A Framework for Cost-Aware Cloud Data Management

Verena Kantere

Cyprus University of Technology
verena.kantere@cut.ac.cy

Abstract. The emerging world of offering information services through the cloud necessitates the coalescence of existing research and business technologies into the provision of all-inclusive solutions for data management. This paper proposes a framework that can support cost-aware data management in the cloud. Users and cloud providers can use the framework to receive and provide services that comply with agreements on data service cost and requirements and allow for profit while being efficient in terms of performance. The proposed framework includes modules that incorporate the notion of monetary cost in current data management, but also modules that take optimization decisions for future data management taking into account both monetary cost and performance. The framework dictates the design of a middleware application that can be plugged on top of a cloud data management system. Such a middleware receives the user's workload and preferences for cost and query performance and controls data management so that the user is satisfied and the cloud provider is viable and, furthermore, profitable. An initial realization of part of the framework as a middleware application has already been constructed, tested and published with promising results.

Keywords: Cloud data services, cloud data management, cost-aware data management, cloud economics.

1 Introduction

The new trend for service infrastructures in the IT domain is called cloud computing, a style of computing that allows Internet users of any expertise to access information services on the web. Information, as well as software is permanently stored in Internet servers and probably cached temporarily on the user side.

Outsourcing the archiving and manipulation of large persistent datasets, such as scientific data of large scale, small-and-medium enterprise data as well as personal datasets gains more and more credence. The general requirement of users is to manage efficiently the data with a dynamic demand for additional or reduced computational support, while investing as little as possible time and money. Unfortunately, current commercial data management tools are insufficient to meet these requirements especially while they are not tailored for specific data and users. Also, they are usually incapable to support the unprecedented scale, rate, and complexity of big data collection and processing.

The cloud computing paradigm seems the right choice for designing infrastructures and moreover applications that can meet the above data management expectations.

R. Meersman et al. (Eds.): OTM 2012, Part I, LNCS 7565, pp. 146–163, 2012.

The cloud alleviates the burden of data management from the users by offering transparent data services for remuneration. Emerging IT business in cloud computing [1, 2, 3] is an effort to offer such services. Such a cloud provider necessitates various technological capabilities. Most importantly, it is required that cloud data services run with minimal capital expenditure, but, nonetheless, can support efficiently multi-user tenancy.

A cloud data service provider offers its services on cloud databases. Such databases support the archiving of user data in the cloud employing caching techniques. Data that reside in the cloud, i.e. cloud data, can benefit from the offered cloud data services. Users are customers of the cloud that consume its resources as a utility service. Specifically, the users can query the cloud data, paying the price for what they use. User payment is employed for coverage of short and long-term costs. Short-term cost refers to the respective query execution, and long-term cost to the self-preservation of the cloud infrastructure and improvement of the cloud services.

We propose a generic framework that aims to coalesce the existing research and business technologies into the provision of an all-inclusive solution for the management of data in the cloud. The framework is designed to provide transparent cost-aware data management on top of a cloud database, whatever the latter is: a traditional DBMS deployed on a cloud infrastructure or a new prototype of a cloud DBMS. The purpose of the proposed framework is to provide an economy solution for a cloud data management system that is inherently bound to the technical solutions. Currently, our focus is on scientific data, since their variety, volume and extreme data management requirements makes them the best candidates for research and experimentation. The framework, however, is generic and applicable to any kind of data that may need cloud management.

The framework aims to support an economy for a cloud infrastructure that handles structured data, i.e. data that are stored in databases. Following the business line in cloud computing, the proposed economy employs a cost model that takes into account all the available resources in a cloud, such as disk space and I/O operations, CPU time and network bandwidth. The economy is self-tuned to the policies that aim to: (i) high quality of individual query services, (ii) increasing overall quality of query services, and (iii) cloud profit. The experience of data management is accumulated and quantified by a regret scheme in order to assist the improvement of cloud services by building new data structures, (i.e. cached data and indexes). The cost of new data structures is amortized to prospective users based on a model that predicts future service consumption. Finally, cloud profit is ensured and maximized using an appropriate pricing scheme. The framework may include an optional module that comprises a query planner and query router that enable the servicing of sets or sequences of queries. The cloud economy can be extended in order to handle such query combinations. In this way, the cloud framework can serve in an optimal manner even demanding tasks of data analysis (which are very common in scientific data management). Finally, associative modules that take care of risk management and estimate data service correlations are necessary in order to use the framework in real cloud environments.

An initial application of a simplified form of the framework is already designed, developed, tested and published in [4, 5, 6]. This experience has proved the applicability and effectiveness of the framework even in a simplified form. Our goal is to extend our existing work with the general and elaborated form of the framework presented in this paper.

2 Related Work

Current research on cloud computing considers an infrastructure that comprises a set of independent edge caches that cooperate in order to deliver web content. Content sharing among caches [7, 8, 9] involves (i) content retrieval from sibling caches instead from the server, (ii) routing and sharing of content updates, and (iii) sharing cache resources, in order to achieve efficient collaborative data placement/replacement/update/lookups etc. Being more compliant with the business domain, this proposal focuses on self-tuned cloud caches that share resources, rather that self-organization of independent caches. Concerning the management of scientific data, existing research solutions [10], consider network bandwidth to be the only important resource, and, therefore, the sole basis for cost computation. However, cloud businesses usually prorate cost to more types of resources. For instance, GoGrid [8] gives network bandwidth for free. A self-tuned cache [11] reduces query execution costs adaptively to the workload. As a step further towards commercial applications, we propose an economy that takes into consideration the overall cost of the infrastructure beyond network bandwidth: disk I/O, storage and CPU.

