Optimal service pricing for a cloud cache

Verena Kantere #1, Debabrata Dash *, Gregory Francois #2, Sofia Kyriakopoulou #3, Anastasia Ailamaki #4

#*Ecole Polytechnique Fédérale de Lausanne*
1015 Lausanne, VD, Switzerland
1verena.kantere, 2gregory.francois, 3sofia.kyriakopoulou, 4anastasia.ailamaki@epfl.ch
* *Carnegie Mellon University*
5000 Forbes Avenue, PA 15213
ddash@cmu.edu

***Abstract*—Cloud applications that offer data management services are emerging. Such clouds support caching of data in order to provide quality query services. The users can query the cloud data, paying the price for the infrastructure they use. Cloud management necessitates an economy that manages the service of multiple users in an efficient, but also, resource-economic way that allows for cloud profit. Naturally, the maximization of cloud profit given some guarantees for user satisfaction presumes an appropriate price-demand model that enables optimal pricing of query services. The model should be plausible in that it reflects the correlation of cache structures involved in the queries. Optimal pricing is achieved based on a dynamic pricing scheme that adapts to time changes. This paper proposes a novel price-demand model designed for a cloud cache and a dynamic pricing scheme for queries executed in the cloud cache. The pricing solution employs a novel method that estimates the correlations of the cache services in an time-efficient manner. The experimental study shows the efficiency of the solution.**

***Index Terms*—cloud data management, data services, cloud service pricing**

I. Introduction

The leading trend for service infrastructures in the IT domain is called *cloud computing*, a style of computing that allows users to access information services. Cloud providers trade their services on cloud resources for money. The quality of services that the users receive depends on the utilization of the resources. The operation cost of used resources is amortized through user payments. Cloud resources can be anything, from infrastructure (CPU, memory, bandwidth, network), to platforms and applications deployed on the infrastructure.

Cloud management necessitates an economy, and, therefore, incorporation of economic concepts in the provision of cloud services. The goal of cloud economy is to optimize: (i) user satisfaction and (ii) cloud profit. While the success of the cloud service depends on the optimization of both objectives, businesses typically prioritize profit. To maximize cloud profit we need a pricing scheme that guarantees user satisfaction while adapting to demand changes.

Recently, cloud computing has found its way into the provision of web services [15], [18]. Information, as well as software is permanently stored in Internet servers and probably cached temporarily on the user side. Current businesses on cloud computing such as Amazon Web Services [14] and Microsoft Azure [19] have begun to offer data management services: the cloud enables the users to manage the data of back-end databases in a transparent manner. Applications that collect and query massive data, like those supported by CERN [17], need a caching service, which can be provided by the cloud [31].

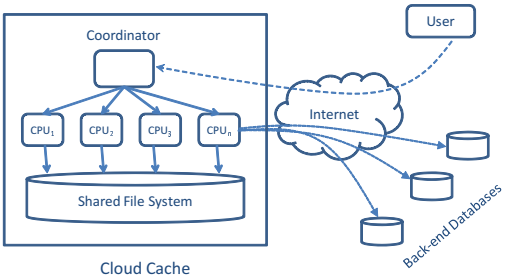

Fig. 1. A cloud cache

The goal of such a cloud is to provide efficient querying on the back-end data at a low cost, while being economically viable, and furthermore, profitable. Figure 1 depicts the architecture of a cloud cache. Users pose queries to the cloud through a coordinator module, and are charged on-the-go in order to be served. The cloud caches data and builds data structures in order to accelerate query execution. Service of queries is performed by executing them either in the cloud cache (if necessary data are already cached) or in a back-end database. Each cache structure (data or data structures) has an operating (i.e. a building and a maintenance) cost. A price over the operating cost for each structure can ensure profit for the cloud. In this work we propose a novel scheme that achieves optimal pricing for the services of a cloud cache.

A. Setting the price for cloud caching services

The cloud makes profit from selling its services at a price that is higher than the actual cost. Setting the right price for a service is a non-trivial problem, because when there is competition the demand for services grows inversely but not proportionally to the price.

There are two major challenges when trying to define an optimal pricing scheme for the cloud caching service. The first is to define a simplified enough model of the price-demand dependency, to achieve a feasible pricing solution, but not oversimplified model that is not representative. For example, a static pricing scheme cannot be optimal if the demand for services has deterministic seasonal fluctuations. The second challenge is to define a pricing scheme that is adaptable to (i) modeling errors, (ii) time-dependent model

changes, and (iii) stochastic behavior of the application. The demand for services, for instance, may depend in a non-predictable way on factors that are external to the cloud application, such as socioeconomic situations.

A representative model for the cloud cache should take into account that the cache structures (table columns or indexes) may compete or collaborate during query execution. The demand for a structure depends not only on its price, but also on the price of other structures. For example, consider the query `select` A `from` T `where` $B=5$ `and` $C=10$. Out of the set of candidate indexes to run the query efficiently, indexes $I_b = T(B)$, $I_c = T(C)$, and $I_{bc} = T(BC)$ are most important, since they can satisfy the conditions in the 'where' clause. If the cache uses I_{bc}, then the indexes I_b and I_c, will never be used, since I_{bc} can satisfy both conditions. Therefore, the presence of I_{bc} has a negative impact on the demand for I_b and I_c. Alternatively, if the cache uses I_b, then I_c can also improve query performance via index intersections, hence increasing the profit for the cloud. Therefore, indexes I_b and I_c have positive impact on each other's demand. An appropriate estimation method is necessary to model price-demand correlations among cached structures.

The peculiarity of the pricing problem for the application of the cloud DBMS, in comparison with other businesses, is that the selling good is not a consumable product, but a persistent service.

A consumable product diminishes with demand and has to be ordered, whereas a cloud cache service can satisfy infinite demand as long as it is maintained. Moreover, the demand for a cache service pauses if this service is not available. A consumable product may cost to maintain depending on the stored amount, whereas the maintenance cost of a cache service depends only on time. Moreover, a cache service may have a setup cost each time it is loaded in the cloud. A big challenge for the cloud is to optimize the set of offered services, i.e. decide which services to offer and when, depending on their demand while they are available. Roughly, the cloud has to schedule online and offline periods of the offered services, which affects the maintenance and the setup cost. Furthermore, the optimization of the cloud profit has to be scheduled for a long period in time while it is flexible during this period to adjust to the real evolution of the service consumption. The long-term profit optimization is necessary in order for the cloud to schedule ahead associative actions for the maintenance of the cloud infrastructure and the cloud data. Moreover, the cloud can schedule the service availability according to the guarantees for the overall revenue estimated by the long-term optimization. Nevertheless, it is important that the long-term optimization process is flexible enough to receive corrections while it is still in progress. The corrections may refer to the difference between the estimated and the actual price influence on the demand of services.

B. Related work

Existing clouds focus on the provision of web services targeted to developers, such as Amazon Elastic Compute Cloud (EC2) [14], or the deployment of servers, such as GoGrid [18]. Emerging clouds such as the Amazon SimpleDB and Simple Storage Service offer data management services. Optimal pricing of cached structures is central to maximizing profit for a cloud that offers data services.

Cloud businesses may offer their services for free, such as Google Apps [15] and Microsoft Azure [19] or based on a pricing scheme. Amazon Web Service (AWS) clouds include separate prices for infrastructure elements, i.e. disk space, CPU, I/O and bandwidth. Pricing schemes are static, and give the option for pay as-you-go. Static pricing cannot guarantee cloud profit maximization. In fact, as we show in our experimental study (see Section VI), static pricing results in an unpredictable and, therefore, uncontrollable behavior of profit.

Closely related to cloud computing is research on accounting in wide-area networks that offer distributed services. Mariposa [35] discusses an economy for querying in distributed databases. This economy is limited to offering budget options to the users, and does not propose any pricing scheme. Other solutions for similar frameworks [38], [8], [29], [21], [4], [22], [26] focus on job scheduling and bid negotiation, issues orthogonal to optimal pricing.

Pricing schemes were proposed recently for the optimal allocation of grid resources in order to increase revenue [36], or to achieve an equilibrium of grid and user satisfaction [25], assuming knowledge of the demand for resources or the possibility to vary the price of a resource for different users. These works are orthogonal to ours, as we do not assume that service demand is known a priori and all users are charged the same for the consumption of the same service. Similarly, dynamic pricing for web services [23] focuses on scheduling user requests. This work is orthogonal to ours, as we require that users' requests for service are satisfied right away. Moreover, dynamic pricing for the provision of network services [27], [13], [3] aims at achieving a game-theoretic equilibrium through price control among competitive Internet Service Providers. This work is orthogonal to ours, as we focus on the maximization of cloud profit in the presence of competitive services within the same cloud provider.

The problem of revenue management through dynamic pricing is well-studied [1]. Based on the rationale that price and demand are dependent qualities, numerous variations of the problem have been tackled, for instance businesses that sell products to retailers [10], seasonal products [40], stochastic demand [9]. Electronic businesses, and therefore cloud businesses can benefit from dynamic pricing policies [30]. Cache services are distinguished from consumable products in two major ways: (i) they are not exhausted while they are consumed and (ii) the demand for a specific service pauses while this is not available. To the best of our knowledge, this is the first work that tackles the problem of optimal pricing of competitive data services within the same cloud cache provider.

