A Fuzzy Predictable Load Balancing Approach in Cloud Computing

Fahimeh Ramezani, Jie Lu, Farookh Hussain
Decision Systems & e-Service Intelligence Lab, Centre for QCIS
School of Software, Faculty of Engineering and IT, University of Technology, Sydney
PO Box 123, Broadway NSW 2007 Australia
Fahimeh.Ramezani@student.uts.edu.au, {Jie.Lu, Farookh.Hussain}@uts.edu.au

Abstract— **Cloud computing is a new paradigm for hosting and delivering services on demand over the internet where users access services. It is an example of an ultimately virtualized system, and a natural evolution for data centers that employ automated systems management, workload balancing, and virtualization technologies. Live virtual machine (VM) migration is a technique to achieve load balancing in cloud environment by transferring an active overload VM from one physical host to another one without disrupting the VM. In this study, to eliminate whole VM migration in load balancing process, we propose a Fuzzy Predictable Load Balancing (FPLB) approach which confronts with the problem of overload VM, by assigning the extra tasks from overloaded VM to another similar VM instead of whole VM migration. In addition, we propose a Fuzzy Prediction Method (FPM) to predict VMs' migration time. This approach also contains a multi-objective optimization model to migrate these tasks to a new VM host. In proposed FPLB approach there is no need to pause VM during migration time. Furthermore, considering this fact that VM live migration contrast to tasks migration takes longer to complete and needs more idle capacity in host physical machine (PM), the proposed approach will significantly reduce time, idle memory and cost consumption.**

Keywords—cloud computing, load balancing, virtual machine migration, workload prediction.

I. INTRODUCTION

Cloud computing is a style of computing where flexible high-performance, pay-as-you-go, and on-demand offering service are delivered to external customers using Internet technologies. Cloud computing services are divided into three classes, according to the abstraction level of the capability provided and the service model of providers, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1]. The key underlying technology in cloud infrastructures is virtualization. Virtualization is a technique for hiding the physical characteristics of computing resources and allows servers and storage devices to be shared and utilization be increased. This simulated environment is called a virtual machine (VM) that is a software abstraction with the looks of a computer system's hardware (real machine) [2].

Cloud computing platform achieves dynamic balance between the servers applying virtualization technology for resource management. In virtualized cloud environment, applying online VM migration technology can achieve online remapping of VMs and physical resources, and dynamic whole system load balancing [3]. In particular, VM migration has been applied for flexible resource allocation or reallocation, by moving VM from one physical machine to another for stronger computation power, larger memory, fast communication capability, or energy savings [4].

Although a significant amount of research has been done to achieve system load balancing [4, 5], more improvement is still needed as most of these approaches tried to migrate VMs, when they became overload. In addition, in most of them predicting VMs' migration time is missed. Considering this fact that the whole VM migration takes much more times and cost in comparison with tasks migration, we propose a fuzzy load balancing method which migrates tasks from overloaded VMs instead of migration VMs to achieve load balancing in cloud environment. We also propose an algorithm to solve the problem of migrating these tasks to new VMs host which is a multi-objective problem subject to minimizing cost, minimizing execution and transferring time. To solve this problem we apply multi-objective genetic algorithm (MOGA). Furthermore, to accelerate load balancing process and reduce response time, we develop a fuzzy prediction method to predict VM workload situation and its migration time.

The rest of this paper is organized as follows. Section II presents the related works about VM migration techniques and VM workload prediction methods. Section III explains the basic concept of expert systems and neural networks, and genetic algorithm. In Section IV, we develop a Fuzzy Prediction Method (FPM) to predict VM migration time. In Section V, we propose a conceptual model and the algorithm of Fuzzy Predictable Load Balancing (FPLB) approach for solving the problem of overloaded VMs by optimal tasks migration from overloaded VMs. Our developed algorithm for solving multi-objective tasks scheduling problem and completing FPLB algorithm, is described in Section VI. The proposed approach is evaluated in Section VII. Finally we present the conclusion and future works in Section VIII.

