A DYNAMIC RESOURCE ALLOCATION IN CLOUD SERVICES THROUGH INTELLIGENT COMBINATORIAL DOUBLE AUCTION APPROACH

Karthikeyan S1 | Dr.Manimegalai R2 | Dr.Hamsapriya T3

1 (Dept. of CSE, Assistant Professor, PSG Institute of Tech and Applied Research, Coimbatore; karthikeyan@psgitech.ac.in)
2 (Dept. of CSE, Professor, Park College of Engineering and Technology, Coimbatore, mmegalai@yahoo.com)
3 (Dept. of CSE, Professor, PSG Institute of Technology and Applied Research, Coimbatore, hamsapriya.t@gmail.com)

Abstract—Nowadays Cloud Computing is emerging as a new paradigm shift where resources are requested on demand and in a very dynamic fashion and pay as you go model. This new cloud change created an ecosystem where several providers offer heterogeneous computing resources to satisfy the customer’s computing demand. So, the demand allocation and adaptation to be the key issue in this ecosystem, because it can deliver the mutual profit for the customers and providers. However, for more cloud users, this allocation is tough task. This work is proposed with the intention of solving this allocation issue, using combinatorial double auction method through Back Propagation Neural Network (BPNN), which encourages the active participation of multiple consumers and multiple providers, to bid for resources and to determine the winners.

Keywords— Cloud; Resource Allocation; Combinatorial Double Auction; Pricing; Activation Function

1. INTRODUCTION
The elastic and on-demand nature of cloud computing assists cloud users to meet their dynamic and fluctuating demands with minimal management overhead, while the cloud ecosystem as a whole achieves economies of scale through cost amortization. Currently, most cloud providers adopt a fixed price policy and charge users a fixed amount as per their usage. Despite their apparent simplicity, fixed-price policies inherently lack market agility and efficiency, failing to rapidly adapt to real-time demand-supply relation changes. Consequently, overpricing and underpricing routinely occur, which either dispel or undercharge the users, jeopardizing overall system social welfare as well as the provider’s revenue. In the cloud computing, idle resources can be integrated and allocated to users in the form of service. A resource allocation mechanism is in need to effectively allocate resources, motivate users to join the resource pool and avoid fraud among users. This can be achieved using Combinatorial Double Auction method. Auction mechanisms have recently attracted substantial attention as an efficient approach to pricing and resource allocation in cloud computing. Multiple parties may join through conducting joint Back propagation neural network learning on the union of their respective data sets. The winner of the auction is then determined by solving an optimization problem.

2. RELATED WORK
Several auction-based models were proposed for addressing resource allocation in the cloud computing environment. Lin et al. [16] proposed a second-price auction mechanism which applies the marginal bid to determine the price of the resource for computation capacity allocation with the assistance of pricing and truth-telling mechanism, which ensures the reasonable interests of cloud service providers and effective allocation of computing resource. Prodan et al. [17] proposed a negotiation-based approach for scheduling scientific applications on heterogeneous computing infrastructures such as grids and clouds, and presented a negotiation protocol between the scheduler and resource manager using a market-based continuous double auction model to manage the access to resources in an open market. Shang et al. [18] divided the cloud resource trading market into the futures market and the spot market, and then proposed a knowledge-based continuous double auction trade model and introduced the probability agent based on historical trading information to determine the probability that future bids will succeed, which can achieve higher market efficiency and stable transaction price.

