BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No X

Sofia • 2012

Fair Resource Allocation: Load and Cost Elasticity Defined Game Theoretic Approach

Sujata Gaddemane, Snehanshu Saha*, Bidisha Goswami*,Sumana Sinha**

**PES Institute of Technology, Bangalore South Campus, Bangalore, India PIN-560100*
Emails:gsujata@gmail.com,snehanshusaha@pes.edu,bidishagoswami@pes.edu, sumanasinha@pes.edu

Abstract: *Cloud Computing is a key methodology for sharing resources. Multi tenancy feature of cloud enables efficient resource sharing among multiple users simultaneously. While the resource sharing is efficient, there is a possibility of performance degradation due to the load imbalance created by the nature of resource allocation. Given an option, users are likely to be attracted towards using servers with lower unit cost, which can lead to increased load on such server and thus resulting in poor performance. This in turn leads to higher response time resulting in increased average cost to the cloud users. Objective of this project is to optimize the resource allocation in cloud environment using the mechanisms defined by game theory. In proposed method, cost charged per user is calculated based not only on the unit cost of the server but also on the current load of the server. Thus the proposed model ensures the users are charged optimally and least load imbalance among the servers after the allocation. The Cobb-Douglas production function is used for computing cost incurred by each client. Multiple experiments are carried out which shows that the load imbalance factor after the allocation among the servers is less than 1 with this proposed method.*

Also, results show that unit cost charged per user is optimized and is always less than maximum unit cost of any server of the system.

Keywords: *Cloud Resource Allocation, Cobb-Douglas, Fair Resource Allocation, Game Theory, Extended Form Game, Nash Equilibrium*

1. Introduction:

Cloud computing, with its multi-tenancy feature, enables efficient sharing of resources among multiple users. The nature of resource allocation, however could lead to load imbalance and therefore performance degradation. Allocating user demands on servers with least fixed unit cost would over-load them. This leads to poor performance and higher response time which in turn results in an increased average service cost per tenant. The objective of this work is to optimize resource allocation in the cloud environment using a game theoretic approach. The calculation of the average service cost per user in the proposed method, is based both on the Fixed unit cost of the server as well as the server's Load. This approach ensures average service cost per tenant and minimal load imbalance among the cloud servers. Cobb Douglas production function is used to compute the average service cost incurred for each tenant. Series of experiments resulted in a post-allocation load imbalance factor of less than 1 among the servers and an optimum unit cost per tenant, which is less than the maximum unit cost of any server in the cloud environment. Resource allocation in a cloud environment is a challenge due to the dynamic nature of user requests, heterogeneous environment, various cost models etc. A cloud service provider prefers to choose a resource allocation policy which ensures lower average service cost per user. A resource allocation strategy that considers only the Fixed unit cost of the server on which the workload will be allocated is not sufficient because it ignores the load on the server. It is important to note that an overloaded server leads to a higher turnaround time and hence an increased average service cost per user.

Consider a bunch of points (Here, Servers-Abstract sense) in space with coordinates $(x, y) : where, x = Fixed - unit - \cos t, y = Load.$ Objective is to identify the optimal x and y pairs or optimal fixed cost and Load values that ensure an optimal average service cost per user.

2. Related Work:

A new class of games called Cloud Resource Allocation Games (CRAGs) is discussed by Jalparti[3] which models resource allocation problem using game theory. The authors capture the interactions between client-client and client-provider and ensure that the clients are charged optimally for their resource usage. They argue that the existing pricing and scheduling models do not provide the price-to-performance guarantee to clients. Their work demonstrates the variation in response time with respect to resource contention in cloud and compares the cost incurred by clients using linear and exponential cost functions. The performance of the model is compared with that of the Round-Robin mechanism. The authors find equilibrium for the resource allocation problem by using the Stackelberg games. Wui et. al.[10] propose resource allocation algorithms for SaaS providers who want to minimize infrastructure cost and SLA violations. They address the situations where SaaS providers use IaaS to host their services. They have implemented cost driven algorithms which considered various QoS parameters (such as arrival rate, service initiation time and penalty rate) from both the customers' and the SaaS providers' perspective. Here, the strategy behind cost minimization is reuse of virtual machines. Toosi et. al.[7] have proposed a model for revenue maximization for IaaS. IaaS cloud providers offer diverse purchasing options and pricing plans, namely on-demand, reservation, and spot market plans. They address a novel problem of maximizing revenue through an optimization of capacity allocation to each pricing plan by means of admission control for reservation contracts, in a setting where aforementioned plans are jointly offered to customers. They device an algorithm based on a stochastic dynamic programming formulation. Xu & Yu[5] also modelled resource allocation in a virtual machine environment using game theory. Their work focuses on fair resource allocation for each user while supporting efficient resource allocation for each physical server. Cloud infrastructure comprises of heterogeneous physical resources like CPU, memory and storage. An intelligent allocation decision should ensure that the resource allocation among various virtual machines is fair. The authors model the resource allocation problem as an extensive form game and identify an optimal resource allocation strategy by finding the game's Sub game Perfect Nash Equilibrium (SPNE) using backward induction method. Nahir et. al.[6] have proposed a model workload factoring using game theory. The benefit a user gets from using cloud is related to usage pattern of other cloud users. Certain heavy cloud users may scare off other users thus resulting in a non-cooperative game. The authors propose a game theoretic approach for fair resource allocation for a wide range of user types. Their results show that there is a unique Nash Equilibrium (NE) strategy which is optimal. Wei et. al.[10] discuss the Quality of Service (QoS) constrained resource allocation problem. They address the parallel computing problem in cloud, where resources across cloud are used for computation. The problem is modelled using game theory and is solved in two stages. In the first stage each participant finds its optimal resource allocation. This optimization problem has been solved using the Binary Integer programming method. In the second stage,

initial strategies are multiplexed and an optimal solution is found for these multiplexed strategies. Rao et. al. [11] discuss about the initial provisioning of resources and subsequent uninterrupted operation of the infrastructure. The cyber and physical components of a cloud infrastructure are subject to attack. A service provider should ensure that its users continue to get planned aggregate computational capacity even in the event of an attack. Service providers deploy redundant components as a counter measure to mitigate and contain infrastructure degradation or attacks. The authors propose a game theoretic approach for initial infrastructure provisioning and operation under uniform cost models. This entails deciding the number of servers to be deployed at various sites and the reinforcement of selected infrastructure components. G. Arun et. al[29] proposes a single resource game based allocation model based on history of winning in a game. Niyato et. al. [12] proposes a game theory based resource and revenue sharing model with a coalition of cloud service providers. In a coalition model, multiple cloud service providers come together and cooperatively form a resource pool and provide cloud services to users. The authors propose a stochastic linear programming game model to guide the sharing of revenue and resource pool among the service providers. Ardagna et. al.[26] propose a game theory based approach to run time management and allocation of resources of IaaS provider among various competing SaaS service providers. Fei et. al. [27] proposes a resource pricing and allocation policy where users can predict the future resource price as well as satisfy budget and deadline constraints. They proposed a game theoretic approach to achieve equilibrium between two conditions budget and deadline. Nezarat et. al.[28] propose an auction based method to determine the auction winner by applying game theory mechanism. This method reaches a Nash equilibrium point where players do not alter their bid for the resource and auctioneer is also satisfied with the utility function. Thus this approach satisfies user's expectation of best resource within the given budget and satisfies profitability expectation of cloud service provider.

