Journal of Universal Computer Science, vol. 24, no. 4 (2018), 372-391
submitted: 14/10/17, accepted: 15/4/18, appeared: 28/4/18 © J.UCS

Cloud Biometric Authentication: An Integrated Reliability and Security Method Using the Reinforcement Learning Algorithm and Queue Theory

A. M. N. Balla Husamelddin
(Northeast Forestry University, Harbin, China
Husamelddin123@yahoo.com)

Guang Sheng Chen
(Northeast Forestry University, Harbin, China
nefujwp@163.com)

Weipeng Jing
(Northeast Forestry University, Harbin, China
weipeng.jing@outlook.com)

Abstract: While cloud systems deliver a larger amount of computing power, they do not guarantee full security and reliability. Focusing on improving successful job execution under resource constraints and security problems, this work proposes an enhanced, effective, integrated and novel approach to security and reliability. To apply a high level of security in the system, our novel approach uses cloud biometric authentication by splitting the biometric data into small chunks and spreading it over the cloud's resources. Reliability is enhanced through successful job execution by employing an adaptive reinforcement learning (RL) algorithm combined with a queuing theory. Our approach supports task schedulers to effectively adapt to dynamic changes in cloud environments. Based on the idea of reliability, we developed an adaptive action-selection, which controls the action selection dynamically by considering queue buffer size and the uncertainty value function. We evaluated the performance of our approach by several experiments conducted in terms of successful task execution and utilization rate and then compared our approach with other job scheduling policies. The experimental results demonstrated the efficiency of our method and achieved the objectives of the proposed system.

Keywords: Biometric Authentication, Security, Reinforcement learning, Q-learning, Reliability, Queuing theory
Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

1 Introduction

The complexity of cloud architecture and cloud systems adds security risks to the adoption of those cloud systems. The characteristic of cloud models in the cloud environment (e.g. resources sharing and multi-tenancy) can observes unsecure objects or behaviour by permitting the allocation of the same resources to deferent users. The risk of unsecure elements can come from exposing the system to cloud environment for computation and storage aims. The organizations and individuals data resident in the same resources due to computing or storing may compromise the hosting machine through channel attacks [Ainapure, Shah and Rao 2018].

Several developments have been made in the distributed systems technology during the recent software applications era involving data sharing, resources sharing and migration from local to web applications which support multiple users system. To this end, a secure and reliable environment is essential to provide an enhaced quality of service. In this paper, we emphasises on the development of service-oriented architectures (SOA). We discuss in details the integration between reliability and security in the cloud services. The reliability and security have been enhanced in our system by distributing the biometric data and optimized task scheduling using a machine learning method proposed as reinforcement learning algorithm merged with queue theory. Distributed services can be designed and developed by SOA where the different domain owners can control it. These architectures are basically a group of services or activities that perform a single task and interconnect to each other over simplified data passing. A cloud service provider in the cloud environment acts as an endpoint of communication for an end user by supporting tasks and requests alienated from service implementation.

The security and reliability of the web applications are essential because web applications contain the data of both enterprises and private customers. Thus, protecting and securing optimum task scheduling of these possessions is an essential part of any cloud model development. Authorization and authentication, which include activity logging and auditing, are usually important steps for protecting assets. There are several traditional protection methods that have been developed for the purpose of protecting, including encryption, password management, vulnerability analysis and intrusion prevention.

The implementation of cloud computing [Srinivasan, Sarukesi, Rodrigues, Manoj and Revathy 2012] has many advantages, such as flexibility and scalability , however, adopting cloud computing also brings new challenges in terms of providing secure and reliable environment.

The cloud resource sharing and the complexity in cloud architecture are a main reason for the security risks of cloud because of the resources allocation ability for different [Chen, Paxson and Katz 2010]. Compared to local or traditional architectures, cloud environments, as distributed architectures, imply an enlarged use of data communication flows and wide networks. Therefore, exposure to risks is possible through the malicious activities like spoofing, side channel attacks and others.

In addition, the adopted cloud models has a related element of risk. In fact, in some cloud models, there are requirements for users to transfer part of their personal data to the cloud service provider. The problem is not only the allocation of personal data to the service provider, but also the user’s inability to apply a particular protection mechanism, t, but also that the user cannot apply a specific protection mechanism, such as access control or encryption, as the cloud resources are entirely controlled by the service provider. To avoid these risks, some key actions must be considered for managing the cloud infrastructure, such as employing security systems tools and system administrators. All types of activities could be performed by these actors (cloud service providers) because they usually have the power to do so and this theoretically breaks safety requirements that corporate policies enforce. However, the sophistication of these fraudulent activities makes them more unreachable. Moreover, much work is needed for the security evaluation in the cloud environment where international certification agencies are not available.

The reliability in cloud services due to bunches of resources that exist in the data centres raises the risk of failure [Microsoft, white paper 2013]. Thus, enhancing reliability of cloud services based on VMs is an essential issue [Bauer and Adams 2012]. Many approaches concerning service reliability issues have been proposed and deployed. For example, fault removal, tolerance, forecasting, and prevention are techniques for achieving reliability enhancement [Kochovski and Stankovski 2018]. The predictability of virtual machine VM failures means that adopting some fault tolerance techniques is important to recover the service affected by a failure event. There are some techniques uses Reinforcement Learning RL for learning automatically that proposed in ([Ranjbari and Torkestani 2018], [Peng, Zhang, Peng and Man 2017]). RL is a goal oriented learning algorithm that interacting with the environment to achieve that goal by learns a policy to optimize a reward [D. Liu, Wei, Wang, Yang and Li 2017].

In our proposed system, verification of each participant in the electronic communication implemented using authentication protocols. The access keys to a number of subsystems can be managed and preserved by an authentication server (AS). The authorized person must connect with the AS in order to be allowed to access the private data. To allow the accessibility of private data and services by the users, authorized persons must first declare their identities through a connection with the AS. Then and by using RL, a comparison and matching between the user identity and the existent data can be done after data retrieving from the cloud.

