International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-2, Issue-3, July 2012

Optimal Service Pricing for Cloud Based Services

Deepak Mishra, Manish Shrivastava

Abstract: **-Cloud computing is a profound revolution in the way it offers the computation capability. The main objective now is to reduce the cost of deploying a service in the cloud and having proper coordinative in between models. Public, private, and hybrid cloud environments all face the performance limitations inherent in today's applications and networks. In order for enterprises to maximize the flexibility and cost savings of the Public, private, and hybrid cloud they must overcome the same latency and bandwidth constraints that challenge distributed IT infrastructure environments. By overcoming application and network performance problems, Cloud Steelhead accelerates the process of migrating data and applications to the cloud, and accelerates access to that data from anywhere Cloud Computing 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. Optimal pricing is achieved based on a dynamic pricing scheme that adapts to time changes. This 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 a time-efficient manner and also applied some prediction technique in between correlation models with the use of cooperative cache from self as well as different hybrid cloud.**

Index Terms—cloud data management, data services, cloud service pricing, Cooperative Cache, Prediction Technique.

I. INTRODUCTION

cloud computing represents a new tipping point for the value of network computing. It delivers higher efficiency, massive scalability, and faster, easier software accessibility. It's about new programming models, new IT infrastructure, and the enabling of new business models. 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 2) cloud profit 3) Cloud web Security. 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 and apply the some intelligence optimization technique 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], Microsoft Azure [19], VMware [41], Google Plus [41], McAfee Cloud Searching Virus [42], Oracle Cloud Data Management [43] have begun to offer data management services:. A used web application passes a query to the server and collects information with the help of query massive data, like those supported by CERN [17], need a caching service and intelligent optimization technique which can be provided by the cloud [31].

The goal of such a cloud is to provide efficient proper way querying on the back-end data at a low cost with intelligent manner while being economically Viable, and furthermore, optimal profitable and also getting reduction of scheduling cost on demand changes. A price over the operating cost for each structure can ensure profit for the cloud. And also internally cloud is periodically update his caches information as required on different cloud and other demand will virtually manages a different dedicated server at a time on cloud. We propose a novel scheme with some pattern recognition technique by the soft computing hat achieves optimal pricing for the services of a cloud cache with explain on demand prediction cost.

1.1 Setting the on Demand Price for Cloud based Caching Services with Prediction cost estimation on Demand model.

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 three major challenges when trying to define an optimal pricing scheme for the cloud caching service with on demand cost prediction. First one 1) the price demand Dependency, to achieve a feasible pricing solution, but not economically feasible as required on demand same services, that is not representative. For Example, a static pricing scheme cannot be optimal if the Demand for services has deterministic seasonal fluctuations; in this case correlation is automatically stopped on between different clouds when demand is increases on particular services because everyone wants to get on demand profit. The second challenge is to define a pricing scheme that is adaptable to 1) modeling errors, 2) Mean Percentage Error (MPE) on load distribution, 3) time-dependent model changes, and 4) stochastic behavior of the application.

A representative model for the cloud cache and prediction techniques should take In to account that the cloud cache structures (table columns or Indexes or correlation cache identification number) may compete or collaborate during query execution and any such a situation will come it will take self-decision on demand load balance, cloud services may be free or offered on a pay-per-usage model, No wasted resources because you pay for what you use [47], A cloud storage gateway provides basic protocol translation and simple connectivity to allow the incompatible technologies to communicate transparently [48]. And the goal is to minimize the number of cross-node distributed transactions, which incur overhead both because of the extra work done on each node and because of the increase in the time spent holding locks at the back-ends. OODBMS caching is keep only positive impact factor or successfully satisfying the information. So by the using cloud optimized cache we directly minimize the following cost 1) **Hash**

Manuscript Received on July 04, 2012.

Deepak Mishra, Information Technology, RGPV/ LNCT/ LNCT, Bhopal (M.P.), India,

Dr. Manish Shrivastava, Information Technology, RGPV/ LNCT/ LNCT, Bhopal (M.P.), India

join costs, 2) **Sort costs,** 3) **Table scan costs, 4) Index block access costs [49].**

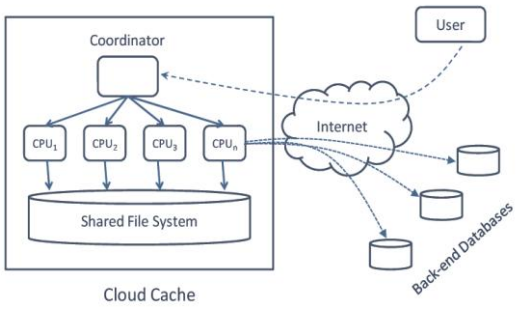

Fig. 1. A cloud cache.

First time cost is high not much more but when more request is do same process then setup and maintains cost is come very lower level by the using caching. So cloud cache service can satisfy infinite demand as long as it is maintained only the demand for a cache service pauses if this service is not available. Moreover, the cloud can schedule the service availability according to the guarantees for the overall revenue estimated by the long term optimization [45]. Nevertheless, it is important that the long-term optimization process is flexible enough to receive corrections while it is still in progress and there have no optimization scheduling or intelligent caches in between public, private & hybrid cloud.

