Efficient Distributed Algorithm for Scheduling Workload-Aware Jobs on Multi-Clouds

Seyed Ali Miraftabzadeh[1], Paul Rad[2] and Mo Jamshidi[3]

Electrical and Computer Engineering Department, University of Texas at San Antonio
San Antonio, Texas, USA

1 Ali.Miraftab@utsa.edu

2 Paul.Rad@utsa.edu

3 moj@wacong.org

Abstract—**Dynamic distributed algorithm for provisioning of resources has been proposed to support heterogeneous multi-cloud environment. Multi-cloud infrastructure heterogeneity implies the presence of more diverse sets of resources and constraints that aggravate competition among providers. Sigmoidal and logarithmic functions have been used as the utility functions to meet the indicated constraints in the Service Level Agreement (SLA). Spot instances as the elastic tasks can be supported with logarithmic functions while the algorithm always guaranteed sigmoidal functions have the priority over the elastic tasks. The model uses diverse sets of resources scheduled in a multi-clouds environment by the proposed Ranked method in a time window "slice". The paper proposes multi-dimensional self-optimization problem in distributed autonomic computing systems to maximize the revenue and diminish cost of services in the pooled aggregated resources of multi-cloud environment.**

***Index Terms*—Cloud Computing, Scheduler, Multi-Clouds, Federated Cloud, Spot Instances.**

I. INTRODUCTION

According to a forecast from International Data Corporation (IDC) [1] the worldwide spending on public cloud services is expected to surpass $107 billion in 2017 [2]. Among different forms of delivering cloud services, IDC recognized the Infrastructure as a Service (IaaS) model as one of the fastest growing categories with compound annual growth rate of 27.2%. IaaS is a promising solution for enabling on-demand access to an elastic pool of configurable and virtual computational services (e.g., computing power, storage, and networks) in a *pay-as-you-go* manner. IaaS providers offer computational services in the form of Virtual Machine (VM) with specific resource characteristics such as computing power, memory, disk space and networking bandwidth along with types of operating systems and installed applications. Despite traditional distributed computing systems such as Grid [3], cloud computing focuses on the monetary focal point while promising flexibility and high performance for users and cost-efficiency for operators. Furthermore, taking advantage of the virtualization technology gains more resource utilization and *Return On Investment* (ROI) [4].

This work was supported by Grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD and by Open Cloud Institute at University of Texas at San Antonio, Texas, US.

In addition, cloud computing is increasing its penetration among industry, research institutes, and universities across applications. An increasing amount of computing is now performed on public clouds, such as Amazon's EC2, Windows Azure, Rackspace, and Google Compute Engine [5-7] or on private clouds hosted by enterprise using open source OpenStack [8] software. In addition, numbers of cloud computing research test beds and production sites such as Chaemelon or JetStream have been supported by National Science Foundation (NSF) to support academic research [9].

As a drastic increment of the customers, a variety of requested types and emerging cloud computing providers profit maximization for customers and providers becomes a more entangled problem in terms of minimizing the cost using resource management [4, 10], performance improvement in networking and applications [11, 12], maximizing request revenue by providing multi price services [13], market segmentation [14] and prognostication the demands [15].

Considering the natural perishable characteristic of the cloud resources produces a primary conclusion to maximize the resource utilization and to avoid wasting the non-storable cloud resources. On the other hand, honoring the associated Service Level Agreement (SLA) which guarantees the certain level of Quality of Service (QoS) imposes another robust constraint and increases the complexity of the problem. A topic of recent interest, the federated cloud and multi-cloud deployment, allows different cloud providers to share resources for increased scalability and reliability. The aim of the federated cloud is to integrate resources from different providers in a way that access to the resources is transparent to users and cost can be reduced [16]. Furthermore, the federated cloud is a possible mechanism for maximizing a provider's profit by leasing part of the available resources in the pool computing center for the requested bids by other providers [16-18]. So the federated cloud implicitly maximizes the resource utilization while keeping the required QoS.

Maximizing the resource utilization requires the resource allocation decision which involves determining what, how many, where, and when to make the resource available to the user [19]. Typically, users choose the type and amount of the cloud resources and providers place the petition for specific ones onto nodes in their datacenter. Running the application efficiently entails matching the type of the resources and the workload characteristics. In addition, the amount should be

sufficient to meet the indicated constraints in SLA. In a *pay-as-you-go* environment like the cloud where users can request or return resources dynamically, time scheduling such as an adjustment is also important to consider. To maximize resources' utilization the job scheduler is responsible for allocating desired resources to an individual job and in parallel it should be guaranteed that each job is given an adequate amount of resources, or its fair share. Such a scheduling decision becomes more complex in the cloud computing as a heterogeneous environment with two players: cloud providers and cloud users. On one side, there are the users who have fluctuating loads with a variety of applications and requests. On the other side, consolidated cloud environments are likely to be constructed from a variety of machine classes, representing different options in configuring computing power, memory and storage [20]. The federated cloud is an exceedingly multi-parts heterogeneous system as a result of sharing diversity sets of datacenters in terms of technical performance and monetary strategies.

