Long-Term Revenue Maximization Pricing Scheme for Cloud

Wen-Kai Huang*, Shao-Shu Huang*, Zheng Li*, Chang-Dong Wang* and Jian-Huang Lai†
*School of Mobile Information Engineering
Sun Yat-sen University, Zhuhai, P. R. China, 519082
Email: huang.wenkai@foxmail.com, shaoshuhuang@foxmail.com, hsqmlz@gmail.com, changdongwang@hotmail.com
†School of Information Science and Technology
Sun Yat-sen University, Guangzhou, P. R. China, 510006
Email:stsljh@mail.sysu.edu.cn

***Abstract*—In recent years, cloud computing has become a new revolution in science and technology. It releases the burden of system maintenance for users and only charges them according to the resources they use. However, the resource pricing system for cloud computing is far from mature, most cloud providers charge users only in a static pricing manner. Though some efforts have been made in developing dynamic pricing schemes for cloud, most of them only focus on maximizing short-term revenue. Currently there is a lack of methodology to optimize long-term revenue for cloud. In this paper, we aim to address this issue by proposing a new pricing scheme which consists of the following four main steps. Firstly, we conduct an empirical study of how users make use of Google's cloud, and find the composition of users according to their using behaviour. Secondly, we introduce a revised agglomerative hierarchical clustering algorithm to cluster users into different types according to their using behaviour with the number of clusters being automatically estimated. Thirdly, for each type of user, a consumption potential prediction model is built by using neutral network. Fourthly, by using dynamic programming, we calculate the optimal strategy that can maximize long-term profit of cloud service provider with considering consumption potential factor generated from prediction. Experiments by simulating users' auction process based on deal making probability evolution have been conducted to demonstrate the effectiveness of our method.**

***Keywords*-cloud computing; auction; dynamic pricing; consumption potential prediction**

I. Introduction

With the development of cloud computing, cloud service has become an important resource for individuals and enterprises, which releases system maintenance for them. Since users only pay according to the amount of resources they consume, appropriate pricing strategy is highly important for cloud service providers.

For Infrastructure as a Service (IaaS) providers, a long-term contract with the fixed price is a preferential choice, since it can guarantee the stability of the infrastructure consumption. Static pricing scheme's simplicity and convenience make it the most widely used pricing scheme nowadays. However, static pricing scheme is far from optimal. This is because IaaS providers often have temporary idle resource, and it is a waste of resources given the fact that these resources are perishable and cause maintenance expense as time goes by. This implies that static pricing scheme can't maximize profits for cloud providers.

A better way compared with static pricing scheme is dynamic pricing based on auction. This kind of deal can be seldom affected by third party. Besides, it can make full use of cloud resources based on some principals of economy, i.e., supply-demand relationship determines deal price.

A. Related Work

Though some cloud providers like Amazon [1] has already brought out their spot instance to charge via auction, [2] proves that Amazon's auction method is not market driven and is generated as a random value twisting around a hidden reserved price within a tightly hidden interval.

To make auction market-driven, many researchers put forward their methods. Most of these methods assume that the intangible hand can push cloud resource market to equilibrium [3] [4] [5] [6].

Based on auction, [5] uses greedy algorithm to set price. Users with front rank of price/requirement value are accepted as more as possible under the condition that request should within total resource limitation and the price is higher than the running cost. However, this method doesn't consider the effect that the changing price has on user's requirements and only focuses on short-term revenue maximization.

To model users' reaction to price, [7] views users' enter rate and exit rate as the Poisson process. Although this model can show users' macroscopic reaction to price, it can not model the difference of consumption behaviour between different kinds of users.

Another shortcoming of [5] is cloud provider's cost hasn't been take into consideration. To address this problem, [8] formulate the allocation problem as a multidimensional knapsack problem extended with reserve price constraint.

Data mining approaches are also used for pricing [9], but it only proposed an framework without clear details and only analysis it in theory with some game theories [10].

