Adaptive Resource Allocation and Provisioning in Multi-Service Cloud Environments

Ayoub Alsarhan, Awni Itradat, Ahmed Y. Al-Dubai, *Senior Member, IEEE*
Albert Y. Zomaya, *IEEE, Fellow* and Geyong Min

Abstract—In the current cloud business environment, the cloud provider (CP) can provide a means for offering the required quality of service (QoS) for multiple classes of clients. We consider the cloud market where various resources such as CPUs, memory, and storage in the form of Virtual Machine (VM) instances can be provisioned and then leased to clients with QoS guarantees. Unlike existing works, we propose a novel Service Level Agreement (SLA) framework for cloud computing, in which a price control parameter is used to meet QoS demands for all classes in the market. The framework uses reinforcement learning (RL) to derive a VM hiring policy that can adapt to changes in the system to guarantee the QoS for all client classes. These changes include: service cost, system capacity, and the demand for service. In exhibiting solutions, when the CP leases more VMs to a class of clients, the QoS is degraded for other classes due to an inadequate number of VMs. However, our approach integrates computing resources adaptation with service admission control based on the RL model. To the best of our knowledge, this study is the first attempt that facilitates this integration to enhance the CP's profit and avoid SLA violation. Numerical analysis stresses the ability of our approach to avoid SLA violation while maximizing the CP's profit under varying cloud environment conditions.

Index Terms—Resource Management, Cloud Computing, Quality of Service, Cloud Service Trading, Economic Model.

1 INTRODUCTION

CLOUD computing has paved the way to enable users to access virtual computing resources on the Internet. This technology helps the cloud providers (CPs) to utilize resources efficiently and to generate extra income. However, the QoS for clients depends on the allocated resources. A CP may trade anything from infrastructure [1,2,3] such as processors, memory, and Internet access. Despite many studies found in the literature under the umbrella of cloud computing, resource management in multi-service environments is still in its infancy. In particular, key issues such as the integration of client satisfaction, QoS provisioning, and adaptive resource allocation policies have not yet been explored. Unlike existing contributions, this work places a great deal of emphasis on integrating the above issues with the aim of avoiding the Service Level Agreement (SLA) violation while maximizing CP profit under varying cloud environment conditions. Thus, in our work, a CP hires Virtual Machines (VMs) to execute clients' jobs and the cost of hiring VMs is amortized through client payments. The set of VMs in the cloud environment is managed by the CP. In particular, we propose an approach for resource management in multi-service environments based on a RL model. The model realizes continuous profit optimization for the CP. It integrates the adaptation of the offered number of VMs for each class of clients with the Request Admission Control policy (RAC). To satisfy QoS demands, the approach includes adaptations of the CP's resources to continuously meet request blocking probability constraints using the price parameter. The following are keys objectives for the proposed RL model:

- Client satisfaction by providing the committed QoS for users. This objective is achieved by offering an adequate number of VMs for serving users' jobs. For this purpose sufficient VMs must be available to serve all classes of users. Hence, the CP serves new requests on the basis of the RAC policy that ensures the request effectiveness and VMs availability.
- System Grade of Service (GoS). RAC policy blocks requests that give less gain provided that the blocking probability constraint is met for all classes of clients. To ensure good GoS to users, requests blocking probabilities must be constrained to acceptable values.
- CP gain which is basically defined as reward minus cost. This objective aims to optimize the CP's gain. In our work, users pay for individual requests. Hence, a CP's gain is computed using the amount of admitted requests.

The success of the proposed RL model framework de-

- *Ayoub Alsarhan is with the Dept. Computer Information System, Hashemite University, Jordan, E-mail: ayoubm@hu.edu.jo.*
- *Awni Itradatis with the Dept. Computer Information System, Hashemite University, Jordan, E-mail: itradat@hu.edu.jo.*
- *Ahmed Al-Dubai is with the School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, U.K. E-mail: a.al-dubai@napier.ac.uk.*
- *Albert.Y. Zomaya is with School of Information Technologies, University of Sydney, E-mail: albert.zomaya@sydney.edu.au*
- *Geyong Min is with Dept. Mathematics and Computer Science, University of Exeter, Exeter, EX4 4QF, U.K, Email: g.min@exeter.ac.uk*

pends on the optimization of both objectives: QoS for clients, and CP gain. However, these objectives conflict with each other. Businesses typically prioritize the gain. In the cloud environment, assurance of QoS standards is not a trivial task. This task requires solid definitions of QoS metrics tightly coupled with resource management policy capable of coping with changes that occur in the cloud environment. The interaction with the changes in the service cloud environment requires a dynamic framework capable of monitoring environmental conditions. Since typical system load fluctuates and changes over time, an optimal static solution is inadequate to solve this problem, especially in cases of system overload and inefficient cloud resources. The major contributions of this paper are as follows:

- Define and formulate the problem of resource adaptation in multi-service cloud environments where different requirements of different classes of clients are considered. In this formulation, gain maximization can be achieved by adaption of leased VMs that influences the admission rate of j^{th} class by increasing the service price for other classes. Gain maximization should be done in such a way that blocking probabilities for all classes do not exceed the constraints blocking probabilities. Service price is used in this problem to adapt cloud resources (VMs) with the gain maximization objective. We show how this concept can be generalized to state-dependent cloud market prices. This study demonstrates how the service price parameter provides a means for controlling almost independently and continuously QoS of different classes of clients. This feature is crucial in the control of a multi-service cloud market where request classes with different VM requirements can encounter very different QoS levels. The new advantages, achieved by the application of gain maximization and Markov decision theory to cloud market control, motivated us to extend and generalize these concepts to multi-service cloud markets. Increasing the price for the wealthiest classes during high demand reduces the hiring rate of VMs for these classes, which helps to meet the QoS constraint for other classes. Although the presentation focuses on multi-service cloud markets, most results and conclusions are of course applicable to the special case of one service market.

- Propose RL mechanism for CPs to model their long-term behaviors. RL is a promising approach to tackle this problem. It involves the synthesis of adaptive control algorithms for serving clients' requests, and it responds to measured cloud market conditions. Two new elements have been added to the RL algorithm for serving client requests for VMs in a cloud market: Markov decision theory and CP gain maximization. Markov decision theory is employed to compute a state dependent leasing policy by executing the RL algorithm. The model considers the economic factors for CPs that include the reward and the cost of hiring VMs. Furthermore, it is used to guarantee QoS for all classes of user. A major challenge for a CP is to charge the different classes appropriately. Service prices should generate maximum economic benefits to the CP. However, prices should also be reasonable with respect to the clients' budget. In this work, we use price as a parameter to enable CPs to generate gain optimally while providing the QoS to all client classes.

- Describe how a QoS-aware scheme is used to obtain a computationally feasible solution to the considered resource adaptation problem in multi-service cloud systems.
- Analyze the performance of the RL model under different cloud environment conditions.

Companies can use our RL framework to lease any virtualized computing resources that are delivered over the web. The rest of this paper is organized as follows. Section 2 describes the system model and assumptions. Works related to the problem are reviewed in Section 3. The RL formulation is presented in Section 4. Section 5 presents the performance evaluation results. Finally, the paper is concluded and future research directions are given.

2 SYSTEM MODEL AND RELATED WORK

This section presents our assumptions. Clients access a CP's VMs using networked client devices, such as desktop computers, laptops, and smartphones. Clients rely on CPs for a majority of their jobs. Requests are sent using a web browser to interact with the CP. Clients can send their requests anytime anywhere. Fig. 1 depicts the architecture of a cloud computing environment.

