Multi-Intellectual Schemes for Dynamic Task Scheduling and Load Balancing in Cloud Environment

Sujatha K¹ and Dr. S. Jagannatha M²

¹. Research Scholar, Department of Computer Applications, M S Ramaiah Institute of Technology, Affiliated to Visvesvaraya Technological University, Belagavi 560 054
². Associate Professor, Department of Computer Applications, M S Ramaiah Institute of Technology, Affiliated to Visvesvaraya Technological University, Belagavi 560 054

Abstract: Load balancing is one of the major issues in cloud computing which allocates a task into VMs with various length and completion time. Hence maintaining load in VMs is a challenging task during task scheduling and allocating. In this paper, we proposed resource aware dynamic task scheduling and secure load balancing using intelligent schemes in cloud environment. The proposed work has following process: (1). Real-Time and Non-Real Time tasks-oriented clustering by using Human Mental Search (HMS) algorithm, in this stage the tasks are clustered into four types such as real-time and short, real-time, and long, non-real time and short and non-real time and long. (2). Energy and Delay aware Clustered Tasks Scheduling, based on the task clustering scheduling is performed and tasks are scheduled in the order of priority which done by using fuzzy VIKOR approach. (3). Security Risks aware VM Clustering, which is done by using Hierarchical Agglomerative Clustering algorithm, this VM clustering process has three entities such as VM’s information collector which collects the information about VMs, second entity is load monitor which manage the whole datacenter and third entity is decision maker which performs task allocation by selecting optimal VM. (4). Co-Resident attacks resisted optimum VM selection which is selected by using continuous-Actor-Critic-Algorithm (C-ACAL). The simulation is conducted by using Cloud Sim that evaluate the performance in terms of response time, latency, resource utilization, throughput, energy consumption and task execution success rate.

Keywords: IoT, cloud computing, Co-resident attacks, VM clustering, Task clustering, human mental search algorithm, hierarchical agglomerative clustering algorithm.

1. INTRODUCTION

Task scheduling is the most fundamental problem in cloud that requires exact match with task requirements and resources. Task scheduling is executed with support of dedicated and reliable cloud resources which currently demands of dynamic scheduling process for online services [1]-[3]. Cloud computing consists of several requests arriving at every second and has the responsibility to respond to all the requests [4]. The cloud should focus on task scheduling and load balancing because, the user
satisfaction lies in the service offered. However, resources in cloud are shorted by CPU, memory and bandwidth required randomly and then it acquired equivalent information to make tasks execution [5]-[8]. When cloud assigns limited resources, then tasks may failure in cloud. Current literature does not meet low latency, energy consumption rate, high resource utilization rate and throughput for datacenters [9]. Certain techniques for task scheduling and load balancing are follows. Meta-heuristic algorithms, evolutionary algorithms, and graph construction methods. These approaches either QoS constraint based or SLA based scheduling such as deadline constraint, budget constraint and others [10].

To overwhelm the issues addressed in existing literature, fog / edge nodes are deployed in a cloud environment. Fog environment provides user requests with computing, storage, and execution services. Tasks scheduling in fog node has become a hotspot area recently [11]-[14]. Task’s characteristics such as QoS and SLA are analyzed to reduce the total execution time and resource usage of individual tasks. Fog nodes can discover dynamic tasks arrived from users which processed without any delay.

In load balancing, VM migration is one of the emerging issues that must be reduced. One of the easy replacements of this issue is that optimum VM selection [15]. Motivated by this, this research has concentrated on the VM selection. In existing VM selection literature, limited criterions were focused such as CPU, memory, task processing cost and so on. For running today’s real-time and online applications, best set of criteria must be demands here. For instance, VM’s type, VM’s current load (number of tasks are running and waiting state) and VM’s history (number of tasks completed executed successfully and non-executed tasks count). Based on that, VM must be matched with arriving task.

1.1 Motivation and Contributions

Task scheduling in Cloud computing is a critical process, since it deals with large amount of data. Further, response time rises for real-time tasks. So that task scheduling must be performed for real-time tasks first and then non-real time tasks are scheduled. Maintenance of this large amount of data is crucial as for improving the utilization of resources in cloud and minimizing response time and completion time. Frequent tasks arrival in cloud data centers rises load imbalance issue. Hence, VM load is wholly excessive. Some of the critical current issues are follows,

- Tradeoff between the make span and resource consumption rate
- Limited parameters ineffective for task scheduling and load balancing

The main objective this research is to minimize execution time and resource usage of each task in fog assisted cloud environment. Further, sub-objectives taken in this research are follows:

- To reduce energy consumption of real-time tasks allocation by resource constrained IoT devices
- To increase the task execution success rate for all types of tasks such as real-time and non-real time
- To maximize load fairness (balanced) that ensures all tasks are matched to the VM’s in a minimum latency.
- To reduce number of VM migrations by selecting the optimum VMs.

The highlights of the proposed work are listed as follows:

- Energy efficient resource allocation is satisfied with significant constraints such as unnecessary wastage elimination, dynamic resource scalability, efficient resource utilization and task and resource location independence. To conserve energy, deep reinforcement learning (C-ACAL) with heuristic algorithms is presented.
For VM allocation and to remove VM eliminating process, Hierarchical Agglomerative Clustering is proposed. It is a clustering analysis method and utilizes it to predict the future utilization tendency. Hence, frequent VM migration can be avoided.

The performance of the proposed work is evaluated with respect to response time, latency, resource utilization, throughput, energy consumption and task execution success rate.

1.2 Paper Organization

The rest of this paper is format as follows: Section II describes the related work and its limitations. Section III explains the overall problem statements which are existed in previous work. Section IV describes proposed system methodology with algorithm and pseudocode. Section V explains the experimental results for the proposed method and the performances are analyzed and compared to existing methods for different kinds of metrics, finally it will be proved that the proposed method achieves superior performance compared to existing methods.

