International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 4 | ISSN : 2456-3307

Multiple Resource Acquisition in Cloud Computing using CABOB Algorithm

B. Vijay
MCA Sri Padmavathi College of Computer Sciences and Technology Tiruchanoor, Andhra Pradesh, India

ABSTRACT

Hybrid cloud may be a composition of two or a lot of clouds (private, community or public) that stay distinct entities however are sure along, providing the advantages of multiple readying models. Hybrid cloud also can mean the flexibility to attach collocation, managed and/or dedicated services with cloud resources. In existing Bastion, a completely unique and economical theme that guarantees knowledge confidentiality not with standing the encryption key is leaked and also the somebody has access to the majority ciphertext blocks. we analyze the safety of Bastion, and that we measure its performance by suggests that of a paradigm implementation. Cloud users submit their necessities, and successively vendors submit bids containing value, QoS and their offered sets of resources. The projected approach is scalable, that is critical providing there area unit an oversized variety of cloud vendors, with a lot of frequently showing. we have a tendency to perform experiments for acquisition value and quantifiability effectuality on the CABOB algorithm using various customary distribution benchmarks like random, uniform, decay and CATS.

Keywords: Hybrid cloud, Bastion, CABOB algorithm, QOS

I. INTRODUCTION

Cloud computing is a data innovation worldview that empowers omnipresent access to shared pools of configurable framework assets and larger amount benefits that can be quickly provisioned with negligible administration exertion, frequently finished the Internet. Distributed computing depends on sharing of assets to accomplish cognizance and economies of scale, like an open utility.

Outsider mists empower associations to center around their center organizations as opposed to exhausting assets on PC foundation and support. Backers take note of that distributed computing enables organizations to keep away from or limit in advance IT foundation costs. Defenders likewise assert that cloud computing enables under takings to get their applications up and running quicker, with enhanced reasonability and less upkeep, and that it empowers IT groups to all the more quickly change assets to take care of fluctuating and flighty demand. Cloud sellers regularly utilize a compensation as-you-go display, which can prompt unforeseen working costs if managers are not acquainted with cloud estimating models. Any default understanding offered by the seller may legally profit the merchant yet not the client, bringing about a befuddle with client prerequisites. Additionally, there regularly is no certain dedication on Service Level Agreements (SLAs). Dynamic valuing is the answer for these sort of issues. Consequently, securing assets from the clients point of view is an essential and fascinating issue. A few issues that are as of now connected with settled valuing are: Most regularly, the agreements in asset acquirement support cloud merchants. There may be examples where the prerequisites of both cloud sellers and cloud clients are befuddled.

SLAs are an essential perspective for big business clients, however it is exceptionally hard to implement SLAs given settled valuing. Dynamic estimating conquers these issues. The use of dynamic valuing in distributed computing is an intriguing yet unexplored zone. Asset acquisition is an imperative test in the present Internet, particularly in huge conveyed frameworks like Grid, cloud, and so on. Asset assignment is an exceptionally dynamic zone of research in Grid. Asset acquisition can be proficient utilizing customary or monetary models. The customary models accept that asset suppliers are non key, while monetary models expect that asset suppliers are objective and astute. In regular strategies, a client pays for the expended benefit. In financial models, a client pays in view of the esteem got from the administration. Consequently monetary models are more proper with regards to cloud computing. The primary quality of financial models is circulating motivations to the members. Yet, there are situations where the members may not act honestly. Subsequently, we accept that cloud merchants are childish and levelheaded. Additionally, the cloud specialist performs turn around barters for the benefit of the cloud client. With expanded interest for cloud assets, particularly for complex assignments requiring various assets, there has been an expanded extension for differences between cloud specialist co-ops and cloud clients. This has brought about insufficient exchanges between the two gatherings, which thusly brings about imperfect utilization of the cloud assets. We propose an asset obtainment approach utilizing combinatorial auctions and instrument configuration, to address these issues.

Auctions are critical components for asset and undertaking distribution in multiagent frameworks. In numerous auctions, a bidder's valuation for a mix of discernable things isn't the total of the individual things' valuations it can be pretty much. Combinatorial auctions (CAs) where bidders can offer on packs of things enable bidders to express complementarity.

