A Fuzzy Logic Based Approach in Choosing the Appropriate Physical Machines for Live Virtual Machines Migration in Cloud Computing

S. R. Hosseini [1], S. Adabi [2], R. Tavoli [3]
[1]*Young Researchers and Elite Club, Chalus Branch, Islamic Azad University, Chalus, Iran*
[2]*Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran*
[3] *Department of Mathematics, Chalus Branch, Islamic Azad University, Chalus, Iran*
roudabeh.hosseini@gmail.com

***Abstract*— Migration of Virtual Machine (VM) has become a critical issue in modern data centers that are working based on virtualization. Among VM Migration challenges, choosing an appropriate Physical Machine (PM) is an important issue. To choose a place for VM several parameters, such as physical topology, migration duration, power consumption, service continuity, and price must be considered. Finding a near optimal place for VM migration that trades-off between some or all of these features is a challenging problem. In this paper, we propose three aspects for ranking potential destination PMs to find the most appropriate PM as VM host. In the first aspect, PMs are ranked in terms of servicing condition using Fuzzy logic technique according to three parameters: workload, performance efficiency and availability. In the second aspect, PMs are ranked in terms of power consuming condition using Fuzzy logic technique according to power, temperature efficiency and power efficiency metrics. In the third aspect, the output of two fuzzy logic engines with communication cost metric is used as the third fuzzy logic engine inputs that rank PMs. The proposed technique has been compared with AppAware algorithm in terms of communication cost and performance efficiency. Experimental results demonstrate that the proposed technique has appropriate improvement in these metrics and outperforms AppAware algorithm.**

***Index Terms*— Fuzzy Logic, Network topology, Performance evaluation, Servers.**

I. Introduction

Virtualization technology has changed the design and operation of data center in recent years [1] and there has been a strong tendency to develop data centers that have application with low dependency with underlying infrastructure and can easily share resources. The basis of cloud computing is virtualization which can manage services easily and reduce energy costs in data centers [2]. Generally, there are two types of migration including the live migration and non-live migration. The most common type of VM migration is the live VM migration. In this migration, VMs move from one PM to another PM while VMs are running [3]. In the non-live migration, the VM is stopped from working in the source of PM and the VM starts working in the destination of PM from the last stage before migrating, when all the processor state, memory pages and disk data are received [3].

Existing decision model for choosing VM migration destination is useful but it has limited application. These models cannot deal with uncertainty and ambiguities, which cannot be driven by crisp values. Using crisp values is a significant problem in their decision making process.

In this paper we propose a method for choosing an appropriate PM focusing on seven important PM ranking metrics including: performance efficiency, power efficiency, communication cost between VMs, power consumption, workload, temperature efficiency and availability. These metrics were not considered in the previous methods, such as Sandpiper, Mirsal and AppAware. In the proposed method, a combination of the mentioned metrics for a logical evaluation of each PM state with the aid of Fuzzy logic method is conducted. This is based on the consideration that the Fuzzy logic method is a proper method for decision-making in uncertainly conditions. As a result, PMs are prioritized with the mentioned metrics and the suitability of each PM is determined.

The remainder of the paper is organized as follows. Section 2 presents a brief overview of previous works. Using the Hierarchal Fuzzy logic method for choosing a near optimal PM for VM migration, the proposed method and algorithm were described in detail in section 3. Section 4 defines the parameters for the simulation. We present the experimental results and analysis of these experiments in section 5, while the conclusions and future works were described in section 6.

II. Literature Review

In this section, we present a brief overview of some of the state-of-the-art approaches on VM migration. Many researches in context of VM migration have concentrated on ways to improve costs, performance, efficiency and flexibility. To detect overloaded server, a method has been proposed in [4] that using TOPSIS algorithm to relocate VMs between clusters. The proposed method consist a control unit that receives PM information and sorts PMs from the most rank to the least rank. The control unit checks the ranks and if it is higher than a predefined threshold, the server is saturated and migration must be done. In the next step, hotspot VMs is