1.2 Related Work

The problem is that, no one proper handling at a time when user demand is increases on cloud services ,like that 1)Ease of Use, 2) Zero maintenance, 3) Automatic scaling, 4) High availability, 5) high level of parallelism, 6) Pay per actual usage vs. pay per instance size, resulting in over subscription, 7) automation and ease-of-use of a Database-as-a-Service, 8) minimal investment and maintenance of in-house hardware, 9) periodically Combining cache interoperability standards, 10) capability of dynamic traffic switching to balance utilization, because this are major problem that's why a price scheme is fluctuated on cloud and The "stateless" and dynamic nature of the cloud poses unique challenges for the "state full" database tier – which is the most sensitive and critical part of the application, and the hardest to scale, all this problem on present cloud , Each development team is free to use whatever local support it likes in this VM—Amazon doesn't care. The creators of one application might choose a Java EE app server and MySQL, for example, while another Group might go with Ruby on Rails. While the service EC2 Provides is quite basic, it's also very general, and so it can be used in many different ways. 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 Go Grid [18], cloud platform providing local support is Force.com, offered by Salesforce.com. Emerging clouds such as the Amazon Simple DB 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.

Microsoft cloud spotlight is provide content sharing, like photo video ,audio etc. but the problem is Content Delivery Networks (CDNs) are critical for enhancing your site's quality, reliability and scalability ,so fast loading and increasing synchronization of content is important , in this condition it will fail [50]. 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],[36][25 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 equilibrium of grid and user satisfaction [25], 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.

Research on the identification of non-correlated indexes using the query structure [39] does not determine the negative and positive correlations with combining intelligent caching. 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 4, it does not satisfy three critical requirements for the pricing scheme: 1) sensitivity to the range of all possible correlations, 2) production of normalized values, and 3) fast computation [45]. 4) Less downtime and power usage, 5) accommodate multiple caching strategies under the same domain. 6) client-side modifications to the cache 5) automatic global web intelligent abstraction.

1.3 Our Proposal

Many Web applications are now hosted in elastic cloud environments where the unit of resource allocation is a virtual machine (VM) instance, A variety of techniques can reduce the latency of communication between VMs co-located on the same server in, say, a private cloud but The cloud caching service can maximize its profit using an optimal pricing scheme. 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, with provide automatic dynamic cache in way of load balancing so distributed data caches is to store fast changing data that is accessed by multiple servers, and solving web service cost and it will give minimum load with dynamic automatic balance and according to situation it will take a self-decision, and giving self-tuning features like

Control Panel – API Built to deliver quick access and real-time solutions, your Cloud Cache CDN, Control Panel gives you everything you need to manage, monitor and distribute your content effortlessly! From control of content behavior to robust reporting options, you get the

Prime tools you need to take charge of your site's technology quickly and easily. **Instant Provisioning** – No waiting! Use ours immediately. **Purge Cache** – Immediately purge your entire cache, or select only a single file. You decide what works best for you. **Reporting** – Get real-time easy-to-read reporting of all relevant metrics when you need them. **Timely Updates** – Make a change and they're done, just like that. **Open API** – full integration with your favourite third-party and custom applications. Privacy **Preserving Mask Matrix** - that allows the cloud to filter out a certain percentage of matched files by using soft computing technique. D**ifferential Query Services** - the queries with higher rank can retrieve higher percentage of matched files.

Intelligent Postmark – Intelligent Postmark is a file-system benchmark that simulates cloud workload in the form of intelligent distributed cache. In each VM we run Post-Mark with dynamic initial files and dynamic transaction, and this all process are done by parallelism automatic tuning then find out Postmark primarily measures better IO performance.

Optimized price demand model – 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. 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 [45].

Price adapting to time changes. Profit maximization is pursued in a finite long-term horizon. The horizon includes sequential, no 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 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. **Modeling Structure correlations**. 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.

Dynamic Cooperative caches modelling –When we are using dynamic cooperative caches in between different model, then top two benefits of cloud computing are found speed and cost. Users can be up and running in minutes instead of weeks or months and this will come from parallelism of dynamic load balance distribution by using elastically scalable grid architecture. And because cloud computing is pay-per-use, operates at high scale and is highly automated, the cost and efficiency of cloud computing is very compelling as well. And in any situation a failure is come, then there no problem because all thing is done by intelligent cache cooperative because we use sharing cache file and this will provide **highly** benefits like **1) On-demand self-service, 2) Broad network access,** 3) **Rapid elasticity,**4)**Measured service** ,5)**Elastic scalability** ,6) **Low upfront costs** ,7)**Economies of scale** ,8)**Operating expense ,** 9)**Simpler to manage** 10)**Greater control of security, compliance and quality of service ,11) Resource pooling.** So this model is removed unpredictable demand patterns problems because intelligent soft computing method is doing scaling up or scaling down of resources for a given application **on demand.**

Neuro - Genetic Price Model - To recap, cloud computing is characterized by real, new capabilities such as self-service, auto-scaling and chargeback, but is also based on many established technologies such as grid computing, virtualization, SOA shared services and large-scale, systems management automation. Apply there some intelligent technique by the using this, for maximization of user profit analysis the previous pattern data from cooperative caches and find out demand price prediction by the using of Artificial neural network with Genetic algorithm and trained the cooperative cache, so we easily getting the information when demand is increase for particular price, then how price is come on minimized and also getting information about Mean Percentage error on between cloud web service cost.