2. LITERATURE SURVEY

In paper [16], fog enabled cloud architecture was designed that analyzes the tasks characteristics and determine resources by a modified firework algorithm. This algorithm works by fireworks radius detection scheme. In fog layer, tasks were clustered using distance between two tasks and arrived tasks were scheduled. Here, task execution location was predicted. Experiments conducted twice for this work means that performance analyze under fog devices and cloud fog infrastructure. The limitation of this research is defined as follows, all tasks assigned with same priority and clustering is not effective since tasks are not optimized. We can get a resource insufficiency issue in this paper.

In paper [17], workload balancing was implemented in IaaS cloud environment. This paper proposed load balancing resource clustering using Bat algorithm which generates optimum number of clusters for resources based on cluster center value. The proposed BAT algorithm gives faster convergence rate. In this work, VM manager was chosen optimum server for task allocation. According to tasks, server selected and deployed in this paper. Each VM instance, CPU and memory rate were computed. Problem of this research is discussed as follows, CPU and memory-based server selection is feasible for small scale environment. It does not suit for heterogeneous requests environment. Weighted round robin-based scheduling considers priority, but suited when size of each task request is same.

In paper [18], a Task Distribution Algorithm (TDA) is proposed to share the workloads to different servers for power reduction of datacenters. In TDA, two heuristic approaches were proposed such as Power Award (PA) Algorithm and Adaptive Harmony Search Algorithm (AHSA) for VM to PM mapping (VM placement) and resource de-allocation. There are several entities used in this paper i.e. request queue, physical machine repository, target VM queue, management node, physical machine manager and virtual machine manager. The research has following problem, Task distribution to VMs affects execution time since de-allocation affects other VM’s performance. QoS factors and SLA requirements must be considered for task scheduling and load balancing.

In paper [19], osmotic computing aided load balancing was proposed which integrates the performance two behaviors such as osmotic and bio-inspired computing. In osmotic algorithm, VM’s deployment and migrations another physical machine was studied. A hybrid artificial bee colony and ant colony optimization algorithms were proposed in this paper for load balancing. The proposed hybrid bio-inspired algorithms better when compared to existing bio-inspired algorithms in terms of energy consumption, number of virtual machine migrations and QoS achievements. Limitation: Number of VM
migrations high when large number of real-time requests arrived from one region. Hybrid optimization algorithms consume more processing time. Analyzing all task characteristics under cloud that increase overhead, this leads to increases completion time for longer jobs.

In paper [20], job scheduling was performed in this paper to increases cloud resources utilization rate. In the first step, tasks execution time was computed and then queue model was developed. Here, all incoming tasks were transformed into hypergraph using Directed Acyclic Graph (DAG). In this graph, Dijkstra Shortest Path Scheduling Algorithm was proposed to determine the optimal task scheduling which was implemented over the hypergraph. Load balancing problem was addressed using three metrics as load, resource utilization rate and jobs complexity level. In application use case, a real-time live video watching case was considered in implementation. This research has following problems, Resource blocking often possible due to presence of co-resident attacks. Graph construction especially by DAG shows high complexity.

In paper [21], genetic and particle swarm optimization algorithms were integrated for VM load balancing. This paper considers both QoS and SLA constraints to decrease resource wastage in VM allocation. Firstly, genetic algorithm was running which initialize population, performed crossover and mutation operations. Secondly, PSO was used to compute the fitness values for incoming tasks. Here, genetic algorithm was addressed initialization problem and PSO addresses the premature convergence issue. Both issues are main problems that degrade the cloud server’s performance. The drawback of genetic algorithm is that it requires high deployment and implementation costs and requires higher number of iterations. However, PSO algorithm has large size of memory requirement issue, if it integrates to PSO, then the amount of memory requirements is twice over. As a result, handling CPU intensive tasks are much harder.

In paper [22], three load balancing problems were formulated such as energy consumption of mobile devices, time consumption of mobile devices and cloudlets load balancing. Here, both (homogeneous and heterogeneous) kinds of cloudlets were focused. For efficient offloading, this paper proposes a modified non-dominated sorting genetic algorithm III (NSGA-III). Experiment results conclude that the proposed NSGA-III algorithm shows better performance since its offload applications to cloudlets. However, Edge nodes can handle tasks and provide results with the specified requirements. But real-time applications need to wait for non-real-time application to complete execution.

In paper [23], energy efficient resource ranking based VM selection scheme was proposed. The proposed scheme consists of Task Classification, Comprehensive Resource Balance Ranking, Processing Element Cost and Resource Utilization Square Model for Migration. In this scheme, tasks were assigned to VMs by optimum manner (task type matched with VM’s type), which improves the resource utilization rate of VM and minimizes the number of VM migrations. Further, reduces the VM’s resource wastage. However, VM selection problem was addressed by consideration of Resource Capacity, Resource Utilization and Energy Consumption of VMs on the respective host. In this case, VM’s resources can be spoiled by co-resident attackers. This must be mitigated for high resource wastage issue.

In paper [24], hybrid model-based load balancing is proposed by using file type formatting in cloud environment. This proposed system used hybrid algorithm which includes SVM and ant colony optimization. SVM classifies the inputs in terms of audio, video, text, and images. Load balancing is done by using ACO. The result shows that the proposed system achieves better performance with scalability and robustness. However, this research used ACO for load balancing, but it considers
sequence of random decision which leads to high latency and SVM takes much time to classify the real
time data which reduces performance of the process.In paper [25],task scheduling is proposed by using
PSO algorithm for cloud environment. To start PSO search they used two algorithms such as longest job
to faster processor and minimum amount of completion time which increase the performance of
convergence and speed. The result shows that the proposed system achieves higher performance for task
scheduling in terms of convergence and load balance. However, PSO can easily fall into the local optimal.