II. RELATED WORKS

A. VM Migration Thechniques

Virtualization technique has improved utilization and system load balancing by enabling VM migration, and provided significant benefits for cloud computing [6]. Several methods have been developed to migrate a running instance of a VM from one physical host to another to optimize cloud utilization. Primary migration relies on process suspend and resume. Many systems [7, 8] just pause the VM and copy the state data, then resume the VM on the destination host. This forces the migrated application to stop until all the memory states have been transferred to the migration destination where it is resumed. These methods cause the application to become unavailable during the migration process. ZAP [9] could achieve lower downtime of the service by just transferring a process group, but it still uses stop-and-copy strategy. To reduce the migration downtime and move the VM between hosts in local area network without disrupting it, VMotion [10] and Xen [11] utilize pre-copy migration technique to perform live migration and support seamless process transfer. In pre-copy migration technique, VMs migrate by pre-copying the generated run-time memory state files from the original host to the migration destination host. If the rate for such a dirty memory generation is high, it may take a long time to accomplish live migration because a large amount of data needs to be transferred. In extreme cases, when dirty memory generation rate is faster than pre-copy speed, live migration will fail. Considering this fact, Jin et al. presented the basic pre-copy model of VM live migration and proposed an optimized algorithm to improve the performance of live migration by limiting the speed of changing memory through controlling the CPU scheduler of the VM monitor [4]. Lin et al. believe that most of the proposed methods for on-demand resource provisioning and allocation, focused on the optimization of allocating physical resources to their associated virtual resources and migrating VMs to achieve load balance and increase resource utilization. Unfortunately, these methods require the suspension of the executing cloud computing applications due to the mandatory shutdown of the associated VMs [12]. To overcome this drawback, they proposed a threshold-based dynamic resource allocation scheme for cloud computing that dynamically allocate the VMs among the cloud computing applications based on their load changes. In their proposed method, they determined when migration should be done but they did not specify the details of how the reallocation will occur.

A fundamental drawback of the most existing researches is that they consider complete VM migration to overcome overload VM and achieve system load balance. In addition, predicting VMs' workload situation and their migration time, are not considered in most of traditional load balancing approaches. In this paper, we propose a new Fuzzy Predictable Load Balancing (FPLB) approach which predicts VMs' workload situation, and transfers tasks from overloads VMs instead of whole VM migration to achieve system load balancing. The proposed approach not only eliminates the suspend and resume process during VM migration, but also omits pre-copy mechanism and producing dirty memory in live VM migration.

B. Workload Prediction Methods

Resource provisioning in compute clouds often requires an estimate of the required capacity for VMs. The estimated VM size is essential for allocating resources commensurate with demand [13]. Meng et al. proposed an algorithm for estimating the aggregate size of multiplexed VMs. They decoupled VM workload into regular and irregular fluctuating components. To forecast regular workload, they simply assumed that the regular patterns will preserve in the future, e.g., a steadily increasing trend keeps increasing at the same rate. On the other hand, for forecasting irregular workload, they performed a time series forecasting technique based on historic workload patterns [13]. Nagothu et al. proposed a new method for load prediction. They separated load prediction into linear and non-linear prediction algorithms categories. They believed a linear prediction algorithm can either involve 1-Dim observation sequences or d-Dim observation space signals [14].

Most of the existing researches in this area have applied prediction methods such as neural networks and linear regression to forecast VMs workload in cloud environment. These prediction methods predict future workload applying previous workload patterns in time slot *t*. In IaaS where VMs are assigned to customers; VMs' workload is affected by customers' behavior and decisions, and significantly changes in seconds. Therefore, upcoming VMs' workload in cloud environment could be independent from their previous workload pattern. To overcome this problem we propose a workload prediction method applying a neural network and an expert system with the specific parameters that control and monitor recent changes in VMs' workload pattern.

III. BACKGROUND

A. Expert Systems

The basic idea behind expert systems (ES) is simply that expertise, which is the vast body of task-specific knowledge, is transferred from a human to a computer. This knowledge is then stored in the computer and users call upon the computer for specific advice as needed. The computer can make inferences and arrive at a specific conclusion. Then like a human consultant, it gives advices and explains, if necessary, the logic behind the advice. A rule-based ES is defined as one, which contains information obtained from a human expert, and represents that information in the form of rules, such as IF–THEN. The rule can then be used to perform operations on data to inference in order to reach appropriate conclusion. These inferences are essentially a computer program that provides a methodology for reasoning about information in the rule base or knowledge base, and for formulating conclusions [15, 16].

B. Neural Networks

A neural network (NN) consists of a number of layers: the input layer has a number of input neurons $X =$

$\{x_1, \ldots, x_m\}$; the output layer has one or more output neurons $O = \{o_1, \ldots, o_n\}$ and several hidden layers with hidden neurons $H = \{h_1, \ldots, h_l\}$ in between. In a fully-connected NN, the neurons at each layer are connected to the neurons of the next layer; these connections are known as synapses. Each synapse is associated with a weight, which is to be determined during training. During the training phase, the NN is fed with input vectors and random weights are assigned to the synapses. After presentation of each input vector, the network generates a predicted output $\hat{y}$. The generated output is then compared with the actual output *y*; the difference between the two is known as the error term which is then used as a feedback to correct the synaptic weights of the network. The training of the NN continues until a specific criterion is met, e.g. the sum of squared errors falls below a certain threshold [17].