The mixed game theory has also been utilized to develop new pricing schemes [9] [11] [12]. They use cooperative game theory to optimize profits for providers, and then

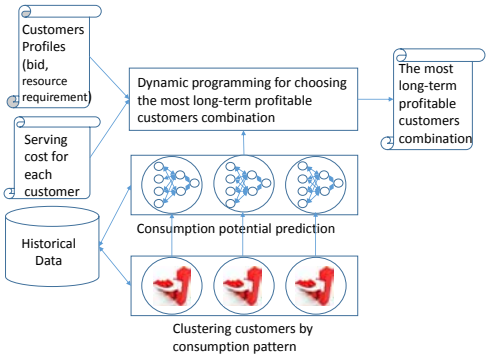
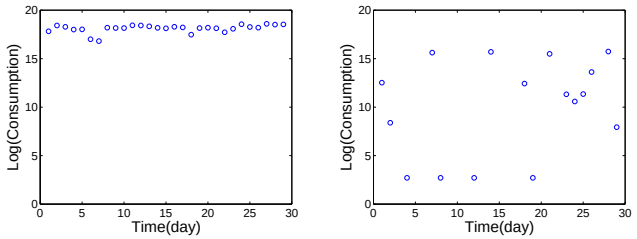


Figure 1. Architecture of the proposed method.



(a) User 1 (b) User 2

Figure 2. Illustration of two users' usage trace of resource.

use competitive game theory to balance the profits between users and providers. However, they did not take into account effects that users with different hardware requirements have on total profits of providers.

B. Our Work

To solve the issues above, we absorb some technologies from previous researches, including applying dynamic programming to solving complexity of computation ability of difference of users, adding cost on serving each user to avoid loss of cloud providers during dynamic programming. Different from existing methods, after exploring importance of difference kinds of users in section II and considering quality of service's influence on users' willingness to continue consumption, we decide to give users who are more important for long-term revenue (i.e. users that consume a lot) some discount when their bids are slightly lower than transaction price and make sure that they are served to keep using a specific cloud provider's service ,thus stabilizing and maximizing the provider's revenue.

To achieve our goal, firstly we divide cloud users into several groups according to their consumption behaviour by using a revised agglomerative hierarchical clustering algorithm where a cluster number estimate method is designed to automatically select the the most suitable number of clusters in section III. Then, a consumption potential prediction model is built for each group of users in section IV by using neural networks. Finally an auction process is carried out with dynamic programming by considering bid of users and their consumption potential prediction results in section V. Architecture of the proposed method is shown in Figure 1.

Experiments in section VI have been conducted to compare the proposed method with the existing auction methods, and experiment results show that the proposed method can choose users in a more reasonable way and gain larger revenue in the long run.

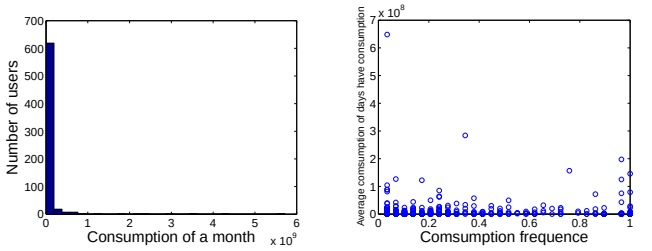
II. User Analysis

To find the heterogeneity of cloud users, we conduct an analysis of usage trace of a Google computer cell consisting of approximately 12,000 machines [13]. The usage trace contains scheduling events, resource demand and usage records of 2,012,242 jobs and 144,648,288 tasks used by 993 users over 29 days. Specifically, a job comprises of one or more tasks, each of which is accompanied by a set of resource requirements used for scheduling (packing) the tasks onto machines. Each task represents a Linux program, possibly consisting of multiple processes, to run on a single machine. Resource requirements for a task include its requirement for CPU cores, RAM and local disk space. The values are normalized to between 0 to 1 according to the largest capacity of the resource on any machine in the trace (which is 1.0). All of these requirements are specified by users when a task is submitted [14].

We collect everyday resource usage of each user as follows. First, we collect all user names from the job event table. Then, for each day, we collect all jobs' id of jobs submitted by each user from the job event table. After finding all jobs of a specific user, we find all tasks belonging to this user. Finally, we sum up resource usage of each task to get total usage of every user for each day. Since we have 3 kinds of resource usage, we sum up all resource with their relative unit price to make the result more visible.