Literature on Resource allocation in Cloud computing topic is vast and it has various dimensions to it. We have mostly focused on resource allocation strategies that adopt game theoretic approach. The literature on such a topic which addresses SaaS cloud running on IaaS cloud is limited.

3. Proposed Model:

This work is aimed at finding the optimal resource allocation strategy for the SaaS provider at a given point in time. Cobb-Douglas production function is used to compute the service cost. Here, the cloud resource allocation problem is modelled using game theory.

Any game theory model consists of 3 elements Osborne[13]

- Players who take part in the game. There can be more than 2 players in a game.

- Actions or different options available for each player at each decision point. There can either be one or more options available for a player.
- Preferences, i.e. ordinal preference of the player over the available options.

We map these three components of Game Theory to resource allocation game as below:

- **Players**: In the cloud resource allocation game, objective of proposed model is to achieve optimal average service cost per user. Thus, users are indirectly playing the game through service demands.
- **Actions**: The distribution strategies among available servers are actions available to the user.
- **Preferences**: Cost incurred by the SP for adopting the demand distribution strategy, will decide the ordinal preference.

Note: In game theory model, equilibrium is computed based on highest utility. However this work aims at finding the resource allocation combinations that result in minimum cost. So utility is computed using U_i.

The model is based on the following assumptions:

- Cloud provides SaaS to users and is built on IaaS. Thus the cloud environment consists of servers whose fixed unit cost varies based on the IaaS vendor.
- SaaS provider is interested in servicing user demands with optimal average service cost per user.
- User requests are addressed at specific time intervals. All requests pending at the decision moment are considered for resource allocation.

  Here we are going to discuss the essential component used in basic model for this article.

3.1. Game Theory:

As discussed by Osborne[13] and Davis[14]**,** game theory helps in modelling strategic situations. Game theory assists in determining the best possible outcome for all players resulting in an optimal outcome for each player, thereby achieving Equilibrium. Equilibrium is an action profile for all players where, no player can get a better outcome by unilaterally changing his action from the Equilibrium profile.

This work models cloud resource allocation using game theory where a game is modelled as an Extensive Form Game (EFG). The objective of this work is to distribute client requests among available servers in a cloud environment in such a way that the average service cost incurred per user is optimal and the post-allocation load imbalance among servers is least.

The optimal solution for such an EFG is obtained by finding the Sub-game Perfect Nash Equilibrium (SPNE) with backward induction being a natural choice for finding SPNE.

3.2. Cobb- Douglas Production Function:

Cobb-Douglas (CD) production function was chosen Cobb and Douglas [15] to compute the cost incurred by SP, on provisioning a resource request in cloud. Cobb-Douglas production function [23] is most extensively used in economics to represent the relationship between the output and the combination of inputs used to obtain it. Following important characteristics of this production function have made it an attractive choice in the economic domain.

- Positive decreasing marginal product
- Constant output elasticity
- Constant returns to scale

The proposed work , models the output Cost to be dependent on input parameters 'Load' and 'Fixed unit cost' and aims to find the optimal cost for the resource allocation. Authors have proved that CD production function is minimized at $\alpha + \beta > 1$ i.e. Increasing returns to scale (IRS).The cost function is defined as

$$C(F, L) = KF^{\alpha} L^{\beta} \quad (1)$$

Where,
C = Total cost incurred by the SaaS provider
L = Load on Server
F = Fixed unit cost of the server
K = Is a positive constant which explains technological influence on the Model.
α and β are output elasticity of the Fixed unit cost and Load.
α and β are output elasticity of labour and capital, respectively in the classical Econometric model in production [23] modified suitably to serve the context of the problem stated. These values are constants determined by available technology. Output elasticity measures the responsiveness of output to a change in levels of either labour or capital used in a production. For example, if = 0.15, a 1% increase in labour, would lead to approximately a 0.15% increase in output.

3.3. Proposed Method:

As discussed by Let there be M physical servers $\{1, 2, ...M\}$ in the cloud environment. Each server i has fixed unit cost F_i, assigned to it. Let L_i be the current load on the server i, which is defined in percentage, highest being 100 and least being 1.
Let there be N $\{1, 2, ...N\}$ users who are simultaneously demanding resources from the cloud. Request from user j is defined as R_j. For all users the resource demand can be represented by the vector $(R_1, R_2, ...R_n)$. Resource demand by each

user is distributed among available servers. It can be represented by the vector $(a_j(1), a_j(2), ...a_j(M))$, where $a_j(k)$ represents resource allocated for user j on server k. Distribution of the user's total demand among all the servers should be equal to the total demand by the user.

$$\sum_{k=1}^{M} a_j(k) = R_j \tag{2}$$

Load factor L_f computes the load imbalance incurred among the servers due to the allocation $(a_j(1), a_j(2), ...a_j(M))$.The goal is to achieve low load imbalance after the allocation. A lower load imbalance ensures the most appropriate allocation. For the allocation $(a_j(1), a_j(2), ...a_j(M))$, the load factor L_f is computed using

$$L_f = \frac{\sum_{i=1}^{M-1} \sum_{k=i+1}^{M} (a_j(i)L_i - a_j(k)L_k)}{M} \tag{3}$$

Cobb- Douglas production function is used for computing the cost of using server i

$$C_i(F, L) = KF_i^{\alpha} L_f^{\beta} \tag{4}$$

For the allocation $(a_j(1), a_j(2), ...a_j(M))$, the total cost incurred by SP for allocating user j's demand, is computed using

$$TotalCostPerUser_j = \sum_{k=1}^{M} a_j(k)C_k \tag{5}$$

Here, the computation considers both load imbalances created by the given allocation and Fixed unit cost of the server. The game theory model is interested in the utility gained by the service provider for a given allocation choice. If there are N users, Utility U_j for allocating user $j's$ demand is computed as a ratio of the cost incurred for user j, to the sum of the cost incurred for all users participating in the game. Following formula represents this computation.

$$U_j = -\frac{TotalCostPerUser_j}{\sum_{k=1}^{N} TotalCostPerUser_k} \tag{6}$$

Outlined below are the main steps involved in the proposed method.

