Dynamic Pricing Model for a Cloud Cache Environment

Saravanan. K [#1] and Sri Vigna Hema. V [*2]

[#]Assistant Professor, Department of Computer Science and Engineering, Regional Centre Anna University Tirunelveli, India

[*]Department of Computer Science and Engineering, Regional Centre, Anna University Tirunelveli, India

***Abstract*—Cloud computing is an emerging innovative IT-based business model which catches the attention of practitioners, for its potentiality of industry adoption, as well as of academicians, for research undertaking in different dimensions. Cloud data management services are prevailing to proffer quality query services via cloud cache environment. The clients can propound their query to the cloud data and charge price as-per-usage. Cloud needs a financial prudence to manage service of multiple users in handy manner. Currently, Cache as a Service (CaaS) model is proposed as an elective services to offer infrastructure which assist to control the cloud economy by affording cloud profit. A dynamic pricing scheme exploits a new-fangled method that estimates the cost of a cache structure and enhances the cloud revenue assure for customer satisfaction using price-demand model premeditated for cloud cache. As a result, it endows with preeminent and cost-efficiency solution for both users and cloud providers.**

***Keywords*—Cloud Computing, Query Services, Cache as a Service, Dynamic Pricing, Cost Efficiency**

I. INTRODUCTION

An emerging trend for delivering computing services over the internet is referred as cloud computing, which acts as a virtualization of resources that maintains and manages itself. There is no need to buy an infrastructure to run a service; instead it gives the option for pay as-you-go need basis. It is a platform where consumer can buy computing and storage power on a rental basis. It provides unlimited infrastructure to store and execute customer data and program. The benefits of cloud computing are: minimized capital expenditure, location and device independence, utilization and efficiency improvement, very high scalability, high computing power. Cloud suppliers buy and sell their services on cloud cache for capital. The building and maintenance cost of used resources is pay per usage.

Customer can pose their queries to the cloud via the cloud coordinator and charged per usage. Then, the cloud looks for a query either in cloud cache or in backend database at a low cost to provide efficient querying. Herein, cloud cache is designed as similar to [8]. Each formation of cache has an operating cost (i.e., a building and a maintenance cost). A price over this formation can guarantee user satisfaction and cloud profit.

Cloud data administration needs a financial system to manage services of multiple users and its motive is to make a pricing model more effective by providing customer satisfaction and maximize the revenue for cloud. For managing revenue, a better response time and support for pricing model using QoS model is provided the same as [1],[4]. Static pricing cannot be an optimal since it has a fixed price concept. It also cannot guarantee the customer satisfaction. To overcome this, dynamic pricing scheme is proposed and its objective is to maximize the cloud profit and incorporates it for customer satisfaction using real time pricing or dynamic pricing [10], [11], [13], and [14].

With the intention of maximizing the cloud profit, Cache as a Service (CaaS) model is introduced using Amazon's Elastic Cloud Cache (EC2) system which forms relational database service [15]. CaaS model is taken as an optional service to offer infrastructure as a service and consists of two main components: 1) An elastic cache model as the architectural foundation, 2) A service model with a pricing scheme as the economic foundation. It provides better performance and profit (i.e., cost efficiency) with a novel pricing scheme. If an extra cost exists, this model provide the High Performance (HP) and non-HP services with the use of Local Memory (LM) as Cache and best value services with the use of Remote Memory (RM) as Cache. It gives better service for both user and provider of the cloud.

The remainder of this paper is organized as follows: Section 2 reviews the related work about pricing and caching concept of cloud computing. Section 3 articulates the overall architectural design of dynamic pricing model for cache cloud environment. Section 4 models and explains the query execution, CaaS and dynamic pricing. Section 5 presents the result and analysis of dynamic pricing. Section 6 concludes the work done and future direction.

II. RELATED WORKS

In a resource souk with a profusion of cloud providers and users, fixed pricing does not reflect the existing market price due to the changing demand and supply. This leads to lower cloud profit and to excessive price, i.e., under-demand [5]. Early on cloud services such as Sun Grid Compute Utility were confined to one resource type, e.g. CPU time [6]. More recent services, such as Amazon S3 [9] and EC2 [8], introduced more resource types, i.e. storage and bandwidth. In

turn to maximize the profit for a cloud, a dynamic pricing scheme is proposed for better computational efficiency and even more money-making.

