Maximizing Profit in Cloud Computing System via Resource Allocation

Hadi Goudarzi and Massoud Pedram
University of Southern California, Los Angeles, CA 90089
{*hgoudarz,pedram*}*@usc.edu*

Abstract—**With increasing demand for high performance computing and data storage, distributed computing systems have attracted a lot of attention. Resource allocation is one of the most important challenges in the distributed systems specially when the clients have some Service Level Agreements (SLAs) and the total profit in the system depends on how the system can meet these SLAs. In this paper, an SLA-based resource allocation problem for cloud computing is considered and a distributed solution to this problem is presented. The processing, data storage, and communication resources are considered as three dimensions in which optimizations are performed. Simulation results demonstrate that the proposed heuristic algorithm is robust (produces high quality solutions independent of the initial solution provided) and produces solutions very close to the "optimum" (best solution found by Monte Carlo simulation).**[1]

I. INTRODUCTION

Demand for computing power has been increasing due to the penetration of information technologies in our daily interactions with the world both at personal and public levels, encompassing business, commerce, education, manufacturing, and communication services. At personal level, the wide scale presence of online banking, e-commerce, SaaS (Software as a Service), social networking and so on produce workloads of great diversity and enormous scale. At the same time computing and information processing requirements of various public organizations and private corporations have also been increasing rapidly. Examples include digital services and functions required by the various industrial sectors, ranging from manufacturing to housing, from transportation to banking. Such a dramatic increase in the computing demand requires a scalable and dependable IT infrastructure comprising of servers, storage, network bandwidth, physical infrastructure, Electrical Grid, IT personnel and billions of dollars in capital expenditure and operational cost to name a few.

The IT infrastructure provided by the datacenter owners/operators must meet various service level agreements (SLAs) established with the clients. The SLAs include compute power, storage space, network bandwidth, availability and security, etc. Infrastructure providers often end up over provisioning their resources in order to meet the clients' SLAs. Such over provisioning may increase the cost incurred on the datacenters in terms of both the electrical energy cost and the carbon emission. Therefore optimal provisioning of the resources is imperative in order to reduce the cost incurred on the datacenter operators as well as the environmental impact. The problem of optimal resource provisioning is challenging due to the diversity present in the client applications being hosted and the SLAs. For example: some client applications may be compute-intensive while others may be memory intensive, some applications may run well together while others do not, etc.

The IT infrastructure provided by the large datacenter owners/operators is often *geographically distributed*. This helps with reducing the peak power demand of the datacenters on the local power grid, allow for more fault tolerance and reliable operation of the IT infrastructure, and even, reduced cost of ownership. A datacenter however comprises of thousands to tens of thousands of server machines, working in tandem to provide services to the clients, see for example [1] and [2]. In such a large computing system, energy efficiency can be maximized through system-wide resource allocation and server consolidation, this is in spite of non energy-proportional characteristics of current server machines [3].

Resource management in distributed systems is one of the most important challenges. Due to more requests for resource in these systems, input size of this problems are much larger than ordinary resource allocation problems in centralized computing systems. There is number of papers discussing the resource allocation in grid computing systems e.g., [4][5]. Due to difference between cloud computing system and grid computing system, it is necessary to address the resource allocation problem in cloud computing systems separately.

In our paper a cloud computing system is composed of some clusters that have different number and possibly different types of servers. The total profit in this system is the total price gained from the clients subtracted by the cost of operating the active servers in the system. Clients in this system are application software that require processing, data storage and communication resources in this "on-demand capacity provisioning" or "lease model of the IT infrastructure" [6]. Clusters and servers are modeled based on these three capabilities: computational, storage, and networking bandwidth. Cost of operation of active servers is related to the degree of their use of these resources [7].

We focus on the SLA based resource allocation in the cloud computing system. Each client in this system is related to a class of clients that have a pre-defined *utility function* based on their response time requirements. To manage the resource allocation problem between two decision times, clients are assigned to only one cluster in each decision epoch. This helps compensate for dynamic changes in the system that do not need another decision at the cloud level and can be handled at a cluster or server level. Although a central resource manager is responsible for the resource management in these cloud computing systems, the local agents are used to parallelize the solution and decrease the decision time. This helps with the scalability of the proposed solution.

The rest of this paper is organized as follows. Related works are presented in the next section. In section III, the system model and problem formulation are presented. The optimization problem and the optimization solution presented for this problem are presented in section IV and V, respectively. The simulation results are presented in the section VI and the conclusions are on the last section.

II. RELATED WORK

The distributed resource allocation problem is one of the most challenging problems in the resource management problems. The SLA based distributed resource allocation has attracted attention of the research community in the last years.