Research on the identification of non-correlated indexes using the query structure [39] does not determine the negative and positive correlations. Identification of index

correlations by modeling physical design as a sub-modular and super-modular problem [5] is restricted to one-column indexes and one index per query. Identification of negative index correlation [2] does not consider the positive and no correlation case. A recent index interaction model [33] attempts to find all index correlations. As we show in Section IV, it does not satisfy three critical for the pricing scheme requirements: (i) sensitivity to the range of all possible correlations, (ii) production of normalized values and (iii) fast computation.

C. Our proposal

The cloud caching service can maximize its profit using an optimal pricing scheme. This work proposes a pricing scheme along the insight that it is sufficient to use a simplified price-demand model which can be re-evaluated in order to adapt to model mismatches, external disturbances and errors, employing feedback from the real system behavior and performing refinement of the optimization procedure. Overall, optimal pricing necessitates an appropriately simplified price-demand model that incorporates the correlations of structures in the cache services. The pricing scheme should be adaptable to time changes.

Simple but not simplistic price-demand modeling. We model the price-demand dependency employing second order differential equations with constant parameters. This modeling is flexible enough to represent a wide variety of demands as a function of price. The simplification of using constant parameters allows their easy estimation based on given price-demand data sets. The model takes into account that structures can be available in the cache or can be discarded if there is not enough respective demand. Optional structure availability allows for optimal scheduling of structure availability, such that the cloud profit is maximized. The model of price-demand dependency for a set of structures incorporates their correlation in query execution.

Price adaptivity to time changes. Profit maximization is pursued in a finite long-term horizon. The horizon includes sequential non-overlapping intervals that allow for scheduling structure availability. At the beginning of each interval, the cloud redefines availability by taking offline some of the currently available structures and taking online some of the unavailable ones. Pricing optimization proceeds in iterations on a sliding time-window that allows online corrections on the predicted demand, via re-injection of the real demand values at each sliding instant. Also, the iterative optimization allows for re-definition of the parameters in the price-demand model, if the demand deviates substantially from the predicted.

Modeling structure correlations. Our approach models the correlation of cache structures as a dependency of the demand for each structure on the price of every available one. Pairs of structures are characterized as competitive, if they tend to exclude each other, or collaborating, if they coexist in query plans. Competitive pairs induce negative, whereas collaborating pairs induce positive correlation. Otherwise correlation is set to zero. The index-index, index-column, and column-column correlations are estimated

```
Global: cache structures S, prices P, availability Δ
queryExecution( )
  if q can be satisfied in the cache then
    (result, cost) ← runQueryInCache(q)
  else
    (result, cost) ← runQueryInBackend(q)
  end if
  S ← addNewStructures()
  return result, cost
optimalPricing (horizon T, intervals t[i], S)
  (Δ, P) ← determineAvailability&Prices(T, t, S)
  return Δ, P
main()
execute in parallel tasks T1 and T2:
T1:
  for every new i do
    slide the optimization window
    optimalPricing(T, t[i], S)
  end for
T2:
  while new query q do
    (result, cost) ← queryExecution(q)
  end while
  if q executed in cache then
    charge cost to user
  else
    calculate total price and charge price to user
  end if
```

Fig. 2. Query execution model for the cloud cache

based on proposed measures that can estimate all three types of correlation. We propose a method for the efficient computation of structure correlation by extending a cache-based query cost estimation module and a template-based workload compression technique.

D. Contributions

This paper makes the following contributions:

- A novel demand-pricing model designed for cloud caching services and the problem formulation for the dynamic pricing scheme that maximizes profit and incorporates the objective for user satisfaction.
- An efficient solution to the pricing problem, based on non-linear programming, adaptable to time changes.
- A correlation measure for cache structures that is suitable for the cloud cache pricing scheme and a method for its efficient computation.
- An experimental study which shows that the dynamic pricing scheme out-performs any static one by achieving 2 orders of magnitude more profit per time unit.

The rest of the paper is structured as follows. Section III models the optimal pricing problem and Section IV models the price-demand correlations for data structures in the cloud cache. Section V describes the solution of the pricing optimization problem and Section VI presents the experimental study. Section VII concludes the paper.

II. Query Execution Model

The cloud cache is a full-fledged DBMS along with a cache of data that reside permanently in back-end databases. The goal of the cloud cache is to offer cheap

efficient multi-user querying on the back-end data, while keeping the cloud provider profitable. Our motivation for the necessity of such a cloud data service provider derives from the data management needs of huge analytical data, such as scientific data [31], for example physics data from CERN [17] and astronomy data from SDSS [20]. Furthermore, a viable, and moreover, profitable data service provider can achieve cost and time efficient management of smaller scientific collections or any type of analytical data, such as digital libraries, multimedia data and a variety of archived data.

Users pose queries to the cloud, which are charged in order to be served. Following the business example of Amazon and Google, we assume that data reside in the same data center and that users pay on-the-go based on the infrastructure they use, therefore, they pay by the query. Service of queries is performed by executing them either in the cloud cache or in the back-end database. Query performance is measured in terms of execution time. The faster the execution, the more data structures it employs, and therefore, the more expensive the service. We assume that the cloud infrastructure provides sufficient amount of storage space for a large number of cache structures. Each cache structure has a building and a maintenance cost.

Figure 2 presents at a high level the query execution model of the cloud cache. The names of variables and functions are self-explanatory. The user query is executed in the cache iff all the columns it refers to are already cached. Otherwise it is executed in the backend databases. The *result* is returned to the user and the *cost* is the query execution cost (the cost of operating the cloud cache or the cost of transferring the result via the network to the user). The cloud cache determines which structures (cached columns, views, indexes) S to build in order to accelerate query execution and reduce the query execution cost. Initially S is empty and gradually it is filled with structures that would have or have benefitted past queries. How S is populated and how the costs of building and maintaining cache structures as well as the query execution cost are computed is an input to the presented optimal pricing scheme. More details on these issues can be found in [7]. Periodically (on predefined time intervals $t[i]$) the cloud performs the pricing scheme proposed in this work. The pricing scheme schedules the availability Δ and sets the prices P of the structures S for a time horizon T as described in the rest of the paper. The goal is to maximize the provider's profit and ensure that the user is not overcharged.

III. Modeling Optimal Pricing

This section describes the problem formulation of maximizing the cloud profit. The presentation of the pricing scheme is guided by *propositions* that state the main rationale of our approach.

A. Problem Formulation

This section defines the objective and the constraints of the problem, and gives the mathematical problem definition.

1) Objective: The cloud cache offers to the users query services on the cloud data. The user queries are answered by query plans that use cache structures, i.e. cached columns and indexes. We assume that the set of possible cache structures is $\mathcal{S} = \{S_1, \ldots, S_m\}$.

Whenever a structure S is built in the cache, it has a one-time building cost B_S. While S is maintained in the cache it has a maintenance cost which depends on time, $M_S(t)$. We assume that each structure is built from scratch in the cloud cache, as the cloud may not have administration rights on existing back-end structures. Nevertheless, cheap computing and parallelism on cloud infrastructure may benefit the performance of structure creation. For a column, the building cost is the cost of transferring it from the back-end and combining it with the currently cached columns. This cost may contain the cost of integrating the column in the existing cache table. For indexes, the building cost involves fetching the data across the Internet and then building the index in the cache. Since sorting is the most important step in building an index, the cost of building an index is approximated to the cost of sorting the indexed columns. In case of multiple cloud databases, the cost of data movement is incorporated in the building cost. The maintenance cost of a column or an index is just the cost of using disk space in the cloud. Hence, building a column or an index in the cache has a one-time static cost, whereas their maintenance yields a storage cost that is linear with time[1]. For more information on the building and maintenance cost of cloud cache structures the reader is referred to [7]. In any case, the cost of a structure S as soon as it is built at time t_{built} in the cache and until it is discarded is:

$$c_S(t) = B_S + M_S(t - t_{built}), \tag{1}$$

Cache services are offered through query execution that uses cache structures. The demand for cache structures is defined as follows:

Definition 1: The demand for a cache structure S, denoted as $\lambda_S(t)$, is the number of times that S is employed in query plans selected for execution at time t.

Naturally, in realistic situations the demand for a structure is measured in time intervals. If a structure S is built in the cache then query plans that involve it can be selected, i.e. $\lambda_S(t) \geq 0$, otherwise not, i.e. $\lambda_S(t) = 0$. Intuitively, there is a trade-off between (i) keeping a structure in the cache and paying the maintenance cost, and (ii) building and discarding the structure occasionally. This trade-off is reinforced towards the one or the other direction by the demand of the structure: if the demand is low, it is possible that it is cheaper to discard the structure from the cache and pay the building cost multiple times, than pay the maintenance cost; if the demand is high, then the opposite tactic may be more profitable for the cloud.

[1]Index updating is assumed to incur rebuilding the index from scratch. Data updates are external factors that cannot be controlled by the optimization procedure. In Section VI we study the effect of updates to the dynamic pricing solution.

The cloud makes profit by charging the usage of structures in selected query plans for a price. Let us assume that the price of a structure S at time t is $p_S(t)$. Then the profit of the cloud at a specific time is:

$$r(t) = \sum_{i=1}^{m} \delta_i \cdot (\lambda_{S_i}(t) \cdot p_{S_i}(t) - c_{S_i}(t)), \delta_i = 0,1 \quad (2)$$

where δ_i represents the fact that the structure S_i is present in the cloud cache. Specifically, a structure may be present or not in the cache at any time point in $[0,T]$ and not present before the beginning of optimization time, i.e.:

$$\delta_i(t) = \begin{cases} 0 \text{ or } 1 & \text{if } t \in [0,T] \\ 0 & \text{otherwise} \end{cases}$$

Based on this, the cost of a structure w.r.t. time becomes:

$$c_S(t) = (1 - \delta_i(t_{0^-}))B_S + M_S(t - t_0), \quad (3)$$

where t_0 is the start time of cost observation.