D.Dash et al. [3] attained the maximum cloud profit by reducing overhead of a query plan cache, and improve designer performance. The drawback of revenue management owing to dynamic pricing service is revised [1]. Dynamic pricing schemes were proposed for optimal pricing under a complex mix of jobs [10], or to outperform fixed pricing strategies [11]. Correspondingly, dynamic pricing for web services [12] concentrate on setting up the user requests for better service. Besides, dynamic pricing for the demand of network services [7], [2] intend to provide an optimal solution for both Internet Service Providers and users. This works is alike to propose objective which focuses on the maximization of profit for the cloud.

The difficulty of improving cost efficiency is well studied [16] by using Cache as a Service (CaaS) model. Based on the design rationale of Elastic Cache system, architectural foundation is structured and providers profit increases due to improvement in server consolidation. Several studies [17], [20] contributes to the improvement in performance of I/O.

III. System Architecture

A dynamic pricing model for a cloud cache environment architectural diagram is provided in Fig. 1 which explains its overall structure and functions. In this section 3, there are three main components which consist of: Application Service, Query Service, and Pricing Service.

A. Application Service

Clients can pose their query via cloud provider to this service. It incorporates the services such as authentication, verification, guarantee, justification, and optimization. Posed queries were called by the query service for further action. This component also helps to return a result back to the clients by processing the other two components.

B. Query Service

The purpose of a query is to display several tables of information onto one data sheet. Query service is used to process a query which posed by a clients via cloud provider. It is further consists of following component Query Engine, Query Tool, Query Reformation, Cache Database, and Backend database which has been explained as follows:

1) Query Engine: A query engine is extremely useful for creating a column oriented back-end database. It is a cache-aware engine which uses several low-level optimization techniques to guarantee execution times and high throughput.

While a query is parsed to a query engine, it will first apply to a cache memory to check whether a query is in cloud cache database or not. If present, it will return a matched query and submit it to pricing service. Otherwise, it will verify a query into backend database for a matched query.

2) Query Tool: Query tool software allows a user to easily select database. The tool used in this service is QTODBC 6.1 which is a Universal Data Access (UDA) tool. It allows to query ODBC data sources, author SQL scripts and queries, return query results to a grid or free-form text, retrieve ODBC driver information, execute multiple SQL scripts or stored procedures simultaneously, and more.

3) Query Reformation: A meaningful word of a query is renovated as keywords to make a searching process more accurate. Thus, the reformed query result will useful for verifying a query either in cache or in backend database.

4) Cache Database: Cloud cache is similar to a history which stores all the information temporary. It maintains a record as cloud cache database and makes a search easily by a customer. The files that automatically request by looking at a web page are stored on hard disk in a cache subdirectory beneath the directory for browser (for example, Internet Explorer). When an end user returns to a page that recently looked at, the browser can get it from the cache rather than the original server. It will be faster and saves time, when a page is loaded in a server side. Thus, it reduces the server workload and is resistance to traffic. Depending on particular browser, cache size varies.

5) Backend database: A back-end database is accessed by users indirectly through an external application rather than by application programming stored within the database itself or by low level manipulation of the data (e.g. through SQL commands). It stores data but does not include end-user application elements such as stored queries, forms, macros or reports. It is not widely used among developers using larger or enterprise database systems. This is because enterprise database systems enforce the use of the client–server model and do not have the option to include the application programming within their databases. All such databases are used as back-end databases and so are redundant.

The results produced by a query service component by retrieving a query either form a cache or backend database is submitted to the pricing service for further estimation of a cost.

C. Pricing Service

Price service is used to evaluate a cost after getting a resultant query either from a cloud cache or backend database. A pricing system which is based upon revenue sharing between the service provider and client is a cause for cost savings.

The pricing service is a component of price, quote on cloud. It acts as a core for every pricing calculation, even if pricing data is passed to a third party. It provides the ability to store and maintain all the pricing data at the pricing organization. Either an organization can be defined as a pricing organization and maintain its own pricing data, or an organization can specify a pricing organization of its choice.

