International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-5, January 2020

Type Ii Fuzzy Logic Controllers for VM Management and Task Assignment in Cloud System

Rashmi Singh Lodhi, R. K. Pateriya

Abstract*: Task distribution and VM (virtual machine) management are the foremost requirements for efficient resource utilization and ensuring SLO (service level objective) of a cloud computing system. To achieve this, it is important to configure VMs depending upon the requirements of the tasks, find proper VM-Task pairs to distribute the tasks over VMs and control the status of VMs. In this paper, a type-II fuzzy logic controller (FLC) based cloud resource management approach is presented. The presented approach contains four type-II FLCs based decision-making systems. The proposed algorithm firstly try to find the most suitable VM-task pair for task assignment, secondly, if it fails in the first step, then it creates a new VM with an appropriate configuration for the given task, at last, it controls the status (Active, Sleep, Shutdown, and Terminate) of running VMs based on their activities and resources utilization to free up resources and reduce power consumption. The simulation result shows that the proposed cloud VM management and task algorithm provides better QoS (quality of service), reduces the resources and power requirement.

Keywords* : Cloud Computing, Cloud Resource Management, Cloud Task Scheduling, Type-II Fuzzy Logic Controller.

I. INTRODUCTION

Because of their ability to deliver reliable, flexible and scalable computational power, cloud computing systems are increasingly being used by industry, government, and academia [1]. Cloud computing systems offer a wide range of advantages for application deployment, in particular, the ability to provide a vast number of resources with a pay-per-use pricing scheme [2]. Cloud computing utilizes virtualization technology for providing facilities of infrastructure, application, and software services [3]. The virtualization platform manages the virtual machines (VMs) for running the applications or services inside a cloud environment. VMs are the virtual instance of a computer system that runs over a layer abstracted from the actual hardware and is created by virtually assigning and organizing the physical resources [3]. The cloud service provider guarantees the minimum Quality of Service (QoS) level, also known as the Service Level Objective (SLO) [4, 5]. Maintaining the SLO is a challenging task, particularly under dynamic load conditions [6]. Increasing resources may be the one possible approach to achieving the SLO, but this will raise the cloud's installation as well as the maintenance and running costs. Another approach is to efficiently utilize the available resources, although this approach does not raise the cost but requires a complex algorithm to dynamically handle the limited resources. Such algorithms need proactive control of VMs configuration, VMs status and task distribution based on current workload requirements, power-saving policies, task queue length, and SLO. In this paper, a type-II fuzzy logic controller (FLC) [7] based approach is presented to achieve such controlling of VMs. The type-II FLC is a modified version of type-I FLC, it was introduced to handle the uncertainties in defining the membership functions with crisp boundaries. The supremacy of type-II over type-I FLC for control problems is already validated in [8].

The rest of the paper is structured as follows. Section 2.0 gives a brief review of the literature. Section 3.0 provides a type-II FLC's overview. Section 4.0 describes the architecture of the proposed system. Section 5.0 presents the simulation results with detailed analysis. Finally, in section 6.0 conclusion and the possibilities of future work are discussed.

II. LITERATURE REVIEW

Several techniques have been proposed for efficient resource utilization and task distribution in cloud computing. In literature, this problem is mostly handled as an optimization problem [9-14] that finds the optimal VM-task pairs which satisfy the objectives (such as QoS, power requirement, processing cost). Although in literature [9-14] optimization is in common, however different optimization techniques are utilized. Like in [9] ant colony optimization (ACO) algorithm based task scheduling is presented which is mainly targeted to achieve green computing (power reduction). In [10] Honey-Bee behavior (HBB) based optimization technique is used to achieve load balancing over available VMs, the presented algorithm also includes the assure handling of task priorities. In [11] a survey of particle swarm optimization (PSO) based scheduling algorithms are discussed.

Manuscript published on January 30, 2020.
* Correspondence Author
Rashmi Singh Lodhi*, PhD Scholar (CSE), Maulana Azad National Institute of Technology (MANIT), Bhopal, India. Email: rrashi_jbp@yahoo.co.in
Dr. R. K. Pateria, Associate Professor, Dept. of Computer Science & Engineering, Maulana Azad National Institute of Technology (MANIT), Bhopal, India. Email: pateriyark@gmail.com

Similarly in [12], [13], and [14], multi-population genetic algorithm, improved harmony search optimization, and cuckoo search algorithm are utilized respectively. As it is already clear that resource utilization and task distribution involves multiple objectives, hence using multi-objective optimization algorithms could help in achieving multiple goals simultaneously.

Knowing the advantages of multi-objective optimization many authors [15-17] adopted this approach. The multi-objective particle swarm optimization presented in [15] tackles three conflicting objectives named as revenue maximization for the service provider, cost minimization for users and maintaining QoS simultaneously. Ant colony based multi-objective optimization algorithm presented in [16], considers two objectives named as performance and cost, under two constraints, makespan (job completion time) and user's budget. While in [17] makespan, cost and resource utilization are taken as objectives and a multi-objective-oriented cuckoo search algorithm is taken for optimization. Apart from optimization and other iterative algorithms, Fuzzy Logic Controllers (FLCs) emerges as a non-iterative and efficient approach, many control and decision-making systems in industrial and domestic applications, using fuzzy logic controllers have been successfully implemented [18-20]. In [21] the fuzzy logic controller based VM selection algorithm to assign the upcoming task, under imprecise memory, bandwidth and disk space specifications. A simple VM allocation method using Fuzzy logic controllers based on Mamdani and Sugeno inference processes is presented in [22]. Type-I and type-II fuzzy logic controllers for load balance and availability (virtual execution unit (VEU)) prediction are presented in [23] for virtual data centers (VDC) with the uncertain workload and uncertain availability of VEU nodes. The job scheduling algorithm which uses fuzzy logic with the genetic algorithm to reduce the iterations is presented in [24]. A task scheduling algorithm that combines particle swarm optimization and fuzzy logic to optimize makespan and waiting time is presented in [25]. Another algorithm with similar objectives but using the genetic algorithm and fuzzy logic is presented in [26].

III. FUZZY LOGIC SYSTEM

In traditional crisp set theory an element can either be a member of a set or not, there is no way by which an element may comprise partial degree of memberships with multiple sets. Because of this in many practical scenario the crisp set definitions are naturally unsuitable [27]. The fuzzy logic uses the fuzzy set concept to solve the complex problems. The Fuzzy logic is a method where a variable may belongs to multiple classes simultaneously with certain degree of membership which is specified by membership function. The fuzzy logic makes decisions using linguistic variables [28]. It is particularly useful in problems where information cannot be precisely described, but it can be approximated by some broad definitions. Fuzzy-logic has been adopted in science industrial and domestic applications due to its simplicity and performance

A. Type-I Fuzzy Logic System

The most common form of fuzzy logic system is type-I, it was firstly proposed by Professor Lofti Zadeh [26]. A typical architecture of type-I FLC is presented in fig. 1, which comprises of six functional blocks: Crisp Input, Fuzzifier, Fuzzy Rule Base, Inference Engine, Defuzzifier, and Crisp Output.

Crisp Input: this is the input interface of the FLC with external environment, from where it receives the crisp data for decision making or controlling.

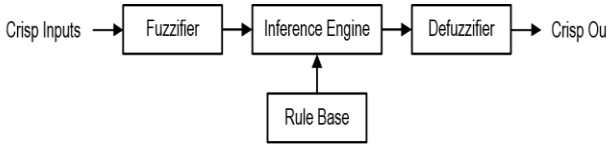


Fig. 1: Typical Architecture Type-I Fuzzy Logic Controller (FLC) System.

Fuzzification: Fuzzification is the method of translating the crisp values in terms of degree of membership with fuzzy sets. The degree of membership is determined by the membership functions. Fuzzification enables the linguistic representation of crisp data.

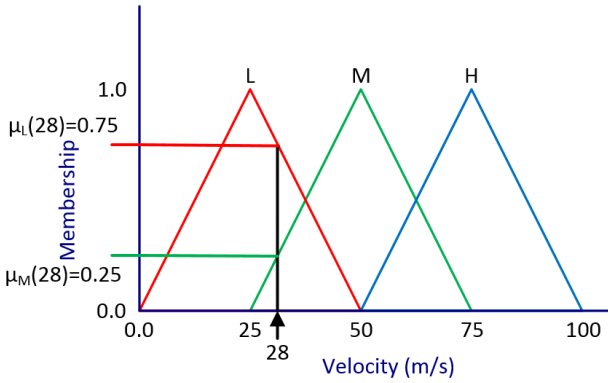


Fig. 2: Fuzzification of Velocity with Triangular Membership Function.