In paper [26] performed task scheduling by using hybrid electro search with genetic algorithm for cloud
environment (HESGA). This research addressed the problem of make span. The proposed system
architecture has three entities such as cloud brokers which connect cloud users and task scheduler;second one is task scheduler which schedules task with optimal resources and third one is
datacenters that provides resource to the user. Then HESGA algorithm performs scheduling and stored
the scheduled result into VM. The result shows that the proposed system reduced make span, response
time and cost compared to other scheduling algorithms. However, genetic algorithm takes much time to
complete the task which increases latency. In paper [27] performs task scheduling using hybrid
algorithm which includes fuzzy TOPSIS and PSO in cloud computing. This research addresses the
problem of high energy consumption and cost. PSO is used to select available task and number of
VMs. The proposed fuzzy TOPSIS algorithm solves the problem of multi objective task scheduling which
includes execution time, cost, energy consumption and weighted sum. The result shows that the
proposed system achieves better SLA and QoS compared to other algorithms. However, PSO can easily
fall into local optimal which reduce the performance of the process.

In paper [28], proposed stochastic load balancing to avoid VM migration overload. Distance between
source PM and destination PM is considered for VM migration. The proposed system has many
components such as profiling of resource demand, hotspot detection and hotspot migration. Heuristic
algorithm is used for hotspot migration. Load balancer present executes in a central server. Each PM
monitor monitored the usage of CPU memory and network which is periodical
ly updated to the resource
usage collector. Migration information is collected from VM migration controller. The result shows that
the proposed system achieves superior performance in terms of migration cost and SLA violation
compared to existing methods.

3. PROBLEM STATEMENT

Load balancing is a crucial problem in cloud computing environment. Tasks arriving requests are
massive currently due to the usage of smart Internet of Things (IoT). IoT device requests are delay
sensitive and should be running concurrent without any execution failure. Therefore, efficient resource
allocation with dynamic task scheduling is an elemental task in cloud datacenter.

- Unfitting Resource Allocation: Cloud resources (CPU, Memory and Bandwidth) must be
distributed as per the Quality of Service (QoS) and Service Level Agreement (SLA) constraints
for any task which minimizes energy and load intake by VM’s.
- Lack of Edge / Fog Devices: Cloud faces with high response time and latency when processing
real-time tasks. Further, handling task characteristics processed in cloud layer, which increases
communication overhead. Edge / Fog based cloud environment reduces response time, latency,
and resource usage. Hence, it’s preferable in workload balancing via accurate resource provisioning in Cloud-IoT Environment.

- Co-Resident Attacks: Further, VMs are at high risks by reason of unrelated requests from IoT devices that target specific VM or multiple VMs on the same physical machine (co-resident attacks). This attack must be mitigated in VM selection.

To address these issues, cloud environment should be running through the dynamic and secure load balancing. In paper [29], addresses large energy consumption of datacenter through dynamic tasks scheduling. The major problem of this work as follows:

- This work does not suit for large scale environments since running all real-time and non-real-time tasks in global queue is a complex job. A centralized task dispatcher is manageable only when small number of tasks arriving. Further, tasks are scheduled based on first come first serve basis. This type of scheduling increases the waiting time of the upcoming tasks.
- Resources are not properly allocated to individual tasks. Here, time-based metrics does not determine the VM’s current traffic rate.

In [30], hybrid optimization algorithm was proposed for workload balancing. The problem of this work is defined as follows:

- This work leads to poor load balancing due to lack of optimum match of task to the VM. Without balancing of workload, tasks are not properly executed. Prior knowledge of resource pattern for individual task is required for modeling.
- This work does not suit for real-time tasks. QoS and SLA of a task can be violated for a longer time.
- Algorithms running for hybrid algorithm increases delay. PSO fall into local optimum issue when large number of requests handling and low convergence rate.

In [31], bag of tasks execution was implemented in fog assisted cloud environment using genetic algorithm. The problem of this research is discussed as follows:

- Tasks are not executed properly due to the consideration of limited criteria (task execution time (task length and CPU rate) and make span. Resource limitation problem still unsolved in this paper. Availability of resources may decrease over a time due to Co-resident attackers. Since, VM leads to high resource wastage.
- Genetic algorithm is more complex which is easily fall into Premature Convergence Issue. Cross over and mutation operators need to be defining accurately that affects the solutions. Further, high number of design considerations is required.

In [32], addresses the load balancing problem in cloud environment in which workload distributes to all VMs in a dynamic manner. The main problems of this work are follows:
• In Q-learning algorithm, workload uncertainty problem is addressed. Q-learning make use of discrete and finite action space by means of some fixed and a priori determined fairness steps to parameterize the generalized proportional fair in each state.

• VMs are at high risks by reason of co-resident attacks. Hence, VM’s risk level must be computed for optimum VM selection. Higher SLA violations and increase in execution time because of consideration to minimum number of parameters for optimal VM selection.

Table 1. Existing System Drawbacks

<table>
<thead>
<tr>
<th>Technique</th>
<th>Topic</th>
<th>Drawback</th>
</tr>
</thead>
</table>
| DTS [29]        | large energy consumption of datacenter through dynamic tasks scheduling | • High latency
                 |                                                                      | • Poor VM allocation                  |
| MOPSO [30]      | Workload balancing by using hybrid optimization algorithm             | • Poor load balancing
                 |                                                                      | • Low convergence                     |
| OTS [31]        | Bag-of-task executed in cloud environment by using genetic algorithm  | • High resource wastage
                 |                                                                      | • Low convergence                     |
| Hybrid-MHA [32] | Workloads are dynamically distributed to all VMs                      | • Workload uncertainty
                 |                                                                      | • Higher SLA violation                 |

Table 1 explains the technique, topic and drawback of the existing systems.