Mathematical interpretation of effectively allocating datacenters' resources in the federated cloud is a multi-dimensional self-optimization problem in distributed autonomic computing systems. As a bridge between providers and users, utility functions have very smart theoretical properties and their use in practical autonomic computing systems started to explore in [21] and [22]. Utility functions can enable a collection of autonomic elements to continually optimize the use of computational resources in a dynamic, heterogeneous environment [23].

There are a lot of works that have been done for two major cloud platforms' advantages: producing the flexible available resources and minimizing the cost of the datacenters [24], [25], [26]. Nevertheless, relatively less work has been done on the provider's sideways to maximize revenue and diminish cost of services produced. In this paper, based on the *Sigmoidal* and *Logarithmic* utility functions the proposed algorithm guaranteed the dynamic and scalable job scheduling in the heterogeneous federated cloud environment for the most flourishing cloud service, IaaS. In parallel, the algorithm guaranteed the minimal management effort or service providers' interaction. The algorithm can rapidly provide and release federated cloud pool resources at once period of processing decision.

II. Ranked Method in Heterogeneous Federated Cloud

Two main stakeholders in the cloud computing environment, cloud providers which produce various services and cloud customers which are client or consumers of the produced services, have their own inevitable momentously operative influence to make the cloud computing environment as a heterogeneous system. Particularly, heterogeneous cloud computing datacenter has been consider in this paper.

Collaboration across the federated cloud requires strapping monitoring and management mechanism which are the key components for provisioning, scheduling and failure management [27]. Efficiently provisioning computing resources in a pool of heterogeneous datacenters has been done by proposing Ranking Method in this paper. Producing a dynamic sequence of the available resources based on the hierarchical characteristics of the cloud providers is the first step of the proposed algorithm. The dynamic sequence could be matched with dynamic pricing which it has been acknowledged by the literature [28], [16]. Since then the model could be considered as one of the dominant methods to maximize the request revenue and consequently get the most out of the resource utilization. The dynamic pricing can be explored by using intermediate parameter as Bids between customers and cloud providers. In addition, The Ranking Method can be administrated based on most instantaneous parameters, in [29] more detail for effective parameters are described. How the Ranking Method can be classified to be implemented is out of this scope and it can be addressed in another paper. What is important for using the method are the properties, as follow:

- similar instances of different providers are situated in the same group (which is named clutch in this paper)
- initially top ranked instances have the uppermost slots and bottom ranked instances have the lowermost slots
- for customer satisfaction, evaluation of the requested service is placed in a specific group based on the lowest and highest requested instances; interpolation and other methods can be used
- before fully occupied resources, top ranked instances are allocated to the corresponding request
- after fully occupied resources, top ranked instances may not be allocated for the request but another ranked instance in a same clutch has more opportunity to be allocated to the task. This action strictly depends on the assigned utility function to the request.
- transcendent ranked instances ameliorate customer satisfactions for the assigned service
- transcendent ranked instances are unintended to be occupied if the number of demand increases
- inferior ranked instances keep the request revenue in order to maximize the cloud revenue
- instances in a same clutch can be reshuffled to maximize the resource utilization if there is profit for the corresponding providers
- hedging from clutch to the adjacent clutch is possible in a case of resource shrinking and strictly depends on the utility function
- each clutch could include different providers

Referring Fig 1., how the proposed algorithm gets advantage of the sigmoidal functions will be explained. It includes four clutches in which five providers make a pool of ranked instances as small, medium, large and extensive large. Six unalike sigmoidal utility functions corresponding to the six types of customers or services are shown in Fig 1. Utility functions 1 and 2 belong to the small clutch in which provider number 5 (P5) has the highest priority to serve any request in the clutch. In this clutch customers who are assigned to utility

function 2, rather than 1, has more chance to take the better service based on the chosen ranked parameter.