determined with some parameters. To avoid transferring large data and reduce cost, the VMs with the lowest RAM utilization are chosen for migration. By migrating hotspot from the overloaded PM to under loaded PM, the load distribution is conducted and the response time is improved [4]. For eliminating hotspots in the data center, Sandpiper algorithm was proposed [5]. Sandpiper provides two monitoring strategies for collecting statistics, namely the black-box and grey-box strategy. The black-box strategy collects statistic from the outside of the VM. The grey-box approach accesses the OS-level statistics, resource usage of VMs and application resident within each VM and migrates overloaded VMs to less loaded servers that can satisfy the need of VM. Optimization bandwidth usage is a primary goal in the data centers [6]. In this context AppAware is an evaluating approach for selecting the most appropriate PM to host VM in terms of minimizing the traffic of data center network. The main aim of AppAware is to put the dependent VMs in close proximity to reduce total traffic in data center physical network. This algorithm takes into account inter-VM dependencies and underlying network topology into host selection. AppAware migrates an overloaded VM to a PM based on a migration impact factor and required resources [7]. Unbalanced temperature in data centers result higher cooling cost [1]. In [8] a multi-objective approach to virtual machine management in data centers was proposed and this approach improves VM performance and temperature efficiency and reduces power consuming [8]. In [9] control architecture for VM migration to trade-off between performance and cost and power is proposed. Tao et al. [10] proposed a triple-objective comprehensive model for solving dynamic migration of VMs that uses a binary graph matching-based bucket-code learning algorithm (BGM-BLA) for evaluating the candidate solutions. The goal of the model is to reduce the energy consumption and communication cost, while reducing migration cost.

To the best of the authors' knowledge, this is the first work that considers the effect of seven important parameters in VM migration approach, which include performance efficiency, power efficiency communication cost between VMs, power consumption, workload, temperature efficiency and availability together.

III. The Proposed Approach

In the proposed approach, we use fuzzy logic method to select a near optimal PM as VMs migration destination because this method efficiently uses human knowledge in vague and inaccurate conditions. PM ranking parameters that make numerical value of PM rank (i.e., PM_rank) are vague and uncertain to be expressed by crisp mathematical models. It is, however, often possible to describe the PM_rank by means of building fuzzy models.

There is a direct relation between the number of fuzzy sets of input parameters of the system and the size of the fuzzy knowledge base [11]. As the number of fuzzy sets of input parameters increases, the number of rules increases exponentially. Obviously by considering seven PM ranking parameters as the number of inputs into the fuzzy system we will face the mentioned problem. In this case, it is recommended to limit the number of inputs used by the system. To face this, we use a hierarchal fuzzy logic structure for such fuzzy logic systems, which leads to the reduction of computational time and maintenance of the systems robustness and efficiency.

To calculate the PM_rank, the metrics are classified based on tree logics. As shown in Figure 1, it includes:

1. calculating PM_rank based on serving conditions,
2. calculating PM_rank based on communication cost and
3. calculating PM_rank based on power consuming

It also includes three fuzzy inference engines designed in the form of Hierarchical Fuzzy inference engine in order to enhance speed of operation and permit more criteria for PMs ranking.

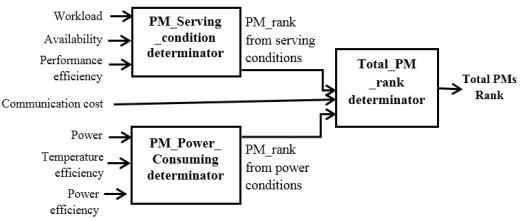

Figure 1: The proposed hierarchical fuzzy inference engine

The proposed approach has two layers. The first layer composes of two types of fuzzy decision controller: Fuzzy PM Serving Condition determinator and Fuzzy PM Power. Consuming determinator is designed to determine the numerical values of PM rank based on serving conditions and PM rank based on power consumption respectively. The second layer is composed of a fuzzy decision controller, Total_PM_Rank, determinator which is designed to determine the total values of PM_rank based on a) the output of Fuzzy PM_Serving_Condition determinator, b) the output of Fuzzy PM_Power_Consuming determinator and c) communication cost.

A fuzzy decision controller is composed of:

1. input and output variables, which are determined based on the knowledge of experts;
2. a fuzzification interface (FI), which has the effect of transforming crisp data into fuzzy sets;
3. a fuzzy rule base (RB), in which a set of fuzzy rules is determined;
4. a fuzzy negotiation decision making logic (DML), that uses them together with the RB to make inference by means of a reasoning method; and
5. a defuzzification interface (DFI), that translates the fuzzy rule action, thus obtaining a real action using a defuzzification method.

Following the five components of each part of PM Serving Condition determinator and PM Power Consuming determinator of the first layer of proposed approach and Total PM Rank determinator of the second layer, the proposed approach is discussed in the following section.

A. PM Serving Condition determinator

The PM Serving Condition determinator ranks each PM in terms of PMs servicing condition. PMs with the highest rank in terms of servicing condition, is the most proper destination for VMs migration. The inputs, fuzzy rules and output of this fuzzy engine are discussed below.

The inputs of PM Serving Condition determinator

The inputs of this fuzzy inference engine comprise three metrics, which are the workload, availability and performance efficiency. It also has a triangular membership function as illustrated in Figure 2. In this respect, we used mamdani type fuzzy logic system to design this fuzzy logic engine. The inputs are discussed in the following.