Our paper considers the resource management problem in a cloud computing system. Key features of our formulation and subsequent proposed solution are that we:

- Consider heterogeneous clusters in number of servers and type of servers deployed in each cluster

[1] This work is sponsored in part by grant from National Science Foundation (NSF).

- Use a three dimensional model of the resources in the clusters, i.e., computational, storage and networking capabilities, and
- Perform distributed decision making to reduce the decision time by parallelizing the solution

No previous work considers all these aspects together when addressing the cloud level resource management problem. In the following we provide a review of most relevant prior work.

Srikantaiah et al. [7] presented energy aware consolidation to decrease the total energy consumption of the cloud computing system. The authors presented an experimental method to model the energy consumption of the servers based on the CPU and disk utilization. Based on this method, a simple heuristic to consolidate the processing works in the cloud computing system is presented.

In [9], the authors extend the work in [8] and present a problem statement with clients that have discrete utility functions. The authors proposed a heuristic to solve the problem of assigning different client classes to different servers to maximize the total profit. Ardagna et al. [10] extend their work to multi-tier profit optimization for continuous utility functions and proposed an iterative approach to reach the solution.

A mathematical formulation for the resource allocation problem in clusters is presented in [11]. The authors describe a method to find the best resource assignment in a cluster in the case that the application has certain resource requirements. Chandra et al. [12] introduce a dynamic resource allocation method in shared clusters to minimize the overall penalty resulting from not satisfying the SLA requirements in the response time. For this optimization, online measurements of the most important parameters in the system are used to predict the next system state and to allocate resources on that basis. An economic approach to manage shared resources and minimize the energy consumption in hosting centers is described in [13]. The authors present a solution that dynamically resize the active servers and respond to the thermal or power supply events by down grading the service based on the SLA.

The problem of multi class queues in clusters is addressed in [14] and the authors used the proposed model to solve the profit optimization problem in the clusters.

III. Problem Formulation and System Model

In this paper, a cloud computing facility (datacenter) which is composed of a number of *clusters* is considered. This cloud computing environment has a central manager that has some information about all clusters as well as the clients. Clusters are characterized by the number and type of *computing*, *data storage*, and *communication resources* that they control. All of these resources are assumed to be allocated within each server. Each cluster comprises of potentially heterogeneous servers chosen from a set of known *server classes*. Each server class is modeled by its processing capacity (C_j^p, normalized by a defined unit capacity), local data storage capacity (C_j^d) and communication capacity (C_j^c) as well as its operation cost which is related to the energy consumption for each server. The operation cost of a server is modeled as a constant operation cost ($cost_j^0$) plus a cost ($cost_j^p$) linearly related to the utilization of the server in the processing domain. Figure 1 shows the structure of the target cloud computing system with 5 clusters and a central management node.

Each client is identified by a unique id, represented by index i. The client class of the i[th] client is shown by $cc(i)$. Each server in the datacenter is similarly identified by a unique id, captured by index j. Servers are grouped into a set of clusters, indexed by k. Set of servers in each cluster is shown by $SS(k)$.

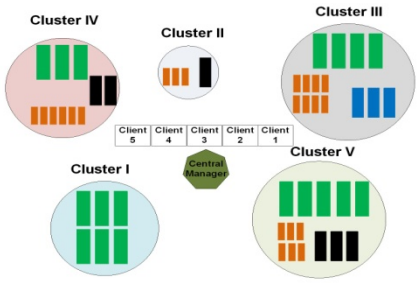

Figure 1. Structure of the cloud computing system in this problem.

In this system, all requests of each client are assigned to a single cluster. A pseudo-Boolean integer (y_{ik}) is used to determine if the i[th] client is assigned to the k[th] cluster (1) or not (0). The presented heuristic in this paper is applicable with a little modification to the cases that this assumption is not made but this restriction is helpful in doing dynamic resource allocation by cluster-level resource managers (in case of a large and sudden change in the service generation characteristics of a client and to satisfy the requested SLA condition at all times.) In each cluster, service requests of the assigned clients are dispersed to servers (requests generated by a single client can be assigned to more than one server). To capture this effect, parameter α_{ij} denotes the portion of the i[th] client's requests assigned to the j[th] server. Figure 2 shows the cluster dispatcher architecture.

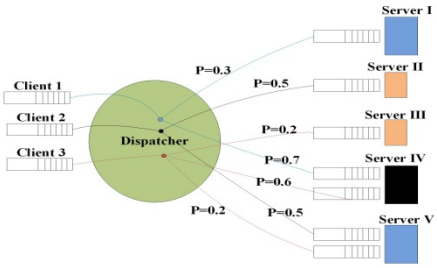

Figure 2. An example of the architecture of the cluster dispatcher.

