N.D.Sheba kumari, B.Bharathi / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue4, July-august 2012, pp.2311-2320

Pre-Eminent Pricing For Cloud Cache

N.D.Sheba kumari, M.Tech(CS), B.Bharathi,Asst.Prof.,
ADAMS Engineering College Kammam,INDIA
ADAMS Engineering College Kammam,INDIA

Abstract

Caching on the edge of the Internet is becoming a popular technique to improve the scalability and efficiency of delivering dynamic web content and it is rising. These clouds inturn helps in providing good quality query services.In order to query the data users pay for the infrastructure they use required for the query. The goal of cloud economy is to optimize user satisfaction and cloud profit.To increase the cloud profit an oppropriate price demand model should guarantee the user satisfaction that enables the best possible pricing of query services. The model should be reliable in that it reflects the correlation of cache structures involved in the queries. Best possible pricing is achieved based on a dynamic pricing scheme that adapts to time changes. This paper proposes a new 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 new 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,price demand modeling,Dynamic Pricing

.

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:1) user satisfaction and 2) 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 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 and Microsoft Azure 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, need a caching service, which can be provided by the cloud. 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. Fig. 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.

1.1 Price Setting for Cloud Cache 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 nontrivial 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 1) modeling errors, 2) time-dependent model changes, and 3) stochastic behavior of the application.

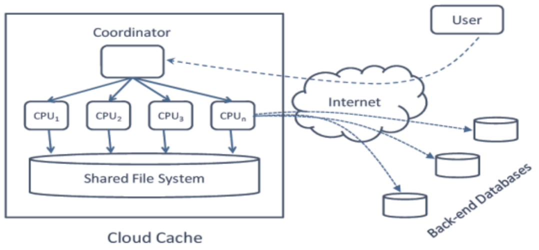

Fig. 1. A cloud cache.

The demand for services, for instance, may depend in a nonpredictable 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 longterm 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.

1.2 Related Work

Existing clouds focus on the provision of web services targeted to developers, such as Amazon Elastic Compute Cloud (EC2) , or the deployment of servers, such as GoGrid . 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 and Microsoft Azure 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, 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. 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 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, or to achieve an equilibrium of grid and user satisfaction, 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 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, 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.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, seasonal products, stochastic demand. Electronic businesses, and therefore cloud businesses can benefit from dynamic pricing policies.Cache services are distinguished from consumable products in two major ways: 1) they are not exhausted while they are consumed, and 2) 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 noncorrelated indexes using the query structure does not determine the negative and positive correlations. Identification of index correlations by modeling physical design as a submodular and super-modular problem 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 attempts to find all index correlations. As we show in Section 4, it does not satisfy three critical requirements for the pricing scheme: 1) sensitivity to the range of all possible correlations, 2) production of ormalized values, and 3) fast computation.

1.3 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 reevaluated 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 nonoverlapping 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 reinjection of the real demand values at each sliding instant. Also, the iterative optimization allows for redefinition of the parameters in the price-demand model, if the demand deviates substantially from the predicted.

1.3 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 nonlinear 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 2 presents the query execution model, Section 3 models the optimal pricing problem, and Section 4 models the price demand correlations for data structures in the cloud cache. Section 5 describes the solution of the pricing optimization problem and Section 6 presents the experimental study. Section 7 concludes the paper.

2. 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 multiuser 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, for example physics data from CERN and astronomy data from SDSS . 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.

Alogorithm:
Global: cache structures *S*, prices *P*, availability Δ
Query Execution ( )
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
OptimalPricin (*T*, *t*[*i*], *S*)
end for
T2:
While new query *q* **do**
(Result, cost)←query Execution (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. Qurey Execution Model for cloud cache

Fig. 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 back-end 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.Periodically (on predefined time intervals $t[i]$) the cloud performs the pricing scheme proposed in this work. The pricing scheme schedules the availability 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 at the same time ensure that the user is not overcharged.

3.MODELING DYNAMIC 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.

3.1 Problem Formulation

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

3.1.1 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 $S = \{S_1, \ldots, Sm\}$. 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 backend 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.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

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

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 1) keeping a structure in the cache and paying the maintenance cost, and 2) building and discarding the structure occasionally. This trade-off is reinforced toward 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. 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

$$r(t) = \sum_{i=1}^{m} \delta_i . (\lambda s_i(t) . ps_i(t) - cs_i(t)), \delta_i = 0,1, \qquad (2)$$

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.,

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

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

$$c_s(t) = \left(1 - \delta_i(t_0)\right) B_s + M_s(t - t_0), \qquad (3)$$

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

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

3.1.2 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.These bounds can be formulated as follows:

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

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

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:

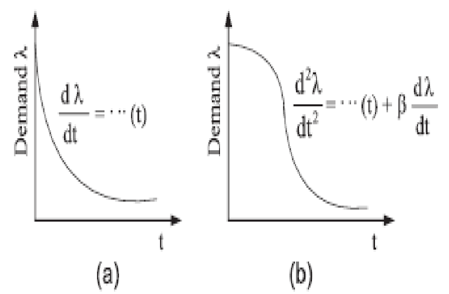

Fig. 3. The shape of the demand versus time function based on a (a) first and (b) second order differential equation.

