Parameter estimation of price-demand model for cloud data services

Sofia Kyriakopoulou
École Polytechnique Fédérale de Lausanne
1015 Lausanne VD, Switzerland
sofia.kyriakopoulou@epfl.ch

ABSTRACT

In this report we describe the module developed to complete and extend the existing implementation that offers optimal pricing of data services in a cloud DBMS. We investigate the Inverse Optimal pricing model problem (IOPM) aiming to estimate the parameters of an existing pricing scheme. Solving the Inverse problem requires measured values from the cloud DBMS. The dimensionality of the data search space poses a challenge for the creation of a dataset of representative values. We propose a method for drawing representative samples from the cloud DBMS adapting the Metropolis Monte Carlo algorithm and tackle the dimensionality problem using a theoretically optimal linear scheme for dimensionality reduction, Principal Component Analysis. For a finite data set, we solve the Inverse pricing model problem using a deterministic method for non-linear optimization. The solution is implemented as a stand-alone, full- fledged module, named IOPM Module, comprised by a Sampling component that adopts the proposed sampling method and an Optimization component that solves the IOPM problem. Furthermore, we demonstrate how the module was integrated into the existing schema. The efficiency of the solution to the IOPM problem is shown in an experimental study.

1. INTRODUCTION

A novel pricing scheme has been proposed for a cloud cache, aiming at the maximization of cloud profit. In [15], the optimal pricing problem has been formulated and solved, by a price-demand model that estimates the correlations of the data services in a time efficient manner. External tools have been used by the authors in order to perform data regression of pairs of price and demand values that are output of a simulated cloud DBMS that has been developed in the DIAS laboratory.

This project completes and extends the existing implementation in order to produce a full-fledged and stand-alone module for optimal pricing, addressing the need for efficient estimation of the proposed model parameters (fitting)[1]. There are two major challenges when trying to estimate the model parameters. The first is to obtain a representative dataset suitable for the model fitting. The correlations the model takes into account cause a burst in dimensionality of the dataset. A combination of sampling methods can be used to create a representative dataset and solve the dimensionality problem. Monte Carlo Sampling is selected in order to draw samples from a simulation of the cloud cache Principal Component Analysis is picked, with a view to transform the number of possibly correlated model variables into a smaller number of uncorrelated variables. The second challenge resides in the procedure that solves the parameter estimation. The estimation of the price-demand model parameters is translated into the IOPM problem. We apply optimal control theory and use data regression methods to solve it. The solution involves the construction of a sum-of-squares function of the residuals between the measured and the simulated data. The estimation of the true parameters is the values that minimize the sum-of-square function. The minimization of this sum-of squared function is carried out using numerical optimization techniques. In order to achieve the best possible solution to the optimization problem, we search through a series of solvers that are available within the optimization tools.

[1] Within this project, the terms parameter estimation and fitting will be used interchangeably

1.1 Problem Motivation

The maximization of cloud profit necessitates a price-demand model that enables optimal pricing of query services. A price-demand scheme has already been designed [15] in DIAS to offer a feasible optimal pricing solution. The authors introduce a static relation between price and demand. The model proposed constraints on price and demand using a set of parameters. The model parameters are the set of values that completely characterize the cloud caching service. The Parameterization of the model is the discovery of this set of values and is crucial for the model's accuracy in the prediction of price values. .

Solving the Inverse problem of Optimal Pricing, i.e. the estimation of parameters from measured values of price and demand requires a representative subset data set of measured values. A representative dataset for the data services in a cloud DBMS can be drawn from a simulation of the cloud cache.

The dimension of the search space for the dataset in terms of the dimension of the variable space grew by the modeling approach of the authors in [15]. In order to make the pricing model realistic, the authors consider the correlations of demand and prices between pairs of structures.

The question then arises as to whether there is a method that one can use to create such a dataset when the model involves variables in a large number of dimensions. The answer is that in the extent of our knowledge, no known method exists for such purpose used in combination with a simulation.

We develop a module that effectively performs parameter estimation and is interoperable with the existing price-demand model implementation. The already implemented price-demand model needs to be open to corrections, while its optimization is still in progress to achieve a flexible long-term optimization. These corrections will mainly address the difference between the estimated and the actual price influence on the demand of services. The estimation of the model parameters should be repeated in each optimization iteration, since the efficiency and flexibility of the solution offered by the pricing model in [15] is sensitive to the price adaptivity to time changes.

1.2 Related Work

There is significant related work in the areas of optimization used for fitting in various research fields. Additionally, Monte Carlo methods have been applied before in distributed computing, through a different perspective.

Optimization has been used for Parameter estimation problems widely in various fields of research, from Computer Vision [16], to Systems Biology [4] and Telecommunications [5]. However, the parameter estimation problem is model specific and each case bears limited resemblance to another.

Monte Carlo techniques are utilized in economic approaches for distributed computing. In [14], an approach is suggested for the utilization of idle computational resources in a heterogeneous cluster. The authors used concurrent Monte Carlo simulations as prototypical applications, to explore issues of fairness in resource distribution, currency as a form of priority, price equilibria and scaling to large systems. Although the common properties are the domain of usage of Monte Carlo methods, our approach utilizes Monte Carlo methods for random sampling of probability distribution functions as model inputs to produce possible outcomes instead of a few discrete scenarios.

1.3 Our Proposal

A combination of methods is applied in order to achieve the best possible solution to the IOPM problem There is no sampling-fitting method that is general purpose. Although the idea that a single method would be best for all problems is an attractive one, it is in fact infeasible due to the great variety of problem domains and their properties.