To model the multi class queues in the system, Generalized Processor Sharing (GPS) is used [15]. It has been shown that the GPS can be implemented by weighted fair queuing if the service times for packets are not too large. To be able to find the analytical form of the response time, requests for each client are assumed to follow a Poisson distribution with mean of λ_i (predicted based on the behavior of the client.) It can be shown [18] from the properties of the Poisson distribution that if these requests are distributed by a system that has one input and some outputs and the input is connected to each output with probability of α_{ij}, the requests at each output have a Poisson distribution with mean of $\alpha_{ij}\lambda_i$.

Except the disk resource that is allocated based on the constant need of the clients ($Disk_i$), it is assumed that all other kinds of resources in the servers and clusters are allocated using the GPS. ϕ_{ij}^p, ϕ_{ij}^c and ϕ_{ij}^d denote Portion of the processing, communication and data storage resources of the j[th] server which is allocated to the i[th] client.

All multi class single server queues can be replaced by single class single server queues using GPS. To model the response time of the requests in the system, by using the known formula for the average service time of the requests in M/M/1 queues, the average response time of the clients' requests in each resource can be computed. To model the total response time in the system, we assume that the service times for different resources are independent and additive. Pipelining in the system shows that this assumption is reasonable because if the concatenated service time and one queue per client were used, the pipelining between

processing and communication was ignored. By this assumption about the queues, communication resource queue has the input from the processing resource queue. So, the overall average response time for a client can be presented as

$$R_i = \sum_{j\epsilon J} \alpha_{ij} \left(\frac{1}{\frac{C_j^p \phi_{ij}^p}{s_{ij}^p} - \lambda_i \alpha_{ij}} + \frac{1}{\frac{C_j^c \phi_{ij}^c}{s_{ij}^c} - \lambda_i \alpha_{ij}} \right) \equiv \sum_{j\epsilon J} \alpha_{ij} (R_{ij}^p + R_{ij}^c) \quad (1)$$

where s_{ij}^p and s_{ij}^c denote the Average processing and communication execution time of the i[th] client's requests on j[th] server with a single unit of resource.

The terms of the summation capture the average service time of the i[th] client in the j[th] server in processing and communication queues. To simplify the equations in rest of the paper, we use R_{ij}^p and R_{ik}^c to denote these terms.

Although the agreed request arrival rates are used to determine the profit, predicted average request arrival rates are used to allocate resources to clients. This can help us to use resources more efficiently in cases that we know that the actual request arrival rates are smaller than agreed request arrival rates.

The goal of the resource management problem is to maximize the total profit from serving clients. In this system, decision making interval (called decision epoch from here on) can be defined based on the behavior of the dynamic parameters in the system. For example, the frequency that request arrival rate changes for different clients affects the acceptable decision time. This is because the solution found by the presented algorithm is acceptable only as long as the parameters used to find the solution are approximately valid. Although some small changes in the parameters can be effectively tracked and responded to by proper reaction of *request dispatchers* in the clusters, large changes cannot be handled by the local managers. In the remainder of this paper, the resource allocation problem for each decision epoch is presented and a solution is presented but we do not discuss the estimation, prediction and dynamic changes in the system because these issues are outside the scope of the present paper.

IV. Optimization Problem

The total profit maximization problem is formulated below.

$$Max \sum_{i=1}^{n} \lambda_i^a U_{cc(i)} \left(\sum_{k=1}^{C} y_{ki} \left(\sum_{j\in SS(k)} \alpha_{ij} \left(R_{ij}^p + R_{ij}^c \right) \right) \right) - \sum_{k=1}^{C} \sum_{j\in SS(k)} \left(x_j cost_j^0 + cost_j^p \sum_{i=1}^{n} y_{ik} \phi_{ij}^p \right) \quad (2)$$

subject to:

$$x_j \geq \sum_{i=1}^{n} y_{ik} \alpha_{ij}, \quad \forall k, j \quad (3)$$

$$\sum_{i=1}^{n} y_{ik} \phi_{ij}^p \leq 1, \quad \sum_{i=1}^{n} y_{ik} \phi_{ij}^c \leq 1, \quad \forall k, j \quad (4)$$

$$\sum_{i=1}^{n} y_{ik} \phi_{ij}^d \leq 1, \quad \forall k \quad (5)$$

$$\sum_{k=1}^{C} \sum_{j\in SS(k)} y_{ik} \alpha_{ij} = 1, \quad \forall i \quad (6)$$

$$C_j^p \phi_{ij}^p / s_{ij}^p \geq y_{ik} \lambda_i \alpha_{ij}, \quad C_j^c \phi_{ij}^c / s_{ij}^c \geq y_{ik} \lambda_i \alpha_{ij}, \quad \forall i, j \quad (7)$$