In this paper we take both security and reliability in consideration as an integrated approach to enhance the performance of the proposed cloud biometric authentication system. This paper discusses concerning security as one of the major steps in protecting IT infrastructures and data, which is remote user authentication. The other section of this paper discusses how reliability of cloud systems can be enhanced by using an RL algorithm merged with queue theory for scheduling the user requests. The rest of this paper is structured as follows: section the related work and state-of-the art have been discussed in section. In section 3, we introduce the related preliminaries concerning the data residing, RL and queue theory. In section 4, we describe in detailes the proposed system followed by the implementation of the system in section 5. The experiments setup along with the results discussed in section 6. We conclude our work and future works in section 7 and 8 respectively.

2 Related work

Resource allocation involves tackling issues related to efficiency by encouraging strategies associated with effective scheduling and matching, schedulers' communication between each other, resources usage status and the execution time for the task. Recently, researchers have been trying to deal with various application services by developing self-adaptive methods for scheduling policy, and this has become a key research field, especially in utility optimization of distributed systems. In [Bu, Rao and Xu 2009], the authors developed an approach for define an exploration and exploitation method shows the capabilities of agents. The agents collaborate with the environment to increase the learning exploration, and they estimate the state using the applications' information. In the meantime, the utility or Q table (which is updated by the agent) is ordinal and iteratively shared in the state to view and evaluate the efficiency of resources. However, the authors did not cover decisions regarding the dynamic environment of distributed resources. In our proposed system, we adopted RL, which deploys a highly dynamic environment in a distributed structure. Our proposed system can adapt with the changes in the dynamic environment and take the decision regarding these changes.

To measure the indicators of performance, such as completion time and waiting time, the [X. Liu, Tong, Zhi, ZhiRen and WenZhao 2014] deployed a Markov request queue model, which was achieved by considering the shared resources among VMs and several failure types. However, there was no buffer size consideration for the subtask, which is not logical in an environment like cloud computing. Using RL as a framework, [Wu, Xu, Zhang and Liu 2011] proposed an algorithm for distributed learning named as Ordinal Sharing Learning (OSL). In their algorithm, a utility table is attached to each scheduler, and this table must be updated in two steps. First, their algorithm uses local rewards to update the table. Second, their approach to update the table, the scheduler uses the utility table of its adjacent. It's the table is driven by its scheduler to the neighbor scheduler after updating the utility table in cooperation with other schedulers, and the neighbouring agent completes the update and delivery of the utility table. Again, there was no buffer size consideration for the subtask.

In [Hussin, Hamid and Kasmiran 2015], the reporters addressed a reliability enhancement in their proposed model. They employed an adaptive RL algorithm to improve the tasks execution with little computational overhead. The authors proposed an approach which is a combination of RL with a neural network to support a scheduler in adapting and observing the dynamic changes in environments. However, the approach deals with a homogeneous environment and does not support a heterogeneous system. [Tesauro, Jong, Das and Bennani 2006] proposed a hybrid scheduling approach that merges both RL and queuing models in (closed and open) loop traffics. In their approach, the system is controlled by a queuing model policy, while RL uses the collected data to perform an offline training. Instead of lookup tables, the authors employ RL to train a nonlinear function for the approximation and enable the scaling to be in larger state spaces. [Vengerov 2007] proposed a general framework and employed RL to dynamically manage the cloud resource allocation between multiple entities. This method implements RL in combination with the fuzzy rule bases which makes it flexible to employ in an environment without the need of existing resource allocation policies. Compared with [Vengerov 2007], we employed more techniques and strategies in our work such as queue theory which is important to be consider in the cloud environment and focused more on RL and queue theory strategies for resource allocation.

3 Preliminaries

3.1 Data residing security

The distribution and fragmentation of data is a potential solution for the security insurance of data residing (stored) on cloud storage, as distribution and fragmentation allow data to be dispraised in all available machines on the cloud storage after the data is split into fragments. However, this method makes it difficult to recover complex data for unauthorized users. Distributing data on different provider platforms and using fragmentation methods can solve the security problems that are triggered by the distrust in the cloud service provider. However, to improve security and to enable proper distribution and fragmentation of the data, it is fundamental to develop tools that can guarantee prompt integrity and data availability without complexing the system. In fact, to execute this approach, a performance reduction related to information retrieval procedures or excessive consumption of resources should be done. Herein, we propose a solution for security issues in the cloud environment which involves high availability of chunked data based on distributed cloud storage. The basic idea of the approach is to fragment the data (biometric data) into small chunks then spread it across the different distributed resources on the cloud. More details are discussed in section 4.

3.2 Reinforcement learning

Reinforcement learning (RL) is one of the machine learning models built on the psychological conception of reinforcement. The working model of RL increases the likelihood of a specific behaviour by offering rewards after the behaviour occurs [Fig. 1]. The RL problem can be stated as a Markov Decision Process (MDP) so that the repeated steps can enhance the processing of jobs.

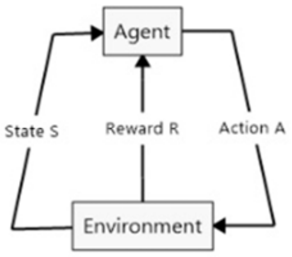


Figure 1: Basic reinforcement learning structure

Without any a priori knowledge, RL captures the performance and policy of a target application. As shown in [Fig. 1], the basic role of RL is to learn from the interaction between the agent and the environment. Here, the agent is the decision-maker, and the agent learns from experience to execute the best action for each state of the environment by maximizing the returned reward.

The fundamentals of RL algorithm are as follows:

1. At each step “s”, an action “a” is chosen which maximizes the value of the function Q(s, a), where Q represents the estimated utility function.
2. Based on the result of this function, the quality of an action in a given state can be determined by an immediate reward. The immediate reward in RL can be defined as to (immediately) determine whether the taken action is correct.