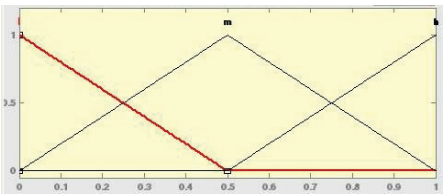

Figure 2: Triangular membership function for ranking the PMs

Availability is defined as the percentage of time which a customer can access to the service" [12], for selecting a PM as VM host. It is important to know that the PM is available on that time or not. In other words, it involves the investigation whether a candidate's destination of PM has sufficient capacity to support a new VM and can satisfy its requirements [1]. Availability of *i'th* PM is determined as Equation (1) [1]:

$$PM_i\_\text{Availability} = \frac{T_t \; - \; T_n}{T_t} \tag{1}$$

where T_t is the total service time and T_n is the total time for which service is not available. According to (1), when ^{PM_i}_Availability tends to one the availability of a PM increased. Obviously, a PM with the highest ^{PM_i}_Availability value is the best destination for a migrating VM in terms of PM availability.

Workload is a set of jobs known as *W* [13]. The PMs unbalanced workload leads to fierce competition to resources in the PMs with heavy workload. For PMs with low workload there is low competition, which will decrease the performance of the VMs running on the PM with heavy workload. Therefore, VMs should be distributed to less loaded PMs to avoid overloading of some PMs. The workload of *i'th* PMs is defined by Equation (2) [14].

$$\begin{cases} PM_i \quad \_Workload = W_{cpu} \times P_{cpu} + W_{mem} \times P_{mem} + W_{network} \times P_{network} \\ W_{cpu} + W_{mem} + W_{network} = 1, \quad 0 \le W_{cpu}, W_{mem}, W_{network} \le 1 \end{cases} \tag{2}$$

where W_{cpu} is the weight of the CPU usage, P_{cpu} is the CPU usage, W_{mem} is the weight of the memory usage, P_{mem} is the memory usage, $W_{network}$ is the weight of the network usage and $P_{network}$ is the network usage. In equation (2), W_{cpu}, W_{mem} and $W_{network}$ can dynamically be adjusted [14]. If the result value is closer to one, PM has heavy workload and if the result value is closer to zero it means the PM has better state and is less loaded.

Performance efficiency represents the amount of use of resource of different types. To avoid resource contention, the efficiency decreases rapidly when the usage of one or more of resources increase the maximum allowed [8]. VMs should migrate to PMs that have better performance efficiency. The performance efficiency of *i'th* PM is defined by Equation (3).

$$\begin{aligned} PM_i\_Eff_i(C) &= \min\left(PM_i\_Eff_i(CPU), PM_i\_Eff_i(IO), PM_i\_Eff_i(Net)\right) \\ PM_i\_Eff_i(CPU) &= 1 - \left(\frac{CPU_i - CPU_{low}}{CPU_{high} - CPU_{low}}\right)^m \\ PM_i\_Eff(IO) &= 1 - \left(\frac{IO_i - IO_{high}}{IO_{high} - IO_{low}}\right)^m \\ PM_i\_Eff(Net) &= 1 - \left(\frac{Net_i - Net_{high}}{Net_{high} - Net_{low}}\right)^m \end{aligned} \tag{3}$$

where CPU_i is the CPU usage (%), CPU_{low} is the CPU usage of idle PM (0%), CPU_{high} is the CPU usage of overloaded PM (100%), IO_i is the disk utilization (%), O_{low} is the IO usage of idle PM (0%), IO_{high} is IO the usage of overloaded PM (100%), Net_i is the network IO usage of PM i, Net_{high} is the highest network IO usage (20 M bytes/sec), Net_{low} is the lowest network IO usage (0) and m is the exponent (set to 3 in implementation) [8].

In Equation (3) if the result value is closer to one, PM is less efficient and if the result value is closer to zero, PM is more efficient

The output of PM Serving Condition determinator

The output of this fuzzy inference engine is used for ranking PMs in terms of servicing condition. As can be seen in Figure 2, the triangular membership function is used for the output variable. The weighted average method is used for the defuzzification and computing the clear output value. If the PM_rank from the PM_Serving_condition perspective is close one, the chance of selecting a PM as a destination for a migrating VM from servicing condition perspective should be increased.

The rule set of PM_Serving_Condition determinator

Table 1 represents the rule set of PM_Serving_Condition determinator, which is provided by expert knowledge. These rules define this fuzzy inference engine behavior.

