Deep Learning Based Security Model for Cloud based Task Scheduling

K. Devi[1*], D. Paulraj[2] and B. Muthusenthil[3]
[1] Department of CSE, SRM Valliammai Engineering College
Chennai,India
[e-mail: devii.jeya@gmail.com]
[2] Department of CSE, RMD Engineering College
Chennai-India
[e-mail:kingrajpaul@gmail.com]
[3] Department of CSE, SRM Valliammai Engineering College
Chennai-India
[e-mail:bmssen@gmail.com]
*Corresponding author:K. Devi

Received April 30, 2020; revised July 6, 2020; accepted July 30, 2020; published September 30, 2020

Abstract

Scheduling plays a dynamic role in cloud computing in generating as well as in efficient distribution of the resources of each task. The principle goal of scheduling is to limit resource starvation and to guarantee fairness among the parties using the resources. The demand for resources fluctuates dynamically hence the prearranging of resources is a challenging task. Many task-scheduling approaches have been used in the cloud-computing environment. Security in cloud computing environment is one of the core issue in distributed computing. We have designed a deep learning-based security model for scheduling tasks in cloud computing and it has been implemented using CloudSim 3.0 simulator written in Java and verification of the results from different perspectives, such as response time with and without security factors, makespan, cost, CPU utilization, I/O utilization, Memory utilization, and execution time is compared with Round Robin (RR) and Waited Round Robin (WRR) algorithms

Keywords: Cloud Computing, Distributed computing, Task scheduling, deep learning, makespan

1. Introduction

Scheduling provides a significant contribution in cloud computing to quickly and easily assign resources for each mission. Project scheduling in the cloud computing environment is used to evaluate appropriate resources for execution of assignments by considering some constraint and parameter. The scheduler, user and virtual machine (VM) clusters are essential components needed to schedule the work in cloud. A user hands over the tasks to schedulers in the cloud environment. The scheduler organizes tasks as per the task requirements, then delivers tasks to VM and at last the user gets the final output from the scheduler. The scheduling can be distinguished according to the execution time as static and dynamic planning.

Scheduling of cloud computing can be divided into three phases

- Resource identification and filtering – The data center broker decides and gathers the status information related to the resources available on the network environment.
- Selection of resources – Resources are chosen according to agreed mission and resource parameters
- Task submission –Submit the tasks in the chosen resources.

The principle goal of the scheduling is to reduce competition for resources and to ensure that parties use resources equally.Scheduling tackle the question for which most of the essential resources will be allocated.

The demand for resources fluctuates dynamically so scheduling of resources is a difficult task. A task scheduler in cloud computing has to fulfil cloud customers with the agreed quality of service (QoS) and improve the income of cloud providers. It is a major difficulty to dispatch effectively and reasonably the tasks of the users for specific sources following the QoS necessities of each cloud computing center and users. In critical application, many scheduling strategies are being used by the master nodes to efficiently distribute its tasks. As the number of cloud users increases, the scheduling becomes very difficult and a suitable scheduling algorithm is required. Appropriate scheduling algorithms are needed to undeniably monitor the various problems and limitations associated with different scheduling techniques.

Throughout the cloud setting various forms of task scheduling have been used. Those include QoS-based scheduling, cost based scheduling, cluster based scheduling, priority scheduling, fuzzy based scheduling, ant colony-based scheduling, particle swarm optimizations algorithm-based job scheduling, genetic algorithm-based scheduling and multiple-processor-based scheduling [1]. In a non-preemptive method, the two most important scheduling concepts are round Robin and weighted round robin strategies.

The round robin algorithm assigns the next VM task in the queue regardless of the load on the VM. The Round Robin strategy does not take into account the resources, priorities and length of tasks. Higher priority and longer activities end up with higher response times. The weighted round robin considers VMs' resource capabilities and assigns increased number of tasks to larger capacity VMs based on the weight given to each VM. But when selecting the appropriate VM it fails to consider the duration of the tasks. These two algorithms are implemented for comparative analysis.

Cloud computing security is one of the main cloud problems. The security-aware scheduling is essential to run the takes in cloud-based customer security specification.

The main findings are as follows:

1.Designed a security model to schedule the tasks in cloud computing.

2.Proposed the deep reinforcement learning for task scheduling in cloud computing.

3.Implementation of the model adopted using CloudSim 3.0 simulator written in Java and verification of the results is also done.

4.In-depth performance analysis from different perspectives, such as response time with and without security factors, makespan, communication cost, CPU utilization, I/O utilization, storage overhead, and execution time.

The road-map of the paper is as follows. In section 2, we review the related work. In Section 3, the system model works as per the following stages.

Stage 1: This stage consists of various tasks and allocate them to the queuing system by considering the execution time and deadline requirements.

Stage 2: Concern with security level classification based on the task load, number of resources required and security requirements. Hence all the tasks can be assigned to a respective security level servers.

Stage 3: Assigning tasks from security level servers into VMs by ensuring proper security level. The output of the system is a n x n matrix of total cost.

In sections 4 model implementation, performance evaluation is done that demonstrates the effectiveness of our model. Finally, we quoted conclusion remarks.

2. Related Work

Guo et al. [2] developed the task scheduling for cloud management system and express a task schedule model that minimizes and resolves the problem through a PSO. They have examined and measured this method on the basis of particle swarm convergence, mutation and local search algorithm. Experimental tests show that the PSO algorithm finds the optimal solution and converges faster than other approaches in major tasks. But they have not considered the energy efficiency and service availability.

Gomathi et al. [3] proposed a hybrid PSO-based task scheduling algorithm, which improves PSO, decreases average running time, improves the usage of resources and provides users with sufficient resources.The experimental results shows that HPSO based task scheduling can attain better load balancing as compared to PSO based scheduling However, this approach does not contribute to broad-based optimization.

Alkayal et al.[4] designed a task scheduling based on a new grading methodology using the PSO algorithm. It tests the three aim functions: processing ECT, TEC, and VM. In MOPSO task scheduling the results of the grading techniques are used to determine the best virtual machine for each job. The tasks in this technique were designed for the VMs to reduce waiting times and boost the system performance.