After getting usage trace of all users, we randomly choose two users to analyse their usage of this month. Their usage traces are shown in Figure 2(a) and Figure 2(b) respectively. The two figures show that usages of different users are different. User 1 represents users that has low usage frequency and low amount of usage, while user 2 represents whose who use resources quite frequently and a lot each time.

Considering maximizing the long-term profits, user 2 is far more important than user 1. So, in the auction process, even in the cases where user 2 pay slight less for the same computing power than user 1, we should choose to serve user 2 rather than user 1. So, in our dynamic pricing method, we estimate users consumption potential and add it to users maximal bid price in auction process to serve more important users when two users bid for almost same price and the price



(a) Distribution of users' monthly consumption.

(b) Distribution of users' usage pattern.

Figure 3. Distribution of users.

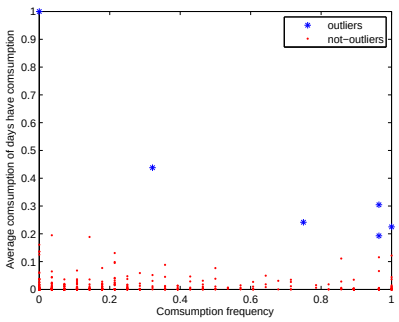


Figure 4. Outliers detection result.

is close to lower bound of average transaction price.

Further statistic shows that 36.7% of the users contribute up to 72% revenue for cloud providers, which can be view as an 80-20 rule [15] for cloud computing. The distribution of users comes out at Figure 3(a), from which we can find that high consumption users are rare and serving them well can greatly affect total income for cloud providers.

The most significant differences between users' usage pattern are the usage ratio and total cost. To get a whole view, we make a statistics on usage ratio and total cost for every user in this month. The result is shown in Figure 3(b). The horizontal axis correspondents to the consuming frequency and the vertical axis represents average consumption of days when a specific user does consume.

We can find in Figure 3(b) that except some outliers, users can be separated into some clusters in terms of their similar behaviour. Consumption behaviour between users in different clusters can have large difference. So, clustering users first and predicting users consumption potential based on the cluster where a specific user belongs to can improve prediction precision.

III. User Clustering

Because the distribution of users' using behaviour does not follow Gaussian distribution and the number of clusters is always changing due to the variation of the market states, the traditional clustering algorithms like k-means [16] and BDSCAN [17] fail to carry out a good clustering result. After trying a lot of clustering algorithms, we find that agglomerating hierarchical clustering algorithm can bring out reasonable clustering result in this scenario after introducing our algorithm for deciding the most suitable number of clusters.

A. Agglomerating hierarchical clustering

The agglomerating hierarchical clustering works as follows:

- Start with the points as individual clusters.
- At each step, we merge the closest pair of clusters.

The distance measurement we use for agglomerating hierarchical clustering algorithm is inner squared distance.

Merge process will continue until only one cluster left. And the best clusters number choosing algorithm is proposed in section III-C after eliminating outliers by the outliers detection algorithm proposed in section III-B.

B. Data preprocessing to eliminate outliers

Since outliers can easily introduce unnecessary spurious clusters, we detect and eliminate them before choosing the best cluster number.

The way we detect outliers is based on observation that if we over split data points (i.e. clustering data into far more clusters that data should be clustered to), clusters that contain only outliers will appear to have only a small number of points.

Our method for eliminating outliers works as follows:

- Firstly, cluster data points into the clusters upper bound which is far more larger than reasonable number of clusters.
- Secondly, count the number of points in each clusters. If there are too few points, i.e., less than a threshold:

$$\alpha * \frac{Num\ of\ points}{clusters\ upper\ bound}$$

which means non-spurious clusters should have more points than a parameter α times average points a cluster should has, we mark them as spurious clusters and the points in them are detected as outliers.

The result of applying this method to detecting outliers for user usage is shown at Figure 4 with α set to 0.1 and clusters upper bound set to 20, for which points marked with blue star are outliers.

For outliers, we redistribute them to give them reasonable labels according to the cluster center they are nearest to by applying a most appropriate prediction model to predict their consumption potential in section IV.

C. Choosing the best clusters number

Our algorithm for choosing the best clusters number is based on zoomed average sum-of-error's changing trend with the number of clusters.