We extend a Monte Carlo sampling algorithm as the sampling strategy used to obtain a dataset from the simulation and apply Principal Component Analysis to diminish the dimensions of the obtained dataset. This dataset will be the input of measured data for the optimization procedure.

We have programmed an optimization procedure that solves the inverse parameter estimation problem, using non-linear deterministic methods.

The sampling-optimization method is implemented as one module, designed to be integrated with the existing implementation.

1.4 Contributions

This project makes the following contributions:

IOPM module. The stand-alone module IOPM (Inverse Optimal Pricing Model Module) is designed to assure both the flexibility and the accuracy of the solution offered by the pricing model. The IOPM module enables:

- The estimation of the initial parameters in the price-demand model for a large-scale structures collection, through its Optimization component.
- The representative sampling from the structures population of the cloud DBMS. This is a procedure is performed into the Sampling component.
- The integration of the optimal pricing solution and price-demand model so that it is possible to re-inject real demand values in the optimization process

A data regression method. The parameter estimation cannot be efficient unless a representative dataset is obtained. The data regression method we propose extends the Metropolis Monte Carlo algorithm to gather samples from the simulation and combines it with the Principal Component Analysis procedure to reduce the dimensions of the extracted dataset.

An experimental study. An experimental study is conducted to show the efficiency of the solution, comparing minimum error rate among different solvers for the parameter estimation problem.

The rest of the report is structured as follows. Section 2 describes the formulation and the solution of the Inverse Optimal Pricing Model problem. Section 3 describes the implementation of the module integration into the existing schema. Section 4 presents the experimental study. Section 5 discusses our proposal and Section 6 concludes this report.

2. THE INVERSE PROBLEM OF OPTIMAL PRICING MODEL

The cloud makes profit from selling its services at a price higher than the cost. Yet, pricing the services is a non-trivial problem because the competition for services leads the demand to a growth non-proportional to the price. A Price-demand model for the cloud cache [15] has been proposed, that takes into account the collaborative or competitive behavior of structures during query execution. The demand for a structure depends not only on its price, but also on the price of other structures.

Since this project coalesces the existing research by the authors of [15], in this section we will refer to the definitions formulated in [15] and are the basis of our work. We will next introduce properly in 2.1 the Inverse Optimal Pricing Model problem that this project deals with and describe the solution approach we selected. We will present in 2.2 a sampling method that enables the Data Regression of the price-demand model.

2.1 Design of the IOPM Module

2.1.1 Preliminaries

It is assumed that data are stored in a cloud database and that cloud caching supports efficient query processing. In the current caching infrastructure the columns of the tables in the back-end database are cached. The cloud infrastructure provides unlimited amount of storage space and therefore any number of structures can be held in the cache. The cloud cache offers to the user query services on the cloud data. For each query, the user is presented with query plans that include cache structures, i.e. cache columns and indexes. A set of possible cache structures is $S=\{S_1,\ldots,S_m\}$.

The demand. The demand λ_s for a structure S is defined as the number of occurrences of the fact that S is included in the query plans selected for execution at time t.

The cost. Once a structure S is built in the cache it has a one-time building cost B_s. A sturcture S has a time-dependant maintenance cost M_s, while maintained in the cache The cost of the structure S from the time it is built in the cache, defined as b_{uilt} until it is discarded is:

$$c_s(t) = B_s + M_s(t - t_{built}) \quad (1)$$

The price. The cloud makes profit charging the user for each structure included in a plan a price $p_s(t)$.

The demand and price for one structure are connected in an ordinary differential equation as follows:

$$p_s(t) = a\frac{d^2\lambda_s}{dt^2} + \beta\frac{d\lambda_s}{dt} + \gamma\lambda_s \quad (2)$$

In the price-demand static relation, three constant parameters interfere: α, β, γ.

A correlation V matrix is defined to capture the correlations of demand and prices between structures. The one structure price-demand equation was expanded to:

$$V \cdot P = A^T \frac{d^2\Lambda}{dt^2} + B^T \frac{d\Lambda}{dt} + \Gamma^T \Lambda \quad (3)$$

Λ and P are the mx1 matrices of demand and prices for the respective structures in S and A, B, Γ are mx1 matrices of parameters.

Before proceeding any further, we would like to clarify some potentially confusing issues of language that arise in this combined approach. This is related to the words 'parameter' and 'variable', which are used in simulation and optimization with perhaps confusing meanings. In modeling and simulation parameters and variables are distinguished, the parameters being entities of a model that are either constant, under direct control or vary independently. Variables are entities whose values are entirely determined by the parameters. While in optimization these words are used interchangeably, the entities that are parameters and variables are in our case exactly the opposite: the variables of the price-demand model [15] are now part of the objective function (the function to be minimized) and the parameters to optimize become variables (because they are varied in the course of the optimization)—this is why our problem is named inverse problems. To avoid confusion, we will always refer to the entities that are parameters in the price-demand model [15] as 'parameters' and the variables in the price-demand model as 'variables'; when we refer to the inverse model parameters that are being optimized, we call them 'adjustable parameters' to stress the fact that their values are going to be adjusted by the optimization method. We must also point out that the optimization literature refers to these as 'variables' as they are effectively varied during the course of the optimization. In addition, the numerical optimization methods themselves can be tuned by a few parameters that are only involved with the way the method proceeds (e.g. step sizes, horizon); we will not refer to these within this project.