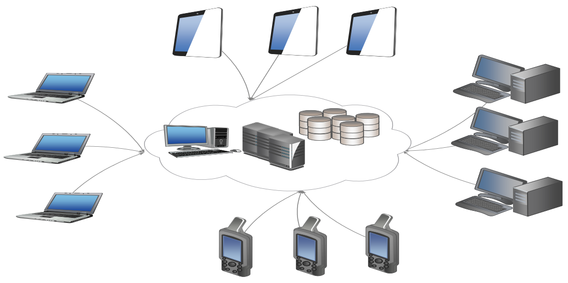

Fig. 1. Cloud computing architecture

The system consists of CP and X clients. The CP has N VMs that are offered to serve multiple classes of clients. The CP specifies the number of VMs N_j for the class j, QoS requirements for each class based on SLA (blocking probability), and the reward of deploying hired VMs for j^{th} class r_j. Each client of class j can request a bundle of VMs. The demand vector of the required VMs for a client of j^{th} class is d_j and it is represented as follows:

$$d_j = (v_1, v_2, v_3, \dots, v_n) \quad (1)$$

where v_i indicates whether a client requests i^{th} VM or not, v_i=1 means i^{th} VM is requested by a client, and v_i=0 means i^{th} VM is not requested. The reward r_j is computed as follows:

$$r_j = \sum_{i=1}^{n} p_j^i v_i \quad (2)$$

where p_j^i is the price for ith VM paid by jth class. We assume that these parameters are changed over time corresponding to the system conditions, such as workload, VM demand, and the service cost. Thus, a CP needs to change the price and the number of VMs offered in each class when needed. A client can use the VMs if he/she agrees to pay. We assume that a client's requests arrival follows Poisson distribution and each client class j has arrival rate λ_j. The service time μ_j for each request of jth class is assumed to be exponentially distributed. The CP faces the challenge of deciding the accepted requests. The CP should consider user demand for service and GoS. User demand is closely related to the reward received from clients.

Indeed, QoS guarantees are important for the cloud market, especially if the number of VMs is insufficient to serve clients' requests. Unfortunately, providing QoS guarantees is a challenging problem since the demand for VMs fluctuates over time. The work in [4] investigates various algorithms for resource provisioning in cloud computing systems. The main concern of the proposed algorithm is minimizing the penalty cost and improving customer satisfaction levels by minimizing QoS constraint violations. The proposed scheme considers customer profiles and providers' quality parameters to handle the dynamic nature of cloud environments. Researchers propose a new technique in [5] that jointly addresses the resource allocation and admission control optimization problems in cloud environments. The proposed technique takes into account the CP's revenues, the cost of servers, and clients' requirements. In [6], a new framework is proposed for dynamic resource provisioning in a virtualized computing environment. They consider switching costs and explicitly encode the notion of risk in the optimization problem.

An analytical model is presented in [7] to evaluate the performance of an IaaS in the cloud system. Several performance metrics are suggested to analyze the behavior of a cloud data center. These metrics include: availability, utilization, and responsiveness. An auction model is used in [8] for resource allocation in cloud environments. The key objective of the model is maximizing the CP's profit. The concept of virtual valuation is used in the proposed auction mechanism. In [9], researchers present a cloud resource procurement approach which helps clients to select an appropriate cloud vendor. The proposed approach implements dynamic pricing for profit maximization. Three mechanisms are suggested for a resource procurement scheme:

- Cloud-dominant strategy incentive compatible (C-DSIC).
- Cloud-Bayesian incentive compatible (C-BIC).
- Cloud optimal (C-OPT).

C-DSIC is a low-bid Vickrey auction. C-BIC is Bayesian incentive compatible and it achieves budget balance. C-BIC neglects the individual rationality of clients. Collaborative cloud computing (CCC) aims to use the resources that belong to different organizations or individuals in a cooperative manner. Authors in [10] propose a CCC platform, called Harmony. The proposed system integrates the resource management and reputation management in a harmonious manner. Three key innovations are incorporated: integrated multi-faceted resource/reputation management, multi-QoS-oriented resource selection, and price-assisted resource/reputation control. Video streaming services require huge storage capacity. Hence, more than one data center should be used to support this service, which is called multi-cloud. Data centers should be monitored and controlled to support QoS. In [11], a closed-loop approach is proposed for optimizing QoS and cost. Moreover, the authors suggest an algorithm to help CPs in managing data centers in a multi-cloud environment.

In [12], the authors formulate the optimal networked cloud mapping problem as a mixed integer programming (MIP) problem to provide a unified resource allocation framework for networked clouds. The proposed model aims to decrease the cost of the resource mapping procedure, while abiding by user requests for QoS-aware virtual resources. It presents a method for the efficient mapping of resource requests onto a shared substrate interconnecting various islands of computing resources, and adopts a heuristic methodology to address the problem.

In [13], different Internet services are hosted in a shared platform and offered to multiple classes of clients. The focus of the work has been to manage the capacity of the shared Internet data centers in such a way as to explore the available resources to the provider's best advantage so that a business goal is maximized. In [14], multiple customers are hosted on a collection of sequentially shared resources. The hosting environment is divided into secure domains. Each domain supports one customer. In this cloud environment, the resources are assigned dynamically to customers based on the work load. This dynamic resource allocation scheme enables flexible Service Level Agreements (SLAs).

An elastic web hosting provider is presented in [15]. A cloud hosting provider (HSP) makes use of the outsourcing technique. In order to take advantage of cloud computing infrastructures, a HSP provides scalable and highly available services to the web applications deployed on it. New middleware architecture is proposed in [16]. The architecture enables platforms to meet the QoS requirements of the applications they host.

The architecture incorporates a load balancer that distributes the computational load across the platform resources. Moreover, the QoS of clients is monitored in the proposed system. However, if the CP cannot support the required QoS, the platform is reconfigured dynamically in order to incorporate additional resources from the cloud. A new framework for resource management is proposed in [17]. The main goal of the proposed framework is to facilitate resource management by reducing the cost of serving users and to meet the QoS agreed with clients. The proposed scheme assigns resources to the clients based on the information provided by service providers. The resources are allocated to the clients according to the business goals and clients requirements. The end-to-end QoS for transaction-based services in multi-domain environments is modeled in [18]. The Mean Opinion Score (MOS) is used as a metric for QoS that is expressed by response time and download time.

The problem of an end-to-end QoS guarantee for Voice over IP (VoIP) services is addressed in [19]. In [20], the average response time of a service request is used as a QoS metric. However, measurement techniques are hard to use in computer service performance prediction especially for cloud environments. The authors in [27] propose a new scheme for resource reservation that maximally exploits discounted rates offered in the tariffs, while ensuring that sufficient resources are reserved in the cloud market. In order to make accurate resource allocation decisions, the scheme predicts the demand for streaming capacity. Authors in [28] propose a new scheme for QoS monitoring management for a system-of-systems (SoS) where any user from any location can share computing resources at any time. The proposed scheme enables QoS monitoring, management, and response for enterprise systems that deliver computing as a service through a cloud computing environment. In [29], authors model the workflow scheduling problem which optimizes both makespan and cost as a Multi-objective Optimization Problem (MOP) for cloud environments. Furthermore, they propose an evolutionary multi-objective optimization (EMO)-based algorithm to solve workflow scheduling problems on an infrastructure as a service (IaaS) platform.

In [30], a novel resource management framework (SPRNT) is proposed to ensure high-level QoS in cloud computing systems. It utilizes an aggressive resource provisioning strategy which encourages SPRNT to substantially increase the resource allocation in each adaptation cycle when the demand for service workload increases. The authors propose a new decentralized cloud firewall framework for individual cloud customers in [31]. They establish a novel queuing theory based model M/Geo/1 and M/Geo/m for quantitative system analysis, where the service times follow a geometric distribution. In [32], the authors propose PriDyn, a novel scheduling framework for monitoring QoS in the cloud market. PriDyn is designed to consider I/O performance metrics of applications such as acceptable latency and convert them to an appropriate priority value for disk access based on the current system state.