Cloud computing is the natural dilation of grid computing [12], as it enables an integrated collaborative use of high-end computing owned and managed by multiple organizations. Grid databases [13] are federated database servers over a grid, which are viewed as a virtual database system through a service federation middleware [14]. Querying the grid data guaranteeing high performance is one of the key issues for distributed queries across large datasets. This issue is inherited in cloud computing, and becomes more complicated, since it is augmented with the issue of providing an accounting service that supports a payment scheme for cloud usage. We take this challenge and we propose a framework of data-aware cloud economy that fills the essential gap of tuning the provision of data management services for remuneration.

Related to the proposed framework for cloud economy is the research on auctioning and accounting systems. Accounting in wide-area networks that offer distributed services have long been the target of research in computing. Mariposa [15] discusses an economic model for querying and storage in a distributed database system. In Mariposa clients and servers have an account in a network bank and users allocate a budget to each of their queries. The processing mechanism aims to service the query in the allotted budget by executing portions of it on various sites. The latter place bids for the execution of query parts, and the bids are accumulated in query brokers. The decision of selecting the most appropriate bids is delegated to the user. In contrast, our cloud proposal can recommend to the user an efficient but also profitable for the cloud query plan. In the spirit of Mariposa, a series of other works have proposed solutions for similar frameworks [16, 17, 18, 19, 20]. These focus on job scheduling and bid negotiation, which are orthogonal issues to those that are tackled by the proposed framework.

Concerning the special case of scientific data, their management has been a challenge until now. The need for in-depth analysis on huge amounts of data has increased the demand for additional computational support. Unfortunately, current commercial data management tools are incapable of supporting the unprecedented scale, rate, and complexity of scientific data collection and processing. Independently of the categories that scientific data belong to, their management is essentially divided into the

following coarse phases: workflow deployment, management of metadata, data integration, data archiving and finally data processing [21]. All these phases suffer from tremendous data management problems that concern automation, online processing, data and process integration and file management [22].

The proposed framework enables the leverage of the management of scientific data by a cloud provider, achieving transparency of data management solutions that are efficient and cheap. Currently, scientists need to tightly collaborate with computer engineers in order to develop custom solutions that efficiently support data storage and analysis for each different experiment [23, 24]. Beyond the fact that constant collaboration of multidisciplinary scientists and engineers is hard, time and effort consuming, the experience gained by such collaboration is not inherited widely to the scientific community, so that next generation experimental setups can benefit from it. It is absolutely necessary to develop generic solutions for storage and analysis of scientific data that can easily be extended and customized. Developing generic solutions is feasible, since there are low-level commonalities in the way that experimental data are represented or analyzed [21, 24]. Therefore, frequently, scientific data processing encompasses generic procedures. Current research, however, has not proposed any solution for the support of such processes.

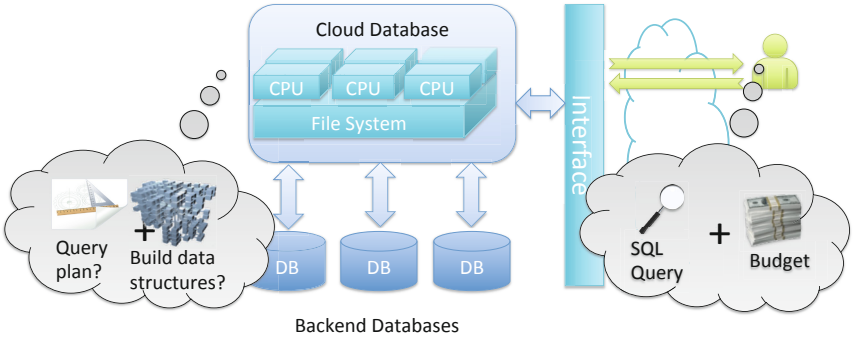

Fig. 1. Operation of a cloud data service provider

3 Operation of a Cloud Data Service Provider

In a cloud environment, there are the users and probably owners of data on one hand and the cloud data service provider on the other. In the general case, the data reside permanently in backend databases. The cloud database supports caching of these data in order to offer efficient query processing. Users pose queries to the cloud, which are charged in order to be served. The goal of the cloud is to provide efficient query processing at low price, while being profitable. Query performance is measured in terms of execution time. Naturally, query execution is accelerated with the number of available data structures, i.e. indexes and cached data. Rationally, the faster the execution, the more expensive the service is. The user defines her preferences concerning the service of her query by indicating the budget she is willing to spend on the query, according to the respective execution time that the cloud can provide.

The cloud receives the query and the budget/execution-time preferences of the user and produces respective alternative query plans. The cost of these query plans is estimated and juxtaposed to the preferences of the user. If the cheapest plan is to execute the query on the backend database (i.e. the permanent storage of the data), the cloud outsources query execution. Otherwise, the query is executed on the cloud database, either on data that is already cached or by caching data from their permanent storage. If there are query plans that the user can afford, according to her budget, then the cloud picks up the most appropriate one of them, w.r.t. the supported policies. If the cost of alternative query plans is over the user budget, the user is presented with the alternative options and can choose which one she is willing to pay for, if there is such one. The profit from each query service is credited to the cloud account and is employed in order to build data structures that can improve the query services. The cost of the new structures is amortized to prospective users that will consume services that employ these structures. Fig. 1 shows graphically the operation of the cloud data service provider.

3.1 Cloud Data Services

The cloud offers services on the data that the user can access. These services are summarized as follows:

1. **Data storage design**: The user may request from the cloud to implement either a user pre-defined or a cloud design for the storage of her data in the cloud database.
2. **Query execution**: The user may request the execution of a query or the execution of a whole workload. The cloud has to create alternative query plans and select one for execution. Furthermore, the cloud has to dynamically parallelize the execution on the cloud infrastructure.
3. **Query optimization**: The cloud has to optimize query execution by creating and maintaining dynamically and automatically (i.e. without the intervention of the user) data structures, such as indexes, cached data columns and data views.