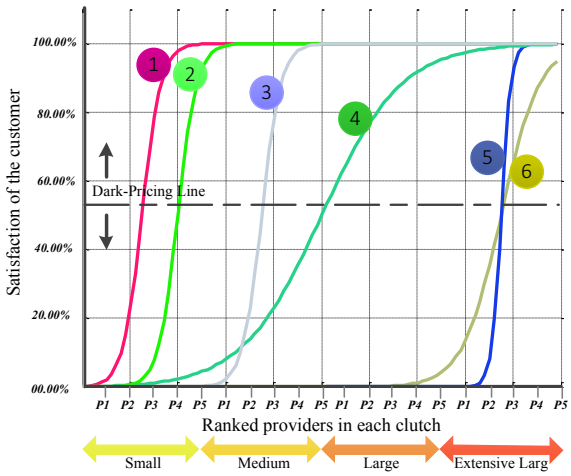

Fig 1. Illustration of Ranking Method

However, utility function 1 gets services simultaneously from P5 if enough resources are available in the federated cloud. By increasing demands and reducing the number of free instances, the chance of getting services from inferior providers increases. The Ranking Method tries to provision resources for all of types of the customers. However, by increasing the number of requests for a specific provider or clutch, based on the Service Level Agreement (SLA), three scenarios are likely to be happened. First, pass the customer to the lowermost neighbor provider or lower ranked provider. Chosen desired sigmoidal function by the customers carries on this scenario automatically. Second, the customer may lose allocated instances if it does not satisfy the provider proposed conditions such as raising the price. This scenario can be implemented by control the whole number of available resources, $(V - z)$, in each processing period. Finally, the customer accept the proposed condition and the allocated instances are remained to serve the customer. Utility function 3 describes the only service in the medium clutch. It spread over the whole clutch from P1 to P5; so each of the serving providers can provision the customer. For large clutch only utility function 4 is presenting services but it spreads not only in large clutch but also from small clutch to extensive large clutch. Consequently, customers in this utility function have a potential to get serve in a variety clutches; the customers have a chance to have an experience in extensive large clutch when enough pooled resources are available. Apparently, it is not a case when federated cloud has an experience in high amount request. This types of utility function expansively helps a lot to maximize the revenue of the requests by providing motivation such as paying less for this types of functions. Finally, in extensive clutch two types of utility functions are managing the requests revenue and resources of the providers. As it is illustrated utility function 5 has the sharpest slop in this simplified properties illustration of Ranking Method. This types of utility function can be fixed in a specific provider with two application: first, serves a customer by a specific ranked or no ranked properties over a time. Next, fixed the instance of the provider for a service after the customer accepts the dynamic conditions of the provider such as dynamic pricing.

So far, it is described that how the Ranking Method provides flexibility to serve the customers dynamically, by applying some principle conditions in the SLA to match the customer requests and utility functions. Maximizing the profits of the cloud providers, as one of the primary goals of the federated cloud, makes the Ranking Method as a method to potentiate the dynamic pricing conditions in the SLA. In the next step mathematical formulation would be studied and the Ranking Method is presented as the distributed algorithm in the federated cloud.

III. Mathematical Formulation of the Algorithm

In this paper, we will present design and implementation of the distributed algorithm of optimized controller for resource allocation of elastic and inelastic traffic using logarithmic and sigmoidal-like utility functions. This algorithm is based on the utility proportional policy to maximize the requests revenue where the fairness among users is in utility percentage and describes the user gratification with the service of the corresponding bid. Furthermore, this algorithm precisely provides information of the available resources and distributes them among the users. It could be implemented as a part of the cloud service brokerage or as a separate unit in cloud computing architecture of IaaS.

Mathematical formulation of the algorithm includes *Sigmoidal* [30] and *Logarithmic* [31] utility functions which have had applications in resource scheduling. These are used as gauges for dynamically resources allocation such that predefined required constrains by SLA are satisfied and cloud providers get the benefit of using all their available resources and minimizes the operating costs. Virtual data center (vDC) is considered as the resource unit; however, the approach can straightforwardly be extended to consider a variety kinds of instances which include different amount of compute, memory, storage and bandwidth resources.

The vDC allocated by controller to the k^{th} service is given by v_k. Presumption of the algorithm is suitable utility function is assigned to the service by using techniques such as interpolation, statistic methods or others. Our objective is to determine vDC that the scheduler should allocate to the specific customer. For this optimization problem we are looking for the strictly concave utility functions that satisfies the following properties:

(a) $U_k(v_k)$ is an increasing function of v_k.
(b) $U(0) = 0$ and $U(\infty) = 1$.
(c) $U_k(v_k)$ is twice continuously differentiable in v_k.
(d) U_k is bounded above.

Note that typically, most utility functions in current networks can be represented by three types of functions:

- Sigmoidal-like
- Strictly concave function
- Strictly convex function

Next, let us consider two major type of tasks: Sigmoidal-like Tasks and Logarithmic Tasks.

- *Sigmoidal-like Tasks*

First type refers to the task that requires the specific amount of resources which are requested by the service. The characteristic of the requested *pay-as-you-go* service match to the suitable sigmoidal function. Usually, inelastic tasks can be defined by sigmoidal function. It can be expressed as:

$$U_k(v_k) = c_k\left(\frac{1}{1+e^{-a_k(v_k-b_k)}} - d_k\right) \quad (1)$$

to satisfy all the required properties it is enough to have $c_k = \frac{1+e^{a_k b_k}}{e^{a_k b_k}}$ and $d_k = \frac{1}{1+e^{a_k b_k}}$. By choices a_k and b_k describe the utility function of the k^{th} user. For example by choosing $a = 5$ and $b = 10$ a good approximation for a step function can be obtained which explains the fixed usage of the allocated resources; for instance, users with the specific requirement of the resources. Normalized sigmoidal-like utility function with $a = 0.5$ and $b = 20$ is another example for representation of the adaptive real-time application.