The framework provides differentiated I/O service to various applications and ensures predictable performance for critical applications in multi-tenant cloud environment. An adaptive framework is proposed in [33] for Service Maximization Optimization (SMO). The framework is designed to improve the QoS of the soft real-time multimedia applications in multimedia cloud computing.

In [34], the authors propose a new pervasive cloud controller for dynamic resource reallocation in cloud environments. The proposed system adapts to volatile time and location-dependent factors, while considering the QoS impact of too frequent migrations and the data quality limits of time series forecasting methods. Authors present new generic cloud performance models in [35] for evaluating Iaas, PaaS, SaaS, and mashup or hybrid clouds. Moreover, they test clouds with real-life benchmark programs and propose some new performance metrics.

The authors propose a new cloud model called SLAaaS – SLA aware Service in [36]. The model considers QoS levels and SLA for clients. Moreover, a novel domain-specific language that allows description of a QoS-oriented SLA associated with cloud services is proposed. However, none of the cloud approaches attempts to meet the agreed QoS of clients while maximizing the profit for the CP using price control parameters. Moreover, these approaches have implemented static provisioning of resources resulting in low resource utilization. Moreover, all of these works concentrated on a single class of clients. Finally, a dynamic behavioral adaptation to the cloud environment conditions was ignored in these strategies.

There are significant differences in our approach not only due to the differences in the system structure but also due to the dynamic nature of the cloud environment. The CP has to deal with the demand uncertainty problem and adapt its resources to meet the QoS for all clients' classes. In this work, we present the client satisfaction oriented resource allocation heuristic as a novel profit-driven trading algorithm. Our approach effectively meets QoS for different classes of clients. Specifically, client's classes, for which their QoS cannot be met, may be allocated more VMs by rejecting the requests for other classes whose QoS are met.

Our main concern in this work is modeling the long term average behavior of the CP as it evolves over time. We extract the optimal control policy to help CP for adapting resources (i.e. VMs) in each state of the system for meeting the QoS constraint for all classes of clients. Hence, the cloud market is an Ergodic dynamical system and this problem should be solved under the umbrella of the Markov Decision Processes (MDP).

Here, our main objective is to maximize profit for the CP by accommodating as many service requests as possible and maintaining a certain quality of service for all clients. This scenario might be best suited to small and mid-size CPs. Our main contribution is the integration of client satisfaction with our RL model. To the best of our knowledge, the work in this study is the first attempt that makes this integration to enhance the CP's profit and avoid SLA violation.

4 The proposed Resources adaptation

In the cloud environment, a resource adaptation control policy is required in conjunction with the RAC algorithm to meet a variety of objectives. When the cloud environment is in an under-loaded condition, RAC tries to accept every request and it allocates as many VMs as possible for all clients' classes. However, client demand for VMs may increase. In this case, some requests should be rejected by RAC to meet QoS for other clients.

We formulate the RAC problem as a Markov decision process (MDP) [21]. However, traditional solutions (value iteration, policy iteration, etc.) are infeasible within MDP due to very large state spaces that make traditional solutions suffer from the curse of the dimensionality problem. Moreover, from a modeling point of view, it is hard to estimate the transition probability in a real cloud environment due to varying environment conditions such as client demand, service cost, etc. Therefore, we choose RL to solve MDP using Q-learning [22]. The Q-learning method does not require the explicit expression of the state transition probabilities and it can handle MDP problems with large state spaces efficiently. Q-learning is one of the most popular RL algorithms [22]. The formulation of this method is presented in the following Section. The symbols used in this paper are listed in Table 1.

TABLE 1
SYMBOLS USED IN THE PAPER

Parameter	Description
N	Number of VMs in the market
X	Number of clients
r_j	Reward for leasing VMs for j*th* class
d_j	The demand vector of the required VMs for a client of j^{th} class
p_j^i	The price for i*th* VM paid by j*th* class
λ_j	Arrival rate for j^{th} class
μ_j	The service time for j^{th} class
S	State space of the cloud environment
s_j	The number of VMs required for j^{th} class
J	The set of client classes
Ω	All possible events in the system
e_0^j	Request arrival event for class-j
e_1^j	Request departure event for class-j
A	Action space
$R(S(t), e_i^j(t), a)$	The intermediate reward for action a at time t
π	A policy that maps the current event and state of cloud environment to an action
π^*	Optimal policy
$v(\pi^*)$	Average reward of policy π^*
D	The time horizon
h^*	Vector of differential reward functions
$\bar{\tau}$	Average transition time corresponding to state-action pair
$E_{e_i^j}$	The expected reward over the probability events
$S(t+1)$	Next state
θ	A vector for the tunable parameters for a policy π
Y	State action tuple
$\nabla v(\pi(\theta))$	Gradient of the reward of policy π with respect to θ
$Q^\theta(Y, a)$	Value function for an action of starting state in state-event action tuple (Y, a) for the policy $\pi(\theta)$
$\nabla v(\pi(\theta^*))$	Gradient of the average reward for $\nabla v(\pi(\theta))$
$\varphi^\theta(Y, a)$	The gradient ∇ with respect to θ at time t for an action of starting state in state-event action tuple (Y, a)
Δ_t	The transition time between state Yand $\acute{\mathrm{Y}}$
$q_{\mathrm{Y}\acute{\mathrm{Y}}}(\tau, a)$	Transition probability between state Yand $\acute{\mathrm{Y}}$
$R_{AD(j)}$	Reward for admitting a new request of class j
u	Unity vector
$\partial_j^\theta(\mathrm{Y}, \mathrm{a})$	Feature vector of state action pair
$R_{DE(j)}$	Reward of departure clients of class j
$h^*(\mathrm{Y})$	Optimal reward vector
π^*	Optimal policy for VM leasing
h_q	Vector parameterized by vector q
$\acute{h}_q$	New vector parameterized by vector q at time t+1
D	Time horizon
q	A parametric vector
δ_t	Discount factor
G	Net gain of accepting client j request
C_j	The cost of serving a request from client j
G^*	Optimal net gain
B_j	Blocking probability for class j
B_c^j	Blocking probability constraint for class j
$\bar{\lambda}_j$	The acceptance rate of clients' j requests
$\acute{p}_j^i$	New price
α	Maximum number of clients arriving to the system
ω	Rate of decrease of the arrival rate as reward r_j increases
ϑ_j	A threshold for the maximum number of the class j requests

A. *Modeling VMs Allocation ON Cloud Market*

Due to the cloud environment, changing service demand can be regarded as quasi-stationary snapshots of the system that will be visible (available) in a periodical order. The set of VMs allocated for class *j* varies over time as the demand changes to meet blocking probabilities for all classes. Note also that in the proposed scheme the final decision as to whether to accept a request or not is made by the CP based on some constraints and objective function.