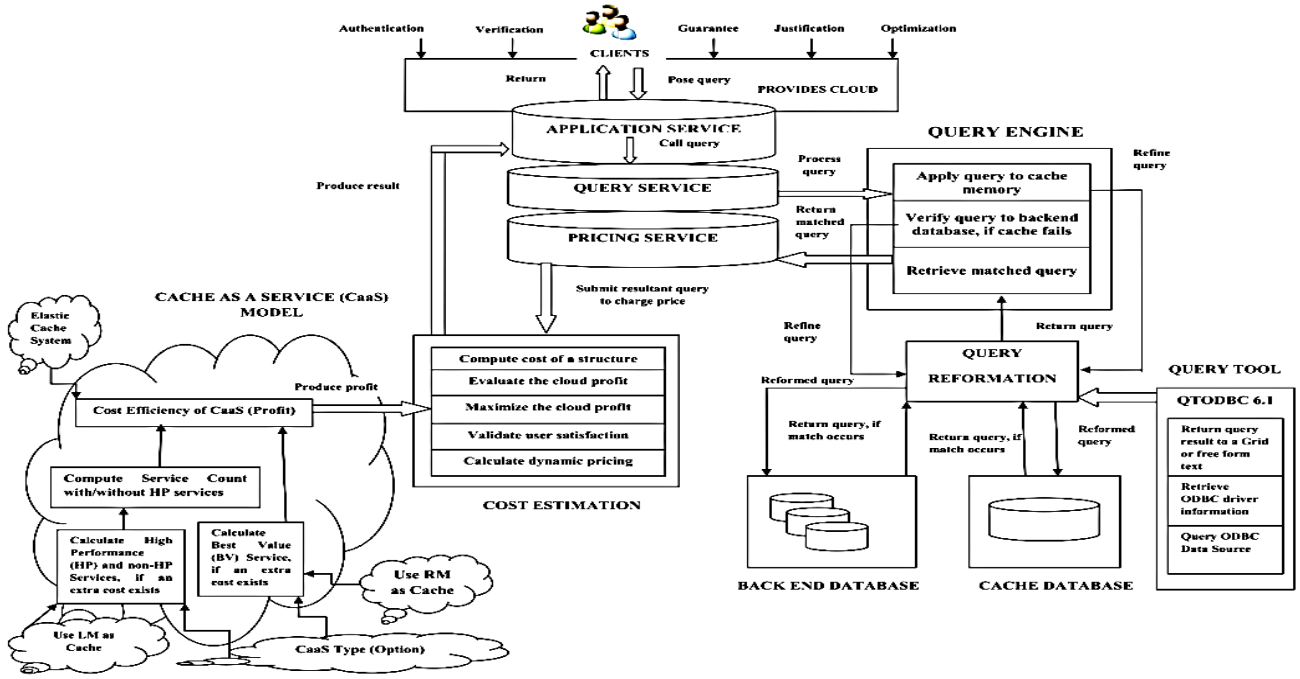


Fig. 1 Dynamic pricing model for a cloud cache environment

This enables users to define price lists, maintain prices for each result, calculate user satisfaction, and assign price lists either directly to customers or indirectly to customer attributes.

1) Cost Estimation: Cost estimation models are mathematical algorithms or parametric equations used to estimate the costs of a resultant query. The algorithms are universally computerized. Models typically function through the input of parameters that describe the attributes of the result. The model then provides as output various resources requirements in cost and time.

The cloud cache bid to the customers query services on the cloud data. The customer queries are responded by query strategy that utilizes cache formation, i.e., cached columns and indexes. The set of possible cache structures {S1… Sn} are in use. Whenever a structure S is constructing in the cache, it has a one-time building cost. At the same time as S is uphold in the cache it has maintenance cost which depends on time.

Compute cost of a cache structure, which computes an operating cost i.e., Building cost (Bs) which is a one-time static cost and Maintenance cost (Ms) yields a storage cost that is linear with time t. Cache services are offered through query execution that uses cache structures. The Cost of a cache structure S,

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

Evaluate the cloud profit, Depends upon demand for a cache structure, the profit of the cloud is estimated. The demand for a cache structure S, denoted as λ_S (t), is the number of times that S is employed in query plans selected for execution at time t.