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

$$f\left(\frac{d^n\lambda_S}{dt^n}, \frac{d^{n-1}\lambda_S}{dt^{n-1}}, \dots, \frac{d\lambda_S}{dt}, \lambda_S(t), \frac{d^m p_S}{dt^m}, \dots, \frac{dp_S}{dt}, p_S(t)\right) = 0, \qquad (7)$$

where m ≤ 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 (7) can be rewritten in its explicit form

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

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: 1) the structure is already built and remains available because the building cost is already amortized, while the maintenance cost is not very high; and 2) 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 identify their values. We choose to consider a second 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 Fig. 3, while keeping the number of parameters to identify low. Therefore, the constraint is:

$$p_S(t) = f\left(\frac{d^3\lambda_S}{dt}\right).$$

We constrain f to be an ordinary differential relation between price and demand

$$p_s(t) = \alpha\frac{d^2\lambda_s}{dt^2} + \beta\frac{d\lambda_s}{dt} + \Upsilon.\lambda_s(t). \qquad (9)$$

The parameters α,β,ϒ 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 (9) so that it captures correlations of demand and prices between pairs of structures. Let us assume that V is a m × m matrix where the row and the column i corresponds to the structure S_i i = 1, . . .,m. Each element v_{ij},i, j = 1, . . .,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 ∧ and P are the m × 1 matrices of demands and prices for the respective structures in S, and A, B, Γ are m × 1 matrices of parameters, then the constraint in (9) becomes

$$V.P = A^T\frac{d^2\wedge}{dt^2} + B^T\frac{d^2\wedge}{dt}\Gamma^T\wedge, \qquad (10)$$

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

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:

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

subject to the constraints:

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

3.2 Generalization of Optimization Objective

The problem of dynamic pricing as formulated in Section 3.1 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 (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, (6), and demand values, (5), meaning maximum price selections as long as the demand for structures is above zero. 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 _. Intuitively, the objective of optimization is the purely egoistic and straightforward maximization of cloud profit. The optimization procedure 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: 1) a guarantee for a low limit on user satisfaction, or 2) 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 (4). 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 3.1 on the demand drop

$$\frac{d\Lambda}{dt} \geq \left.\frac{d\lambda}{dt}\right|_{min},$$

where λ_{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

$$u(t) \equiv p_s(t) - c_s(t) \quad (12)$$

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

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

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 Fig. 4b).
If the altruistic tend is expressed as a new maximization goal, the optimization objective is a combination of (4) and (12)

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

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.

4 SOLVING THE DYNAMIC PRICING PROBLEM

The problem of optimal pricing is an optimal control problem 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.

4.1 Design of the Solution

The objective function of the problem is the maximization of an integral, i.e., $max \int_0^T (r(t) - w.utdt$. 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 $\lambda_s(t).p_s(t)$ in (2) and $p_s(t)$ in (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: 1) the nonlinearity of the objective function, 2) the presence of both integer inputs (the δ_is control binary variables) and continuous inputs and states (the p_is and the δ_is, respectively), and 3) 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 mixedinteger nonlinear programming, which present the advantage of being implementable online. A way to implement dynamic optimization tools on real systems is to
proceed as follow:

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, 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 $t_j \epsilon [0,T], j = 0, \dots, k,$ 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

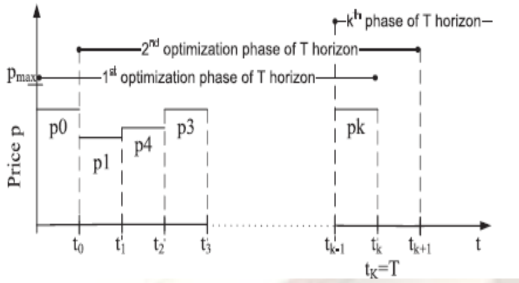

*Fig.4.*optimization procedure divided into short time intervals and iterations on a sliding time window

which structures to built or discard on each $t_j \epsilon [0,T], j = 0, \dots, k,$ and which price to assign to each built structure

$$\max_{\delta,p} R(t) = \sum_{j=0}^{j=k-1} \int_{t_j}^{t_{j+1}} \sum_{i=1}^{m} [\delta_i(t) \cdot (\lambda_{S_i}(t) \cdot p_{S_i}(t) - c_{S_i}(t))] dt. \quad (15)$$