- *Logarithmic Task*

It is desired to use all resources in a pool of vDCs as much as possible. In a case of temporary real-time services, the same as spot instances, normalized logarithmic utility function describes the good description. It is referred to the mission of which a minimum number of resources are needed and it has a desire to occupy more resources in a case of availability. Mathematical expression of the function is as:

$$U_k(v_k) = \frac{\log(1+m_k v_k)}{\log(1+m_k v_{max})} \quad (2)$$

where v_{max} is the point in which the task occupies all available resources as same as what is mentioned in the SLA. For example, if the user is fully satisfied by 30 vDCs, v_{max} should be set to 30. Any allocated vDCs less than 30 leads to the minus satisfaction but it does not mean user is loss its performance since the performance is based on the required resources for doing the specific job. It means if the user is using all 30 vDCs there is no request issues from the user for share the resources to others. But if the user is not fully using allocated resources and it is confirmed by the monitoring and negotiation process then the user can share the resources by management process. In such a case the customer and the provider get the advantage simultaneously. In (2) m_k is the rate of increasing utility percentage with the allocated rate v_k.

Both function types are satisfied the constraints as in (a) to (d). In Fig 2. the normalized sigmoidal-like and Logarithmic examples of utility functions are shown.

To maximize the resource allocation based on the utility function the objective function is defined as follow:

$$\begin{aligned} &\max_{v} \prod_{k=1}^{M} U_i(v_i) \\ &\text{Subject to } \sum_{k=1}^{M} v_k \le R \\ &\qquad v_k \ge 0 \quad i = 1,2,\dots.,M \end{aligned} \quad (3)$$

where $v = \{v_1, v_2 v_3, \dots, v_M\}$ and M is the number of the requests. This formulation ensures non-zero resource provisioning for customers while guarantees minimum predefined constraints are met. Furthermore, using *Sigmoidal* function leads to the more resource allocation for corresponding tasks when the specific amount of resources are

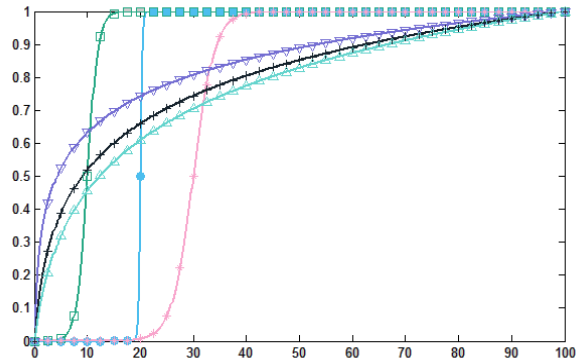

Fig 2. The sigmoidal-like utility functions and logarithmic utility functions

required. Now we do have an optimization problem for proportional resource allocation based on the utility functions [32]. Prove of existence of the global optimal solution for (3) is straightforward. Since (3) is a convex optimization problem a unique trace-back global optimal solution is exist for the problem.

Objective function of $\arg\max_{v} \prod_{k=1}^{M} U_k(v_k)$ is equivalent to $\arg\max_{v} \sum_{k=1}^{M} \log\big(U_k(v_k)\big)$ and for both applied functions, *Sigmoidal* and *Logarithmic*

$$\begin{aligned} &\frac{d}{dv_i} log\, U_i(v_k) > 0 \\ &\frac{d^2}{dv_i^2} log\, U_i(v_k) < 0 \end{aligned} \quad (4)$$

So both functions are strictly concave natural logarithms. Therefore, the optimization problem is a convex optimization problem and there exist the unique trace-back global optimal solution [33].

1) Dynamic Distributed Resource Provisioning Approach

To make the optimization problem as the negotiated process for provision resources dynamically, based on demand and supply, this paper gets advantage from the dual problem to distribute the optimization problem among the federated cloud stakeholders. In communication network it has been done similarly in [34], [35], [32] by defining the Lagrangian function as:

$$\begin{aligned} &L(v,p) = \\ &= \sum_{k=1}^{M} \log\big(U_k(v_k)\big) - p\big(\sum_{k=1}^{M} v_k + z - R\big) \\ &= \sum_{k=1}^{M} \big(\log\big(U_i(r_i)\big) - p v_k\big) + p(R-z) \\ &= \sum_{k=1}^{M} L_k(v_k,p) + p(R-z) \end{aligned} \quad (5)$$

where $z \ge 0$ is the slack variable (we discuss in more detail about the slack variable later) and p is Lagrange multiplier or the dark price which is the intermediate parameter in negotiation between the federated cloud and customers. To match the ranked instances by Ranking Method which is proposed in the paper, dark price is weighted by the ranked order which is called bid in the paper. Bidding strategy attracts a lot of researcher especially for spot instances [36], [37]. By assigning a set of utility functions not only spot instances but

also other tasks can be evaluated simultaneously to make the decision for resources provisioning in the federated cloud. Satisfying the simultaneous scheduling for multi types of tasks the k^{th} bid for the instance can be given by $w_k = pr_k$. Subsequently, The dual problem objective function can be written as:

$$D(p) = \max_{v} L(v, p) \tag{6}$$

and the dual problem is given by

$$\min_{p} D(p) \quad \text{Subject to } p \geq 0 \tag{7}$$

to find the optimal answer driving a derivation of the new optimization problem leads to the interesting formulation for dark price by using $\sum_{k=1}^{M} w_k = p \sum_{k=1}^{M} r_k$ and solving the equation for p:

$$\frac{\partial D(p)}{\partial p} = R - \sum_{k=1}^{M} v_k - z = 0 \qquad p = \frac{\sum_{i=1}^{M} w_i}{R-z} \tag{8}$$

latest equation entails critical information to optimize the problem for p:

- Summation of the bids (from all requests)
- Available Instances (all the active resources)

which both of them are available in the centralized node, monitoring and negotiation process.