C. A Multi Objective Genetic Algorithm

A multi-objective genetic algorithm (MOGA) is concerned with the minimization of multiple objective functions that are subject to a set of constraints. In MOGA for solving multi-objective optimization problems, first initial population whose scale is *N* is generated randomly. The first generation child population is gained through non-dominated sorting [18] and basic operations such as selection, crossover and mutation. Then, from the second generation on, the parent population and the child population will be merged and sorted based on fast non-dominated. Calculate crowding distance among individuals on each non-dominated layer. According to non-dominant relationship and crowding distance among individuals, select the appropriate individuals to form a new parent population. Finally, new child population is generated through basic operations of genetic algorithm and so on, until the conditions of the process end can be met [19].

IV. A Fuzzy Prediction Method for Determining the VM Migration Time

Considering this fact that future VMs' workload could be independent from their previous workload pattern, we propose a Fuzzy Prediction Method (FPM) that not only applies neural network to predict workload patterns in VMs, but also applies an expert system to control near future changes in workload patterns for every VM, and determine the time that VMs will be overloaded and need to be migrated. To design the FPM, we first determine the conditions which lead to a VM to be overloaded. Then the ES rules are extracted from determined conditions. We run FPM every 5 minutes and control VM workload situation for the past 2 minutes.

A. VM Migration Conditions

If VM_{wc} be the VM workload capacity and $VM_{et}(t)$ be the number of executing tasks in the VM as a time series in time slot *T*, then the VM will be overloaded at the time *x* when:

$$\lim_{t \to x} VM_{et}(t) = VM_{wc} \quad (1)$$

and Equation 1 could have an answer for variable *x* during next 2 minutes, if following conditions be satisfied where *ct* is current time, and $t_i \in T = \{ct - (120\ seconds), ct\}$:

Condition 1: Time series $VM_{et}(t)$ rises one or more times during *T*. It means:

$$VM_{et}(t_i) < VM_{et}(t_i + 1)\ , \exists\ \{t_i\} = T^* \subset T \quad (2)$$

It means: $VM'_{et}(t_i) \geq 0\ , \exists\ \{t_i\} = T^* \subset T$

Condition 2: There would be overloading time for the VM if: $VM_{wc} - \max_{t_i \in T} VM_{et}(t_i) < 2$ (3)

Condition 3: Rao (2011) proposed a metric of Productivity Index (PI) and use it to measure the system processing capability. He defined PI as:

$$PI\ (t) = \frac{\mathrm{CW(t)}}{\mathrm{CC\ (t)}} \quad (4)$$

where $CW(t)$ is the amount of completed work and $CC(t)$ is the amount of resource (CPU) consumed during the time slot *t*. An overloaded system means that its *cost* keeps increasing but with stagnated or compromised *yield*. Virtual machine will be overloaded if PI begins to drop. Although Rao believes that for online identification, the single PI metric is not enough to identify system state because any change of *PI* can be either due to the system capacity or the input load change [20]. Considering this fact, to control VM's workload situation during *T,* We determine following conditions that show PI drops during T:

$$PI(t_i) > PI(t_i + 1)\ , \exists\ \{t_i\} = T^+ \subset T \quad (5)$$

It means:

$$PI'(t_i) \leq 0\ , \exists\ \{t_i\} = T^+ \subset T$$

In addition, the time series $PI(t)$ has a decreasing trend:

$$PI(ct - 120s) > PI(ct) \quad (6)$$

Condition 4: Time series $VM_{et}(t)$ has a raising trend:

$$VM_{et}(ct - 120\ s) < VM_{et}(ct) \quad (7)$$

B. The Variables of Fuzzy Prediction Method

The *input* variables of proposed FPM are defined as follows: the number of executing tasks in the VM at current time as $VM_{et}(ct)$, the maximum number of executing tasks in VM during *T* as $\max_{t_i \in T} VM_{et}(t_i)$, the change in number of executing tasks in the VM at random time $t_r \in T^*$ as $VM_{et}(t_r + 1) - VM_{et}(t_r)$, the $VM_{et}(ct) - VM_{et}(ct - 120\ s)$ that shows $\mathrm{VM_{et}(t)}$ raised during *T*, the amount of completed tasks in the VM till current time as $CW(ct)$, the amount of CPU consumed till current time as $CC(ct)$, the change in productivity index at random time $t_r \in T^+$ as $PI(t_r) - PI(t_r + 1)$, the $PI(ct - 120s) - PI(ct)$ that shows PI(t) decreased during *T*. In addition, in this approach a neural network model is trained, given the previous historic workload patterns (training data set) to predict VM workload pattern. The neural network prediction results $NN_R(t)$ (desired output of neural network) will be applied by proposed FPM as an input variable. The FPM's

output variable is defined as $VM_{WL}(t)$ that denotes the predicted VM workload situation in near future. The FPM's variables membership functions are defined as follow:

TABLE I. FUZZIFICATION OF $VM_{et}(t)$ AND $\max_{t_i \in T} VM_{et}(t_i)$: INPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
VL	Very Low	0	$0.25 * VM_{wc}$
L	Low	$0.25 * VM_{wc}$	$0, 0.5 * VM_{wc}$
M	Medium	$0.5 * VM_{wc}$	$0.25 * VM_{wc}, 0.75 * VM_{wc}$
H	High	$0.75 * VM_{wc}$	$0.5 * VM_{wc}, 1 * VM_{wc}$
VH	Very High	$1 * VM_{wc}$	$0.75 * VM_{wc}$
Universe of discourse: (0, VM_{wc})			

TABLE II. FUZZIFICATION OF $VM_{et}(t_i + 1) - VM_{et}(t_i), t_i \in T^*$ AND $VM_{et}(ct - 120s) - VM_{et}(ct)$: INPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
VL	Very Low	-1	-0.5
L	Low	-0.5	-1,0
M	Medium	0	-0.5,0.5
H	High	0.5	0,1
VH	Very High	1	0.5
Universe of discourse: (-1,1)			

TABLE III. FUZZIFICATION OF $CW(t)$: INPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
VL	Very Low	0	$0.25 * CU$
L	Low	$0.25 * CU$	$0, 0.5 * CU$
M	Medium	$0.5 * CU$	$0.25 * CU, 0.75 * CU$
H	High	$0.75 * CU$	$0.5 * CU, 1 * CU$
VH	Very High	$1 * CU$	$0.75 * CU$
Universe of discourse: (0, 100% Virtual CPUs utilization=CU)			

Note: Virtual CPUs determines how many physical CPUs can be used by a VM. The number of virtual CPUs together with the scheduler credit determine the total CPU resource allocated to a VM [20].

TABLE IV. FUZZIFICATION OF $CC(t)$: INPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
VL	Very Low	0	$0.3 * N$
L	Low	$0.25 * N$	$0, 0.5 * N$
M	Medium	$0.5 * N$	$0.25 * N, 0.75 * N$
H	High	$0.75 * N$	$0.5 * N, 1 * N$
VH	Very High	$1 * N$	$0.75 * N$
Universe of discourse: (0, Total number of executed tasks during T=N)			

TABLE V. FUZZIFICATION OF $PI(t_i + 1) - PI(t_i), t_i \in T^+$ AND $PI(ct - 120s) - PI(ct)$: INPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
VL	Very Low	-1	-0.5
L	Low	-0.5	-1,0
M	Medium	0	-0.5,0.5
H	High	0.5	0,1
VH	Very High	1	0.5
Universe of discourse: (-1, 1)			

TABLE VI. FUZZIFICATION OF $NN_R(t)$: INPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
U	Underload	0,0.3	0.7
O	Overload	0.7,1	0.3
Universe of discourse: (0,1)			

TABLE VII. FUZZIFICATION OF $VM_{WL}(t)$: OUTPUT

Set	Linguistic term	α level cuts	
		1-level cut	0-level cut
U	Underload	0,0.3	0.7
O	Overload	0.7,1	0.3
Universe of discourse: (0,1)			

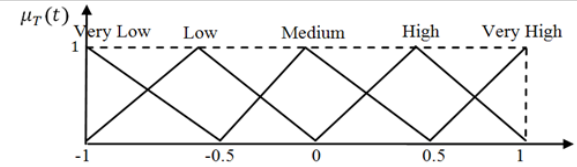

Fig. 1. The Membership Function of Input Variables of Tables 2 and 5

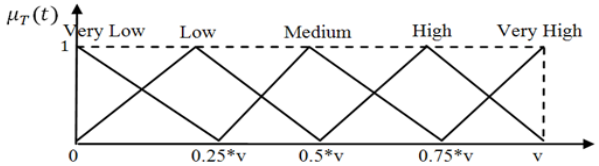

Fig. 2. The Membership Function of Input Variables of Tables 1, 3 and 4

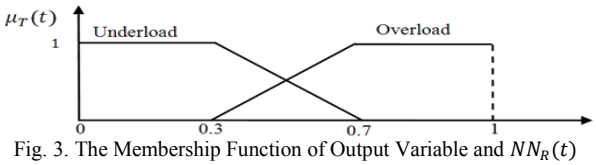

Fig. 3. The Membership Function of Output Variable and $NN_R(t)$

C. *The Expert System Rules*

We determine the ES rules base on aforementioned conditions to control VM's workload changes and predict the time of VM migration. The two main rules are described as follows. For random $t_r \in T$ and $\forall\, t_i \in T$ and *ct* as current time:

- **If** $NN_R(ct) = H$ **and** $VM_{et}(ct) = H$ **and** $\max_{t_i \in T} VM_{et}(t_i) = H$ **or** $VM_{et}(t_r + 1) - VM_{et}(t_r) = H$ **and** $CW(ct) = VH$ **and** $CC(ct) = H$ **or** $PI(t_r) - PI(t_r + 1) = L$ and $PI(ct - 120s) - PI(ct) = L$ and $VM_{et}(ct) - VM_{et}(ct - 120\ s) = \boldsymbol{H}$ **then** $VM_{WL} = O$
- **If** $NN_R(ct) = H$ **and** $VM_{et}(ct) = VL$ **and** $\max_{t_i \in T} VM_{et}(t_i) = H$ **or** $VM_{et}(t_r + 1) - VM_{et}(t_r) = VL$ **and** $CW(ct) = L$ **and** $CC(ct) = VL$ **or** $PI(t_r) - PI(t_r + 1) = H$ and $PI(ct - 120s) - PI(ct) = H$ and $VM_{et}(ct) - VM_{et}(ct - 120\ s) = VL$ **then** $VM_{WL} = U$

V. A FUZZY PREDICTABLE LOAD BALANCING APPROACH

In this section, we propose a FPLB approach. This approach contains a conceptual model and an algorithm which are designed to achieve system load balancing by migrating tasks from overloaded VMs. In addition, in this approach to decrease energy consumption and costs, we avoid choosing idle PMs as a new PM host, because if we transfer tasks to an idle PM, we have to turn it on and this action will increase energy consumption and costs [5].

The complex applications in cloud environment are classified into two groups: (1) computing intensive, and (2) data intensive applications. To transfer data intensive applications, the scheduling strategy should decrease the data

movement to reduce the transferring time; but for transferring computing intensive tasks, the scheduling strategy should schedule the data to the high performance computer [21]. In this paper, we consider bandwidth as a variable to minimize the tasks transferring time for data intensive applications. In addition we consider new host PM's properties (memory, hard disk, etc) to enhance performance utilization for computing intensive applications.

In cloud environment, there are some tasks schedulers that consider task types, priorities and their dependencies to schedule tasks in optimal way and transfer them to the specific VM's resources. In our proposed approach, we design a blackboard, where all cloud schedulers which manage VMs on clouds, share their information about VMs, their features and their tasks. Furthermore, the criteria of QoS as SLA information are mentioned in this blackboard. In proposed FPLB approach, there is a central scheduler that transfers tasks from an overhead VM to a new similar and appropriate VM. This scheduler applies the information of the blackboard to find an appropriate host VM for the task. In addition, the proposed FPM is used in FPLB approach to predict VM migration time to accelerate load balancing process and reduce response time. The proposed FPLB approach is illustrated in Figure 4 and its algorithm is described as follows:

Step1- Gathering data and information about virtual machine managers (VMMs), VMs, PMs and SLA information, in the global blackboard as inputs*:*
  1. VMs tasks information:
    1.1. The number of executing tasks
    1.2. Tasks' execution time and locations
    1.3. Tasks' required resources (number of required processors)
  2. PMs' Criteria (total/current)
    2.1. CPU (number and speed of the processors)
    2.2. Free Memory and Free Hard disk
    2.3. Bandwidth
    2.4. PM situation: Idle or active
    2.5. Its host VMM
  3. SLA information
  4. The objectives of the tasks migration optimization model and their information:
    4.1. Minimizing cost
      4.1.1. Cost policy information
    4.2. Minimizing execution time and transferring time
      4.1.2. Execution information
      4.1.3. Bandwidth information

Step2- Monitoring data and information to determine VMs' workflow information

Step3- Predicting VM migration time applying proposed FPM:
  1. Determining $\{t_i\} = T^* \subset T$ when $VM'_{et}(t_i) \geq 0$
  2. Determining $\{t_i\} = T^+ \subset T$ when $PI'(t_i) \leq 0$
  3. Determining $\max_{t_i \in T} VM_{et}(t_i)$
  4. Determining NN results about VMs' workload situation (Input for FPM)
  5. Calculating other FPM input variables
  6. Determining overloaded VMs and their migration time applying proposed FPM

Step4- Determining the list of tasks which should be migrated from overloaded VM's, and the list of candidate VMs to be the new host

Step5- Finding optimal homogeneous VMs as a new host for executing the tasks of the overloaded VMs, which is a multi-objective task migration problem, applying MOGA (this step will be described in Section VI).