All the above services incur an infrastructure and administration cost that, through their provision, is alleviated from the user and burdens the cloud data service provider. The proposed framework aims at handling this economic burden in the best possible way, such that the provider remains viable and furthermore profitable.

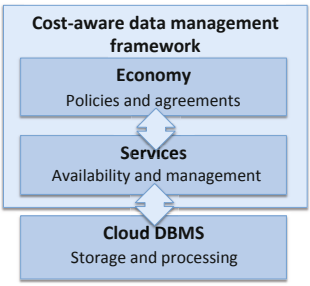

Fig. 2. Collaboration of the framework and the cloud DBMS

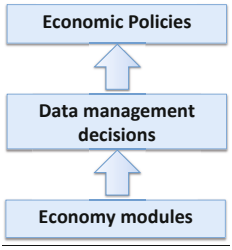

Fig. 3. Internal operation flow in the proposed framework

3.2 Cloud Economic Policies

As mentioned, a query plan is selected according to supported policies. Our framework supports the following policies, which can lead to a viable and even more a profitable operation of the cloud data service provider.

1. **Individual user satisfaction**: The service provider should keep each user satisfied with the data services she receives. User satisfaction is achieved if the following requirements are satisfied:

   - Respect user preferences for query execution: The quality of the data services that the user receives should comply with her preferences for query execution. For example, a user that requests a fast or cheap query execution, should receive query execution that is fast or cheap, respectively.
   - Avoid over-charging data services: Each service should be priced close to its actual cost, i.e. the cloud intention is to provide cost-competitive services.
   - Respect user budget constraints: The user should receive services that are always priced in her declared budget.

2. **Data service improvement**: The cloud provider should be able to offer improved services as time goes by. Service improvement can be sought along two directions:

   - Offer broader and appropriate services: The cloud has to increase the variety of offered services with time. This means that it should build more data structures, which can be used in offered query execution plans. Moreover, the cloud has to change or discontinue services that are no longer useful or popular.
   - Offer services at a lower price: Ideally, as time goes by, the cloud should be able to offer the same services, e.g. the same data structures, at a lower price.

3. **Cloud profitability**: A commercial cloud provider is a business and as such it aims at making profit. The latter can be guaranteed by the following:
   - Set optimal price: Services should be priced above their actual cost in a way that the overall cloud profit is maximized.
   - Schedule optimal availability: The cloud has the option of discontinuing services that are not popular and creating services that are. This means that data structures that are not used often in user query execution should be evicted from the cloud DBMS and data structures that are used often should be constructed and maintained. Fig. 2 and Fig. 3 show conceptually how the framework sits on top of a cloud DBMS and what is the overall internal operation the framework.

3.3 Cloud Layering Architecture

As it is widely known, the cloud computing paradigm dictates the offer of services on three levels, namely the infrastructure (IaaS), the platform (PaaS) and the software (SaaS). In case of cloud data service provision, the infrastructure is the cloud cluster, which includes CPUs, disks, memory and I/O and network communication; the platform is the cloud DBMS deployed on the infrastructure and the software is a specific database application. A cloud data service provider may offer all three levels as an

all-inclusive service, i.e. such a provider may own the infrastructure and manage both the cloud DBMS and the database applications on top of it. Alternatively, the cloud data service provider may not own the infrastructure, but rent the latter from an IaaS provider, and manage only the cloud DBMS and the database applications. Even more, the cloud data service provider may rent the cloud database from a PaaS provider. In any of the three cases, the offered data services incur a cost that is either the direct cost of operating the cloud infrastructure, or the cost of renting it through mediators. Fig. 4 shows the 3 alternative architectures. Our framework takes the infrastructure cost as an input, no matter if this originates from operating or renting the infrastructure. Therefore, the framework is applicable to any of the three layering architectures.

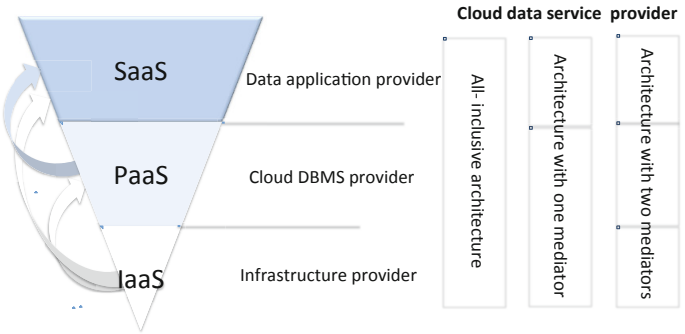

Fig. 4. Overall architecture and mediation of the cloud data service provider

4 Cost-Aware Data Management

The cloud data service provider needs to perform traditional data management, i.e. data management that aims at performance, while taking into consideration cost restrictions. The latter are input to the provider, which has to decide what to output as an offered service. The input can be broken down to four categories:

1. **Data requests:** This refers to what the user asks for. Usually, it is query execution, but it can be a workload execution or other data tasks, such as data replication.
2. **Data updates:** This refers to the rate and extent of data update. When data is updated in the backend databases (in case it is permanently stored out of the cloud), the updated data has to be reloaded into the cloud database. Moreover, data update incurs an update of all relevant data structures.
3. **Infrastructure cost:** This refers to the cost of operating or renting the infrastructure, namely CPU, disk, I/O operations and network.
4. **User budgets:** This refers to how much the users are willing to spend on monetary compensation for the data services they receive.