On the other hand since the $L(v, p)$ is separable in v we can rewrite the equation as:

$$\max_{v} \sum_{k=1}^{M} \left(\log\left(U_k(r_k)\right) - pv_k\right) = \sum_{k=1}^{M} \max_{r_k} \left(\log\left(U_k(r_k)\right) - pv_k\right) \tag{9}$$

which implies, the optimization problem can be solved for each utility function separately:

$$\max_{v_k} \left(\log\left(U_k(v_k)\right) - pv_k\right) \quad \text{Subject to } p \geq 0 \quad v_k \geq 0 \;\; i = 1,2,\dots,M \tag{10}$$

Equation (7) and (10) distribute the optimization problem (3) into two parts. (7) is in quest of the dark price which requires supply and demands information and can be considered to be solved in the federated cloud. (10) which separately maximizes provisioning of the resources for each utility function and it can be counted up as a customer's attorney. Between these two optimization problems dark price is an intermediate parameter to moderate the criteria for both sides. Establish an iterative process to drawn out the settle down point in which stakeholders get the mutual advantage is the chosen approach of the algorithm. The mutual proliferation leads to the maximization of the request revenue and utilization of the pooled vDCs in federated cloud environment.

IV. Proposed Algorithm for Resource Scheduling

So far, the reasons and requirements for implementation of the expedient scheduling algorithm have been studied in various aspects. Putting together all the described aspects of the proposed algorithm has been projected in two mutually joint algorithms, Back shown in Algorithm (1) and Feed in Algorithm (2). In this section terms explanation and mathematical formulation of the algorithms have been described. Dark Price is considered as the iterative parameter between Back-Algorithm which is the customers' attorney and Feed-Algorithm which is the federated cloud lawyer. Feedback process will continue to settle down in profitable point for customers and providers in the pool of vDCs. Each active tasks send the initial bid to the broker. The broker makes a decision based on the difference of two consequent bids per task with the pre-specified gage as δ. If the absolute difference get greater than δ the broker specify the new shadow price based on the utility function type. Each user receives the shadow price and solve its own optimization problem $v_{k(n)} = \max_{v_k} \left(\log\left(U_k(v_k)\right) - pv_k\right)$ for v_k which is used to calculate the new bid $w_k(n) = p(n) r_k(n)$. The broker collects all the bids sequentially and the process would be repeated until $|w_k(n) - w_k(n-1)|$ gets less than the pre-specified threshold δ. The Back and Feed are the same as UE and eNodeB algorithm in [32]. There are modifications which are applied in the algorithm to match them with Ranked Method; scaling factor as $(V - z)$ to make the algorithm as an applicable process in the proposed Ranked Method.

Algorithm 1 Back (k^{th} *Utility Function*)

Initiate its bid as $w_k(1)$ to Feed
loop
  Received shadow price $p(n)$ from the Feed
  if STOP from Feed **then**
    Calculate allocated rate $v_n^{opt} = \frac{w_k(n)}{p(n)}$
    STOP
  else
    Solve $v_k(n) = \max_{v_k} \left(\log\left(U_k(v_k)\right) - pv_k\right)$
    Send new bid $w_k(n) = p(n) r_k(n)$ to Feed-Algorithm
  end if
end loop

Algorithm 2 Feed

loop
  input $V - z$ and scale the functions by $(V - z)^{-1}$
  if kth utility function is the active
  Receive bids $w_k(n)$ from Back-Algorithm {Let $w_k(0) = 0 \; \forall_i$}
  if $|w_k(n) - w_k(n-1)| < \delta \; \forall_k$ then
    Allocate rates, $v_k^{opt} = \frac{w_k(n)}{p(n)}$ to kth Utility Function
    STOP
  else
    Based on the specified group for p_k:
      Calculate $p(n) = \frac{\sum_{k=1}^{M} w_k(n)}{V-z}$
    Send new shadow price based on the users category
  end if
end loop