1.4 Contributions

This paper makes the following contributions:

1. 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.

2. An efficient solution to the pricing problem, based on nonlinear programming, adaptable to time changes.

3. A correlation measure for cooperative cache structures that is suitable for the cloud cache pricing scheme and a method for its efficient computation.

4. Soft computing technique is giving parallel and distributing cooperative caches which are done and take self-decision when any user poses a query on cloud for finding information about price prediction for particular demand at a time and model is highly trained as self-tuning from cooperative caches data from current and previous price demand model data.

5. Neuro - Genetic techniques is giving functionality cooperation on different caches scheme, like system create only default cache but highly trained network on cloud is create a different type cache file like 1) Purge cache, 2) default cache, 3) cooperative cache 4) load balance -distribution cache, 5) cooperative optimization cache, 6) external query poses query information cache and simulating on different ways as need of demand self-take a decision and self-destroy this caches file.

6. An experimental study which shows that the highly trained network controller with dynamic pricing demand scheme outperforms any static one by achieving to orders of magnitude more profit per time unit with the future price on level of demand.

2 QUERY EXECUTION MODEL

Cloud databases can offer significant advantages over their traditional counterparts, including increased accessibility, automatic failover and fast automated recovery from failures, automated on-the-go scaling, minimal investment and maintenance of in-house hardware, and potentially better performance. At the same time, cloud databases have their share of potential drawbacks, including security and privacy issues as well as the potential loss of or inability to access critical data in the event of a disaster or bankruptcy of the cloud database service provider this basic problem is giving more hazards for web user which using cloud services so apply the intelligent cooperative cache concept [51] Like OODBMS Cache enables certain tables, rows and columns and session information with network address reference from Database to be cached in the memory of the middle tier servers, access it with the highly trained network cloud and apply self-tuning methodology for the delivering very low latency and high throughput. Data remains synchronized with Database and is accessed through a standard interface first time when using. RDBMS In-Memory Database Cache also supports clustering for elastic scalability and high availability. 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]. Users pose queries to the cloud, which are charged in order to be served. Following the business example of Amazon and Google, Microsoft, MacAfee, Panda etc., we assume that data reside in the same data centre and that users pay on-the-go based on the infrastructure they use, therefore, they pay by the query. 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 maintenance cost [45] and offered security cost on web user pricing scheme adaptation level.

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 [45].

Modified algorithm is: Intelligent Parametric Organization algorithm:
Input: Generate the super plan contains the access the data.
Output: Top down optimizer
Procedure: process steps
1. We are works based on star schema
2. Star schema contains the different dimensions of tables
3. New queries also are joining inside the dimension tables
4. Generate the dynamic load allocation with first iteration to next iteration.
5. We are generate good join planner specification process
6. Join planner works based on cache systems
7. Using the time variation changes new cache systems, we are create under reduced cost building processor
8. It can works on sequential interval amount of time
9. Provides the optimization results
10. Optimization results show the integration.
11. We are increases iteration and integrate the number of iteration process.
12. Gets the results as a minimized cost with feasible and optimal solution
13. Optimal solution focus on discretization (genetic algorithm-AI, NN)
14. It can works on branch and bound algorithm
15. We are gets the target results identification process

Fig. 2. Query execution model for the **Intelligent Parametric Organization algorithm**.

Fig. 2 represent Query execution model for the Intelligent Parametric Organization algorithm represents at a high level the query execution model of the cloud cache. The names of variables and functions are self-explanatory form OODBMS but the cache model is modified by cloud admin. The user query is executed in the cache if all the columns it refers to are already cached. Otherwise it is executed in the back-end databases. The modified result is returned to the user with prediction model demand price 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, previous related demand price, offered price, future price reduction probability offered, offered price correlation),in destination cloud cache is periodically update by the source cloud as like router.

Artificial Neural Network. The back-propagation learning algorithm is one of the most important developments in neural networks. This learning algorithm is applied to multilayer feed-forward networks consisting of processing element with continuous differentiable activation functions. ANN & GA is finding best survival of genes from previous network cloud caches with poses query by the user demand request and intelligent cloud is self-take a decision for heuristic Cloud Cache Data Set then again find, what Mean absolute percentage error (MAPE) will be come on heuristic Cloud Cache and decide which one is best for the user demand. Now we again apply Hybrid model (GA Tuned & ANN) and find out of cloud load traffic information, backend traffic load, number of client request traffic load, client request load distribution by the cloud, measurement of cloud dedicated server on demand creation with control of number of parallelism cache transfer information from network server and A real coded and binary chromosome will be considered for optimization of the weight of ANN.

Training Algorithm of ANN:

The error back-propagation learning algorithm can be outlined in the following algorithm:

Step 0: Initialize weights and learning rate (take some small random values).