2.1.2 Problem Formulation

Mathematically, studying a system given a model that describes its behavior such as (3) does and trying to use the actual results of some measurements of the observable variables to infer the actual values of the model parameters is characterized as an *Inverse Model Problem* [19], named for the cloud DBMS economy modeling as Inverse Optimal Pricing Model (IOPM) problem. The choice of model parameters to be used to describe a system is generally not unique. A particular choice of model parameters is a *parameterization* of the system.

Model Space It is possible to introduce an abstract set of points, a *manifold*, each point of which represents a conceivable model of the system. This manifold is named the *model space* and is denoted by *M*. Once a parameterization is chosen, with each point M of the model space *M* a set of numerical values $\{m^1,\ldots,m^n\}$ is associated. Each point M of *M* is named a *model and represented with m.*

Data space The space of all conceivable results of the measurements corresponds to a *data manifold*, which may be represented with the symbol *D*. Any exact result of the measurements corresponds to a particular point D on the manifold D. The coordinates $\mathbf{d}=\{d^i\}$ are the possible realizations and are named *data vectors*.

In (3) we observe the *forward problem.* Solving the forward problem means predicting the error-free values of the observable variables **d** that correspond to the model **m**. The model parameters needed to completely describe the system are $\{m^1,m^2,m^3\}=\{A,B,\Gamma\}$. To obtain information on model parameters we have to perform some observations during an experiment, i.e. we have to perform a measurement of observable coordinates $\{d^1,d^2\}=\{\Lambda,P\}$. Let:

$$\begin{aligned} d^1 &\rightarrow \Lambda = \{\lambda_1,\ldots,\lambda_m\} \\ d^2 &\rightarrow P = \{[p_{11},\ldots,p_{1m}],\ldots,[p_{m1},\ldots,p_{mm}]\} \end{aligned} \quad (4)$$

The relation between **d** and **m** as exhibited in (3) is clearly not linear. Assume that when a measurement is completed, the simulation delivers a given value of **d**, denoted $\mathbf{d}_{\mathbf{meas,}}$. The data coordinates d represent pairs of observed values of demand over the set of prices of structures, **i.e**.

$$d_{meas} = (\Lambda_{meas}, P_{meas}) \quad (5)$$

The estimation of the parameters of the price-demand model is achieved with the solution of the following optimization problem:

$$\min Err(t) = \sum_t ((\Lambda(t) - \Lambda^{meas}(t))^2 \quad (6)$$

subject to constraints:

$$0 \le \lambda_i, i = 1,\ldots,m$$

$$V \cdot P^{meas} = A^T \frac{d^2\Lambda}{dt^2} + B^T \frac{d\Lambda}{dt} + \Gamma^T \Lambda$$

The above problem is an optimal control problem with a finite horizon. The state variables are the price P and the demand Λ, and the control variables are the parameters A, B, Γ. The mx1 matrices of demand defined as Λ^{meas} is one of the coordinates of the data vector.

A full exploration of the data space is generally too expensive to make and creates a data vector, referred to as $d_{obs.}$ d_{obs} has coordinates in large dimensions for a large set of structures S_i,i=1,…,n . We are therefore left with the possibility to construct a data vector, denoted as *dataset*, defined over a space of lower dimension. This *curse of dimensionality* means that the systematic process for solving (6) is really applicable only to rather small dimensions of the data vector coordinates. In mathematical terms, the problem we investigate can be stated as follows: given the m-dimensional datavector, where i corresponds to the structure S_i,i=1,..,n ,

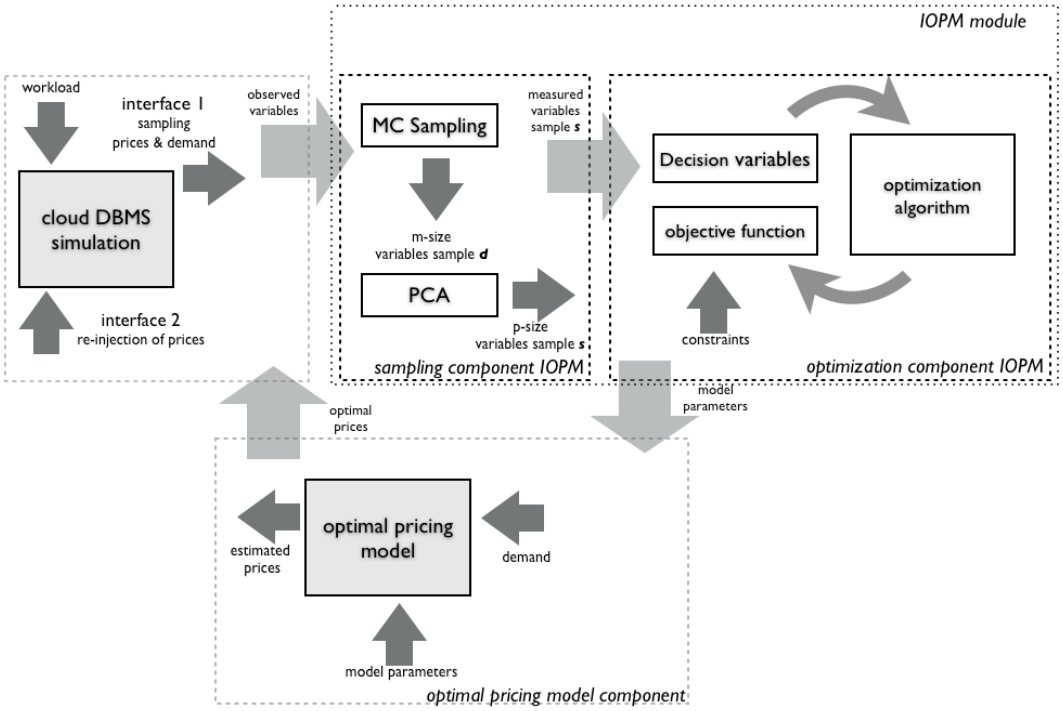
$$d = (\lambda_i, [p_{i1},\ldots,p_{in}])^T, \quad (7)$$

find a lower dimensional representation of it,

$$s = (\lambda_i, [p_{i1},\ldots,p_{ik}])^T \quad (8)$$

with $k \le n$, that captures the content in the original data, according to some criterion.