In our work, the cloud environment is modeled as a discrete-time event system. The events can be represented as stochastic variables with appropriate probability distribution. In order to utilize the RL algorithm, we need to identify the system states, actions, and objective function. Given a service demand, the required number of VMs for each class, the state space *S* of the cloud environment is given by:

$$S = \left\{\sum_{\forall\, j\in J} s_j \le N\right\} \tag{3}$$

where s_j is the number of VMs required for j^{th} class, and J is the set of clients' classes. Let Ω denote the finite set of all possible events in the system where:

$$\Omega = \left\{e_0^1, e_1^1, e_0^2, e_1^2, e_0^3, e_1^3, \dots\dots, e_0^J, e_1^J\right\} \tag{4}$$

where e_0^j denotes the request arrival event for class-*j*, and e_1^j indicates the departure of class*j* request (e.g. e_1^j=1 means a request for class *j* departs the system while e_1^j=0 means no request for class j departs the system). We note that in this cloud environment the status is changed at epochs of new request arrival and request departure which are associated with event e_i^j. However, when a request is served and departs the system, its VMs are released and a decision must be made whether to admit a new request or not. Let A denote the set of allowed actions when the current state of cloud environment and the current event are given. This set A can be defined as:

$$A = \{a: a \in \{0,1\}\} \quad (5)$$

where $a = 0$ denotes that a request is rejected, and $a=1$ indicates that the CP accepts the request. Let π be a stationary policy that maps the current event and state of cloud environment to an action. The mapping process using policy π is an embedded finite state Markov chain evolving in continuous time. Despite the chain evolving in continuous time, we need only to consider the system transition at epochs where the events and decision take place. Let $e_i^J(t)$ be the event that occurs at time t and assume $S(t)$ is the state of the cloud environment at time t during interval $[t-1, t]$. Then, the intermediate reward for action a at time t is computed as follows:

$$R(S(t), e_i^j(t), a) = \begin{Bmatrix} r_j, & e_i^J(t) = e_0^j = 1 \; and \; a = 1 \\ 0, & e_i^j(t) = e_0^j = 1 \; and \; a = 0 \end{Bmatrix} \quad (6)$$

Our objective is to find the optimal policy π^* that maximizes the average reward $v(\pi^*)$ such that $v(\pi^*) \geq v(\pi)$ and this can be expressed as follows:

$$v(\pi^*) = \lim_{D\to\infty} \frac{\sum_{t=0}^{D} R(S(t), e_i^j(t), a)}{D} \quad (7)$$

where D is the time horizon. For policy π, any state can be reached by any other state and the limit in (7) exists and is independent of the initial state. The optimal policy π^* can be generated by solving Bellman's equation for average reward [24]:

$$h^*(S(t)) + v(\pi^*)\overline{\tau}(S(t), a) = \\ \arg\max_{a\in A}\left\{E_{e_i^j}\left[R\left(S(t), e_i^j(t), a\right) + h^*(S(t+1))\right]\right\} \quad (8)$$

where $v(\pi^*)$ is the optimal reward, $h^*$is the vector of differential reward functions, $\bar{\tau}S(t)$ is the average transition time corresponding to state-action pair, $E_{e_i^j}$ is the expected reward over the probability events, and $S(t+1)$ is the next state which is a function of $S(t)$, $e_i^j(t)$, and a.

B. Optimizing VM Allocation in Cloud Environments

In this section, we propose a new algorithm for extracting an optimal policy for VM trading in cloud environments using RL. The extracted policy is performed at the CP and it helps the CP to map each state of the cloud environment with action. The embedded Markov chains $\{S(t), e_i^j(t), a\}$ in the cloud environment evolve within state space $(S \; x \; \Omega)$ x A. Assume θ be a vector for the tunable parameters for a policy π. Our objective is to find a policy $\pi(\theta^*)$ which will translate into actions for given states and events such that $v(\pi(\theta^*)) > v(\pi(\theta))$. Assume $Y = \left(S(t), e_i^j(t)\right) \in S \; x \; \Omega$. We can find the optimal policy $v(\pi(\theta^*))$ by improving the gradient of the average reward for $\nabla v(\pi(\theta))$ which is given by:

$$\nabla v(\pi(\theta)) = \frac{\sum_{\forall Y\in Sx\;\Omega}\sum_{\forall a\in A(Y)} p_\theta(Y)\pi(\theta)(a|Y)\varphi^\theta(Y,a)Q^\theta(Y,a)}{\sum_{\forall Y\in Sx\;\Omega} p_\theta(Y)\sum_{\forall a\in A(Y)}\pi(\theta)(a|Y)\bar{\tau}(Y,a)} \quad (9)$$

The gradient ∇ with respect to θ can be improved as follows:

$$\varphi^\theta(Y,a) = [\varphi_0^\theta(Y,a), \ldots, \varphi_{W-1}^\theta(Y,a)]^t \quad (10)$$

$$\varphi_z^\theta(Y,a) = \frac{\frac{\partial}{\partial\theta(z)}\pi(\theta)(a|Y)}{\pi(\theta)(a|Y)}, \quad z = 0, \ldots., Z-1 \quad (11)$$

The value function for an action of starting state in state-event action tuple (Y, a) for the policy $\pi(\theta)$ is computed as follows:

$$Q^\theta(Y,a) = \\ R(Y,a) - v(\theta)\bar{\tau}(Y,a) + \textstyle\sum_{\forall\acute{Y}} p_{Y\acute{Y}}(a)h^\theta(\acute{Y}) \quad (12)$$

$$\bar{\tau}(Y,a) = \textstyle\sum_{\forall\acute{Y}}\int_0^\infty \tau q_{Y\acute{Y}}(d\tau, a) \quad (13)$$

The function $q_{Y\acute{Y}}(\tau, a)$ is computed as follows:

$$q_{Y\acute{Y}}(\tau,a) = \Pr[\tau(t+1) \leq \tau(t) | Y(t+1) = \acute{Y}, Y(t) = \\ , and \; a(t) = a] \quad (14)$$

where $\Delta_t = t - t - 1$ is the transition time between state $\acute{Y}$ and $\acute{Y}$. Since the exact model for the cloud environment is unknown, we estimate $Q^\theta(Y,a)$ as follows:

$$\grave{Q}_u^\theta(Y,a) = \textstyle\sum_{j=1}^{J} u(j)\partial_j^\theta(Y,a) \quad (15)$$

where the unity vector u is parametric and $\partial_j^\theta(Y,a)$ is the feature vector of state action pair. The following lemma is needed to guarantee extraction of the optimal policy.

Theorem 1: *For all* $S(t) \in S, t \geq 0, J \geq j \geq 1, v(\pi^*(S(t+1))) \geq v(\pi^*(S(t)))$

Proof: Obviously, $v(\pi^*(S(t) + e_i^J)) \geq v(\pi^*(S(t)))$. Since e_i^J either takes the value 0 (request depart) or 1 for request arrival. If a request of class j arrives (e_1^J) then CP gets a reward r_jN_j based on (4) otherwise it gets 0. Let $R_{AD(j)}$ represent the reward for admitting a new request of class j. Then:

$$R_{AD(j)}v(\pi^*(S(t))) = \\ max(r_jN_j + v(\pi^*(S(t) + e_i^J)), v(\pi^*(S(t))) \quad (16)$$

Assume $R_{DE(j)}$ models the departure of a class j request, which is defined as:

$$R_{DE(j)}v(\pi^*(S(t))) = v(\pi^*(S(t) - e_0^j)) \quad (17)$$

Thus (7) could be re-written as:

$$v(\pi^*(S(t))) = \sum_{j=1}^{J} \lambda_j \, R_{AD(j)} v(\pi^*(S(t-1))) + \sum_{j=1}^{J} \mu_j \, R_{DE(j)} v(\pi^*(S(t-1))) \quad (18)$$