The auction models used in the three literatures mentioned above all belong to single-item auction, and with the development of the research on auction-based models, combinatorial auction as a kind of multi-item auction that can deal with the combinatorial requirements of buyers has widely applied to allocate resources in the cloud environment. Zaman et al. [19] formulated the problem of virtual machine allocation in clouds as a combinatorial auction problem and proposed three mechanisms: FIXED-PROCE, CA-LP (Combinatorial Auction - Linear Programming), CA-GREEDY (Combinatorial Auction - Greedy) to solve it, and the experimental results showed that CA-GREEDY is better for general purpose VM instance allocation problem while CA-LP can be served for special scenarios. Fujiwara et al. [20] proposed a market-based resource allocation mechanism that allows
participants to trade services by means of a double-sided combinational auction. This mechanism enables users to order a combination of services for workflows and co-allocations and providers to reserve future/current services in a forward/spot market, which is a little similar to [18], and the two markets run independently to make predictable and flexible allocations at the same time. Actually, the auction-based model was used as an allocating method in the grid computing earlier and then applied in the cloud environment. Furthermore, auction-based resources allocation has been a hot topic in grid literature for a decade, so many research works in grid computing have great reference value and can be used in cloud computing. Liang et al. [21] proposed a resource allocation model based on reverse auction to allocate grid resources, which can satisfy user’s QoS demand on deadline and budget and have better performance than a commodity market-based allocation model. Grosu et al. [22] proposed and investigated first-price auction protocol, Vickrey auction protocol and double auction protocol, and they found the double auction protocol favors both users and resources. Das et al. [23] proposed a resource allocation agreement based on combinatorial auction, in which users bid for every combination of resources and use approximation algorithm to solve the auction problem. Schnitzler et al. [24] improved the combinatorial auction, proposed multi-attribute combinatorial auction and the effectiveness of mechanism is proved from the aspects of economy, computing performance and practicality.

However, most of the works rarely take fraud behaviors of malicious users into account and lack the corresponding punishments. In addition, they also rarely take the comprehensive aspects from buyer, seller and market into consideration. In our opinion, the allocation mechanism should be efficient to market and be convenient and fair to the buyer and seller. So we proposed BPNN (Backpropagation Neural Network) based on the combinatorial double auction.

3. PROBLEM FORMULATION

The traditional pricing method (i.e.) fixed charge for demanded resources is being replaced in the proposed system by throwing the light on the new resource allocation technique, online auction.

Multiple providers, who are willing to allocate their resources and multiple consumers, who are willing to demand resources take part in this auction. Participating consumers pay an initial minimum amount as prescribed by the providers. This in turn is delivered as a benefit for both providers and the winning consumer. Remaining consumers join a new auction and continue bidding. Providers create individual accounts with their unique icons, clod details etc., thus revealing themselves to the consumers.

On the other hand, consumer’s data are not revealed to others, thus providing more privacy to the consumers.

Separate logins for providers and consumers, blogs for consumers, upload provider’s details etc., ease the auction process further. Pricing, bidding and winner are determined using Backpropagation Neural Network(BPNN) algorithm.

Algorithm:(BPNN)

Input: SDR, BUD, REP, TF, sample-base, Label (indicating whether this is the first call to Algorithm1, 1 means yes, 0 means no)

Output: BPoDS.

1: Set MNoS be the required minimum number of samples in sample-base to train BPNN;
2: Set N be the number of samples recorded in sample-base;
3: if N<MNoS then
4:     if Label==1 then
5:         Set BPoDS be a random number between 0 and BUD;
6:     else
7:         Get ΔBP randomly from the uniform distribution within the interval [ 0, BUD-BPoDS]; /*ΔBP is the bidding price adjustment amplitude. */
8:         BPoDS= BPoDS + ΔBP;
9:     end if
10: else
11:     if Label==0 then
12:         Update SDR by (4);
13:     end if
14:     if ΔBP==1 then
15:         Set BPoDS be a random number between 0 and BUD;
16:     else
17:         Get ABP randomly from the uniform distribution within the interval [ 0, BUD-BPoDS]; /*ABP is the bidding price adjustment amplitude. */
18:         BPoDS= BPoDS + ΔBP;
19:     end if
20: else
21:     if Label==0 then
22:         Update SDR by (4);
23:     end if
24:     for j ϵ {1,2,3,4} do
25:         if ΔBP==1 then
26:             Set BPoDS be a random number between 0 and BUD;
27:         else
28:             Get ABP randomly from the uniform distribution within the interval [ 0, BUD-BPoDS]; /*ABP is the bidding price adjustment amplitude. */
29:             BPoDS= BPoDS + ΔBP;
30:         end if
31:     end for
4. BACKPROPAGATION NEURAL NETWORK LEARNING

Backpropagation works far faster than earlier approaches to learning, making it possible to use neural nets to solve problems which had previously been insoluble. Today, the backpropagation algorithm is the workhorse of learning in neural networks. The backpropagation algorithm is a clever way of keeping track of small perturbations to the weights (and biases) as they propagate through the network, reach the output, and then affect the cost.