Table 1
The rule set of PM_Serving_Condition determinator

Input metrics			Output
Performance efficiency	Availability	Workload	PMs servicing condition
H ∨ M	L	L	L
L	L	L	M
M ∨ L	M	L	M
H	H ∨ M	L	L
M	H	L	M
L	H	L	H
M ∨ H	L	M	L
H	M	M	L
M	M	M	M
L	M ∨ H	M	M
H	H	M	L
M	H	M	L
H	L	H	L
L	L	H ∨ M	L
H ∨ M	M ∨ L	H	L
M ∨ L	M	H	M
L ∨ M ∨ H	H	H	L

B. PM_Power_Consuming determinator

The PM_Power_Consuming determinator ranks each PM in terms of PMs power consuming. The PM with the highest rank in terms of power consuming is the best destination for a migrating VM. The inputs, fuzzy rules and output of this fuzzy engine are discussed below.

The Inputs of PM_Power_Consuming determinator

The inputs of this fuzzy inference engine consists of three metric, namely the power efficiency, temperature efficiency and power utility, and it has a triangular membership function as illustrated in Figure 2. We used mamdani type fuzzy logic system to design this fuzzy logic engine. The inputs are discussed in the following.

The *power* utility is a function of the resource utilization by a PM in a time interval [15]. A PM with lower power utilization should be selected as a VM host. The Power utility of *i'th* PM is shown in Equation (4) [15]:

$$PM_i\_power = P_{idle} + P_{cpu}\frac{U_{cpu}}{C_{cpu}} + P_{disk}\frac{U_{disk}}{C_{disk}} \quad (4)$$

where P_{cpu} is the maximum dynamic power usage of the CPU, U_{CPU} is the CPU consumption as a percentage of the total CPU capacity [%], U_{disk} is the disk usage as a percentage of the total bandwidth capacity [%], C_{disk} is the total disk bandwidth capacity and P_{idle} is the power utilization by a PM when it is idle [17]. According to (4) if PM_i_Power value close to one, the power utilization of the PM increases. Obviously a PM with the highest PM_i_Power value is the best destination for a migrating VM in term of power utilization.

With the increase of PM workload, the CPU temperature is raised. By increasing the CPU temperature, the performance of PM is affected and lead to an unbalanced heat in the data center, resulting in the increase of the data center cooling cost. To reduce the data center cooling cost and the power consumption, selecting a PM with a lower temperature as VM host is momentous. Temperature efficiency of *i'th* PM is calculated by Equation (5) [8]:

$$PM_i\_Eff_i(T) = 1 - \left(\frac{T_i - T_{low}}{T_{high} - T_{low}}\right)^m \quad (5)$$

where *Ti* is the temperature of PM *i*, T_{Low} is the temperature for an idle PM (15°C), T_{high} is the temperature for overloaded PM (55°C) and *m* is the degree set to 3 in implementation [8].
In Equation (5), if the result value is close to one, PM is less suitable and if the result value is close to zero, the PM is more suitable in term of temperature efficiency.

Power efficiency represents how much useful work is done by the consumed power. By using the linear power model, Power efficiency of *i'th* PM is calculated by Equation (6) [8]:

$$PM_i\_Eff_i(P) = \frac{Workload_i}{Power_i} \quad (6)$$

The power efficiency increases with the utilization of the CPU and reaches the highest point when CPU usage is 100% [8].

The Output of PM_Power_Consuming determinator

The output of this fuzzy inference engine is used for ranking the PMs in terms of power consuming. As can be seen in Figure 2, the triangular membership function is used for the output variable. For defuzzification and computing the clear output value, the weighted average method is used. If the result value is closer to one, PM is more appropriate in terms of power consuming state.

The rule set of PM_Power_Consuming determinator

Table 2 illustrates the rule set for PM_Serving_Condition determinator, which is determined based on knowledge of experts.

Table 2
The rule set of PM_Serving_Condition determinator

Input metrics			Output
Temperature efficiency	Power efficiency	Power	PMs Power consuming condition
L ∨ M	L	L	L
H	L	L	M
L	M	L	L
M	M	L	M
H	M	L	H
L	H	L	M
M ∨ H	H	L	H
L	L	M	L
H ∨ M	L	M	M
L ∨ M	M ∨ H	M	M
H	M	M	M
H	H	M	H
L ∨ M ∨ H	L	H	L
L	M ∨ H	H	L
H ∨ M	M ∨ H	H	M

C. Total_PM_Rank determinator

The Total_PM_Rank determinator ranks each PM to find a near_optimal destination for a migrating VM. PM with the highest PM_rank will be chosen as a destination for a migrating VM.