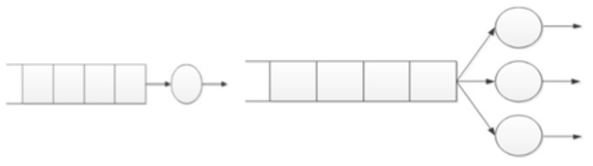
3. The Q(s, a) function represents the immediate reward for making a decision (an action) to achieve the best utility (Q) for the resulting state and it can be state as following:

$$Q(s,a) = r(s,a) + \gamma \max_{a'}(Q(s',a')), \qquad (1)$$

Where $r(s,a)$ is the immediate reward, γ is the relative value of the delayed reward against immediate rewards (0 to 1), and *s'* is the new state after action *a'*. The terms *a* and *a'* are actions in states *s* and *s'*, respectively.

3.3 Queuing theory

This section discusses the main queuing model terms and their ability to be employed by scalable scenarios. Queuing theory has been employed in the most recent studies and defined as the study of waiting lines in a mathematical way [Robertazzi 2012]. Queuing theory also used to evaluate performance metrics (e.g. the tasks average waiting time or the queue length for tasks) as well as modeling internet applications and traditional servers. The mean arrival rate for client requests that arrive at the system represented by λ and those requests are queued until they are processed. In the model, one or more servers (VM) may be available for the requests at a mean service rate of μ [Fig. 2]. We can describe a queue briefly as A/B/C/K/N/D or A/B/C as a shorter notation. [Fig. 2] shows the basic structure of the queuing models.



(a) *(b)*

Figure 2: Simple queuing models; (a) one server; (b) multiple servers

Elements K, N and D are non-compulsory, and the assumption was that K = 1, N = 1 and D =FIFO (First Come First Served / First In First Out) in the absence of those elements. The description of each element's notation is as follows:

- A: The distribution of inter-arrival time
- B: The distribution of service time
- C: Servers total number
- K: Queue length or system capacity. This refers to the maximum allowable number of requests in the system. Extra arrivals must be prevented if the number of requests exceeds this maximum. If K is absent, the capacity is assumed to be unlimited.

- N: Calling population. This represents the population size from which the requests originate. The population is supposed to be unlimited if this number is absent. An open queuing model is used when the requests originate from an infinite population, whereas a closed queuing model is used when the requests are based on a finite customer's population.
- D: Service discipline or priority order. This refers to the methods of task serving in the queue. The common method is FCFS / FIFO, in which the requests are served according to its arrival order. Other methods include LIFO/LCFS (Last In First Out / Last Come First Out).

M, D and G are the most representative values for inter-arrival A and service time B. Markovian (represented by the M value) denotes a Poisson process, which is illustrated by parameter λ and specifies the arrival requests total number per time unit. Moreover, an exponential distribution is employed by the service time or inter arrival. In the case of the D value, the inter-arrival or service times are always constant or deterministic. G is another regularly used, which is related to a distribution with a particular mean and variance. In our model, several queues are integrated into the same queuing network. The real cloud system can be defined as multi-tier applications within a queuing network. For instance, every tier is formulated as a queue with single or multiple servers.

Queuing theory is categorized by frequent arrival and service rates anticipated for systems with a stationary nature. The queuing model parameters must be periodically recalculated in where models conditions are unstable. There are some useful performance metrics that can be estimated in a system according to a stationary queuing model. The performance metrics include; the arrival rate λ, inter-arrival time, the average waiting time in the queue, the average number of requests in the queue and service time. Two main approaches can be used for obtaining those metrics which include: analytic methods and simulation methods. Analytic methods are commonly valid for simple models, while simulation methods are suitable for more complex scenarios. Little's Law [Little and Graves 2008] can be used as a typical formula for simple queuing models that have proper service process and request arrival distributions (e.g. standard statistical distributions, such as Normal or Poisson). This formula can be represented as the requests' average number is equal to each requests' average duration multiplied by the average request arrival rate of λ. A related definition of Little's Law states that the average length of queue can be considered as the mean of arrival rate multiplied by the waiting time average in the queue. Some queuing models can be resolved using analytic approaches, such as the M/M/1 and G/G/1 processes. In Poisson-based models, service times and inter-arrival times display an exponential distribution. For example, in the case of the M/M/1 model, the mean response time R can be computed as follows:

$$R = \frac{1}{(\mu - \lambda)} \tag{2}$$

Where μ is a service rate and λ is the arrival rate.

In a model-free system, the system learns value functions in a direct way instead of constructing an environmental model. Estimating value functions correctly is the key to model-free systems, and it plays an important role in our study. Before we discuss our model in detail, we examine the epsilon-Greedy method, which is a method that is typically employed to balance exploitation and exploration. In this method, actions are chosen by an agent based on the greedy policy. However, the policy changes at times, and actors take exploratory actions. The exploratory actions ratio is denoted by the parameter ꜫ.

4 The proposed model

4.1 The integration of cloud computing with biometric recognition

In order to allow access of biometric authentication to a cloud platform, implementation of an application on the client desktop must be done on the user side, and an AS (Authentication Server) must be connected to the Keystone module. Keystone is a service that provides accounting and authentication for the whole platform and can be installed on a virtual machine. There are many challenges in cloud computing applications and services, including unreliability, latency and malicious behaviour. These challenges are usually associated with the shared environments in the hosted area or domain. In particular, security in cloud computing is one of the main problems facing outsourced data that is adopted by enterprises and public bodies. Since there is a lack of user control of the cloud infrastructure, it is hard to trust the provider of cloud service. For all these reasons an improved security system is an essential issue for protecting the data. Innovative algorithms and architectures must be developed to contribute to these challenges.

We propose an innovative architecture for distributed cloud storage to solve this security problem. The model splits user data into small chunks and shares them among the different virtual machines on the cloud resources. The next step is to grant the user (the host of the system's master node) comprehensive control of the distributed storage system. The master node (the user) controls the blocks mapping of the distributed data and maintains the tree of the namespace. In this way, the location information needed to accumulate the data is only known by the user. Thus, the security of users' data can be guaranteed even if a malicious object has access to part of the information on one of the nodes because the information will be incomplete. This proposed solution is also can be applied to avoid the malicious manner of the cloud service provider.