In fig. 2, an example of determining the membership of velocity with fuzzy sets L (Low), M (Medium) and H (High). The figure shows that for velocity 28 m/s (black line vertical), the membership values for L, M, and H fuzzy sets are 0.75 (red line horizontal), 0.25 (green line horizontal) and 0 respectively. For the simplicity in this example a triangular membership function is chosen, however a number of membership functions exists such as trapezoidal, bell-shaped, sigmoidal, etc. [29]. The selection of membership function depends upon the application and expert's knowledge.

Fuzzy Rule Base: It stores the rules that linguistically connects the fuzzy inputs and outputs. It generally contains IF-THEN rules chain. For example, the rule base of an automatic vehicle speed controlling system may contains following rules:

- IF velocity is L, THEN set Accelerator to H
- IF velocity is M, THEN set Accelerator to M
- IF velocity is H, THEN set Accelerator to L

Here only three rules are shown however in practical system may contains hundreds of rules, also defining rules requires expertize knowledge about the field of application.

Inference Engine: It generates the fuzzy outputs by processing the fuzzy inputs according to the rules defined in rule base.

Defuzzifier: The inference engine produces fuzzy outputs, which must be transformed to a crisp value before it can be used with any non-fuzzy system.

The Defuzzifier is used to perform this conversion. De-Fuzzification can be carried out in many ways, such as the use of center of gravity, center of area, middle of maxima, and mean of maxima, etc. [29].

B. Type-II Fuzzy Logic System

Although the type-I FLC's have been successfully used in many applications, it faces problems when applied on real world unstructured dynamic environments [7]. Such problems are caused by non-fuzzy (strictly defined) membership functions boundaries. Since the strictly defined membership functions boundaries cannot handle linguistic uncertainties.

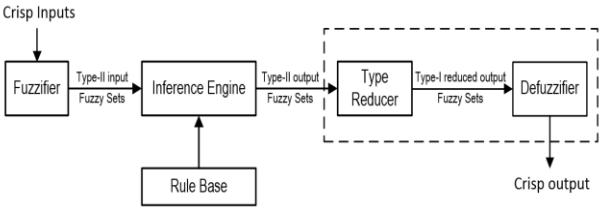


Fig. 3: Typical Architecture Type-II Fuzzy Logic Controller (FLC) System.

For example, consider a type-I fuzzy set as presented in fig. 2, we can see that the velocity of 28 m/s has a 0.75 crisp membership with the linguistic label of "Low". Hence if the input velocity is 28 m/s, then its membership to the "Low" in type-I set will be a strictly defined (certain) and crisp value of 0.75. However, the center and endpoints of shown triangular type-I fuzzy set will vary with the road type, the country, the context, the human preferences and other aspects and uncertainties. Hence, employing this linguistic label with a FLC to control the vehicle velocity, then the continuously tuning of type-I FLC would be needed, to handle all the confronted uncertainties. Alternatively, a group of separate type-I sets and type-I FLC will be needed to handle the each possible situation.

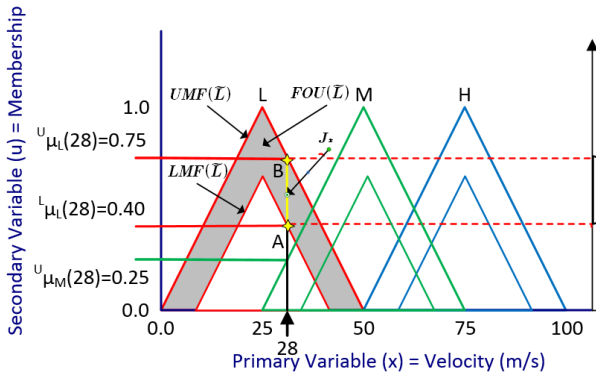


Fig. 4: Fuzzification by General Type-II Fuzzy Set.

To overcome the limitations of type-I fuzzy system (FS), the general type-II FS uses a 3D membership functions with the x, y and z-axis are called primary variable, secondary variable and secondary MF (membership function) value and are denoted by x, u and $\mu_{\tilde{A}}(x,u)$ respectively [31]. The secondary MF value at each point of the space bounded by lower membership function (LMF) and an upper membership function (UMF) is called its footprint of uncertainty (FOU). If the secondary MF value remains constant then the general type-II FS is called as interval type-II FS.

The rule base remains the same in type-2 FLC as in type-1 FLC. The inference engine incorporates the fired rules and provides a mapping of fuzzy sets from type-2 input to type-2 output fuzzy sets. Then the type-reducer handles the type-2 fuzzy outputs of the inference engine, which combines the output sets and performs a centroid calculation leading to type-1 fuzzy sets called the type-reduced sets. The iterative Karnik-Mendel (KM) technique is used to produce type-reduced fuzzy sets. The convergence of the KM method is proportional to the number of fired rules and can therefore cause the type-2 FLC to have a computational bottleneck. The type-reduced sets are then defuzzified (taking the type reduced set average) to obtain crisp output [7].

IV. PROPOSED ALGORITHM

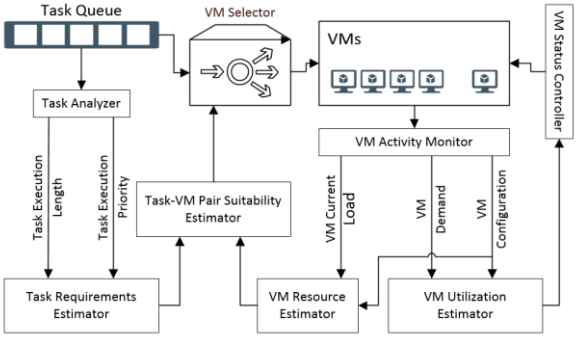


Fig. 5: The proposed system architecture.

This section describes the each component (as shown in fig. 5) of proposed algorithm.

A. System Architecture

The proposed system architecture is shown in fig.5 the system contains many functional blocks, however our main contribution is in four blocks of type-II fuzzy logic estimators. The block's details are as follows:

- ***Task Analyzer:*** The functioning of this block is to extract:
  - The Current Task Length: length of the current task in the queue in Mega Instructions (MI) and
  - Task Priority: how quickly the task need to be served.

Information from task queue.

- ***VM Activity Monitor:*** The functioning of this block is to monitor the activity and configuration parameters of each VM. The three main observations taken by this block are:
  - VM Demand: how many times a VM is accessed in a session,
  - VM Current Load: the length of the currently executing task (in MI), remaining in the VM, and
  - VM Configuration: the execution capacity of the VM in Mega Instructions per Second (MIPS).

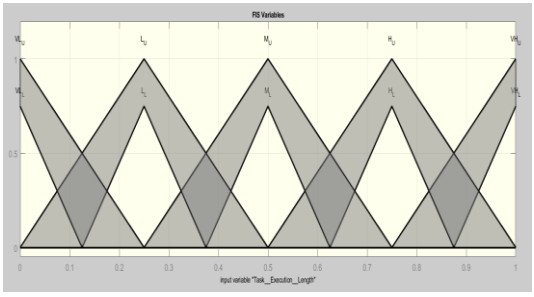
- ***VM Status Controller:*** The functioning of this block is to force the VMs to one of the following four states:
  - Active: the VM is running and is either executing a task or waiting for a task,
  - Sleep: the VM is formed and the resources have been acquired by VM, it is already booted but it's all resource are forced to power saving mode,
  - Shutdown: the VM's resources are reserved, but in power off state, and
  - Terminate: all the resources of VM is reclaimed by cloud, the VM no longer exists.

  The status of a VM is decided by the input control signal coming from the VM Utilization Estimator block. This incoming signal is compared with three different threshold values related to the each state δ_{sleep}, $\delta_{shutdown}$ and $\delta_{terminate}$ such that, $1 < \delta_{sleep} < \delta_{shutdown} < \delta_{terminate} < 0$. The mathematical representation is as follows:

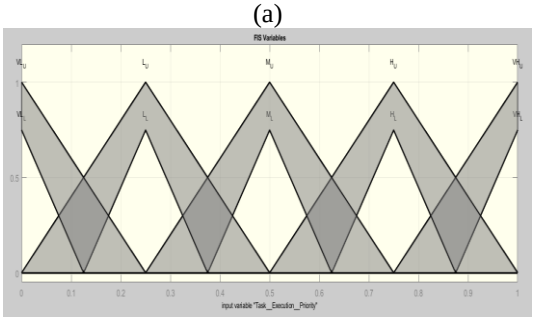
$$VM_{status} = \begin{cases} Running, & if\ VM_{uti} > \delta_{sleep} \\ Sleep, & elseif\ \delta_{sleep} \geq VM_{uti} > \delta_{shutdown} \\ Shutdown, & elseif\ \delta_{shutdown} \geq VM_{uti} > \delta_{terminate} \\ Terminate, & elseif\ VM_{uti} \leq \delta_{terminate} \end{cases} \#(1)$$

- ***VM Selector:*** The functioning of this block is to assign the current task to the selected VM according to the decision taken by Type-II Fuzzy Logic based Task-VM Pair Suitability Estimator block.