In combinatorial auctions, the champ assurance is a non-trifling assignment. In genuine cloud frameworks, there are additionally anticipated that would be an extensive number of cloud sellers. Consequently, conceiving an adaptable answer for performing combinatorial sales in a cloud is non paltry and fascinating. The arrangement of offers are spoken to as tree hubs. The tree hubs are named as either winning or losing. The tree is looked utilizing profundity first hunt. Utilizing heuristics, the commitment of unallocated things are figured. This commitment alongside the income created from offers is utilized to choose whether to incorporate an offer in the arrangement of best arrangements. Before presenting the offers to the CABOB calculation, we play out a preprocessing advance to standardize the offer that is being created by the cloud merchants. By doing this, each offer has whole number esteems related with it for every asset being offered for. In the underlying advance, the arrangement of assets are separated to such an extent that no offer incorporates assets from more than one subset. The victor is resolved independently in every subset to accelerate the inquiry. CABOB utilizes an upper edge on the income the unallocated assets can contribute. On the off chance that the momentum arrangement isn't superior to the ideal arrangement, CABOB prunes the inquiry way. We utilize a direct programming (LP) plan for assessing the upper limit. In the wake of evaluating the upper edge, we apply a whole number unwinding where we can either acknowledge the offer totally, or reject the offer totally. Our arrangement empowers the end client to robotize the different asset choice process and scale the same for extensive asset demands. Our work enables a cloud to agent in choosing the best arrangement of cloud merchants who can benefit client demands. This part of astute asset distribution in a cloud up until now was not investigated in awesome detail, and our own is the main push to

achieve the same. We consider cloud asset offerings from various cloud merchants, and have a tendency to accept as likely a future situation where institutionalization and interoperability between sellers are far reaching. Consequently, we actualized the proposed approach utilizing a standard cloud merchants dataset in light of client demands, and found that the victor assurance for combinatorial sell-offs in distributed computing can be accomplished by boosting the benefit to the cloud sellers while in the meantime giving the best offer of asked for assets to the end client. Our work likewise gives the privilege to end clients that they simply need to put their asset demands without stressing over the instrument of securing them. The cloud representative performs barters in the half and half cloud condition and gives the asked for assets at the most ideal cost and Quality of Service (QoS) to the end client.

II. CABOB ALGORITHM

There is no polynomial time algorithm to solve winner determination for combinatorial auctions. Equation is a well known winner determination problem and is NP-complete. In one approach, approximation algorithms are used. These approximate algorithms do not guarantee optimal solutions, but in special cases lead to better solutions.
Another approach is to restrict allowable bids. Even though there are some restrictions under which we can solve in polynomial time, doing so leads to economic inefficiencies. So Sandholm and Suri propose an algorithm to solve the unrestricted winner determination problem using search. This algorithm is popularly called the Branch on Bids (BOB) algorithm.

Symbol	Description
n	Number of cloud vendors
N	A set of cloud vendors, $\{1, 2, ..., n\}$
m	Number of resources
M	A set of resources, $\{1, 2, \ldots, m\}$
B_i	Bid submitted by cloud vendor i
B	A set of bids, $\{B_1, B_2, \ldots, n\}$
S_i	Set of resources provided by cloud vendor i
p_i	Price quoted by cloud vendor i
q_i	Normalized QoS of the cloud vendor i
G	Bid Graph
C	Set of bid graph components
ϵ	Number of bid graph components
V	Number of vertices in bid graph G
E	Number of edges in bid graph G

Notation table

The set of bids are represented as tree nodes. Tree nodes are labeled as either winning (xj = 1) or losing (xj = 0). The tree is searched using DFS. Using heuristics, the contributions of unallocated items are calculated. This contribution along with the revenue generated from bids is used to decide whether to include a bid in the best solution set. This is the main idea of the BOB algorithm. In BOB, there is an one-to-one correspondence between tree leaves and feasible solutions, unlike branch-on-items algorithms where not every feasible solution is represented by any leaf. However, BOB was not implemented fully though several attempts were made in implementing the same. Our algorithm CABOB (Combinatorial Auction Branch on Bids) facilitates combinatorial auctions in cloud computing environments. It incorporates many of the techniques proposed in BOB and other algorithms. The skeleton of CABOB is a depth-first branch-and-bound tree search that branches on bids. Before submitting the bids to the CABOB algorithm, we perform a preprocessing step to normalize the bid that is being produced by the cloud vendors. Since each bid is a tuple, we submit a simple weighted sum of the cost and QoS parameters of each and every resource in the tuple. The weighted sum is defined by Ii = qi + (Sf · ci), where Ii is a constant which is the weighted sum of cost and QoS of bid i, and Sf is the scaling factor for the cost of

the bid i. By doing this, each bid has integer values associated with it for each resource it is bidding for. Algorithm 1 gives the detailed pseudocode. To begin with, the set of resources offered by a vendor is partitioned into pairwise-disjoint subsets. The winner is determined separately in each subset to hasten the search. At each search node, Algorithm 1 uses a data structure called the bid graph, denoted by G. The nodes of graph G represent the bids of unallocated resources. Two nodes in G share an edge whenever the corresponding bids share resources. Let V be the set of vertices of G, and E be the set of edges. At any point of time, |V | ≤ n and |E| ≤ n(n–1)/ 2.