The proposed solution has the following features:

- A cloud system with distributed storage and a moderately trusted environment;
- chunked data spread over different VMs hosted by different service providers, which enhances the data security;
- and reliability ensured by the optimized scheduling methods using the RL algorithm and queuing theory.

To enhance the reliability in our proposed system, the incoming tasks and the utility values for the scheduling strategy are analysed by the scheduler and then reviews the better methodology to assign arriving tasks to the appropriate virtual machines. The task scheduling procedure in our model can be formulated as an MDP, in which S represents state space, A represents action set, and r(s, a) represents the immediate reward function. The task set is then assigned to the chosen scheduling strategy where it is scheduled to the appropriate virtual machine. In our model, the available VMs for the requests execution in the cloud environment are taken as a formation of state space.

As additional support for security implementation, the following aspects must be considered:

- The authentication to the cloud access should never transfer outside the cloud itself.
- The authentication server and other cloud controlling servers cannot be accessed outside the cloud.
- The data comparison and transferring are not related to the user and must be implemented inside the cloud, and no one should be associated with the model file.

4.1.1 Distributed storage

The distributed storage system has two main categories in terms of architectures: Client-server and Peer-to-Peer [Placek and Buyya 2006]. The best fit for our proposed system is a client-server architecture because it achieves the objectives of the system. In a client-server based system architecture, the service provided by the server to the clients. The server responsibilities are consistently sharing, authenticating, scheduling and serving client requests. In our model, the server's role can be held by the master node, and the client's role can be held by the slave nodes. The system works in a moderately trusted environment because slave nodes or VMs are hosted on the cloud. The biometric data are replicated through several geographical sites as a solution for improving scalability, redundancy and data availability in the cloud environment or distributed storage systems.

4.1.2 Resource allocation based on RL

There are three main components in the RL algorithm: state space, action space and immediate reward. The following is a description of how we represent those three components in our model.

As we mentioned the state space is formed by VMs that are equipped for performing the tasks in the cloud computing platform. In our system, the state space is defined as a matrix of $m \times n$ elements assuming m is the total number of the available VMs. The following matrix S represents the state space and the matrix J represents the jobs needs to be scheduled:

$$\boldsymbol{S} = \begin{bmatrix} VM_{11} & VM_{12} & \dots & VM_{1n} \\ \vdots & \vdots & \dots & \vdots \\ VM_{j1} & VM_{j2} & & VM_{jn} \\ \vdots & \vdots & & \vdots \\ VM_{m1} & VM_{m2} & \cdots & VM_{mn} \end{bmatrix} \qquad \boldsymbol{J} = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ J_{j1} & J_{j2} & & J_{jn} \\ \vdots & \vdots & & \vdots \\ J_{m1} & J_{m2} & \cdots & J_{mn} \end{bmatrix} \qquad (3)$$

The job can be successfully scheduled only if it meets the specified constrains and it illustrated by following function:

$$f(job, vm) = \begin{cases} 1, meet\ the\ constraine \\ 0, otherwise \end{cases} \tag{4}$$

Using Markov Decision Process for the job scheduling, the system's current state at the moment of scheduling is simply relevant to the preceding state. The action set is represented as 'reject' or 'receive' the job by the virtual machine. As mentioned above, 'receive' is represented by 1 and it is true only if the constraints are met (waiting time for a request is less than the mean, and the remaining queue buffer is less than the max); otherwise, 'reject' and it is represented by 0. The immediate reward is employed to return the system's running state and provide a feedback of the job scheduling efficiency.

The proposed algorithm technique operates in the following three steps: adapt action selection, update the Q value and adjust the learning coefficient. In the following scenario, we define the job scheduling mechanisms of different virtual machines (continued in section 3.3). If a job J_i is assigned to a virtual machine VM_j, we can define the expected execution time $E(J_i, VM_j)$ to be represented as the immediate reward $r(J_i, VM_j)$. The expected execution time for every job denoted as:

$$E(J_i, VM_j) = \frac{J_i ins}{VM_j ps} + \frac{J_i size}{VM_j bw} \tag{5}$$

$J_i ins$ is the number of job instructions, $VM_j ps$ is the processing speed of the virtual machine, $J_i size$ is the i^{th} job size, and $VM_j bw$ is the network band width for the j^{th} virtual machine. Alternatively, the immediate reward denoted as:

$$r(J_i, VM_j) = \frac{J_i ins}{VM_j ps} + \frac{J_i size}{VM_j bw} \tag{6}$$

To optimize the jobs scheduling and enhance the reliability in our system, we employ Q-learning. This approach (Q-learning) has demonstrated to be one of the most significant approaches in reinforcement learning, specifically for learners who do not know much about the environment. The working mechanism of our proposed system is shown in Algorithm.1.

With the continuously snowballing formats, and after every immediate reward "*r*" is claimed, the mean Q-value of the action "*a*" in state "*s*", denoted by:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * [r_{t+1} + \gamma * Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{7}$$

Where α is a learning rate parameter, and γ represents the discount rate to guarantee convergence in continuing task by the updated reward. By employing a Q-learning algorithm, the agent learns the optimal scheduling and updates the Q-value table based on the estimated results. The policy estimate model estimates the suboptimal policy of each agent and acquires a whole-situation optimal policy. The optimal scheduling continuously updates the Q-value table for the agent until the learning progress is finished. This method of updating has many advantages such as it can efficiently stabilize the exploration and exploitation process then deliver the optimal policy.