Structures can be built and discarded at any time $t \in [0,T]$ and the total profit of the cloud is $R(T) = \int_0^T r(t)dt$. The goal is to maximize the total profit in $[0,T]$ by choosing which structures to build or discard and which price to assign to each built structure at any time:

$$\max_{\delta,p} R(t) = \int_0^T r(t)dt \quad (4)$$

2) Problem constraints: It is necessary to constrain the optimization of the objective 4, so that a reasonable and correct solution can be found.

Value constraints. It is straightforward that both the demand and the price of a structure must be positive numbers. Furthermore, it is necessary to impose an upper bound on the price. The reason is that the optimum solution is to instantaneously raise the price of at least one structure to infinity, if this is allowed[2]. These bounds can be formulated as follows:

$$0 \leq \lambda_i, i = 1, \ldots, m \quad (5)$$

$$0 \leq p_i \leq p_{max}, i = 1, \ldots, m \quad (6)$$

Dynamics of the demand. Naturally, the demand and the price of a structure are connected variables: intuitively, as the price for a structure increases the demand decreases and vice versa. In order to to solve the optimization problem 4, a mathematical relationship, which models the interaction between demand and price, is necessary. However, this mathematical relationship should have a structure as flexible as possible, so that, upon a proper identification of its parameters, it is able to represent as many as possible functions of demand and price. We make the following assumption:

Proposition 1: The demand of a structure S has memory: the demand at time t depends on the demand before t. Consequently, the relationship between price and demand can be modeled as an ordinary differential equation, which can be written in the general case in its implicit form:

$$f(\frac{d^n\lambda_S}{dt}, \frac{d^{n-1}\lambda_S}{dt}, \ldots, \frac{d\lambda_S}{dt}, \lambda_S(t), \frac{d^m p_S}{dt}, \ldots, \frac{dp_S}{dt}, p_S(t)) = 0 \quad (7)$$

where $m \leq n$, to respect the causality principle, as $m > n$ would imply that demand could change (due to a change of price) before the price has changed.

In particular, since there is no inertia in setting a price for a structure, $m = 0$ and Equation 7 can be rewritten in its explicit form:

$$p_S(t)) = f(\frac{d^n\lambda_S}{dt}, \frac{d^{n-1}\lambda_S}{dt}, \ldots, \frac{d\lambda_S}{dt}, \lambda_S(t)) \quad (8)$$

Justification 1: Economies as well as societies tend to behave in a way that reflects past experience. More formally, an economic system, such as the cloud cache and its users has *inertia*, which means that the current system behavior depends on past and influences future behavior. Concerning the cloud cache, this means that the demand for structures has a time-resistant effect. For example, assume that the demand for a structure built in the cloud cache does not drop as fast as expected in a memory-less system w.r.t. price increase. Two intuitive exemplifying reasons for this are: (i) the structure is already built and remains available because the building cost is already amortized, while the maintenance cost is not very high; and (ii) the structure, for example a cached column is requested for the execution of numerous queries, because it involves information that is currently very popular to users.

Associating the price with the nth derivative of demand for a structure, guarantees n degrees of freedom for the shape of their relationship. Therefore, the bigger the order n is, the more flexible the price-demand relationship is. Yet, as the order n increases, the number of parameters of the price-demand relationship increase and more information is needed in order to (see Section V) identify their values. We choose to consider a 2nd order differential equation as it is versatile enough to represent a price-demand relationship, where the demand drops smoothly at the beginning of time [3], as depicted in Figure 3, while keeping the number of parameters to identify low. Therefore, the constraint is: $p_S(t) = f(\frac{d^3\lambda_S}{dt})$. We constrain f to be an ordinary differential relation between price and demand:

$$p_S(t) = \alpha\frac{d^2\lambda_S}{dt} + \beta\frac{d\lambda_S}{dt} + \gamma \cdot \lambda_S(t) \quad (9)$$

The parameters α, β, γ are constrained to be constants. This means that the price model considers a static relation between demand and price. In order to make the pricing model realistic, we have to consider the influence of the price of one structure to the demand of the rest. Therefore, it is necessary to extend Eq. 9 so that it captures correlations of demand and prices between pairs of structures. Let us

[2]Mathematically, the integral of Eq. 4 goes to infinity if the price for one structure is infinite and the demand for this structure is not zero. If the demand is zero, the profit, $\infty \times 0$ is undefined. Moreover, all numerical solvers need upper bounds in order to produce solutions.

[3]Note that an abrupt drop is expressed by a first order differential equation, which is encapsulated in the second order one, as the parameter a can be set to 0.

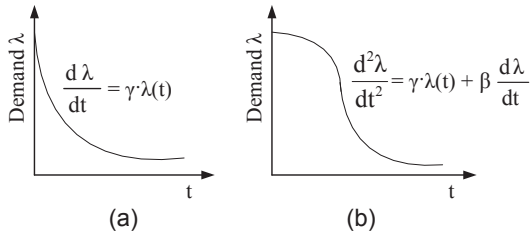

Fig. 3. The shape of the demand vs. time function based on a first (a) and a second (b) order differential equation.

assume that V is a $m \times m$ matrix where the row and the column i corresponds to the structure S_i $i = 1, \ldots, m$. Each element v_{ij}, $i, j = 1, \ldots, m$ corresponds to the correlation of the price of S_j to the demand of S_i. We call V the *correlation* matrix of prices and demands. If Λ and P are the $m \times 1$ matrices of demands and prices for the respective structures in $\mathcal{S}$, and A, B, Γ are $m \times 1$ matrices of parameters, then the constraint in Eq. 9 becomes:

$$V \cdot P = A^T \frac{d^2\Lambda}{dt} + B^T \frac{d\Lambda}{dt} + \Gamma^T \Lambda \tag{10}$$

Eq. 10 is actually a set of constraints of the form: $\sum_{j=1}^{j=m} b_{i,j} \cdot p_{S_j}(t) = \alpha_i \frac{d^2\lambda_{S_i}}{dt} + \beta_i \frac{d\lambda_{S_i}}{dt} + \gamma_i \cdot \lambda_{S_i}(t)$.

Problem definition. The previous discussion leads to the following problem formulation for optimal pricing:

The maximization of the cloud DBMS profit is achieved with the solution of the following optimization problem:

$$\max_{\delta, p} R(t) = \int_0^T \sum_{i=1}^m [\delta_i(t_j) \cdot (\lambda_{S_i}(t) \cdot p_{S_i}(t) - c_{S_i}(t))] dt$$

subject to the constraints:

$$0 \leq \lambda_i, i = 1, \ldots, m,$$
$$0 \leq p_i \leq p_{max}, i = 1, \ldots, m,$$
$$V \cdot P = A^T \frac{d^2\Lambda}{dt^2} + B^T \frac{d\Lambda}{dt} + \Gamma^T \Lambda$$

B. Generalization of Optimization Objective

The problem of optimal pricing as formulated in Section III-A consists of a sole objective: the maximization of the cloud profit, subject to some constraints. From a mathematical point of view, we expect a solution that is on the boundaries of the *feasible area*, meaning a solution along the constraints of the problem that satisfies the objective. The constraints on the price-demand dependency in Eq. 10 do not actually constrain the sought solution, but only the value of the optimal profit, if the solution is applied; therefore, the sought solution is expected to be on the boundaries of the allowed price, Eq. 5, and demand values, Eq. 6, meaning maximum price selections as long as the demand for structures is above zero, as shown in Figure 4(a). This is called a *bang-bang* solution and the mathematical reason for this expectation is that the objective of the problem is linear w.r.t. the control variables: the price p and the structure availability δ. Intuitively, the objective of optimization is the purely egoistic and straightforward maximization of cloud profit. The optimization procedure

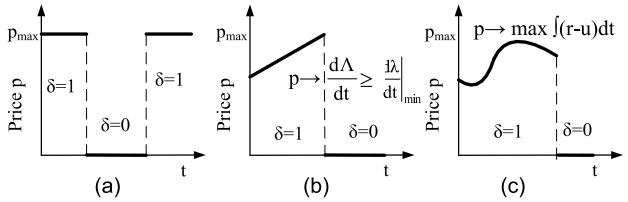

Fig. 4. While structures are available, the optimization of the objective function may lead to choices of price values that are: (a) on the boundaries, (b) change linearly or (c) follow a trajectory.

shall try to achieve this goal as soon as possible, resulting in charging the highest possible prices as long as there is structure demand. Of course, the freedom of choosing the availability of structures complicates the optimization goal, but does not change the decision for maximum charge whenever availability for a structure is decided.

Naturally, one would expect that the user dissatisfaction from high service charge, which is the actual reason for the demand drop, should be taken into consideration in a real cloud business. Simply, the cloud risks to permanently lose the dissatisfied users in an open-market world. The user satisfaction is an altruistic tend of the optimization that is opposite to the egoistic tend of cloud profit.

Proposition 2: The altruistic tend of pricing optimization is expressed as: (i) a guarantee for a low limit on user satisfaction, or (ii) an additional maximization objective.