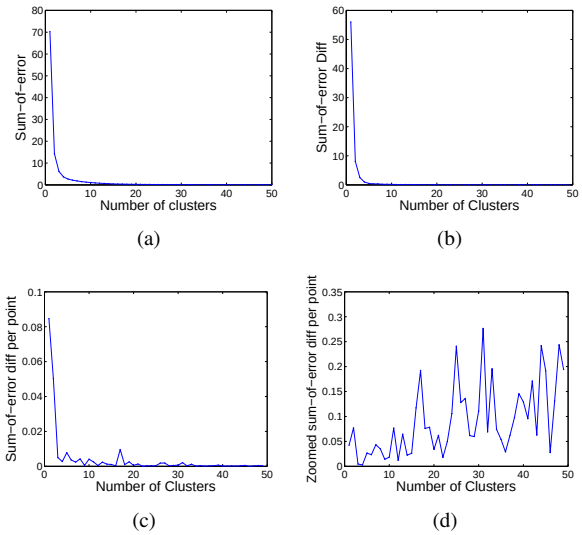


Figure 5. Illustration of the best cluster number determination method: (a) Sum-of-errors for each number of cluster; (b) Difference of sum-of-errors between neighbour cluster number; (c) Difference per point of sum-of-errors between neighbour cluster number; (d) Zoomed difference per point of sum-of-errors between neighbour cluster number.

For agglomerating hierarchical clustering algorithm, sum-of-error reduces with increasing of cluster number. Their relationship of google-cluster users' usage trace is shown in Figure 5(a). In Figure 5(b) the difference between a pair of sum-of-error neighbour also reduces with increasing of cluster number because agglomerating hierarchical clustering algorithm merge the nearest clusters each time.

But after dividing difference between neighbour sum-of-error with number of points within the cluster we merged, as shown in Figure 5(c) ,we can find neighbours of them are not monotonous. Detecting outliers by this value is firstly proposed in [18] and [19] uses this method in their clustering method for choosing the best clusters number for hierarchical clustering.

But, we observed that, with decreasing of the range of value in smaller clusters and existing of spurious clusters, we should zoom the result with range of newly merged cluster to reduce influence caused by decreasing range. The result is shown in Figure 5(d). Observing it, we can find that when clusters number rises from one to two and from two to three the zoomed sum-of-error difference can be reduced greatly. But when the cluster number rises from three to four and from four to five, reduction is far more less than previous two merges. So we should divide users into three clusters.

To write it formally, choosing the best number of clusters to cluster data into work as follows:

- Firstly, calculate sum-of-error for different number of clusters with standard agglomerating hierarchical clustering algorithm. The range of cluster number is chosen from one to the specified clusters number upper bound.
- Secondly, calculate difference between neighbour pairs of sum-of-error.
- Thirdly, divide difference between neighbour pairs of sum-of-error drawn from previous step by number of points in the newly merged clusters.
- Fourthly, divide result obtained in previous step by range of values of data in newly merged cluster.
- Finally, find the first jump point between high-efficient reduction zoomed sum-of-error per point calculated at previous step with low-efficient reduction points. The last point in high-efficient reduction part is the last reasonable merge we should make, and the clusters number corresponding to this point is the best clusters number we choose to merge data points to.

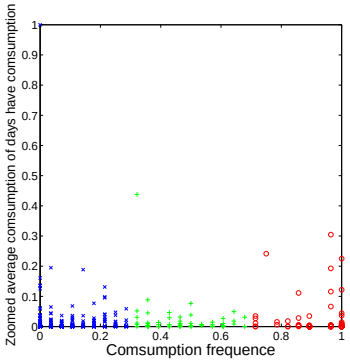


Figure 6. Users clustering result.

After applying our revised agglomerating hierarchical clustering algorithm to cluster all users according to their usage, clustering result is shown in Figure 6, in which three clusters are generated automatically.

In our users' usage prediction model, we will use this revised algorithm to cluster users into difference clusters, and users in the same cluster are treated as the same type. Then, each user's future consumption potential will be predicted by the prediction model it belongs to. This step will be introduced in the following section.