1. Read all the input parameters, i.e. Load and fixed unit cost of all the participating servers.
2. Find all possible resource allocation combinations for each user based on demand by user and number of servers available.
3. Find values of α and β for each server using Gradient Descent method.
4. Compute the cost for each allocation combination of all users, using Cobb-Douglas production function (5). Select best combinations based on least cost.
5. Model the game using Extensive Form Game using these best selections. Compute the utility using (6) for each user and for each allocation combination.
6. Optimal solution of the game is SPNE. SPNE is computed using backward induction method.

Refer Appendix A for algorithms to find SPNE.

4. Discussion about Elasticity:

Output elasticity of Cobb-Douglas production function is the accentual change in the output in response to a change in the levels of any of the inputs Cobb and Douglas [15]. In (1), α and β are the output elasticity of fixed unit cost and Load respectively. Accuracy of α and β values is the key to deciding the right resource allocation combination. Different approaches were analyzed before arriving at final decision.

4.1. Computing Elasticity via Gradient Descent:

Gradient Descent algorithm was used to find the values of α and β. Gradient Descent is an optimization algorithm used for finding the local minimum of a function. Given a scalar function $F(x)$, gradient descent finds the $\min_x F(x)$ by following slope of the function $\frac{\partial F}{\partial x}$. This algorithm selects initial values for the parameter x and iterates to find the new values of x which minimizes $F(x)$.

Minimum of a function $F(x)$ is computed by iterating through following step,

$$x_{n+1} \leftarrow x_n - \delta \frac{\partial F}{\partial x}$$

Where,

x_n = initial value of x

x_{n+1} = new value of x

$\frac{\partial F}{\partial x}$ = slope of function $F(x)$

δ = step size, which is > 0, forces algorithm to makes small jump

The objective of this paper is to find resource allocation pattern, which incurs optimal cost for the service provider, i.e. to find

$$\min_{\alpha,\beta} C(F, L) \tag{7}$$

s.t.c

$$0 < \alpha$$
$$0 < \beta$$
$$\alpha + \beta > 1$$

Table 1 and Table 2 show values of and computed using Gradient Descent method for various combinations of Load and Fixed unit cost.
Refer Appendix A for Gradient Descent algorithm.

4.2. Computing Elasticity via Constraint Optimization

Let the assumed parametric form of (1) be

$$C = K' + \alpha \log(F) + \beta \log(L) \tag{8}$$

Consider a set of data points.

$$C_1 = K' + \alpha F_1' + \beta L' \tag{9}$$
$$\cdots$$
$$C_N = K' + \alpha F_N' + \beta L_N' \tag{10}$$

Where,

$$K' = \log(K)$$
$$F_i' = \log(F_i)$$

Constraints on parameters: This results in a constrained optimization problem. The objective function to be minimized is $(y - Ax)^T (y - Ax)$ and this is a quadratic form in x. If the constraints are linear in x, then the resulting constrained optimization problem is a Quadratic Program (QP).

A standard form of a QP is :

$$\min x^T Hx + f^T x \tag{11}$$

Subject to the constraints,

$Cx \le b$ Inequality Constraints

$C_{eq} x = b_{eq}$ Equality Constraints

Suppose the constraints are that α and β are > 0 and $\alpha + \beta > 1$. The quadratic program can be written as (neglecting the constant term $y^T y$).

$$\min x^T (A^T Ax) - 2y^T Ax \tag{12}$$

s.t.c

$$0 < \alpha$$
$$0 < \beta$$
$$\alpha + \beta > 1$$

In standard form as given in (11), the objective function can be written

as :

$$\min x^T Hx + f^T x \tag{13}$$

Where,

$$x = \begin{pmatrix} K' & \alpha & \beta \end{pmatrix}^T \ , \ H = A^T A \ , \ f = -2A^T y$$

The inequality constraints can be specified as

$$C = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & -1 \end{pmatrix} \text{ and } b = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

By solving above constrained quadratic problem, following values were obtained: log(K) = 0.627368, K = 4.2393, α = 0.671056, β = 0.333944. This result satisfies the condition $\alpha + \beta > 1$

Elasticity α, β and K from both the computations are very close, supporting our choice of α, β and K.

TABLE 1: Simulation results: α & β values on Server1 and Server2 with assumption $K = 1$ Results satisfy the condition $\alpha + \beta > 1$

Server Cost	Server Load	α	β
50	10	1.8768	0.0275
50	3	1.664	0.005
5	80	1.963	0.000054
10	40	1.938	0.0007
40	25	1.918	0.0289
30	6	1.8208	0.0056
10	20	1.924	0.00115

TABLE 2: Simulation results: α & β values on Server1 and Server2 without any assumption on K. Results satisfy the condition $\alpha + \beta > 1$

Server Cost	Server Load	α	β	K
50	10	0.611749	0.389197	4.288795
50	3	0.567333	0.434660	4.286146
5	80	0.719920	0.281966	4.288413
10	40	0.685039	0.315826	4.288273
40	25	0.640219	0.360577	4.289915
30	6	0.605103	0.397328	4.286653
10	20	0.670304	0.331262	4.286881

4.3. Optimality of Parameter

The computational exercise to estimate "good fit value of α and β " are further justified by the analytical evidence. This evidence is derived from the functional properties of CD production function, which is modelled and applied as a function of two variables, fixed unit cost and Load. Achieving minimum cost is influenced by values of α and β of (4), for the given combination of Fixed unit cost and Load. Here is the proof for convexity of Cobb- Douglas w.r.t α and β .

A c^2 function $f : U \subset R_n \to R$ defined on a convex open set U is convex if and only if the Hessian matrix $D^2 f(x)$ is positive semi-definite for all $x \in U$. A matrix H is positive semi-definite if and only if its $2^n - 1$ principal minors are all ≥ 0 [Appendix C].