$$\phi_{ij}^d = z_{ij} \, Disk_i / C_j^d, \quad \forall i, j \quad (8)$$

$$z_{ij} \in \{0,1\}, \; z_{ij} \geq \alpha_{ij}, \quad z_{i\bar{j}} \leq \sum_j \alpha_{ij} - \alpha_{i\bar{j}} - \epsilon \quad \forall i, j \quad (9)$$

$$y_{ik} \in \{0,1\}, \; \sum_{k=1}^{C} y_{ik} = 1, \quad \forall i \quad (10)$$

$$x_j \in \{0,1\}, \quad \forall j, k \quad (11)$$

$$\phi_{ij}^d \geq 0, \phi_{ij}^p \geq 0, \phi_{ik}^n \geq 0, 0 \leq \alpha_{ij} \leq 1, \quad \forall i, j \quad (12)$$

where x_j denotes a pseudo-Boolean integer to determine if the j[th] server is ON (1) or OFF (0) and ϵ is a small positive value. Moreover, $U_{cc(i)}(R_i)$ denotes the non-increasing utility function of the class client $cc(i)$ with response time equal to R_i and λ_i^a is the agreed arrival rate of the i[th] client in its contract with the cloud service provider.

In this problem, α_{ij} 's, ϕ_{ij}^* 's, x_j 's and y_{ki} 's are the variables whereas the other parameters are constant or functions of these variables. In the objective function, the first term is the summation of the service prices which are the function of the response time and the second term is the summation of the operation cost of the active servers in the clusters. In the constraints, (3) determines the active servers based on the allocated resources. Constraint (4) and (5) are used to limit the summation of the processing, communication and data storage resource utilities in each server in the clusters. Constraint (6) ensures that all requests generated by a client are served in one server cluster. Constraint (7) shows the lower limit on the resource shares in each server. Constraint (8) shows the disk requirement of a client assigned to a server. Assignment of a client to a server is determined with z_{ij} which is equal to zero if α_{ij} is zero and otherwise is equal to one based on the constraint (9). Constraints (10) to (12) specify the variable domains.

Problem (2) may be decomposed to the assignment of clients to clusters and allocation of resources to the assigned clients in each cluster. For the first problem, the initial client set (S) should be partitioned into smaller sets (S_i), where

$$S_i \cap S_j = \emptyset, \quad \forall i \neq j: \bigcup_{i=1}^{C} S_i = S \quad (13)$$

For the second problem, resources in the clusters should be allocated to the assigned set of clients to maximize the earned profit. This problem can be presented as follows.

$$Max \sum_{i\in S_k} \lambda_i^a U_{cc(i)} \left(\sum_{j\in SS(k)} \alpha_{ij} \left(R_{ij}^p + R_{ij}^c \right) \right) - \sum_{j\in SS(k)} \left(x_j cost_j^0 + cost_j^p \sum_{i=1}^{n} y_{ki} \phi_{ij}^p \right) \quad (14)$$

subject to the constraints of the problem presented in (2) for each cluster. This problem is a mixed integer non-linear program. As can be seen in the problem formulation, the number of constraints that should be considered to find the optimal solution is $O(|SS(k)| \times |S_k|)$, in which $|S_k|$ is the number of clients in the S_k and $|SS(k)|$ is the number of servers in the k[th] cluster.

It can be shown that even if the number and types of the active servers are determined in the problem and the discrete utility functions are estimated with continuous decreasing utility function based on response time, the objective function is neither convex nor concave (the Hessian matrix is not positive or negative definite) and so the problem cannot be solved with the convex optimization methods.

V. Optimization Method

The problem presented in the previous section is a hard problem at both cloud and cluster levels. The simple problem solvers cannot solve this problem except in the case of very small input size by running exhaustive search or by using stochastic optimization methods such as the Simulated Annealing or Genetic Search. In this section, a heuristic solution is presented for this problem.

The presented heuristic is focused on distributed decision making instead of centralized management. This is because complexity of the solution in case of centralized management is high whereas distributed decision making can handle the problem in parallel and reduce the time needed to reach a good solution by a big factor with limited amount of communication. Figure 3 shows the pseudo code of the overall heuristic.

A good initial solution is generated in the first step of this heuristic. To find an initial solution, clients are processed

sequentially and assigned to the best cluster on that time. This greedy algorithm is repeated for a number of times to generate different initial solutions and the best initial solution is selected.

