ISSN 1330-3651(Print), ISSN 1848-6339 (Online)
UDC/UDK 004.93/.94.032.26

INTELLIGENT ADAPTIVE MULTI-PARAMETER MIGRATION MODEL FOR LOAD BALANCING VIRTUALIZED CLUSTER OF SERVERS

Mohsen Tarighi, Seyed Ahmad Motamedi, Saeed Sharifian

Original scientific paper
The most important benefit of virtualization is to get a load balanced environment through Virtual Machine (VM) migration. Performance of clustered services such as Average Response Time is reduced through intelligent VM migration decision. Migration depends on a variety of criteria like resource usage (CPU usage, RAM usage, Network Usage, etc.) and demand of machines (Physical (PM) and Virtual (VM)). This is a multi-criteria migration problem that evaluates, compares and sorts a set of PMs and VMs on the basis of parameters affected on migration process. But, which parameter(s) has dominant role over cluster performance in each time window? How can we determine weight of parameters over oncoming time slots? Current migration algorithms do not consider time-dependent variable weights of parameters. These studies assume fixed weight for each parameter over a wide range of time intervals. This approach leads to imprecise prediction of recourse demand of each server. Our paper presents a new Intelligent and Adaptive Multi Parameter migration-based resource manager (IAMP) for virtualized data centres and clusters with a novel Artificial Neural Network (ANN)-based weighting analysis named Error Number of Parameter Omission (ENPO). In each time slot, weight of parameters is recalculated and non-important ones will be attenuated in ranking process. We characterized the parameters affecting cluster performance and used hot migration with emphasis on cluster of servers in XEN virtualization platform. The experimental results based on workloads composed of real applications, indicate that IAMP management framework is feasible to improve the performance of the virtualized cluster system up to 23 % compared to current algorithms. Moreover, it reacts more quickly and eliminates hot spots because of its full dynamic monitoring algorithm.

Keywords: *ANN (Artificial Neural Network), load balancing, parameter dynamic weight, virtualized cluster of servers*

Inteligentni adaptivni više-parametarski migracijski model za uravnoteženje opterećenja virtualne skupine servera

Izvorni znanstveni članak
Najvažnija korist virtualizacije je dobivanje okruženja s ujednačenim opterećenjem kroz prenošenje (migraciju) virtualnim strojem (VM). Djelovanje usluga u skupinama (klasterima), kao što je prosječno vrijeme reakcije - Average Response Time - reducirano je inteligentnom odlukom VM o prenošenju. Prenošenje ovisi o nizu kriterija poput korištenja resursa (uporaba CPU, korištenje RAMa, korištenje mreže, itd.) i potrebe za strojevima (fizičkim (PM) i virtualnim (VM)). To je više- kriterijski problem prenošenja koji procjenjuje, komparira i sortira niz fizičkih i virtualnih strojeva (PM i VM) na osnovu parametara istaknutih u procesu prenošenja. Ali, koji parametar (parametri) ima dominantnu ulogu nad djelovanjem klastera u određenom vremenskom odjeljku? Kako možemo odrediti težinu parametara u nadolazećim vremenskim razmacima? Postojeći algoritmi prenošenja (migration algorithms) ne uzimaju u obzir težine parametara koje se mijenjaju ovisno o vremenu. Te analize pretpostavljaju fiksnu težinu za svaki parametar kroz široki raspon vremenskih intervala. To dovodi do netočnog predviđanja o traženju rješenja za svaki server. U našem se radu predstavlja novi Inteligentni i Adaptivni Multi Parametarski (IAMP) upravljač resursima na bazi prenošenja (migracije) za virtualizirane centre podataka i klastere s novom na umjetnoj neuronskoj mreži (ANN) temeljenoj analizi težina nazvanoj Error Number of Parameter Omission (ENPO). U svakom se vremenskom razmaku težina parametara ponovo izračunava te će nevažni parametri biti oslabljeni u postupku rangiranja. Obilježili smo parametre koji utječu na performansu klastera i koristili hot migration s naglaskom na skupini servera u XEN platformi virtualizacije. Eksperimentalni rezultati temeljeni na radnim opterećenjima sastavljenim od stvarnih aplikacija pokazuju da je primjenom IAMP-a moguće poboljšati rad virtualnog klaster sustava do 23 % u usporedbi s postojećim algoritmima. Što više, on brže reagira i eliminira vruće točke zbog svog potpuno dinamičkog upravljačkog algoritma.