Let f_{opt} be the value of the best solution found so far, as a global variable. We define min as the minimum revenue the cloud vendor expects at the end of the auction and g as the revenue returned at a particular iteration on running the function CA. We start searching by invoking CA(G, 0, 0). Initially, the bid graph G and fopt are empty. We construct the bid graph G incrementally by adding bids Bi. We call algorithm 2 to find the components of the bid graph G. Algorithm 2 is a standard algorithm. First we run DFS and annotate each vertex of G with discover and finish times. Afterward, we compute the transpose graph and perform DFS according to decreasing order of finishing time of the vertices. The vertices of the DFS forest are the separate components of the graph. In an undirected graph, the transpose is the very same graph itself. Hence, in line 2 of Algorithm 2, we perform DFS on the same graph twice. The time complexity of Algorithm 2 is Θ(|V | + |E|). We run Algorithm 2 on G, which results in k independent graphs. In line 12, CA uses an upper threshold on the revenuethe unallocated resources can generate.

Algorithm 1: $\mathcal{CA}(G, g, min)$

Input : Bid Graph G, revenue generated from winning bids g, minimum revenue min per $\mathcal{CA}$

Output: Set of winning bids F_{opt_solved}

1 **if** $|E| = \frac{n(n-1)}{2}$ **then**
2 $\quad f_{opt} \leftarrow \max B$;
3 $\quad$ return f_{opt} ;
4 **end**
5 **if** $|E| = 0$ **then**
6 $\quad$ Accept all the remaining bids;
7 $\quad$ update f_{opt} and return f_{opt};
8 **end**
9 `FindConnectedComponents` (G, C) ;
10 $\alpha \leftarrow |C|$;
11 `// ϵ is the number of components`
12 **for** $i \leftarrow 1$ **to** ϵ **do**
13 $\quad$ calculate an upper threshold $(UT)_i$;
14 **end**
15 **if** $\sum_{i=1}^{\epsilon}(UT)_i \leq min$ **then**
16 $\quad$ return 0;
17 **end**
18 Apply *Integer Relaxation*;
19 **for** $i \leftarrow 1$ **to** ϵ **do**
20 $\quad$ calculate lower threshold $(LT)_i$;
21 **end**
22 $\Delta \leftarrow g + \sum_{i=1}^{\epsilon}(LT)_i - f_{opt}$;
23 **if** $\Delta > 0$ **then**
24 $\quad f_{opt} \leftarrow f_{opt} + \Delta$; $min \leftarrow min + \Delta$;
25 **end**
26 **if** $n < 1$ **then**
27 $\quad$ Choose next bid B_k to branch on ;
28 $\quad f_{opt\_old} \leftarrow f_{opt}$; $f_{in} \leftarrow \mathcal{CA}(G, g + p_k, min - p_k)$;
29 $\quad min \leftarrow min + (f_{in} - f_{opt\_old})$;
30 $\quad \forall B_j$ s.t. $B_j \neq B_k$ and $S_j \cap S_k \neq \emptyset, G \leftarrow G \cup B_k$;
31 $\quad f_{opt\_old} \leftarrow f_{opt}$; $f_{out} \leftarrow \mathcal{CA}(G, g, min)$;
32 $\quad min \leftarrow min + (f_{in} - f_{opt\_old})$;
33 $\quad$ Return $max(f_{in}, fout)$;
34 **end**
35 $F_{opt\_solved} \leftarrow 0$; $H_{unsloved} \leftarrow \sum_{i=1}^{\epsilon}(UT)_i$;
36 $L_{unsloved} \leftarrow \sum_{i=1}^{\epsilon}(LT)_i$;
37 **for** *each component* $c_i \in C$ **do**
38 $\quad$ **if** $F_{opt\_solved} + H_{unsloved} \leq min$ **then**
39 $\quad\quad$ return 0;
40 $\quad$ **end**
41 $\quad t_i' \leftarrow F_{opt\_solved} + (L_{unsloved} - (LT)_i)$;
42 $\quad f_{opt\_old} \leftarrow f_{opt}$;
43 $\quad f_{opt\_i} \leftarrow \mathcal{CA}(G_i, g + t_i', min - t_i')$;
44 $\quad min \leftarrow min + (f_{opt\_old} - f_{opt})$;
45 $\quad F_{opt\_solved} \leftarrow F_{opt\_solved} + f_{opt\_i}$;
$\quad H_{unsloved} \leftarrow H_{unsloved} - H_i$;
46 $\quad H_{unsloved} \leftarrow H_{unsloved} - H_i$;
47 **end**
48 return F_{opt_solved}