The inputs of Total_PM_Rank determinator

The inputs of Total_PM_Rank determinator are:

1. output of Fuzzy PM_Serving_condition determinator
2. output of Fuzzy PM_Power_Consuming determinator
3. communication cost

This fuzzy inference engine has a triangular membership function, as shown in Figure 2 and mamdani type fuzzy logic system is used for designing of this fuzzy logic engine. The communication cost parameter is discussed below. The network *communication cost* is the time taken to communicate and swap data between VM_i and VM_j [16]. By moving the dependent VMs, which exchange a large volume of network traffic closer to each other, the network communication cost will be reduced. The VM communication cost in VL2 and Tree topology are determined in Equations (7) and (8) [18] respectively:

$$C_{ij}^{VL2} = \begin{cases} 0 & if\ i = j \\ 1 & if \left\lfloor \frac{i}{p_0} \right\rfloor = \left\lfloor \frac{j}{p_0} \right\rfloor \\ 5 & if \left\lfloor \frac{i}{p_0} \right\rfloor \neq \left\lfloor \frac{j}{p_0} \right\rfloor \end{cases} \quad (7)$$

$$C_{ij}^{Tree} = \begin{cases} 0 & if\ i = j \\ 1 & if \left\lfloor \frac{i}{p_0} \right\rfloor = \left\lfloor \frac{j}{p_0} \right\rfloor \\ 3 & if \left\lfloor \frac{i}{p_0} \right\rfloor \neq \left\lfloor \frac{j}{p_0} \right\rfloor \ \cap \ \left\lfloor \frac{i}{p_0 p_1} \right\rfloor = \left\lfloor \frac{j}{p_0 p_1} \right\rfloor \\ 5 & if \left\lfloor \frac{i}{p_0 p_1} \right\rfloor \neq \left\lfloor \frac{j}{p_0 p_1} \right\rfloor \end{cases} \quad (8)$$

where P_0 is the fan-out of the access switch and p_1 is the fan-out of the aggregation switch [18]. One of advantages of VL2 is that VL2 can be easily implemented with low cost [17]. In VL2 Topology, the cost is a function of fan-out of the access switch (*P0*) and can be calculated with Equation (7). In equation (8) the cost between two VMs is a function of access switches (*p0*) fan-out as well as the fan-out of the aggregation ones (*p1*) [18]. After the normalization process, if the result value is closer to one, PM has more communication cost and if the result value is closer to zero, the PM is more suitable and has less communication cost.

The output of Total_PM_Rank determinator

This fuzzy inference engine ranks the PMs as VMs migration destination in the output. The membership function is defined according to Figure 2. If the result value is closer to one, it means that PM is more appropriate as a VM migration host.

The rule set of Total_PM_Rank determinator

Table 3 illustrates the rule set for Total_PM_Rank determinator, which is determined based on knowledge of experts.

Table3
The rule set of Total_PM_Rank determinator

Input metrics			Output
PM power consuming state	**Communication cost**	**PMs service state**	**PMs ranking**
L ∨ M	L	L	L
H	L ∨ M	L	M
L ∨ M	M ∨ H	L	L
H	H ∨ L	L ∨ M	L
L ∨ M	L	M	M
L ∨ M	M	M	L
H	M	M ∨ H	M
L	H	M	L
L	L	H	M
M ∨ H	L	H	H
M	M	H	M
L ∨ M ∨ H	H ∨ M	H	L

D. Proposed Algorithm

We proposed a fuzzy logic-based approach for VM migration to minimize the cost and improve performance efficiency. Application dependencies and network topology are considered in the decision making processes. The base procedure for the proposed approach is shown in Algorithm1. In the proposed algorithm, the set of physical machines in the data center is defined as $P = \{P_1, P_2, P_3, ..., P_m\}$, set of virtual machines is defined as $V = \{V_1, V_2, V_3, \ldots, V_n\}$ and the set of overloaded virtual machines is defined as $O = \{V_1, V_2, \ldots, V_k\}$, such that $O \subset V$. The algorithm takes as input the number of VMs and PMs. Creat_datacenter(m) function creates a datacenter with m PMs, which have tree or VL2 network topology. Creat_state(n,m) defines which V_i creat on PM_i and then V_i will be created with a predefined percentage of PM_i 's resources by creat_vm(n,m,pm,state) function. Next, we calculate the communication cost between PMs in datacenter topology by creat_dcost(m). We define Load(V_i) as the vector of CPU, memory and storage load requirements of virtual machine V_i. Capacity(P_i) is defined as the available capacity of physical machine P_i regarding its CPU, memory and storage. In the next step, we find overloaded VM regarding its CPU usage. PM that has CPU utilization more than the predefined threshold is detected and VMs, which have the most CPU usage is selected for migration. In next step, we obtain a dependency graph, G = (V, E), where V is the set of VM and E is the set of edges, defined as E = (Vi, Vj): Vi, Vj $V\varepsilon$ c, such that if Vi and Vj are dependent to each other, a communication takes places between them. W(V_i, V_j) is the traffic demand of each edge, which is directly coordinated based on the traffic transferred between V_i, V_j . The total communication weight TW(V_i) of all overloaded VMs incoming edges is computed, as $TW(Vi) = \sum_{\forall Vj \in V} W(vi, Vj)$. Overloaded VMs are sorted in