- Type-II Fuzzy Logic Decision Making System: each fuzzy block used in this system produces a scalar value in a closed interval of [0, 1]. we proposed the use of four such blocks:



(a)

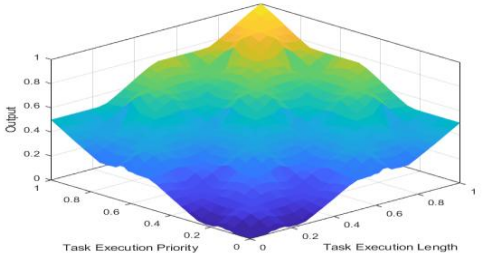


(b)

Task Execution Priority

Task Execution Length	VL	L	M	H	VH
VL	*VL*	*VL*	*L*	*L*	*M*
L	*VL*	*L*	*L*	*M*	*M*
M	*L*	*L*	*M*	*M*	*H*
H	*L*	*M*	*M*	*H*	*H*
VH	*M*	*M*	*H*	*H*	*VH*

(c)

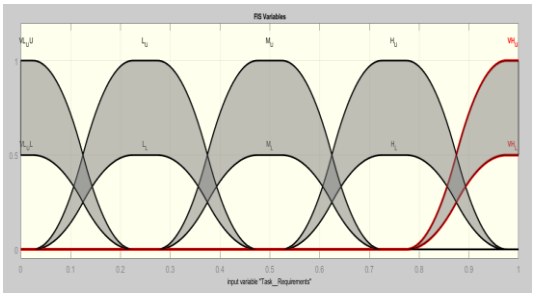


(d)

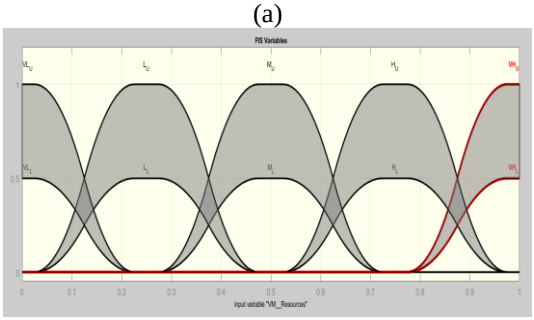
Fig. 6: Interval Type-II Fuzzy Logic System construction details for Task Requirements Estimator, (a) Membership functions for Fuzzification of input variables Task Execution Length, and (b) Task Execution Priority, both uses triangular membership functions having narrow fuzziness at peaks, (c) shows the Rule Base for inputs and output relations (only AND rules are used), (d) shows the surface plot of the designed system.

- ***Task Requirements Estimator:*** The work of this block is to estimate the resource demands of the task by using task execution length and priority as inputs. For example a long task with high priority can be interpreted as a long task need to execute as quickly as possible, which means the task demands a VM which can execute the task within the priority time, this ultimately indicated by the task requirement estimator as higher output value. The fuzzy system architecture for this block is shown in fig. (6), the purpose of the using triangular membership function is the consideration that both the input variables gradually changes their membership with different sets. The narrow peak (difference between UMF and LMF at peak) of membership functions shows greater certainty of value at center of each set.
- ***VM Resource Estimator:*** The work of this block is to estimate the current resources capability of a VM, by using VM configuration and Load as inputs. For example a VM configured using high speed processor with idle (no load) running state, can be interpreted as a high configuration VM ready to execute task immediately and quickly, which means the VM is capable to execute the long task within the priority time, this ultimately indicated by the VM resource estimator as higher output value. The fuzzy system architecture of this block is identical to the Task Requirement Estimator.
- ***Task-VM Pair Suitability Estimator:*** As explained above that the Task Requirements Estimator and VM Resource Estimator have demand and capability relation, this block estimates the suitability between task requirements and VM capability.

The fuzzy system architecture for this block is shown in fig. (7), the purpose of the using Pi-shaped membership function is the consideration that both the input variables gradually changes their membership initially and then after remains stable to moved set. The wider peak (difference between UMF and LMF at peak) of membership functions shows greater uncertainty of value at center of each set.



(a)

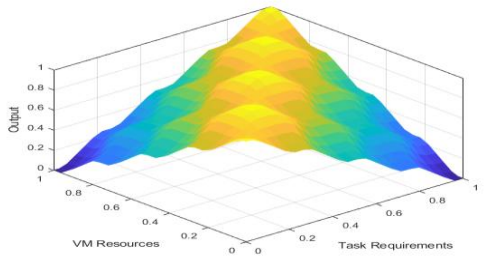


(b)

VM Resources

Task Requirements	VL	L	M	H	VH
VL	*VH*	*H*	*M*	*L*	*VL*
L	*H*	*M*	*H*	*M*	*L*
M	*M*	*H*	*VH*	*H*	*M*
H	*L*	*M*	*H*	*VH*	*H*
VH	*VL*	*L*	*M*	*H*	*VH*

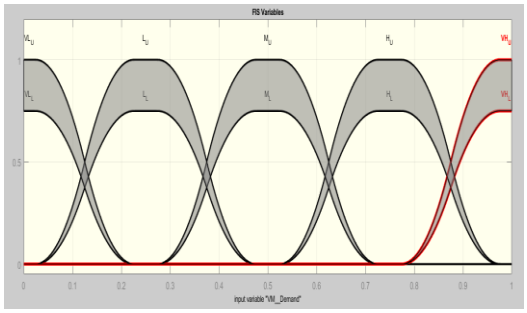
(c)



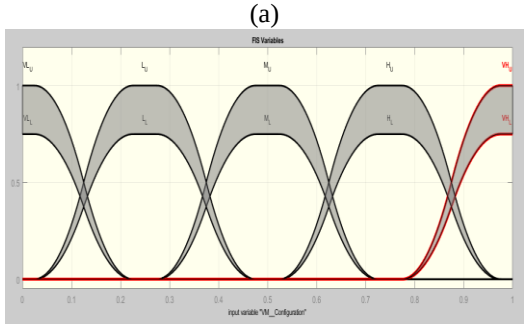
(d)

Fig. 7: Interval Type-II Fuzzy Logic Controller construction details for Task-VM Pair Suitability Estimator, (a) Membership functions for Fuzzification of input variables, Task Requirements, and (b) VM Resources, both uses Pi-shaped membership functions having narrow fuzziness at bottom, (c) shows the Rule Base for inputs and output relations (only AND rules are used), (d) shows the surface plot of the designed system.

- ***VM Utilization Estimator:*** The work of this block is to estimate controlling inclination of a VM status towards task or towards cloud, by using VM configuration and demand as inputs. The higher value from this estimator indicates that the VM status should be inclined towards cloud or VM's status should be controlled to benefit cloud in terms of power or resources by forcing the VM status to sleep state, shutdown or terminate. On the other hand the lower value from this estimator indicates that the VM status should be inclined towards task or VM's status should be controlled to benefit task in terms of readiness and compatibility by forcing the VM status to running state. The fuzzy system architecture for this block is shown in fig. (8), the also uses the Pi-shaped membership function like Task-VM Pair Stability Estimator, but with relatively smaller peak (difference between UMF and LMF at peak) of membership functions shows a bit greater certainty of value at center of each set. Furthermore, its surface plot is not symmetric but skewed toward VM Demand as we provided slightly more weight to VM Demand (for better QoS) than VM Configuration (for better power saving).