Fig. 4 depicts the proposed repeated optimization over a sliding time prediction horizon of length T. For simplicity, we consider equal time intervals, $t_{j+1} - t_j = t_{j+2} - t_{j+1}, i = 0, \dots, k-2.$ The ptimization is performed repeatedly for k prediction horizons beginning at t_{start} and ending at tend, such that: $[t_{start}, t_{end}], t_{start} = 0, t_1, \dots, T$ 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 tstart 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 toward 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 einitialization in the optimization horizon, as well as the number of optimization repetitions. Formally

$$\begin{aligned} p_i &\rightarrow P_i = \{p_{i_1}, \dots, p_{i_k}\}, & i &= 1, \dots, m, \\ \lambda_i &\rightarrow \Lambda_i = \{\lambda_{i_1}, \dots, \lambda_{i_k}\}, & i &= 1, \dots, m, \\ \delta_i &\rightarrow \Delta_i = \{\delta_{i_1}, \dots, \delta_{i_k}\}, & i &= 1, \dots, m. \end{aligned} \quad (16)$$

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: $p = a.t + b$ with static a, b, the number of variables in the problem is doubled.

4.2 Parameters Estimation

Concerning the constraints on the price-demand dependency in (10), it is necessary to estimate the parameters A, B, Γ. For this, the nonhomogeneous m order system of second order differential equations in (10), has to be solved. One way to do is to transform the system into a $2 \times m$ 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

$$\Lambda = F\left(t, A, B, \Gamma, \Lambda(0), \frac{d\Lambda}{dt}\Big|_{t=0}\right) \cdot P(t), \quad (17)$$

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 (10) are independent, i.e., if the m differential equations are independent.

Proposition 3. It is always possible to manage the cache structures in a way that the constraints in (10) are independent differential equations.

Justification ***3.*** Independency of the constraints in (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 (17). The fitting is performed based on a sample data set of pricedemand values. Ideally, we need a data set with the values for ∧ for all combinations of a set of price values $p_v \epsilon \left[0, max_p\right]$, where max_p is a maximum value, for all price variables P. The fitting of (17) 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 $\left(\left[p_{v_1,\dots,}p_{v_m}\right], \lambda_{v_i}\right), i = 1, \dots\dots, m, \lambda_{v_i}$ 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 $p'_v = (\left[p_{v_i} + e_1, \dots . p_{v_m} + e_m\right], e_i \approx 0, i = 1, \dots ., m$. We propose to estimate the demand rate as $\left.\frac{d\lambda_i}{dt}\right|_{t=0} = e_{\lambda_i}$, assuming that the smallest price change in two consequent observation time points is e_i.

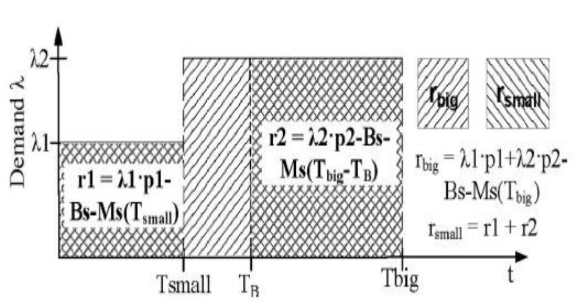

Fig.5.The optimization procedure may give a higher profit if performed in a long time period

4.3 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 (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 Fig. 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 $M_s(t) = a.t$and S is built once at time t = 0. The cloud profit in $[0, T_{small})$ is $r_{small} = \lambda_1 . p_1 - B_s - M_s(T_{small})$. 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}) , th(e cloud may decide to build S again, at $t \geq T_{small}$, resulting in profit $r_{big-small} \leq \lambda_2 . p_2 - B_s - M_s(T_{big} - T_{small})$. For the long-term optimization the profit is: $r_{big} = \lambda_1 . p_1 + \lambda_2 . p_2 - B_s - M_s(T_{big})$ 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.

4.4 Simplicity of the Model

We have assumed that the parameters of the constraints in (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 (10) would be more realistic, they would 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 modeled. 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.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.

Finally, as the optimization proceeds, new data are collected and this data can clearly be used to reidentify the price/demand model periodically.

5 CONCLUSION

We define an appropriate price-demand model and we formulate the dynamic pricing problem. The proposed solution allows: on one hand, longterm 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. This paper proposes a novel pricing scheme designed for a cloud cache that offers querying services and aims at the maximization of the cloud profit.