descending order of their TW(V_i), thus overloaded VM which has the most traffic will have the priority to migrate to the selected PM. The algorithm will try to place the dependent VMs close to each other in the data center topology. For all the overloaded VMs, these steps will be done:

Check all side constraints (CPU, memory, disk capacity limits), if all server side constraints are satisfied, PMs ranking process will start. For all PMs, the value of seven inputs metrics is calculated and normalized between [0,1], and then three proposed engine will be called to rank the PMs. PM with the highest rank will be chosen as VM migration Destination. The capacity of source and destination PM will be updated after all migration. The migration decision steps are repeated until a mapping has been identified for all overloaded virtual machines.

IV. Simulation

A testbed is developed in Matlab program to evaluate the performance of the proposed approach. The VMs migration is done among PMs of a data center. Two data center topologies named as VL2 and Tree are considered. We generate a range of scenarios. All the input parameters required for setting simulation testbed and their possible values are shown in Table 4. We run simulated experiments for small topologies for 300 scenarios and 288 scenarios for large ones.

Algorithm 1 Algorithm for virtual machine migration

INPUTS: number of PMs (m) , number of VMs (n)
pm=creat datacenter(m); % m is PMs number
state=creat_state(n,m);% n is VMs number
vm=creat vm(n,m,pm,state);
distance=creat dcost(m);% Network distance
for each PM P_k in P
for each VM Vi in O
Calculate each PM used resources based on it's VMs
Capacity(P_k) % available server side capacity on physical machine Pi regarding its CPU,memory And storage
End for
End for

Calculate Overloaded VMs O $\subset$ VM
[w,g]=creat_dependency(n,O); % Dependency graph G=(V,E) with weights W
For each VM Vi in O

$$\mathrm{TW(Vi)} = \sum_{\forall Vj \in V} W(vi, Vj)$$

End for
sort Vi $\in$ O in decreasing order of TW(Vi)
M $\rightarrow$ 0 //migration set
for each VM Vi in O
for each PM Pk in P
if check_server_constrant(Vi , PK)==false
continue;
Workload(Vi, Pk), Cool(Vi, Pk), eff (Vi, Pk)= min{ eff_CPU(Pk), eff io(Pk), eff net(Pk) }
Power(vi, Pk), Availablity(vi, Pk), Performance(vi, Pk),Power_eff(vi, Pk)

$$\mathrm{Cost(Vi, Pk)} = \sum_{\forall Vj \in D(vi)} Cost(Vi, \mathrm{Pk}, \mathrm{Vj}, \mathrm{C(vj)})$$

Fuzzy 1(i)= evalfis([workload(i); Availablity (i); performance(i)])
Fuzzy_2(i)= evalfis([power(i);cool(i)])
Fuzzy_3(i)= evalfis([Fuzzy_1 (i);cost(i); Fuzzy_2 (i)])
End if
End for

Table4
Simulation parameters

Simulation parameters	Quantity domain	
VMs dependencies	Uniforms distribution 1-3	
Fraction of overloaded VMs	0.8 , 0.4 , 0.2	
Architecture(PM)	Tree, VL2 [7]	
# of VMs	Large topology	Small topology
	20-240 [7]	5-12 [7]
# of PMs	100 [7]	7-10 [7]

V. Main Result

The proposed approach was compared with AppAware algorithm [7] in terms of communication cost and performance efficiency. In Figure 3, the proposed approach was compared with AppAware algorithm in terms of the average of performance efficiency, in two VL2 and Tree topologies with 240 VMs. The X-axis indicates the number of VMs and Y-axis and is assigned to the average of performance efficiency metric. As shown in Figure 3, the proposed approach demonstrates the performance efficiency improvement (based on Equation (4)) in both VL2 and Tree topologies. Table 5 represents the amount of performance efficiency improvement in comparison to the AppAware algorithm in tree topology, and Table 6 represents the amount of performance efficiency improvement in comparison to the AppAware algorithm in VL2 topology.