For $t = 0$, $v(\pi^*(S(0)))=0$, $\forall S(t) \in S$. It is clearly $v(\pi^*(S(0) + e_i^J)) \geq v(\pi^*(S(0)))$. Generally, we need to prove the pervious inequality for all values of t taking into account the arrival ($R_{AD(j)}$) and departure ($R_{DE(j)}$) of clients requests. Since $R_{DE(j)}$ and $R_{DE(j)}$ have positive values and under linear combination, the lemma can be proved using induction on t. Suppose that $v(\pi^*(S(t-1) + e_i^J)) \geq v(\pi^*(S(t-1)))$. It is clear from (16) that $max(r_j N_j + v(\pi^*(S(t) + e_i^J + \acute{e}_i^J)), v(\pi^*(S(t))) \geq max(r_j N_j + v(\pi^*(S(t) + e_i^J)), v(\pi^*(S(t)))$. Hence, the arrival event also satisfies the inequality. For the departure event, it is easy to prove that the departure satisfies the inequality by using (17). Therefore, we showed that the optimal policy should maximize the average reward for all events in the system which mean $v\left(\pi^*(S(t))\right)$ is a non-decreasing function. ▯

The main idea of using RL is to extract the optimal policy by finding a scalar $\acute{v}$ and the optimal reward vector $h^*(Y)$ that is obtained using the simulation. Optimal policy can be obtained using greedy policy:

$$\pi^* = \underset{a \in A}{\arg\max} \, R(S(t), e_i^j(t), a) + \acute{h}_q(S(t+1)) \quad (19)$$

where $\acute{h}_q$ is a vector parameterized by vector q and it is computed as follows:

$$\acute{h}_q(S(t)) = \sum_{t=0}^{D} q(d) \delta_t(S(t)) \quad (20)$$

where $q = [q(0), \ldots, q(D)]$ is a parametric vector and it is the feature vector for the cloud environment state. In our cloud computing market, if a CP admits a new client request of class j, it gets $r_j N_j$ using (6). The net gain of accepting the client j request is computed as follows:

$$G = r_j N_j + \acute{h}_q(S(t+1)) - h_q(S(t)) \quad (21)$$

Net gain can be derived using the cost of serving a client request as follows:

$$r_j N_j + \acute{h}_q(S(t+1)) - \acute{h}_q(S(t)) = r_j N_j - C_j \quad (22)$$

where C_j is the cost of serving a request of client j. The RL policy in (19) can be expressed in terms of net gain where the CP attempts to maximize the net gain as follows:

$$G^* = arg \max_{j \ni J} r_j N_j - C_j \quad (23)$$

CP uses RL policy to make decisions that give high net gain. RL policy must explore all actions and favor the one that generates the highest gain. Therefore, the learning process gradually favors actions that appear more worthy than others by trying out a variety of actions over time. In our model, RL policy chooses the best action that gives the maximum gain with high probability and other actions are selected with less probability. However, the CP should consider the blocking probability constraint where the blocking probability for class j requests should not exceed B_c^j. For class j, blocking probability is computed as follows [23, 24, 25]:

$$B_j = \frac{(\lambda_j - \overline{\lambda}_j)}{\lambda_j} = 1 - (\frac{\overline{\lambda}_j}{\lambda_j}) \quad (24)$$

where $\overline{\lambda}_j$ is the acceptance rate of clients j requests. CP uses a reward parameter adaptation to achieve this objective, since in general an increase of r_j causes a decrease of arrival rate λ_j that causes decreasing of B_j, and vice versa. The arrival rate depends on the reward. The new arrival rate of users is calculated as follows [26]:

$$\lambda_j = \alpha e^{-\omega \acute{p}_j^i} \quad (25)$$

where α is the maximum number of clients arriving to the system, ω represents the rate of decrease of the arrival rate as price p_j^i increases, and $\acute{p}_j^i$ is the new price. Here we assume ω is given a priori. There is an inverse relationship between the reward and the demand for the VMs. Since a change of r_j influences the optimal solution for the VM trading problem, the reward parameter adaptation should be integrated with the VM adaptation.

Theorem 2: *To maximize the net gain of the CP in the cloud environment and to maintain QoS for all client classes, a request of class j can be accepted if and only if* $s_j < \vartheta_j$, *where* ϑ_j *is a threshold for the maximum number of class j requests.*

Proof: Assume $r_j > r_i$, $\forall i \neq j$. Let $\lambda_j = \lambda_i, \forall i \neq j$. It is clear that $v\left(\pi^*(S(t) + e_j^J)\right) - v\left(\pi^*(S(t))\right) \leq v\left(\pi^*(S(t) + e_j^J + e_j^J)\right) - v\left(\pi^*(S(t) + e_j^J)\right)$ because admitting more requests of clients j involves more gain. We know that if $v\left(\pi^*(S(t) + e_j^J)\right)$ is greater than $v\left(\pi^*(S(t))\right)$, $v\left(\pi^*(S(t) + e_j^J + e_j^J)\right)$ and is also greater than $v\left(\pi^*(S(t) + e_j^J)\right)$. $v\left(\pi^*(S(t) + e_j^J)\right) > v\left(\pi^*(S(t))\right)$, means that accepting class j requests will give more gain than rejecting them after $t+\acute{t}$ stages from the initial state $S(t)$ while $v\left(\pi^*(S(t) + e_j^J + e_j^J)\right) > v\left(\pi^*(S(t) + e_j^J)\right)$ means that accepting a class j request generates more gain after t+$\acute{t}$ stages from the initial state $S(t) + e_j^J$ than accepting other classes requests. However, if the CP continues accepting class j requests then the blocking probability of class j becomes zero ($\overline{\lambda}_j$=λ_j). Despite the CP maximizing its gain by accepting class j requests, the QoS for other classes is degraded significantly and their blocking probabilities becomes high($\bar{\lambda}_j \cong 0$). Thus, we can find a threshold ϑ_j at the system state $S(t)$, such that the class j request can be accepted, if the number of class j requests in the system is smaller than the threshold. Otherwise, the gain for accepting an arrival class j request will degrade QoS for other users. Hence, some requests should be rejected. The following algorithm is used to find the optimal policy π^*:

Algorithm. Finding the optimal policy

```
1  Arbitrarily initialize $S(0) \in S$, and $\bar{v}_0$;
2  for t = 1 to D do
3    At state $S(t)$ generate an event $e_i^j(t)$
4      $\tau(t) = t - t - 1$
5      Perform updates:
6          $l_t = R(S(t), e_i^j(t), a) - \bar{v}_{t-1}\tau(t) + \tilde{h}_{u_{t-1}}(S(t)) - \tilde{h}_{u_{t-1}}(S(t-1))$
7          $u_t = u_{t-1} + \varrho_t l_t \nabla_u \tilde{h}_{u_{t-1}}(S(t-1))$
8          $\bar{v}_t = \bar{v}_{t-1} + \Upsilon_t(R(S(t-1), e_i^j(t-1), a)) - \bar{v}_{t-1}\,\tau(t)$
9          $a = \arg\max_{a\in A}\left\{R\left(S(t), ue_i^j(t), a\right) + \tilde{h}_{u_t}(S(t+1))\right\}$
10         Compute $R\left(S(t), e_i^j(t), a\right)$
11 End for
```

where ϱ_t, Υ_t are small step size parameters to decrease the convergence speed of the resulting training scheme and $\nabla_u \tilde{h}_{u_t}(S)$ is computed as follows:

$$\nabla_u \tilde{h}_{u_t}(S) = \varphi(S) \tag{26}$$

5 PERFORMANCE EVALUATION

In this section, the performance of the proposed resource adaption scheme is evaluated using discrete event based simulations. Requests are served based on the gained reward and the QoS constraints. The uniform distribution is used to generate the number of the requested VMs for each request. The parameters used in the simulation experiments are given in Table 2. Note that some of these parameters are varied according to the considered scenarios in our experiments. The key performance measures of interest in the simulations are:

(1) CP's net gain which is computed using equation (23).