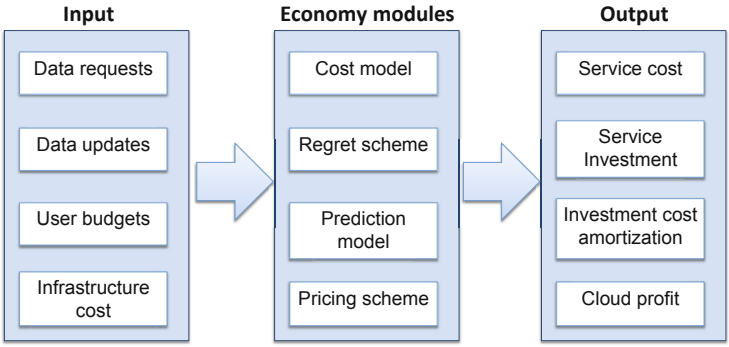

Fig. 5. Operation of the proposed framework

Considering the above input, the cloud has to decide what are the services it should offer, which can be broken down to two decisions: (i) how to service each individual request and (ii) what should be the available services. In order to take these decisions, the cloud has to answer to the following questions:

1. **What is the cost of a possible service?**
   Assuming that there is a pool of possible services to be offered alternatively for the fulfillment of a data request, the cloud needs to decide which one complies best with all the cost restrictions.
2. **In which services should money from the cloud account be invested?**
   The cloud needs to create pools of services that can serve alternatively or complementary data requests. For example two query plans may each include a different index, and these indexes can be used alternatively or in combination to serve the same query. To create these services the provider needs to invest cloud money, which will cover the cost of operating the infrastructure.
3. **What is the depreciation of a service since it becomes available?**
   In order to keep the cloud data service provider viable, the cloud money invested in the creation of possible services needs to be recuperated from user payments. The cloud needs to know how much and for how long it can recuperate the cost.
4. **How can the cloud make profit?**
   The cloud data service provider is usually a business, and, as such, it aims not only to be economically viable, but also profitable. Thus, it is essential for the provider to know if and how much profit it can make from the offered services.

To answer the above questions our framework comprises four basic generic modules: (i) a service cost model, (ii) a data management regret scheme, (iii) a prediction model for service consumption and (iv) a service pricing scheme. These are presented in the following. The operation of the proposed framework is shown in Fig. 5.

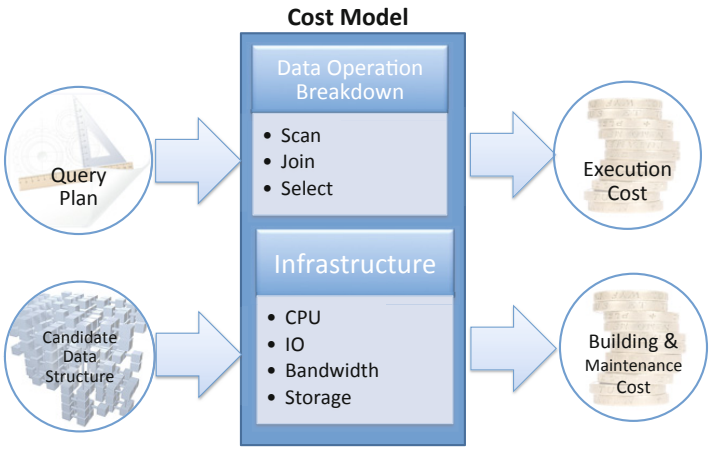

Fig. 6. Operation of the cost model

4.1 Service Cost Model

A request for query execution received by the provider is distributed to the appropriate CPU node(s), or to the backend databases (the actual distribution and execution algorithm depends on the cloud DBMS and is a black box to framework). It is essential for the economy of the cloud provider to estimate correctly the utility cost of the cloud infrastructure. Based on such estimations, the cloud can decide the compensation requested from the users for the offered query services. The cost model takes as input either (i) a query execution plan or (ii) a data structure that is to be built from scratch or to be updated. For the query plan, the cost model estimates what is the execution cost and for a data structure it estimates what is (a) the cost for building and (b) the cost of maintaining the structure in the cloud DBMS. In order to estimate the total cost, the model takes into account (i) what are the necessary primitive data operations for each input (e.g. scan, join, select, project) and (ii) how much of the infrastructure will be used (CPU, I/O operations, network bandwidth and disk space). The operation of the cost model is shown in Fig. 6.

It is safe to assume that the cloud provides unlimited amount of storage space, CPU nodes, and very high-speed intra-cloud networking. Also, the CPU nodes are usually all identical to each other. Compared to TCP bandwidth on Internet, the intra-cloud bandwidth is orders of magnitude faster and we can ignore the overhead associated with it. We assume that the storage system is based on a clustered file system, where the disk blocks are replicated and stored close to the CPU nodes accessing them. Also, we assume that the virtual disk is a shared resource for all the CPU nodes in the cloud DBMS (share-all architecture).

Since the cloud DBMS considers many alternative possible query plans, i.e. design configurations for executing a query, accurate and fast estimation of the cost associated with each plan is very important. The execution time of a query is estimated based on a respective query plan. Currently, in our work we have only considered

plans that run thoroughly in the backend or in the cloud DBMS. Concerning queries that run in the cloud DBMS completely, the estimation of the execution cost is determined based on a plan that is suitable for the cloud. Therefore, the implementation of the cost model module needs to be aware and take into account statistics on query execution on the specific cloud DBMS. Such statistics can be requested by the cost model from the cloud DBMS based on query templates. Then, specific queries can be matched with the appropriate template and their execution cost can be estimated based on the statistics for the template.

As mentioned, the cost model is used to estimate the cost of building or maintaining (i.e. updating) a data structure, too. The latter can be cached data columns from the permanent data storage, views and indexes on cached columns. For the estimation of the building and maintenance cost of a data structure, the cost model has to use statistics on creating similar structures, too. These statistics can be collected by the cloud DBMS optimizer during a training period, or interleaved training periods.