(a)

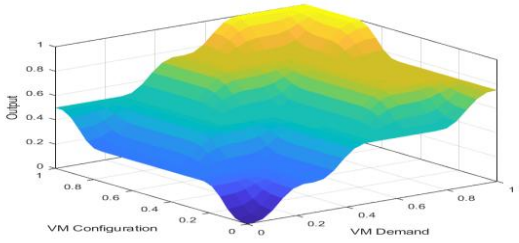


(b)

VM Resources

Task Requirements	VL	L	M	H	VH
VL	*VL*	*L*	*L*	*L*	*M*
L	*L*	*L*	*M*	*M*	*M*
M	*M*	*M*	*M*	*H*	*H*
H	*M*	*H*	*H*	*H*	*VH*
VH	*H*	*H*	*H*	*VH*	*VH*

(c)



(d)

Fig. 8: Interval Type-II Fuzzy Logic Controller

construction details for VM Utilization Estimator, (a) Membership functions for Fuzzification of input variables, VM Demand, and (b) VM Configuration, both uses Pi-shaped membership functions having narrow fuzziness at bottom and average fuzziness at peaks, (c) shows the Rule Base for inputs and output relations (only AND rules are used), (d) shows the surface plot of the designed system.

B. Algorithm Explanation

The algorithm explanation needed following terminology:

Table I: Terms and their Definitions.

Symbol	Term	Description
L^i	Length of i^{th} task in queue.	This represents the number of instructions (cycles) required to execute the task. In this work the task length is taken in units of MI (millions of instructions).
l_i	Length of the task remaining in i^{th} VM.	This represents the number of instructions to be executed by a VM before starting the execution of a new task.
P^j	Execution Priority of j^{th} task.	Indicator of SLO for executing the task in given time frame. The guaranteed task completion time for the j^{th} Task is given as $1/P^j$
C_i	Processing Capacity of i^{th} VM	This represents the instructions executing rate of VM. In this work it is described in MIPS (millions of instructions per second).
D_i	Demand of i^{th} VM	This represents the number of times a VM is selected for input task in a session.
W_i	Waiting time for i^{th} VM	The time taken by the VM before starting the execution of the task. This occurs when VM has not finished the previously assigned task. Calculated as l_i/C_i
E_i^j	Execution time for j^{th} task by i^{th} VM	Calculated by L^j/C_i.
T_i^j	Total task completion time for j^{th} task by i^{th} VM	This includes the waiting time and execution time of a task, and calculated as $W_i + E_i^j$.
$fun_{TRE}(\cdot)$	Type-II Fuzzy task requirements estimator function	A functional representation of Task Requirements Estimator block, it takes two inputs (1) Task Execution Length (2) Task Execution Priority.
$fun_{VRE}(\cdot)$	Type-II Fuzzy VM resources estimator function	A functional representation of VM Resources Estimator block, it takes two inputs (1) VM Current Load (2) VM Configuration.
$fun_{TVP}(\cdot)$	Type-II Fuzzy Task-VM pair compatibility estimator function	A functional representation of Task-VM Pair Compatibility Estimator block, it takes two inputs (1) Task Requirements (2) VM Resources.
$fun_{VMU}(\cdot)$	Type-II Fuzzy VM pair Utilization estimator function	A functional representation of VM Utilization Estimator block, it takes two inputs (1) VM Demand (2) VM Configuration.
TVP	2D array to store Task-VM pair Satisfy SLO.	A 2D array with rows equal to number of tasks in queue and columns equal to maximum number of VMs in Cloud. A nonzero entry in an element, say (j, i) of this array represents that i^{th} VM satisfies the SLO for j^{th} task.

The proposed could be explained by the pseudo codes shown in fig. 9 and flowchart shown in fig. 10.

```
Main Loop
N = numbers of VMs in the Cloud.
M = numbers of Tasks in Queue.
/* Step 1: calculate the total task completion
 time for each task in each VM */
for i = 1 to N
    for j = 1 to M
        W_i = l_i / C_i
        E_i^j = L^j / C_i
        T_i^j = W_i + E_i^j
    endfor
endfor
/* Step 2: find Task-VM pairs that can satisfy SLO */
TVP = 0
for j = 1 to M
    pairNotFound = 1
    for i = 1 to N
        if (T_i^j > 1/P^j)
            pairNotFound = 0
            TVP[j, i] = 1
        endif
    endif
    if (pairNotFound == 1)
        Call CreateNewVM(L^j, P^j)
        TVP[j, i] = 1

endfor
Call SelectVM(L^j, P^j, TVP)
Call VMStatusControl(D_i, C_i, l_i)
End Main Loop
```

```
SelectVM
Inputs: L^j, P^j, TVP
TVP_est[M × N] = 0;
for i = 1 to M
    for i = 1 to N
        if (TVP[j, i] == 1)
            tmp_1 = fun_TRE(L^j, P^j)
            tmp_2 = fun_VRE(l_i, TP_i)
            TVP_est[j, i] = F_VMR(tmp_1, tmp_2)
        endif
    endfor
endfor
for i = 1 to M
    index = max(TVP_est)
    if (index = ∅)
        assign VM_index[1] ← Task^index[2]
        TVP_est[:, index[2]] = 0
    else
        Call CreateNewVM(L^i, P^i)
    endif
endfor
End SelectVM
```

```
CreateNewVM
Inputs: L^j, P^j
        C_new = L^j × P^j
        VM ← C_new // Set Configuration
        VM ← Start // Start VM
        VM ← L^j   // Assign Task
create new VM with configuration C_VM^new;
assign Task to VM_new.
End CreateNewVM
```

```
VMStatusControl
Inputs: D_i, C_i
for i = 1 to N
    if (l_i == 0)
        tmp = fun_VMU(D_i, C_i)
        if(tmp ≤ δ_terminate)
            Reclaim VM_i Resources
        elseif (tmp ≤ δ_shutdown)
            Shutdown the VM_i
        elseif (tmp ≤ δ_sleep)
            Sleep the VM_i
        else
            do nothing
        endif
    endif
endfor
End VMStatusControl
```

Fig. 9: Pseudocodes for the proposed algorithm

The proposed could be explained by the pseudo codes shown in fig. 9 and flowchart shown in fig. 10. At first step the cloud manager scans the Task Queue, and extracts the information about Task Execution Length and Task Execution Priority for each task in the queue. It also scans for the information about VM Configuration, VM Current Load and VM Utilization Rate for each VM in the cloud. It is assumed that each measured parameters are normalized between the intervals $[0, 1]$. Since according to SLO the task must be delivered according to their priorities, hence for each task a VM satisfying the priority execution criteria is searched. If for any task the required VM is not found a new VM is created according to the task requirements and the task is assign to it immediately. Now for the each task that has not currently assigned to a VM, Task-VM Suitability with all the available VMs are estimated. To do this firstly the single scalar variable indicating the Task Requirements and single scalar variable indicating the VM Resources are calculated. The Task Requirements is estimated using two parameters related to task named as Task Execution Length and Task Execution Priority, on the other hand VM Resources are estimated using VM Current Load and VM Configuration.
Once the Task distribution is done another work of the proposed algorithm is to save the power and reclaim the resources from the VMs, while making sure that the QoS and SLO is not compromised. This is performed by calculating the VM Utilization Estimation for each VM, the estimation returns a scalar value indicating that weather the VM should kept in Active State (if VM is in high or very demand) to immediately start execution of a task, Sleep Sate (if VM is in average demand also not acquiring much resources) to save the power and execute the task with minimum delay, Shutdown State (if VM is low demand and acquiring high resources) to save the power, however with average delay in task starting execution, and Terminate State (if VM has lowest demand and acquiring high or very resources) to reclaim the resources for future VM formation. The VM is forced to one of the four above mentioned status by VM Status Controller which compares the VM Utilization value with four pre-decided thresholds to decide this.

V. SIMULATION RESULTS

The details about experimental setup, results and discussion are given in this section to illustrate the efficacy of the proposed scheme. We used the MATLAB to model the proposed system, instead of commonly used tool CloudSim [32] as we found that it is much easier to design type-II fuzzy logic controller using already available GUI based Toolbox [33], also the MATLAB language is fairly simple and compact than Java and it comes with enormous mathematical functions pre-installed. Although the CloudSim can perform more detailed simulation, but for our purpose the MATLAB based cloud model with limited capability is adequate.

A. Performance Metrics

The metrics we use to check and compare the algorithms are discussed in this section. We consider five performance metrics, SLO Failure (Number of Tasks), SLO Failure (Total Execution Length of Failed Tasks), VM Wakeups, VM Reboots, VM Creations, and Resource Utilization Efficiency. These metrics are defined as follows:

SLO Failure (Number of Tasks) or SLO Deadline Violation: It is characterized as the total number of tasks to which cloud failed to execute within a guaranteed time limit.

SLO Failure (Total Execution Length of Failed Tasks): it is related to the SLO Failure (Number of Tasks), but instead to counting the number of tasks it sum up the tasks execution length of failed tasks.

VM Wakeups: It represents the number of VMs wakeup from sleeping state, (This operation is performed if the VMs in active state cannot meet the current task SLO).