IV. User Consumption Potential Evaluation and Prediction Model

In order to evaluate a certain type of users' consumption potential during a week in the future, we build neural networks to do prediction. To improve precision of prediction, we build a neural network for each type of users.

A. Input features

Input for a specific neutral network is a user's consumption frequency and average consumption of each hour when user does consume in the previous four weeks.

Formulas for calculating input features of each week in the previous four weeks are introduced as follows.

1) Consumption frequency: According to the actual situation of the Amazon's dynamic auction, we use $t = 1hour$ as the basic data sample time interval. So, for a specific user, his weekly consumption frequency can be calculated by

$$ConsumptionFrequncy = \frac{num\ of\ used\ hours}{hours\ a\ week}$$

2) Average consumption of each hour when user does consume: This feature can be generated by dividing total consumption by the number of hours have been consumed.

This two features will be generated for all previous four weeks and all these eight features make up our input vector for a specific user to predict his consumption potential.

B. Output feature

Our consumption potential prediction model has only one output, a user's total potential resource consumption in the following week.

C. Training

To avoid random noise of short-term usage, we choose to update neural networks once a week. We use a sliding window to find previous four weeks of each week in previous two months to generate input features and actual usage of its following week as output feature to training neural networks.

D. Testing

Prediction result of our neural network is evaluated by comparing prediction result with actual consumption of each specify users' usage of next week.

E. Inheritance and Evolution

The number of users' clusters changes over time. To make neural network predictors' quantity agree with number of clusters, we introduce following ways to handle the evolution of neural networks:

- Calculate each cluster's center point's distance to clusters in preview auction
- If number of clusters for current run is less than or equals to previous auction, directly inherit training result of neural network belong to preview cluster.
- If number of clusters for current run is larger than previous auction, we firstly inherent the most similiar neural network from previous training result. Then we train these neural network. The number of newly added neural network equals to the number of newly added clusters.

After training our neural network periodically, we can use it to predict consumption potential for every users participating in an auction process. The consumption potential for each user will be used as an important factor for choosing the best combination of users to serve in the following section.

V. Dynamic Programming For Maximized-Revenue

To get the user combination which can produce the optimal long-term revenue, we apply dynamic programming algorithm to do auction.

Long-term revenue of serving a specify users is made up of three parts:

$$Revenue_{user}(t) = \beta * Potential(t) + Bid(t) - Cost(t)$$

where

β is a parameter, which represents discount ratio we decide to afford to a user according to his consumption potential.

$Potential(t)$ stands for a user's consumption potential.

$Bid(t)$ stands for a user' bid.

$Cost(t)$ stand for cost of serving a user.

So, the optimal target for the whole system is to choose the best users combination to maximize overall revenue with limited resources.

To use dynamic programming to solve this optimal problem, firstly let's bring out some definitions. Define that the number of users is U, the number of total resource is N, users' bid list is $B[]$, the resource request of each user is in list $R[]$, users' consumption potential list is $P[]$ and cost for serving each user is $C[]$. Dynamic programming is used to calculate the maximum revenue $dp[u][r]$, which represents the maximum revenue of only first u users with resource requirement of r participating in auction.

Initially, none of users participate in auction, so the maximal revenue of zero users is 0.

$$dp[0][1:r]=0; \tag{1}$$

Then, we keep adding one more users that participating in auction until all of users are added. After each user participate in auction, the maximal revenue can be evaluated by following formula:

$$dp[u+1][r] = \begin{cases} dp[u][r] & \text{If } r \le R[u] \\ \max(dp[u][r], dp[u][r-R[u]] + B[u] + \beta * P[u] - C[u]) & \text{Otherwise} \end{cases} \tag{2}$$

In the first case, after adding a new user, if his resources demand is larger than residual resources of the whole system, we can not serve him and the optimal revenue of serving users containing him is equal to serving users without him taking part in auction.

For the second case, if resource requirement of a new user can be satisfied, then the maximal revenue after adding this user to users participating in auction is the larger one between the maximal revenue of without serving the user and serving the user.

Users' choosing result is saved in $A = [u][r][j]$, which means to maximize revenue when the first u users with r

resources requirement come for auction, user j should be served if $A[u][r][j] = 1$.