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 as,

$$\lambda s_i(t) \geq 0(if\ in\ cache), \lambda s_i(t) = 0(otherwise) \quad (2)$$

The cloud makes profit by charging the usage of structures in selected query plans for a price $p_S(t)$.

$$r(t) = \sum_{i=0}^{m} \delta_i(\lambda_{s_i}(t).p_{s_i}(t) - c_{s_i}(t) \quad (3)$$

where δ_i represents the fact that the structure Si is present in the cloud cache. Distinctively, 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.,

$$\delta_i = \begin{cases} 0\ or\ 1, & if\ t\ belongs\ to\ [0,T] \\ 0, & otherwise \end{cases} \quad (4)$$

The goal is to maximize the total profit by choosing which structures to build or discard and which price to assign to each built structure at any time.

Maximize Cloud Profit, The maximization of the cloud DBMS profit is achieved with the solution of the optimization problem which is subject to the constraint on price demand dependency.

$$max\ R(t) = \int_0^T r(t)dt \quad (5)$$

The philanthropic tend of dynamic pricing optimization is expressed as: 1) a guarantee for customer satisfaction, or 2) a further intention of profit maximization.

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

$$\frac{d\Lambda}{dt} \geq \frac{d\lambda}{dt}\Big|_{\min} \quad (6)$$

where $\lambda_{\min}$ is the selected minimum value of demand drop rate.

Validate user satisfaction, A pricing optimization should guarantee for a low limit on user satisfaction and thus can be defined as the difference of the structure price and the actual cost,

$$u(t) = p_s(t) - c_s(t) \quad (7)$$

In this case, the problem can hold either a new constraint or a new optimization objective. In the first case, the constraint can be $u(t) \leq r_{min}$, where r_{min} is the selected minimum value of cloud profit.

Calculate a dynamic pricing, To achieve an optimal price that guarantees the user satisfaction and maximizing the cloud profit, a dynamic pricing is calculated. Dynamic pricing, also known as time-based pricing, occurs when customers are divided into two or more groups with different prices charged to each group. When done successfully, price discrimination can increase the profit of a cloud. Optimized cloud profit is a combination of (4) and (5),

$$max\ R(t) = \int_0^T (r(t) - w.u(t))dt \quad (8)$$

where w is the weight that calibrates the influence of the optimization procedure.

2) Cache as a Service Model: The CaaS model consists of two main components: An elastic cache system as the architectural foundation and a service model with a pricing scheme as the economic foundation as shown in Fig. 2.

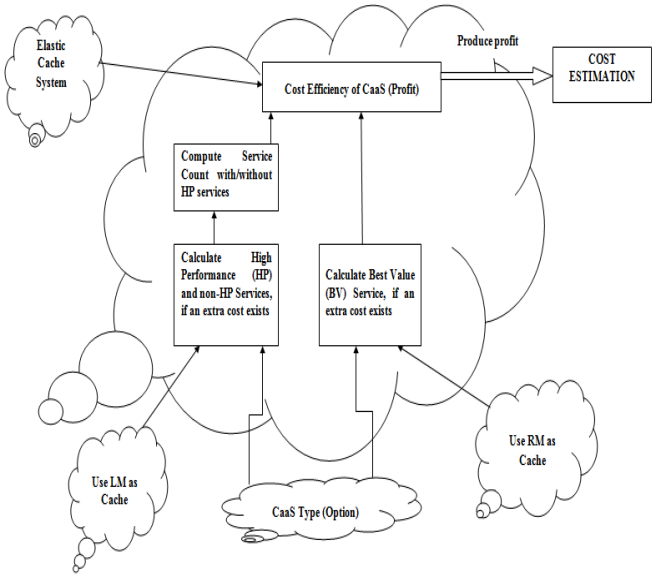


Fig. 2 CaaS model

Modeling elastic cache system, Elastic cache system is designed and is the key component in realizing CaaS. Important factors to design this elastic cache system are cache medium, communication between a cache server and a Virtual Machine (VM) and the implementation level of cache system. Cache medium consists of three options to implement cache devices are Local Memory (LM), Remote Memory (RM), and Solid State Drive (SSD). In an implementation level, elastic cache can be deployed at either application or OS level. A VM demands RM for a use as a disk cache and implement a new block device driver (RM cache device).