Justification 2: There are two policies in order to incorporate an altruistic tend in pricing optimization. The first is to give a much lower priority to user satisfaction than cloud profit, which results into a constraint (static or time-dependent) that passively restricts the maximization of profit, i.e. expression (i). The second is to handle it as a secondary goal of the pricing optimization, which results into a new objective that actively restricts profit maximization. 'Passive' restriction means that the altruistic tend turns down pricing solutions proposed by the optimization procedure, while 'active' restriction means that the altruistic tend is involved in the proposition of pricing solutions.

If the altruistic tend is expressed as low-limit guarantee on user satisfaction, then it can be formulated as an additional constraint of the optimization problem of Section III-A on the demand drop:

$$\frac{d\Lambda}{dt} \geq \frac{d\lambda}{dt}\bigg|_{min} \tag{11}$$

where $\dot{\lambda}_{min}$ is the selected minimum value of demand drop rate. Alternatively, the user satisfaction can be defined as the difference of the structure price and the actual cost:

$$u(t) \equiv p_S(t) - c_S(t) \tag{12}$$

In this case, the problem can accommodate, either a new constraint or a new optimization objective. In the first case, the constraint can be:

$$u(t) \leq r_{min} \tag{13}$$

where r_{min} is the selected minimum value of cloud profit. Adding one of the constraints 11 or 12 to the optimization problem does not change the objective of the optimization,

which attempts to maximize the prices while satisfying the new constraints, (see Figure 4(b)).

If the altruistic tend is expressed as a new maximization goal, the optimization objective is a combination of Eq. 4 and Eq. 12:

$$\max_{\delta,p} R(t) = \int_0^T (r(t) - w \cdot u(t))dt \tag{14}$$

where w is a weight that calibrates the influence of the altruistic tend to the optimization procedure. The augmented optimization objective 14 leads the optimization procedure to seek a trajectory that balances the opposite egoistic and altruistic tends, (see Figure 4(c)).

IV. Modeling Price-Demand Correlations

The pricing scheme depends on the estimated values of price-demand correlations for all structures, which as stored in the matrix V (see the constraint 10). The key to the maximization of profit is the maintenance of collaborations and the elimination of competitions between structures, by pricing the structures appropriately. The success of the scheme depends greatly on the accuracy of the estimation of the correlation degree for all candidate structures. We refer to the elements, $v_{ij}, i, j = 1, \ldots, m$ of V, as *correlation coefficients*, defined as follows:

Definition 2: For any pair of structures S_i and S_j we define the symmetric correlation coefficient $v_{ij} \equiv v_{ji}$ that represents the combined usage of S_i and S_j in executed query plans.

A. Correlation Requirements

In order to construct a measure for correlation estimation, we define the following requirements[4].

Proposition 3: The correlation coefficient v_{ij} should satisfy the following requirements:

- R_1 v_{ij} is negative if S_i can replace S_j and the opposite, positive if they collaborate, and zero if they are used independent of each other in query plans.
- R_2 v_{ij} can be normalized for any pair of S_i and S_j.
- R_3 v_{ij} is easy to compute.

Justification 3: R_1: The sign of the coefficient v_{ij} denotes the competitive or collaborative behaviour between a S_i and S_j. If their presence does not affect each other, the coeficcient should be zero. We give an example.

Example 1: In a workload with only one query, Q `= select` A `from` T `where` B `= 'b' and` C `= 'c'` , the columns B and C should have positive correlation, while the indexes $I_{A-D} = T(A,B,C,D)$ and $I_{A-E} = T(A,B,C,D,E)$ should have negative correlation, and an irrelevant to the query index $T(E,F)$ should have zero correlation. It is straightforward that the pricing scheme requires these properties from the correlation coefficients V.

[4]Please note that the correlation requirements that we propose are tailored to the problem in hand. These requirement may be too strict for other use cases of management of data structures

R_2: The correlation coefficients V determine the price of all the structures in the cloud cache (see constraint 10). If their values are not normalized, the pricing scheme is biased towards specific structures with high coefficient values.

R_3: It is necessary to compute all correlation coefficients V before the structures are materialized or even selected by the cloud cache. Materialization and selection of cache structures is an online procedure performed for each query execution. Therefore, the correlation coefficients must be computed efficiently and scalably.

With respect to these requirements, we discuss a recently proposed correlation measure and its limitations. Then we propose a new measure that satisfies all the requirements.

B. Limitations of the Existing Approaches

Recently Schnaitter et al. [33] proposed a technique that computes the correlation between indexes. This section lists the limitations of this approach, while the limitations of other approaches is discussed in Section I-B.

Given a set of indexes $I \subseteq S$ and two indexes from the set, $\{S_i, S_j\}$, their correlation coefficient v_{ij}^q given a query q, is:

$$v_{ij}^q = \max_{X \subseteq I, \{S_i,S_j\} \setminus X} \frac{co_q(X) - co_q(X_i) - co_q(X_j)}{co_q(X_{ij})} + 1 \tag{15}$$

Where, co_q is a function that gives the cost of q given a set of indexes. The set X is a subset of I that does not contain the two indexes S_i and S_j. Moreover, $X_i \equiv X \cup \{S_i\}$, $X_j \equiv X \cup \{S_j\}$, and $X_{ij} \equiv X \cup \{S_i, S_j\}$. The above measure finds the maximum benefit that an index gives compared to another index for a given query and any subset of the set, normalized by the total cost of the query using both indexes. Since the query cost is monotonic, it is necessary that $co_q(X) > co_q(X_i) > co_q(X_{ij})$, $co_q(X) > co_q(X_j) > co_q(X_{ij})$.

Measure 15 does not satisfy the requirement R_1: for indexes that can replace each other the correlation is not negative. Since $co_q(X) > co_q(X_i) \approx co_q(X_j) \approx co_q(X_{ij})$, the measure is positive when the indexes are similar. It does not satisfy R_2 too: the produced values do not range in a bounded domain, therefore it is hard to perform normalization. Finally, it does not satisfy R_3: determining the coefficient requires exponentially large number of expensive optimizer calls even for a small I.

C. Structure Correlation Measure

We propose correlation measures that overcomes the limitations of the above technique. For indexes, we propose the measure:

$$v_{ij}^q = \frac{co_q(\{S_i\}) + co_q(\{S_j\}) - 2 \cdot co_q(\{S_i, S_j\})}{co_q(\{\}) - \min_{\{a,b\}} co_q(\{a,b\})} - 1 \tag{16}$$

Measure 16 identifies the individual benefits that the indexes S_i and S_j provide, and normalizes their sum w.r.t. the maximum benefit achievable by any pair of indexes $\{a,b\}$.

Proposition 4: Measure 16 satisfies the requirements $R_1 - R_3$.

Justification 4: R_1: We show that R_1 is satisfied by proving its satisfaction for the extreme cases of structure collaboration and competition. *Case 1:* If S_i and S_j do not co-exist in query plans, then let us assume that S_i is very beneficial to a query q, hence $co_q(X_i) \to 0$ and S_j has no effect on it, hence $co_q(X_j) \to co_q(\{\})$. Since the cost function is monotonic [33], $co_q(X_{ij}) = co_q(X_i) = \min_{\{a,b\}} co_q(\{a,b\}) \to 0$. Hence, $v_{ij} \to 0$. *Case 2:* If S_i and S_j collaborate tightly in the extreme case, $co_q(X_i) = co_q(X_j) \to co_q(\{\})$, but $co_q(X_{ij}) \to 0$. Then, $v_{ij} \to 1$. *Case 3:* If the indexes are the same, then $co_q(X_j) = co_q(X_i) = co_q(X_{ij})$, implying that $v_{ij} = -1$.
R_2: Since the cases discussed above are extreme, all structure correlation cases fall between them and, therefore their value is bounded by $[-1,1]$.
R_3: Section **??** proposes a method that ensures an efficient computation of the correlation coefficients.

For columns, we propose the following measure:

$$v_{ij}^q = \begin{cases} 1 & \text{if } S_i \neq S_j \text{ and both used in q} \\ -1 & \text{if } S_i = S_j \text{ and used in q} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

If two distinct columns appear in the same query, then they collaborate, otherwise they do not. Self-correlation for a column is set to -1, as a column can replace itself.

For a pair of index S_j and column S_i, we use the following measure:

$$v_{ij}^q = \begin{cases} 1 & \text{if } S_j \notin S_i \text{\& both can be used in q} \\ -1 & \text{if } S_j \in S_i \text{\& both can be used in q} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

The index and the column correlate if the index does not contain the column, and both are useful to the query. If the index contains the column then the column is redundant in presence of the index, therefore, they compete. Finally, if the above conditions are not satisfied, then they do not collaborate, therefore the coefficient is 0.

So far, we discussed correlation of structures w.r.t. a specific query. We extend the correlation computation for a workload. If v_{ij}^q is the correlation of S_i and S_j for query q, then the coefficient for an entire workload is:

$$v_{ij} = \frac{\sum v_{ij}^q co_q(\{\})}{\sum co_q(\{\})} \tag{19}$$

Measure 19 normalizes the coefficients by using the maximum cost of the query. This allows the "heavy" queries to provide more weight to the coefficient, when compared to the "lighter" queries.

Computing this measure requires $O(|I|^2)$ optimizer calls to determine the index correlation coefficients, compared to the exponential number of calls proposed by the state-of-the-art method, but it is still expensive to make so many optimizer calls on every query. We next describe techniques to reduce the computation overhead.