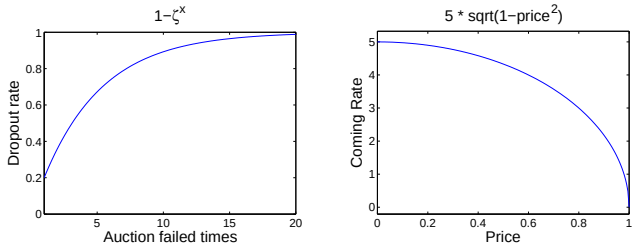
Overall dynamic programming algorithm for maximizing long-term revenue is summarized in Algorithm 1.

Algorithm 1 Dynamic Programming For Maximized-Revenue

Input:
Total User Number U
Total Resource Number N
Users' maximum bid list $B = []$
Users' consumption potential list $P = []$
Discount ratio β
Users' serving cost list $C = []$
Users' resource requirement list $R = []$
Auction result list $A = [][][]$
for $r = 0 : N$ **do**
  dp[0][r] = 0;
  for $u = 0 : U$ **do**
    A[0][r][u] = FALSE;
  end for
end for
for $u = 0 : U$ **do**
  for $n = 0 : N$ **do**
    if $n < R[u]$ **then**
      dp[u+1][n] = dp[u][n];
      for $k = 0 : U$ **do**
        A[u+1][n][k] = A[u][n][k];
      end for
    else if dp[u][n] > dp[u][n-R[u]]+B[u]+β*P[u]-C[u] **then**
      dp[u+1][n] = dp[u][n];
      for $k = 0 : U$ **do**
        A[u+1][n][k] = A[u][n][k];
      end for
    else
      dp[u+1][n] = dp[u][n-R[u]]+B[u]+β*P[u]-C[u];
      for $k = 0 : U$ **do**
        A[u+1][n][k] = A[u][n-R[u]][k];
      end for
      A[u+1][n][u] = TRUE;
    end if
  end for
end for
Output:
Maximized-Revenue is $dp[U][N]$
Auction result Record is $A[U][N]$

VI. Experiments

In this section, experiments are conducted to show the effectiveness of the proposed method. We compare the proposed method with a state-of-the-art auction method — view cloud allocation as a multidimensional knapsack problem extended with an additional (reserve price) constraint. Their performances are evaluated based on total revenue during a long time.



(a) Users' departure rate (b) Users' coming rate.

Figure 7. Users changing hypotheses.

A. Experiment Procedure

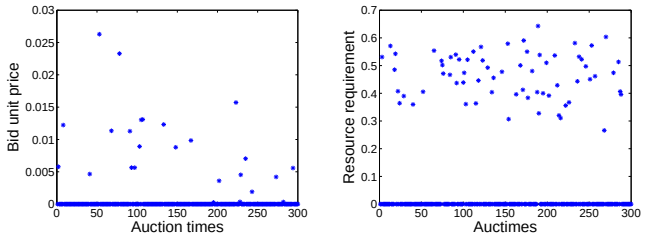
Google-cluster data provides the consumption record of 993 users in a month. But unfortunately they use a static pricing strategy without considering some significant factors we need such as the maximum acceptable price, the interaction between auction result and users' auction decisions. Moreover, the Amazon EC2 does not seem to make the users' data public, though Amazon EC2 uses a dynamical pricing strategy. Therefore, we try to combine the distribution characteristics of google-cluster data and four hypotheses to generate more reliable synthetic data.

The four hypotheses are listed as follows.

- Hypothesis 1: A certain user's bid, resource requirement and auction frequency follow Gaussian distribution $N(\mu, \sigma^2)$. Parameters μ and initial σ of Gaussian distribution for each user are decided based on the users group a specific user belongs to.
- Hypothesis 2: If a certain user fails current auction, in the next auction the variance of his bid will diverge. On the contrary, if a certain user succeeds, in the next auction his bid will converge.Figure 8(a) is an example of how a user's unit resource bid changes with auction times.
- Hypothesis 3: we model users' departure rate as below

$$DepartureRate = 1 - \zeta^2$$

  where ζ is parameter set different according to whether a user win a bid process. For individual users, after observation, we find that users departure rate is higher when they fail in auction. So, we set departure rate of users who succeed in auction to 0.01% and users who fail in auction to 0.2%. Figure 7(a) illustrates user's departure rate when he continuously fails in auction.
- Hypothesis 4: To model new users' arrival, we assume that new users' arrival rate is increasing with decreasing transaction price. And the curve of this model is shown at Figure 7(b).