Step 1: Perform Steps 2-9 when stopping condition is false.

Step 2: Perform Steps 3-8 for each training pair.

Phase 1: Feed forward Algorithm

Step 3: Each input unit receives input signal X_i and end it to the hidden unit (i =1 to n).

Step 4: Each hidden unit Z_j (j = 1 to p) sums its weighted input signals to calculate net input $Z_{inj} = V_{0j} + \sum_i X_i V_{ij}$

Calculate output of the hidden unit by applying its activation functions over Z_{inj} (Binary or bipolar sigmoidal activation functional):

$$Z_i = f(Z_{inj})$$

And send the output signal from the hidden unit to the input of output layer units.

Step 5: For each output unit y_k (k= 1 to m), calculate the net input:

$$y_{ink} = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

And apply the activation function to compute output signal

$$y_k = f(y_{ink})$$

Phase 2: Back-propagation of error AlgorithmStep 6: Each output unit y_k (k =1 to m) receives a target pattern corresponding to the input training Pattern and computes the error correction term:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

On the basis of the calculated error correction term, update the change in weights and bias:

$$\Delta W_{jk} = \alpha \delta_k Z_j$$
$$\Delta W_{0k} = \alpha \delta_k$$

Also, send δ_k to the hidden layer backwards.

Step 7: Each hidden unit (W_j , j = 1 to p) sums its delta inputs from the output units:

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k W_{jk}$$

The term δ_{inj} gets multiplied with the derivative of $f(Z_{inj})$ to calculate the error term:

$$\delta_k = \delta_{inj} f'(Z_{inj})$$

On the basis of the calculated δ_j , update the change in weights and bias:

$$\Delta v_{ij} = \alpha \delta_j x_i$$
$$\Delta v_{0j} = \alpha \delta_j$$

Weight and bias updation (Phase III)

Step 8: Each output unit (y_k , k=1 to m) updates the bias and weights:

$$w_{jk(new)} = w_{jk(old)} + \Delta w_{jk}$$
$$w_{0k(new)} = w_{0k(old)} + \Delta w_{0k}$$

Each hidden unit (z_j , j=1 to p) updates the bias and weights:

$$v_{jk(new)} = v_{ij(old)} + \Delta v_{ij}$$
$$v_{0k(new)} = w_{0j(old)} + \Delta v_{0j}$$

Step 9: Check for the stopping condition. The stopping condition may be certain number of epochs reached or when the actual output equals the target output [52].

The above algorithm uses the incremental approach for updating of weights, i.e., the weights are being changed immediately after a training pattern is presented, When a BPN is used as a classifier, it is equivalent to the optimal Bayesian discriminate function for asymptotically large sets of statistically independent training patterns.

Genetic Algorithm. An implementation of a genetic algorithm begins with a population of typically random chromosomes One then evaluates these structures and allocates reproductive opportunities In such a way that those chromosomes which represent a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions The goodness of a solution is typically defined with respect to the current population. So this will help on 1)selection, 2) cross over , 3) mutation, 4) reassembly cloud network caches, 5) decomposition: fitness.

3 MODELING OPTIMAL PRICING ON USER DEMAND REQUEST

This section describes the problem formulation of maximizing the cloud profit with intelligent cloud decision with cooperative solicitor network caches information. The presentation of the pricing scheme is guided by propositions that state the main heuristic 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 intelligent cloud admin query plans that use cache structures, i.e., cached columns, views, indexes, previous related demand price, offered price, future price reduction probability offered, offered price correlation. We assume that the set of possible cache structures is **S = {S1, . . . , Sm }.**

Whenever a structure S is built in the cache, it has a onetime building cost B_S. While S is maintained in the cache it has a low maintenance cost which depends on time with network solicitor, M_S (t). Heuristic 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 optimized cached columns. 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

$Cs(t) = B_S + M_S(t \_ t_{built})$. ……. (1)

$Co\_caches = S_{cacheIndex}(t) + N_{cacheIndexdest}$

Cache services are offered through query execution that Uses cache structures, cooperative caches (Co_caches) is combination of caches which is maintained by self OODBMS query in cloud $S_{cacheIndex}(t)$ and network cloud cooperation caches $\mathbf{N_{cacheIndexdest}}$ maintained self-tuned heuristic function on the controls of this with respect to time (t).

Definition 1. The demand for a cache structure S, denoted as λs(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. λs(t)> 0, otherwise not, i.e., λs(t)=0. Intuitively, there is a trade-off between 1) keeping a structure in the cache and paying the maintenance cost, and 2) soft computing model is dynamically maintaining the structure occasionally and 3) maintaining load balancing of caches transfer on network traffic, on user demand request with respect to time (t).

1. Index updating is assumed to incur rebuilding the index from scratch. Data updates on caches from network solicitor are external factors but that can be controlled by the heuristic optimization procedure. In Section 6, we study the effect of updates to the dynamic pricing solution.