Algorithm 2: FindConnectedComponents(G, C)

Input : Bid Graph G
Output: Set of components $C = \{c_1, c_2, \ldots, c_n\}$

```
1 // DFS annotates each vertex with
    discover and finishing time
2 DFS (G) ;
3 // In undirected graph G, G^T = G
4 // Consider vertices in decreasing
    finishing time
5 DFS (G) ;
6 Vertices in each tree of the depth-first forest is a
  separate component ;
```

If the current solution is not better than the optimal solution fopt, CA prunes the search path. We use an LP formulation for estimating the upper threshold. The main aim of using upper threshold is to speed up the search path pruning without affecting the optimality. Before starting the LP, one could look at the condition in line 14 to determine the minimum revenue the LP has to produce so that the search branch would not be pruned. Once the LP solver finds a solution that exceeds the threshold, it could be stopped without pruning the search branch. If the LP solver does not find a solution that exceeds the threshold and runs to completion, the branch could be pruned. However, CA always runs the LP to completion, since it uses the solutions from the LP and the dual in several ways. After estimating the upper threshold, we apply an integer relaxation where we can either accept the bid completely or reject the bid completely, as is shown in line 17. Partial acceptance is not possible, by the very nature of combinatorial auctions. A case can be noted where a single cloud vendor gives an exclusive offer of providing all the resources with a good cost tradeoff. CA calculates a lower threshold on the revenue that the remaining resources can contribute, as shown in line 21. If the lower threshold is high, it can allow fopt to be updated, leading to more pruning and less search in the subtree rooted at that node. Any lower thresholding technique could be used here. We use the following rounding technique. CA solves the remaining LP, which gives an acceptance level xj , 0 ≤ xj ≤ 1, for every remaining bid Bj . We insert all bids with xj ≥ 0.5 into the lower-threshold solution. We then try to insert the rest of the bids in decreasing order of xj , skipping bids that share resources with bids already in the lower threshold. Based on the value of the lower bound obtained, we calculate the value of the increment, which is nothing but the difference of the sum of current revenue obtained and the summation of lower bounds and the current fopt. If this is greater then zero, then we update the values of fopt and min as shown in line 23. If the number of independent subgraphs is less than 1, we choose the next bid to branch upon and update the values of fopt and min accordingly. Finally, for each of the subgraphs that is being obtained, we recursively call CA to obtain the best auction results and declare a set of cloud vendors as the winners. This can be seen in lines 28 through 50. After each iteration, we check whether the solution obtained covers most if not all of the requested resources from the cloud vendors. Then for each of the resources that is not being procured, we update the values of min and fopt and recursively call CA as shown in lines 26 through 45. Finally the set of winning cloud vendors is returned in line 47. Our algorithm does not make copies of the LP table, but incrementally adds rows from the LP table as bids are removed into G as the search proceeds down a path. Hence, it has linear time complexity.

III. CONCLUSION

Hence, we have proposed the CABOB algorithm, a domain specific improvement of the CABOB algorithm, to permit fast winner determination in combinatorial auction mechanisms, and found a way to produce optimal resource procurement for the user requesting a set of resources. When tested with an actual sample dataset of cloud computing, we found that resource procurement in combinatorial auctions in the proposed manner is far superior compared to sequential auctions. Also, combinatorial auctions in cloud computing can be scaled to large user requirements. We foresee a scenario where combinatorial auctions using this approach will be extensively used by a very large numbers of cloud

users to procure sets of resources economically from the many cloud vendors who offer myriad sets of resources with different specifications that cannot be meaningfully compared and analyzed in any other way. Our algorithm CABOB (Combinatorial Auction Branch on Bids) thus has advantages for both the service providers and the cloud users. As the number of resources requested increases, the challenges faced by service providers increase. This creates a need for service providers to come up with better procurement models that ensure quality of service while also improving utilization and profitability. This can be done at scale using our approach.