2.1.3 Problem Solution



We need to perform effective measurement of the observable variables, reduce the dimensions of the dataset and utilize the to be preferred and applied once the datavectors' dimensions is reduced.

Figure 1: The IOPM Module integrated with the existing implementation

diminished dataset as input to solve the Inverse Optimal Price Model problem, as formulated in 2.1.2

The measurement of the observable coordinates is conducted in the cloud simulation. A representative dataset **x** can be drawn from the cloud DBMS structures population using the Metropolis algorithm, as will be explained in detail in 2.2.1.

Principal component analysis (PCA) is the best, in the mean-square error sense, linear dimension reduction technique [10]. In essence, using PCA we reduce the dimension of the dataset by finding a few orthogonal linear combinations (the PCs) of the original datavectors coordinates with the largest variance.

We have

$$s = Wx \quad (9),$$

where $W_{k\times n}$ is the linear transformation weight matrix.

The first PC, s_1 ,is the linear combination with the largest variance. We have

$$s_1 = x^T w_1 \quad (10)$$

where the m-dimensional coeffcient vector $\mathbf{w_1} = (w_{1,1}, \ldots, w_{1,n})^T$ solves

$$w_1 = \arg\max_{\|w=1\|} Var\{x^T w\} \quad (11)$$

The popular linear programming methods cannot be applied to this problem due to their requirement of objective function linearality in terms of the adjustable parameters. Non-linear methods are thus

The solution of the inverse problem can be performed in a fully non-linear way. The problem is convex, therefore the solution is a global minimum. The problem of minimizing the square error as formulated in (6) was solved based on a deterministic method for non-linear optimization. The strategy used to scan the data space of the price-demand model is to repeat the solution of the ODEs at different values of the datavector.

Three different optimization algorithms are used to solve the optimization problem described in (6). Two algorithms are of the interior point type, and one is of the active set type. These algorithms are known to have fundamentally different characteristics; for example, interior point methods follow a path through the interior of the feasible region while active set methods tend to stay at the boundaries. We utilize a solver [20] that provides both types of algorithm for greater flexibility in solving problems and allows crossover during the solution process from one algorithm to another. The implementation of the solution also provides a multistart option for promoting the computation of the global minimum.

The measurement (Sampling) in the Cloud DBMS , the dimensions reduction of the dataset and the optimization procedure followed to solve the Inverse Optimal Price Model will be combined in a stand-alone module. The *IOPM module* is designed as a full-fledged stand-alone component that can be integrated into the existing implementation. The module is depicted in Figure 1. The Cloud DBMS and the Optimal Pricing Module comprise the existing implementation.

2.2 Method for Data Regression in the IOPM

The effectiveness of the parameters estimation depends highly on the quality of the sample dataset. The regression of the Price-Demand model (3) that represents our initial motivation was translated into the fitting of (6). The dataset that allows us to solve the optimization problem is developed in phases. A sampling method based on the Metropolis algorithm [21] was selected to gather the dataset from the simulated cloud DBMS.

2.2.1 Selecting a Sampling Method

Several techniques have been suggested for the extraction of a dataset. Researchers have previously used Sequential Importance Sampling(SIS) [9]. The fundamental problem in SIS though is that designing the required good biased distributions becomes more complicated as the system complexity increases, the initial simulation needs to be broken down into several smaller sub-problems, to avoid the memory issues that the dimensionality of our problem would have otherwise caused.

The scaling technique was used in [15] in order to obtain the sample dataset. This method is simple to implement and usually provides conservative simulation gains. Yet, this method pushes mass into the complementary region $p>p_{max}$ which is undesirable.

We selected the Monte Carlo Sampling method [7] for the sample dataset creation. Unlike the conventional uniform sampling methods, this method is able to select object consistent with the underlying distribution. It is simple, efficient and yet powerful and fast-convergent. Experiments were performed in [8] to examine the qualities of the samples by comparing their statistical properties and showed that the samples selected with the Monte Carlo Sampling are *bona fide* representative.

2.2.2 Extending Monte Carlo Sampling in a Cloud DBMS

The sampling process is divided into two phases. In Phase 1 the sampling component interfaces with the simulation to gather the observed variables dataset. We extend the Monte Carlo Sampling method into a new method suitable for the representative sampling of the Cloud DBMS. In Phase 2, the Metropolis algorithm is adopted to create the n-size variables sample **x**. In figure 2 we present the drawing procedure of a representative sample from the Cloud DBMS structures population using the Metropolis algorithm [21].

Phase 1.

At this point we describe a method to generate points in the data space *D* with a relative probability inversely proportional to the scaling of prices,.