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

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

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

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

Where t_0 is the start time of cost observation. Structures can be built and discarded at any time t Є [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.[2] these bounds can be formulated as follows:

$0 \le \lambda_i$, i = 1,……..,m . (5)
$0 \le p_i \le p_{max}$, i = 1,……..,m . (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 solve the optimization problem (4).

2. Mathematically, the integral of (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, ∞ * 0 is undefined.

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 is**

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

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

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

Justification 1. As the cloud cache and its users has inertia, which means that the current system behavior depends on past and influences future behavior. 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.

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.

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

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

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

The parameters **α, β, γ** are constrained to be constants. This means that the price model considers a static relation between demand and price. 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 (Si **i = 1, . . ., m**.) Each element **(v_{ij}, i, j = 1, . . ., m)**corresponds to the correlation of the price of **Sj** to the demand of **Si**. We call **V** the correlation matrix of prices and demands. If (Λ) 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

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

(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 OODBMS 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

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 **δ** intuitively.

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,

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

$$\frac{d\Lambda}{dt}\ge\left.\frac{d\lambda}{dt}\right|_{min}, \quad (11)$$

where **λmin** is the selected minimum value of demand drop rate.

$$u(t)\equiv p_S(t)-c_S(t). \quad (12)$$

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

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

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

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

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.

4. MODELING PRICE-DEMAND CORRELATIONS

The pricing scheme depends on the estimated values of price-demand correlations for all structures, which is stored in the matrix **V** (see the constraint (10)). 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, ($\mathbf{v_{ij}}$, **i, j = i., m**) of **V**, as correlation coefficients, defined as follows:

Definition 2. For any pair of structures **Si** and **Sj** we define the symmetric correlation coefficient ($\mathbf{v_{ij} \equiv vji}$) that represents the combined usage of **Si** and **Sj** in executed query plans.

4.1 Correlation Requirements

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

Proposition 3. The correlation coefficient vij should satisfy the following requirements:

R1. $\mathbf{v_{ij}}$ is negative if $\mathbf{S_i}$ can replace $\mathbf{S_j}$ and the opposite, positive if they collaborate, and zero if they are used independent of each other in query plans.

R2. $\mathbf{v_{ij}}$ can be normalized for any pair of **Si** and **Sj**.

R3. v_{ij} is easy to compute.

Justification 3. R1: The sign of the coefficient $\mathbf{v_{ij}}$ denotes the competitive or collaborative behaviour between a S_i and $\mathbf{S_j}$.

Example 1. In a workload with only one query = select A from T where B = 'b' and C = 'c', the columns B and C should have positive correlation, while the indexes $I_{A\text{-}D}$ = T(A,B,C,D) and $I_{A\text{-}E}$ = 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 requirements may be too strict for other use cases of management of data structures.

R2: The correlation coefficients **V** determine the price of all the structures in the cloud cache (see constraint (10)).

R3: It is necessary to compute all correlation coefficients **V** before the structures are materialized or even selected by the cloud cache.

4.2 Limitations of the Existing Approaches

Recently Schnaitter et al. [33] proposed a technique that computes the correlation between indexes. Given a set of indexes I ⊆ S and two indexes from the set, {Si, Sj}, their correlation coefficient $\mathbf{v^q_{ij}}$ given a query q is

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

Proposition 4. Measure (16) satisfies the requirements R1 - R3.

Justification 4. **R1**: We show that R1 is satisfied by proving its satisfaction for the extreme cases of structure collaboration and competition.

Case 1: If $\mathbf{S_i}$ and $\mathbf{S_j}$ do not coexist in query plans, then let us assume that $\mathbf{S_i}$ is very beneficial to a query q, hence $\mathbf{co_q}$(Xi) →0 and Sj has no effect on it, hence $\mathbf{co_q}$(Xj) → $\mathbf{co_q}$({}). Since the cost function is monotonic [33],$\mathbf{co_q}$(Xij) =$\mathbf{co_q}$(Xi) =$\min_{\{a,b\}}\mathbf{co_q(\{a,b\})}$ → 0. Hence, $\mathbf{v_{ij}} \rightarrow 0$.

Case 2: If $\mathbf{S_i}$ and $\mathbf{S_j}$ collaborate tightly in the extreme case, $\mathbf{co_q}$(Xi) = $\mathbf{co_q}$(Xj) → $\mathbf{co_q}$({}) , $\mathbf{co_q}$(Xij) → 0.Then $\mathbf{v_{ij}} \rightarrow 1$.

Case 3: If the indexes are the same, then $\mathbf{co_q}$(Xi) = $\mathbf{co_q}$(Xj) =$\mathbf{co_q}$(Xij) , implying that $\mathbf{v_{ij}} = -1$.

R2: Since the cases discussed above are extreme, all structure correlation cases fall between them and, therefore their value is bounded by [-1, 1].