IV. REFERENCES

[1]. P.Mell and T.Grance, The NIST Definition of Cloud Computing.National Institute of Standards and Technology (NIST), US Dept.of Commerce, Sep.2011, 2L.Badger, T.Grance, R.Patt-Corner, and J.Voas, Cloud Computing Synopsis and Recommendations.National Institute of Standards and Technology (NIST), US Dept.of Commerce, May 2012,

[2]. Y.O.Yazir, C.Matthews, R.Farahbod, S.Neville, A.Guitouni, S.Ganti, and Y.Coady, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, ser.CLOUD '10.Washington, DC, USA: IEEE Computer Society, 2010

[3]. F.Ridder and A.Bona, Four Risky Issues When Contracting for Cloud Services.Gartner Research Report G00210385, Feb.2011.

[4]. R.Weiss and A.Mehrotra, "Online dynamic pricing: Efficiency, equity and the future of e-commerce," Virginia Journal Of Law And Technology, vol.6, no.2, 2001.

[5]. C.Charnes, J.Pieprzyk, and R.Safavi-Naini, "Conditionally secure secret sharing schemes with disenrollment capability," in ACM Conference on Computer and Communications Security (CCS), 1994, pp.89-95.

[6]. A.Desai, "The security of all-or-nothing encryption: Protecting against exhaustive key search," in Advances in Cryptology (CRYPTO), 2000, pp.359-375.

[7]. C.Dubnicki, L.Gryz, L.Heldt, M.Kaczmarczyk, W.Kilian, P.Strzelczak, J.Szczepkowski, C.Ungureanu, and M.Welnicki, "HYDRAstor: a Scalable Secondary Storage," in USENIX Conference on File and Storage Technologies (FAST), 2009, pp.197-210.

[8]. M.Durmuth and D.M.Freeman, "Deniable encryption with negligible detection probability: An interactive construction," in EUROCRYPT, 2011, pp.610-626.

[9]. S.Penmatsa and A.Chronopoulos, "Price-based user-optimal job allocation scheme for grid systems," International Symposium on Parallel and Distributed Processing, p.396, April 2006.

[10]. X.Xie, J.Huang, H.Jin, S.Wu, M.Koh, J.Song, and S.See, "Pricing strategies in grid market: Simulation and analysis," International Conference on Grid and Cooperative Computing, pp.532-538, October 2008.

[11]. R.Buyya, D.Abramson, J.Giddy, and H.Stockinger, "Economic models for resource management and scheduling in Grid computing," Concurrency and Computation: Practice and Experience, vol.14, no.13-15, pp.1507-1542, 2002.

[12]. I.Foster, C.Kesselman, C.Lee, B.Lindell, K.Nahrstedt, and A.Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," in International Workshop on Quality of Service, 1999.

[13]. H.Casanova and J.Dongarra, "Netsolve: A network server for solving computational science problems," in The International Journal of Supercomputer Applications and High Performance Computing, 1995.

[14]. S.J.Chapin, D.Katramatos, J.F.Karpovich, and A.S.Grimshaw, "The legion resource management system," in Proceedings of the Job

Scheduling Strategies for Parallel Processing, ser.IPPS/SPDP '99/JSSPP '99.London, UK: Springer-Verlag, 1999.

[15]. S.Parsons, J.A.Rodriguez-Aguilar, and M.Klein, "Auctions and bidding: A guide for computer scientists," ACM Computing Surveys, vol.43, no.2, Jan.2011,

[16]. B.Rochwerger, J.Tordsson, C.Ragusa, D.Breitgand, S.Clayman, A.Epstein, D.Hadas, E.Levy, I.Loy, A.Maraschini, P.Massonet, H.M.noz, K.Nagin, G.Toffetti, and M.Villari, "RESERVOIR— when one cloud is not enough," IEEE Computer, Mar.2011.

[17]. M.Bichler, J.Kalagnanam, K.Katircioglu, A.J.King, R.D.Lawrence, H.S.Lee, G.Y.Lin, and Y.Lu, "Applications of flexible pricing in business-to-business electronic commerce," IBM Syst.J., vol.41, no.2, pp.287-302, 2002.

[18]. Y.Narahari, C.Raju, K.Ravikumar, and S.Shah, "Dynamic pricing models for electronic business," Sadhana, vol.30, pp.231-256, 2005,

[19]. S.Chaisiri, B.-S.Lee, and D.Niyato, "Optimization of resource provisioning cost in cloud computing," IEEE Transactions on Services Computing, vol.5, 2012