Wu et al.[5] proposed an algorithm for the QoS task scheduling. In this algorithm, the user's right, the task length ,the expectation and the pending waiting time are combined to determine the goals and prioritize the tasks in consideration. The experimental results indicate that the algorithm achieves good efficiency and load balance through QoS driving driving preference and execution.

Ali et al. [6] suggested a clustered algorithm for software-based cloud activities. This algorithm incorporates various task attributes such as user category, task importance, task size, and task latency to calculate the task priority. The experimental results indicate that the GTS algorithm provides minimum run time for all tasks and minimum latency for various tasks is obtained compared to both Min-Min and TS algorithms.

Agarwal et al.[7] suggested cloud priority scheduling frameworks. This model is designed to reduce the execution time of tasks. VMs are prioritized in conjunction with a million instructions per second. Activities with the lowest priority are scheduled for VMs with the highest priority. The algorithm results are compared with the first fit (FF) and round robin (RR) algorithms, which have higher GPA performance than FF and RR.

Mehranzadeh et al. [8] proposed fuzzy logic for task scheduling. This scheduling method is capable of scheduling data center VMs. By comparing it with the two scheduling techniques of FCFS and RR, the results show the effectiveness of the algorithm. This algorithm affects outside priorities when several jobs are scheduled, and the creation of rules is a very difficult task for fuzzy logic as it affects time for calculation

Zhang et al.[9] developed a task scheduling algorithm in cloud computing focused on fuzzy clustering with a parallel approach. Their research focuses on parallel scheduling, particularly in the oil and seismic scanning industries, with particular emphasis on computing with high presentations which is necessary for huge data processing. The main disadvantage of this clustering method is that the cluster descriptor has no interpretability

Niazmand et al. [10] provided an enhanced ant colony optimization Algorithm to arrange grid computing tasks. The JSWA algorithms measure parameters like latency, requests, reliability, costs and recognition time. To reduce overall execution costs, Pandey et al. [11] developed a scheduling approach using PSO. They compared the PSO and the BRS algorithms, indicating that compared to the BRS PSO saves three times as much as the cost. However, data transfers between one compute node to the next take more and more time to transmit and store.

Feng et al. [12] proposed a task scheduling methodology using PSO algorithm and the Pareto dominance theory. This theory determines optimized schedulers for the multi-objective optimization of resources based on the registration of the resource, total execution time and QoS for each task. This method only works for basic tasks without a convergence principle for problem solving.

Juan et al.[13] suggested a cloud-based work scheduling technique using an improved algorithm based on PSO. They developed a cost vector model to measure preparation costs and a solution based on input tasks and QoS parameters.Even though the method has lot of complexity, provides an effective improvement in the scheduling.

Alkayal et al.[14] used the PSO algorithm to build a new ranking technique in multi-object-based scheduling of tasks. Here the tasks were designed for the VMs in order to minimize waiting time and increasing the device throughput. Dordaie et al.[15] suggested a work scheduling algorithm in the cloud using hybrid PSO and hill-climbing algorithm to solve the difficulties. This approach was well designed, but it requires more time to accomplish the task. Likewise, Verma and Kaushal[16] have clarified the multi-objective hybrid PSO algorithm for scientific workflow scheduling.

Gao et al. [17] introduced multi-objective function in job shop scheduling. In order to reduce execution time and preparation costs, they used the versatile job scheduling method.Keshanchi et al. [18] have developed an enhanced genetic algorithm and priority queues to plan tasks in the cloud environment. Rare selection elitism technique was used to avoid early convergence and randomly generated graph statistical analyzes were performed.

Shishido et al.[19] suggested a technique for scheduling using genetic algorithms. Here, they measured the scheduling efficiency using a security algorithm and the cost-conscious programming workflow. Su et al. [20] described a cost efficiency-based task scheduling methodology for the execution of large cloud programs.They put forward two heuristic strategies for scheduling the task in the cloud environment. The first technique, based on the concept of Pareto dominance during runtime, maps the tasks for most economical VMs. The second method eliminates non-critical activities' monetary costs.

Karatza [21] proposed a methodology for gang scheduling depend on the clustering systems. Gang schedulation is a process that deals with the planning strategy of parallel and space-sharing systems. The migration strategy is used to reduce the disruption in the schedule caused by the planned employment of gangs. Two homogeneous clusters have been replicated to determine the presentation of specific workloads. The effect of transition on the service time of parallel tasks was addressed. If this algorithm is used for task planning, it shows that the fusion or splitting decision is incompetent to correct.

3. Deep reinforcement learning-based security

3.1 User workload model

Fig. 1. Shows the security model for task scheduling in cloud computing.
This model composed of four models:

1. User workload model
2. Security classifier model
3. Deep reinforcement learning-based task scheduling model
4. Price model with security