Cobb- Douglas function for 2 inputs is: $f(x, y) = cx^{\alpha} y^{\beta}$

Its Hessian is:

$$\begin{bmatrix} cx^{\alpha} y^{\beta} (\ln(x))^2 & cx^{\alpha} y^{\beta} \ln(x) \ln(y) \\ cx^{\alpha} y^{\beta} \ln(x) \ln(y) & cx^{\alpha} y^{\beta} (\ln(y))^2 \end{bmatrix}$$

$$\Delta_1 = cx^{\alpha} y^{\beta} (\ln(x))^2 \quad (14)$$

$$\Delta_2 = cx^{\alpha} y^{\beta} (\ln(y))^2 \quad (15)$$

$$\Delta_3 = c^2 x^{2\alpha} y^{2\beta} (\ln(x))^2 (\ln(y))^2 - c^2 x^{2\alpha} y^{2\beta} (\ln(x))^2 (\ln(y))^2 \quad (16)$$
$$= 0$$

Conditions for a function to be convex are,

$\Delta_1 \geq 0,$

$\Delta_2 \geq 0,$

$\Delta_3 \geq 0$

From the conditions defined for the resource allocation problem we know that, $x > 0, y > 0, \alpha > 0, \beta > 0, c > 0$ .With these conditions on input values, Δ_1 and Δ_2 as given in (8) and (9) are always ≥ 0 and Δ_3 is always zero as given in (10)

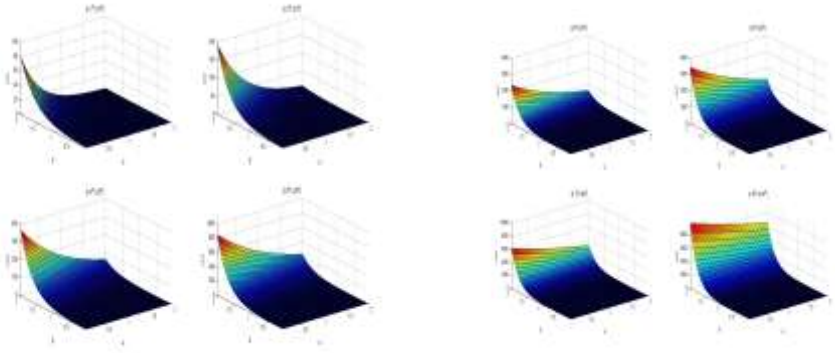

Fig-1 : Plot of CD of α, β for minimum cost with different combinations of Load Factor and Fixed Cost

Hence it is established that Cobb- Douglas w.r.t α and β is convex. The 3-D graphs in Fig 1 are plotted with α and β against Cost. Here, x-axis represents output elasticity α of fixed unit cost on the server and y-axis represents output elasticity β of Load of the server.

These graphs are plotted by taking different values for L and F . It is evident from the graphs, that the Cobb-Douglas function obtains minimum Cost when $\alpha + \beta > 1$

Refer Appendix B for the proof for obtaining minimum cost while $\alpha + \beta > 1$

5. Illustration of the Algorithm:

5.1. Simulation Environment:

The proposed model is simulated using a Python program. Input to this program is
- Number of servers
- Number of users
- Fixed unit cost of each server
- Load on each server
- Service demands of all participating users

The program displays results on the screen, which includes
- Average service cost for each user
- Average service cost across the system
- Allocation strategy per user
- Final resource allocation across the system
- Load balancing factor of allocation
- Utility of the allocation strategy

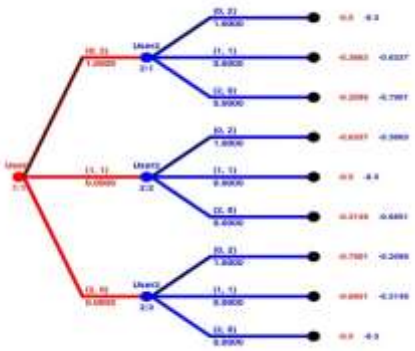

Fig-2: Resource Allocation-EFG

Fig 3 and Fig 4 are the actual screen shots from the simulation environment.This program also generates a file in the format that is understood by the tool by McKelvey[16] and can display the results in the tree format as shown in Fig 2.

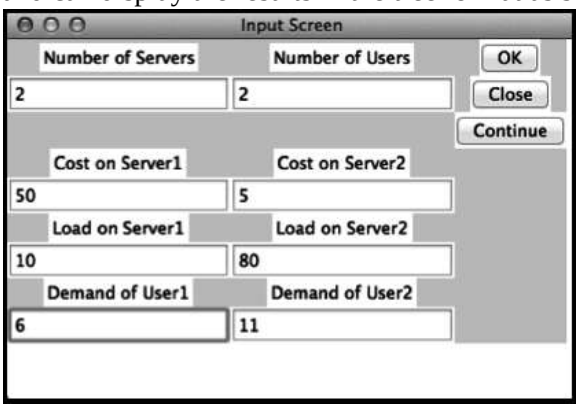

Fig-3: Simulation Environment: Input Screen

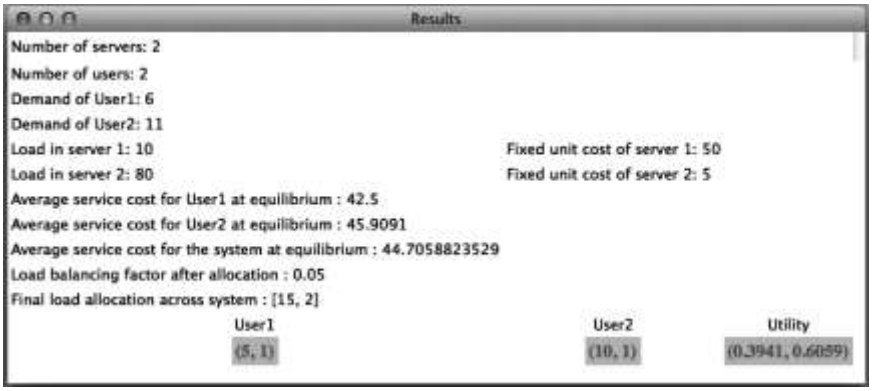

Fig-4: Simulation Environment: Results

5.1. Work Flow:

Here is an example to show complete working of the algorithm. Let us consider a cloud consisting of two servers, configured as given in Table 3. Two users user1 and user2 are requesting for CPU hours (2, 2) respectively. The utility computed for all possible allocation of user demands is shown in Table 4.

In the table the entry under (1,1) under column User1 Allocation indicates that out of 2 hours of demand from user1, 1 hour is allocated on server1 and another hour is allocated on server2.