R3: We ensure efficient computation of the correlation coefficients by reducing the set of possible query plans. For columns, we propose the following measure:

$$v^q_{ij} = \begin{cases} 1, & \text{if } S_i \neq S_j \text{ and both used in q,} \\ -1, & \text{if } S_i = S_j \text{ and used in q,} \\ 0, & \text{otherwise.} \end{cases} \qquad (17)$$

If two distinct columns appear in the same query, then they collaborate, otherwise they do not. Self-correlation or a column is set to -1, as a column can replace itself. For a pair of index Sj and column Si, we use the following measure:

$$v^q_{ij} = \begin{cases} 1 & \text{if } S_j \notin S_i \quad \text{and both can be used in q,} \\ -1, & \text{if } S_j \in S_i \quad \text{and both can be used in q,} \\ 0, & \text{otherwise.} \end{cases} \qquad (18)$$

We extend the correlation computation for a Workload. If $\mathbf{v^q_{ij}}$ is the correlation of Si and Sj for query q, then the coefficient for an entire workload is

$$v_{ij} = \frac{\sum v^q_{ij} co_q(\{\})}{\sum co_q(\{\})}. \qquad (19)$$

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 $\mathbf{O(|I|^2)}$ 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 speed up the correlation computation using the observation that, even though the total number of index combinations is $\mathbf{O(|I|^2)}$ the set of possible plans is typically much smaller. 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].

5 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, $\mathbf{p_i s}$, 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 $\boldsymbol{\delta_i s}$. 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.

5.1 Designing the Optimization Solution

The objective function of the problem is the maximization of an integral, i.e., $\mathbf{max}\int^T_0\mathbf{(r(t) - w.v(t))dt.}$ 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 $\boldsymbol{\lambda_s}$**(t) . ps(t)** in (2) and **ps(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 λ_{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 mixed-integer nonlinear programming (MINLP) [11], which present the advantage of being implementable online.
We propose the division of the prediction horizon **[0, T]** into time intervals: let us assume that there are time points $t_j \varepsilon [0,T]$, **j = 0, . . . ,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 $t_j \varepsilon [0,T]$, **j=1,..k**. and which price to assign to each built structure.

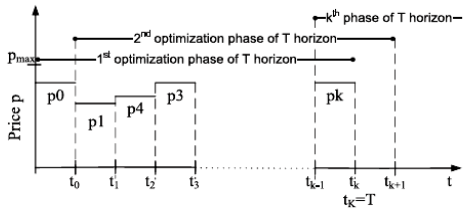

Fig. 3. The optimization procedure is divided into short time intervals and iterates on a sliding time window.

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

Fig. 3 depicts the proposed repeated optimization over a sliding time prediction horizon of length T. For simplicity, we consider equal time intervals, $t_{j+1} - t_j = t_{j+2} - t_{j+1}$, 0,……,k-2. The optimization is performed repeatedly for k prediction horizons beginning at t_{start} and ending at tend, such that: [t_{start} , t_{end}], t_{start} = 0, t1 . . . T and t_{end} = T,T + t1, 2T, respectively.

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

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.

5.2 Estimating the Parameters Structure

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 * 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

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

Where F is a **m * m** matrix of functions on time and elements of the parameter matrices A, B, Γ. If the **m** constraints in (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 (10) are independent differential equations.

Justification 5. 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, assume two structures S1 and S2. 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 S1 and S2 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 S1 and S2 are always employed together in the cloud; therefore, they can be represented as a set of structures with a single price [22].

5.3 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).

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 i.e. $\lambda_s(t) = \lambda_2$,t ε [0 , T_{small}] corresponding to price p1 and $\lambda_s(t) = \lambda_2$,t ε [T_{small} , T_{big}) corresponding to price p2 (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 $r_{small} = \lambda 1 * p1 - B_S - M_S$ (**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 ≥ T_{small}, resulting in profit $r_{big\text{-}small} \leq \lambda 2 * p2 - B_S - M_S$(T big _ T small). For the long-term optimization the profit is: $r_{big} = \lambda 1 * p1 + \lambda 2 * p2 - B_S - M_S$(T big). Obviously, $r_{big} > r_{small} + r_{big\text{-}small}$. Therefore, the result of the two-phase short-term optimization procedure is not as optimal as that of the one-phase long-term procedure.

5.4 Discussion on the Model Simplicity

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. 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 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 are collected and this data can clearly be used to re-identify the **price/demand** model periodically.

6 EXPERIMENTAL EVALUATIONS

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

6.1 Experimental Setup and Methodology

The cloud cache is set up with one back-end database. The cache is operated under a TPC-H-based workload, which consists of seven 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 three categories, low, medium, and high. Queries arrive at 10 second intervals. We copy the setup in [24], the distribution of the query templates in one phase consisting of 10,000 queries. We select this

workload, as it is portable across different OODBMS, 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. As an indication, while varying the price from the building cost (cost) to $\mathbf{p_{max}}$ = 10 * cost, the demand varies from 0 up to 8,000 queries, with many values around 4,000. 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 1) to avoid high cloud profit by solely exploiting high demand values $\boldsymbol{\lambda_i S}$ and 2) force the pricing scheme to fluctuate $\boldsymbol{\lambda_i S}$ in order to maximize the profit. The price variable for each structure ranges from 0 to 100 % of the respective building cost, i.e. $0 \leq \mathbf{p_i} \leq \mathbf{B_{Si}} * 100$. The experiments measure 1) the average cloud profit per time point, 2) the average user loss per time point, and 3) the execution time. Cloud profit is defined in (2) and user loss is the user satisfaction as defined in (12). The dynamic pricing scheme is compared with a static pricing scheme that fixes the cloud profit to a specific percentage of the building cost.