This approach requires to prepare first the system in a configuration $r^n \subseteq D$ which we denote by o(old). This corresponds to a dataset including the set of prices for all structures scaled by q. Next we generate a new trial configuration r^{in} by adding a small random displacement Δ to o(old) and we denote this by o(new). Since this new displacement either leads Cloud to overprice, or under-price structures[2], we set the a weighting factor inversely proportional to the displacement, for each object as follows:

$$\forall obj_i \in r^{iM}$$

We define: (12)

$$W(obj_i) := \frac{1}{\Delta_i(r^{iM})}$$

Offline, we generate N^3 o(new) configurations to feed Phase 2 algorithm, each one containing n datavectors. The algorithm that follows is repeated n times in order to sample all structures. The Phase 1 of the sampling method is depicted in Figure 2.

Phase 2. Datavectors d are selected randomly from the dataset gathered in Phase 1, one after another.

As defined in (7), each object is a datavector of the form:

$d_i = (\lambda_{vi}\,[p_{v1},\ldots p_{vn}])$.

The Metropolis sampling method guarantees that the chance of a datavector d being selected into the sample is strictly dependent on the probability of the datavector d. The algorithm is repeated for each structure that we wish to sample.

```
(1)  locate an object with the max weight W(x)
     as the first object x1 of the sample
(2)  for i from 1 to N-1 do
(3)     randomly select a datavector d;
(4)     compute θ=W(y)/W(xi);
(5)     if θ>=1 then xi+1=d;
(6)     else generate a random number R;
(7)        if R<=θ then xi+1=d;
(8)        else xi+1=xi;
(9)        end if;
(10)    end if;
(11) end for.
```

Figure 2:The Monte Carlo/Metropolis Sampling

2.2.3 Dimensions Reduction

In the price-demand model, we deal with the curse of dimensionality. In the essence, for a given sample size, there is a maximum number of structure prices that each demand is correlated above which the demand estimation of our model will degrade rather than improve. For a density of n structures and n dimensions, the size of sample **x** is n^n.

One way to avoid the curse of dimensionality is to project the data onto a lower-dimensional space. Two approaches have been established [10] to perform dimensionality reduction:

- Feature extraction: creating a subset of new features by combinations of the existing features
- Feature selection: choosing a subset of all the features (the ones more informative).The problem of feature extraction can be stated as:

---

[2] Cloud is expected to pursue a 100% profit over cost. This translates to a 2.0 scaling. A scaling factor below 2.0 is consider under-pricing of the cloud service, while a scaling factor over 2.0 is respectively over-pricing.

[3] the N times that sampling is repeated should not be confused with the constant n that represents the number of structures in the cloud cache.

Given a feature space $x_i \in R^n$ find a mapping $s=f(x):R^n \rightarrow R^k$ with $k<n$ such that the transformed structures vector $y_i \in R^K$ preserves (most of) the information in R^n.

An optimal mapping s=f(x) will be one that results in no increase in the minimum probability of error. In general, the optimal mapping s=f(x) will be a non-linear function, However, there is no systematic way to generate non-linear transforms. For this reason, feature extraction is commonly limited to linear transforms: s=Wx. Within the realm of linear feature extraction, we will use the Principal Component Analysis technique which is the oldest and most commonly used dimension reduction technique (also called the Karhunen-Loeve transform) to reduce the n dimensions of the sample.

Principal Component Analysis

PCA is theoretically the optimal linear scheme, in terms of least mean square error, for compressing a set of high dimensional vectors into a set of lower dimensional vectors.

If a multivariate dataset is visualized as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA supplies the user with a lower-dimensional picture, a "shadow" of this object when viewed from its most informative viewpoint.

In PCA, the optimal[4] approximation of a random vector $x \in \Re^k$ by a linear combination of k (k<n) independent vectors is obtained by projecting the random vector x onto the eigenvectors ϕ_i corresponding to the largest eigenvalues λ_i of the covariance matrix Σ_x. The obtained dataset ***s*** contains objects of the form:

$s_i = (\lambda_{vi}, [p_{v1}, \ldots p_{vn}])$.

This final dataset ***s*** would be utilized as the dataset of measured variables in the optimization component in order to estimate the model parameters.

3. INTEGRATION OF THE IOPM MODULE

The IOPM module is integrated into the existing implementation, so that it is possible to re-inject of real demand values in the optimization process and in the future re-identify the price-demand model when needed. This section describes the research methodology followed in order to achieve seamless integration.

3.1 Integration Alternative Approaches

The current working environment is a mixture of Java, Matlab working under a Microsoft Platform. Matlab includes an interface to Java. It allows Matlab to access Java classes (one may call Java classes from Matlab) but not vice versa. Three alternative solutions exist to the extent of our knowledge, each one of which was attempted in order to interface Matlab with Java:

1. An approach based on Matlab Builder™ JA that enables to create Java™ classes from Matlab programs.
2. An approach based on the Java Native Interface (JNI) and Matlab's C Engine library.
3. An approach based on the Java Runtime class.

Approach 1. Initially Matlab Builder JA was examined because of its great portability and the advantage that it runs on all platforms. Matlab Compiler was used to build deployable components that make the Matlab computations accessible to the Java programs. The builder encrypts Matlab functions and generates a Java wrapper around them so that they behave just like any other Java class. However, once the Matlab based Java classes were integrated into the existing Java programs it was made obvious that certain Matlab objects that generated m-code (such as the tomSym objects of the Tomlab toolbox functions) could not be compiled by the Matlab compiler.