Since server2 has lower load, this algorithm allocates more load to server2 and the allocation is (0, 2) to user1 and (0, 2) to user2, i.e. both users get 2 hours on server2. From the Table 4, it is clear that the allocation (0, 2) for user1 and (0, 2) for user2 charges minimum for these users.

Fig 2 gives the visual representation of the resource allocation for the above example in EFG format. An open source tool Gambit McKelvey (2014) is used for generating this visual representation. In this figure, user1 and user2 are 2 game players. Since both users are demanding 2 hours of CPU, both user1 and user2 have 3 strategies (0, 2), (1, 1) and (2, 0). user1 takes the action first, followed by user2. The value pairs at the leaf node indicate the utilities of user1 and user2 for the actions chosen respectively.

Let us start from the leaf node of the tree. Let us consider the sub game of user2; for any strategy selected by user1, strategy (0, 2) gets maximum utility compared to other strategies. user1 being aware of this, chooses (0, 2) which maximizes its own utility. This is backward induction procedure which finds SPNE for the game. Thus SPNE for the given example is [(0, 2); (0, 2)] which means both users get 2 hours on server2. The edges highlighted in the tree given in Fig. 2 indicate the strategies selected by each user.

Table 3: Server Configuration

Server	Load	Fixed Unit Cost
1	40	50
2	10	50

Table 4: Simulation Results: Allocation, Cost and Utility

User 1 Allocation	User 2 Allocation	Utility
(2,0)	(2,0)	(-0.5,-0.5)
(2,0)	(1,1)	(-0.6851, -0.3149)
(2,0)	(0,2)	(-0.7901, -0.2099)
(1,1)	(2,0)	(-0.3149, -0.6851)
(1,1)	(1,1)	(-0.5,-0.5)
(1,1)	(0,2)	(-0.6337, -0.3663)
(0,2)	(2,0)	(-0.2099, -0.7901)
(0,2)	(1,1)	(-0.3663, -0.6337)
(0,2)	(0,2)	(-0.5,-0.5)

6. Result

Goal of this work is to distribute user demands on servers by ensuring optimal service cost per user as well as least load imbalance among the servers. Following components are measured to determine the efficiency of the resource allocation algorithms.

- Average service cost per user
- Average service cost across the system
- Load imbalance factor among the servers due to selected allocation strategy.

Average service cost per user and across the system is expected to be lower than or equal to the maximum cost charged per unit time by any server. Load imbalance factor is expected to be lower than 1 among the servers after the allocation.

Table 5: Simulation Results: Allocation of Resources and Load Factor, with $\alpha = 1.9, \beta = 0.02$ and $K = 1$

Serv1 Cost	Serv2 Cost	Serv1 Load	Serv2 Load	user1	user2	Load Factor
50	50	10	10	(3,3)	(5,6)	0.05
50	50	10	3	(1,5)	(3,8)	0.005
10	50	10	10	(4,2)	(6,5)	0.15
50	5	10	80	(5,1)	(10,1)	0.05

50	10	10	40	(5,1)	(9,2)	0.1
5	30	25	6	(1,5)	(2,9)	0.045

Table 6: Simulation Results: Allocation of Resources and Load Factor, with $\alpha = 0.6710, \beta = 0.3339$ and $K = 4.2393$

Serv1 Cost	Serv2 Cost	Serv1 Load	Serv2 Load	user1	user2	Load Factor
50	50	10	10	(3,3)	(5,6)	0.05
50	50	10	3	(1,5)	(3,8)	0.005
10	50	10	10	(3,3)	(6,5)	0.05
50	5	10	80	(5,1)	(10,1)	0.05
50	10	10	40	(5,1)	(9,2)	0.1
5	30	25	6	(1,5)	(2,9)	0.045

Actual results meet the expectations. Table 5 shows allocation strategies adopted by the algorithm. Here two users user1 and user2 are requesting for (6, 11) hours respectively.

In Table 5, row 1, load and fixed unit cost on both the servers are same. Algorithm has allocated resources on both the servers equally. In row 2, allocation is influenced by lower load on server2 and hence has allocated more resources on server2. In row 3, allocation is influenced by lower cost on server1 and hence more load is allocated on server1. Though cost is low on server 1, all the load is not allocated on server1 because, such an allocation creates more load imbalance among the servers. Similar intelligent decisions can be noticed in other allocations too.

Table 7: Simulation Results: Unit Cost per user and system, with $\alpha = 1.9, \beta = 0.02, K = 1$

Serv1 Cost	Serv2 Cost	Serv1 Load	Serv2 Load	Avg.Ser Cost User1	Avg.Ser Cost User2	System Cost
50	50	10	10	50	50	50
50	50	10	3	50	50	50
10	50	10	10	23.33	28.18	26.47
50	5	10	80	42.5	45.9	44.7
50	10	10	40	43.3	42.72	42.94
5	30	25	6	25.83	25.45	25.58

Table 8: Simulation results: Unit Cost per user and system, with $\alpha = 0.6710, \beta = 0.3339$ and $K = 4.2393$

Serv1 Cost	Serv2 Cost	Serv1 Load	Serv2 Load	Avg.Ser Cost User1	Avg.Ser Cost User2	System Cost
50	50	10	10	50	50	50
50	50	10	3	50	50	50
10	50	10	10	30	28.18	28.82
50	5	10	80	42.5	45.9	44.7
50	10	10	40	43.3	42.72	42.94
5	30	25	6	25.83	25.45	25.58

Table 7 shows the average service cost per user and for the system at equilibrium. In row 3, the load on both the servers is same and average service cost per user is less than average unit cost of the system. Here unit cost is 30. But average service cost for all users and across the system is less than the average 30. Row 4 indicates when the load on servers is different, average service cost per user is greater than the average system unit cost, but is always less than the maximum cost of a single system. Since both load and Fixed unit cost influence the resource allocation, though load is lower on server1, some resources are allocated on server2 which has lower Fixed unit cost. Thus it ensures average service cost per user and across system is less than the maximum unit cost of any single system.