(2) QoS for users, which is represented using a blocking probability that is computed using equation (24).

Each simulation run consists of 100000 requests. The results are averaged over enough independent runs so that the confidence level is 95% and the relative errors do not exceed 5%. We examine the performance under different parameter settings.

TABLE 1
SIMULATION PARAMETERS

Parameter		Value
Number of VMs		100
Number of clients		150
Number of requests per client		Random
λ_1 (arrival rate of class 1)		1
λ_2 (arrival rate of class 2)		1
λ_3 (arrival rate of class 3)		1
λ_4 (arrival rate of class 4)		1
Blocking probability constraint for class 1		0.25
Blocking probability constraint for class 2		0.2
Blocking probability constraint for class 3		0.15
Blocking probability constraint for class 4		0.1
Service price of class 1		5
Service price of class 2		10
Service price of class 3		15
Service price of class 4		20
Number of served requests		100000
α		150
ω		0.5
δ_t		0.9
ϱ_t		300 ms
Simulation Devices	Intel i5 Core	2.50GHz
	Process cores	2 x 2.50GHz
	RAM	6 GB
	OS	Windows 7 64 bit

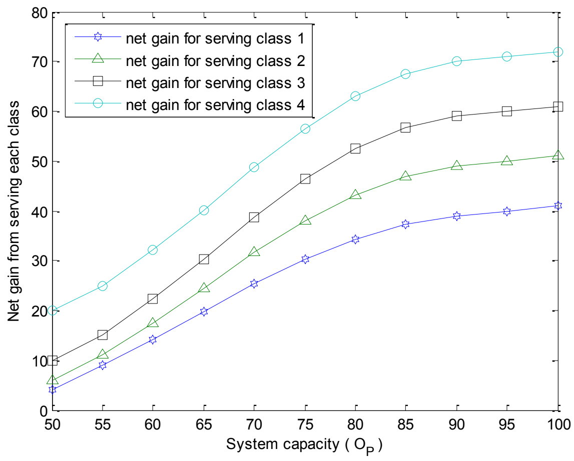

Fig 2. CP's net gain under different system capcities

A. *Net gain as a function of system capacity and service demand*:

System capacity refers to the numbers of VMs that a CP can offer to clients. Fig. 2 shows the net gain for the CP with different offered VMs. Now, the question is whether the interplay between the increased gain and the increased system capacity can generate higher gain. Utilizing more VMs means serving more clients and, hence, serving more requests. This interplay has two mutually opposite effects on the gain.

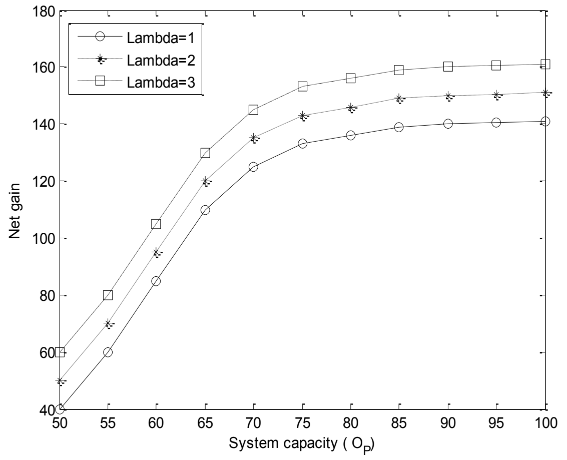

Fig 3. CP's net gain under different service demand

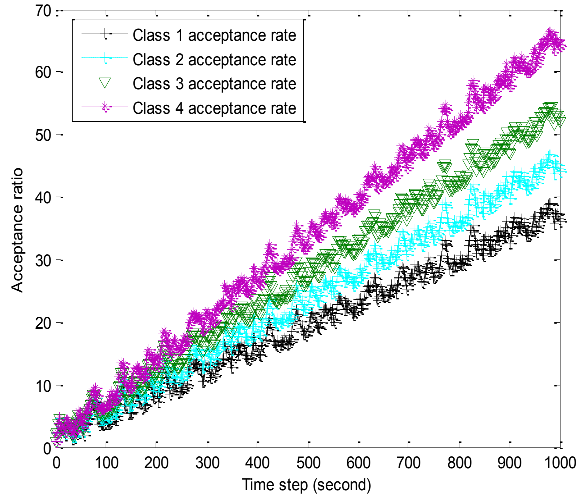

Fig 4. Percentage of accepted requests over time

Clearly, increasing the number of admitted requests has a positive effect on the gain. On the other hand, serving more requests pushes down the demand for service, and thus, individual payments decrease. If the net effect is positive, the gain will increase. The results in Fig. 2 show the ability of the RL scheme to maximize CP gains as the system capacity increases. The RL scheme prioritizes class 4 since the clients of this class pay more than others. The net gain of class 4 is higher than others because the CP serves more requests for this class. It is clear that the net gain increases for all classes as the number of offered VMs increases but after a certain number the gain becomes stable and the increment in the number of VMs does not affect the gain due to the limited number of clients. Hence, CP reaches the maximum gain for the given arrival rate.

Scalability is the capability of a scheme to increase the output under an increased load. We conduct experiments to analyze the ability of our scheme to work on larger systems.

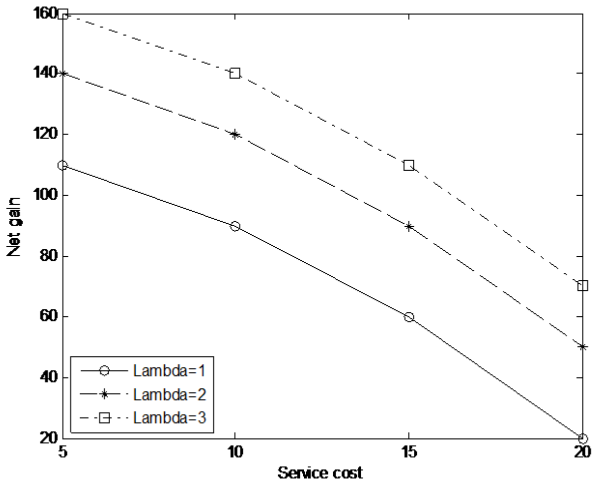

Fig 5. CP's net gain under different service cost

Fig. 3 displays the reported net gain for different service demands under different system capacities. It is clear that the gain is increased for a higher workload (i.e. arrival rate) and for the system with higher capacity (i.e number of VMs). A CP can serve more clients when it has more VMs. From Fig. 3, we can find out that our scheme can maximize a CP's gain under different service demands and within different system capacities.

B. *RL for maximizing CP's net gain:*

The RL scheme serves the requests that generate higher gain over time. Fig. 4 shows the percentage of accepted requests for each class of clients over time. The RL policy focuses not only on a specific class of clients but it considers the QoS of the whole clients. Thus, it can guarantee the blocking probability for each class to be less than certain thresholds. Clearly, the CP prefers to serve class 4 requests since they pay more. However, the CP should meet the blocking probability constraint for other classes. However, when the workload is high and the cloud environment is becoming saturated, the CP selects more requests from class 4 provided that QoS constraints for other classes are met. Many factors prevent the CP from reaching the maximum gain. These factors include the cost of renting a processor (c_j), the service paid by a client, and client behavior. Fig. 5 shows reported total net gain from serving all classes for different values of c_j. It can be observed that in any period when the cost is high, the reported gains are always less than the periods with lower cost. Clearly, the net gain is decreased as the service cost is increased. Hence, a CP should expect less gain in a period with high service cost.