4. PROPOSED METHODOLOGY

In this research, resource aware dynamic task scheduling and secure load balancing is focused by multiple intellectual approaches. The proposed work aims to balance the load on the individual VM and
multiple VMs on the same physical machine and schedule the arriving tasks. We have three different tiers in our system. They are the IoT device tier, Fog Tier and Cloud Tier. Each of these tiers will play an important role in the process of achieving the objective of the proposed system. The proposed three tier architecture for dynamic task scheduling and load balancing has certain entities as IoT Devices / Users, Edge Gateway, Fog Node, Secure Cloud Broker and Datacenter. The proposed system has four consecutive phases such as,

1. Real-Time and Non-Real Time Tasks Oriented Clustering
2. Energy and Delay aware Clustered Tasks Scheduling
3. Security Risks aware VM Clustering
4. Co-Resident Attacks Resisted Optimum VM Selection

A. Real-Time and Non-Real Time Tasks Oriented Clustering

Each IoT device task is forwarded from gateway to fog node. Fog node consists of Cluster Manager and Scheduler. The cluster manager will receive tasks and clustered into four types such as (i). Real-time and Short, (ii). Real-time and Long, (iii). Non-Real time and Short, and (iv). Non-Real time and Long. Cluster manager clusters tasks by using Human Mental Search Algorithm, which is an optimization-based clustering algorithm that classifies each task and put into their corresponding cluster. The procedure of the algorithm is shown in pseudo code. For clustering, we have considered two main policies such as QoS and SLA in which several constraints are considered as Task Size, Task Type, Task Deadline, Task Arrival Time, Device Energy Level, CPU required, Memory required, and I/O bandwidth Required.

**Pseudocode: Human Mental Search**

```
1. Begin
2. P=Population initialization of P_pop bids
3. Calculate bids cost
4. b*= finding best bid from initial population
5. for i from 1 to P_pop do
6. α= random integer number between lower bound and upper bound
7. end for
8. for I from 1 to maxI do//mental search
9. for i from 1 to P_pop do
10. μ= random integer number between mx,my
11. end for
12. for i from 1 to P_pop do
13. for j from 1 to μ do
14. A = \left( \frac{2}{\text{max}} \right) * \frac{s}{T_t} * \frac{s}{t_*} * (b^* - b^j)
15. ns_{j} = P_{i} + A;
16. End for
17. q= finding ns with lower cost
```

http://www.webology.org
18. if cost(q)<cost(P_i)
   19. P_i = q
   20. end if
   21. end for

//clustering

22. cluster P_pop bids into N clusters
23. calculate mean cost value for every cluster
24. choose cluster with lowest mean cost value as the Winner cluster

25. Winner= choose the best bid in the Winner cluster
   26. for i from 1 to P_pop do
   27. for t from 1 to P_var do
   28. P_n^i = P_n^i + U * (x * Winner_n - P_n^i)
   29. end for
   30. end for
   31. for i from 1 to P_pop do
   32. a= random integer number between lower bound and upper bound
   33. end for
   34. b^+ = finding best bid in the current bids
   35. If (cost(b^+)<cost(b^)) do
   36. b^+ = b^+
   37. end if
   38. end for
   39. end begin
Figure 1. Proposed System Architecture

Figure 1 represents the proposed system architecture which has three layers such as IoT layers, fog layers and cloud layers. IoT layer has many IoT devices. Gateway forwards the IoT task to the fog node. Fog consists of two entities such as cluster manager and scheduler which are performed task clustering. And third layer is cloud layer which has cloud broker that consists of three entities such as VM’s information collector, load monitor and decision maker for balancing cloud load. The overall system focused to...
balance the load on the individual VM and multiple VMs on the same physical machine and schedule the arriving tasks.

Description: First initialize the population of bid and next calculate the cost of the bid. Generate random integer between upper bound and lower bound for current and maximum iterations. Generate a random integer for minimum mental process and maximum mental process. Then finding lowest cost bit for clustering. To choose lowest cost valued for finding winner cluster. Here also generate a random integer between lower bound and upper bound for finding best bid from the current bid. In our process four clusters are formed such as real time and short, real time and long, non-real time and short and non–real time and long.

B. Energy and Delay aware Clustered Tasks Scheduling

Cluster manager directs all the tasks to the scheduler. The scheduler finds the cluster priority and executes by QoS and SLA constraints. The scheduler will perform the proposed algorithm and achieve the objective. Based on clustered result, four individual queues (Q1, Q2, Q3 and Q4) are created by assigning priority for tasks. Each task is prioritized by Task Waiting Time and Energy level. For scheduling tasks within the cluster, Fuzzy VIKOR Approach is proposed. It is a ranking method which is used to rank the tasks based on its priority.

Step 1: Fuzzy VIKOR approach is defined as matrix format

There isn priority which can be defined as \( P_i (i = 1, 2 \ldots m) \) evaluated based on the queue selected that is \( Q_j (j = 1, 2 \ldots n) \). Each criterion has five ranks \( R = 1,2,3,4,5 \). Decision matrix is defined as follows, \( \text{Mat} = \{ \text{mat}_{i,j,g} | i = 1,2, \ldots m, j = 1,2, \ldots n, g = 1,2, \ldots 5 \} \)

\[
\text{Mat} = \begin{bmatrix}
P_1 & Q_1 & Q_2 & \ldots & Q_n \\
P_2 & m_{11} & m_{12} & \ldots & m_{1n} \\
P_3 & m_{21} & m_{22} & \ldots & m_{2n} \\
P_n & m_{x1} & m_{x2} & \ldots & m_{xn}
\end{bmatrix}
\]  

\( R = [r_1, r_2, \ldots, r_n] \)

Where, \( P_1, P_2, \ldots, P_n \) represents priorities, \( Q_1, Q_2, \ldots, Q_n \) are the evaluating criteria’s. \( R \) represents the ranking of the criteria’s.