Pricing Model, If an extra cost exists, the performance and profit (cost efficiency) are improved by calculating High Performance (HP) services and Non-HP services. HP and Non HP Services is the average number of services (VMs) per physical machine with/without HP cache option. Then, the average number of services (service count or sc) per physical machine with/without HP requests is calculated. Best Value (BV) is determined, if users send a request with a RM cache option. Finally, performance gain that users experience with HP and BV is provided as high profit (Cost Efficiency) which is useful for further dynamic pricing model.

Finally, the result of an estimated cost is returned to an application service which in turn returns an optimal price to corresponding clients.

IV. IMPLEMENTATION

The dynamic pricing optimization problem is implemented and run in CloudSim 3.0 toolkit. Clouds enable platform for dynamic and flexible application. So, users can access and deploy applications from anywhere in the internet driven by demand and QoS requirements.

A. Cloud Environment

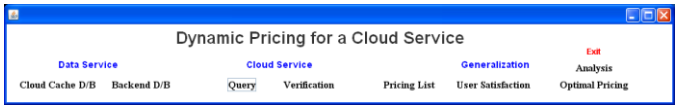
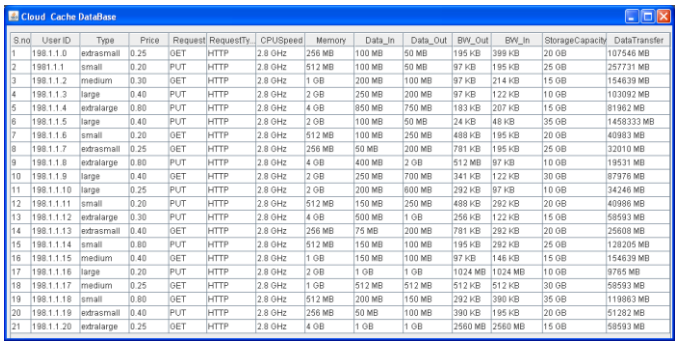


Fig. 3 Cloud Area

Fig. 3 is a Cloud area which is created to make a process for cloud services. It includes data service which consists of Cloud cache database and backend database. The Cloud Service is provided for Query application and Query verification. A Generalization process is for showing price list, user satisfaction list and an optimal pricing list.

B. Cloud Cache Database and Backend Database



S.no	User ID	Type	Price	Request	RequestTy..	CPUSpeed	Memory	Data_In	Data_Out	BW_Out	BW_In	StorageCapacity	DataTransfer
1	198.1.1.0	extrasmall	0.25	GET	HTTP	2.8 GHz	256 MB	100 MB	50 MB	195 KB	399 KB	20 GB	107546 MB
2	198.1.1.1	small	0.20	PUT	HTTP	2.8 GHz	512 MB	100 MB	50 MB	97 KB	195 KB	25 GB	257731 MB
3	198.1.1.2	medium	0.30	GET	HTTP	2.8 GHz	1 GB	200 MB	100 MB	97 KB	214 KB	15 GB	154639 MB
4	198.1.1.3	large	0.40	PUT	HTTP	2.8 GHz	2 GB	250 MB	200 MB	97 KB	122 KB	10 GB	103092 MB
5	198.1.1.4	extralarge	0.80	PUT	HTTP	2.8 GHz	4 GB	950 MB	750 MB	183 KB	207 KB	15 GB	81962 MB
6	198.1.1.5	large	0.40	PUT	HTTP	2.8 GHz	2 GB	100 MB	50 MB	24 KB	48 KB	35 GB	1458333 MB
7	198.1.1.6	small	0.20	GET	HTTP	2.8 GHz	512 MB	100 MB	250 MB	488 KB	195 KB	20 GB	40983 MB
8	198.1.1.7	extrasmall	0.25	GET	HTTP	2.8 GHz	256 MB	50 MB	200 MB	781 KB	195 KB	25 GB	32010 MB
9	198.1.1.8	extralarge	0.80	PUT	HTTP	2.8 GHz	4 GB	400 MB	2 GB	512 MB	97 KB	10 GB	19531 MB
10	198.1.1.9	large	0.40	GET	HTTP	2.8 GHz	2 GB	250 MB	700 MB	341 KB	122 KB	30 GB	87976 MB
11	198.1.1.10	large	0.25	PUT	HTTP	2.8 GHz	2 GB	200 MB	600 MB	292 KB	97 KB	10 GB	34246 MB
12	198.1.1.11	small	0.20	PUT	HTTP	2.8 GHz	512 MB	150 MB	250 MB	488 KB	292 KB	20 GB	40986 MB
13	198.1.1.12	extralarge	0.30	PUT	HTTP	2.8 GHz	4 GB	500 MB	1 GB	256 KB	122 KB	15 GB	58593 MB
14	198.1.1.13	extrasmall	0.40	GET	HTTP	2.8 GHz	256 MB	75 MB	200 MB	781 KB	292 KB	20 GB	25608 MB
15	198.1.1.14	small	0.80	PUT	HTTP	2.8 GHz	512 MB	150 MB	100 MB	195 KB	292 KB	25 GB	128205 MB
16	198.1.1.15	medium	0.40	GET	HTTP	2.8 GHz	1 GB	150 MB	100 MB	97 KB	146 KB	15 GB	154639 MB
17	198.1.1.16	large	0.20	PUT	HTTP	2.8 GHz	2 GB	1 GB	1 GB	1024 MB	1024 MB	10 GB	9765 MB
18	198.1.1.17	medium	0.25	GET	HTTP	2.8 GHz	1 GB	512 MB	512 MB	512 KB	512 KB	30 GB	58593 MB
19	198.1.1.18	small	0.80	GET	HTTP	2.8 GHz	512 MB	200 MB	150 MB	292 KB	390 KB	35 GB	119863 MB
20	198.1.1.19	extrasmall	0.40	PUT	HTTP	2.8 GHz	256 MB	50 MB	100 MB	390 KB	195 KB	20 GB	51282 MB
21	198.1.1.20	extralarge	0.25	GET	HTTP	2.8 GHz	4 GB	1 GB	1 GB	2560 MB	2560 MB	15 GB	58593 MB