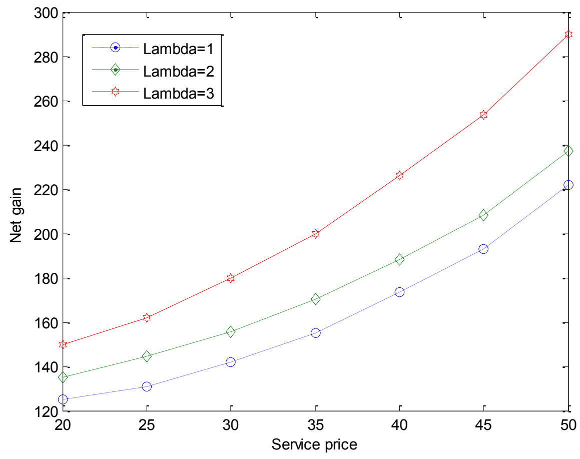

Fig 6. CP's net gain for different values of service price

Fig. 6 shows the net gain under different values of service price paid by clients of class 4. We observe that the net gain increased by increasing the service price but in this experiment the sensitivity of the arrival rate to the increment reward is 0.1 and the reported net gain is taken for different values of arrival rate. Although this experiment focuses on a single class of client, the results and conclusions are of course applicable to the multiple classes of clients.

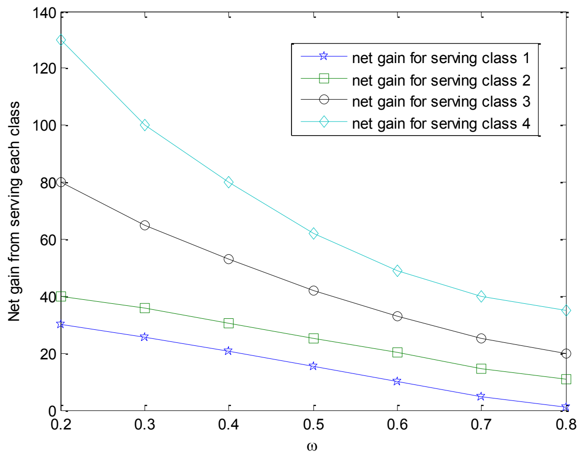

Fig 7. CP's net gain for different values of omega

To study the relationship between the net gain and client's behavior we plot the net gain against different values of ω in Fig. 7. It is clear as the value of ω increases the net gain decreases significantly. Large values of ω decrease the demand for service (arrival rate). Large values of ω model the clients who care more about the cost of service instead of the satisfaction level they may get from the product (VMs).

Apparently, there is an inverse relationship between prices charged and service demand. Once a CP increases prices, clients may start looking for another provider. Therefore, a CP should think about the tradeoff between the benefits a client receives from a service and the price they are willing to pay.

C. *Tradeoff between service level and net gain:*

In this experiment, we investigate how the QoS levels for clients affect the CP's gain. From Fig. 8, we notice that when the CP provides high level of QoS in terms of blocking probability its gain is degraded significantly. As B_c^j decreases, the QoS requirement for clients becomes stricter in such a way that more requests should be protected from rejection. For this, user requests must be admitted more often to meet blocking probability constraints and the CP cannot select worthy requests. By contrast, as B_c^j increases, the QoS requirements become less strict so that more requests can be rejected upon their arrival and the CP selects the worthy requests in terms of reward. A RL-based scheme should keep the blocking probabilities for all classes below the target value regardless of the offered load.

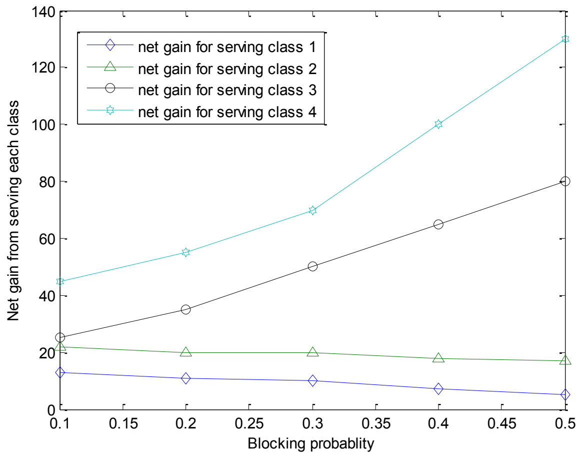

Fig 8. Net gain from serving each class of clients under different blocking probabilities

D. *Maintaining QoS for clients*

A CP with well dimensioned system capacity and correctly chosen service price provides the desired QoS and maintains blocking probabilities in acceptable range for all classes of clients. While our adaptation scheme tries to maximize the CP's gain by increasing the number of offered VMs for wealthiest clients during periods of high demand, it maintains QoS by bringing blocking probabilities back to its constrained range by increasing the service price. In any market, the price of the good or service is determined to meet the objectives of sellers. In this experiment, we consider the disequilibrium scenario in the market case where the service price for a certain class does not conform to the QoS constraints. Such a case may arise, for example, due to

the limitation in the amount of VMs to be allocated to this class. In our system, class 1 pay less than other classes. Hence, the demand for VMs of class 1 increases. Consequently, more requests of this class are rejected and the blocking probability of this class is getting higher. Since the excess VMs demand for class 1, VMs for other classes should be relocated to this class (i.e., clients of class 1 that are not satisfied with the allocated VMs can deviate to buy VMs of other classes without changing the price). To achieve this, CP has to increase prices for other classes. Clients are often less likely to rent VMs at higher prices.
Fig. 9 shows the price adaptation for all classes when the blocking probability for class 1 is growing. The CP increases the price of service for classes of clients where QoS constraints are met to decrease the demand for these classes. The CP increases the prices for class 2, class 3, and class 4. This action decreases the demand for VMs and this enables the CP to meet the QoS for class 1 by allocating more VMs for this class. The results show our scheme's ability to bring blocking probabilities back to their constrained range by adapting service price.

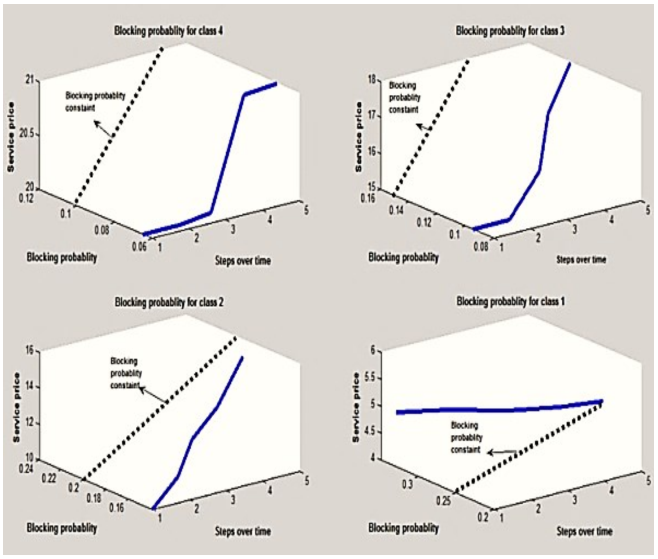

Fig 9. Adapting service price to meet QoS constraint for class 1

6 Conclusion

In this paper, we have formulated the QoS provisioning problem for the adaptive resource allocation in cloud environments as a constrained MDP to find the optimal policy that can maximize the gain for the CP and guarantee QoS constraints. The emphasis has been placed on employing RL approaches to learn a nearly optimal decision policy that helps a CP to adapt its resources to meet the system objectives and to solve the QoS provisioning problem. The optimal scheme is extracted under conditions where the service demand made by clients is "uncertain". For the CP, our solution is proved optimal and supports QoS for clients. It is interesting that when the QoS requirements become stricter, the CP prefers less profit. On the other hand, when clients become less strict for QoS, a CP can generate more gain. However, gains made by more competing CPs are less than that made by one CP. We are in the process of carrying out similar analysis taking into account the competition among CPs. We plan to derive the optimal solutions for CPs in an uncertain market. In this market, clients will be compensated for any degradation in QoS and service price will be derived based on the QoS constraints. Beside adopting fair pricing policy, we will consider a cloud market in which there are multiple CPs and potential clients. Furthermore, we intend to carry similar analysis on real systems.