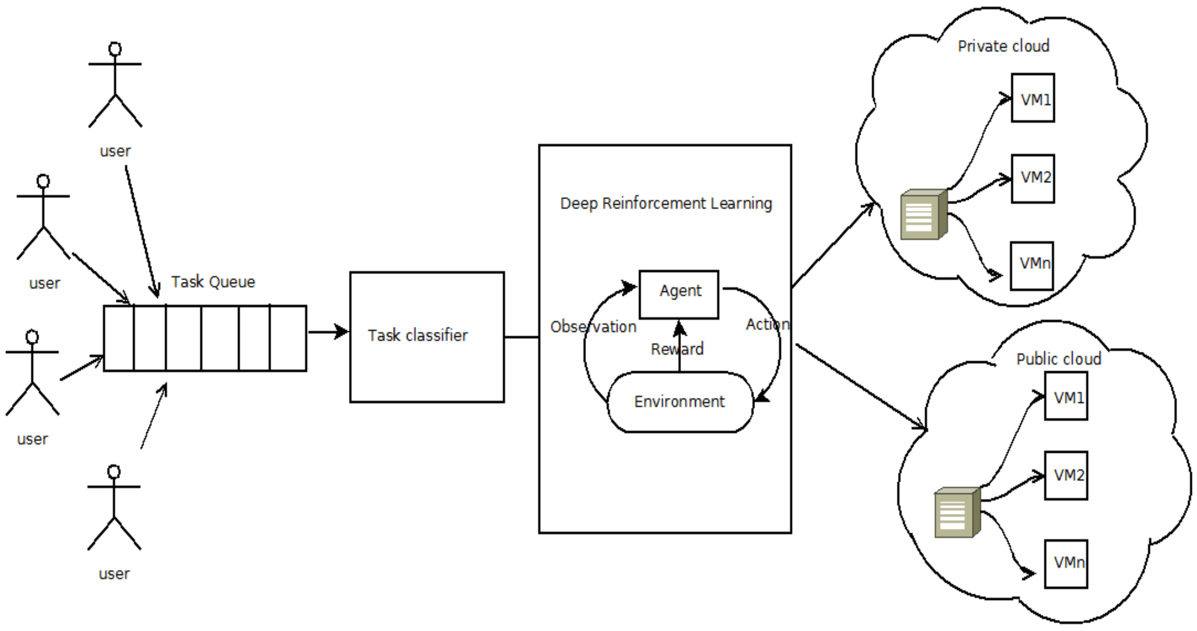

Fig. 1. Security model for task scheduling in cloud computing

3.1 User workload model

In the cloud environment, there are *n* tasks to be processed and *x* number of VMs available. Each VM is associated with two parameters VM_{CPU} and VM_{MEM}. Each task is associated with 6 parameters T_{CPU}, T_{MEM}, T_{ET}, $T_{DT,}$ T_{IC} and T_S. **Table 1** gives the notations and its descriptions used in the model.

Table 1. Notations and their descriptions

Notations	Description
VM_{CPU}	Amount of CPU available in the VM
VM_{MEM}	Amount of memory available in the VM
$VM_{I/O}$	Amount of memory available in the VM
T_{CPU}	Amount of CPU required for task execution.
T_{MEM}	Amount of memory required for task execution.
$T_{I/O}$	Amount of I/O required for task execution.
T_{ET}	Estimated time for task execution
T_{DT}	Deadline time for task execution
T_{IC}	Instruction count of the task
T_S	Security level of task
T_{start}	Scheduled start time provided by cloud service provider CSP
S_{CPU}	Capacity of CPU
S_{MEM}	Capacity of Memory
$S_{I/O}$	Capacity of I/O
R_{CPU}	CPU utilization ratio
R_{MEM}	Memory utilization ratio
$R_{I/O}$	I/O utilization ratio

Algorithm

Input: Set of tasks $T= (t_1, t_2,\ldots\ldots t_n)$ with deadline time $T_{DT}= (t_{1DT},t_{2DT}\ \ldots..,t_{nDT})$ and instruction count of task $T_{IC}=(t_{1IC},t_{2IC},\ldots\ldots.t_{nIC})$
Output: Task loaded into the queue or rejected

```
1: Initialize $CPI$ = constant value and clock cycle $CT$= constant value
2.   for $i$=1 to $n$ do
3:     Compute execution time $ET = CPI * t_{iIC} * CT$
4:       if  $ET + T_{start} > t_{iDT}$
5:            Place task inside the task queue
6:       else
7:            Reject the task
8:  end
```

3.2 Security classification model

The task submitted by the user contains CPU usage, task size and memory to identify the resource demand of the task and security level. Here tasks are classified based on the usage of CPU, memory and I/O into three categories such as CPU intensive, memory-intensive and I/O intensive. The capacity the system is as recognized S_{CPU}, $S_{I/O}$, and S_{MEM}. Then, Task Category(TC) measures the ratios within the program for each task of CPU, I/O, and memory. For each task *Ti*, calculate these ratios ($R=T/S$) R_{CPU}, $R_{I/O}$, and R_{MEM} by its parameters T_{CPU}, $T_{I/O}$, and T_{MEM} and S_{CPU}, $S_{I/O}$, and S_{MEM}. The largest of these three ratios is regarded as the task category.

$$TC = \max\,(R_{CPU}, R_{I/O}, R_{MEM}) \tag{1}$$

Finally, all tasks are divided into three queues CPU_{TC}, I/O_{TC}, and MEM_{TC} of CPU intensive, I/O intensive, and memory-intensive by the task category *TC*.
In the cloud data center CSP group the VMs into three levels based on the usage of CPU, memory and I/O to provide the security to the tasks. For instance, some tasks may require less security that may be loaded in level 1. If the task requires medium-level security, it is loaded into level 2 and the high secure tasks are loaded into level 3.
Level 1 contains only CPU intensive tasks. The operational mode of the encryption algorithm is offline and the RSA algorithm is used with the key size of 1024 bits for both encryption and decryption.
Level 2 contains I/O intensive tasks. Operational modes are both offline and online and Advance cryptography protocol is used to provide promising security. RSA algorithm is used for encryption and decryption using the key size of 2048 bits.
Level 3 contains Memory intensive tasks. The operational mode of the algorithm is only offline. RSA algorithm of key size 2048 bits and elliptic curve signature algorithm I of key size 164 bits used for encryption and decryption.

$$CPU_{TC} = \{U_1, U_2, \ldots, \ldots, U_i\}$$
$$I/O_{TC} = \{U_{i+1}, U_{i+2}, \ldots, \ldots, U_j\}$$
$$MEM_{TC} = \{U_{i+j+1}, U_{i+j+2}, \ldots\ldots, U_{n-i-j}\}$$

Here, all the resources sort rather than *classify*, due to the amount and dynamism

Proposed Algorithm for task classification.