To generate the final solution, a local search is used to improve the quality of the initial solution. It is important to note that this local search is not only used to change client assignment to decrease the resource saturation in some of clusters but also to combine the clients to decrease the number of active servers in the clusters. For example, if a server in a cluster serves a client and unassigned capacities in other servers is enough to serve that client with the same price, this local search will transfer the client to the other servers so as to decrease the cost of operation.

```
Algorithm Resource_Alloc ()
// Find an initial solution
For iter = 1 to num_init_solns {
    For k = 1 to num_clusters {
        curr_state_k = state of the cluster at end of prev. epoch;}
    Randomize client processing order;
    For i = 1 to num_clients {
        For k = 1 to num_clusters {
            (α_ij , P̃_ik ) = Assign_Distribute (i,k); }
        k_opt,i = argmax_k P̃_ik ;
        Assign client i to cluster k_opt,i based on its α_ij values;
        Update current state of cluster indexed by k_opt,i ; }
        Find total profit;}
Select the best initial solution;
// Improve the solution by optimizing α_ij, φ^p_ij, φ^c_ij, x_j values
While (Steady){
    For i = 1 to num_servers {
        (φ^p_ij , φ^c_ij) =Adjust_ResourceShares(α_ij);}
    For i = 1 to num_clients {
        α_ij =Adjust_DispersionRates (φ^p_ij , φ^c_ij);}
    For i = 1 to num_clusters {
        TurnON_servers(α_ij, φ^p_ij, φ^c_ij);
        TurnOFF_servers (α_ij, φ^p_ij, φ^c_ij) ;}
    Find total profit; }
```

Figure 3. Pseudo code for the overall resource allocation heuristic.

In this section, first a method to find a good starting point for the system is presented. Next, an iterative approach to improve the quality of solution with assigned clients is presented.

A. Finding an Initial solution

To start assigning clients to the clusters, each cluster is assumed to have an initial state. This initial state can be a result of the resources allocated to the previously assigned and running clients on the servers or other applications that are not related to the cloud computing system and are running on the cluster. This initial state can be specified in terms of the used (already allocated) capacity of the processing, data storage and communication resources in the clusters and different servers within the clusters.

To generate an initial solution, a greedy approach to sequentially select a cluster and add a client to the selected cluster is used. For each client, the best possible cluster to execute the application is found by finding the highest approximated profit for the client on each cluster with respect to the current state of the clusters. This process continues until all clients are assigned to the clusters. An approximated version of the profit is used to capture incompleteness of information in the system with respect to the assigned clients to the servers and clusters.

Assignment of each client to a server is defined by finding the amount of α_{ij}, ϕ_{ij}^p and ϕ_{ij}^c. To find these values, the utility function is used by a linear form of $(-a_i R_i + b_i)$.

In each step, a client (say, the i[th] client) is assigned to one of the clusters. To find the best possible profit for the client in each cluster (say the k[th] cluster), the following optimization problem is solved.

$$Max\tilde{P}_{ik} \triangleq Max\ \lambda_i^a \left[-a_i\left(\sum_{j\in SS(k)} \alpha_{ij}\left(R_{ij}^p + R_{ij}^c\right)\right) + b_i\right] - \sum_{j\in SS(k)} \left(\phi_{ij}^p cost_j^p + \phi_{ij}^c cost_j^0\right) \quad (15)$$

$$\phi_{ij}^p \le 1-\phi_j^p,\ \phi_{ij}^c \le 1-\phi_j^c,\ \phi_{ij}^d \le 1-\phi_j^d \quad \forall j$$

$$0 \le \alpha_{ij} \le 1,\ \ \sum_{j=1}^{n(k)} \alpha_{ij} = 1,$$

$$C_j^p \phi_{ij}^p / s_{ij}^p \ge \lambda_i \alpha_{ij},\ C_j^d \phi_{ij}^d / s_{ij}^d \ge \lambda_i \alpha_{ij},\ \phi_{ij}^d = z_{ij} C_j^d / Disk_i \quad \forall j$$

in addition to constraint (9). Notice that ϕ_j^p, ϕ_j^c and ϕ_j^d denote the previously allocated portion of the processing, communication and data storage resources in the j[th] server, respectively.

This objective function is neither a convex nor a concave function. To solve this problem, α_{ij} can be fixed in order to make the problem a concave optimization problem with respect to ϕ_{ij}^p and ϕ_{ij}^c (which can be solved optimally and in polynomial time by the Lagrangian method and Karush-Kuhn-Tucker (KKT) conditions [17]) as follows:

$$\phi_{ij}^p = \left(\frac{\lambda_i \alpha_{ij} s_{ij}^p}{C_j^p} + \sqrt{\frac{a_i \lambda_i^a \alpha_{ij} s_{ij}^p}{C_j^p (cost_j^p)}}\right)_{\lambda_i \alpha_{ij} s_{ij}^p / C_j^p}^{1-\phi_j^p} \quad (16)$$

Note that ϕ_{ij}^c is calculated using the same formula by changing the superscripts from p to c. The parentheses with two limits mean that the value in the parentheses is limited between these upper and lower bounds. To respect the disk constraint, only servers that have enough remaining disk capacity for the client is taken to the account to generate the initial solution.