VM Reboots: It represents the number of VMs rebooted from shutdown state, (This operation is performed if the VMs in active state and sleeping state cannot meet the current task SLO).

VM Creations: It represents the number of new VMs are created, (This operation is performed if the VMs in active state, sleeping state and shutdown state cannot meet the current task SLO).

Resource Utilization: Shows how effectively cloud resources are used to execute the tasks, and is measured as follows:

$$\begin{aligned} &Resource\ Utilization\ Efficiency \\ &= \frac{Total\ Task\ Length\ Served}{Total\ Execution\ Capacity\ Used} \times 100 \\ &= \frac{\sum_{i=1}^{N} L^i}{\sum_{i=1}^{N} \left(\sum_{j=1}^{A_i} C_j^i \right)} \times 100, \#(2) \end{aligned}$$

where, N: is the total number of events (discrete events of task arrival) in simulation time, L^i: is the load arrived in cloud at event i, A_i: is the number of VMs active at event i. C_j: Execution capacity of the j^{th} VM at event i.

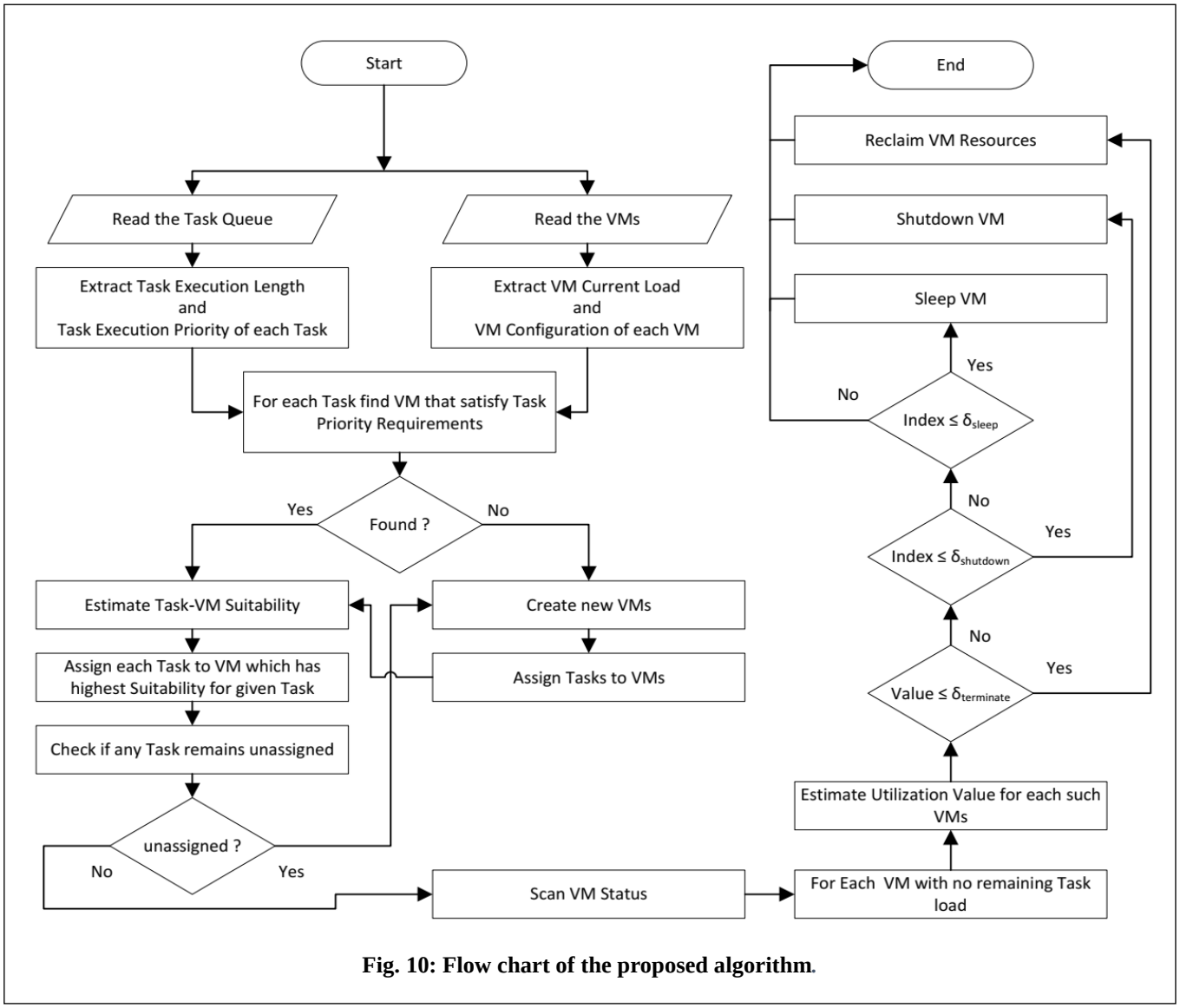


Fig. 10: Flow chart of the proposed algorithm.

Power Consumption: the power consumption of a physical server as presented in [34-36]:

$$P_{Total} = P_{Dynamic} \times U_{Avg} + P_{Idle}, \#(3)$$

where, P_{Total}: is the total power consumption by the server, $P_{Dynamic}$: is the dynamic power consumption during the load. P_{Idle}: is the power consumption during no work load.
However we modified the above formulation as in propose system two additional states P_{Sleep} and $P_{Shutdown}$ are required, the modified equation can be written as:

$$P_{Total} = P_{Dynamic} \times U_{Avg} + P_{State}(s)$$

$$where, \quad P_{State}(s) = \begin{cases} P_{Idle}, & s = 1 \\ P_{Sleep}, & s = 2 \\ P_{Shutdown}, & s = 3 \end{cases}, \#(4)$$

where, P_{Sleep}: Power consumption during sleeping state. $P_{Shutdown}$: Power consumption during Shutdown state. The values for $P_{Dynamic}, P_{Idle}, P_{Sleep}$ and $P_{Shutdown}$ are (as provided in [37]): $P_{Dynamic} = 240\ Watts$, $P_{Idle} = 150\ Watts, P_{Sleep} = 10\ Watts, P_{Shutdown} = 0\ Watts$.

B. Simulation Setup for Cloud System

In order to properly simulate the algorithm, the configuration parameters need to be configured, with their values are listed in **Error! Reference source not found.**II.

Table II: The simulation parameters and their values.

Configuration Parameter	Value
Total Processing Capacity Available in Cloud	100 MIPS
Task Arriving Model	Poisson Distribution
Average Task Execution Length	10, 25, 50, 75, 100 MI (λ—Poisson Distribution)
Task Execution Priority	Folded or Half Normal Distribution ($\mu = 0, \sigma = 0.25$)
Task Queue Length	10
Threshold Sleep (δ_{sleep})	0.75
Threshold Shutdown ($\delta_{shutdown}$)	0.50
Threshold Terminate ($\delta_{terminate}$)	0.25
Total Simulation Time	24 Hours (1440 Minutes)
Simulation Repetition	50 times

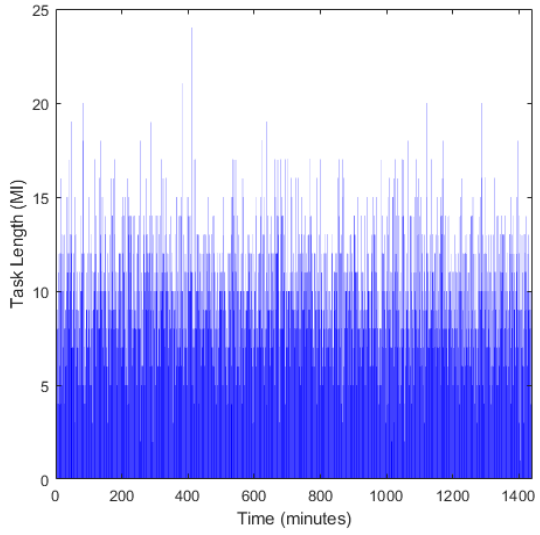
Table III: VM state transition time considered during the simulation.