(a) Variation of a user's unit resource bid.

(b) Variation of a user's resource request.

Figure 8. Variation of a user's bid info.

With those four hypotheses, we develop an on-line data generation algorithm.

1) Users initialization: To initial users' coming at the first auction, we follow these steps:

a) According to data distribution characteristics of google-cluster data, we initialize the type number of users to K by calculating the optimal number of users' types via hierarchical clustering algorithm. Then we calculate the consumption frequency and mean consumption quantity of every interval respectively.

b) We'll generate K groups data of the first interval respectively, where every group contains the following data:

- Maximum willing deal unit price
- Consumption quantity
- The probability of attending auction
- Deal price generated by dynamic programming
- The first deal record containing deal price and maximum willing total amount

The distributions of the first three values for each type of users are different, which can be obtained according to the consumption pattern of each type of users.

During early auction intervals biding trace is not long enough for training neural networks for prediction, so we only simply use greedy algorithm with dynamic programming to do auction and record these auction histories.

After logging enough data, we can train neural networks and use them to predict consumption potential of each user and do long-term revenue maximal auction.

2) Part one: For the first five weeks: We only apply simple greedy algorithm with dynamic programming to do auction process.

After auction process, we update the four parameters listed as follows

- Whether or not a user can purchase the product successfully determines the variances change of the three Gaussian distributions which can be used to generate maximum willing unit price.
- Generate consumption quantity randomly according to Gaussian distribution.
- Generate the probability of consuming randomly according to range of the type user.
- Calculate deal price by dynamic programming and generate deal record.

After updating, all of these dates are logged for clustering users and training neural networks for auctions after the fifth weeks.

In this way, we can generate the first five weeks' data used to build neural network at the first time.

3) Part two: After the fifth week: After getting enough data for training neural networks, we can apply our whole algorithm to do auction process.

When obtaining one week data, we are supposed to cluster again via agglomerating hierarchical clustering algorithm. For each user, if his type in each week during five weeks doesn't meet the requirement that the number of a certain type is more than all the other types, we ignore this data record, i.e. abandon it.

To train neural networks, we use the first four nearest weeks' data as the input of neural network and use the fifth weeks data as the output to train neural network.

Then, we predict all users' consumption potential with newly trained neural networks.

Finally, we use dynamic programming to choose the most profitable users combinations, and auction results are logged for following runs.

4) Users' usage and max willing offered price generation: Every interval data is generated according to the last interval data with following steps:

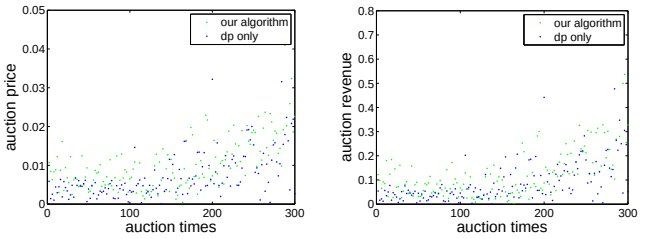
- Determine the Gaussian parameters change which can be used to generate maximum willing unit price according to whether a user can purchase the product successfully .
- Randomly generate consumption quantity according to Gaussian distribution.
- Randomly generate the probability of consuming according to consumption range of the type user.
- Calculate deal price by adding consumption potential P with maximal willing price for each user via dynamic programming and generate deal record.

5) User departure: Any user can depart at any time. We model this by using hypothesis 3 and we remove all exiting users before the coming of new auction.

6) User arrival: New users arrive before auction take place. We use hypothesis 4 to model their arrival rate. New users are chosen according to the distribution of users who take part in auction for google-cluster i.e. each time we add a user whose consumption pattern is produced by choosing one user's pattern from google-cluster' customers randomly.

B. Experiment Evaluation

To evaluate our method, we compare our auction method versus the method brought out by Sergei *et al.*[8]. They view cloud allocation problem as a multidimensional knapsack problem extended with an additional constraint (reserve price) and use dynamic algorithm to solve it.