Ključne riječi: *ANN (umjetna neuronska mreža), dinamička težina parametra, uravnoteženje opterećenja, virtualna skupina servera*

1 Introduction

Virtualization technology, due to its capabilities of isolating, consolidating and migrating workloads, has recently emerged for data centres and cluster systems [1]. Non-virtualized Data Centres provide static allocation of resources to hosted applications. For dynamic workloads, fixed allocation may cause underloading/overloading PMs. The former lead to unutilized resource and the later violates SLA. Therefore, resources should be managed efficiently [2].

Virtualization allows a single PM to share resources across multiple services. Each service runs in a separate VM container [3]. Perhaps, the biggest advantage of virtualization is the ability to migrate an entire OS out of overloaded/overheated servers to less loaded PMs in order to handle dynamic workloads [4]. This ability overcomes difficulties that traditionally have made process migration. XEN work [5, 6] showed that VM migration can enable highly responsive provisioning in clusters. VMware [7, 8] and XEN [9] have implemented live migration with extremely short downtimes. VM migration-based load balancing policy should be structured intelligently because it affects the performance of a cluster. Migration coordinates the use of processing and bandwidth capacity of source and target hosts.

Various economic models have been presented to allocate shared resources of physical machines across a number of services [9]. In these approaches, physical recourses have equal influence on cluster state. Therefore, they are semi-dynamic methods. We have introduced an intelligent and full-dynamic migration model. IAMP model considers dynamic weight for resources to make fast response system for hot spot detection and migration. Designed algorithm fulfils SLA agreements while improves overall utilization of cluster. To weighting parameters affecting cluster performance, we implemented ANN-based analysis technique. Implication is for virtualized clusters with heterogenic XEN-based nodes.

The rest of this paper is structured as follows: Section 2 presents motivation and related work. Section 3 talks about IAMP resource allocation model, performance metric and migration parameters. Section 4 presents weighting approach and corresponding neural network concepts. Section 5 includes case study, experimental

environment and model evaluations. In Section 6, algorithm analysis and detailed discussion are presented. Finally, we conclude in Section 7.

2 Motivation and related work

VMware DRS uses migration to perform load balancing in response to CPU and RAM pressure [7]. DRS monitors memory and processor usage of physical nodes to respond to potential SLA violations. Although Virtual Machines with I/O intensive workloads might saturate the bandwidth of the Network Interface Card (NIC) of a single host and leads to performance degradation, DRS does not make VM placement decisions based on NIC bandwidth usage criterion.

Sandpiper monitors system resource usage and triggers migrations if necessary [10]. It considers multiple resources for each server. Sandpiper defines a new metric called volume. The volume captures the degree of load along resources in a non-weighted method and does not consider the parameters importance in decision.

Sandpiper mounts over XEN-based nodes and improves cluster performance through migration. XEN is a widely used ×86 VMM which can support VMs with strong resource isolation and performance guarantee [1]. In [19], a performance model of migration is build using statistical methods such as regression. Specifically, they did a series of experiments by migrating a XEN-based VM and allocating different amount of CPU share to the migration.

[20] issues to achieve the goal of management multiple virtual machine migrations across physical machines without disruption.

Tarighi et al. [11] presented a model to migrate VMs between cluster nodes using TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) algorithm. Fuzzy migration Decision Model was used to find and move the most overloaded virtual machines from the most overloaded PM to the least underloaded one. However, weights of the parameters were assumed equal.

To weight migration parameters, Analytic Hierarchy Process (AHP) has been applied in [12]. The relative importance of parameters is determined by pair-wise comparisons matrix. However, AHP requires domain expert judgments and preferences.

