accuracy of execution time estimation with the low computational cost in MOQP. The original optimization approach in IReS uses Weighted Sum Model [17] with user policy to find the best candidate. However, Multi-objective Optimization algorithms have more advantages than WSM [13, 20]. Hence, after having a set of predicted cost function values for each query plan, a Multi-objective Optimization algorithm, such as Non-dominated Sorting Genetic Algorithm II [10] is applied to determine a Pareto plan set. At the final step, the weight sum model \( S \) and the constraint \( B \) associated with the

<table>
<thead>
<tr>
<th>Cost</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( M )</th>
<th>( R^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.640</td>
<td>0.4916</td>
<td>0.2977</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.557</td>
<td>0.6313</td>
<td>0.0482</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.971</td>
<td>0.9481</td>
<td>0.8232</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.878</td>
<td>0.4855</td>
<td>2.7056</td>
<td>4</td>
<td>0.7571</td>
</tr>
<tr>
<td>23.274</td>
<td>0.0125</td>
<td>2.7268</td>
<td>5</td>
<td>0.7705</td>
</tr>
<tr>
<td>30.216</td>
<td>0.9029</td>
<td>2.6456</td>
<td>6</td>
<td>0.8371</td>
</tr>
<tr>
<td>29.978</td>
<td>0.7233</td>
<td>3.0640</td>
<td>7</td>
<td>0.8788</td>
</tr>
<tr>
<td>31.702</td>
<td>0.8749</td>
<td>4.2847</td>
<td>8</td>
<td>0.8876</td>
</tr>
<tr>
<td>20.860</td>
<td>0.3354</td>
<td>2.1082</td>
<td>9</td>
<td>0.8751</td>
</tr>
<tr>
<td>32.836</td>
<td>0.8521</td>
<td>4.8217</td>
<td>10</td>
<td>0.8945</td>
</tr>
</tbody>
</table>

**Figure 2:** DREAM module.
Algorithm 1 Calculate the predict value of multi-cost function

1: function ESTIMATECOSTVALUE(R^2_{require}, X, M_{max})
2:   for n = 1 to N do
3:     R^2_n ← 0 //with all cost function
4:   end for
5:   m = L + 2 //at least m = L + 2
6:   while (any R^2_n < R^2_{n-require}) and m < M_{max} do
7:     for c_n(p) ≤ 8 do
8:       R^2_n = 1 - SSE/SST
9:       ˆc_n = β_n0 + β_n1x_1 + ... + β_nLx_L
10:   end for
11:   m = m + 1
12: end while
13: return ˆc_n(p)
14: end function

Algorithm 2 Select the best query plan in \( P \)

1: function BESTINPARETO(P, S, B)
2:   P_B ← p ∈ P \{n ≤ |B| : c_n(p) ≤ B_n
3:   if P_B ≠ ∅ then
4:     return p ∈ P \{c(p) = min(WeightSum(P_B, S))
5:   else
6:     return p ∈ P \{c(p) = min(WeightSum(P, S))
7:   end if
8: end function

Figure 3: Comparing two MOQP approaches

4 EVALUATION

DREAM has been implemented on top of IReS platform. It has been validated with experiments.

4.1 Implementation

Our experiments are executed on a private cloud [14] with a cluster of three machines. Each node has four 2.4 GHz CPU, 80 GiB Disk, 8 GiB memory and runs 64-bit platform Linux Ubuntu 16.04.2 LTS. The system uses Hadoop 2.7.3 [24], Hive 2.1.1 [18], PostgreSQL 9.5.14 [24], Spark 2.2.0 [28] and Java OpenDK Runtime Environment 1.8.0. IReS platform is used to manage data in multiple database engine and deploy the algorithms.

4.2 Experiments

TPC-H benchmark [31] with two datasets of 100MB and 1GB is used to have experiments with DREAM. Experiments with TPC-H benchmark are executed in a multi-engine environment consisting of Hive[18] and PostgreSQL[24] deployed on a private cloud [14]. In TPC-H benchmark, the queries related to two tables are 12, 13, 14 and 17. These queries with two tables in two different databases, such as Hive and PostgreSQL, are studied.

4.3 Results

To estimate the quality of DREAM in comparison with other algorithms, Mean Relative Error (MRE), a metric used in [1] is used and described as below:

\[
\frac{1}{M} \sum_{i=1}^{M} \left| \hat{c}_i - c_i \right|, \quad (15)
\]

where \( M \) is the number of testing queries, \( \hat{c}_i \) and \( c_i \) are the predict and actual execution time of testing queries, respectively. IReS platform uses multiple machine learning algorithms in their model, such as Least squared regression, Bagging predictors, Multilayer Perceptron.

In IReS model building process, IReS tests many algorithms and the best model with the smallest error is selected. It guarantees the predicted values as the best one for estimating process. DREAM is compared to the Best Machine Learning model (BML) in IReS platform with many observation window \( N, 2N, 3N \) and no limit of history data. The smallest size of a window, \( N = L + 2 \) [27], where \( L \) is the number of variables, is the minimum data set DREAM requires. As shown in Tables 3 and 4, MRE of DREAM are the smallest values between various observation windows. In our experiments, the size of historical data, which DREAM uses, are very small, around \( N \).

5 CONCLUSION

This paper is about medical data management in cloud federation. It introduces Dynamic Regression Algorithm (DREAM) as a part of MIDAS and on top of IReS, an open source platform for
complex analytics work-flows executed over multi-engine environments. DREAM aims to address variance in a cloud federation and to provide accurate estimation for MOQP. Preliminary results with DREAM and TPC-H benchmark are quite promising with respect to existing solutions.

In the future, we plan to validate our proposal with more cloud providers (and their associated pricing model and services) and data management systems. We will also define new strategies to choose QEPs in a Pareto Set.

ACKNOWLEDGMENT

The authors would like to thank members of SHAMAN team at Univ Rennes, CNRS, IRISA and University of Ottawa School of Electrical Engineering and Computer Science for insightful comments.

REFERENCES


Table 4: Comparison of mean relative error with 1GiB TPC-H dataset.

<table>
<thead>
<tr>
<th>Query</th>
<th>BML-N</th>
<th>BML-2N</th>
<th>BML-3N</th>
<th>BML</th>
<th>DREAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0.349</td>
<td>0.854</td>
<td>0.341</td>
<td>0.480</td>
<td>0.335</td>
</tr>
<tr>
<td>13</td>
<td>0.396</td>
<td>0.843</td>
<td>0.457</td>
<td>0.487</td>
<td>0.349</td>
</tr>
<tr>
<td>14</td>
<td>0.468</td>
<td>0.664</td>
<td>0.539</td>
<td>0.790</td>
<td>0.318</td>
</tr>
<tr>
<td>17</td>
<td>0.620</td>
<td>0.611</td>
<td>0.681</td>
<td>0.970</td>
<td>0.536</td>
</tr>
</tbody>
</table>