Complex mappings between input and target patterns could be learnt in an elegant and practical way by non-linear connectionist network. It also overcomes many limitations associated with neural network learning algorithm of the previous generation, such as perception algorithm. At the same time, it includes the basic ingredients of the general connectionist recipe: local computations, global optimisations and parallel operations. But most interestingly it showed that input-output mappings could be generated during learning by the discovery of internal representations of the training data. These representations were sometimes clever, non-trivial and not originally intended or even imagined by the human designer of the back propagation architecture network. The internal representations learned by the back propagation algorithms had an “intelligent flavour” that was difficult for artificial intelligence researchers to ignore. Altogether these features contributed to the success of back propagation as a versatile tool for computer modellers, Engineers and cognitive scientists in general.

A. Architecture

It is made up from an input, an output and one or more hidden layers. Each node from input layer is connected to a node from hidden layer and every node from hidden layer is connected to a node in output layer. There is usually some weight associated with every connection. Input layer represents the raw information that is fed into the network. This part of network is never changing its values. Every single input to the network is duplicated and send down to the nodes in hidden layer. Hidden Layer accepts data from the input layer. It uses input values and modifies them using some weight value, this new value is than send to the output layer but it will also be modified by some weight from connection between hidden and output layer. Output layer process information received from the hidden layer and produces this output. This output is than processed by activation function.

B. Setting Weights

The way to control NN is by setting and adjusting weights between nodes. Initial weights are usually set at some random numbers and then they are adjusted during NN training. According to Fogel [2002] focus should not be at changing one weight at time, changing all the weights should be attempted simultaneously. Some NN are dealing with thousands, even millions of nodes so changing one or two at time would not help in adjusting NN to get desired results in timely manner. Logic behind weight updates is quite simple. During the NN training weights are updated after iterations. If results of NN after weights updates are better than previous set of weights, the new values of weights are kept and iteration goes on. Finding combination of weights that will help us minimize error should be the main aim when setting weights. This will become bit more clear once the learning rate, momentum and training set are explained.

C. Training the Neural Network

Running the network consist of a forward pass and a backward pass. In the forward pass outputs are calculated and compared with desired outputs. Error from desired and actual output is calculated. In the backward pass this error