4.2 Data Management Regret Scheme

The data management regret scheme accumulates and quantifies the experience of the cloud provider in taking data management decisions in order to help the provider to take better, in terms of cost and performance, decisions in the future. In effect, this means that the cloud aims to execute in the future similar data requests to current ones in a cheaper and faster manner. This can be achieved if the cloud builds and maintains appropriate data structures and utilizes the right parts of the infrastructure (i.e. how many CPUs and how much memory to use for servicing the data requests, and how much disk to use for data replication).

Essentially, the absence of a data structure in the cloud DBMS prohibits the provider from offering services, i.e. query execution plans, which employ this structure. Also, having to rent (or switch on, depending on the cloud provider architecture and mediation) more infrastructure means a 'cold start' for caches and a delay in CPU and disk utilization. The goal of the regret scheme is to monitor how many times the provider has not been able to offer a query plan (or a data operation in general) that would satisfy the user preferences in the best possible way due to the unavailability of a data structure or part of the infrastructure. To achieve this, query plans have to be broken down into basic plans and compared to each other with respect to the data structures they comprise and the infrastructure (especially the parallelization degree, i.e. number of CPUs used). This comparison leads to the formulation of a 'regret' for the unavailability of data structures and infrastructure, which has to be quantified and accumulated, so that it can be an indication that the investment in building and offering this data structure is a good or bad data management decision. The regret scheme can also include the decision making part for creating data structures or adding infrastructure, based on the accumulated regret. This decision making has to be based on the policies presented in Section 3.2, with a customizable prioritization. The operation of the regret scheme is shown in Fig. 7.

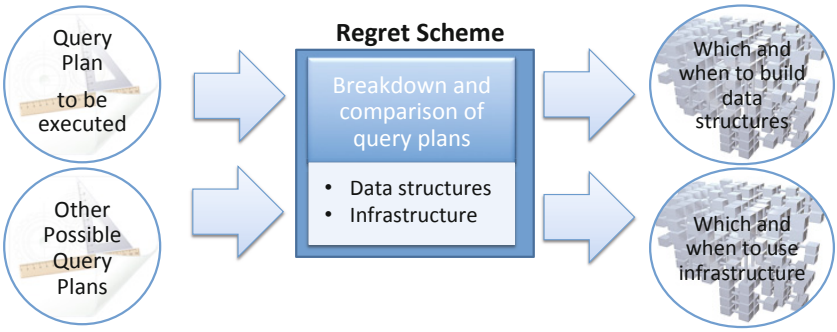

Fig. 7. Operation of the regret scheme

4.3 Prediction Model

The regret leads to the improvement of data services (see Fig. 8). Yet, the cost of investment in new data structures is paid from the credit in the cloud account. The intention of the provider is to remain economically viable, by amortizing this cost to prospective users that receive data services that use the new inventory. The goal of amortization is to reduce the individual cost of these services. Cost reduction increases the potential that the offered services are cheaper and well-within the user's budget. In such a case, the services become competitive and have a high chance of being offered instead of alternative services (for example the faster and the cheaper a query plan is, the most probable it is to be selected for execution, among alternative query plans). The provider benefits from the difference of actual cost and offered user compensation; therefore, the provider increases its credit, which gives the opportunity for more investments directed by regret, leading to even more quality data services.

The framework comprises a prediction model for the amortization of the building and maintenance cost of new data structures to prospective data requests that will be executed using them. If data structures are employed in the service of a lot of prospective data requests, then its building cost can be longer amortized, making individual payments smaller. The opposite holds for structures that are predicted to be used by few prospective queries. The prediction should be ongoing in order to fix errors using feedback from real incoming workload.

There has been notable work on workload prediction in grid and super-computer environments [25, 26, 27, 28]. These works deal with the problem of workload modeling for the production of synthetic workloads employed for the evaluation of real systems. They focus on the site execution locality [25] or the temporal locality [27] of workload characteristics such as runtime, memory and CPU usage etc [26]. These works are orthogonal to the prediction model we need and focus on workload characteristics of the data level. Specific works on modeling job arrivals [28, 29] can be complementary to the discussed prediction model, since they can provide the workload distribution for special grid environments. Structure usage prediction has been studied in other areas, such as the processor cache-line access and survivability prediction [30]. One proposal is a trace-based mechanism that predicts when a cache-line

has been last accessed [31]. Another is a time-based mechanism that predicts the death of a block after a specific timeout period [32]. A third one is a counting-based predictor [33] that predicts the line to be dead after a fixed number of accesses. The prediction model included in the proposed framework should be on the lines of the last one, extended, however, to utilize the computing and storage power available to full processors compared to the limited power on a cache controller.

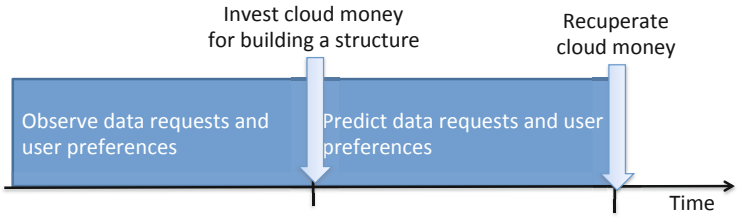

Fig. 8. Observation is followed by prediction of data requests and user preferences

4.4 Pricing Scheme

The cloud provider makes profit from selling its services at a price that is higher than the actual cost. Setting the right price for a service is a non-trivial problem, because when there is competition the demand for services grows inversely but not proportionally to the price. There are two major challenges when trying to define an optimal pricing scheme for the cloud caching service. The first is to define a simplified enough model of the price-demand dependency, to achieve a feasible pricing solution, but not an oversimplified model, which is not representative. For example, a static pricing scheme cannot be optimal if the demand for services has deterministic seasonal fluctuations. The second challenge is to define a pricing scheme that is adaptable to (i) modeling errors, (ii) time-dependent model changes, and (iii) stochastic behavior of the application. For instance, service demand may depend in a non-predictable way from factors external to the cloud application, such as socioeconomic situations.