Input: Set of taks (*T*1, *T*2,.., *Tn*) *with* T_{CPU}, $T_{I/O}$, and T_{MEM} and S_{CPU}, $S_{I/O}$, and S_{MEM}
Output: Task queues CPU_{TC}, I/O_{TC}, MEM_{TC}

```
1:  for $i$=1 to $n$ do
2:    Calculate the task ratio of CPU, I/O and
      memory
4:    $R_{CPU}=T_{CPU}/S_{CPU}$
5:    $R_{I/O}=T_{CPU}/S_{CPU}$
6:    $R_{MEM}=T_{MEM}/S_{MEM}$
7:    if (max($R_{CPU}$, $R_{I/O}$, $R_{MEM}$) == $R_{CPU}$ then
8:          $TC \rightarrow CPU_{TC}$
9:     end
10:    else if (max($R_{CPU}$, $R_{I/O}$, $R_{MEM}$) == $R_{I/O}$ then
11:       $TC \rightarrow I/O_{TC}$
12     end
13:    else if (max($R_{CPU}$, $R_{I/O}$, $R_{MEM}$) == $R_{MEM}$ then
14:       $TC \rightarrow MEM_{TC}$
15:    end
16: end
17:  Sort the all three queues in ascending order
```

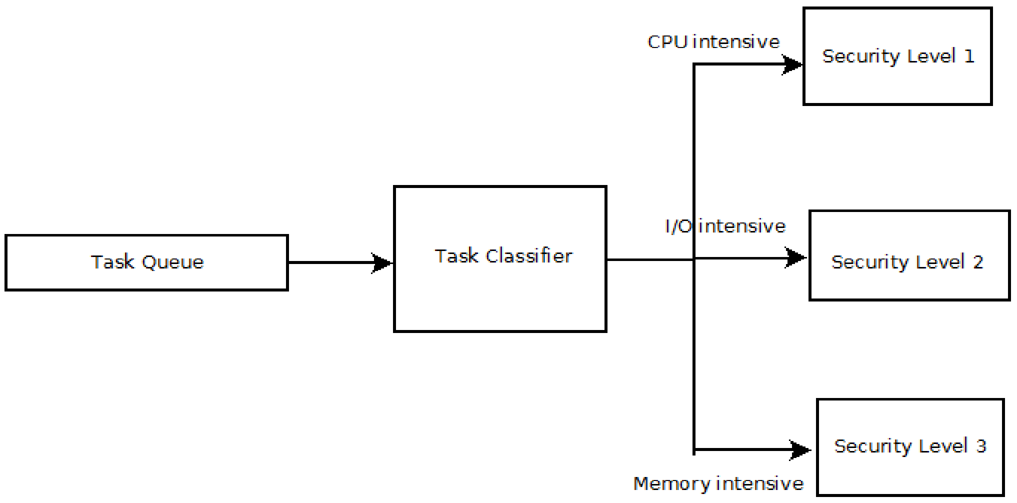

Fig. 2. Security level classifier

Based on the type of task the tasks are classified into three security levels in **Fig. 2 and Fig. 3**.

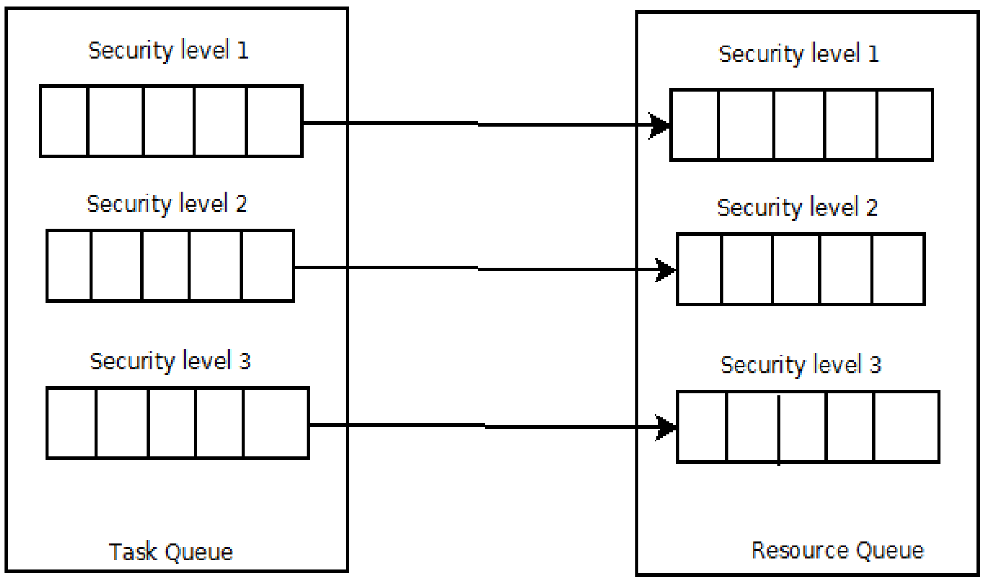

Fig. 3. Task allocation

3.3 Deep Mapping Algorithm for VM allocation

The Deep Mapping reinforcement algorithm is used to map the tasks to corresponding virtual machines. The algorithm for the training of deep queueing networks is updated using the expert replay and target network, so that a large neural network with a high converging speed can be created.

Experience replay:

The internal loop of the algorithm stores the random task t_r in memory Δ and uses the queue-learning algorithm to match the randomly picked experience from the collected samples. This is the best way to learn regular queue in many respects. The efficiency of task size is higher, since each step is highly replayed with many weight updates many times. With the help of randomly selected experience, the learning experience provides higher efficiency than the sequential experience. The randomly selected task experience make our procedure stable.

Target network:

A neural network model used in deep queue learning to generate target VM IDs. The target VM id has structured with different parameters. The parameter of target VMs are CPU, I/O and memory response time. In every γ step the parameters of the target VM IDs are evaluated from the evaluation network. Whenever the mismatch happens between standard queue learning and deep learning network eliminates the divergence.