References

[1] S. Chee, and C. Jr, "Cloud Computing: Technologies and Strategies of the Ubiquitous Data Center," CRC Press,Boca Raton, U.S.A, 2009.

[2] B. Sosinsky, "Cloud Computing Bible," John Wiley & Sons, San Francisco, U.S.A, 2011.

[3] A. Alsarhan and A. Al-Khasawneh, "Resource trading in cloud environments for utility maximisation using game theoretic modelling approach," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 31, no. 4, pp.319-333, 2016.

[4] L. Wu, SK.Garg, S. Versteeg and R. Buyya, "SLA-based resource provisioning for hosted software as a service applications in cloud computing environments,"*IEEE Transactions on services computing*, vol. 99, no.1, pp. 465-485, 2013.

[5] J. Almeida, V. Almeida, D. Ardagna, I. Cunha, C. Francalanci, and M. Trubian, "Joint admission control and resource allocation in virtualized servers, "*Journal of Parallel and Distributed Computing*, vol. 70, no. 4, pp. 344-362, 2010.

[6] D. Kusic, JO. Kephart, JE. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control,"*Cluster Computing*, vol.12, no 1, pp.1–15, 2009.

[7] B. Dario, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems,"*IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp.560-569, 2014.

[8] A. Alsarhan, K. Al-Sarayreh, A. Al-Ghuwairi, and Y. Kilani, "Resource trading in cloud environments for profit maximisation using an auction model,"*International Journal of Advanced Intelligence Paradigms* , vol.,6, no. 3, pp. 176-190, 2014.

[9] A. S. Prasad and S. Rao, "A Mechanism Design Approach to Resource Procurement in Cloud Computing', *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 17-30., 2014.

[10] H. Shen and G. Liu, "An Efficient and Trustworthy Resource Sharing Platform for Collaborative Cloud Computing,"*IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 862-875, 2014.

[11] W. Chen, J. Cao and Y. Wan, "QoS-aware virtual machine scheduling for video streaming services in multi-cloud', *Tsinghua Science and Technology*, " vol. 18, no. 3, pp. 308-317, 2013..

[12] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervelló-Pastor and A. Monje, "On the optimal allocation of virtual resources in cloud computing networks," IEEE Transactions on Computers, vol. 62, no. 6, pp. 1060-1071, 2013.

[13] B. Abrahao, V. Almeida, J. Almeida, A. Zhang, D. Beyer and F. Safai, "Self-adaptive SLA-driven capacity management for Internet services,",*Proc.NOMS*, pp. 557–568, 2006.

[14] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, S. Krishnakumar, D. Pazel, J. Pershing and B. Rochwerger," Oceano—SLA-based management of a computing utility,"*Proc.IEEE/IFIP*, pp. 855–868, 2001.

[15] J. O. Fitó, I. Goiri and J. Guitart, "SLA-driven elastic cloud hosting provider," *Proc.PDP'10*, pp. 111–118, 2010.

[16] G. Lodi, F. Panzieri D. ,Rossi and E. Turrini, "SLA-driven clustering of QoS-awareapplication servers,"*IEEE Transactions on Software Engineering*, vol. 33, no. 3, pp.186–197, 2007.

[17] J. Ejarque, M. de Palol, I. Goiri, F. Juli`a, J. Guitart, R. Badia and J. Torres, "SLA-Driven semantically-enhanced dynamic raesource allocator for virtualized service providers,"*Proc.eScience*, pp. 8–15, 2008.

[18] D. Mei, B. Meeuwissen and F. Phillipson, "User perceived Quality-of-Service for voice-over-IP in a heterogeneou multi-domain network environment," *Proc ICWS*, pp. 1–13, 2006.

[19] D. Mei and H. B. Meeuwissen, "Modelling end-to-endQuality-of-Service for transaction-based services in multidomain environement," *Proc ITC19*, pp. 1109-1121, 2005.

[20] J. Martin and A. Nilsson, "On service level agreements for IP networks," *Proc INFOCOM*, pp. 1-6, 2002.

[21] L. Put, "Morkov Decision Processes: discrete stochastic dynamic progrmming," New York Wiley, 1994.

[22] H. Watkins, and P. Dayan, "Technical Note: Q-leaming",*Machine Lemming*, vol. 8, no. 3-4,pp.279-292, 1992.

[23] S. Ferretti, V. Ghini, F. Panzieri, P. Michele, and E. Turrini, "QoS - Aware Clouds," *Proc IEEE CLOUD*, pp. 321-328, 2010.

[24] S. Sutton and G. Barto, "Reinforcement Learning: An Introduction," The MIT Press, Cambridge, 1998.

[25] P. Beckmann, "Elementary Queuing Theory and Telephone Traffic" A volume in a series on telephone traffic published by Lee's ABC of the Telephone, Geneva, IL, 1977.

[26] G. Gallego and G. Ryzin, "Optimal dynamic pricing of inventories with stochastic demand over finite horizons,"*Management Science*, Vol. 40,No. 8 ,pp. 999-1020, 1994.

[27] A. Alasaad, K. Shafiee, H. Behairy, H. M., and V. C. M Leung, "Innovative Schemes for Resource Allocation in the Cloud for Media Streaming Applications",*IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1021-1033, 2015.

[28] P. C. Hershey, S. Rao, C. B. Silio and A. Narayan, "System of systems for Quality-of-Service Observation and response in cloud computing environments," *IEEE Systems Journal*, vol. 9, no. 1, pp. 212-222, 2015.

[29] Z. Zhu, G. Zhang, M. Li and X. Liu,"Evolutionary multi-objective workflow scheduling in cloud,"*IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344-1357, 2016.

[30] J. Liu,J. Y. Zhang, Y Zhou, D. Zhang and H. Liu, "Aggressive resource provisioning for ensuring QoS in virtualized environments, "*IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 119-131, 2015.

[31] M. Liu, W. Dou, S.Yu, and Z. Zhang,"A decentralized cloud firewall framework with resources provisioning cost optimization,"*IEEE Transactions on Parallel and Distributed Systems*," vol. 26, no. 3, pp. 621-631, 2015.

[32] N. Jain, and J. Lakshmi, "PriDyn: enabling differentiated I/O services in cloud using dynamic priorities',*IEEE Transactions on Services Computing*," vol. 8, no. 2, pp. 212-224, 2015.

[33] G. Jia, G. Han, D. Zhang, L. Liu., and L. Shu, "An adaptive framework for improving quality of service in industrial systems',*IEEE Access*,"vol. 3, no. , pp. 2129-2139, 2015.

[34] D. Lučanin, and I. Brandic,"Pervasive cloud controller for geotemporal inputs, "*IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 180-195, 2016.

[35] K. Hwang, X. Bai, Y. Shi, Li,M., G. Chen, and Y. Wu, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 130-143, 2016.

[36] Damián Serrano, Sara Bouchenak, Yousri Kouki, Frederico Alvares de Oliveira Jr., Thomas Ledoux, Jonathan Lejeune, Julien Sopena, Luciana Arantes, Pierre Sens, "SLA guarantees for cloud services," *Future Generation Computer System*, vol. 54, no. C, pp. Issue C, pp. 233-246, 2016.