Algorithm.1 Task scheduling based on RL

```
1:   For each pair $(s_t, a_t)$, initialize the table entry $Q(s_t, a_t)$
2:   Observe the current state s
3:   Repeat
4:     Select an action $a_t$ and execute it
5:     Obtain immediate reward r
6:     Perceive the new state $s_{t+1}$
7:     Update the table entry for $Q(s_t, a_t)$ as follows:
8:     $Q(s_t, a_t) = Q(s_t, a_t) + \alpha * [r_{t+1} + \gamma * Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
9:     $s_t = s_{t+1}$
10:    $a_t = a_{t+1}$
11:  End for
```

In a large networking system, to avoid the negative influence of merging speed and optimized value and to enhance the learning process, we have created an abstract state by dividing state space *S* into subcategory "s". The proposed system selects the most proper state or VMs to allocate the tasks based on streaming buffer (VM max length waiting queue M).

$$S = M - \frac{M}{f} \; where \; f \begin{cases} 1, if\ VM\ queue\ is\ full\ (fu) \\ 2, if\ VM\ queue\ is\ middle\ (mi) \\ 3, if\ VM\ queue\ is\ over - middle\ (om) \\ 4, if\ VM\ queue\ is\ semi -\ free\ (sm) \\ 0, if\ VM\ queue\ is\ free\ (fr) \end{cases} \quad (8)$$

[Fig. 3] shows the working model of our proposed system. The system first observes the resources state from the environment, learns them and then ranks the VM according to its max length waiting queue (M) and the other attributes by the agent. The highest rank will be given to the free VM queue (the highest priority), similarly a lower rank will be given to the semi-free VM queue and less rank will be given to the middle VM queue and so on. The VM with the full queue is not in the scheduling scope (return 0). The task dispatcher dispatches the task to the VM queue with high priority, so the action is the selection of free VM $a(fr)$=0 as the first option, $a(sm)$=4 as the second and so on.

The reinforcement-based scheduling in our model is framed by the collaboration between a reinforcement-agent and resources. At a specific point of time, the resource's state *S* is observed by the system (i.e., free, full, middle). Then, according to the resource's status, the system determines the right action A for the state. The learning policy in our proposed model, consults the weight value using the resource availability function f and the given state, while the experiences evaluations are provided. When there is no prior knowledge about resources' states, rewards are compulsory for the agent to collect useful information that is needed to evaluate the experiences then the agent takes a decision (an action) based on this information. The action refers to the schedule and assigns tasks to the most satisfactory resource based on the highest priority. After the action is executed, the reinforcement system obtains the reward to specify the significance of its action. The obtained reward formulates the scheduling policy after each episode which is updated according to previous taken action considering that each action has its correspondence state. The agent refers to its scheduling policy in each step to define the most suitable action for the current state. The agent saves this triplet state-action-reward in its learning module to take better decision in the future while regularly updating the learning policy and global utility value.

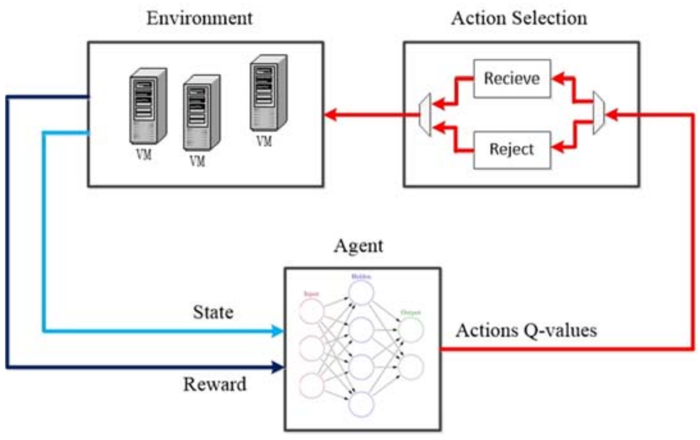


Figure 3: The basic structure of the proposed RL-based scheduling model

4.1.3 Task scheduling method

The proposed system deploys a task scheduling method created by a global task queue with a limited buffer-size and a reinforcement-learning task scheduler. In our proposed model, the arrival of the user's requests at the system is modeled as a Poisson process with a mean arrival rate of λ. In a cloud computing environment, one or more servers (VMs) may be available with mean service rate of the requests μ.

In order to achieve perfect results, the queuing system in our model is structured into three sub-models. The proposed queue model is designed using [Khazaei, Misic and Misic 2013] as a reference. However, we redesigned the model to have sub-models that interact with each other so that the output of one of them is the input of the other at the same time [Fig. 4].

The first sub-model of our queue system is represented by the global queue with a task dispatcher. This part of queue is responsible of receiving user's requests and arranging them in a finite queue using equation 8. As mentioned above, the task arrival is modelled as a Poisson process with a mean arrival λ^{glob} for the first sub-queue (global). The global queue in our system is modelled as M/M/1 queue model. The mean response time, MR^{glob}, for the first sub-queue (global) in our model is calculated taking in account the condition $\lambda^{glob} < \mu^{glob}$ as follows:

$$MR^{glob} = \frac{1/\mu^{glob}}{1\text{-}\lambda^{glob} / \mu^{glob}} \Big|_{\lambda^{glob}<\mu^{glob}} \tag{9}$$

The second sub-model (conn) is a set of parallel connections combined with a set of computing servers, and each server has a sub-buffer queue and a VM. The scenario of the process in each computing server is as follows: the dispatcher in the first sub-queue dispatches the user's request to the nominated buffer queue. After that, the VM picks the user's request from the buffer queue and delivers it to the computing server. Finally, the VM delivers the results after execution to the third part of the queue system (global transmitter sub-queue). The second sub-queue of our system (conn) is modelled as M/M/1/m if the buffer size is *m*. The probability of assigning a user's request to the i^{th} buffer queue is represented by p_i. The mean response time for the user's request to be executed in the i^{th} buffer queue is calculated as follows:

$$MR^{conn} = \frac{1/\mu^{glob}}{1\text{-}\lambda_i^{glob} / \mu^{glob}} \Big|_{\lambda^{conn}=p_i\lambda^{glob} \; and \sum_{i=1}^{N} p_i=1} \tag{10}$$