Step 2: Fuzzy decision matrix construction

The fuzzy rating \( \text{mat}_{i,j} \) is the rating of priority \( P_i \) with respect to the trapezoidal fuzzy numbers is modification from the process of arithmetic rank average and calculated using the given equation

\[
\text{Mat} = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{r=1}^{5} \text{mat}_{i,j,g} \otimes \text{TZFN} = [\text{Mat}_{i,j}]_{m \times n}
\]

\[
[\text{Mat}_{i,j}]_{m \times n} = \begin{bmatrix}
\tilde{m}_{11} & \tilde{m}_{12} & \ldots & \tilde{m}_{1n} \\
\tilde{m}_{21} & \tilde{m}_{22} & \ldots & \tilde{m}_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{m}_{x1} & \tilde{m}_{x2} & \ldots & \tilde{m}_{xn}
\end{bmatrix}
\]

Where, \( \tilde{R} = [\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n] \). Trapezoidal fuzzy number is defined as follows,

\[
\text{Mat}_{i,j} = (\tilde{q}_{i,j}, \tilde{r}_{i,j}, \tilde{s}_{i,j}, \tilde{t}_{i,j})
\]
Step 3: Measure the fuzzy importance rank of criteria

The fuzzy rank values for every criterion is defined as the importance of each criterion. Relative value is directly proportional to the number of queues present for getting priorities. The number of queues gets the priority \( j \), and then the fuzzy importance of priority is given as,

\[
\tilde{R}_j = \frac{\tilde{\sigma}_j}{\sum_{i=1}^{n} \tilde{\sigma}_i}
\]

(6)

\( \tilde{\sigma}_j \) represent standard deviation for criterion \( Q_n \). Standard deviation \( \tilde{\sigma}_j \) is given as follows,

\[
\tilde{\sigma}_j = \frac{1}{S} \sum_{x=1}^{S} (\tilde{m}_{xn} - \tilde{m}_n)^2
\]

(7)

\( \tilde{m}_n = \frac{1}{S} \sum_{x=1}^{S} \tilde{x}_{mn} \), \( 0 \leq \tilde{R}_j \leq 1 \) and \( S = \) Total amount of priorities.

Step 4: Determine fuzzy best value (\( \tilde{m}_j \)) and fuzzy worst value (\( \tilde{m}_j \))

\[
\tilde{m}_j = \max_i \tilde{m}_{ij}
\]

(8)

\[
\tilde{m}_j = \min_i \tilde{m}_{ij}
\]

(9)

Step 5: Computation for normalized fuzzy decision matrix

In this step is used to ensure the criterion value between zero to one. VIKOR method used linear normalization to stabilize which is defined as follows,

\[
\tilde{A}_i = \sum_{j=1}^{n} \left[ \frac{\tilde{m}_{ij} - \tilde{m}_{ij}}{\tilde{m}_j - \tilde{m}_j} \right]
\]

(10)

\[
\tilde{B}_i = \max_j \left[ \tilde{R}_j \left( \frac{\tilde{m}_{ij} - \tilde{m}_{ij}}{\tilde{m}_j - \tilde{m}_j} \right) \right]
\]

(11)

Step 6: Calculate index VIKOR \( \tilde{I}_i \)

\[
\tilde{I}_i = u \left( \frac{\tilde{A}_i - \tilde{A}^-}{\tilde{A}^+ - \tilde{A}^-} \right) + (1 - u) \left( \frac{\tilde{B}_i - \tilde{B}^-}{\tilde{B}^+ - \tilde{B}^-} \right)
\]

(12)

\[
\tilde{A}^+ = \max_i \tilde{A}_i, \tilde{A}^- = \min_i \tilde{A}_i
\]

(13)

\[
\tilde{B}^+ = \max_i \tilde{B}_i, \tilde{B}^- = \min_i \tilde{B}_i
\]

(14)

\( u \) is determined the weight of the priority and VIKOR index value is denoted as \( u=0.5 \).

Step 7: perform sorting operation

Sorting the value \( \tilde{A}, \tilde{B} \) and \( \tilde{I} \) in descending order. The higher priority in order of \( \tilde{I} \) is the maximum possible value which is based on rank that was done and denote as \( P^{(1)} \), second largest priority is denoted as \( P^{(2)} \) and the smallest value is denote as \( P^{(n)} \).

Step 8: The priority \( P^{(1)} \) is the higher priority with the maximum value of \( \tilde{I} \) which has two conditions for providing compromise solution.

Condition 1: The difference index VIKOR\( \tilde{I} \) between \( P^{(1)} \) and \( P^{(2)} \) must be greater than or equal to DI.
\[ \hat{I}_{(p^{(2)})} - \hat{I}_{(p^{(1)})} \geq DI, DI = \frac{1}{s-1} \]

Condition 2: Priority \( p^{(1)} \) must be present in \( \bar{A} \) or \( \bar{B} \)

If condition 1 is not satisfied then the priorities \( p^{(1)}, p^{(2)}, \ldots, p^{(n)} \) considered together with its best \( p^{(n)} \) determined by the relationship as,

\[ \hat{I}_{(p^{n})} - \hat{I}_{(p^{1})} < D \]

If condition 2 is not satisfied, then both priorities \( p^{(1)} \) and \( p^{(2)} \) are determined as the best priorities.

C. Security Risks Aware VM Clustering

Due to attackers who target a single or multiple VM, this research has considered VM’s risk level for clustering. Clustering similar VM’s reduces task execution time and avoid latency in resource allocation. Scheduled tasks forwarded to secure cloud broker for execution via appropriate resource matching. This entity consists of three elements as follows

VM’s Information Collector: VM’s risk level is identified via continuous monitoring of VM’s task execution performance. When any VM’s affected or are at high risk then it’s isolated from cluster. Since high risks VM does not meets the specified execution time.

Load Monitor: This entity has a primary role to manage the whole datacenter. Here, cluster management and cluster purity validation have implemented. A Hierarchical Agglomerative Clustering algorithm is used for VM’s cluster formation for that purpose four criterions are used such as VM Running Type, Latency, Resource Rate, and Task Execution Success Rate.