In Figure 4, the proposed approach was compared with AppAware algorithm in terms of the average of communication cost, in two VL2 and Tree topologies. The X-axis indicates the number of VMs and Y-axis is assigned to the average of the communication cost metric. As shown in Figure 4, it can be observed that the proposed algorithm outperforms the AppAware algorithm in terms of average communication cost in both the VL2 and Tree topologies and the amount of communication cost improvement in comparison to the AppAware algorithm in tree topology is represented in Table 7. Table 8 represents the amount of communication cost improvement in comparison to the AppAware algorithm in VL2 topology. On average, the proposed approach improved communication cost much more in VL2 topology in comparison to the tree topology. These results confirm that the consideration of the average combinatorial effects of the seven parameters (Performance efficiency, power efficiency, communication cost between VMs, Power consumption, Workload, Temperature efficiency and Availability) by the proposed system, the accuracy of proposed algorithm increases which leads to better results in both VL2 and Tree topologies.

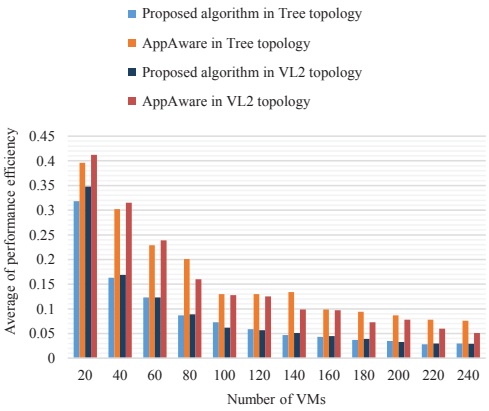

Figure 3: Comparison of proposed approach and AppAware in term of performance efficiency in large topology

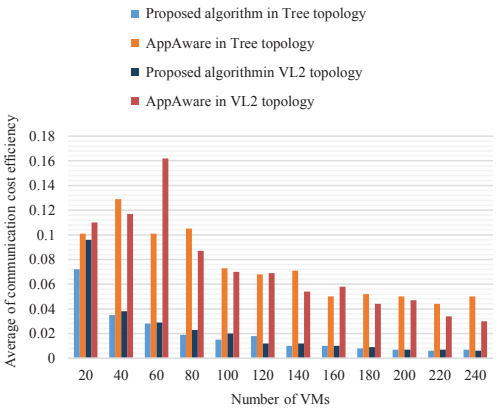

Figure 4: Comparison of proposed approach and AppAware in term of communication cost in large topology

Table 5
The amount of performance efficiency improvement in proposed approach in Tree topology in compare with AppAware

Number of VM	Proposed approach in Tree topology	AppAware in Tree topology	Amount of improvement
20	0.318	0.396	12.91
40	0.163	0.302	19.91
60	0.123	0.229	13.75
80	0.087	0.201	14.27
100	0.073	0.13	6.55
120	0.059	0.13	8.16
140	0.047	0.134	10.05
160	0.043	0.099	6.22
180	0.037	0.094	6.29
200	0.035	0.087	5.7
220	0.028	0.078	5.42
240	0.03	0.076	4.98

Table 6
The amount of performance efficiency improvement in proposed approach in VL2 topology in compare with AppAware

Number of VM	Proposed approach in VL2 topology	AppAware in VL2 topology	Amount of improvement
20	0.348	0.412	10.9
40	0.169	0.315	21.3
60	0.123	0.239	15.2
80	0.089	0.16	8.5
100	0.062	0.128	7.6
120	0.057	0.125	7.8
140	0.051	0.099	5.3
160	0.045	0.097	5.8
180	0.039	0.073	3.7
200	0.033	0.078	4.9
220	0.03	0.06	3.2
240	0.029	0.051	2.3

Table 7
The amount of communication cost improvement in proposed approach in Tree topology in compare with AppAware

Number of VM	Proposed approach in Tree topology	AppAware in Tree topology	Amount of improvement
20	0.072	0.101	3.2
40	0.035	0.129	10.8
60	0.028	0.101	8.1
80	0.019	0.105	9.6
100	0.015	0.073	6.3
120	0.018	0.068	5.4
140	0.01	0.071	6.6
160	0.01	0.05	4.2
180	0.008	0.052	4.6
200	0.007	0.05	4.5
220	0.006	0.044	4
240	0.007	0.05	4.5

Table 8
The amount of communication cost improvement in proposed approach in VL2 topology in compare with AppAware

Number of VM	Proposed approach in VL2 topology	AppAware in VL2 topology	Amount of improvement
20	0.096	0.11	1.6
40	0.038	0.117	8.9
60	0.029	0.162	15.9
80	0.023	0.087	7
100	0.02	0.07	5.4
120	0.012	0.069	6.1
140	0.012	0.054	4.4
160	0.01	0.058	5.1
180	0.009	0.044	3.7
200	0.007	0.047	4.2
220	0.007	0.034	2.8
240	0.006	0.03	2.5