The conditions $\lambda^{conn} = p_i \; \lambda^{glob}$ and $\sum_{i=1}^{N} p_i=1$ are used to ensure the stability of the queue. The stability response time for the user's request to be allocated to any of the buffer queue is given as follows:

$$MR^{conn} = \sum_{i=1}^{n} p_i MR_i^{conn} = \sum_{i=1}^{n} p_i \frac{1/\mu^{glob}}{1\text{-}\lambda_i^{glob} / \mu^{glob}} \tag{11}$$

The third part of our queue system (trans) is responsible for receiving the executed results from the second sub-queue (conn) and transmit those results to the client or the cloud user. Similar to the first part of our queue system (glob), the third part (trans) can be also represented as a global queue but with a task transmitter instead of a task dispatcher. Thus, all the tasks involved a global queue twice, once at the arrival and once after the execution. The transmitter receipts the outcome results and transfers them back to the users. The mean of arrival rate at the transition queue (trans) and at the receiving queue (glob) is equal if there are no dropped tasks. The mean service rate for the transmitter is represented by the μ^{trans} conditioning that $\lambda^{glob} < \mu^{glob}$. This part of the queue system (tarns) is modelled as M/M/1, and the mean response time is denoted as:

$$MR^{trans} = \frac{1/\mu^{trans}}{1\text{-}\lambda^{trans} / \mu^{trans}} \tag{12}$$

The total response time in our cloud environment is calculated as follows:

$$MR^{total} = MR^{glob} + MR^{conn} + MR^{trans} \tag{13}$$

As a general scenario for task scheduling, the task dispatcher transfers the user's requests from the main or global queue to the sub-queue. The task dispatching at the designated moment of time executed following specific steps. First, according to parameters, such as the execution of pre-task conditions in VMs, the allocation remainder capacity s_i of every VM and the prediction of the execution time of the present task, the task dispatcher create a scheduling policy. Second, the i^{th} user request is assigned to the nominated sub-queue. Finally, the allocation remainder capacity updates the task dispatcher and waits for the following request. Reinforcement learning algorithm is responsible for executing the aforementioned steps. According to the state of resources, the agent takes an action after receiving the reward from the previous action.

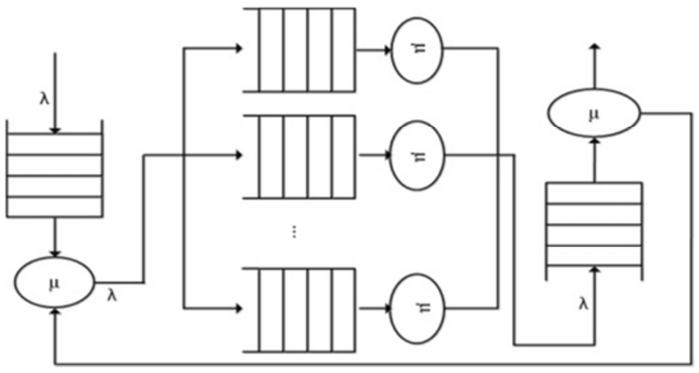


Figure 4: The queue structure in our proposed model

In the entire processing operation of the cloud computing system, the tasks will be continuously executed by a computing server until the buffer queue becomes empty. The capacity of the VMs in every buffer queue varies in real time. Hence, to decrease the response time and ensure good QoS for the user, the task dispatcher is professionally scheduling the tasks with multiple computing servers. Moreover, the task dispatcher plays an important role in the cloud environment by balancing the load between the nodes.

5 Implementation of the proposed system

Biometric technology assists users in daily life and makes the interactions of security and logging procedures run smoothly.

Some of our proposed system applications can be addressed as following:

- Use case: real-time watch list detection and identifying known individuals. The proposed system can be used for a real-time face recognition security system, because it is the fastest and most accurate under all conditions. This can help in building safer environment and enhance the security in smart houses. The steps for this case of application and other cases are almost the same and we can summarize it as: capture, process according to proposed model, assess, match, and react.
- Use case: access control: authenticating individuals accessing secured or controlled area (e.g. smart houses or offices etc.).
- Use case: Access control, the proposed system also can be used for fingerprint identification.

[Fig. 5] shows the implementation (application) of our system and how it works for cloud biometric auothentication. The following procedures can be applied to the authentication of the cloud platform in the proposed model as an application for security implementation:

a) Observing biometric data:
   1. The user can interact through the desktop fingerprint scanner or other devices in case of using another biometric method (e.g. a surveillance camera for face recognition).
   2. By using an application for converting files, a model file produces from the original fingerprint.
   3. Split the model file into small chunks and spread those small chunks on different cloud resources to make the access of the entire files difficult for unauthorized persons.
   4. The client application (user) contacts the API (Application Programming Interfaces, which allows various services to communicate with each other) and sends the model file.
   5. The API service connects with the authentication server (AS) and asks for authentication to send the model file.

b) Biometric data retrieval:
   1. The user can interact through the desktop fingerprint scanner or surveillance camera other devices in case of using another biometric method (e.g. a for face recognition).
   2. The authentication server compares the model file with its databases, which are spread over multiple cloud service providers.
   3. Once the file has been recognized, the server allows the user to access the file.

As a response to the user request on the client desktop, the proposed system uses the RL algorithm and queue theory to schedule the user requests on the cloud.