---

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Name</th>
<th>Abbreviation</th>
<th>Full Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
<td>CEC</td>
<td>Cloud Service Consumer</td>
</tr>
<tr>
<td>PA</td>
<td>Provider Agent</td>
<td>CA</td>
<td>Consumer Agent</td>
</tr>
<tr>
<td>AI</td>
<td>Attention Interception</td>
<td>AI</td>
<td>Consumer Identifier</td>
</tr>
<tr>
<td>DS</td>
<td>Demand Service</td>
<td>BPD</td>
<td>Billing Price of DS</td>
</tr>
<tr>
<td>ToDS</td>
<td>Type of DS</td>
<td>SToDS</td>
<td>Starting Time of DS</td>
</tr>
<tr>
<td>EToDS</td>
<td>Ending Time of DS</td>
<td>CPuToDS</td>
<td>CPU Speed of DS</td>
</tr>
<tr>
<td>MI</td>
<td>Million Instructions</td>
<td>MIps</td>
<td>MF Per Second</td>
</tr>
<tr>
<td>MEM</td>
<td>MEMory</td>
<td>MEMDCS</td>
<td>MEM Capacity of DS</td>
</tr>
<tr>
<td>CB</td>
<td>Cega Bytes</td>
<td>STO</td>
<td>STO Usage</td>
</tr>
<tr>
<td>SToDCS</td>
<td>STO Capacity of DS</td>
<td>NETB</td>
<td>Network Bandwidth</td>
</tr>
<tr>
<td>NEDoT</td>
<td>NET of DS</td>
<td>TS</td>
<td>Task Size</td>
</tr>
<tr>
<td>DV</td>
<td>Data Volume</td>
<td>MYP</td>
<td>Maximum Partition</td>
</tr>
<tr>
<td>PlSoDS</td>
<td>Platform and</td>
<td>MRIPCSP</td>
<td>the required Minimum</td>
</tr>
<tr>
<td></td>
<td>Software of DS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>Provider Identifier</td>
<td>SS</td>
<td>Surprised Service</td>
</tr>
<tr>
<td>PoCPuToS</td>
<td>Price of CPU of SS</td>
<td>CPuESS</td>
<td>CPU Speed of SS</td>
</tr>
<tr>
<td>PoMEMC5S</td>
<td>Price of MEM of SS</td>
<td>MEMC5S</td>
<td>MEM Capacity of SS</td>
</tr>
<tr>
<td>PoSToCSS</td>
<td>Price of STO of SS</td>
<td>STOCCS</td>
<td>STO Capacity of SS</td>
</tr>
<tr>
<td>PoNETBSS</td>
<td>Price of NETB of SS</td>
<td>NETBSS</td>
<td>NETB of SS</td>
</tr>
<tr>
<td>SToE8S</td>
<td>Starting Time of SS</td>
<td>ETR8S</td>
<td>Ending Time of SS</td>
</tr>
<tr>
<td>PlSoCSS</td>
<td>Platform and</td>
<td>MRIPFCSC</td>
<td>the required Minimum</td>
</tr>
<tr>
<td></td>
<td>Software of DS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDR</td>
<td>Supply and Demand Ratio</td>
<td>BUD</td>
<td>BUD=</td>
</tr>
<tr>
<td>REP</td>
<td>REPutation</td>
<td>CFT</td>
<td>CoET</td>
</tr>
<tr>
<td>TF</td>
<td>Time-Frame</td>
<td>ANRJU</td>
<td>Asking Price of the</td>
</tr>
<tr>
<td>TJS</td>
<td>Total Surplus</td>
<td>TJS</td>
<td>Resource per Unit</td>
</tr>
<tr>
<td>NJS</td>
<td>Total Reputation</td>
<td>MOJ</td>
<td>Multi-Objective Optimization</td>
</tr>
<tr>
<td>SOO</td>
<td>Single-Objective</td>
<td>WSM</td>
<td>Weighted Sum Method</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
<td>SPD</td>
<td>Surprerity Degree</td>
</tr>
<tr>
<td>AHP</td>
<td>Analytic Hierarchy</td>
<td>ISA</td>
<td>Local Search Improvemenet</td>
</tr>
<tr>
<td>Process</td>
<td>Algorithm</td>
<td>CDCA</td>
<td>Stable CDCA</td>
</tr>
<tr>
<td>CDA</td>
<td>Continuous Double Auction</td>
<td>CDCA</td>
<td>Stable CDCA</td>
</tr>
<tr>
<td>TWR</td>
<td>Total Winning Rate</td>
<td>DWR</td>
<td>Dishonest Winning Rate</td>
</tr>
<tr>
<td>CM</td>
<td>Greedy Method</td>
<td>Mbps</td>
<td>Million bits per second</td>
</tr>
</tbody>
</table>

---
is used to alter the weights in the network in order to reduce the size of the error. Forward and backward pass are repeated until the error is low enough (users usually set the value of accepted error). When training NN, we are feeding network with set of examples that have inputs and desired outputs. If we have some set of 1000 samples, we could use 100 of them to train the network and 900 to test our model. Choosing the learning rate and momentum will help with weight adjustment.

Setting right learning rate could be difficult task, if learning rate is too small, algorithm might take long time to converges. On the other hand, choosing large learning rate could have opposite effect, algorithm could diverge. Sometimes in NN every weight has it’s own learning rate. Learning rate of 0.35 proved to be popular choice when training NN.

Larose [2005] claimed that momentum term represents inertia. Large values of momentum term will influence the adjustment in the current weight to move in same direction as previous adjustment.

D. Activation function

According to Faqs.org [2010] activations function are needed for hidden layer of the NN to introduce nonlinearity. Without them NN would be same as plain perceptions. If linear function were used, NN would not be as powerful as they are. Activation function can be linear, threshold or sigmoid function. Sigmoid activation function is usually used for hidden layer because it combines nearly linear behavior, curvilinear behavior and nearly constant behavior depending on the input value Larose [2005]. SUM is collection of the output nodes from hidden layer that have been multiplied by connection weights added to get single number and put through sigmoid function (activation function). Input to sigmoid is any value between negative infinity and positive infinity number while the output can only be a number between 0 and 1.

5. AUCTION PROCESS

Auction process includes three major stages: (1) Price Prediction (2) Price matching (3) Winner determination. However, all these stages rely basically on a single technique called Back Propagation Neural Network (BPNN) algorithm.