Load balancing factor across the system for the selected allocation $(l(1), l(2), ...l(M))$ is computed using

$$L_f = \frac{\sum_{i=1}^{M-1} \sum_{k=i+1}^{M} (l(i)L_i - l(k)L_k)}{M} \qquad (17)$$

Where, L_i is current load on server i

$l(i)$ is new load on server i

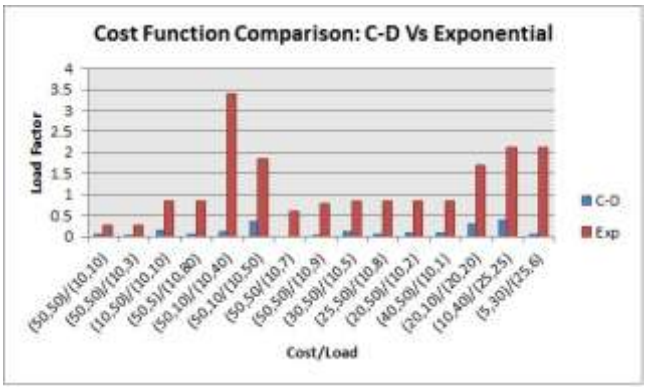

Fig-5: Bar graph for local factor behavior according to different cost/load among the servers

From Table 5, it is clear that after the allocation the load imbalance factor is always lower than 1. Fig 5 is a bar graph which compares Load factor achieved with CD and exponential cost functions. This shows the impact of fixed cost and load among the two servers on the Load factor after the allocation. X-Axis of the graph represents cost among the servers / load among the servers. (50,5)/(10,80) indicates that cost in server1 is 50 and in server2 is 5; load on server1 is 10 and on server2 is 80. The difference in Load factor with CD and exponential cost functions is evident from the graph.

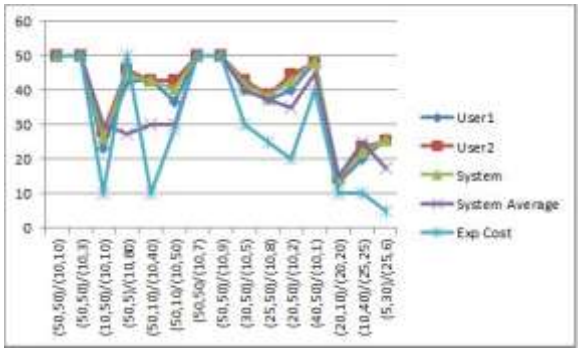

Fig-6: Line graph of average service cost per user, across the system and system average unit cost

Similarly, Fig. 6 plots the average service cost per user, unit cost across the system and actual average system cost for different combinations of load and fixed cost of the system. This chart also plots average service cost per user when exponential cost function is used. The exponential cost function allocates user requests on any one of the servers. The allocation is based on the server having least of sum of fixed unit cost and load.

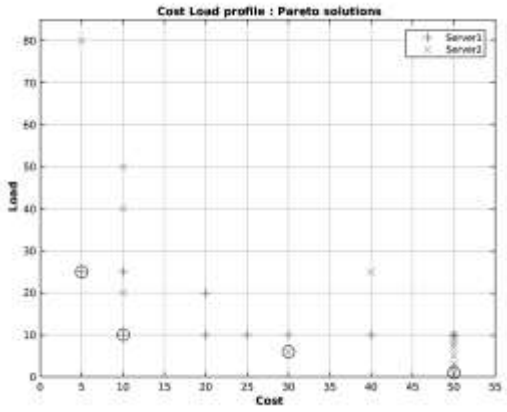

Fig-7: Pareto optimal(Load, Cost) pairs

In the case, the data is given i.e. (cost, load) candidates, since there are only two parameters, one can identify the Pareto optimal solutions just by looking at the plot in Fig. 7. Server1 and Server2 candidates considered together. In the plot, the Pareto optimal solutions are circled. These are: (Cost, Load) pairs (10,10), (5,25), (50,1) and (30,6). And in the simulation environment, these servers were considered on priority for resource allocation.

Further, it is established that Cobb-Douglas production function is convex for the possible set of input values of the allocation problem (Section2.2). It is also proved that the cost minimization is achieved when $\alpha + \beta > 1$

In Essence, this is an aggregate ranking problem, where ranking on two separate parameters, load and cost need to be merged into a single ranking based on load and cost both, which should turn out to be a reasonably efficient strategy for the service provider. Given the set of (cost, load) candidate solutions, it seems that identifying the Pareto optimal solutions is the best thing to be done. Assume, there exists a fundamental set N of well-defined alternatives i.e. load-cost pairs, but only a subset of the alternatives are feasible. The standard allocation problem in our case is to be able to estimate if a given allocation is efficient. However, this implies that, it is not dominated /beaten by any other allocation that can actually be achieved with the existing resources. This is a classic Pareto optimality formulation.

7. Prediction and Forecasting:

Linear regression is an approach for modeling the relationship between a dependent variable y and one or more explanatory variables denoted by x. When one explanatory variable is used, the model is called simple linear regression. When more than one explanatory variable are used to evaluate the dependent variable, the model is called multiple linear regression model. Applying multiple linear equation

model to predict a response variable y as a function of predictor variables x_1 and x_2 takes the following form.

$$y = b_0 + b_1 x_1 + b_2 x_2 + e \tag{19}$$

Here, x_1 and x_2 are Load and Fixed unit cost respectively, y is the response variable which determines the allocation, b_0, b_1, b_2 are regression coefficients and e is the error term.

The values for fitting are taken from the simulation to measure the goodness of fit and forecasting efficacy.

Given the sample (x_{11}, x_{21}, y_1) to (x_{1n}, x_{2n}, y_n) of n observations, the model consists of following n equations.

$$y_1 = b_0 + b_1 x_{11} + b_2 x_{21} + e_1 \tag{20}$$

$$y_2 = b_0 + b_1 x_{12} + b_2 x_{22} + e_2$$

$$\vdots$$

$$y_n = b_0 + b_1 x_{1n} + b_2 x_{2n} + e_n \tag{21}$$

So we have,

$$\begin{pmatrix} y_1 \\ y_2 \\ . \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{21} \\ 1 & x_{12} & x_{22} \\ . & . & . \\ 1 & x_{1n} & x_{2n} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ . \\ b_n \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ . \\ e_n \end{pmatrix}$$

i.e. in matrix notation, $y = Xb + e$ (22)

Allocation of variation:

$$SS0 = n\bar{y}^{-2} \tag{23}$$

$$SST = SSY - SS0 \tag{24}$$

$$SSE = y^T y - b^T X^T y \tag{25}$$

$$SSR = SST - SSE \tag{26}$$

Where,

SSY =sum of squares of Y

SST =total sum of squares

$SS0$ =sum of squares of y

SSE =sum of squared errors

SSR = sum of squares given by regression

Coefficient of determination

$$R^2 = \frac{SSR}{SST} = \frac{SST - SSE}{SST} \tag{27}$$

Coefficient of Multiple correlation

$$R = \sqrt{\frac{SSR}{SST}} \tag{28}$$

Table 9: Multiple Linear Regression Results

	K=1	K=4.3
Number of Observations	22	22
Error Degree of freedom	19	19
R-squared	0.935	0.896
Adjusted R-Squared	0.881	0.885
p-values	$3.252e^{-12}$	$4.78e^{-10}$

8. Conclusion

The work proposes resource allocation model using Extensive Form Game (EFG) and Cobb-Douglas production function. The allocation policy, contrary to the plethora of strategies available in the literature, is not based on a single-point measure and therefore is devoid of single-point failure. Results support that game theoretic approach and Cobb-Douglas production function together provide an optimal solution for resource allocation problem in cloud.