To find the complete solution, α_{ij} is assumed to have discrete values and for each server the solution for complete range of α_{ij}'s is found with the closed form formula. For all inactive servers in each server class in the cluster ($\phi_j^p = 0$ and $\phi_j^c = 0$), this problem needs to be solved only once. Now, the final solution for (15) can be found by dynamic programming (DP) method to combine the solutions found for each server and generate the best assignment solution (α_{ij} for different servers to satisfy constraint of $\sum_{j=1}^{n(k)} \alpha_{ij} = 1$). For this DP, at most one of the all possible α_{ij} in each server is selected for the final solution. To manage the complexity and parallelize the solution, this DP computation can be done for each server class in the cluster and the results should be combined after finding all the data. This procedure is called *Assign_Distribute* (*i,k*) in the pseudo code.

B. Improving the solution by changing resource allocation

After assignment of clients to clusters and allocation of resources, the total utility may be increased using local search methods. Total profit maximization problem without changing the assigned set of clients to clusters is neither a convex nor a concave problem and we decompose it to the following parts.

1) Maximizing utilities with constant $\boldsymbol{\alpha_{ij}}$

This improvement can be done by optimizing resource allocation in each server once the set of its assigned clients and α_{ij} are fixed. This problem is formulated as follows.

$$Max \sum_{i\in S_{ij}} \alpha_{ij}\lambda_i\left(-a_i\left(R_{ij}^p + R_{ij}^c\right) + b_i\right) - cost_j^p \sum_{i\in S_{ij}} \phi_{ij}^p \quad (17)$$

subject to:

$$\phi_{ij}^p \ge \lambda_i s_{ij}^p \alpha_{ij} / C_j^p,\ \ \sum_{i\in S_{ij}} \phi_{ij}^p \le 1,$$

$$\phi_{ij}^c \ge \lambda_i s_{ij}^c \alpha_{ij} / C_j^c,\ \ \sum_{i\in S_{ij}} \phi_{ij}^c \le 1,$$

where S_{ij} denotes the set of assigned clients that has α_{ij} greater than zero. This optimization maximizes the total utility without changing the assignment parameters.

By changing the problem to minimization form, the hessian matrix of the objective function is positive definite, so the optimization problem is convex and the solution can be determined by the KKT conditions. Based on this method, if γ^p is a fixed parameter, the i[th] client has ϕ_{ij}^p in the following form:

$$\phi_{ij}^p = \left(\frac{\alpha_{ij}\lambda_i s_{ij}^p}{C_j^p} + \sqrt{\frac{a_i \alpha_{ij} \lambda_i s_{ij}^p}{C_j^p (\gamma^p - cost_j^p)}} \right)^1_{\lambda_i s_{ij}^p \alpha_{ij}/C_j^p} \quad (18)$$

Note that ϕ_{ij}^c calculation uses the same formula by changing the superscripts from p to c and removing the $cost_j^p$ term. By applying the constraints in problem (17), the amount of γ^p and γ^c can be determined by a numerical optimization method such as binary search. This procedure is denoted as *Adjust_ResourceShares*(α_{ij}) in the pseudo code.

2) Making servers active or inactive

Servers can be turned ON or OFF to increase the utility price or decrease the operation cost and improves the total profit in the system. If by making a server active, the total utility improvement is more than the operation cost of the server, this action will improve the total profit. For each client, increasing the utility is based on a change in its response time.

For selecting servers to activate, the best possible improvement after activation should be found and compared to the operational cost of the server. In each cluster, if a server class exists that has at least one inactive server, the problem to find the best set of clients to be assigned to the new server to maximize the total profit should be solved to decide whether activate a server or not. Finding the best improvement in the total profit by making a server active is a nonlinear mixed integer programming problem and we used decomposition method and dynamic programming to find a suboptimal solution with low complexity. Details of this problem are omitted for brevity.

This optimization should be executed for each server class in each cluster that has at least one inactive server in the final solution. This procedure is called *TurnON_servers*(α_{ij}, ϕ_{ij}^p, ϕ_{ij}^c) in the pseudo code.

When portion of the i[th] client's requests served by the new active server is zero, solution of this optimization problem can be used to adjust the assignment parameters to optimize the total profit while the allocated resources to different portion of the client's requests (ϕ_{ij}^*) are fixed. This is the dual problem of the problem presented in (17). This procedure is denoted as *Adjust_DispersionRates* (ϕ_{ij}^p , ϕ_{ij}^c) in the pseudo code.