Decision Maker: The purpose of decision maker is to perform Resource Allocation process through optimum VM selection. It makes the decision based on the cluster results. If any task needs VM then decision maker sends a request to the load monitor for updating current information of the VMS. The load monitor updates the current idle state of the VM to the decision maker for allocating the task to the optimum VM which improves the performance and reduces the latency.

### Pseudo code: Hierarchical Agglomerative Clustering

1. HAC(D₁, ..., Dₙ)
2. for m ← 1 to n
3. do for i ← 1 to n
4. do Cluster[x][i] ← similarity (Dᵢ, Dᵢ)
5. M[n] ← 1 (Active clusters)
6. S[ ] (Sequence of merges are assembled as cluster)
7. for k ← 1 to n-1
8. do(i, t) ← Arg max((i, t) \neq t'M[|i|]=1 \land M[|t|]=1) Cluster[i][t]
9. S.APPEND((I, t)) (Store Merge)
10. For j ← 1 to n
11. Do Cluster[i][j] ← similarity (i, t, j)
12. Cluster[j][i] ← similarity (i, t, j)
13. \( M[t] \leftarrow 0 \) (Deactivate Cluster)
14. return \( S \)

Description: First initialize the data points and find the similarity between the data points to form the cluster. If the similarity is same as those two data points that are connected to each other which is known as active cluster and that are assemble as sequence manner. Then all the sub clusters are merging as a single cluster. Until meets zero this process will be continuing.

Figure 2 represents the process of security risks aware VM clustering. In fog layer the tasks are scheduled by scheduler, that tasks are send to secure cloud broker for selecting optimum VM. The secure cloud broker has three entities such as VM’s information collector, load monitor and decision maker. Information collector collects the information about VM’s; if VM has high risk then it will be isolated from the cluster. Load monitor monitored the entire datacenter and the VMs are clustered by using hierarchical agglomerative clustering algorithm. Finally, decision maker takes decision to select optimum VM.

D. Co-Resident Attacks Resisted Optimum VM Selection

Various VMs used same physical resources such as CPU, memory and storage devices are known as co-resident VMs which introduced a new type of attack called as co-resident attack. In this type of attack malicious nodes create various types of channels between malicious VM and targeted VM to utilize the resources. Based on the task’s requirements, resource is estimated by using Continuous-Actor Critic Algorithm (C-ACAL). The proposed C-ACAL has superior performance than Q-learning, improved Q-learning and SARSA. For VM selection, Task Completion Time, VM Type, VM Configuration, VM Current Load and VM’s Risk Level are used. These metrics finds the required resources for tasks execution.

Pseudocode: Continuous Actor Critic Algorithm

1629 http://www.webology.org
INPUT: states $s \in ST$, actions $a \in AC(st)$

Initialize $\alpha, \gamma$

OUTPUT: policy $\pi(st, ac)$ responsible for selecting action $ac$ in state $st$.

1. $\forall st \in ST, ac \in AC(st)$ do
2. $q(st, ac) \leftarrow 0$
3. $\pi(st, ac) \leftarrow \frac{e^{q(st, ac)}}{\sum_{x=1}^{|AC(st)|} e^{q(st,x)}}$
4. end
5. while True do
6. Initialize $st$
7. for($i = 0; i < t_m; i = i + 1$) do
8. Choose $ac$ from $st$ use $\pi(st, ac)$
9. Action $ac$, observe $re$, $st'$
10. $\mu = re + \gamma U(st') - U(st)$
11. $U(st) \leftarrow U(st) + a(re + \gamma U(st') - U(st))$
12. $q(st, ac) \leftarrow q(st, ac) + \beta \mu$
13. $\pi(st, ac) \leftarrow \frac{e^{q(st, ac)}}{\sum_{x=1}^{|AC(st)|} e^{q(st,x)}}$
14. $st \leftarrow st'$
15. end
16. end

Description: State and action is considered as input and policy is considered as output. Policy has the responsibility for selecting action in state. If policy is true, then initialize action and state. Next choose the action from the state by using policy. Based on the existing state reward the current action is chose and policy is updated. This process is done continuously.

5. EXPERIMENTAL RESULTS

This section describes the simulation results, and it has three sub sections such as simulation setup, comparison study and research summary for the proposed work than the previous work.

A. Simulation Setup

The simulation of the proposed system is explained in this section, which is implemented using CloudSim. It is useful for modeling and simulation of cloud entities and services. It was introduced by cloud computing and distributed systems laboratory at university of Melbourne. Initially, we create a CloudSim environment with n number of IoT Devices / Users, Edge Gateway, Fog Node, Secure Cloud Broker, and the Datacenter. The scope of CloudSim is to stimulate environment consist of large amount of IoT devices. Table 2 depicts software configuration. Table 3 depicts CloudSim configuration which includes application module, fog node, proxy server, cloud server and IoT devices.

Table 2. System Configurations

<table>
<thead>
<tr>
<th>CPU</th>
<th>Pentium (R) Dual-Core</th>
</tr>
</thead>
</table>

http://www.webology.org
| System configuration | | CPU E5700 @ 3 GHz |
|----------------------|------------------|
| Type of topology     | fully connected  |
| memory               | 4 GB             |
| Operating System     | Windows 7 ultimate [x86]-32 bit processor |
| language             | java             |
| development kit      | JDK 1.8          |
| IDE                  | Netbeans 8.0     |

Table 3. CloudSim Configurations

<table>
<thead>
<tr>
<th>CloudSim configuration</th>
<th>Application Module</th>
<th>Fog node</th>
<th>Proxy server</th>
<th>Cloud Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPS</td>
<td>1000</td>
<td>Memory</td>
<td>10 MB</td>
</tr>
<tr>
<td></td>
<td>Storage capacity</td>
<td>11 TB</td>
<td>MIPS</td>
<td>2800</td>
</tr>
<tr>
<td></td>
<td>Resource cost</td>
<td>3.0</td>
<td>Memory cost</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Delay</td>
<td>100ms (between proxy server-cloud)</td>
<td>MIPS</td>
<td>2800</td>
</tr>
<tr>
<td></td>
<td>Delay</td>
<td>1000 ms and longer (between devices)</td>
<td>MIPS</td>
<td>44800</td>
</tr>
<tr>
<td>IoT devices</td>
<td>Delay</td>
<td>1ms (between IoT sensors to fog devices)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-----------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td>1500</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM(GB)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Result of Cloud Analyst Simulation

Figure 3 represent the cloud analyst which the fog nodes and IoT nodes in the cloud environment are depicted. In which cloud environment are distributed application that deploys in various geographical locations because it is accessed by users around the world. CloudSim used to allow control and repeatability of experiments. Cloud anlayst also shows that the task transmission over the internet.