6.2. Experimental Results View

This section summarizes the experimental results.

6.2.1Pricing with Dynamic Structure Availability

Assuming that all structures are constantly available (i.e., fixed caching but changeable with permission of cloud), and, therefore built once in the cache at the beginning of pricing and maintained ever since, i.e., $\delta i = 1$, $i = 1,\ldots, m$ always. As the optimization horizon is extended the profit drops because structures are maintained in the cache even though their demand drops; 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 (14) for big values of weight, i.e., w = 30, 40, resulting in zero profit and user loss. 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. The results of this experiment are in accordance with the results of the works in [37].

6.2.2 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 rebuilt. Contrary to pricing with fixed because optimization procedure takes advantage of long-term predictions in order to schedule the structure availability in a more optimal way.

6.2.3 Sensitivity of the Optimization Schedule

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.

6.2.4 Performance Comparison with Analysis

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. 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. 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].

6.2.5 Correlation Binding of Structures

This section presents the index correlations achieved using (16) and compares the proposed measure for correlation coefficients (19) with the state-of-the-art measure (15) [33]. Furthermore, it is also bounded by the range [-1, 1].

6.2.6 Predicting the Price Demand for Structures

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.

6.2.7 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. Specifically, an update of structure S at time t incurs a reset of the respective δ parameter from 1 to 0 at that time. 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 CONCLUSIONS

This work proposes a novel pricing demand scheme designed for a cloud cache that offers querying services and aims at the maximization of the cloud profit with predictive demand price solution on economic way of user profit. The proposed solution allows: on one hand, long-term profit maximization with price minimization on request of same demand, and, on the other, dynamic calibration to the actual behaviour of the cloud application, while the optimization process is in progress.

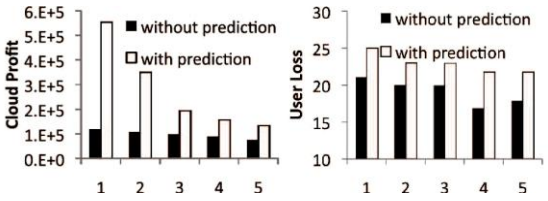

Fig. 4. Optimization using or not predictions for updates for 1-5 updates on average per structure.

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] G.R. Bitran and R. Caldentey, "An Overview of Pricing Models for Revenue Management," Manufacturing and Service Operations Management, vol. 5, no. 3, pp. 203-209, 2003.