VI. Conclusion

In this paper, we proposed a hierarchical fuzzy logic system for ranking PM as destination of VM migration aiming to reduce communication cost and improve the performance efficiency of PMs. To design this hierarchical fuzzy logic system, a classification is provided and a proposed metrics is classified in three aspects, namely the PM_Serving_Condition, PM_Power_Consuming and Total_PM_Rank. Using the hierarchal fuzzy logic system, which consider seven parameters (Performance efficiency, power efficiency, Communication cost between VMs, Power consumption, Workload, Temperature efficiency and Availability) and the significant role in ranking the potential destination of PMs for migrating VM together, the number of fuzzy rules in the system are reduced, thereby reducing the computational time (which is critical in cloud environment). The experimental results show that a proposed algorithm outperforms AppAware in terms of communication cost and performance efficiency due to the combined effects of the seven parameters. For future works, we can develop our work in VM migration among clouds, learn the proposed rules and develop better membership functions.

References

[1] R. Boutaba, Q. Zhang and M. F. Zhani, "Virtual Machine Migration in Cloud Computing Environments. Benefits, Challenges, and Approaches", *IGI Global*, 2013, pp. 383-408.

[2] R. Buyya , C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms. Vision, hype, and reality for delivering computing as the 5th utility", *Future Generation computer systems,* Vol.25, no.6, 2009, pp.599-616.

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, and A. Warfield, "Live migration of virtual machines", In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, Vol.2 , 2005, pp. 273-286.

[4] M. Tarighi, S. A. Motamedi and S. Sharifian, "A new model for virtual machine migration in virtualized cluster server based on Fuzzy Decision Making", 2010*, arXiv preprint arXiv.1002.3329*.

[5] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper. Black-box and gray-box resource management for virtual machines", *Computer Networks*, Vol.53, no.17, 2009, pp. 2923-2938.

[6] H. Liu, H. Jin, C. Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines", *Cluster computing*, Vol.16, no.2, 2013, pp.249-264.

[7] V. Shrivastava, P. Zerfos, K. W. Lee, H. Jamjoom, Y.H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers", In *Proceedings of IEEE INFOCOM*, 2011, pp. 66 -70.

[8] J. Xu, and J. Fortes, "A multi-objective approach to virtual machine management in datacenters", In *Proceedings of the 8th ACM international conference on Autonomic computing,* 2010, pp. 225–234.

[9] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral. Dynamically managing power, performance, and adaptation cost in cloud infrastructures", In *Proceedings of the IEEE International Conference on Distributed Computing Systems ICDCS,* 2010, pp.62–73.

[10] F. Tao, C.Li,T. Liao and Y.Laili, "BGM-BLA: a new algorithm for dynamic migration of virtual machines in cloud computing", 2015.

[11] M. Mohammadian., "Designing Customized Hierarchical Fuzzy Logic Systems For Modelling and Prediction", *4thAsian-Pacific Conference on Simulated Evolution and Learning*, pp.18-22, 2002, Singapore.

[12] S. K., Garg, S., Versteeg & R., Buyya, "A framework for ranking of cloud computing services", *Future Generation Computer Systems*, Vol.29, no.4, 2013, pp.1012-1023.

[13] A. Burkimsher, I. Bate and L. S. Indrusiak, "A survey of scheduling metrics and an improved ordering policy for list schedulers operanking on workloads with dependencies and a wide variation in execution times", *Future Generation Computer Systems*,Vol. 29, no.8, 2013, pp. 2009-2025.

[14] H. Mao, Z. Zhang, B. Zhao, L. Xiao, and L. Ruan, "Towards deploying elastic Hadoop in the cloud". In *Cyber-Enabled Distributed Computing and Knowledge Discovery CyberC, 2011 International Conference on* , 2011, October, pp. 476-482.

[15] D. Minarolli, and B. Freisleben, "Distributed Resource Allocation to Virtual Machines via Artificial Neural Networks", In *Parallel, Distributed and Network-Based Processing PDP, 2014 22nd Euromicro International Conference on,* 2014, pp. 490-499.

[16] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, "Starling. Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration", In *Parallel Processing ICPP, 2010 39th International Conference on ,* 2010, September, pp. 228-237.

[17] A. Hammadi, and L. Mhamdi, "A survey on architectures and energy efficiency in Data Center Networks", *Computer Communications*, Vol. 40, 2014, pp.1-21.

[18] X. Meng, V. Pappas and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement", In *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1-9.