Approach 2. With Java's Native Interface [17], one can write a wrapper class for The Mathwork's C Engine Library. Building a Java application that contains native libraries was proven to be quite cumbersome, although this method allows one to take advantage of the functionalities that this library provides for its use with C programs. Data transfer was only character and stream-based, therefore inefficient for larger amounts of data (high latency, low transfer rates compared to direct memory access). A drawback with the Matlab Engine is that a Matlab session requires much memory [3] and takes more time to start than a compiled standalone Tomlab solution. The major drawback though of this method is that it is foremost platform-dependent. Different compilation, installation and setup routines apply to Unix-based systems. This would jeopardize the module's migration in the future in a Unix-based system, therefore this approach was soon abandoned.

3.2 Reaching Interoperability

We selected Java Runtime class to start a new Matlab process in order to reach interoperability between those two technologies. Communication of the Java programs and the Matlab scripts is achieved through acquiring Matlab's standard input and output streams. Through Java classes we run the set of Matlab commands that initiate the Matlab scripts and we retrieve the results. A drawback, however, is the fact that parsing of the Matlab output stream has to be done manually.

Matlab can be started directly form a Java class, either on the same machine, or, under Unix-based systems even on another machine. This method is Platform-independent, required little Java code to be implemented, needs no additional installation procedure or system setup changes and will most likely work with future versions of Matlab.

In the future, if we need to parse the output stream to further interpret or process the results in Java, we would need to find a way to format the contents of a variable in Matlab in a way that is easier to parse and possibly more efficient in transfer size.

4. EXPERIMENTAL EVALUATION

We present the experimental study for a cloud cache system that uses the optimal pricing solution module.

4.1 Experimental Setup and Methodology

Setup. The cloud cache is set up with one back-end database. The cache is operated under a TPCH-based workload, which consists of 7 TPCH query templates and simulates the query evolution of a million SDSS-like queries against a 2.5TB back-end database hosted on Amazon EC2.

We copy the setup used in [15] to evaluate the maximization of the cloud profit using the Price-Demand model.

[4]optimality is defined as the minimum of the sum-square magnitude of the approximation error

The authors in [15] selected this workload as it is more portable across different DBMS and the queries are tunable by using the query generation mechanism of the TPC-H benchmark. The building costs and the maintenance costs are determined using Amazon's pricing model. The building cost is on average 7 orders of magnitude higher than the maintenance cost.

The optimal pricing solution problem is implemented and run in Matlab 7.10 using the tool Tomlab [18].

Methodology. The initial demand of all structures is set to zero in order to avoid high cloud profit by solely leveraging high demand values. The price variable is represented as a percentage of the cost. The cost equation for the cloud DBMS is defined in 2.1.1 The initial price in each round of the simulation is scaled to 2.0 times the original cost, after varying the scaling factor from $0<q<10$ and observing that the meaningful mass of results resides between scaling restricted to 1-3 times the original costs.

The sampling in the cloud cache endures for 1000 rounds, which according to bibliography [7] is considered sufficient to gather a representative sample. In each round, in the sample gathered for the datavectors $(\lambda, ([p_{v1},\ldots p_{vm}],_{vi})$ as described in 2.2.2 we represent the average values of the variables demand and price throughout the round, for each one of the 159 structures present in the simulation. In each round, prices are displaced by Δ ranging from [0,1.00]. At the end of the simulation, the Phase 1 collection of 1000 round sets, represented in a 1000x159 matrix is fed to the Monte Carlo sampling algorithm. The final sample is constituted of a matrix size 20x20, correlating the demand λi of each one of the 20 unique structures with the principal 20 prices, as produced by the Principal Component Analysis.

4.2 Experimental Results

This section summarizes the experimental results

4.2.1 Preliminary Experiments

Initially we observed the demand for the collection of structures available in the cloud cache with respect to the values of prices when the simulation starts. The simulation runs 10 times and in the beginning of each round prices are scaled by a factor of 1-10 compared to the original cost. Observing the structures' overview of demand over the scaling of prices (included in Appendix Figure 1), we concluded that the region of prices that is of interest extends to prices between 1 to 3 times the original cost, considering that a) the cloud will not under-price the services and b) when prices exceed 5 times the cost, only the indispensable structures are bought. We therefore set the base for sampling to prices at 2–times the original cost. In each simulation round, a random displacement $\Delta\in[0,1.00]$ will be applied to the base of prices, as described in Section 2.2.2.

The overview of price-demand confirmed our claiming that the demand for a price has a dependency not only on its own price, but also on the price of others. As prices grow, we remark that the demand rises for structures that were less popular at lowest prices, meaning that these structures were favored against more expensive ones, under the assumption that their service is complimentary.

Measuring the standard deviation in the sample gathered by Monte Carlo Sampling, we appreciate that prices are clustered tightly around their mean. The STD value does not exceed 7.7e-3, which is 5 orders of magnitude below the price values.

Principal Component Analysis on the 1000x159 dataset that occurred from the Monte Carlo Sampling leaves 20x20 structures. Clustering of structures is depicted in Figure 3. Clusters 2 and 5 represent those that contribute greatest to the demand, in a percentage of 89.9%.

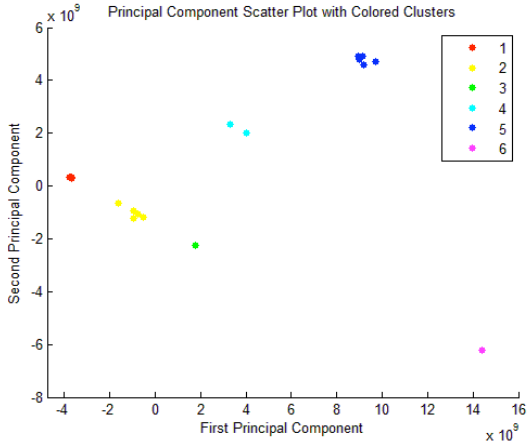


Figure 3: Principal Component Analysis

4.2.2 Parameter Estimation Evaluation

The reduced dataset of size 20x20 is used for the optimization process applied in order to estimate the model of price-demand parameters, as formed in 2.2.1.