[2] N. Bruno and S. Chaudhuri, "An Online Approach to Physical Design Tuning," Proc. Int'l Conf. Data Eng. (ICDE '07), 2007.
[3] X.-R. Cao, H.-X. Shen, R. Milito, and P. Wirth, "Internet Pricing with a Game Theoretical Approach: Concepts and Examples," IEEE/ACM Trans. Networking, vol. 10, no. 2, pp. 208-216, Apr. 2002.
[4] C. Chen, M. Maheswaran, and M. Toulouse, "Supporting Co-Allocation In an Auctioning-Based Resource Allocator for Grid Systems," Proc. 16th Int'l Parallel and Distributed Processing Symp. (IPDPS '02), 2002.
[5]S. Choenni, H.M. Blanken, and T. Chang, "On the Selection of Secondary Indices In Relational Databases," Data and Knowledge Eng., vol. 11, no. 3, pp. 207-233, 1993.
[6] D. Dash, Y. Alagiannis, C. Maier, and A. Ailamaki, "Caching All Plans with One Call to the Optimizer," Proc. Self-Managing Database Systems (SMDB), 2010.
[7] D. Dash, V. Kantere, and A. Ailamaki, "An Economic Model for Self-Tuned Cloud Caching," Proc. IEEE Int'l Conf. Data Eng. (ICDE '09), 2009.
[8] C. Ernemann, V. Hamscher, and R. Yahyapour, "Economic Scheduling In Grid Computing," Proc. Eighth Int'l Workshop Job Scheduling Strategies for Parallel Processing (JSSPP '02), 2002.
[9] G. Gallego and G. van Ryzin, "Optimal Dynamic Pricing of Inventories with Stochastic Demand over Finite Horizons," Management Science, vol. 40, no. 8, pp. 999-1020, 1994.
[10] A. Ghose, V. Choudhary, T. Mukhopadhyay, and U. Rajan,"Dynamic Pricing: A Strategic Advantage for Electronic Retailers," Proc. Conf. Information Systems and Technology (CIST), 2003.
[11] I.E. 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, vol. 39, pp. 1283-1294, 2003.
[13] L. He and J. Walrand, "Pricing Differentiated Internet Services," Proc. IEEE INFOCOM, pp. 195-204, 2005.
[14] http://aws.amazon.com/, 2011.
[15] http://code.google.com/appengine/, 2011.
[16] http://tomopt.com/tomlab/, 2011.
[17] http:/www.cern.ch/, 2011.
[18] http://www.gogrid.com/, 2011.
[19] http://www.microsoft.com/azure/, 2011.
[20] http://www.sdss.org/, 2011.
[21]M. Kradolfer and D. Tombros, "Market-Based Workflow Management," Int'l J. Cooperative Information Systems, vol. 7, pp. 297-314, 1998.
[22] J. Li and R. Yahyapour, "Negotiation Model Supporting Co-Allocation for Grid Scheduling," Proc. IEEE/ACM Seventh Int'l Conf. Grid Computing, 2006.
[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, vol. 3, no. 3, pp. 221-242, 2005.
[24] T. Malik, X. Wang, R. Burns, D. Dash, and A. Ailamaki, "Automated Physical Design In Database Caches," Proc. Workshop Self-Managing Database Systems (SMDB), 2008.
[25] T. Malik, R.C. Burns, and A. Chaudhary, "A Financial Option Based Grid Resources Pricing Model: Towards an Equilibrium between Service Quality for User and Profitability for Service Providers," Proc. Advances in Grid and Pervasive Computing, pp. 13- 24, 2009.
[26] V. Marbukh and K. Mills, "Demand Pricing and Resource Allocation In Market-Based Compute Grids: A Model and Initial Results," Proc. Int'l Conf. Networking (ICN), pp. 752-757, 2008.
[27] Y. Masuda and S. Whang, "Dynamic Pricing for Network Service: Equilibrium and Stability," Management Science, vol. 45, no. 6, pp. 857-869, 1999.
[28] M. Morari and J.H. Lee, "Model Predictive Control: Past, Present and Future," Computers and Chemical Eng., vol. 23, no. 4/5, pp. 667- 682, 1999.
[29] R.A. Moreno, "A.B.: Job Scheduling and Resource Management Techniques In Economic Grid Environments," Proc. Across Grids 2003, pp. 25-32, 2004.
[30]Y. Narahari, C.V.L. Raju, K. Ravikumar, and S. Shah, "Dynamic Pricing Models for Electronic Business," Dynamic Pricing Models for Electronic Business, vol. 30, pp. 231-256, 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, Dec. 2008.
[32] S. Papadomanolakis, D. Dash, and A. Ailamaki, "Efficient Use of the Query Optimizer for Automated Database Design," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB '07), pp. 1093-1104, 2007.
[33] K. Schnaitter, N. Polyzotis, and L. Getoor, "Modeling Index Interactions," Proc. VLDB Endowment, vol. 2, no. 1, pp. 1234- 1245, 2009.
[34] B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki, "Dynamic Optimization of Batch Processes: Ii. Role of Measurements In Handling Uncertainty," Computers and Chemical Eng., vol. 27, pp. 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," Int'l J. Very Large Data Bases, vol. 5, no. 1, pp. 48-63, 1996.
[36]A.Sulistio, K. Kyong Hoon, and R. Buyya, "Using Revenue Management to Determine Pricing of Reservations," Proc. IEEE Int'l Conf. e-Science and Grid Computing, pp. 396-405, 2007.
[37] X. Wang, T. Malik, R.C. Burns, S. Papadomanolakis, and A. Ailamaki, "A Workload-Driven Unit of Cache Replacement for Mid-Tier Database Caching," Proc. 12th Int'l Conf. Database Systems for Advanced Applications (DASFAA '07), pp. 374-385, 2007.
[38] M.P. Wellman, W.E. Walsh, P.R. Wurman, and J.K. Mackie-mason, "Auction Protocols for Decentralized Scheduling," Games and Economic Behavior, vol. 35, pp. 271-303, 2001.
[39] K.-Y. Whang, G. Wiederhold, and D. Sagalowicz, "Separability: An Approach to Physical Database Design," IEEE Trans. Computers, vol. C-33, no. 3, pp. 209-222, Mar. 1984.
[40] P.-S. You and T.C. Chen, "Dynamic Pricing of Seasonal Goods with Spot and Forward Purchase Demands," Computer and Math. Applications, vol. 54, no. 4, pp. 490-498, 2007.
[41] www.vmware.com
[42] http://en.wikipedia.org/wiki/Google%2B
[43]http://www.mcafee.com/us/products/security-as-a- service/index.aspx
[44]http://www.oracle.com/technetwork/oem/cloud-mgmt-496758.html
[45]Verena Kantere, Debabrata Dash, Gre´ gory Franc¸ois, Sofia Kyriakopoulou, and Anastasia Ailamaki, "Optimal Service Pricing for a Cloud Cache", IEEE Transaction On Knowledge & Data Engineering, VOL. 23, NO. 9, SEPTEMBER 2011
[47]http://searchcloudcomputing.techtarget.com
[48]http://storagedecisions.techtarget.com/seminars/cloud_storage.html
[49]http://www.dba-oracle.com/art_builder_cpu_io.htm
[50http://windows.microsoft.com/en-IN/windows/explore/cloud
[51]http://www.webopedia.com/TERM/C/cloud_database.html
[52]S.N.Sivnandan, S.N.Deepa, " Introduction To GeneticAlgorithm", ISBN 978-3-540-73189-4 Springer Berlin Heidelberg New York.