State		Term	Value (Seconds)
From	To		
Non-Existence	Running	T_{Create} (VM Creation Time)	90.0
Shutdown	Running	$T_{Booting}$(VM Booting Time)	50.0
Sleep	Running	T_{Wakeup} (VM Wakeup Time)	25.0

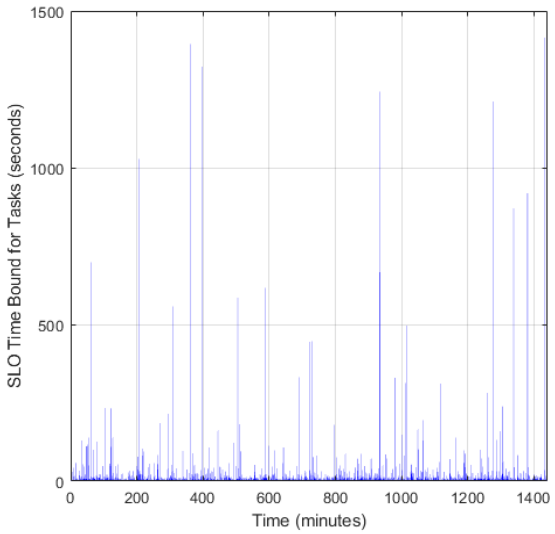
During the simulation the tasks arrival is modeled as a Poisson process at a rate of λ (Average Task Execution Length), while the Task Execution Priority is modeled using Folded or Half Normal Distribution also the Task SLO Time Bound is considered as the inverse of Task Execution Priority.

C. Simulation Results and Discussion

The behavior of the developed model is described through figures 11 to 14. The figure 11 (a), shows the cloud incoming work load characteristics, as it can be seen that the task length peak value goes up to the 2.5 times of average task length (λ), whereas the SLO time bound values (figure 12(b)) show much higher variations. The difference is occur due to utilization of different probability distribution function.



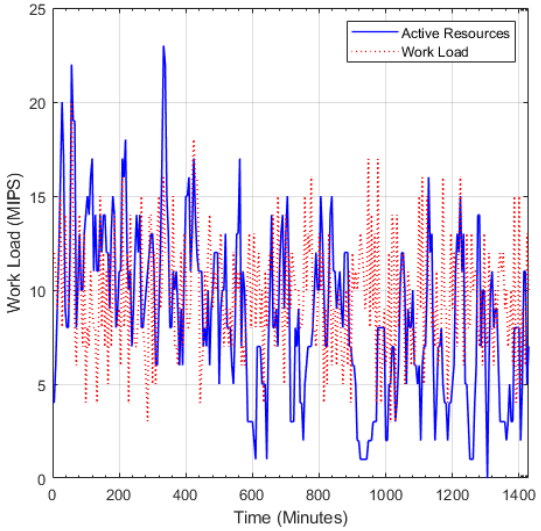
(a)



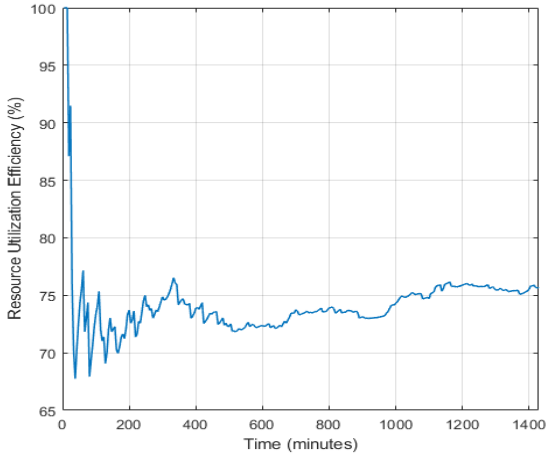
(b)

Fig. 11: Plot (a) shows the Task Arrival with time for λ=10, and (b) shows the respective SLO Guaranteed task execution time bound (inverse of Task Priority) for each task.

The figure 12(a) shows the comparison between the work load in the cloud and the active resources over the time, it can be seen that the active resources are closely related to the work load in the cloud, which proves the effectiveness of the proposed algorithm. As the similarity between these two corresponds to efficient utilization of resources, which can again be seen in figure 12(b), resource utilization efficiency of the cloud over the time is plotted. The resource utilization efficiency starts at 100% at $t = 1$, thereafter it decreases to 75%, this happens because the proposed algorithm is not only focused to resource utilization but also on SLO and energy minimization. For those some VMs are kept on to minimize the response time, and some are kept in sleeping modes to save power. However, the resource utilization efficiency as the work load increases as shown in figure 15.



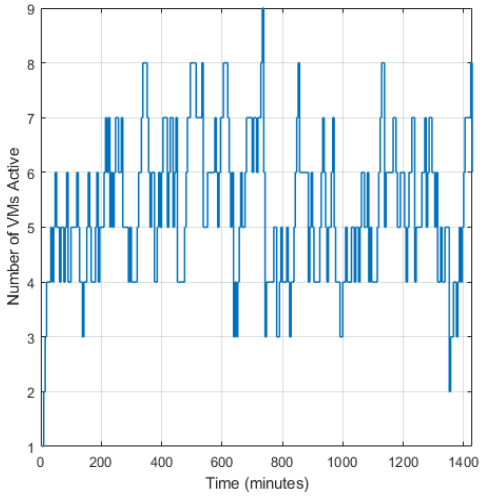
(a)



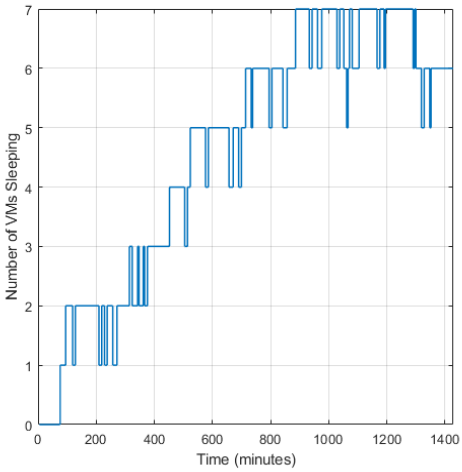
(b)

Fig. 12: Plot (a) shows the comparison of Total Work Load and Active Resources (Processing Power) in Cloud at any instant of time for λ=10, and (b) shows the variations in Cloud Resource Utilization Efficiency over the time.

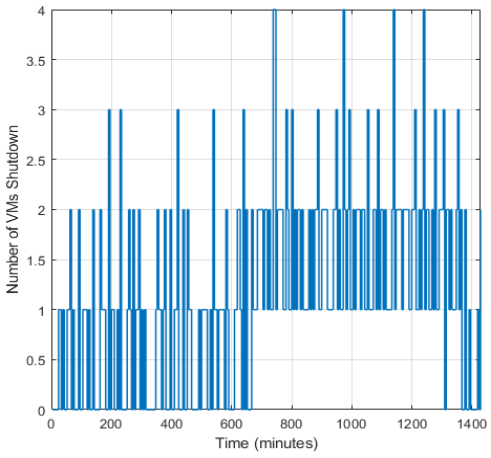
The behavior of the proposed algorithm and the concept of different operational modes of VM can be understood through figure 13, the figure 13(a) shows how the number of active VMs varies over the time to meet the task demands, however on the other hand it can also be seen that the VM are continuously sent to sleeping (figure 13(b)) and shutdown (figure 13(c)) state to minimize the power consumption. Lastly the unused VMs are terminated and its resources are merge with the free resources for future utilization (figure 13(d)).



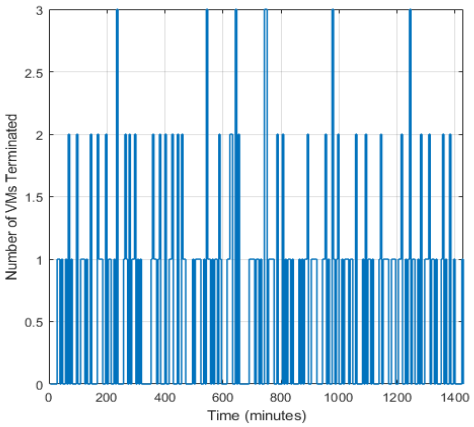
(a)



(b)

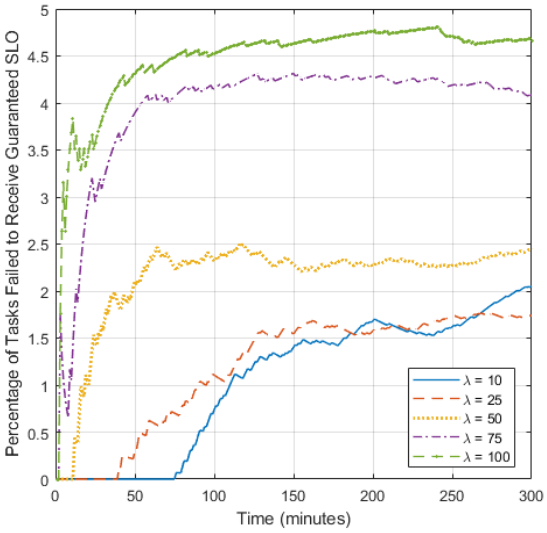


(c)

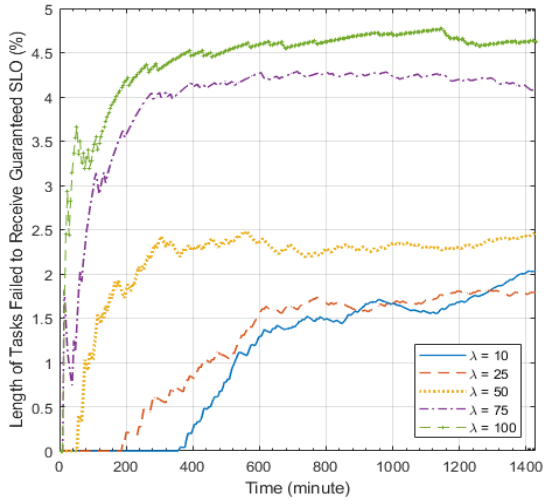


(d)

Fig. 13: Plot (a) shows the variations in total number of Active VMs in Cloud over the time, (b) shows the variations in total number of VMs in sleeping state, (c) shows the variations in total number of VMs in shutdown state and, (d), shows the termination of VMs over the time.