Deep mapping into VMs:

It is a puzzle, which is aimed to explore new VMs without specifying time to respond. This method works with a greedy method that reduces the value (which we prefer to have a greater value) to select an altered action; otherwise, choose the highest answer time and reduce the factor to a minimum value in the next cycle. This approach provides good response time VM IDs

Deep Mapping Algorithm

```
1:Initialize historical memory dataset Δ to capability Ω
2:Initialize DL time δ test-deadline Q
3:for instance  = 1, E  do
4:     Set the initial cloud environment
5:     Start sequence $s_1=\{x_1\}$
6:      for i = 1, T do
7:          With probability e, select random task $t_i$
8:           Otherwise, choose $t_i$=max Q($s_i$, t, δ)
9:            Execute $t_i$ and observe next $x_{i+1}$
10:             if *reject* == 1 then
11:                 Run *DQN* again to get new task $t_i$'
12:                  if $t_i' \neq t_i$ then
13:                        $t_i \rightarrow t_i'$
14:                   end
15:             end
16:              Set $s_{i+1} = s_i, t_i, x_{i+1}$
17:             Store $VM(s_{i+1}, t_i, VM_i, s_t)$ in Δ
18:             $target_j$=  {$VM_j$ if episode terminates at step j+1
19                              $VM_j + \zeta$ max Q($s_{j+1}$,t', δ'), otherwise)}
20:         Every step train convolution network deeply $\xi$
21:         Every step copy $Q$ to $Q$'
22:          end
```

```
23:  end
24: return all task $VMs$ $VM$ IDs, $Ts$
```

3.4 Deep Reinforcement learning Price model

In this section, we describe the price estimation model for Iaas in public/private clouds. Each virtual machine in the cloud can be either active or inactive, so the price of the virtual machine is considered by the both active and inactive.

C^{a}_{vm} - denotes the cost of active virtual machine

C^{ia}_{vm} - denotes the cost inactive virtual machine

C^{a}_{vm} - includes the sum of networking cost, storage cost and computational cost.

$$C^{a}_{vm} = C^{a}_{nw} + C^{a}_{st} + C^{a}_{Comp} \tag{2}$$

C^{a}_{nw} is calculated by cost of virtual machine and the utilization time of the virtual machine.

$$C^{a}_{nw} = P^{T}_{vm} * \text{usage time } t \tag{3}$$

C^{a}_{st} is determined by the virtual machine usage time and the storage function which involves the total number of I/O activities ,CPU utilization and memory usage for a given time.

$$C^{a}_{st} = P^{T}_{vol} * t + P^{T}_{io} * t + P^{T}_{CPU} * t \tag{4}$$

C^{a}_{Comp} is determined by the NW-Number of workloads under virtual machine, CW-complexity of work load ,SR-security requirements and MR-monitoring requirements.

$$P^{a}_{comp} = ((NW * CW) * SR) + ((NW * CW) * MR) \tag{5}$$

$$C^{a}_{comp} = P^{a}_{comp} * t \tag{6}$$

Monitoring requirements are set to the range between 1 -5.These values are set based on the shift .Morning shift -1, Night shift -2, Evening shift -3, and general shift 4 and 24/7 shift.
Workload values are set between the scales 1-2.200MB task scale value-1 and the 1000 MB task scale 2.
The total cost value is determined by the following formula

$$C = C^{a}_{vm} + C^{ia}_{vm} \tag{7}$$

4. Experimental results and discussion

4.1 Experimental setup

In this proposed work, we have adopted the CloudSim 3.0 simulator written in Java. Cloud simulator is used to create cloud data centers (DC), VM, computational resources and the management of cloud systems such as scheduling and provisioning of VM. **Table 2** gives the hardware requirements of the proposed work. **Table 3** gives the software requirements of the proposed work.

Table 2. Hardware requirements

Component	Specification
Operating System	Windows 10 64 bit-OS
Processor	Intel® Pentium® Core™i7-6700T CPU@2.80 GHz
RAM	8.00 GB
System type	64 bit OS
Hard disk	I TB

4.2 Performance metrics

The main objective of our proposed model is to obtain efficiency in terms of execution time, make-span, task scheduling and load balancing along with the security factors.

Table 3. Software requirements

Entities	Specifications	Range
Cloudlets/tasks	Number of tasks	10-200
	Length(CPU)	400 - 1000 MIPS
	File Size	1000 MB
	Task length	1500MIPS-3000MIPS
VM	Host	10-100
VM/physical machine	Storage	300GB
	Bandwidth	100mbps
	Memory	4096MB
	Buffer Capacity	20
	MIPS/PE	1860/2660
	Bandwidth Cost	0.2/MB

- Efficiency is reflected through the execution time of the tasks in the model
- Scheduling efficiency is reflected through the deadline verification and response time.
- Load balancing is efficiency is achieved through the utilization of the CPU, memory and IO resources.
- Tasks scheduling efficiency is achieved through the deep learning model.
- Security factors are implemented through the classification of the level discussed earlier.

Response time:

The basic experiment is verified by the response time- they take arrival rates in terms of task size were 1500 - 3000 KB. The results are shown in **Fig. 4**. It shows the response time of all the tasks size with security and without security. Our proposed model shows a greater impact. Comparison of execution time with and without security factors are tabulated. Even though the security model takes a little higher response time compared with normal repose time, it will reflect a greater impact on overall system performance.
Response time=the time interval between the task arrival and the completion

$$TR_e = T_c - T_a \tag{8}$$

TR_e- task response time
T_c – task execution time
T_a- task arrival time

$$Tc=(T_{DLT}+T_{transfer}+T_{Exe}+T_{Security}) \tag{9}$$

T_{DLT} - Deep learning algorithm execution time of the task
$T_{transfer}$ – Time taken to transfer the task
T_{Exe} – Actual execution of the task
$T_{Security}$ –Time to run the security algorithms.