Elasticity are computed based on Fixed unit cost and Load using gradient descent method that ensures optimal average service cost per user. This model ensures minimal load imbalance among the servers after the resource allocation. When there is difference in Fixed unit cost and Load among the servers, results in Table 5 shows that the Load balancing factor after the allocation is always < 1. The cost incurred by SP for unit resource usage is less than the average unit cost across the system when load among the servers is same as illustrated in Table 7. Cost incurred per unit resource usage is less than the maximum cost of any single server when the load among the servers is different.

Results from Multiple Linear Regression model establishes that the proposed model is a reasonably good fit with R^2 close to the upper bound, 1 for both conditions K = 1 and K = 4.33. The regression model is a further empirical testimony to the choice of elasticity adopted in the incubation stage of the model. With values of the elasticity matching our heuristics and theoretical guarantees for optimal conditions, the predictive model is able to achieve very good fitting with respect to both types

of constant technological progress estimation K=1 and K=4.33. The resource allocation strategy adopted by our model, essentially a two-pronged decision theoretic model, is remarkably similar to the simulation results as reported in Table 5 and Table 6, Table 7 and Table 8. Pareto Optimality conditions further bolstered the strategy by indicating the best pairs in load and cost for allocation. The authors wish to stress on the fact that cost computation is done by formulating cost as a function of Load and Fixed unit cost of the hardware which renders the decision function (cost) as a bi-variate function. This, in turn, endows the decision function with a dynamic handle of two parameters, instead of one which is often done in literature. Cobb Douglas formulation, implementation and theoretical analysis, throughout the paper including the appendices, demonstrate this beyond reasonable doubt.

Future allocation strategies in a scaled up environment could benefit from any of the below:

1. Cost factor can be taken into consideration by the vendor/SP provided he/she has a lot of resources to allocate from. Assume the SP has resources $r_1, r_2 ..... r_n$ with the load and cost built in as features of each server/resource, we could do a hierarchical clustering of the resources by using Euclidean distances and allocate jobs to clusters/member of a cluster based on the hierarchies; that way the allocation decision is taken based on features, load and cost.

2. Recalculate the load on the system at each step of backward induction. In the proposed work, initial load on the system is considered for cost computation. Solution can further be enhanced by calculating the load on the systems after allocation for each user and before selecting best option for the next user.
3. Explore and exploit functionally greater flexible forms for 2 input variables CES production function, another member in the family of Cobb Douglas production function works beautifully for 2 input parameters only, sometimes even better than Cobb Douglas in terms of subtle handling of curvature fluctuations. Marginally better result in cost and load optimization is theoretically possible and begs investigation in the context of the present manuscript.
4. Generalize the model to accommodate more than 2 input variables. Proposed work is based only on the interaction between system load and Fixed unit cost of the server for the cost computation. The Cobb-Douglas production function used in this exercise is scalable and it can be modified to accommodate more than two input parameters. The corresponding formulation with n parameters, n > 2 is proved in (Refer 23, 2017)**.** An immediate consequence of the theorem cited is the possibility to find optimal resource allocation in cloud based on more than 2 input parameters. Cobb Douglas production is equipped to handle more than two input parameters without experiencing curvature violation and the conditions for minimum(global) could be obtained in an inductive manner .

9. Algorithm:

9.1. Algorithm to compute $\alpha \& \beta$ using Gradient Descent method:

$$\frac{\partial c}{\partial \alpha} \leftarrow F^{\alpha} L^{\beta} \ln(L)$$

$$\frac{\partial c}{\partial \beta} \leftarrow F^{\alpha} L^{\beta} \ln(F), r \text{ e } peat$$

$$\alpha_{n+1} \leftarrow \alpha_n - \delta \frac{\partial c}{\partial \alpha}$$

$$\beta_{n+1} \leftarrow \beta_n - \delta \frac{\partial c}{\partial \beta}$$

$$\alpha_n \leftarrow \alpha_{n+1}$$

$$\beta_n \leftarrow \beta_{n+1}$$

Until

$$(\alpha_{n+1} > 0) \,||\, (\beta_{n+1} > 0) \,||\, (\alpha_{n+1} + \beta_{n+1} > 0)$$

9.2. Algorithm to find Optimum Allocation Combination at Equilibrium (Algorithm1)

$FindAllocation\_\max Selections \leftarrow 3$

$read(numberOfUsers, numberOfServers)$

$For\_i \leftarrow 1, numberOfServers$

$read(loadonServers[i])$

$read(FixedCostonServer[i])$

End_for

$For\_i \leftarrow 1, numberOfServers$

$compute(loadBalancingFactor[i])\_from(3)$

End_for

$For\_j \leftarrow 1, numberOfUsers$

$read(DemandByUser[j])$

End_for

$For\_j \leftarrow 1, numberOfUsers$

$allocPerUser[j] \leftarrow Call\_FindAllAllocationCombinationDemandByUser[j],$

$numServers, Call\_ComputeCostPerCombinationallocPerUser[j],$

$LoadBalancingFactor, FixedCostonServers, numServers,$

$sort\_allocPerUser\_based\_on\_\min imumCost$

$FinalSelection[j] \leftarrow select\_\max Selection\_from\_allocPerUser$

End_for

$For\_j \leftarrow 1, numberOfUsers$

$finalCombination[j] \leftarrow Call\_GenerateCombinationsFinalSelection[j]$

End_for

$utilitycombinations \leftarrow Call\_ComputeUtilityFinalCombination, numUsers$

$equlibriumCombination \leftarrow utilityCombinations[1]$

In the above algorithm, function FINDALLALLOCATIONCOMBIS() finds all possible allocation combinations for each user based on user demand and available servers. The function GENERATECOMBINATIONS(), finds all combinations of allocations possible for all users together on available servers.