(a)



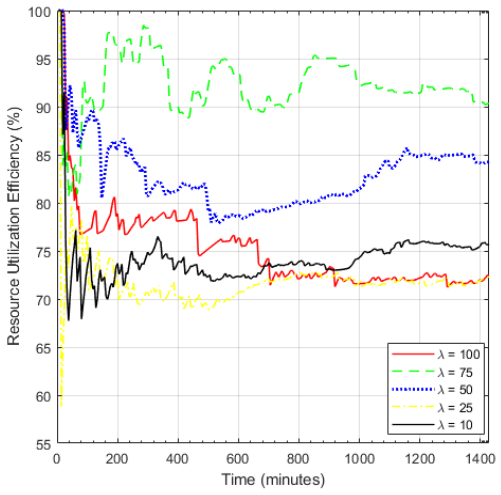
(b)

Fig. 14: Plot (a) shows the percentage of tasks failed to get

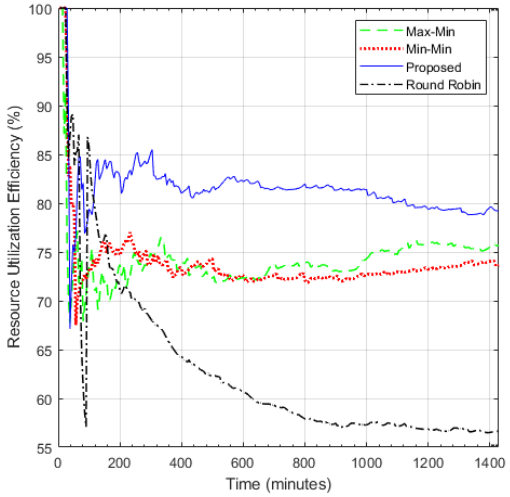
the guaranteed SLO in Cloud at any instant of time for different values of λ, and (b) shows the total length of failed tasks in percentage over the time.

The impact of work load variation on resource utilization efficiency is shown in figure 15(a), it shows that the efficiency increases with work load and reaches around 95% average for the $\lambda = 75$, however further increase in work load causes overloading which decreases it efficiency.

A comparison with other algorithm for resource utilization efficiency is presented in figure 15(b), for $\lambda = 50$, which shows that proposed algorithm outperforms the standard algorithm by 5% of margin.



(a)



(b)

Fig. 15: Plot (a) shows the Resource Utilization Efficiency for different average task length conditions ($\lambda = 10, 25, 50, 75, 100$) in Cloud at any instant of time for $\lambda = 10$, and (b) shows the comparison of some standard algorithms with the proposed algorithm for Resource Utilization Efficiency variations in Cloud over the time.

The comparison of the proposed algorithm with standard algorithms for different performance measures are presented in Table IV to VI. To avoid the uncertainty in the results, due to randomness involved in the process, the simulation for each configuration is repeated for 50 times, then from the obtained results the mean, standard deviation and best values are calculated and presented in table below.

Table IV: Comparison of Resource Utilization Efficiency (%), for different work load conditions.

Average Task Length (λ)	Proposed			Max-Min			Min-Min			Round Robin		
	Mean	*Std.*	*Best*	*Mean*	*Std.*	*Best*	*Mean*	*Std.*	*Best*	*Mean*	*Std.*	*Best*
25	72.77	2.68	75.24	71.26	2.61	74.38	68.33	2.33	71.64	60.63	2.63	65.27
50	82.91	4.46	85.10	74.71	4.59	79.22	73.28	4.68	76.47	65.95	5.95	68.92
75	92.18	10.23	95.66	68.09	9.54	73.18	69.28	9.72	74.89	59.22	9.82	66.81
100	75.16	15.25	85.52	65.15	15.83	69.53	60.55	15.05	67.51	62.74	16.74	72.40

Table V: Comparison of SLO failure or SLO Deadline Violation (%), for different work load.

Average Task Length (λ)	Proposed			Max-Min			Min-Min			Round Robin		
	Mean	*Std.*	*Best*	*Mean*	*Std.*	*Best*	*Mean*	*Std.*	*Best*	*Mean*	*Std.*	*Best*
25	1.41	2.08	1.26	3.61	2.44	3.36	3.45	2.58	3.25	6.31	4.12	5.11
50	2.26	3.94	2.12	5.34	4.15	5.03	6.02	4.36	5.67	8.53	7.57	7.62
75	4.19	6.32	3.97	8.86	6.24	7.75	7.76	6.12	6.77	10.22	9.24	8.49
100	4.45	9.02	4.17	10.53	10.38	9.29	9.87	9.15	8.83	14.45	11.81	11.72

Table VI: Comparison of Power Consumption in Percentage of 100% Resource Utilization, for different Average Task Length.

Average Task Length (λ)	Proposed			Max-Min			Min-Min			Round Robin		
	Mean	*Std.*	*Best*	*Mean*	*Std.*	*Best*	*Mean*	*Std.*	*Best*	*Mean*	*Std.*	*Best*
25	346.5	6.39	339.6	354.2	8.46	344.1	356.3	8.55	344.42	397.6	14.34	367.3
50	567.2	24.6	553.4	661.0	28.2	626.4	673.3	28.4	634.3	732.7	56.8	662.1
75	776.5	47.4	744.9	1067.9	69.2	981.6	1015.4	69.0	956.3	1148.5	96.2	1051.9
100	1260.1	112.3	1204.8	1360.5	149.6	1259.0	1560.3	153.6	1419.9	1663.7	186.8	1464.8

The table 3, shows that the proposed algorithm at $\lambda = 25$, provides similar results to Min-Max and Min-Min, however as the λ increases and it achieves the around 15% better resource utilization efficiency at $\lambda = 75$, for mean value and for best value this increases to around 20%. For the SLO dead line violation the proposed algorithm reduces the SLO failure rate to half as compare to other algorithms (Table IV). For the makespan the proposed algorithm outperform other algorithm for all values of λ and it achieves the maximum improve of 30%. When compared for the power consumption the proposed algorithm decreases the power consumption at average load condition by around 10%, at higher load around 20%, and for full load around 15%.

VI. CONCLUSION

In this paper, a type-II fuzzy logic based VM Status and task assignment scheme for the Cloud system is presented. The complete proposed system architecture consists of four type-II fuzzy logic based decision making blocks, configured to decide the task requirements, VM resource capabilities, VM utilization, and Task-VM pair suitability. The application of fuzzy logic simplifies the complexity in decision making such that the task requirements may depend upon many parameters, hence in non-fuzzy system, it will be very difficult to generate a single valued scaler to effectively reflect the task requirements. Furthermore, the utilization of type-II fuzzy logic provides flexibility in handling the uncertainties of input variable definitions. The simulation results and discussion presented in the previous section verifies that the proposed algorithm increases resource utilization efficiency, while decreases the SLO deadline violation, Makespan, and power consumption. It also shows that instead of task scheduling the proper VM status management could also lead to significant power saving, without affecting the cloud performance. The performance of the proposed algorithm can be further improved in the future by optimizing the fuzzy rules and membership functions.