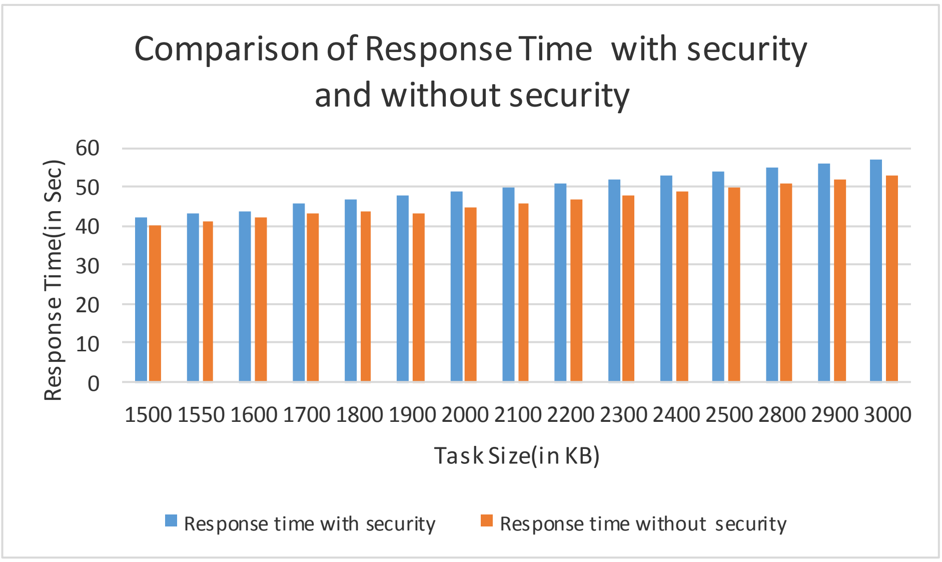

Fig. 4. Response time of all the tasks size with security and without security.

Execution time of the task is measured using the equation number 9.The results are compared with the RR and WRR Algoithms.The compartive analysis is given in the **Fig. 5**.

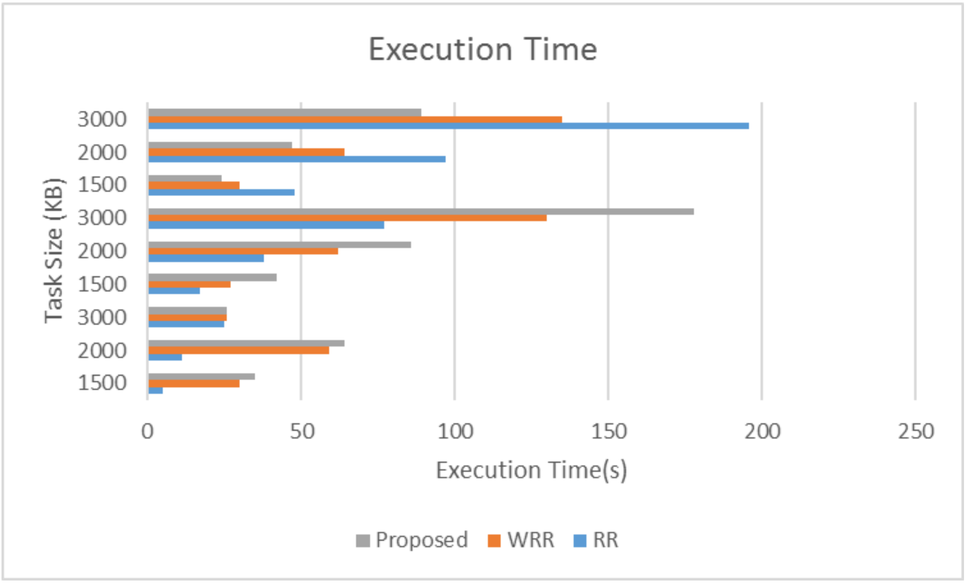

Fig. 5. Execution time Comparison

Makespan:

Makespan is used to determine the optimal completion period by comparing the last task's completion time when all tasks are scheduled. Here T_{ij} defines the time that resource r_i needs to complete task t_i.

$$\text{Makespan} = \max\{T_{ij} \text{ for i tasks mapped to j VM}\} \quad (10)$$

In **Fig. 6** makespan time is compared to Round Robin (RR), Weighted Round Robin (WRR) and proposed approach.Our proposed approach takes lesser the makespan compared to other two methods.

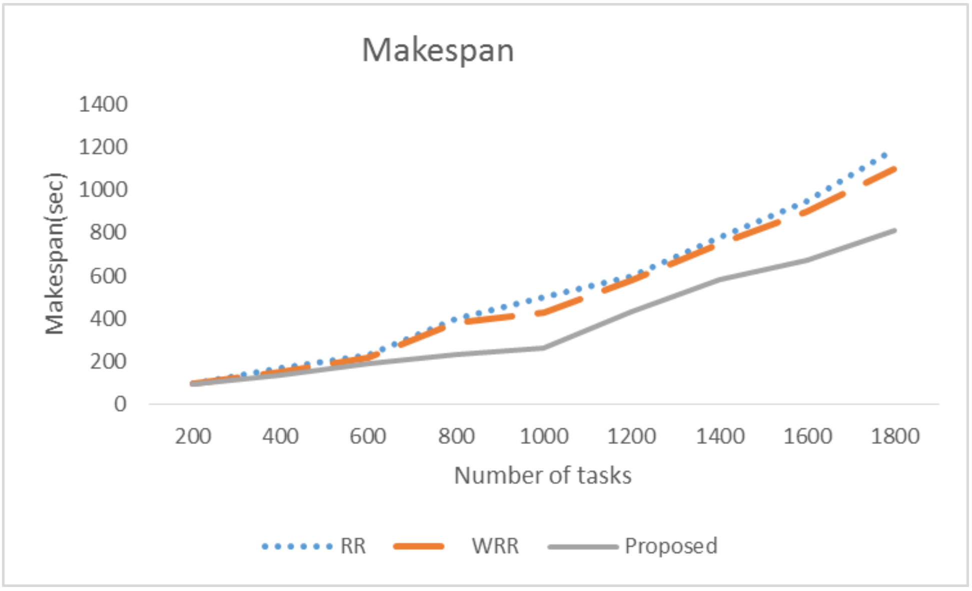

Fig. 6. Makespan Comparison.