V. Simulation Results

Ranking method is applied to the variety sets of utility functions. Convergence of the proposed algorithm is

acknowledged by the simulation. In the simulations, to show the generality of the algorithm, it is supposed that instances of the providers are ranked from 1 to 100 which they are included in four clutches with the same size. So one can interpret each number to the instances with the specific parameters. In the first simulation, six utility functions corresponding to sigmoidal functions in Fig 1. are simulated. Through classification of the instances it is assumed almost the same instances are put to the same clutch and for clearly illustrate of the algorithm only computing power is studied as the gauge to make distinguish between the instances. However, the algorithm has the potential to study all the instances properties such as computing power, memory, disk space and networking bandwidth factors at one simulation. In this case the classification methods should be used to weight the gauge. In the first simulation the computing power is used as the weighted parameter to provide the gauge. The necessitated manipulation to the algorithm is related to the $(V-z)$ as the maximum available resources. The pool of resources in vDC of the federated cloud is the other explanation of the $(V-z)$ manipulation. In the algorithm V is the maximum number of resources and z is the reserved resources for the specific tasks or customers. Since by increasing $(V-z)$ just the scaling of the functions are varied, the convergence of the optimization problem is not denied. The manipulation which is used for this simulation is as follow

$$(V-z) = \sum_{i=1}^{I} \sum_{n}^{N} C_{i,n} P_i^n \qquad (11)$$

Which I is the maximum number of providers, N is the maximum number of the clutches $C_{i,n}$ is coefficient for the ith provider in the nth clutch and P_i^n is the corresponding provider advantageous over other providers in the lucrative specification terms. The scaling factor is $(V-z)^{-1}$ for all the functions. In the first simulation $P_i^n = 1$ which indicates similarity in the performance of the providers and $C_{i,n} = b_{i,n}$.

Fig 3. describes the convergence of the algorithms by weighting the ranked clutches in the x axis and scaling all the functions based on the available calculated resources. $(V-z) = 240$ is considered with the iterations $n = 20$. The rates of different functions with the number of iteration is illustrated in Fig 3. After giving enough iterations the algorithm is settle down for the sigmoidal functions as 13, 21, 38, 6, 89 and 76 correspondingly. This simulation show for the 4th sigmoidal function the allocated instances is placed in the lowest clutch.

It means the algorithm provides the lowest priority for the 4th function since the 4th function (with $a = 0.12$) is spread throughout the operating boundaries.

In the second case the available resources are decreases to $(V-z) = 100$. From the results in Fig 4. and referring to Fig 1. it can be understood sigmoidal function number 5 is satisfied with the appropriate instances which is matched with its request. However sigmoidal function number 6 is not confined in its primary clutch and it is downgraded to the small clutch. The reason for that is sigmoidal function number 6 has the a value as 0.25 which spreads the function outside of its primary clutch despite of the sigmoidal function number 5 with the a value as 1 which is strictly confined the sigmoidal function in its primary clutch. The same reason is applied for function number 4. Since it spreads widely throughout the operating boundaries it gets even less ranked instance than function number 1, 2 and 3. Finally, sigmoidal functions 1 and 2 get the same instances which means low ranked clutch has the opportunely to serve more customers. It means in a case of having higher requests than the available resources, petitions for the low ranked clutches arise dramatically.

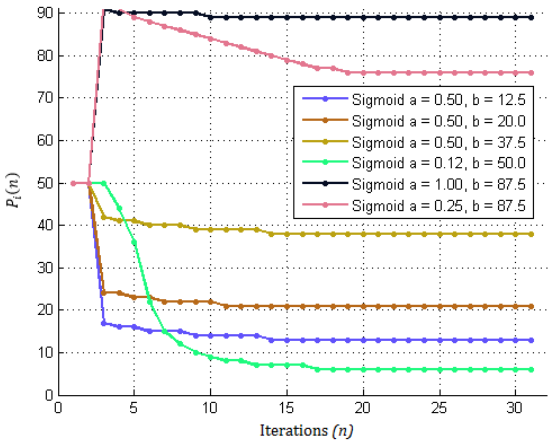

Fig 3. Allocated ranked providers convergence $P_i(n)$ with number of iterations n for $(V-z) = 240$

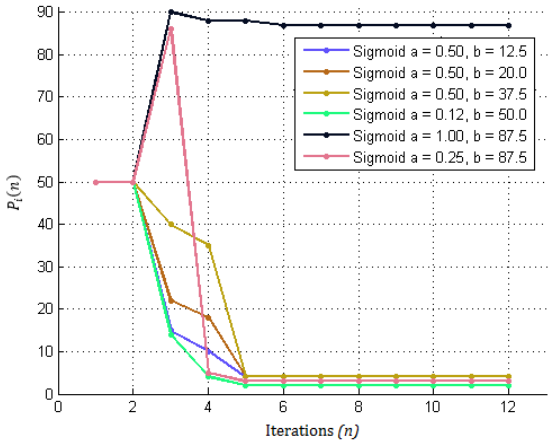

Fig 4. Allocated ranked providers convergence $P_i(n)$ with number of iterations n for $(V-z) = 100$