Pricing schemes were proposed recently for the optimal allocation of grid resources in order to increase revenue [34], or to achieve an equilibrium of grid and user satisfaction [35], assuming knowledge of the demand for resources or the possibility to vary the price of a resource for different users. Similarly, dynamic pricing for web services [36] focuses on scheduling user requests. Moreover, dynamic pricing for the provision of network services [37, 38] aims at achieving a game-theoretic equilibrium through price control among competitive Internet Service Providers. The problem of revenue management through dynamic pricing is well-studied [39]. 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 [40], seasonal products [41], stochastic demand [42]. Data services necessitate a new dedicated pricing scheme since they are distinguished from consumable products in two major ways: (i) they are not exhausted while they are consumed and (ii) the demand for a specific service pauses while this is not available.

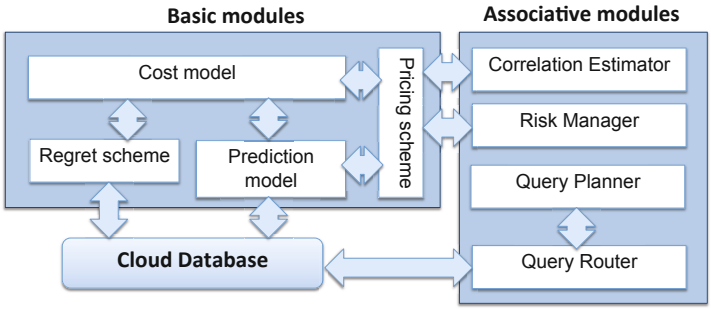

Fig. 9. Modules of the cost-aware data management framework

5 Associative Framework Modules

Section 4 describes the basic modules of the proposed framework in order to achieve cost-aware data management in a cloud data service provider. Beyond the basic modules, the framework should include some associative modules that are necessary in order to make the framework applicable and effective in a real cloud environment:

1. **An estimator of the correlation of offered data services:** All three modules, the regret scheme, the prediction model and the pricing scheme, make decisions about offered data services. These decisions may be ineffective or even wrong if essential correlations between data services are ignored. All three modules could benefit from information on such correlation, input by a separate associative module that is able to estimate it.
2. **A middleware for workload execution:** The basic modules operate and take decisions on a query-per-query basis. Yet, in a real cloud data management environment users would most probably like to receive services for execution of large workloads and not for individual queries. A middleware that takes care of planning the execution of a group or even a workflow of data requests and routes these to the cloud DBMS is necessary.
3. **A risk manager:** All basic modules take decisions based on cost and performance estimations. Naturally, estimations are expected to be wrong occasionally. In such cases, the respective decisions are wrong and have a negative effect to the cloud provider operation, and to its economic viability and profitability. We need to have some knowledge about the extent of such negative effects and relate them with recovery and calibrating procedures. A module that can take all such information into account, and estimate and manipulate the risk they incur for the provider, is necessary in a real cloud environment.

Fig. 9 shows all the modules of the framework.

5.1 Estimator of Data Service Correlation

It is expected that the employment of data structures in the provision of data services will exhibit utility correlations. Therefore the regret scheme but, most importantly, the pricing model should take into account that the data structures may compete or collaborate during query execution. 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 Ib =T(B), Ic =T(C), and Ibc =T(BC) are most important, since they can satisfy the conditions in the 'where' clause. If the cloud DBMS uses Ibc, then the indexes Ib and Ic, will never be used, since Ibc can satisfy both conditions. Therefore, the presence of Ibc has a negative impact on the demand for Ib and Ic. Alternatively, if the cloud DBMS uses Ib, then Ic can also improve query performance via index intersections, hence increasing the profit for the cloud. Therefore, indexes Ib and Ic have positive impact on each other's demand. Thus, it is necessary to provide a measure that estimates demand correlations among data structures. We form three requirements for this measure.

1. The measure is able to capture both competitiveness and collaborativeness of data structures.
2. The computational complexity is low so that its performance is satisfying even in the presence of a big number of data structures.
3. The measure computes normalized values, in order to provide fair comparison.

Recently, the authors of [43] proposed a technique that computes the correlation between indexes. Yet, this technique does not satisfy any of the above requirements.

Beyond measuring the correlation of data structures, the module could also measure the correlation of the performance of different CPUs or data partitions on the cloud disk. Since data services utilize data structures for accelerating data search, as well as CPU and data partitions for parallelization of execution, we can use the correlation measurements of all the latter in order to classify data structures and parts of the infrastructure. Such classifications can be used to form basic data services and build complex ones on top of them. The correlation of the latter can be, therefore, based on the correlation of basic data services that are involved.

5.2 A Middleware for Workload Execution

In a real cloud environment, we expect users to ask for the service of a workload and not just individual queries. Such a workload can be a group or even a workflow of data requests, in which queries can be interleaved by algorithmic processing. Such a query workflow may represent an experimental analysis on scientific data and may be accompanied by data computing that has to be performed between queries. Thus, it is necessary to provide in the proposed framework a middleware that takes as input a set of queries and routes these queries into the cloud appropriately, in parallel and sequential combinations. The middleware architecture can enable the realization of analysis workflows by providing interaction of queries. The challenge in designing the middleware is to achieve a functionality that resembles that of a declarative

programming language. The analysis workflow should be received by the middleware as a program by a interpreter. The comprised queries would correspond to the parameterized functions of the program that have to be executed in a specific (partial) order.

The middleware includes an overall query planner that schedules the queries involved in a workflow or, even more, in many workflows. The planner has to compute, update and use estimations for the overall cost and performance. Using such estimations, the planner enables efficient parallel processing of independent queries. For this task we can employ experience in job scheduling and query optimization.