To find a server to make inactive, the servers are ranked based on the total approximated utility function and server with lowest approximated utility is the first candidate for being inactive. To check that if making a server inactive can increase the total profit of the system, all clients that are assigned to the selected server are removed from the server and reassigned to the active servers. This may help in finding better assignment of clients to servers. If the total profit of the new system is more than the previous case, the selected server is turned OFF otherwise the selected server is removed from the candidate set of servers to turn OFF to process other servers and explore more cases in next iterations.

This procedure is called *TurnOFF_servers*(α_{ij}, ϕ_{ij}^p, ϕ_{ij}^c) in the pseudo code.

VI. EXPERIMENTAL RESULTS

To evaluate the proposed solution, we implemented the resource allocation method. There is no existing software framework for this, so we ended up implementing all components of the system, from clients to servers and clusters by ourselves. This implementation is done for resource allocation of the system that is modeled in section III.

The number of clusters in the cloud computing, the number of different server classes, and the number of different utility classes are set to 5, 10 and 5, respectively. We used normalized amounts for most of the components in the system instead of real values. For each utility class, s^p and s^c (mean of the execution time) for the clients are generated with (uniformly distributed) random variables between 0.4 and 1. The value of λ_i for each client is set with a random variable between 0.5 and 4.5. The utility class of each client is set with a random assignment form the existing utility classes. C^p, C^c and $cost^0$ for each server class are set with random variables between 2 and 6 and values of $cost^p$ is set with random variable between 1 and 3. The amount of C^d for each server class is set with another random variable between 2 and 6. The amount of $Disk_i$ for each client is set with a random variable between 0.2 and 2.

For the implemented heuristic, number of iterations for the initial solution was set to 3. The best initial solution from these three initial solutions was used to find the final solution after the local search. To compare the results of the proposed heuristic with a known resource allocation method, a modified version of the Proportional Share (PS) scheduling [8] was used. The original PS distributes the client's requests between all active servers; this strategy increases the response time of the clients. Also the class of clients is not considered in the original version of this scheduling scheme. We thus modified the original PS to decrease the number of the servers that each client is assigned to and consider the utility class of the clients. In this modified scheme, values of C^p for the active servers are added and the problem of finding the resource shares in the PS is solved by the assumption of having only one server with the processing capacity equal to the sum of all server processing capacities. Furthermore, the average of the processing service rate of the clients on the active servers is used as the processing service rate on that server. Also to consider the class of clients, this average service rate is multiplied with the slope of the utility function.

After this calculation, the amounts of the processing capacity for different clients are allocated based on a method inspired from the First Fit heuristic for the bin packing [16]. Again the heuristic is changed to fit our problem characteristics as follows. If the best server (with respect to the service rate) cannot meet the minimum required processing capacity for the client, the remaining capacity is allocated to the client and the next free server provides the rest of the required computational capacity. To process the clients, they are sorted based on their slope of utility with respect to the response time to serve the clients that are more sensitive to a change in the response time earlier in the heuristic. Also to find the best possible set of active servers, an iterative approach is used. A similar approach is used for the communication resource allocation. The quality of the solution generated from this modified algorithm is much better than the original PS.

To show the quality of the presented solution in client assignment and resource allocation in clusters, we generated different client assignments randomly and allocate the resources in the clusters based on the proposed solution. For each randomly generated solution, local search is used to increase the quality of the final solution. For this local search, after generating the final solution, the clients are picked one at a time and is removed from the assigned cluster and then the best cluster to serve the client is found based on the available condition of the clusters. This repeats until no further reassignment is possible.

Figure 4 shows the average results of the resource allocation in cluster for different number of the clients for (i) the proposed solution (i.e., the *Resource_Alloc* algorithm), (ii) the modified PS

Figure 4. Comparison between performances of proposed solution, modified PS, and "best solution found" with Monte Carlo simulation.

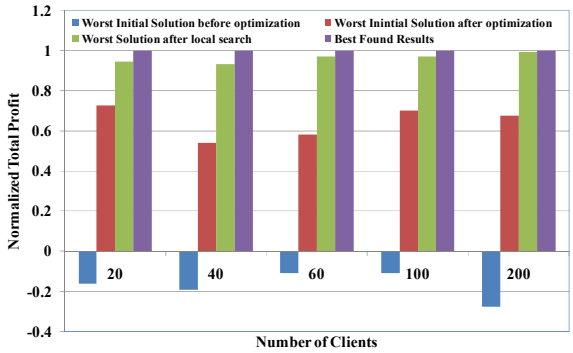

Figure 5. Comparison between random initial solution and final result.

plus finding the best set of active servers, and (iii) the best randomly generated and subsequently optimized solution (using the local neighborhood search). In this figure all the profit is normalized by the best found profit. For the latter "nearly optimal" solution, for given number of clients, at least 20 (5 for 200 clients) different scenarios are considered and for each one at least 10,000 random solutions are generated and optimized in order to find the best possible solution from this Monte Carlo like simulation.