We speed up the correlation computation using the observation that, even though the total number of index combinations are $O(|I|^2)$ the set of possible plans is typically much smaller. The plans are typically tree structured, with the leaves accessing the indexes or the tables, and the internal nodes represent the aggregation or the joins. We observe in our earlier work–INUM [32]– that, on many occasions for different pair of indexes, the internal nodes remain exactly the same, and only the leaves change to reflect the change in the indexes. INUM uses a systematic method to identify the conditions on which the internal nodes change in a plan, therefore accurately identify the plans to be reused. Even INUM issues hundreds of calls to the optimizer to find the internal nodes of the plans that can be reused. Given access to the optimizer, the overhead can be drastically reduced to just two calls per query by using the internal optimizer structures [6].

V. Solving the Optimal Pricing Problem

The problem of optimal pricing is an optimal control problem [11] with a finite horizon, i.e. the maximum time of optimization T is a given finite value. The free variables are the prices of the cache structures, p_is, called the *control variables*, and the dependent variables, called *state variables*, is the demand for the structures, λ_is and the availability of the structures δ_is. The problem is augmented with bounds on the values of both the control and the state variables and by a constraint on the dependency type of the state on the control variables.

A. Designing the solution

The objective function of the problem is the maximization of an integral, i.e. $max \int_0^T (r(t) - w \cdot u(t))dt$. The optimality scope of the sought solution depends on the convexity of the objective function. The latter is bilinear w.r.t. the demand and the price (this is the result of factor $\lambda_S(t) \cdot p_S(t)$ in Eq. 2 and $p_S(t)$ in Eq. 12). It is not possible to prove that the objective function is convex and, therefore, there is no guarantee of global optimality of the solution.

Due to: (i) the nonlinearity of the objective function, (ii) the presence of both integer inputs (the δ_is control binary variables) and continuous inputs and states (the p_is and the λ_is , respectively), and (iii) the potentially large scale of the system (when m is high), it is almost impossible to find an analytical solution to the optimization problem. This calls for numerical optimization techniques, such as mixed-integer non-linear programming (MINLP) [11], which present the advantage of being implementable online. A way to implement dynamic optimization tools on real systems is to proceed as follows:

1) solve the MINLP problem along a fixed prediction horizon to compute a sequence of values for the control variables
2) apply the first values to the system
3) slide the prediction horizon and go back to 1)

This approach, referred to as *Optimal Control with Receding Horizon* or as *Model Prective Control* (for which a trajectory is tracked) in the control literature, has been successfully applied to a very large number of uncertain,

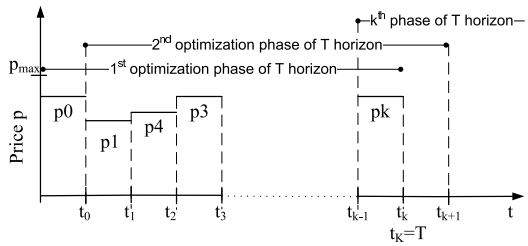

Fig. 5. The optimization procedure is divided into short time intervals and iterates on a sliding time window.

complex and nonlinear systems, in simulation as well at lab or industrial scales. This methodology has shown its ability to improve the performances of a large class of systems, despite the use of simplified models, the presence of uncertainty on model parameters, model mismatch, and process disturbances.

We propose the division of the prediction horizon $[0,T]$ into time intervals: let us assume that there are time points $t_j \in [0,T]$, $j = 0,\ldots,k$, such that $t_0 = 0$ and $t_k = T$ on which built structures can be built or discarded. Therefore the problem is to maximize the total profit in $[0,T]$ by choosing which structures to built or discard on each $t_j \in [0,T]$, $j = 1,..,k$ and which price to assign to each built structure:

$$\max_{\delta,p} R(t) = \sum_{j=0}^{j=k-1} \int_{t_j}^{t_{j+1}} \sum_{i=1}^{m} [\delta_i(t_j) \cdot (\lambda_{S_i}(t) \cdot p_{S_i}(t) - c_{S_i}(t))] dt \tag{20}$$

Figure 5 depicts the proposed repeated optimization over a sliding time prediction horizon of length T. For simplicity, we consider equal time intervals, $t_{j+1} - t_j = t_{j+2} - t_{j+1}, j = 0,\ldots,k-2$. The optimization is performed repeatedly for k prediction horizons beginning at t_{start} and ending at t_{end}, such that: $[t_{start}, t_{end}]$, $t_{start} = 0, t_1, \ldots, T$ and $t_{end} = T, T + t_1, 2T$, respectively. In this way we achieve, on one hand, to optimize by taking into account the inertia of the cloud behavior in a long prediction horizon, and on the other, to improve the optimization by tuning the initial values of both the control and the state variables at each time interval $[t_j, t_{j+1}]$ to the values predicted by the current optimization results. We can further improve the optimization procedure, by *injecting* the real values of the state variables, if these are available. Specifically, if the actual time is close to the starting time t_{start} of an optimization phase, then the real demand values of the structures are available; if the real values are different than the values predicted by the previous optimization phase, then the real values can substitute the predicted ones in the new optimization phase, calibrating the procedure towards an improved overall result.

We transform the problem into a MINLP one by substituting each control and state variable into a of k-arity set of variables, where k is the number of time intervals of control variable re-initialization in the optimization horizon, as well as the number of optimization repetitions. Formally:

$$\begin{aligned} p_i &\rightarrow P_i = \{p_{i_1}, \ldots, p_{i_k}\}, i = 1,\ldots,m \\ \lambda_i &\rightarrow \Lambda_i = \{\lambda_{i_1}, \ldots, \lambda_{i_k}\}, i = 1,\ldots,m \\ \delta_i &\rightarrow \Delta_i = \{\delta_{i_1}, \ldots, \delta_{i_k}\}, i = 1,\ldots,m \end{aligned} \tag{21}$$

For simplification, we consider all the control variables in a time interval to be static, which means that prices and availability of structures are constant. Application-wise, we assume that the availability of structures and their prices are set at the beginning time of each repetition of the optimization procedure. Of course, we could refine this simplification by considering prices to be functions of time in each interval. Yet, this would augment the number of variables dramatically, reducing the efficiency of the method. For example, even for linear dependency of price on time: $p = a \cdot t + b$ with static a, b, the number of variables in the problem is doubled.

B. Estimating the parameters

Concerning the constraints on the price-demand dependency in Eq. 10, it is necessary to estimate the parameters A, B, Γ. For this, the non-homogeneous m order system of second order differential equations in Eq. 10, has to be solved. One way to do is to transform the system into a $2 \cdot m$ order system of first order differential equations, by breaking each second order equation into a set of two. The result in both cases is a set of equations that show the dependency of demand on price involving the parameters:

$$\Lambda = F(t, A, B, \Gamma, \Lambda(0), \frac{d\Lambda}{dt}\Big|_{t=0}) \cdot P(t) \tag{22}$$

where F is a $m \times m$ matrix of functions on time and elements of the parameter matrices A, B, Γ, as well as the initial values of the demand and the rate of demand at the beginning of time. The solution of the system is possible, if the m constraints in Eq. 10 are independent, i.e. if the m differential equations are independent.

Proposition 5: It is always possible to manage the cache structures in a way that the constraints in Eq. 10 are independent differential equations.

Justification 5: Independency of the constraints in Eq. 10 means that there are no pair of cache structures for which the demand depends in the exact same way from the prices of all the cache structures. Intuitively, this is not a problem: assume two structures S_1 and S_2. If these are competitive, each one has a negative dependency on its own price and a positive dependency on the price of the other; therefore, it is not possible that they create the same constraint. If S_1 and S_2 are collaborative, creating the same constraint means that they depend on the exact same way on each other's price and on the price of the rest of the structures; this fact implies that S_1 and S_2 are always employed together in the cloud; therefore, they can be represented as a set of structures with a single price.

The parameters A, B, Γ can be estimated by performing curve fitting (e.g. the least square method), on Eq. 22. The fitting is performed based on a sample dataset of price-demand values. Ideally, we need a dataset with the values for Λ for all combinations of a set of price values $P_V \in [0, max_p]$, where max_p is a maximum value, for all price variables P. The fitting of Eq. 22 necessitates the initial values of demand and demand rate at the beginning of time. Since time is an orthogonal issue to the curve fitting

problem, we can order P_V and assume that for the fitting of each pair of data that consists of price values of all structures and the respective demand value $([p_{v_1},\ldots,p_{v_m}],\lambda_{v_i})$, $i=1,\ldots,m$, λ_{v_i} is the initial value of demand w.r.t. time. In order to get the initial value of demand rate at $t=0$, we need another measurement of demand for each structure λ'_{v_i} that is really close to λ_{v_i}, i.e. $\lambda_{v_i}-\lambda'_{v_i}<e_{\lambda_i}\approx 0$. This can be achieved by slightly changing the values in P_V, producing $P'_V=([p_{v_1}+e_1,\ldots,p_{v_m}+e_m], e_i\approx 0, i=1,\ldots,m$. We propose to estimate the demand rate as $\frac{d\lambda_i}{dt}\Big|_{t=0}=e_{\lambda_i}$, assuming that the smallest price change in two consequent observation time points is e_i .