Step6- Considering obtained optimal tasks migration schema, determining following information as the outputs:
  1. New optimal cost and optimal execution time
  2. Current VMs properties (Executing tasks, CPU, etc.)

Step7- Transferring tasks and their corresponding data to the determined optimal host VMs

Step8- Updating blackboards and schedulers' information according to the outputs of Step 4.

Step9- End.

VI. An Algorithm for Solving Multi-Objective Tasks Migration Problem Using MOGA

In this section, we describe a sub-algorithm to complete the Step 5 of FPLB algorithm and solve the multi-objective tasks migration problem. This sub-algorithm determines an optimal tasks scheduling model to assign tasks from overloaded VMs to the new host VMs applying MOGA. Among different MOGA methods, we apply Deb's NSGAII [19]. NSGAII not only has good convergence and distribution mechanism, but also has higher convergence speed. This sub-algorithm applies data and information which are determined in Steps 1 to 4 of the FPLB algorithm as its inputs, then finds the optimal schema to assign arrival tasks from overloaded VMs to host VMs, conducting following steps:

Step5.1. Determining candidate host VMs set by choosing the set of VMs which satisfy the constraints about host VMs' properties as $VM_{set} = \{vm_1, \dots, vm_m\}$

Step5.2. Eliminating the list of overloaded VMs (which are determined in Step 2 applying proposed FPM) from candidate host VMs set.

Step5.3. Determining the set of tasks which should migrate from overloaded VMs as immigrating tasks set: $T_{set} = \{t_1, \dots, t_n\}$

Step5.4. Applying MOGA to solve the multi-objective problem and assign the immigrating tasks to the optimal host VMs to minimize execution and transferring time and processing cost, conducting following steps:

Step5.4.1. Initializing population P_0 which is generated randomly

Step5.4.2. Assigning rank to each individual based on non-dominated sort

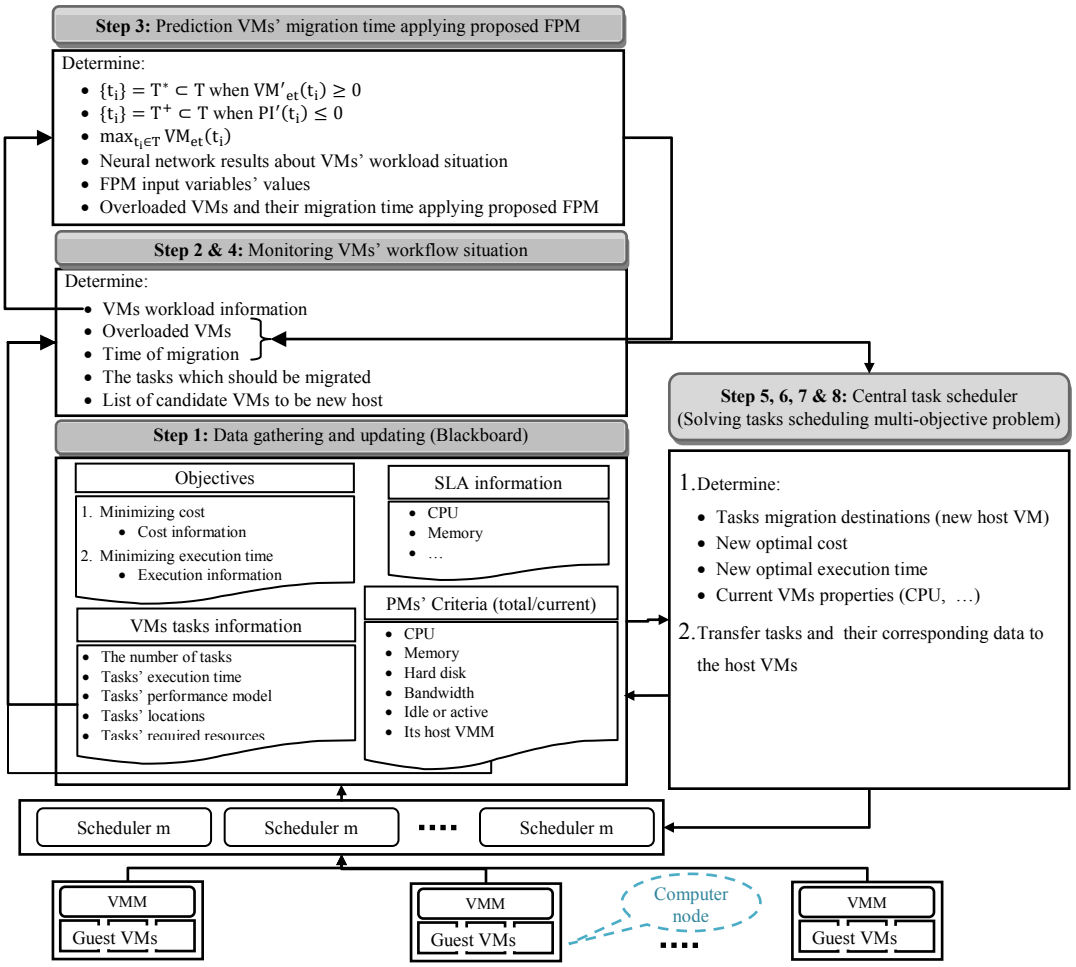

Fig. 4. The Conceptual Model of FPLB Approach

Step5.4.3. Implementing binary tournament selection, crossover and mutation on the initial population and creating a new population Q_0 and set $t=0$.

Step5.4.4. Merging the parent P_t and the child Q_t to form a new population $R_t = P_t \cup Q_t$.

Step5.4.5. Adopting non-dominated relationship to sort population and calculate the crowding distance among population on each layer.

Step5.4.6. Selecting the former N individuals as the parent population, namely $P_{t+1} = P_{t+1}\,[1{:}\,N]$ (Elite strategy).

Step5.4.7. Implementing reproduction, crossover and mutation on population P_{t+1} to form population Q_{t+1}.

Step5.4.8. If the termination conditions are met, output results as optimal tasks migration schema; otherwise, update the evolutionary algebra counter $t=t+1$ and go to step 5.4.4.

VII. EVALUATION

A system prototype is being developed based on the proposed FPLB and will be evaluated against the determined features. However, in this paper, the evaluation is presented by comparison of the proposed FPLB approach with traditional whole VM migration methods applying three parameters:

(1) Power Consumption: considering this fact that the less number of active PM means the less power consumption [5], we applied following ratio to compare power consumption after load balancing applying FPLB approach:

$$R_{pc} = \frac{number\ of\ active\ PM}{number\ of\ overloaded\ VMs} \qquad (10)$$

In proposed approach, to transfer extra tasks from overloaded VM, we just need to find a new similar VM on an active PM as a new host and there will be no need to turn a new PM on. In contrast, for whole VM migration, more hardware capacity is needed and it is impossible for every case to avoid choosing idle PM. Therefore, R_{pc} in FPLB approach compare to VM migration technique has lower value. Therefore, we have less "power consumption" after load balancing using FPLB approach and:

$$R_{pc_{Offline\ VM}} \geq R_{pc_{Online\ VM}} > R_{pc_{NewApproach}}$$

(2) Idle Memory: to compare the efficiency of FPLB approach, we apply idle memory that is prepared during the load balancing process as:

$$M_{im}(t) = OriginalVM_m(t) + HostVM_m(t) \quad (11)$$

where $OriginalVM_m$ and $HostVM_m$ are the amount of original VM memory and host VM respectively.

In offline VMs migration, the original VM should be suspend during VM migration time, and its memory and the amount of memory in the new host PM which is determined for host VM will be idle. In online VMs migration, although VM will not be suspended during migration process, the amount of memory in the new host PM will be idle in this time. In contrast, there is no VM migration in FPLB approach and the process of suspend and resume for original VM is eliminated. In conclusion, there will be no downtime for VMs and no idle memory in FPLB approach. As the results:

$$M_{im}(t)_{Offline\,VM} > M_{im}(t)_{Online\,VM} > M_{im}(t)_{NewApproach}$$

(3) Load Balancing Time Consumption: A part of load balancing time consumption is equal to VM migration time and preparation time for determining new PM host. In offline and online VM migration the total migration time is equal to migration one whole VM. This time in our approach is reduced to the time for transferring some extra tasks from overloaded VM. In addition, as in FPLB approach VMs' workload situation and the time of VM migration is predicted applying proposed FPM, the process of determining new VM host will start before VM overloading happen and load balancing system will be ready to transfer extra tasks when VM become overloaded without wasting time for preparation, in conclusion if $T_{lb}(t)$ is the value of load balancing time:

$$T_{lb}(t)_{Offline\,VM} > T_{lb}(t)_{Online\,VM} > T_{lb}(t)_{FPLB\,Approach}$$

VIII. Conclusion and Future Work

VM migration technique has been applied for elastic resource allocation, by migrating overload VM from one PM to another to achieve stronger computation power, larger memory, fast communication capability, or energy savings.