As can be seen, the performance of the modified PS is not comparable to the proposed solution in part because all three dimensions of the resources are not considered. Also the best point for energy consumption is not calculated in this solution and this affects the quality of the solution. As can be seen from this figure, the differences between the average quality of the proposed solution and the optimal cases are not more than 9%. This shows that the proposed solution has acceptable solution quality in all attempted scenarios.

Figure 5 shows the normalized total profit of the worst random solution, profit of this solution after optimization by the local search, as well as the worst case profit of the proposed solution with respect to the found solution with Mont Carlo simulation in different scenarios for system parameters. As can be seen, quality of solution improves dramatically after the optimization in the initial solution.

The presented algorithm has different parts containing generating initial solution and improving the found initial solution. The computational complexity of the algorithm in case of finding the initial solution is $O(nC|j|g)$, in which g and |j| denote the granularity of α_{ij} which is used in the dynamic programming technique and the number of total servers in the system. By assumption of homogeneous number of servers in the clusters, distributed nature of the algorithm reduces this cost by a factor of $C^2$by adding the communication complexity. Also the computational complexity of the parts that improves the initial solution using the numerical optimization techniques and dynamic programming can be improved by a factor of C using the distributed computation.

VII. CONCLUSION

In this paper, the problem of multi dimensional SLA-based resource allocation in the cloud computing system is considered. SLA is defined based on a utility function which decreases when the average response time of the requests is increased. The response time based on the different allocation of resources for different servers and the clusters is modeled and used in the profit optimization problem. The problem is formulized based on Generalized Processor Sharing and an elaborate multi-stage heuristic algorithm is used to solve the problem. The system is modeled to demonstrate the quality of the final solution based on a modified version of the proportional share allocation and the optimal result based on the enumeration. In future works, the model will be expanded to deployment of complex multi-tier applications in a cloud computing infrastructure.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia. (2010, A view of cloud computing. Commun ACM 53(4), pp. 50-58.

[2] R. Buyya. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009.

[3] L. A. Barroso and U. Hölzle, The Case for Energy-Proportional Computing, *IEEE Computer*, vol. 40 (2007).

[4] K. Krauter, R. Buyya, and M. Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.* 32, 2 (Feb. 2002),

[5] R. Buyya and M. Murshed. (2002, 11). GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and Computation Practice & Experience 14(13-15), pp. 1175-220.

[6] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, Capacity Leasing in Cloud Systems using the OpenNebula Engine, *Workshop on Cloud Computing and its Applications*, 2008.

[7] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In Workshop on Power Aware Computing and Systems (HotPower '08). San Diego, USA, December 2008.

[8] Z. Liu, M. S. Squillante and J. L. Wolf. On maximizing service-level-agreement profits. Presented at 3rd ACM Conference on Electronic Commerce.

[9] L. Zhang and D. Ardagna. SLA based profit optimization in autonomic computing systems. Presented at ICSOC '04: Proceedings of the Second Int. Conf. on Service Oriented Computing, November 2004.

[10] D. Ardagna, M. Trubian and L. Zhang. (2007, SLA based resource allocation policies in autonomic environments. Journal of Parallel and Distributed Computing 67(3), pp. 259-270.

[11] C. Santos, X. Zhu, and H. Crowder. A mathematical optimization approach for resource allocation in large scale clusters. Technical Report HPL-2002-64, HP Labs, March 2002.

[12] A. Chandra, W. Gongt and P. Shenoy. Dynamic resource allocation for shared clusters using online measurements. International Conference on Measurement and Modeling of Computer Systems SIGMETRICS 2003.

[13] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat and R. P. Doyle. Managing energy and server resources in hosting centers. Presented at 18th ACM Symposium on Operating Systems Principles (SOSP'01), October 21, 2001.

[14] M. N. Bennani and D. A. Menasce. Resource allocation for autonomic clusters using analytic performance models. Presented at Second International Conference on Autonomic Computing.

[15] Z. Zhang, D. Towsley and J. Kurose. Statistical analysis of generalized processor sharing scheduling discipline. Presented at ACM SIGCOMM '94 Conf. on Communications Architectures, Protocols and Applications.

[16] S. Martello and P. Toth. Knapsack Problems: Algorithms and Computer Implementations. Wiley, 1990.

[17] J.Nocedal and S. J. Wright. Numerical Optimization. Springer-Verlag, 1999.

[18] A. Papoulis. Probability, Random Variables, and Stochastic Processes. McGraw-Hill, 3rd edition, 1991.