Fig. 4 Cloud Storage (Cloud Cache DB)

Fig. 4 shows the Cloud Cache database in which the user's history is stored temporary. It has an attributes of QoS parameter and the user's general information. Thus it acts fast when a server loads a page. This form shows the Backend database that accessed by users indirectly through an external application. It includes the main attribute of QoS parameter.

C. Query Process

Fig. 5 is used to apply the query that posed by a user. If a query applies it will show the message as query applied.

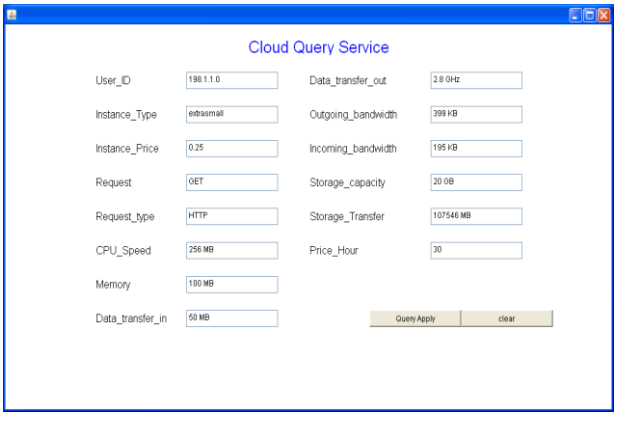


Fig. 5 Cloud Query Service

D. Query Verification

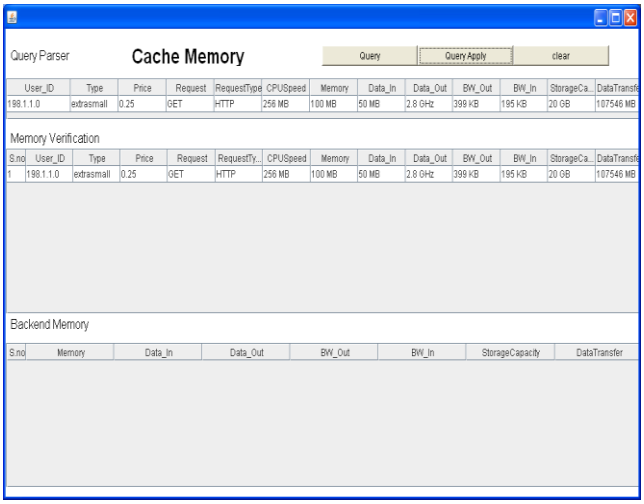


Fig. 6 Query verified in cloud cache

Fig. 6 is used for verifying whether a query is in a cloud cache or in backend database. In this, query pose by a user matches the query that already in cache so it displayed in cloud cache. Otherwise it will verify backend and functions as similar as cache. If both fail, it will display a message as no such database and it will proceeds by pricing.