This paper proposed a new FPLB approach to achieve system load balancing by migrating arrival tasks from overloaded VM to another homogeneous VM instead of whole VM migration. The proposed approach has ability to determine overloaded VMs and predict their migration time. This approach also contains a multi-objective tasks migration model subject to minimizing cost, execution time and transferring time. In proposed approach there is no need to pause VM during migration time. In addition, the proposed approach will significantly reduce time, memory and cost consumption, because unlike tasks migration, VM live migration takes longer to complete and needs more idle capacity in host PM. Furthermore, proposed approach decreases energy consumption by avoiding choosing idle PMs as a new host PM. This approach also accelerates load balancing process and reduces response time applying proposed FPM.

In our future work we will improve our proposed method for predicting VM migration time considering SLA parameters.

References

[1] R. Buyya, J. Broberg, and A. Goscinski, "Cloud computing, Principles and Paradigms," 2011.

[2] M. Rosenblum, "The reincarnation of virtual machines," *Queue,* vol. 2, p. 34, 2004.

[3] C. Jun and C. xiaowei, "IPv6 virtual machine live migration framework for cloud computing," *Energy Procedia,* vol. 13, pp. 5753-5757, 2011.

[4] H. Jin, W. Gao, S. Wu, X. Shi, X. Wu, and F. Zhou, "Optimizing the live migration of virtual machine by CPU scheduling," *Journal of Network and Computer Applications,* vol. 34, pp. 1088-1096, 2011.

[5] X. Liao, H. Jin, and H. Liu, "Towards a green cluster through dynamic remapping of virtual machines," *Future Generation Computer Systems,* vol. 28, pp. 469-477, 2012.

[6] N. Jain, I. Menache, J. Naor, and F. Shepherd, "Topology-Aware VM Migration in Bandwidth Oversubscribed Datacenter Networks," *Automata, Languages, and Programming,* pp. 586-597, 2012.

[7] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the migration of virtual computers," *ACM SIGOPS Operating Systems Review,* vol. 36, pp. 377-390, 2002.

[8] A. Whitaker, R. S. Cox, M. Shaw, and S. D. Gribble, "Constructing Services with Interposable Virtual Hardware," in *Proceedings of the 1st symposium on networked systems design and implementation (NSDI)*, 2004, pp. 169-82

[9] S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The design and implementation of Zap: A system for migrating computing environments," *ACM SIGOPS Operating Systems Review,* vol. 36, pp. 361-376, 2002.

[10] M. Nelson, B. H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," 2005, pp. 25-25.

[11] C. Clark, K. Fraser, S. Hand, and G. H. Jacob, "Live migration of virtual machines," in *In Proceedings of 2nd ACM/USENIX Symposium on Network Systems, Design and Implementation (NSDI)*, 2005.

[12] W. Lin, J. Z. Wang, C. Liang, and D. Qi, "A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing," *Procedia Engineering,* vol. 23, pp. 695 – 703, 2011.

[13] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceeding of the 7th international conference on Autonomic computing*, 2010, pp. 11-20.

[14] K. M. Nagothu, B. Kelley, J. Prevost, and M. Jamshidi, "Ultra low energy cloud computing using adaptive load prediction," in *World Automation Congress (WAC), 2010*, 2010, pp. 1-7.

[15] M. Naderpour and J. Lu, "Supporting situation awareness using neural network and expert system," in *International Conference on uncertainty Modeling in Knowledge Engineering and Decision Making (FLINS 2012)* Turkey, Istanbul, 2012, pp. 993-998.

[16] M. Naderpour and J. Lu, "A fuzzy dual expert system for managing situation awareness in a safety supervisory system," in *2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Australia, Brisbane, 2012, pp. 1-7.

[17] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems,* vol. 28, pp. 155–162, 2012.

[18] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation,* vol. 2, pp. 221-248, 1994.

[19] Y. Zhang, C. Lu, H. Zhang, and J. Han, "Active vibration isolation system integrated optimization based on multi-objective genetic algorithm," in *Computing, Control and Industrial Engineering (CCIE), 2011 IEEE 2nd International Conference on*, 2011, pp. 258-261.

[20] J. Rao, "Autonomic management of virtualized resources in cloud computing," 2011.

[21] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm," *Journal of Networks,* vol. 7, pp. 547-553, 2012.