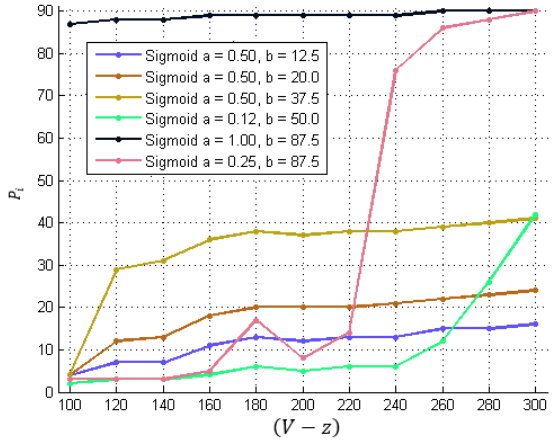

Fig 5. Allocation of ranked providers for $100 \leq (V - z) \leq 300$

In the federated cloud this is the situation which competition between the providers arises to serve more number of customers by offering lower size of instances. Actually when there are enough available resources the algorithm provides the customers by the highest performance resources based on their desired utility functions. However, when the requests is getting higher than the available resources, based on the customers chosen model, customers with bounded utility functions maintain their primary resources with the small variation but the allocated resources for other types of functions are varied to satisfy each chosen model by the customers. In Fig 5. The steady state rate of different sigmoidal functions with different $(V - z)$ has been illustrated As mentioned before monitoring system in parallel with brokering strategies control the situation and decide the final decision. Study the utility function number 4 and 6 is interesting. Since the requested resources by this functions have not been satisfied until the rest functions reach their corresponding requested resources.

A. Implementation of Spot Instances with the Logarithmic Function

The distributed algorithm has the interesting application in spot instances pricing model to control the resource allocation. Spot instances is following the method which impose dynamic price based on supply and demand. Since maximizing revenue of requests is one of the federated cloud aims, a dynamic resource allocation has been simulated. **Error! Reference source not found.**Fig 6. Shows the sigmoidal and logarithmic functions with their characteristics. As **Error! Reference source not found.**Fig 7. shows the distributed algorithm gives the sigmoidal functions higher priority than the logarithmic function. This is because, the algorithm start giving the resources to the logarithmic functions after the steady state rate of all the sigmoidal functions exceed their corresponding inflection points. Furthermore, the majority of resources are allocated to the tasks with sigmoidal functions. To illustrates the application of logarithmic functions for spot instances **Error! Reference source not found.**Fig 8. describes the characteristic of the algorithm; the allocation of resources for logarithmic functions have been carried on when the available resources are more than the summation of the inflection points. This feature of the algorithms has the potential to use as the spot instances dynamic model. In a case of the available provider with the specific feature the instances can be provisioned to the customer of the spot instances and when there is a request of permanent use of the resources on the other specific conditions the spot is return back to the system as the available resource.

VI. Conclusion

The Ranking Method has been introduced to support elastic and inelastic services in the heterogeneous environment of the federated cloud. By using sigmoidal functions, the distributed algorithm dynamic provisioning of the resources is carried on in the proposed Ranking Method. To maximize the usage revenue, in a pool of resources, the combination of logarithmic and sigmoidal functions has been proposed to support the spot instances for the elastic tasks while guaranteed the sigmoidal functions always have the priority; which means the algorithm support the elastic and inelastic applications simultaneously. The convergence of the algorithm is simulated for a variety sets of the sigmoidal and logarithmic functions.

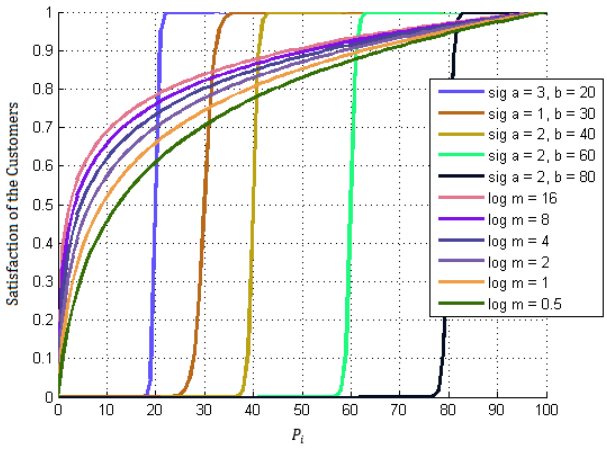

Fig 6. The sigmoidal and the logarithmic utility functions

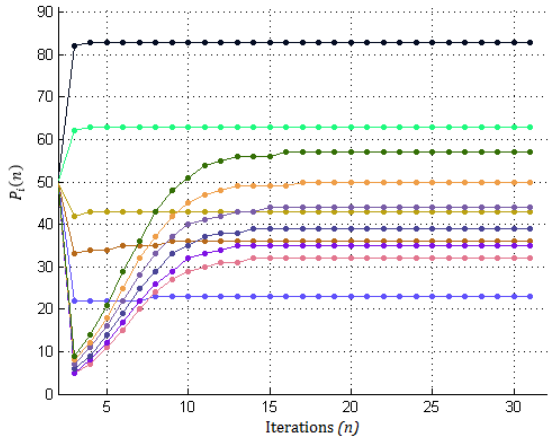

Fig 7. Convergence of the allocation for the ranked providers $P_i(n)$ for sigmoidal and logarithmic functions for $(V - z) = 500$