(a) Transaction Price Comparison (b) Transaction Revenue Comparison

Figure 9. Experiment result.

Initially, we make 400 people participate in auction and the consumption behaviour of them are decided according to users' distribution we obtain in section II.

These two auction methods run for 300 auction intervals, and we compare transaction price and transaction revenue of cloud providers between these two methods.

C. Experiment Result

Transaction price and transaction revenue graphs of applying two method for auction are shown in Figure 9(a) and Figure 9(b).

Transaction price graph Figure 9(a) shows that compared with existing method, transaction price generated by our method is more stable than the one conducted by other method, which makes it more convenient for users to auction with our method compared with using other method. In addition, transaction price of our method is higher than transaction price conducted by other methods and make cloud providers receive more unit profit.

Transaction revenue graph Figure 9(b) shows that after applying our method, cloud providers can generally gain higher total revenue compared with existing method, which make cloud providing more profitable.

After comparing on these two most important indexes for cloud pricing method in cloud providers' view, our method is far more better than existing method and can be used as a dynamic pricing guideline for cloud providers.

VII. Conclusion

In this paper, we have presented a long-term revenue maximization algorithm to provide a more reasonable auction method for IaaS cloud provider. Firstly we partition users to different groups based on their consumption behaviour. Then we predict each user's consumption potential by constructing a neural network based predictor for each type of users. Finally, dynamic programming is utilized to choose best users to serve with considering both bid price and consumption potential of each user. Experimental results have confirmed the effectiveness of the proposed method.

References

[1] (2015) Amazon EC2 spot instances. [Online]. Available: http://aws.amazon.com/ec2/purchasing-options/spot-instances/

[2] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 spot instance pricing," in *Cloud Computing Technology and Science (CloudCom)*, 2011, pp. 304–311.

[3] U. Lampe, M. Siebenhaar, A. Papageorgiou, D. Schuller, and R. Steinmetz, "Maximizing cloud provider profit from equilibrium price auctions," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 83–90.

[4] P. Bonacquisto, G. D. Modica, G. Petralia, and O. Tomarchio, "A strategy to optimize resource allocation in auction-based cloud markets," in *2014 IEEE International Conference on Services Computing*, 2014, pp. 339–346.

[5] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds," *IEEE Transactions On Cloud Computing*, vol. 1, no. 2, pp. 129–141, 2013.

[6] "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM*, pp. 1–26, 2002.

[7] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," in *Proc. of the 3th IEEE Transactions On Cloud Computing*, 2013, pp. 158–171.

[8] S. Chichin, Q. B. Vo, and R. Kowalczyk, "Adaptive market mechanism for efficient cloud services trading," in *2014 IEEE International Conference on Cloud Computing*, 2014, pp. 705–712.

[9] W.-T. Tsai and G. Qi, "DICB: Dynamic intelligent customizable benign pricing strategy for cloud computing," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 654–661.

[10] P. K. Dutta, *Strategies and Games:Theory and Practice*. MIT Press, 1999.

[11] P. Xiao and Z.-G. Hu, "Hybrid game based virtual resource pricing model in cloud environment," *Computer Integrated Manufacturing Systems*, vol. 20, no. 1, 2014.

[12] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010, pp. 513–517.

[13] (2014.) Googleclusterdata - traces of google workloads. [Online]. Available: http://code. google.com/p/googleclusterdata/

[14] C. Reiss, J. Wilkes, and J. Hellerstein, *Google cluster-usage traces: format+schema*, version of 2013-05-06 ed.

[15] J. M. Juran, *Quality Control handbook*. McGraw-Hill New York, 1951.

[16] J. M. Queen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the 5th Berkeley Symposium on Mathematics, Statistics, and Probabilities*, 1967, pp. 281–297.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[18] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," in *Pattern Recognition Letters*, 2003, pp. 1641–1650.

[19] C. Zhong, D. Miao, R. Wang, and X. Zhou, "DIVFRP: An automatic divisive hierarchical clustering method based on the furthest reference points," *Pattern Recognition Letters*, pp. 2067–2077, 2008.