5.3 Incorporating Risk Management

All basic framework modules operate on the basis of various estimations. Yet, it is very probable that the real operation of the cloud provider would prove these estimations wrong, in which case the provider runs the risk to become economically non-profitable, and furthermore, non-viable. The uncertainty of the behavior of such an environment comes from several sources: unknown changes in service availability (e.g. data updates), unpredictable extremes of service requests (spikes), and unpredictable failures of cloud hardware. Thus, it is necessary to include in the framework an associative module that handles the risk that originates from these factors. The risk module estimates the risk based on statistics and includes tailored risk management procedures for the rest of the modules. Having a separate module to handle the risk allows the implementation, testing and customization of various risk management techniques, without affecting the design of the basic modules. Achieving the realization of an effective risk module will allow the design of a wide range of SLAs between the users and the cloud data service provider.

6 Current State of Work and Future Plans

In [4] we make an initial proposal of an economic model suitable for a self-tuned cloud cache. We have defined the cloud infrastructure for data that are massively collected and queried, such as scientific data and we have studied the characteristics of query workloads that are amenable to caching, and, therefore, economically suitable to be served by a cloud data management system. Typically, the queries must have two properties: first, they have data access locality, i.e. they mostly target a specific part of the data; second, queries have temporal locality, i.e. similar queries are posed close in time. We make an initial proposal of a cost model that takes into account all possible query and infrastructure expenditure. Our experimental study proves that the proposed solution is on the right track and viable for a variety of workloads and data.

In [5] we present a prediction model that estimates the survivability of the new data structure in the cloud DBMS, which is based on two factors: (i) usefulness of the structure, i.e. the probability that the structure is employed in data services w.r.t. time, and (ii) the stability of the structure, i.e. the probability that data related to the structure are not updated w.r.t. time.

In [6] we present a pricing scheme that is suitable for a cloud cache. The scheme is based on a novel price-demand model designed specifically for a cloud cache. There are two major challenges in designing such model: First, we need a model that represents the reality in a simplified, in order to be computable at runtime with thousands of parameters, but not over-simplified manner, in order to capture the factors that determine the price-demand relation. Together with the pricing scheme we also showed a first effort in designing a correlation estimator for data structures.

We consider all the above work to be the first step towards the exploration of the applicability of the proposed framework in a cloud data service provider. We intend to revisit, extend and generalize the current work so that the modules work for sets or workflows of queries or data requests beyond queries. Also, most of the current work does not take into account correlations of data services and parallelization of execution on many machines. Furthermore, we have not worked yet on the middleware for workload execution and the risk manager. Finally, the above work was produced in parts that were designed and tested separately. We need to go a step further and extend the work so that all modules are units that constitute a black box for the outside environment, yet they collaborate transparently, as in the proposed framework.

Until now our experimental study has been based on real and synthetic datasets and query workloads. The real datasets and workloads were from SDSS 2006 [44]. In the future we intend to perform experimental studies on a diversity of data and queries that will allow for a thorough testing of the framework. Furthermore, we can use these real datasets and workloads in order to produce synthetic ones that will allow for a thorough sensitivity testing. Roughly, synthetic datasets should vary: (i) selectivity, (ii) joins, and (iii) aggregation of query operations; and by (i) varying the value range of existing attributes or entity properties, and (ii) adding more attributes or entities. The general goal of our methodology is to perform thorough sensitivity experimentation on all aspects of the proposed data-aware cloud economy.

7 Conclusion

This paper proposes a generic framework for cost-aware data management in a cloud environment. The framework comprises basic and associative modules that each take care of one aspect of integrating cost-aware and traditional, performance-aware, data management. The modules are designed to give input and output to each other and to the cloud DBMS that executes the user data requests. In this way, each one of them can be realized in a manner that is suitable for the cloud application at hand, without interfering with the realization of the rest. The whole framework can be plugged on top of a cloud DBMS. We have already promising results from a first effort to realize the basic modules of the framework for a cloud cache.