E. Price List Details

Fig. 7 is used to show the pricing list for a respective query. It calculates the price for a cache structure as mentioned in project description.

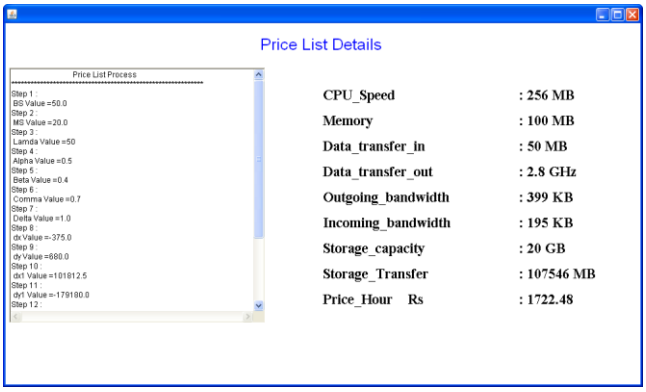


Fig. 7 Pricing Service

F. User Satisfaction Details

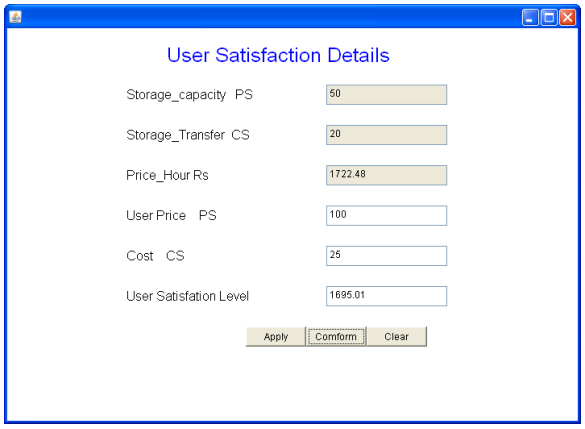


Fig. 8 Dynamic pricing via user

Fig. 8 is to apply the dynamic pricing scheme as user need basis by user. Here, user quotes the price and actual cost and it is applied for satisfaction level that then made to conformed a price

G. Optimal Price Details

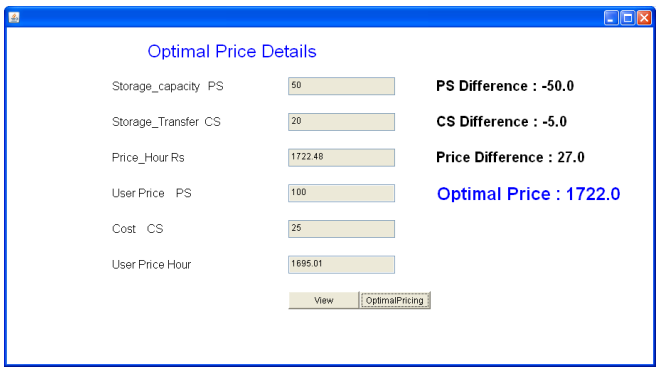


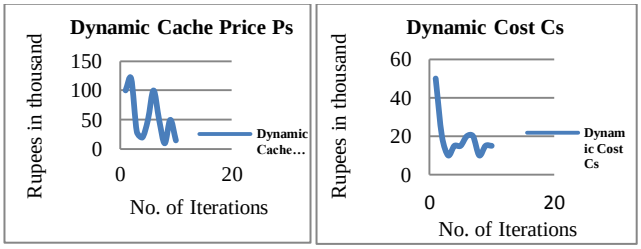
Fig. 9 Price Optimization

Fig. 9 is to choose the optimal price that guarantees a user satisfaction. In this user's quoted price can be viewed and optimal price is chosen and also shows the difference between price listed before and user's price list.

V. Result Analysis

A. Graph for Dynamic Pricing

This section presents result on the dynamic pricing scheme with cache price, cost, and cloud providers profit, as shown in Fig. 10.