Figure 4. Process of Human Mental Search algorithm

Figure 4 represent the process of human mental search algorithm; first initialize the parameters of HMS such as task size, task type, task deadline, task arrival time, device energy level, CPU required, memory required, and I/O bandwidth required. HMS is used to cluster the task into four types such as real time and short, real time and long, non-real time and short, non-real time and long which is done by cluster manager.

B. Comparison Study

This section describes the comparison of the proposed and existing system in terms of Response time vs. Number of Tasks, Latency vs. Number of Tasks and VMs, Resource utilization vs. Number of Tasks and
VMs, Throughput vs. Number of Tasks and VMs, and Energy Consumption vs. Number of Tasks and VMs and Task Execution Success Rate vs. Number of Tasks and VM.

a. Impact of response time

Response time denotes total amount of time taken by VM for responding given request which is evaluated with respect to the number of tasks. Response time also determines the difference between the completion time and initiation time.

\[
\tau = \zeta - \eta
\]

(17)

Where, \( \tau \) is a response time and \( \zeta \) represent as completion time and \( \eta \) represent initiation time. Figure 5 represent the comparison of existing and proposed system response time with respect to number of tasks. From the figure the response time increases exponentially with increase in number of tasks, but this proposed system the tasks are clustered and scheduled based on priority which reduces response time and increases the efficiency of this system. The comparison result shows that the proposed system achieves low response time compared to existing system such as MOPSO and DTS.

![Figure 5: Response Time vs. No. of Tasks](image1)

b. Impact of latency

The time it takes to reach the task from source to destination is known as latency which reduces the accuracy and performance of the proposed system. If scheduling or allocation is delayed, then QoS value will be degraded.

![Figure 6: Latency vs. No. of Tasks](image2)
Figure 6 represent the comparison of three methods including proposed method with respect to the number of tasks. The figure shows that the proposed system achieves low latency compared to existing system, because our proposed system performs tasks clustering and priority-based scheduling which reduces latency and increase the accuracy of the system. But existing system does not perform clustering hence it takes much time to complete task scheduling and allocation which increase latency and reduce QoS values. Similarly figure 7 represent the comparison of latency with respect to the number of VMs. In our proposed system VMs also clustered which reduce allocation time hence proposed system achieves low latency compared to existing systems like MOPSO and DTS.

c. Impact of Resource Utilization

The resource that it utilizes by the task to complete the process is known as resource utilization which improves the performance of the proposed system and it saves the energy of the resources in data center.

\[ RU = \hat{\rho} - \hat{\omega} \]  

(18)

Where, RU is a resource utilization and \( \hat{\rho} \) represent total resource and \( \hat{\omega} \) is remaining resource. Therefore, the resource utilization is states that the difference between total resource and remaining resource is known as resource utilization.

Figure 8. Resource Utilization vs. No. of Tasks
Figure 9. Resource Utilization vs. No. of VMs

Figure 8 represents the comparison of existing and proposed system resource utilization with respect to the number of tasks. The comparison result shows that the proposed system archives high resource utilization compared to existing systems like MOPSO and DTS. In our proposed system tasks are clustered initially and the clustered tasks are scheduled based on priority which utilizes high resource compared to existing systems. Similarly figure 9 represents the comparison of resource utilization with respect to number of VMs. In this research VMs are also clustered for selecting optimum VM which utilize high resources compared to existing systems.

d. Impact of Throughput

Throughput states that the rate of successful reception of tasks by the destination node which is one of the significant parameters to determine the accuracy of the process.

Figure 10. Throughput vs. No of Tasks
Figure 11. Throughput vs. No of VMs

Figure 10 represent the comparison of existing and proposed system throughput with respect to number of tasks and number of VMs. Figure 11 shows that the proposed system achieves high throughput compared to existing systems like MOPSO and DTS. Tasks are allocated in an optimum VM without any fault which increases throughput. And this research performs against co-resident attack, but existing system does not perform co-resident attack which reduce throughput and accuracy thus the proposed system has better throughput.

e. Impact of Energy Consumption

Total amount of energy consumed by the task to complete the process is known as energy consumption which is one of the QoS based metric which increase the performance of the proposed system.

\[ EC = I - \sigma \]  

Where \( EC \) represent energy consumption and \( I \) is an initial energy and \( \sigma \) denotes residual energy. Therefore, energy consumption determines the difference between initial energy and residual energy.

![Energy Consumption vs. No. of Tasks](image1)

Figure 12. Energy Consumption vs. No. of Tasks

![Energy Consumption vs. No. of VMs](image2)

Figure 13. Energy Consumption vs. No. of VMs

Figure 12 represent the comparison of existing and proposed system energy consumption with respect to number of tasks. The comparison result shows that the proposed system consumes low energy compared to existing systems like MOPSO and DTS. The proposed system tasks are clustered initially which reduces overhead hence it utilize low energy. Similarly, figure 13 represent the comparison of energy consumption with respect to number of VMS. Here continuous actor critic algorithm is used to select...
optimum VM which learns environment and continuously changes the rewards which reduces overhead and energy consumption.