The dataset that we used considers the price correlations of 10 structures, restricted to a size 10x10 by memory limitations. The 20x20 dataset required iterations that exceeded 400 which was the upper limit of our system. In the best case the optimization converged to a solution with an error rate of 2.2017e-12, when optimization was employed for 100 iterations, using various solvers in Tomlab. The IOMP problem parameters $\{m_1,m_2,m_3\}=\{A,B,\Gamma\}$ were estimated according to this optimization.

The efficiency of the optimization process depends highly on the choice of the initial values and the bounding of the solution. The final estimation occurred after fine tuning of the initial values and might be subject of further refinement. The first phase of the optimization begins with guesses of all parameter values and uses these to solve for the parameters of the corresponding production term. The second phase takes these estimates to improve the prior parameter guesses. The phases are iterated until a solution is found or the process is terminated for other reasons.

Different solvers were tested in the optimization component. All algorithms attempt to find a local optimum. The algorithms in the solvers CONOPT [22] and SNOPT [23] are based on fairly different mathematical algorithms, and they behave differently on most models. This means that while CONOPT is superior for some models, SNOPT will be superior for others. SNOPT is considered ideal for models few nonlinearities outside the objective function and was therefore a reasonable choice.

Figure 4 depicts the optimization results with solver KNITRO, short for "Nonlinear Interior point Trust Region Optimization" [20], which was selected upon comparison with other solvers. This is a solver designed for large problems with dimensions running into the hundred thousand, highly regarded for its robustness and efficiency, particularly in nonlinear optimization problems.

The results of the comparison are presented in Table 1. In Figure 5 we provide a graphical comparison of the solvers we tested.

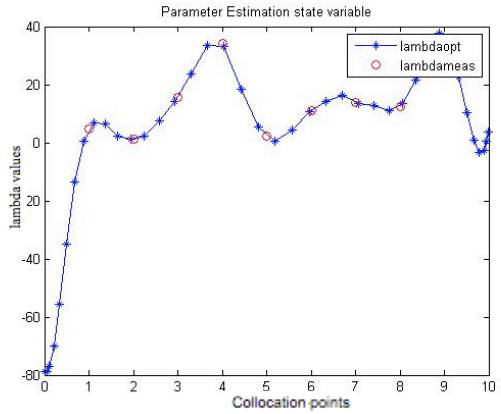


Figure 4: Optimization Process with KNITRO solver

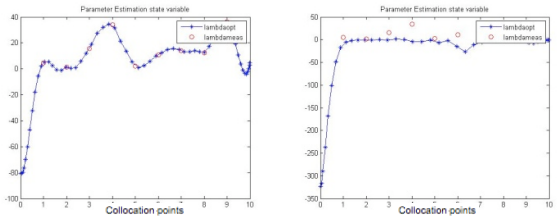


Figure 5: Optimization with solvers (a) CONOPT , (b) SNOPT

Table 1: Solvers Comparison on Parameters Estimation

Solver	KNITRO	SNOPT	CONOPT
Error rate	2.2017e-12	8.5462e-005	4.849243e+003

5. DISCUSSION

The implementation of this project depends greatly on Tomlab Matlab Toolbox, which was used under evaluation license. The license had to be renewed every 21 days, therefore the programming workflow was interrupted quite many times. The implementation of the Sampling component of IOPM represented a bottleneck for the project, as the Optimization component of IOPM although implemented early enough, could only be tested when the dataset was prepared. Finally, the system restrictions posed the biggest obstacles to this effort, underlining the necessity to migrate the module to the Cloud Cluster in order to evaluate its performance on large-scale data.

6. CONCLUSIONS

The IOPM module includes a stand-alone module that expands the underlying pricing-scheme and is integrated with the optimal pricing solution with the price-demand model. We presented the IOPM problem formulation and solution through non-linear optimization procedure. We described the method adopted to draw representative samples from the cloud DMBS in order to construct a dataset suitable for fitting and perform Data Regression. We further demonstrated how the module was integrated into the existing implementation. Our short-term goal is to elaborate on the parameter estimation under a high-performance platform that will enable a large-scale optimization process.

7. REFERENCES

[1] A. Sulistion, K. Kyong Hoon and R. Buya. Using revenue management to determine pricing of reservation. In IEEE e-Scence, pages 396-405,2007.

[2] Amir Globerson, Naftali Tishby: Sufficient Dimensionality Reduction. Journal of Machine Learning Research (JMLR) 3:1307-1331 (2003).

[3] Andreas Klimke and Barbara Wohlmuth and Universität Stuttgart Extending Matlab with Java – Hanselman, Littlefield (2001).

[4] Carmen G. Moles, Pedro Mendes and Julio R. Banga Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods, Genome Research CSH Press (2003)

[5] Christos Michalakelis, et al Development of a demand model for the simultaneous estimation of the interaction between a product's diffusion and price: an application to telecommunications, CTTE (2007)

[6] F Wood, K Esbensen, P Geladi,Principal component analysis - Chemometr. Intel. Lab. Syst (1987).

[7] Frenkel Dea, Introduction to Monte Carlo Methods, Computational Soft Matter, John Von Neumann Institute for Computing, Julich, NIC Series, vol 23 (2004).