REFERENCES

1. Prevost, John J., KranthiManoj Nagothu, Brian Kelley, and Mo Jamshidi. "Prediction of cloud data center networks loads using stochastic and neural models." In 2011 6th International Conference on System of Systems Engineering, pp. 276-281. IEEE, 2011.
2. Hilman, Muhammad Hafizhuddin, Maria Alejandra Rodriguez, and Rajkumar Buyya. "Task Runtime Prediction in Scientific Workflows Using an Online Incremental Learning Approach." In 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), pp. 93-102. IEEE, 2018.
3. Kovari, Attila, and P. Dukan. "KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE." In 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, pp. 335-339. IEEE, 2012.
4. Wieder, Philipp, Joe M. Butler, Wolfgang Theilmann, and Ramin Yahyapour, eds. Service level agreements for cloud computing. Springer Science & Business Media, 2011.
5. Son, Seokho, Gihun Jung, and Sung Chan Jun. "An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider." The Journal of Supercomputing 64, no. 2 (2013): 606-637.
6. Albano, Luca, Cosimo Anglano, Massimo Canonico, and Marco Guazzone. "Fuzzy-Q & E: Achieving QoS Guarantees and Energy Savings for Cloud Applications with Fuzzy Control." In 2013 International Conference on Cloud and Green Computing, pp. 159-166. IEEE, 2013.
7. Hagras, Hani. "Type-2 fuzzy logic controllers: a way forward for fuzzy systems in real world environments." In IEEE World Congress on Computational Intelligence, pp. 181-200. Springer, Berlin, Heidelberg, 2008.
8. Castillo, Oscar, Leticia Amador-Angulo, Juan R. Castro, and Mario Garcia-Valdez. "A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems." Information Sciences 354 (2016): 257-274.
9. Ari, Ado Adamou Abba, Irepran Damakoa, Chafiq Titouna, Nabila Labraoui, and Abdelhak Gueroui. "Efficient and scalable ACO-based task scheduling for green cloud computing environment." In 2017 IEEE International Conference on Smart Cloud (SmartCloud), pp. 66-71. IEEE, 2017.
10. D. B. L.d. and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing, vol. 13, no. 5, pp. 2292–2303, 2013.
11. Masdari, Mohammad, Farbod Salehi, Marzie Jalali, and Moazam Bidaki. "A survey of PSO-based scheduling algorithms in cloud computing." Journal of Network and Systems Management 25, no. 1 (2017): 122-158.
12. Wang, Bei, and Jun Li. "Load balancing task scheduling based on Multi-Population Genetic Algorithm in cloud computing." In 2016 35th Chinese Control Conference (CCC), pp. 5261-5266. IEEE, 2016.
13. Agrawal, Mayur, Rishabh Bansal, Ankur Choudhary, and Arun Prakash Agrawal. "Hetrogenous Computing Task Scheduling Using Improved Harmony Search Optimization." In 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), pp. 11-15. IEEE, 2018.
14. Agarwal, Mohit, and Gur Mauj Saran Srivastava. "A cuckoo search algorithm-based task scheduling in cloud computing." In Advances in Computer and Computational Sciences, pp. 293-299. Springer, Singapore, 2018.
15. Sheikholeslami, Fereshteh, and Nima Jafari Navimipour. "Service allocation in the cloud environments using multi-objective particle swarm optimization algorithm based on crowding distance." Swarm and Evolutionary Computation 35 (2017): 53-64.
16. Zuo, Liyun, Lei Shu, Shoubin Dong, Chunsheng Zhu, and Takahiro Hara. "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing." Ieee Access 3 (2015): 2687-2699.
17. Madni, Syed Hamid Hussain, Muhammad Shafie Abd Latiff, and Javed Ali. "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds." Arabian Journal for Science and Engineering 44, no. 4 (2019): 3585-3602.
18. Larsen, P. Martin. "Industrial applications of fuzzy logic control." International Journal of Man-Machine Studies 12, no. 1 (1980): 3-10.
19. Bonissone, Piero P., Vivek Badami, Kenneth H. Chiang, Protap S. Khedkar, Kenneth W. Marcelle, and Michael J. Schutten. "Industrial applications of fuzzy logic at General Electric." Proceedings of the IEEE 83, no. 3 (1995): 450-465.
20. Bai, Ying, Hanqi Zhuang, and Dali Wang, eds. Advanced fuzzy logic technologies in industrial applications. Springer Science & Business Media, 2007.
21. Nine, Md SQ Zulkar, Md Abul Kalam Azad, Saad Abdullah, and Rashedur M. Rahman. "Fuzzy logic based dynamic load balancing in virtualized data centers." In 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1-7. IEEE, 2013.
22. Adami, Davide, Andrea Gabbrielli, Stefano Giordano, Michele Pagano, and Giuseppe Portaluri. "A fuzzy logic approach for resources allocation in cloud data center." In 2015 IEEE Globecom Workshops (GC Wkshps), pp. 1-6. IEEE, 2015.
23. Kong, Xiangzhen, Chuang Lin, Yixin Jiang, Wei Yan, and Xiaowen Chu. "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction." Journal of network and Computer Applications 34, no. 4 (2011): 1068-1077.
24. Javanmardi, Saeed, Mohammad Shojafar, Danilo Amendola, Nicola Cordeschi, Hongbo Liu, and Ajith Abraham. "Hybrid job scheduling algorithm for cloud computing environment." In Proceedings of the fifth international conference on innovations in bio-inspired computing and applications IBICA 2014, pp. 43-52. Springer, Cham, 2014.
25. Alla, Hicham Ben, Said Ben Alla, Abdellah Ezzati, and Ahmed Mouhsen. "A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing." In International Symposium on Ubiquitous Networking, pp. 205-217. Springer, Singapore, 2016.
26. Shojafar, Mohammad, Saeed Javanmardi, Saeid Abolfazli, and Nicola Cordeschi. "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method." Cluster Computing 18, no. 2 (2015): 829-844.
27. Zadeh, Lotfi A. "Fuzzy sets." Information and control 8, no. 3 (1965): 338-353.

28. Zadeh, Lotfi Asker. "The concept of a linguistic variable and its application to approximate reasoning—I." Information sciences 8, no. 3 (1975): 199-249.
29. Zhao, Jin, and Bimal K. Bose. "Evaluation of membership functions for fuzzy logic controlled induction motor drive." In IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02, vol. 1, pp. 229-234. IEEE, 2002.
30. Van Leekwijck, Werner, and Etienne E. Kerre. "Defuzzification: criteria and classification." Fuzzy sets and systems 108, no. 2 (1999): 159-178.
31. Mendel, Jerry M. "General type-2 fuzzy logic systems made simple: a tutorial." IEEE Transactions on Fuzzy Systems 22, no. 5 (2013): 1162-1182.
32. Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience 41, no. 1 (2011): 23-50.
33. Tufan Kumbasar (2020). Interval Type-2 Fuzzy Logic System Toolbox (https://www.github.com/kumbasart/type-2-fuzzy-logic-systems-matlab-toolbox), GitHub. Retrieved January 2, 2020.
34. X. Fan, W.-D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in Proceedings of the 34th Annual International Symposium on Computer Architecture, ser. ISCA ’07. New York, NY, USA: ACM, 2007, pp. 13–23.
35. Meisner, David, and Thomas F. Wenisch. "Peak power modeling for data center servers with switched-mode power supplies." In Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design, pp. 319-324. ACM, 2010.
36. Zhang, Zhiming, Chan-Ching Hsu, and Morris Chang. "Cool cloud: A practical dynamic virtual machine placement framework for energy aware data centers." In 2015 IEEE 8th International Conference on Cloud Computing, pp. 758-765. IEEE, 2015.
37. Raj, VK Mohan, and R. Shriram. "A study on server sleep state transition to reduce power consumption in a virtualized server cluster environment." In 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), pp. 1-6. IEEE, 2012.

AUTHORS PROFILE

Rashmi Singh Lodhi, received her B.E. Degree in Computer Science and Engineering from Smart Ashok Technological Institute (SATI), Vidisha (M.P.), India, in 2004, she also completed her Master Degree in Information Technology from SATI in 2012. She is currently pursuing PhD in Computer Science and Engineering, from Maulana Azad National Institute of Technology (MANIT), Bhopal, India. Her interest include Cloud Computing, Distributed Computing, Grid Computing, Machine Learning, and Fuzzy Logic Systems.

Dr. R.K. Pateriya, is presently working as Associate Professor in the Dept. of Computer Science & Engineering in Maulana Azad National Institute of Technology (MANIT), Bhopal, India. He has published more than 45 papers in international, national journals and conferences. He has already guided many PhD scholars and MTech students, and many more are continuing their work under his guidance. His interest area includes Cloud Computing, Computer Networks and Information Security.