C. Optimization horizon

An important issue is to estimate the appropriate length of the time period, in which we seek to optimize the cloud profit. Specifically, we have to determine the value of T which represents the optimization horizon of Eq. 4. Intuitively, a long horizon allows the optimization procedure to take into account the inertia of the system, whereas a short horizon may preclude the procedure from taking into account important long-term effects of current optimization decisions.

Example 2: Assume a structure S with demand $\lambda_S(t)$ and an optimization procedure of two short phases $[0,T_{small})$ and $[T_{small},T_{big})$ or a procedure with one long phase $[0,T_{big})$. For simplicity, the demand is a step function as shown in Figure 6, i.e. $\lambda_S(t)=\lambda_1, t\in[0,T_{small})$ corresponding to price p_1 and $\lambda_S(t)=\lambda_2, t\in[T_{small},T_{big})$ corresponding to price p_2 (for simplicity we ignore structure correlations). Assume that the building cost of S is B_S and the maintenance cost is $M_S(t)=a\cdot t$ and S is built once at time $t=0$. The cloud profit in $[0,T_{small})$ is $r_{small}=\lambda_1\cdot p_1-B_S-M_S(T_{small})$. If $r_{small}<0$, the cloud decides to discard S and the second optimization phase starts with S not available. Since the demand is significant in (T_{small},T_{big}), the cloud may decide to build S again, at $t\geq T_{small}$, resulting in profit $r_{big-small}\leq\lambda_2\cdot p_2-B_S-M_S(T_{big}-T_{small})$. For the long-term optimization the profit is: $r_{big}=\lambda_1\cdot p_1+\lambda_2\cdot p_2-B_S-M_S(T_{big})$. Obviously, $r_{big}>r_{small}+r_{big-small}$. Therefore, the result of the two-phase short-term optimization procedure is not as optimal as that of the one-phase long-term procedure.

Naturally, the prediction of future behavior of a system is subject to unpredictable perturbations. Hence, the longer the horizon is, the more error-prone the optimization procedure is, as the prediction accuracy of the behavior of demand, tends to decrease with time.

D. Discussion on the model simplicity

We have assumed that the parameters of the constraints in Eq. 10 are constant. Yet, it is possible that in a real system the dependency of demand on the prices changes with time, because of any reasons. This means that the parameters, A,B,Γ should be time-varying. Even though the dynamics of Eq. 10 would be more realistic, they would

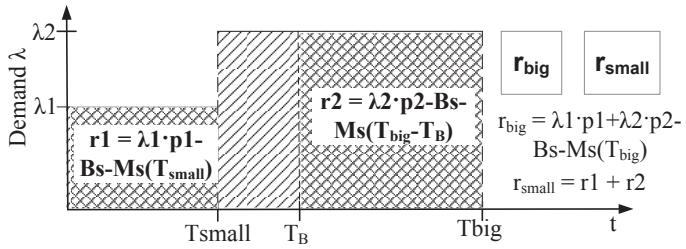

Fig. 6. The optimization procedure may give a higher profit if performed in a long time period.

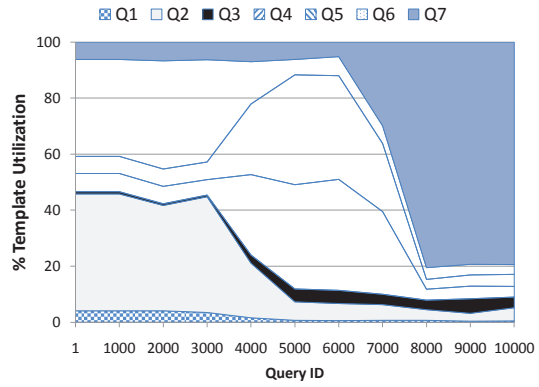

Fig. 7. The workload comprises phases of 10000 queries that are produced based on 7 TPC-H templates.

highly increase the complexity of the problem, as there is no way, without *a priori* knowledge to determine time varying parameters with more confidence than fixed parameters contrary to what can happen for physical systems where degradation, e.g., of physical parameters can be models.

Hence the problem falls in the scope of optimization of uncertain systems (potentially subject to model mismatch or parametric uncertainty or disturbances), which is an active research domain [12], [34]. In this context it can be shown that the use of measurements and of feedback is able to reject a part of the detrimental impact of parametric uncertainty on the optimal performances. In our case, real demand values are fed back as the optimization horizon slides, which increases the robustness of the proposed approach. As mentioned, *Model Predictive Control* has been widely used in Industry, where accurate dynamic models are almost never available. In these situations using tendency models (i.e. models that capture the main trends of a process) and measurements is generally sufficient to improve the process performances up to such a level that the costly efforts for identifying a more accurate process model are not justified by the loss of optimality [28]. Finally, as the optimization proceeds, new data is collected and this data can clearly be used to reidentify the price/demand model periodically.

VI. Experimental Evaluation

We present the simulation study for a cloud cache system that uses the proposed pricing model.

A. Experimental Setup and Methodology

Setup. The cloud cache is set up with one back-end database. The cache is operated under a TPC-H-based

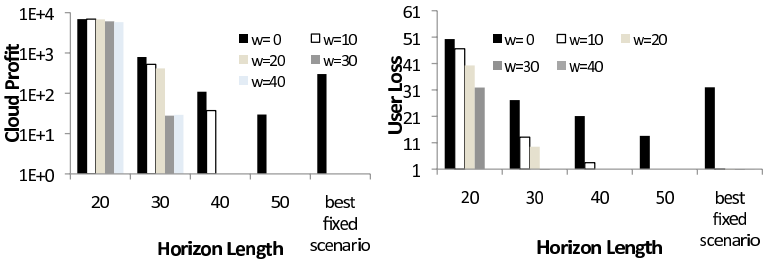

Fig. 9. Cloud profit and user loss using dynamic pricing on fixed structure availability.

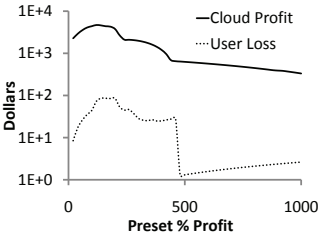

Fig. 10. Cloud profit using static pricing.

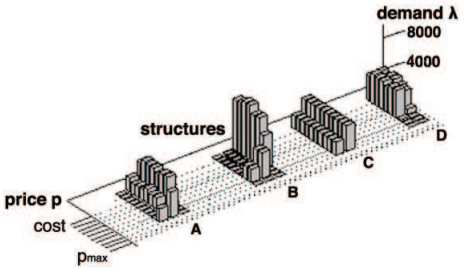

Fig. 8. Representative sets (A, B, C, D) of structures of the workload and their correlation w.r.t. the price-demand relation

workload, which consists of 7 TPC-H query templates and simulates the query evolution of 1 million SDSS [20] queries against a 2.5TB back-end database. The SDSS workload consists of phases that show locality in data access that repeats. In each phase the query execution cost may fall in 3 categories, low, medium and high. Queries arrive at 10 second intervals. We copy the setup in [24], where the workload simulates the change-of-columns co-occurrence over time for the SDSS workload. The authors first plot the column co-occurence matrix, and temporal locality of the columns in the SDSS workload. Then they select 7 queries and change the query composition over time to simulate similar column co-occurence and locality and the query execution cost. Figure 7 shows the distribution of the query templates in one phase consisting of 10000 queries. We select this workload, as it is portable across different DBMS, allows for the employment of techniques to improve the runtime of correlation estimations, and the queries are tunable by using the query generation mechanism of the TPC-H benchmark. The building and the maintenance costs are determined using Amazon's pricing model and are based on statistics for the cost of executing the SDSS queries. On average, the building cost is 7 orders of magnitude bigger than the maintenance cost. The detailed parameters for the setup are given in [7]. The pricing model decides on the building, maintenance, or destruction, and the pricing of 25 structures selected by a commercial physical designer. The correlation of the structures and the sensitivity of their demand in price changes is variable. As an indication, Figure 8 shows 4 sets of structures (A,B,C,D); while varying the price from the building cost ($cost$) to $p_{max} = 10 \cdot cost$, the demand varies from 0 up to 8000 queries, with many values around 4000. Set A contains two structures that collaborate, one more expensive than the other, and one that is competitive; set B is similar, but two expensive structures are highly competitive to a third that is cheap; set C contains two structures that are necessary to many queries and not correlated to others; set D contains two collaborative structures of comparable cost. The pricing optimization problem is implemented and run in Matlab 7.8.0 using the tool Tomlab [16].

Methodology. The initial demand for all structures is set to a very low value in order (i) to avoid high cloud profit by solely exploiting high demand values λ_is and (ii) force the pricing scheme to fluctuate λ_is in order to maximize the profit. The price variable for each structure ranges from 0 to 100% of the respective building cost, i.e. $0 \leq p_i \leq B_{S_i} * 100$. The experiments measure (i) the average *cloud profit* per time point, (ii) the average *user loss* per time point and (iii) the execution time. Cloud profit is defined in Eq. 2 and user loss is the user satisfaction as defined in Eq. 12. We present experiments for versions of the dynamic pricing model that vary the (i) weight w of the user satisfaction objective in Eq. 14, s.t. $0 \leq w \leq 40$, (default is $w = 0$) (ii) the length of the optimization horizon T in Eqs. 4 and 14, s.t. $20 \leq T \leq 50$, (default is $T = 50$) (iii) the size of optimization intervals (here called phases) t_j in Eq. 20 (by default set to 10 time points), and (iv) the price-demand functions in Eq. 8 – fitted in second order (default and as defined in Eq. 10) and first order differential equations. The dynamic pricing scheme is compared with a static pricing scheme that fixes the cloud profit to a specific percentage of the building cost. Also, we present results on the proposed correlation method concerning the quality of the estimations and the execution time.