A. Price prediction

Input: SDR, BUD, REP, TF, sample-base, Label (indicating whether this is the first call to Algorithm1, 1 means yes, 0 means no)

Output: BPoDS
1: Set MNoS be the required minimum number of samples in sample-base to train BPNN;
2: Set N be the number of samples recorded in sample-base;
3: if N < MNoS then
4: if Label == 1 then
5: Set BPoDS be a random number between 0 and BUD;
6: else
7: Get ΔBP randomly from the uniform distribution within the interval [0, BUD-BPoDS]; /*DBP is the bidding price adjustment amplitude. */
8: BPoDS = BPoDS + ΔBP;
9: end if
10: else
11: if Label== 0 then
12: Update SDR by (4);
13: end if
14: a1= SDR, a2= BUD, a3= REP, a4= TF;
15: for j ϵ {1,2,3,4} do
16: uj= Σi=1N wi aij ;
17: end for
18: for j ϵ {1,2,3,4} do
19: bj= (uj);
20: end for
21: p1= Σj=1N vij bji ;
22: BPoDS ==p1;
23: end if
24: return BPoDS;

B. Price matching

Input: MRN (the maximum round number), TCT (the tender collection time at each round)

Output: ask_pricem×n
1: Set m and n be the number of CSCs and CSPs respectively;
2: Initialize all elements in matrix flagm×n to be 0; /* indicates whether price matching between CSCi and CSCj succeeds, 1 means yes, 0 means no. */
3: k= 1;
4: while (k ≤ MRN) do
5: Collect tenders from CSCs and CSPs until TCT timeouts;
6: for i ϵ {1,2,…,m} do
7: for j ϵ {1, 2, n} do
8: switch (type of CSCi’s tender)
9: case VMS: Calculate ask_priceij by (13); break;
10: case CPS: Calculate ask_priceij by (14); break;
11: case DBS: Calculate ask priceij by (15); break;
12: case STS: Calculate ask priceij by (16);
13: end switch
14: end if
15: return BPoDS;
14: if BPoDSi ≥ ask_priceij then
15: flagij= 1;
16: end if
17: end for
18: end for
19: for i ∈ {1, 2,...,m} do
20: if (all elements of ith row in flagm×n are 0) then
21: Notify the CSCi to re-bid;
22: end if
23: end for
24: for j ∈ {1,2,...,n} do
25: if (all elements of jth column in flagm×n are 0) then
26: Notify the CSPj to re-ask;
27: end if
28: end for
29: k= k + 1;
30: end while
31: return ask_pricem×n;

C. Winner determination

Input: INoS (population size), MNoI (the maximum number of iterations)

Output: OSS (the optimal seed set)
1: Do sowing to generate INoS seeds initially and do seed refinement;
2: Choose one seed randomly as the benchmark;
3: Calculate the SPD of each seed to the benchmark;
4: Set MSPDBT and MSPD* to be the maximum SPD and initialize
OSS with all seeds corresponding to MSPDBT ;
5: i= 1;
6: while i ≤ MNoI do
7: Do seeding;
8: Do pollination;
9: Do dispersion;
10: Improve local search ability;
11: Do seed refinement;
12: Calculate the SPD of each seed to the benchmark;
13: Set MSPDBT ; be the current maximum SPD;
14: if MSPDBT > MSPD* then
15: Replace OSS with all seeds corresponding to MSPDBT ;
16: MSPDBT = MSPD*;
17: end if
18: if MSPDBT == MSPD* then
19: Put all seeds corresponding to MSPDBT into OSS;
20: end if
21: Do selection;
22: i= i + 1;
23: end while
24: Get necessary information from CSC and CSP winners as samples and put them into the corresponding PA’s and CA’s sample-base.
25: return OSS;

6. CONCLUSION

An intelligent combinatorial double auction based dynamic resource allocation approach is proposed for cloud services. The system framework is devised to provide a comprehensive solution. A reputation system is used to suppress dishonest participants. A price formation mechanism is proposed to predict price and determine eligible transaction relationship an intelligent combinatorial double auction based dynamic resource allocation approach is proposed for cloud services. The system framework is devised to provide a comprehensive solution. A reputation system is used to suppress dishonest participants. A price formation mechanism is proposed to predict price and determine eligible transaction relationship. Simulation results validate the effectiveness of our proposed approach and demonstrate its superiority on economic efficiency and trustfulness.

REFERENCES