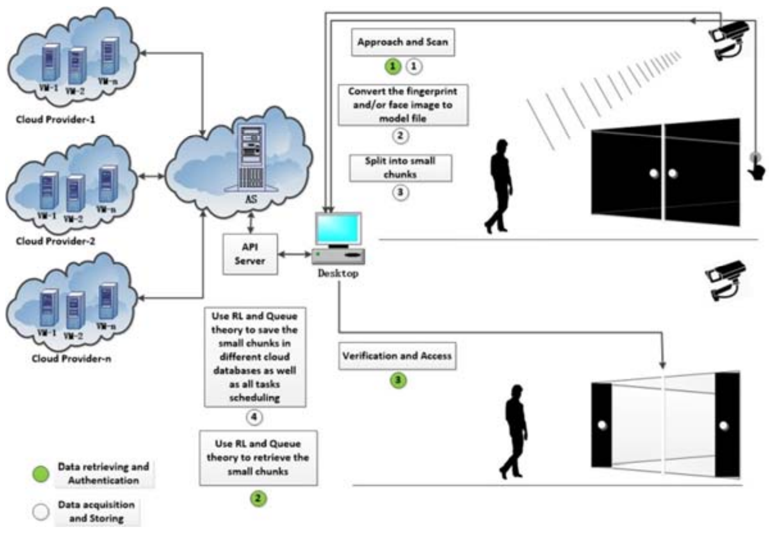

Cloud Provider-1
VM-1
VM-2
VM-n
Cloud Provider-2
Cloud Provider-n
AS
API Server
Desktop
Approach and Scan
1
Convert the fingerprint and/or face image to model file
2
Split into small chunks
3
Use RL and Queue theory to save the small chunks in different cloud databases as well as all tasks scheduling
4
Use RL and Queue theory to retrieve the small chunks
2
Verification and Access
3
Data retrieving and Authentication
Data acquisition and Storing

Figure 5: The proposed system implementation

6 Experimental results

In our experiments, we used the CloudSim framework to develop our reliability-enhancement model. CloudSim [Calheiros, Ranjan, De Rose and Buyya 2009] is an extensible framework that supports simulation modelling of service scheduling, virtual resource allocation and other functions. To evaluate our system's formance, we conducted several experiments and then compared the results with the proposed approach. The comparison of the performance was conducted with two other scheduling policies: the Greedy scheduling policy and the Random scheduling policy. [Tab. 1] shows the experimental setup in our simulation system.

Parameter	**Value**
Number of VMs	10–100
Number of tasks	10000–100000
The time interval between the cloud task	10–200 ms
VM frequency	1000 million instructions per second
VM memory (RAM)	2048 megabyte
VM bandwidth	10000 megabyte per second

Table 1: Parameters and values

The RL parameters values were set as follows: The Gamma, γ, value was set as 0.5, the Alpha, α, value was set as 0.5, the Epsilon, ε, value was set as 0.01, and the Poisson distribution λ was set as 10.0.

[Fig. 6] shows a comparison between successful tasks with the arrival rates ranging from 10,000 to 100,000. Compared with the Greedy model, the proposed model demonstrated more efficiency.

The second experiment concerned determining how the various ranges of VMs, from 10 to 100, can influence the reliable performance and the learning capability between the RL, Greedy and Random models. [Fig. 7] shows the increment in VMs can affect the reliability performance at high levels with the Greedy and Random approaches, while there is no effect with the RL method and the successful task rate is continue fixed.

In [Fig. 8] and [Fig. 9] the utilization rate is plotted with respect to the different amount of VMs and tasks numbers respectively. The figures show that our approach has achieved the highest utilization rate, which indicates the method's reliable efficiency and utility in job scheduling. This specifies that our method facilitates the allocation of appropriate resources for the requested tasks through adaptive learning experiences. Lastly [Fig. 10] shows a comparison between RL and Greedy scheduling policy in term of tasks executions under different time intervals. The proposed approach demonstrated the efficiency of the scheduling policy.

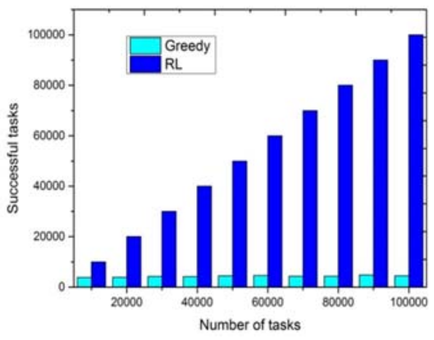


Figure 6: Comparisons of successful tasks under various task numbers

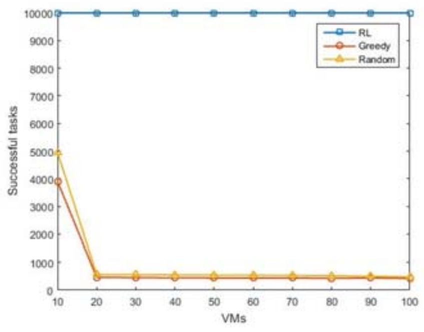


Figure 7: Comparisons of successful tasks under various VMs

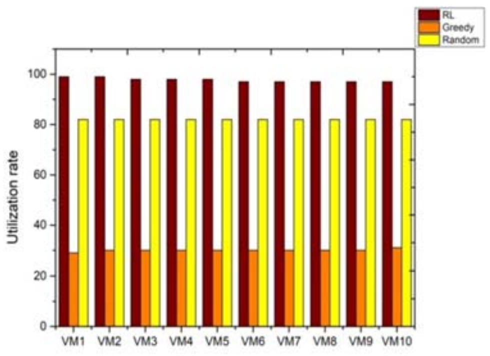


Figure 8: Utilization rate under various VMs

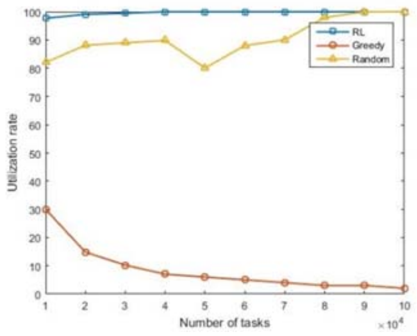


Figure 9: Utilization rate under various task numbers

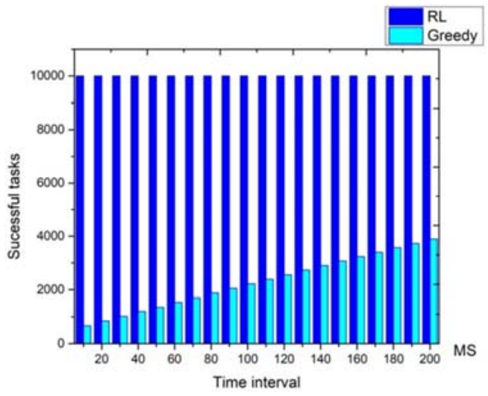


Figure 10: Comparisons of successful tasks under various time intervals

7 Conclusions

In this paper, we proposed an integrated security framework that uses a cloud biometric authentication and demonstrated how this framework can enhance reliability and security by splitting user data into small chunks and spreading it over different virtual machines. Moreover, we discussed how employing reinforcement learning and a queuing theory can enhance reliability. The system that we propose achieves reliable execution of the tasks by helping schedulers effectively adapt to dynamic changes in cloud environments. Based on the concept of reliability, we developed an adaptive action-selection method supported by queue theory, which aims to control the action selection dynamically by considering queue buffer size and the uncertainty value function. We tested our model and compared it with other methods.