B. Experimental Results

This section summarizes the experimental results.

1) Pricing with fixed structure availability: This section presents results on the dynamic pricing scheme assuming that all structures are constantly available (i.e. fixed caching), and, therefore built once in the cache at the beginning of pricing and maintained ever since, i.e. $\delta_i = 1, i = 1,..,m$ always. The problem boils down to pricing the structures so that the cloud gains maximum profit while ensuring that the demand is not drastically reduced because of the pricing. Figure 9 shows the profit generated by dynamic pricing as a function of different optimization horizon lengths for various weight values w. As the optimization horizon is extended the profit drops because structures are maintained in the cache even though their demand drops; the user loss drops too, but with a slower rate.

Naturally, the bigger the weight w, the smaller the profit and the user loss. Yet, for long horizons, the maintenance of non-profitable structures makes it impossible to satisfy the combined optimization objective in Eq. 14 for big values of weight, i.e. $w = 30, 40$, resulting in zero profit and user loss. Figure 8 shows also the profit and user loss for the best fixed pricing and fixed availability scheme: assuming that we have complete knowledge of the workload, we select the *best* structures to build at the beginning of time. The *best* structures are selected after observation of the matrix V (we spotted groups of collaborative and competitive structures and we experimented in order to find the subset that increases profit; the combinations to examine were few). Experimentation with various fixed prices of these structures resulted in maximum possible profit equal to about \$400 and user loss equal to about \$30.

We compare the profit made using dynamic pricing with that made using static pricing that charges each structure $10-1000\%$ more than the actual building cost, and does not consider the correlation between the structures; Figure 10 shows the profit as a function of the fixed profit percentage. As the preset profit percentage increases, the cloud profit increases up to about \$4700 which occurs at about 120%, while the maximum profit for dynamic pricing with fixed availability is about \$8000. Beyond 120%, the profit drops gradually. The reason for this drop is the inverse correlation of price and demand: a very high price reduces the demand to zero and the high price does not compensate for the reduced demand. At almost 500% preset for profit the user loss drops sharply to close to zero values. The user loss remains low and comparable to that of dynamic pricing; when the profit does not grow for high preset values, the user loss grows because the user pays high prices for the small number of structures which are still in demand. The results of this experiment are in accordance with the results of the works in [37].

2) Pricing with choice on structure availability: This section presents results on the dynamic pricing scheme assuming that structures are initially built in the cache, but during optimization they can be discarded and re-built. Figure 11 shows that the choice on structure availability increases the average profit by two orders of magnitude and decreases the user loss by one order of magnitude, on average w.r.t. the horizon length. Contrary to pricing with fixed availability, the profit increases as the horizon is extended. The reason is that the optimization procedure takes advantage of long-term predictions in order to schedule the structure availability in a more optimal way.

3) Sensitivity to the optimization schedule: The optimization procedure is sensitive to the horizon length, the number of optimization intervals t_j in Eq. 20 and their length, as shown in Figure 16. Keeping the total time of optimization fixed, the profit increases as the number of intervals increases (and, therefore their length decreases), because the procedure is allowed to change the structure availability more often, in order to achieve optimality. Nevertheless, the effect of increasing the number of intervals is faded out if the optimization is repeated in multiple horizons, rather than performed for one long horizon. Naturally, since $w = 0$, the user loss increases and drops if the profit increases or drops while the optimization horizon remains the same. As the number of horizons increases, the profit decreases (and therefore their length decreases) because the procedure cannot predict adequately the demand change.

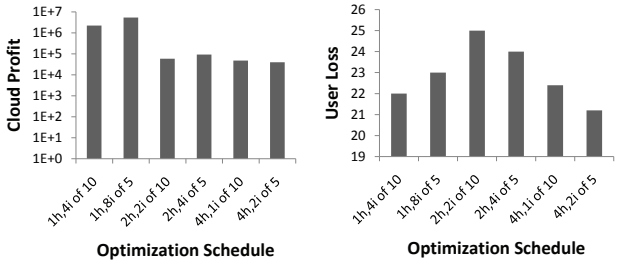

Fig. 16. Profit and loss for various optimization schedules. The label-xh, yi of z-represents x horizons, with y intervals of z time units each.

4) Performance comparison: We compare the performance of the optimization procedure employing first and second order differential equations for the pricing model. Models using first order equations are faster to solve, hence preferred over second-order differential equations if the real-world constraint can be modeled using them. Figure 12 shows using a first-order differential equation makes the procedure slightly faster than using a second-order differential equation. The second-order formulation, however, is more generic and we use it as default.

Figure 12 also shows that relaxing the δ variable makes the solver an order of magnitude faster than the problem with δ variables on average. Therefore, the solver spends most of the time in the branch and bound method that seeks the optimal integer values [16]. The reason is that the problem is not convex–the solver cannot easily determine the lower bounds for pruning search branches.

5) Correlation of structures: This section presents the index correlations achieved using Eq. 16 and compares the proposed measure for correlation coefficients Eq. 19 with the state-of-the-art measure Eq 15 [33]. We name the measure Eq. 15 "SPG-measure". We show the trade-off of performance against the accuracy of the cost estimation procedure. Figure 13 shows the distributions of about 500 index correlations sampled randomly from the candidate indexes. The correlations computed using Eq. 16 is distributed both in the positive and negative values, showing that the measure detects both positive and negative correlations. Furthermore, it is also bounded by the range $[-1, 1]$. In comparison, most of the correlations computed using Eq. 15 are positive and have value close to zero. SPG-measure is useful if only top interacting indexes are interesting; if the problem requires correlation estimation between all pairs of structures, SPG-measure fails to distribute the correlations in the target range.

6) Optimization in presence of updates: The optimization procedure works under the assumption that data structures do not have to be evicted and rebuilt due to data updates. Even though updates cannot be controlled by the optimization procedure, if they can be predicted, they can be used as new constraints on the optimization problem.

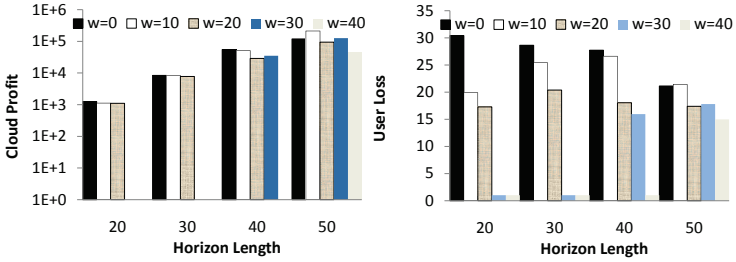

Fig. 11. Cloud profit and user loss using dynamic pricing with optional structure availability.

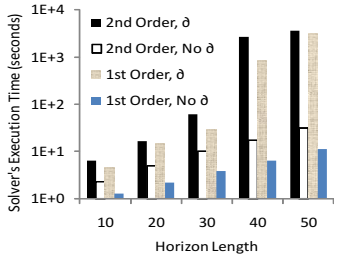

Fig. 12. Scaling of 1st and 2nd order differentiation

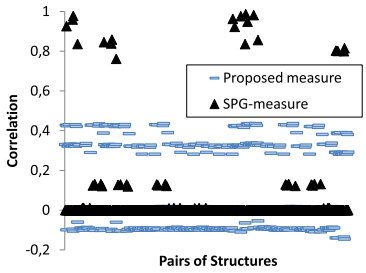

Fig. 13. Comparison of the estimated correlations using two measures.

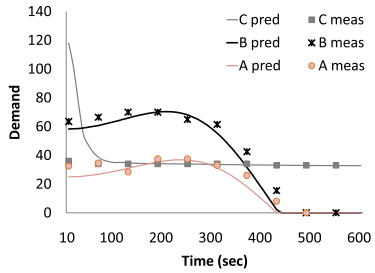

Fig. 14. Prediction of demand change and real demand change in time.

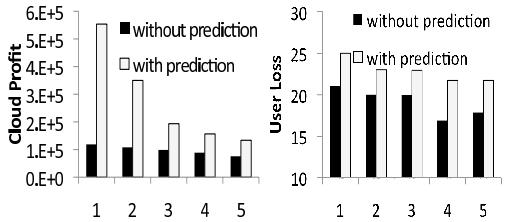

Fig. 15. Optimization using or not predictions for updates for 1-5 updates on average per structure.

Specifically, an update of structure S at time t incurs a reset of the respective δ parameter from 1 to 0 at that time. Figure 15 shows the results of optimization in case update times for structures are predicted or not. The results are for 1 up to 5 updates on average per each structure. The cloud profit is bigger if updates are predicted. Yet, as the number of updates increases, the profit drops and is closer to profit in the case of no update prediction. User loss is bigger ($w = 0$ for these experiments) in case of update prediction, since the optimization sets higher prices for the structures.

7) Predicting the demand for structures: Figure 14 shows the comparison of the real demand fluctuations after price change with the predictions of the differential equations that model the price-demand relation (the parameter estimation of the model precedes this procedure). The figure shows the comparison for three structures for which the price was changed from the building cost to 10 times the latter. The demand for these structures shows qualitative differences: the demand for A reacts smoothly to price change after some weak inertia to the workload; the demand for B shows similar inertia but after that it drops abruptly; the demand for C shows great inertia to the workload (this is an indication of a necessary structure to query execution). All three demand fluctuations are predicted very accurately by the respective differential equation, which exhibits the flexibility of the proposed price-demand model.