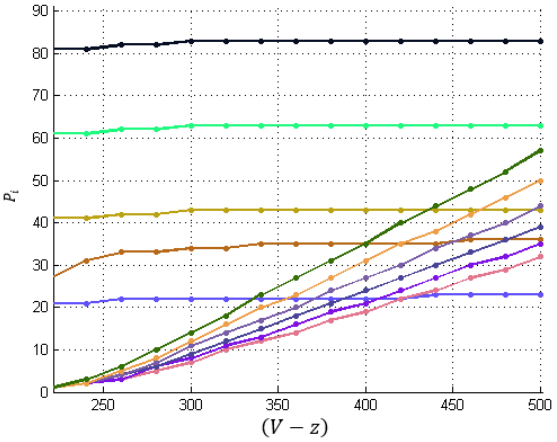

Fig 8. Allocation of ranked providers for sigmoidal and logarithmic functions for $220 \leq (V-z) \leq 500$

VII. REFERENCES

[1] International Data Corporation. http://www.idc.com/.

[2] A. Nadjaran Toosi, "On the economics of infrastructure as a service cloud providers: pricing, markets, and profit maximization," 2014.

[3] I. Foster, and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*: Elsevier, 2003.

[4] A. Beloglazov, and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 24, no. 7, pp. 1366-1379, 2013.

[5] Google Compute Engine. https://cloud.google.com/products/compute-engine/.

[6] Amazon EC2. http://aws.amazon.com/ec2/.

[7] Windows Azure. http://azure.microsoft.com/.

[8] Openstack cloud software. http://www.openstack.org/.

[9] Chameleoncloud. https://www.chameleoncloud.org/.

[10] P. Rad, V. Lindberg, J. Prevost, W. Zhang, and M. Jamshidi, "ZeroVM: secure distributed processing for big data analytics." pp. 1-6.

[11] P. Rad, R. V. Boppana, P. Lama, G. Berman, and M. Jamshidi, "Low-latency software defined network for high performance clouds." pp. 486-491.

[12] M. Muppidi, P. Rad, S. S. Agaian, and M. Jamshidi, "Container based parallelization for faster and reliable image segmentation." pp. 1-6.

[13] A. Gohad, N. C. Narendra, and P. Ramachandran, "Cloud Pricing Models: A Survey and Position Paper." pp. 1-8.

[14] W. Wang, B. Li, and B. Liang, "Towards optimal capacity segmentation with hybrid cloud pricing." pp. 425-434.

[15] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach." pp. 433-441.

[16] M. Mihailescu, and Y. M. Teo, "Dynamic resource pricing on federated clouds." pp. 513-517.

[17] E. Elmroth, F. G. Márquez, D. Henriksson, and D. P. Ferrera, "Accounting and billing for federated cloud infrastructures." pp. 268-275.

[18] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, and J. Caceres, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development,* vol. 53, no. 4, pp. 4: 1-4: 11, 2009.

[19] G. Lee, *Resource allocation and scheduling in heterogeneous cloud environments*: University of California, Berkeley, 2012.

[20] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." p. 7.

[21] A. Byde, M. Sallé, and C. Bartolini, "Market-based resource allocation for utility data centers," *HP Lab, Bristol, Technical Report HPL-2003-188*, 2003.

[22] T. Kelly, "Utility-directed allocation."

[23] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility functions in autonomic systems." pp. 70-77.

[24] L. A. Barroso, "Warehouse-Scale Computing: Entering the Teenage Decade," 2011.

[25] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture,* vol. 8, no. 3, pp. 1-154, 2013.

[26] J. Hamilton. "Cost of power in large-scale data centers," 11; http://perspectives.mvdirona.com/.

[27] M. Kozuch, M. Ryan, R. Gass, S. Schlosser, and D. O'Hallaron, "Cloud management challenges and opportunities." pp. 43-48.

[28] H. Xu, and B. Li, "Dynamic cloud pricing for revenue maximization," *Cloud Computing, IEEE Transactions on,* vol. 1, no. 2, pp. 158-171, 2013.

[29] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection." pp. 558-565.

[30] J.-W. Lee, R. R. Mazumdar, and N. B. Shroff, "Downlink power allocation for multi-class wireless systems," *IEEE/ACM Transactions on Networking (TON),* vol. 13, no. 4, pp. 854-867, 2005.

[31] G. Tychogiorgos, A. Gkelias, and K. K. Leung, "Utility-proportional fairness in wireless networks." pp. 839-844.

[32] A. Abdel-Hadi, and C. Clancy, "A utility proportional fairness approach for resource allocation in 4G-LTE." pp. 1034-1040.

[33] S. Boyd, and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2004.

[34] S. H. Low, and D. E. Lapsley, "Optimization flow control—I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking (TON),* vol. 7, no. 6, pp. 861-874, 1999.

[35] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "Rate control in communication networks: shadow prices proportional fairness and stability, J," *Cluster Comput,* vol. 8, no. 2-3, pp. 135-145, 2005.

[36] Y. Song, M. Zafer, and K.-W. Lee, "Optimal bidding in spot instance market." pp. 190-198.

[37] S. Karunakaran, and R. Sundarraj, "Bidding Strategies for Spot Instances in Cloud Computing Markets," 2014.