Load balancing:

The load balancer selects each of the VMs that completes all of their assigned tasks, then chooses the heavily loaded VM from the list and calculates the completion time of those tasks in the heavily loaded VM and the total loaded / idle. If the minimum loaded VM can complete any of the jobs present in the severely loaded VM in the shortest possible time, then that job will be migrated to the minimum loaded VM. This load balancing factor is measured through the utilization of CPU, memory and I/O resources in the private cloud. The results are shown in **Table 4** and **Fig. 7**

Table 4. Resource Utilization

Number of requests	RR			WRR			Proposed		
	CPU	MEM	I/O	CPU	MEM	I/O	CPU	MEM	I/O
50	0.25	0.4	0.65	0.22	0.38	0.58	0.18	0.36	0.48
100	0.2	0.35	0.55	0.18	0.35	0.5	0.16	0.35	0.4
150	0.15	0.3	0.45	0.15	0.3	0.45	0.15	0.3	0.35
200	0.05	0.15	0.15	0.05	0.15	0.2	0.05	0.15	0.1

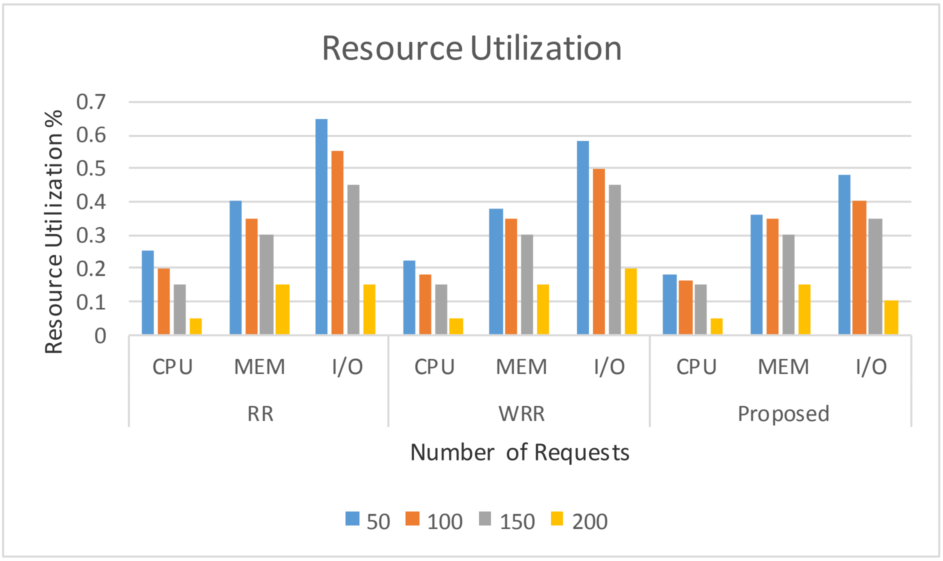

Fig. 7. Utilization of CPU, memory and I/O resources comparison

Cost

Cost of the task execution is based the number of virtual machines used for execution. The cost computation is performed based on the equation number 7 mentioned in the previous section. Fig. 8 shows the comparative analysis of cost involved in task execution. The proposed method takes the less cost compared to the RR and WRR.

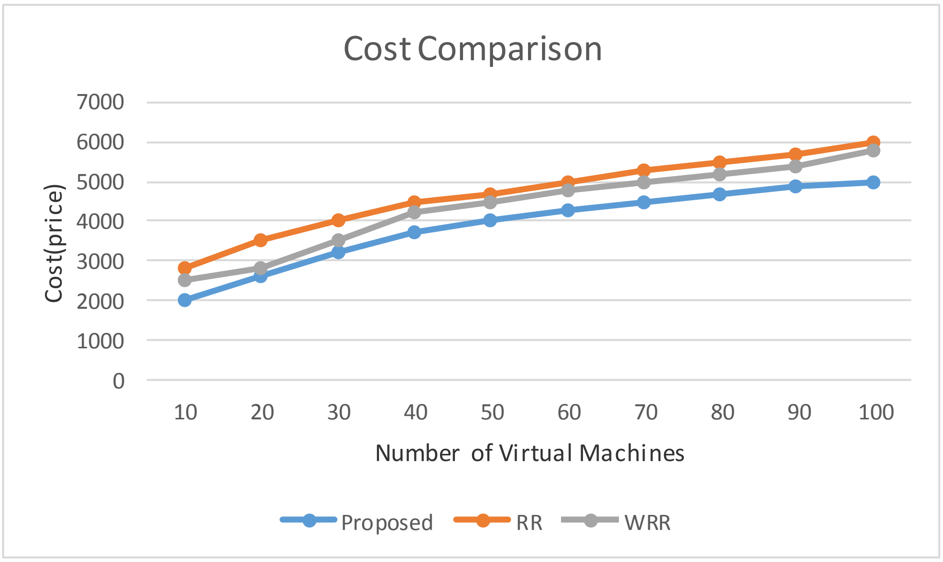

Fig. 8. cost comparison

5. Conclusion

This paper presented a new approach to design a deep learning-based security model for task scheduling in cloud computing. To achieve this, we implemented using CloudSim 3.0 simulator written in Java and verification of the results from different perspectives, such as response time with and without security factors, makespan, cost, CPU utilization, I/O utilization, memory utilization, and execution time is also verified with RR, WRR and our model is done. The experiments show that our model outperforms the ad-hoc heuristics.