VII. Conclusions

This work proposes a novel pricing scheme designed for a cloud cache that offers querying services and aims at the maximization of the cloud profit. We define an appropriate price-demand model and we formulate the optimal pricing problem. The proposed solution allows: on one hand, long-term profit maximization, and, on the other, dynamic calibration to the actual behavior of the cloud application, while the optimization process is in progress. We discuss qualitative aspects of the solution and a variation of the problem that allows the consideration of user satisfaction together with profit maximization. The viability of the pricing solution is ensured with the proposal of a method that estimates the correlations of the cache services in an time-efficient manner.

References

[1] Gabriel R. Bitran and Rene Caldentey. An overview of pricing models for revenue management. In *Manufacturing & Service Operations Management*, volume 5, pages 203–209, 2003.

[2] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. *ICDE'07*.

[3] Xi-Ren Cao, Hong-Xia Shen, R. Milito, and P. Wirth. Internet pricing with a game theoretical approach: concepts andexamples. *Networking, IEEE/ACM Transactions on*, 10(2):208–216, 2007.

[4] Ch Chen, Muthucumaru Maheswaran, and Michel Toulouse. Supporting co-allocation in an auctioning-based resource allocator for grid systems. In *IPDPS*, 2002.

[5] S. Choenni, H. M. Blanken, and T. Chang. On the selection of secondary indices in relational databases. *DKE*, 11(3), 1993.

[6] D. Dash, Y. Alagiannis, C. Maier, and A. Ailamaki. Caching all plans with one call to the optimizer. In *SMDB*, 2010.

[7] D Dash, V. Kantere, and A. Ailamaki. An economic model for self-tuned cloud caching. In *ICDE, workshop of SMDB*, 2009.

[8] Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Economic scheduling in grid computing. In *SSPP*, 2002.

[9] Guillermo Gallego and Garrett van Ryzin. Optimal Dynamic Pricing of Inventories with Stochastic Demand over Finite Horizons. *Management Science*.

[10] A. Ghose, V. Choudhary, T. Mukhopadhyay, and U. Rajan. Dynamic pricing: A strategic advantage for electronic retailers. In *CIST*, 2003.

[11] IE. Grossmann and Z. Kravanja. *Large-scale Optimization with Applications: Optimal design and control*. Springer, 1997.

[12] M. Guay and T. Zhang. Adaptive extremum seeking control of nonlinear dynamic systems with parametric uncertainty. *Automatica*, 39:1283–1294, 2003.

[13] Linhai He and J. Walrand. Pricing differentiated internet services. In *INFOCOM*, pages 195–204, 2005.

[14] http://aws.amazon.com/.

[15] http://code.google.com/appengine/.

[16] http://tomopt.com/tomlab/.

[17] http:/www.cern.ch/.

[18] http://www.gogrid.com/.

[19] http://www.microsoft.com/azure/.

[20] http://www.sdss.org/.
[21] Markus Kradolfer and Dimitrios Tombros. Market-based workflow management. *IJCIS*, 7, 1998.
[22] Jiadao Li and Ramin Yahyapour. Negotiation model supporting co-allocation for grid scheduling. In *ICGC'06*.
[23] Z. Lin, S. Ramanathan, and H. Zhao. Usage-based dynamic pricing of Web services for optimizing resource allocation. *Information Systems and E-Business Management*, 3(3), 2005.
[24] T. Malik, X. Wang, R. Burns, D. Dash, and A. Ailamaki. Automated physical design in database caches. In *SMDB*, 2008.
[25] 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 in Grid and Pervasive Computing*, pages 13–24, 2009.
[26] Vladimir Marbukh and Kevin Mills. Demand pricing & resource allocation in market-based compute grids: A model and initial results. In *ICN*, pages 752–757, 2008.
[27] Y. Masuda and S. Whang. Dynamic Pricing for Network Service: Equilibrium and Stability. *Management Science*.
[28] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Comput. Chem. Engng*, 23:667–682, 1999.
[29] Rafael A. Moreno. A.B.: Job scheduling and resource management techniques in economic grid environments. In *Across Grids 2003*, pages 25–32, 2004.
[30] Y. Narahari, C. V. L. Raju, K. Ravikumar, and S. Shah. Dynamic pricing models for electronic business. In *Whiite Paper. Indian Institute of Science*, 2005.
[31] Series of meetings of the EPFL-IC-IIF-DIAS lab with the data management group of the European Organization for Nuclear Research (CERN) started on the December 9th 2008.
[32] S. Papadomanolakis, D. Dash, and A. Ailamaki. Efficient use of the query optimizer for automated database design. In *VLDB*, pages 1093–1104, 2007.
[33] K. Schnaitter, N. Polyzotis, and L. Getoor. Modeling index interactions. In *VLDB*, 2009.
[34] B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki. Dynamic optimization of batch processes: Ii. role of measurements in handling uncertainty. *Comput. Chem. Engng*, 27:27–44, 2003.
[35] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. *VLDB J.*, 5(1):48–63, 1996.
[36] A. Sulistio, K. Kyong Hoon, and R. Buyya. Using revenue management to determine pricing of reservations. In *IEEE e-Science*, pages 396–405, 2007.
[37] Xiaodan Wang, Tanu Malik, Randal C. Burns, Stratos Papadomanolakis, and Anastasia Ailamaki. A workload-driven unit of cache replacement for mid-tier database caching. In *DASFAA*, pages 374–385, 2007.
[38] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jeffrey K. Mackie-mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:2001, 1998.
[39] Kyu-Young Whang, G. Wiederhold, and D. Sagalowicz. Separability: An approach to physical database design. *IEEE Transactions on Computers*, 33(3):209–222, 1984.
[40] Peng-Sheng You and Ta Cheng Chen. Dynamic pricing of seasonal goods with spot and forward purchase demands. *Comput. Math. Appl.*, 54(4):490–498, 2007.

Verena Kantere Verena Kantere is a tenure-track lecturer at the Department of Electrical Engineering and Information Technology at the Cyprus University of Technology. She has received a Diploma (2000) and a Ph.D. (2007) from the National Techincal University of Athens, (NTUA) and a M.Sc. degree from the University of Toronto (2003). During her graduate studies her research interests focused on problems of data exchange and coordination in Peer-to-Peer (P2P) overlays with structured and unstructured data, as well as multidimensional data sharing. Until recently, she worked as a postdoctoral researcher (2008-2010) at the Ecole Polytechnique Federale de Lausanne (EPFL). Her research focuses on the provision of cloud data services and proposes solutions for the incorporation of cost in existing data management techniques.

Debabrata Dash Debabrata Dash is a Senior Scientist at ArcSight (an HP Company). He completed his PhD on database management systems from Carnegie Mellon University. His thesis is titled "Automated Physical Design: A Combinatorial Optimization Approach" and studies the mathematical modeling of the query optimizers in the context of physical design. Furthermore, it applies the model to improve the quality of the suggested design as well as the efficiency of the physical designers. He holds a B. Tech. degree from the Indian Institute of Technology at Kanpur.

Grégory François Grégory François received in 1998 his Chemical Engineering Diploma from the French National Chemical Engineering School of Nancy, France, and his M.Sc. in Chemical Engineering from the "Institut National Polytechnique de Lorraine", also in France. In 2004, he received his PhD degree from the Automatic Control Laboratory from EPFL. His research focused on measurement-based optimization of dynamic processes. From 2005 to 2006, he worked as a postdoctoral researcher at the Industrial Implementation Group of the Laboratory of Composites and Polymer Technology of EPFL . He performed technical cost-modeling and optimization for the high-volume manufacturing of composites part for the automotive industry. Since 2006 he has been holding a Tenure Assistant Professor position in the University of Perpignan, France. There, he participated to the creation of the Chemical Engineering Department, while performing research on modeling, control and optimization of renewable energy production and consumption, and of waste water treatment via anaerobic digestion. In 2008 he joined back the Automatic Control Laboratory of EPFL, where he works as a Senior Research Associate and Lecturer. His research covers the fields of automatic control, optimal control, and static or dynamic optimization of continuous or discontinuous processes. He also gives the lectures on Optimal Control theory at the Doctoral School of EPFL .

Sofia Kyriakopoulou Sofia Kyriakopoulou is a M.Sc. student in Computer Science at EPFL. She holds a Diploma in Electrical and Computer Engineering from NTUA. She has worked on networking-related projects at Nokia Siemens Networks Research, in Athens and has been a member of the Information Management Unit group of Institute of Communication and Computer Systems, at NTUA working on projects concerned with semantic technologies deployed in collaborative and personal data management.

Anastasia Ailamaki Anastasia (Natassa) Ailamaki is a Professor of Computer Science at EPFL. She earned her Ph.D.in Computer Science from the University of Wisconsin-Madison in 2000. Her research interests are in database systems and applications; in particular (a) in strengthening the interaction between the database software and the underlying hardware and I/O devices, including flash technology, and (b) in automating database design and computational database support for scientific applications. She has received a Finmeccanica endowed chair from the Computer Science Department at Carnegie Mellon (2007), a European Young Investigator Award from the European Science Foundation (2007), an Alfred P. Sloan Research Fellowship (2005), six best-paper awards at top conferences (2001-2006), and an NSF CAREER award (2002).