[8] Hong Guo, Wen-Chi Hou, Feng Yan, Qiang Zhu: A Metropolis Sampling Method for Drawing Representative Samples from Large Databases. SSDBM (2004)

[9] Isabel Beichl, Francis Sullivan: The Other Monte Carlo Method. Computing in Science and Engineering 8(2): 42-47 (2006).

[10] Jolliffe I.T. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4

[11] Kenneth Holmström: Solving Applied Optimization Problems Using Tomlab , Proceedings from MATHTOOLS '99, the 2nd International Conference on Tools for Mathematical Modeling, St. Petersburg, Russia (1999).

[12] Powell, Warren B, Approximate Dynamic Programming: Solving the Curses of Dimensionality. Wiley, ISBN 0470171553 (2007).

[13] Regula, G.1, Ledergerber, U et al, Using Monte Carlo simulation to optimize sampling strategies for monitoring antimicrobial resistance (2006)

[14] Tanu Malik, Randal C. Burns and Amitabh Chaudhary. A financial option based grid resources pricing model: Towards an equilibrium between service quality for user and profitability for service providers. In Advances of Grid and Pervasise Computing, pages 13-24, 2009

[15] Verena Kantere, Debabrata Dash, Anastasia Ailamaki, Optimal service pricing for a cloud cache

[16] Wolfgang Förstner: Reliability analysis of parameter estimation in linear models with applications to mensuration

problems in computer vision. Computer Vision, Graphics, and Image Processing (CVGIP) 40(3):273-310 (1987)

[17] Sun Microsystems, Inc. Java Native Interface, 2002. http://java.sun.com/j2se/1.4.1/docs/guide/jni/index.html [cited April 10, 2003].

[18] http://tomopt.com/

[19] Albert Tarantola, Inverse Problem Theory and Methods for Model Parameter Estimation, SIAM (2005)

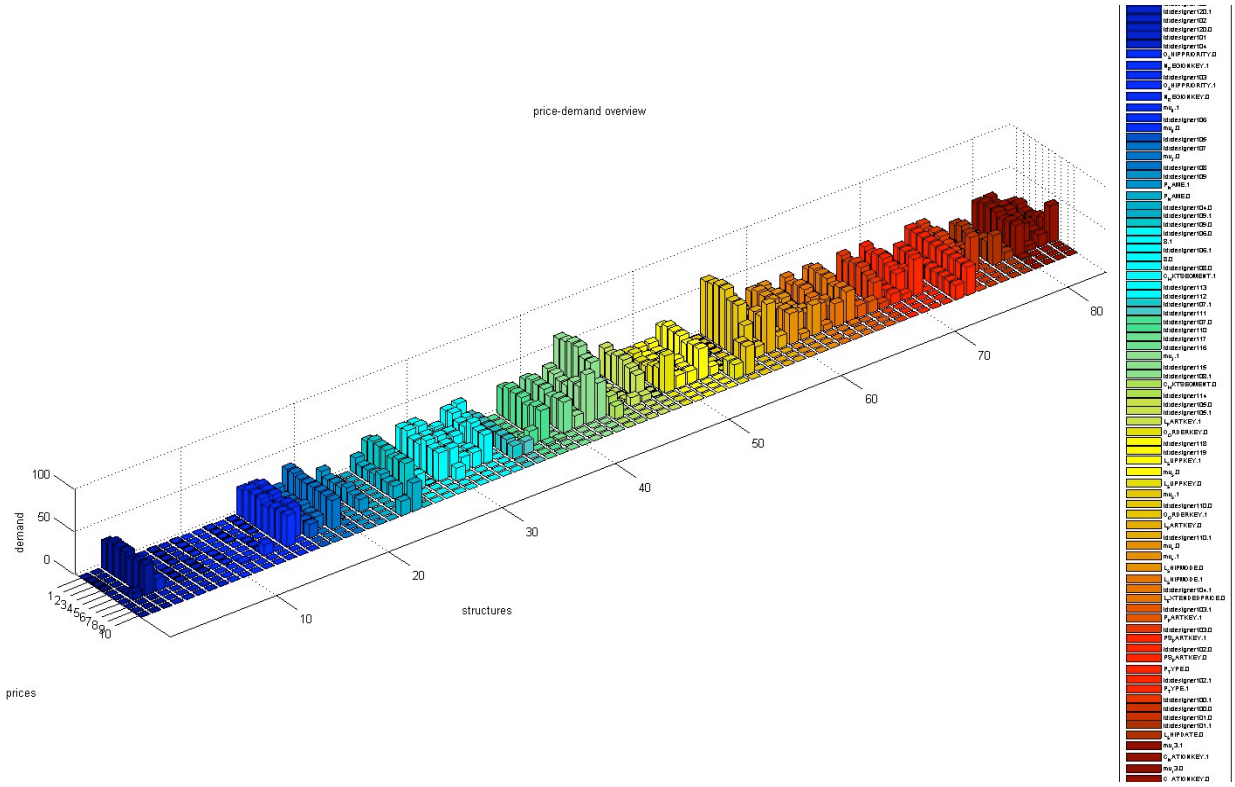
[20] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. Mathematical Programming, Series B, 100(1):27–48, (2004).

[21] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. "Equation of state calculation by fast computing machines." Journal of Chemical Physics, 21(6):1087–1092, 1953.

[22] http://www.aimms.com/features/solvers/conopt

[23] http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm

8. APPENDIX



Appendix Figure 1: Price-Demand overview for all structures