References

[1] A. R. Aruna rani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Futur. Gener. Comput. Syst*., vol. 91, pp. 407-415, 2019 Article (CrossRef Link)
[2] L. Guo, S. Zhao, S. Shen, and C. Jiang, “Task scheduling optimization in cloud computing based on heuristic Algorithm,” *J. Networks*, vol. 7, no. 3, pp. 547–553, 2012.
[3] B. Gomathi and K. Krishnasamy, "Task scheduling algorithm based on Hybrid Particle Swarm Optimization in the cloud computing environment," *J. Theor. Appl. Inf. Technol.*, vol. 55, no. 1, pp. 33–38, 2013. Article (CrossRef Link)
[4] E. S. Alkayal and N. R. Jennings, “Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing,” in *Proc. of 41st Conf. Local Comput. Networks Workshops*, pp. 17–24, 2016. Article (CrossRef Link)
[5] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, “A task scheduling algorithm based on QoS-driven in Cloud Computing,” *Procedia Comput. Sci*., vol. 17, pp. 1162–1169, 2013. Article (CrossRef Link)
[6] H. Gamal El-Din Hassan Ali, I. A. Saroit, and A. M. Kotb, "Grouped tasks scheduling algorithm based on QoS in a cloud computing network," *Egypt. Informatics J.*, vol. 18, no. 1, pp. 11–19, 2017. Article (CrossRef Link)
[7] D. A. Agarwal and S. Jain, “Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment,” *Int. J. Comput. Trends Technol*., vol. 9, no. 7, pp. 344–349, 2014. Article (CrossRef Link)
[8] A. Mehranzadeh and S. Mohsen Hashemi, “A Novel-Scheduling Algorithm for Cloud Computing based on Fuzzy Logic,” *Int. J. Appl. Inf. Syst*., vol. 5, no. 7, pp. 28–31, 2013. Article (CrossRef Link)
[9] Q. Zhang, H. Liang, and Y. Xing, "A Parallel Task Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing Environment," *Int. J. Mach. Learn. Comput.*, vol. 4, no. 5, pp. 437–444, 2014. Article (CrossRef Link)
[10] E. Niazmand, J. Bayrampoor, A. G. Delavar, and A. R. K. Boroujeni, “Jswa An Improved Algorithm For Grid Workflow Scheduling Using Ant Colony Optimization,” *J. Math. Comput. Sci.*, vol. 6, no. 4, pp. 315–331, 2013. Article (CrossRef Link)
[11] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A particle s warm optimization-based heuristic for scheduling workflow applications in cloud computing environments,” in *Proc. of Proc.- Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 400–407, 2010. Article (CrossRef Link)
[12] M. Feng, X. Wang, Y. Zhang, and J. Li, “Multi-objective particle swarm optimization for resource allocation in cloud computing,” in *Proc. of 2012 IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst. IEEE CCIS 2012*, vol. 3, pp. 1161–1165, 2012. Article (CrossRef Link)
[13] J. Wang, F. Li, and A. Chen, "An improved PSO based task scheduling algorithm for a cloud storage system," *Adv. Inf. Sci. Serv. Sci*., vol. 4, no. 18, pp. 465–471, 2012.
[14] E. S. Alkayal, N. R. Jennings, and M. F. Abulkhair, “Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing,” in *Proc. of Conf. Local Comput. Networks, LCN*, pp. 17–24, 2016. Article (CrossRef Link)

[15] N. Dordaie and N. J. Navimipour, “A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments,” *ICT Express*, vol. 4, no. 4, pp. 199–202, 2018. Article (CrossRef Link)

[16] A. Verma and S. Kaushal, “A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling,” *Parallel Comput*., vol. 62, pp. 1–19, 2017. Article (CrossRef Link)

[17] J. Gao, M. Gen, L. Sun, and X. Zhao, “A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems,” *Comput. Ind. Eng.*, vol. 53, no. 1, pp. 149–162, 2007. Article (CrossRef Link)

[18] B. Keshanchi, A. Souri, and N. J. Navimipour, “An improved genetic algorithm for task scheduling in the cloud environments using the priority queues : Formal verification , simulation , and statistical testing,” *J. Syst. Softw.*, vol. 124, pp. 1-21, 2017. Article (CrossRef Link)

[19] H. Y. Shishido, J. C. Estrella, C. F. M. Toledo, and M. S. Arantes, “Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds,” *Comput. Electr. Eng*., vol. 69, pp. 378–394, 2018. Article (CrossRef Link)

[20] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang, “Cost-efficient task scheduling for executing large programs in the cloud,” *Parallel Comput*., vol. 39, no. 4–5, pp. 177–188, 2013. Article (CrossRef Link)

[21] Z. C. Papazachos and H. D. Karatza, “The impact of task service time variability on gang scheduling performance in a two-cluster system,” *Simul. Model. Pract. Theory*, vol. 17, no. 7, pp. 1276–1289, 2009. Article (CrossRef Link)

Devi.K (Devi Karuppiah) received the B.E degree in Computer Science and Engineering from the Manonmaniam Sundharanar University, in 2003, the M.E degree from the Anna University, Chennai, in 2005.She is currently an Assistant Professor in the Department of Computer science engineering at SRM Valliammai Engineering College, Chennai. Her current research interests are Cloud fault tolerance, Scheduling, Deep learning approaches in Cloud and Network security. She is member of the ISTE, CSI and Indian Science Congress.

Dr. Paulraj D received his PhD degree in computer science and engineering from Anna University.He has 20 years of experience including 9 years industrial experience. He has received his B.E (CSE) degree first class from Bangalore University in 1993, M.E (CSE) and Ph.D in Service Oriented Architecture from Anna University respectively in the year 2004 and 2012. During his Ph.D he has proved that Semantic Web Services can be composed using Process Model Ontology instead of Service Profile Ontology. He has published several research publications in refereed International Journals with high impact factor and International Conferences as well. He is the life member of ISTE and a member of IET. He had received IET Men Engineer Award in the year 2012. His research interests include Machine Learning, Data Science, Service Oriented Architecture (SOA),Semantic Web Services, Computer Network and Interface, Cloud and Grid Computing, Big Data Analytics, Block Chain Technologies, Augmented Reality, Virtual Reality.

Dr.Muthusenthil Balasubramanian received the B.E degree in Electronics & Communication Engineering from Madras University, in 1996 Masters Degree from Satyabama University in 2007, Doctorate from Anna University, Chennai 2016. He has been a Research Scientist in Wookyoung Information technology, Daegu, South korea and currently, he works has a Associate Professor at SRM Valliammai Engineering College, Chennai. His interests are in mobile ad-hoc networks, network security, video security, network attacks, privacy preservation, trust evaluation and cloud computing. He is affiliated with scientific publications, he has served as invited reviewer.