9.3. Algorithm to compute cost incurred by a given combination (Algorithm 2)

$Compute\_\cos tPerCombinationallocPerUser[\,j\,],$

$loadBalancingFactor, FixedCostonServers, numServers$

$For\_i \leftarrow 1, len(allocPerUser)$

$For\_\ j \leftarrow 1, numServers$

$State\_(\alpha, \beta) \leftarrow Call\_Gradient\_Descent$

$FixedCostonServers, LoadBalancingFactor,\ j$

$State\_\cos t \leftarrow (allocPerUser[i][\,j\,]) \times (\ fixedCostonServer[\,j\,]^{\alpha})$

$times(loadBalancingFactor[\,j\,]^{\beta})$

End_For

End_For

9.4. Algorithm to compute utility of the selected combination(Algorithm 3)

$compute\_UtilityFinalCombinations, numUsers$

$State\_totalCost \leftarrow 0$

$For\_i \leftarrow 0, numUsers$

$State\_totalCost \leftarrow totalCost + \cos t[i]$

End_For

$For\_i \leftarrow 0, numUsers$

$State\_utility[i] \leftarrow (\cos t[i] \div totalCost)$

End_For

$For\_i \leftarrow 0, numUsers$

$State\_utility[i] \leftarrow (-1) \times utility[i]$

End_For

References

1. http://aws.amazon.com/ec2/, Retrieved on January 17,2017-Amazon elastic compute cloud (ec2).
2. http://appengine.google.com/, Retrieved on January 17,2017-Google app engine
3. V. Jalaparti, G. D. Nguyen, I. Gupta, and M. Caesar (2010), Cloud resource allocation games, University of Illinois, Tech. Rep.2010.
4. Albert Greenberg, James Hamilton, David A. Maltz, Praveen Patel, The Cost of Cloud: Research Problems in Data Center Networks, Computer Communication Review, 39(1): 68:73 Microsoft Research, Redmond WA
5. X. Xu and H. Yu,A game theory approach to fair and efficient resource allocation in cloud computing, Mathematical Problems in Engineering, 2014, p. 14.

6. A. Nahir, A. Orda, and D. Raz, Workload factoring with the cloud: A game-theoretic perspective, IEEE INFOCOM, 2012, p. 25662570.
7. Toosi A, Vanmechelen K, Kotagiri R n rao and Buyya R(2015) Revenue Maximization with Optimal Capacity Control in Infrastructure as a Service Cloud Markets, IEEE transactions on Cloud Computing, Vol 3, No 3, July-September 2015.
8. Zhang Y, Fu X, K Ramakrishnan: Fine-Grained Multi-Resource Scheduling in Cloud Data Centers. ACM SIGCOMM Computer Communication Review - SIGCOMM '09, Volume 39 Issue 4, 2009 Pages 51:62
9. Zhao L, Lu L, Jin Z and Yu C(2015) Virtual Machine Placement for Increasing Cloud Providers Revenue IEEE Transactions on Services Computing, no. 1, pp. 1, PrePrints PrePrints, doi:10.1109/TSC.2015.2447550
10. Wei G, Vasilakos A, Zheng Y, and Xiong N. (2010)A game- theoretic method of fair resource allocation for cloud computing services vol. 54, 2010, pp. 252–269.
11. Rao N., Poole S, Ma C, He. H, Zhuang J., and Yau. D (2012) Cloud computing infrastructure robustness: A game theory approach, Proceedings of the International Conference on Computing, Networking and Communications Maui Hawaii, January 2012.
12. Niyato. D, Vasilakos. A, and Kun. Z, (2011) Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach, Newport Beach, May 2011, pp. 215–224.
13. Osborne. M, An Introduction to Game Theory. New York: Oxford University Press, 2004.
14. Davis M.D, Game Theory, A Nontechnical Introduction. Dover Publications, 1983.
15. Cobb, C. W and Douglas, P. H (1928), A theory of production, American economic review, 1928, pp. 139–165.
16. McKelvey, Richard D., McLennan, Andrew M., and Turocy Theodore L. Gambit: Software tools for game theory, 2014. Retrieved on January 17,2017 from: http://www.gambit-project.org
17. Mokhtar S. Bazaraa, Hanif D. Sherali, and C.M. Shetty Nonlinear programming: theory and algorithms, 3rd ed. Wiley-Interscience, 2006
18. Saha. S Ordinary Differential Equations: A Structured Approach. Cognella publishing, 2011.
19. Dwivedi. N, Dwivedi. A, Saha. S, Mathur. A, and Ginde. G, (2015) Jimi, journal internationality modeling index-an analytical investigation. 4th National Conference on Scientometrics and Internet Of Things (IoT), pp. 40–49.
20. How to deliver saas using iaas. Retrived on January 17,2017 from http://www.rightscale.com/blog/enterprise-cloud-strategies/how-deliver-saas-using-iaas
21. Google vs. AWS pricing:Google cuts are first of 2015. Retrieved on January 17,2017 from http://www.rightscale.com/blog/cloud-cost-analysis/ google-v's-aws-pricing-google-cuts-are-first-2015
22. Wu. L, Garg. S, and Buyya R(2014) SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments 2011, pp. 195–204.
23. Cobb-Douglas production function. Retrieved on January 17,2017 from http://economicpoint.com/ production-function/cobb-douglas
24. Saha. S, Sarkar. J, Dwivedi. N, Dwivedi. A, Anand MN, and Roy. R (2016) A novel revenue optimization model to address the operation and maintenance cost of a data center in the Journal of Cloud Computing, vol. 5, no. 1, January 2016, pp. 1–23.
25. Jain, Raj, The art of computer systems performance analysis. John Wiley and Sons, 2008.
26. D. Ardagna, B. Panicucci, and M. Passacantando (2011), "A Game Theoretic Formulation of the Service Provisioning Problem in Cloud Systems," in WWW ACM, pp. 177–186
27. Fei. T and F. Magoules(2010):Resource Pricing and Equilibrium Allocation Policy in Cloud Computing, in Proc. IEEE International Conference on Computer and Information Technology (CIT 2010), pp. 195-202.
28. Nezarat A, Dastghaibifard G,(2015): Efficient Nash Equilibrium Resource Allocation Based

on Game Theory Mechanism in Cloud Computing by Using Auction(2015). Xia C-Y, ed. PLoS ONE;10(10):e0138424. doi:10.1371/journal.pone.0138424
29. G Arun Kumar, Arvind Sundarensan, Snehanshu Saha, Bidisha Goswami, Shakti Mishra - A Novel Probabilistic strategy for Delay corrected allocation in shared resource systems. CIT, volume 2, 2017.