8 Future Work

In future work, we are planning to expand the security framework and accelerate the work on applications and solutions developed for biometrics technology and employed in security systems by using more tools and algorithms.

References

[Ainapure, Shah and Rao 2018] Ainapure, B. S., Shah, D., Rao, A. A.: 'Understanding Perception of Cache-Based Side-Channel Attack on Cloud Environment'; In Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. Springer, Singapore (2018), pp. 9–21. https://doi.org/10.1007/978-981-10-3376-6_2

[Bauer and Adams 2012] Bauer, E., Adams, R.: 'Reliability and availability of cloud computing'; John Wiley & Sons (2012).

[Bu, Rao and Xu 2009] Bu, X., Rao, J., Xu, C.-Z.: 'A reinforcement learning approach to online web systems auto-configuration'; In Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on. IEEE (2009), pp. 2–11.

[Calheiros, Ranjan, De Rose and Buyya 2009] Calheiros, R. N., Ranjan, R., De Rose, C. A., Buyya, R.: 'Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services'; ArXiv Preprint ArXiv:0903.2525 (2009).

[Chen, Paxson and Katz 2010] Chen, Y., Paxson, V., Katz, R. H.: 'What's new about cloud computing security'; University of California, Berkeley Report No. UCB/EECS-2010-5 January, Vol. 20, No. 2010 (2010), pp. 2010–5.

[Hussin, Hamid and Kasmiran 2015] Hussin, M., Hamid, N. A. W. A., Kasmiran, K. A.: 'Improving reliability in resource management through adaptive reinforcement learning for distributed systems'; Journal of Parallel and Distributed Computing, Vol. 75 (2015), pp. 93–100.

[Khazaei, Misic and Misic 2013] Khazaei, H., Misic, J., Misic, V. B.: 'A fine-grained performance model of cloud computing centers'; IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 11 (2013), pp. 2138–2147.

[Kochovski and Stankovski 2018] Kochovski, P., Stankovski, V.: ‘Dependability of Container-Based Data-Centric Systems’; In M. Ficco & F. Palmieri (Eds.), Security and Resilience in Intelligent Data-Centric Systems and Communication Networks. Academic Press (2018), pp. 7–27. https://doi.org/10.1016/B978-0-12-811373-8.00001-X

[Little and Graves 2008] Little, J. D., Graves, S. C.: ‘Little’s law’; Springer (2008), pp. 81–100.

[D. Liu, Wei, Wang, Yang and Li 2017] Liu, D., Wei, Q., Wang, D., Yang, X., Li, H.: ‘Adaptive Dynamic Programming with Applications in Optimal Control’; Springer (2017).

[X. Liu, Tong, Zhi, ZhiRen and WenZhao 2014] Liu, X., Tong, W., Zhi, X., ZhiRen, F., WenZhao, L.: ‘Performance analysis of cloud computing services considering resources sharing among virtual machines’; The Journal of Supercomputing, Vol. 69, No. 1 (2014), pp. 357–374.

[Microsoft, white paper 2013] Microsoft, white paper: ‘Resilience by design for cloud services A structured methodology for prioritizing engineering investments’; (2013, May).

[Peng, Zhang, Peng and Man 2017] Peng, C., Zhang, C., Peng, C., Man, J.: ‘A reinforcement learning approach to map reduce auto-configuration under networked environment’; International Journal of Security and Networks, Vol. 12, No. 3 (2017), pp. 135–140. https://doi.org/10.1504/IJSN.2017.084342

[Placek and Buyya 2006] Placek, M., Buyya, R.: ‘A taxonomy of distributed storage systems’; Reporte Técnico, Universidad de Melbourne, Laboratorio de Sistemas Distribuidos y Cómputo Grid (2006).

[Ranjbari and Torkestani 2018] Ranjbari, M., Torkestani, J. A.: ‘A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers’; Journal of Parallel and Distributed Computing, Vol. 113 (2018), pp. 55–62.

[Robertazzi 2012] Robertazzi, T. G.: ‘Computer networks and systems: queueing theory and performance evaluation’; Springer Science & Business Media (2012).

[Srinivasan, Sarukesi, Rodrigues, Manoj and Revathy 2012] Srinivasan, M. K., Sarukesi, K., Rodrigues, P., Manoj, M. S., Revathy, P.: ‘State-of-the-art cloud computing security taxonomies: a classification of security challenges in the present cloud computing environment’; In Proceedings of the international conference on advances in computing, communications and informatics. ACM (2012), pp. 470–476.

[Tesauro, Jong, Das and Bennani 2006] Tesauro, G., Jong, N. K., Das, R., Bennani, M. N.: ‘A hybrid reinforcement learning approach to autonomic resource allocation’; In Autonomic Computing, 2006. ICAC’06. IEEE International Conference on. IEEE (2006), pp. 65–73.

[Vengerov 2007] Vengerov, D.: ‘A reinforcement learning approach to dynamic resource allocation’; Engineering Applications of Artificial Intelligence, Vol. 20, No. 3 (2007), pp. 383–390.

[Wu, Xu, Zhang and Liu 2011] Wu, J., Xu, X., Zhang, P., Liu, C.: ‘A novel multi-agent reinforcement learning approach for job scheduling in Grid computing’; Future Generation Computer Systems, Vol. 27, No. 5 (2011), pp. 430–439.