Thus, the proposed system achieves low energy consumption.

f. Impact of Task Execution Success Rate

Task execution success rate states that the numbers of tasks are successfully received by destination node from the source node. It is computed by total number of tasks are sent and total number of tasks are received successfully.

\[ E = \frac{i}{o} \]  

(20)

Where, \( E \) represent task execution success rate and \( i \) is number of tasks sent and \( o \) represent number of tasks received.

Figure 14 represent the comparison of existing and proposed system task execution success rate with respect to number of tasks. Figure shows that the proposed system achieves high success rate compared to existing system. The proposed system performs security risks aware VM clustering and co-resident attack resisted optimum VM selection which increases the security hence it improves task execution success rate. Similarly figure 15 represent the comparison of existing and proposed system task execution success rate with respect to number of VMs. The result shows that the proposed system achieves high task execution success rate compared to existing system because of performing security risk aware clustering and co-resident attack resisted optimum VM selection which is done by continuous actor critic reinforcement algorithm it learns environment and continuously update the rewards which increase the task execution success rate.

![Figure 14 Task Execution Success Rate vs. No. of Tasks](http://www.webology.org)
C. Research Summary

In this section we explain how the proposed system has high performance compared to existing system. Figure 5-15 represents the better performance of the proposed system in terms of response time, latency, resource utilization, throughput, energy consumption and task execution success rate. The major motives for results are discussed as below,

- We perform real time and non-real time tasks-oriented clustering by using human mental search algorithm. This method is used to reduce overhead and it is used task-based parameters and resource-based parameters which reduces latency and energy consumption.
- Tasks are scheduled based on priority and each task is prioritized by task waiting time and energy level which is done by using fuzzy VIKOR approach that reduces energy consumption and latency.
- This research considers VMs risk level for clustering which is monitor by secure cloud broker. VMs are cluster by using hierarchical agglomerative clustering algorithm hence it reduces task execution time and increase task execution success rate.
- Optimal VM selection is done by continuous actor critic algorithm which learns environment and update reward continuously hence it increases task execution success rate and throughput.

Table 4 explains the numerical analysis of the proposed system that compared to the existing systems like MOPSO and DTS.

Table 4. Comparison of Proposed and Existing Approaches

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Proposed vs. Existing Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MOPSO</td>
</tr>
<tr>
<td>Latency(s)</td>
<td>107.5</td>
</tr>
<tr>
<td># of Task</td>
<td></td>
</tr>
<tr>
<td># of VM</td>
<td>70</td>
</tr>
</tbody>
</table>

1638 http://www.webology.org
<table>
<thead>
<tr>
<th></th>
<th># of Task</th>
<th>210</th>
<th>130</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response time (ms)</td>
<td># of Task</td>
<td>32.5</td>
<td>42.5</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td># of VM</td>
<td>35</td>
<td>45</td>
<td>70</td>
</tr>
<tr>
<td>Resource Utilization (%)</td>
<td># of Task</td>
<td>37.5</td>
<td>42.5</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td># of VM</td>
<td>70</td>
<td>90</td>
<td>133.3</td>
</tr>
<tr>
<td>Throughput (kbps)</td>
<td># of Task</td>
<td>32.5</td>
<td>27.5</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td># of VM</td>
<td>40</td>
<td>33.3</td>
<td>23.3</td>
</tr>
<tr>
<td>Energy consumption (kwh)</td>
<td># of Task</td>
<td>30</td>
<td>41.2</td>
<td>52.5</td>
</tr>
<tr>
<td></td>
<td># of VM</td>
<td>36.6</td>
<td>43.3</td>
<td>53.3</td>
</tr>
</tbody>
</table>

6. CONCLUSION

In this paper, resource ware dynamic task scheduling and secure load balancing is proposed in IoT -fog-cloud environment. To reduce overhead, we proposed real time and non-real time tasks-oriented clustering, all the tasks are clustered by using human mental search algorithm which considered resource-based parameters and task-based parameters for clustering. Next tasks are scheduled by using fuzzy VIKOR approach, each task is scheduled based on priority that considers waiting time and energy level for given priority. After scheduling the tasks are allocated in an optimal VM. For this purpose, we perform security risks aware VM clustering to reduce execution time and latency. For secure load balancing we employ secure cloud broker that consider three entities such as VMs information collector, load monitor and decision maker. VMs information collector monitors the risk level of the VM, if any VM is affected then it is isolated from cluster. In load monitor VMs are cluster by using hierarchical agglomerative clustering algorithm for managing the whole datacenter, based on the result of load monitor decisions are made to select optimum VM. Continuous actor critic algorithm is used to resist co-resident attack in optimum VM selection. Finally, the simulation is executed to improve the performance in terms of response time, latency, resource utilization, throughput, energy consumption and task execution success rate.
REFERENCES


Ms. Sujatha K, Research Scholar, Department of Computer Applications, M S Ramaiah Institute of Technology, Bangalore did her M.E Computer Science in Anna University Affiliation during 2016 and M.C.A in Madras University Affiliation during 2004. Along with 2 years of industry experience, she has 14 years of teaching experience in VTU affiliated Colleges and Jain University. She has published research articles in various National / International Conferences and Journals. She has conducted faculty development programme, bridge courses and workshops on Mobile Application and NoSQL databases domain and participated/presented in National and International Level Conferences.

Dr. S Jagannatha M, Associate Professor, Department of Computer Applications, M S Ramaiah Institute of Technology, He did Ph.D. Computer Science at VTU in 2014, MPhil-Computer Science at MS University in 2003, MCA at Bangalore University, Bangalore. He has 27 years of teaching experience and 2 years of industry experience. He has published 60 research articles in various National / International Conferences / International Journals. He is working as a reviewer in different International Journals. He has conducted various National Level workshops and Delivered Lecture and Conducted National Conferences. His areas of interest are Distributed Database, Cloud computing, Object Technology, Software Engineering and Performance Engineering.