References

1. http://aws.amazon.com/ec2
2. http://www.gogrid.com

3. http://code.google.com/appengine
4. Dash, D., Kantere, V., Ailamaki, A.: An Economic Model for Self-Tuned Cloud Caching. In: ICDE, pp. 1687–1693 (2009)
5. Kantere, V., Dash, D., Gratsias, G., Ailamaki, A.: Predicting cost amortization for query services. In: ACM SIGMOD, pp. 325–336 (2011)
6. Kantere, V., Dash, D., Francois, G., Kyriakopoulou, S., Ailamaki, A.: Optimal Pricing for a Cloud Cache. The IEEE TKDE, Special Issue on Cloud Data Management 23(6), 1345–1358 (2011)
7. Ramaswamy, L., Liu, L., Iyengar, A.: Cache clouds: Cooperative caching of dynamic documents in edge networks. In: ICDCS, pp. 229–238 (2005)
8. Ramaswamy, L., Liu, L., Iyengar, A.: Scalable delivery of dynamic content using a cooperative edge cache grid. IEEE TKDE 19(5) (2007)
9. Bhattacharjee, S., Calvert, K.L., Zegura, E.W.: Self-organizing wide-area network caches. In: IEEE Infocom, pp. 752–757 (1998)
10. Malik, T., Burns, R.C., Chaudhary, A.: Bypass caching: Making scientific databases good network citizens. In: ICDE, pp. 94–105 (2005)
11. Wang, X., Burns, R.C., Terzis, A., Deshpande, A.: Network-aware join processing in global-scale database federations. In: ICDE, pp. 586–595 (2008)
12. Foster, I.: What is the grid? a three point checklist (2002), http://www-fp.mcs.anl.gov/foster/articles/whatisthegrid.pdf
13. Nieto-Santisteban, M.A., Gray, J., Szalay, A.S., Annis, J., Thakar, A.R., Omullane, W.J.: When database systems meet the grid. In: CIDR, pp. 154–161 (2005)
14. Watson, P.: Databases and the grid. Grid Computing: Making The Global Infrastructure a Reality, Technical Report (2001)
15. Stonebraker, M., Aoki, P.M., Litwin, W., Pfeffer, A., Sah, A., Sidell, J., Staelin, C., Yu, A.: Mariposa: A wide-area distributed database system. VLDB J. 5(1) (1996)
16. Wellman, M.P., Walsh, W.E., Wurman, P.R., Mackie-Mason, J.K.: Auction protocols for decentralized scheduling. Games and Economic Behavior 35, 2001 (1998)
17. Ernemann, C., Hamscher, V., Yahyapour, R.: Economic Scheduling in Grid Computing. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 128–152. Springer, Heidelberg (2002)
18. Moreno, R., Alonso-Conde, A.B.: Job Scheduling and Resource Management Techniques in Economic Grid Environments. In: Fernández Rivera, F., Bubak, M., Gómez Tato, A., Doallo, R. (eds.) Across Grids 2003. LNCS, vol. 2970, pp. 25–32. Springer, Heidelberg (2004)
19. Kradolfer, M., Tombros, D.: Market-based workflow management. International Journal of Cooperative Information Systems 7 (1998)
20. Chen, C., Maheswaran, M., Toulouse, M.: Supporting co-allocation in an auctioning-based resource allocator for grid systems. In: IPDPS, pp. 89–96 (2002)
21. The Office of Science Data-Management Challenge. Report from the DOE Office of Science Data-Management Workshops (March-May 2004)
22. Ailamaki, A., Kantere, V., Dash, D.: Managing scientific data. Communications of ACM 53(6), 68–78 (2010)
23. Gray, J., Szalay, A.S., Thakar, A., Stoughton, C., van Berg, J.: Online Scientific Data Curation, Publication, and Archiving. CoRR cs.DL/0208012 (2002)
24. Gray, J., Liu, D.T., Nieto-Santisteban, M.A., Szalay, A.S., DeWitt, D.J., Heber, G.: Scientific Data Management in the Coming Decade. CoRR abs/cs/0502008 (2005)
25. Feitelson, D.G.: Locality of sampling and diversity in parallel system workloads. In: ICS, pp. 53–63 (2007)

26. Li, H., Groep, D.L., Wolters, L.: Workload Characteristics of a Multi-cluster Supercomputer. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 176–193. Springer, Heidelberg (2005)
27. Li, H., Muskulus, M., Wolters, L.: Modeling correlated workloads by combining model based clustering and a localized sampling algorithm. In: ICS, pp. 64–72 (2007)
28. Minh, T.N., Wolters, L.: Modeling Parallel System Workloads with Temporal Locality. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2009. LNCS, vol. 5798, pp. 101–115. Springer, Heidelberg (2009)
29. Minh, T.N., Wolters, L.: Modeling job arrival process with long range dependence and burstiness characteristics. In: CCGRID, pp. 324–330 (2009)
30. Calder, B., Grunwald, D.: Next cache line and set prediction. In: ISCA, pp. 287–296 (1995)
31. Lai, A.-C., Fide, C., Falsafi, B.: Dead-block prediction & dead-block correlating prefetchers. In: ISCA, pp. 144–154 (2001)
32. Hu, Z., Kaxiras, S., Martonosi, M.: Timekeeping in the memory system: predicting and optimizing memory behavior. In: ISCA, pp. 209–220 (2002)
33. Kharbutli, M., Solihin, Y.: Counter-based cache replacement and bypassing algorithms. IEEE Transactions in Computing 57(4), 433–447 (2008)
34. Sulistio, A., Kyong Hoon, K., Buyya, R.: Using revenue management to determine pricing of reservations. In: IEEE e-Science, pp. 396–405 (2007)
35. Allenotor, D., Thulasiram, R.K., Thulasiraman, P.: A Financial Option Based Grid Resources Pricing Model: Towards an Equilibrium between Service Quality for User and Profitability for Service Providers. In: Abdennadher, N., Petcu, D. (eds.) GPC 2009. LNCS, vol. 5529, pp. 13–24. Springer, Heidelberg (2009)
36. Lin, Z., Ramanathan, S., Zhao, H.: Usage-based dynamic pricing of Web services for optimizing resource allocation. Inf. Systems and E-Business Management 3(3), 221–242 (2005)
37. Masuda, Y., Whang, S.: Dynamic Pricing for Network Service: Equilibrium and Stability. Management Science 45(6), 857–869 (1999)
38. Cao, X.-R., Shen, H.-X., Milito, R., Wirth, P.: Internet pricing with a game theoretical approach: concepts and examples. ACM Transactions on Networking 10(2), 208–216 (2007)
39. Bitran, G.R., Caldentey, R.: An overview of pricing models for revenue management. MSOM 5(3), 203–229 (2003)
40. Ghose, A., Choudhary, V., Mukhopadhyay, T., Rajan, U.: Dynamic pricing: A strategic advantage for electronic retailers. In: ICIS, p. 28 (2003)
41. You, P.-S., Chen, T.C.: Dynamic pricing of seasonal goods with spot and forward purchase demands. Comput. Math. Appl. 54(4), 490–498 (2007)
42. Gallego, G., van Ryzin, G.: Optimal Dynamic Pricing of Inventories with Stochastic Demand over Finite Horizons. Management Science 40(8), 999–1020 (1994)
43. Schnaitter, K., Polyzotis, N., Getoor, L.: Modeling index interactions. In: VLDB, pp. 1234–1245 (2009)
44. `http://www.sdss.org`