OPTIMAL SERVICE PRICING FOR A CLOUD CACHE

ABSTRACT:

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

EXISTING SYSTEM:

Existing clouds focus on the provision of web services targeted to developers, such as Amazon Elastic Compute Cloud (EC2), or the deployment of servers, such as Go Grid. There are two major challenges when trying to define an optimal pricing scheme for the cloud caching service. The first is to define a simplified enough model of the price demand dependency, to achieve a feasible pricing solution, but not oversimplified model that is not representative.

A static pricing scheme cannot be optimal if the demand for services has deterministic seasonal fluctuations. The second challenge is to define a pricing scheme that is adaptable to

(i) Modeling errors, (ii) time-dependent model changes, and (iii) stochastic behavior of the application. The demand for services, for instance, may depend in a nonpredictable way on factors that are external to the cloud application, such as socioeconomic situations.

Static pricing cannot guarantee cloud profit maximization. In fact, as we show in our experimental study, static pricing results in an unpredictable and, therefore, uncontrollable behavior of profit. Closely related to cloud computing is research on accounting in wide-area networks that offer distributed services. Mariposa discusses an economy for querying in distributed databases. This economy is limited to offering budget options to the users, and does not propose any pricing scheme. Other solutions for similar frameworks focus on job scheduling and bid negotiation, issues orthogonal to optimal pricing.

Disadvantage:

- A static pricing scheme cannot be optimal if the demand for services has deterministic seasonal fluctuations.
- Static pricing results in an unpredictable and, therefore, uncontrollable behavior of profit.

PROPOSED SYSTEM:

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.

Price adaptivity to time changes:

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

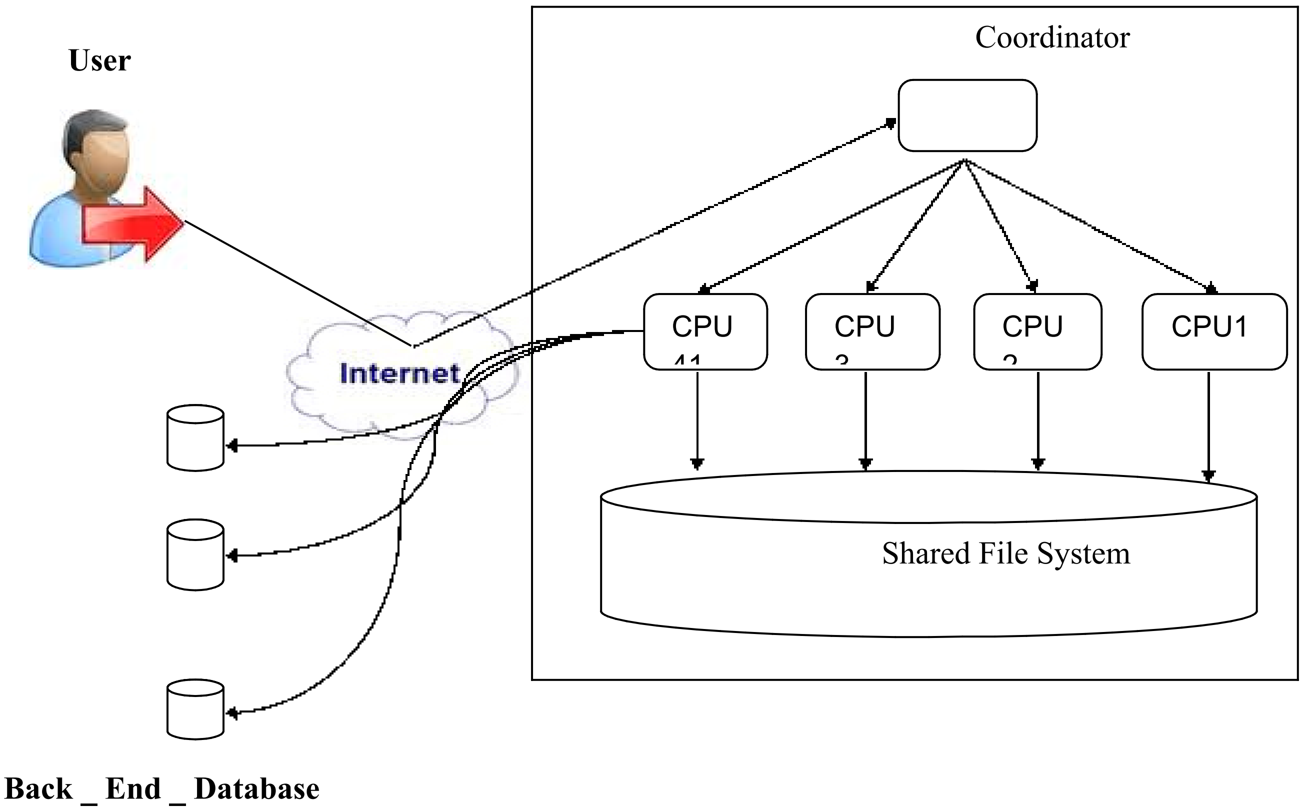
Modeling structure correlations:

Our approach models the correlation of cache structures as a dependency of the demand for each structure on the price of every available one. Pairs of structures are characterized as competitive, if they tend to exclude each other, or collaborating, if they coexist in query plans. Competitive pairs induce negative, whereas collaborating pairs induce positive correlation. Otherwise correlation is set to zero. The index-index, index column,
and column-column correlations are estimated based on proposed measures that can estimate all three types of correlation. We propose a method for the efficient computation of structure correlation by extending a cache based query cost estimation module and a template-based workload compression technique.

Advantage:

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

ARCHITECTURE:



ALGORITHM:

```
**Global**: cache structures $S$, prices $P$, availability $\Delta$
**Query Execution** ( )
**if** $q$ can be satisfied in the cache **then**
(result, cost)$\leftarrow$runQueryInCache (q)
**else**
(result, cost)$\leftarrow$runQueryInBackend (q)
**end if**
$S\leftarrow$addNewStructures ()
**return** result, cost
**optimalPricing** (horizon $T$, intervals $t[i]$, $S$)
$(\Delta, P)\leftarrow$determineAvailability&Prices (T, t,S)
**return** $\Delta, P$
```

```
main ()
execute in parallel tasks T1 and T2:
T1:
for every new i do
slide the optimization window
OptimalPricin (T, t[i], S)
end for
T2:
While new query q do
(Result, cost)←query Execution (q)
end while
if q executed in cache then
Charge cost to user
else
Calculate total price and charge price to user
end if
```

System Requirements:

Hardware Requirements:

➢ System	: Pentium IV 2.4 GHz.
➢ Hard Disk	: 40 GB.
➢ Floppy Drive	: 1.44 Mb.
➢ Monitor	: 15 VGA Colour.
➢ Mouse	: Logitech.
➢ Ram	: 512 Mb.

Software Requirements:

➢ Operating system	: Windows XP.
➢ Coding Language	: ASP.Net with C#
➢ Data Base	: SQL Server 20**05**

MODULES:

Query Execution:

The cloud cache is a full-fledged DBMS along with a cache of data that reside permanently in back-end databases. The goal of the cloud cache is to offer cheap efficient multi-user querying on the back-end data, while keeping the cloud provider profitable. Service of queries is performed by executing them either in the cloud cache or in the back-end database. Query performance is measured in terms of execution time. The faster the execution, the more data structures it employs, and therefore, the more expensive the service. We assume that the cloud infrastructure provides sufficient amount of storage space for a large number of cache structures. Each cache structure has a building and a maintenance cost.

Optimal Pricing:

We assume that each structure is built from scratch in the cloud cache, as the cloud may not have administration rights on existing back-end structures. Nevertheless, cheap computing and parallelism on cloud infrastructure may benefit the performance of structure creation. For a column, the building cost is the cost of transferring it from the backend and combining it with the currently cached columns. This cost may contain the cost of nte grating the column in the existing cache table. For indexes, the building cost involves fetching the data across the Internet and then building the index in the cache.

Since sorting is the most important step in building an index, the cost of building an index is approximated to the cost of sorting the indexed columns. In case of multiple cloud databases, the cost of data movement is incorporated in the building cost. The maintenance cost of a column or an index is just the cost of using disk space in the cloud. Hence, building a column or an index in the cache has a one-time static cost, whereas their maintenance yields a storage cost that is linear with time.

REFERENCE:

Varena Kantere, Debabrata Dash, Gregory Francois, Sofia Kyriakopolou an dAnastasia Ailmaki, “Optimal Service Pricing for a Cloud Cache”, **IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No.9, September 2011.**