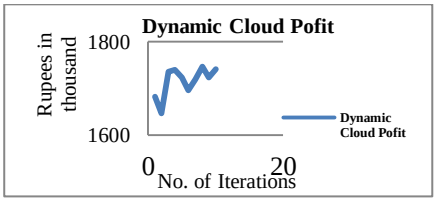


Fig. 10 (a), (b), (c) Results of dynamic pricing

B. Comparison Graph between Dynamic Ps vs. Cs vs. Cloud Profit

This section presents result on the dynamic pricing scheme with the comparison of cache price, cost, and cloud providers profit, as shown in Fig. 11.

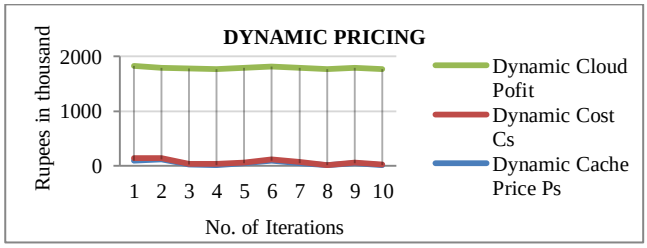


Fig. 11 Comparison of dynamic pricing

C. Comparison Graph between Ps Diff vs. Cs Diff vs. Cloud Profit Diff

This section presents result on the dynamic pricing scheme with the comparison of cache price, cost, and cloud providers profit, as shown in Fig. 12.

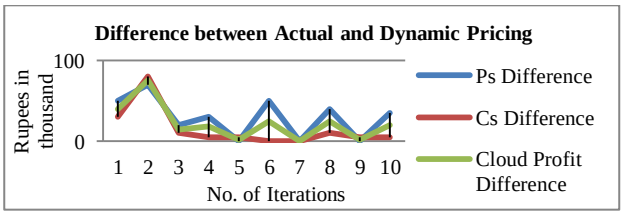


Fig.12 Comparison of actual vs. dynamic pricing difference

VI. Conclusion

This work proposes a dynamic pricing scheme that maximizes the cloud profit and also guarantees user satisfaction that adapts to demand changes. It also provides an efficient querying on the back-end data at a low cost, while being economically viable, and furthermore, profitable. The Cloud economy may be tried to estimate in the future for maximizing the Cloud providers profit and performance improvement (i.e., Cost Efficiency) by reducing the number of active physical machines using Cache as a Service (CaaS) model. Elastic cache system and service model with pricing scheme will be designed and implemented to be cost effective.

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] 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.
[3] 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.
[4] 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.
[5] 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.
[6] I.E. Grossmann and Z. Kravanja, Large-Scale Optimization with Applications: Optimal Design and Control. Springer, 1997.
[7] L. He and J. Walrand, "Pricing Differentiated Internet Services," Proc. IEEE INFOCOM, pp. 195-204, 2005.
[8] http://aws.amazon.com/elastic cloud cache, 2013.
[9] http://aws.amazon.com/s3/, 2013.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] L. Wang, J. Zhan, and W. Shi, "In Cloud, Can Scientific Communities Benefit from the Economies of Scale?," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 2, pp. 296-303, Feb. 2012.
[16] Hyuck Han, Young Choon Lee, Woong Shin, Hyungsoo Jung, Heon Y.Yeom, and Albert Y.Zomaya, "Cashing in on the Cache in the Cloud," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 8, pp. 1387-1399, Aug. 2012.
[17] M.D. Dahlin, R.Y. Wang, T.E. Anderson, and D.A. Patterson, "Cooperative Caching: Using Remote Client Memory to Improve File System Performance," Proc. First USENIX Conf. Operating Systems Design and Implementation (OSDI '94), 1994.
[18] T.E. Anderson, M.D. Dahlin, J.M. Neefe, D.A. Patterson, D.S. Roselli, and R.Y. Wang, "Serverless Network File Systems," ACM Trans. Computer Systems, vol. 14, pp. 41-79, Feb. 1996.
[19] A. Menon, J.R. Santos, Y. Turner, G.J. Janakiraman, and W. Zwaenepoel, "Diagnosing Performance Overheads in the Xen Virtual Machine Environment," Proc. First ACM/USENIX Int'l Conf. Virtual Execution Environments (VEE '05), 2005.
[20] L. Cherkasova and R. Gardner, "Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor," Proc. Ann. Conf. USENIX Ann. Technical Conf. (ATC '05), 2005.