This paper proposes a dynamic parameter weighting approach that balances cluster load adaptively and quickly. Our novel weighting algorithm called enpo method predicts the effect of each parameter for next time slots. In different time intervals, different parameter(s) may have dominated role on cluster average response time.

3 IAMP Migration-Based Load Balancing Model

We model migration-based resource allocation problem within a datacentre that runs different type of application workloads, demands and SLA requirements. Nodes of cluster are heterogonous. Operating system, applications and data are accessed by storage networking technologies SAN systems.

At the beginning of each time slot, IAMP explores for optimum useful migrations. It ranks physical machines based on a set of criteria and determines which PM is near to be overloaded and which server is underloaded. Parameters weights are derived from neural network and *enpo* algorithm. Neural Network estimates the function that relates migration parameters and the cluster average response time. Samples are gathered at the end of each time interval whose duration is determined by the algorithm frequency. Then, *enpo* analysis function operates over the neural network to approximate the weights of parameters. Parameters weight is applied for migration deciding algorithm. A server to become overloaded is assigned higher score and underloaded ones are placed at the bottom of the ranking list. Next step is to compare the virtual machines running on the overloaded physical servers and try to reduce average cluster response time by migrating the best appropriate VM(s) between candidate source and destination PMs while decreasing SLA violations. IAMP also considers Unbalanced Factor (UF) issue for migration decisions. UF is a measure which indicates the distribution of VM workloads among the PMs [13].

If the selected VM does not meet the requirements, the algorithm moves on to the next VM until a match is found. In the case of finding no VM on the source candidate PM, the next overloaded physical machine is being verified.

3.1 Cluster performance metric

Server overloading affects both the server throughput and the response time. When the service rate of the resource cannot keep up with arrival rate of incoming requests, the queue length of that resource becomes large and thus the response time theoretically increases to infinity. Our goal is to avoid this dramatically increasing by replacing VM(s) over cluster nodes.

For file server, performance is measured in terms of both throughputs and response time but response time is more important especially in the highly interactive environment because throughput cannot normally be determined from latency alone. For web applications, the response time is the most important factor to create a convenient user experience. A server that is too busy is unable to satisfactorily respond to client requests and large delays cause user frustration because nobody spends much time exploring a site with a sluggish response time.

In this paper, Average Response Time ($\overline{R_t}$) of the cluster is used as performance metric. When the server is overloaded, additional requests cannot be processed. As a result, coming requests for virtual machines are queued up, resulting in a significant increase in response times.

3.2 Effective parameters

The main server resources are the NIC, CPU, RAM and HDD [14]. Depending on resource usage pattern of cluster applications, the demands for some server resources may be low while one or more resources are over utilized. For example, network interface becomes bottleneck when there are some Network-Intensive virtual machines concurrently transmitting large files.

Long response time of a cluster server is an undesirable phenomenon in virtualized data centres. To

balance overall system response time, lots of parameters should be considered. These parameters can be divided into two categories PM and VM parameters. In Tab. 1, seven parameters that illustrate the physical machines status have been listed. The first criterion is percentage of PM's CPU used (P_{cpu}) that certainly depends on processor usage of all hosted virtual servers. This parameter, however, cannot solely be used to establish equality or inequality among heterogeneous machines. 30 % usage of a 1 GHz CPU differs to 30 % of a 2 GHz CPU. Therefore, the processor clock speed (C_{cyl}) should be taken into account, too. C_{cyl} makes comparing two nodes processing status meaningful.

Memory utilization (P_{ram}) and memory capacity (C_{ram}) are another pair of parameters. P_{ram} changes for different applications due to varying memory usages and access patterns. For RAM-Intensive applications, when the amount of the available memory on the server is lacking, the performance will drop significantly and user might see QoS degradation and SLA violation because of memory swapping.

Next pair of parameters is network bandwidth utilization (P_{nic}) and node network bandwidth (C_{nic}).

Table 1 Physical machine's parameters

Parameter	Description	Unit
P_{cpu}	Percentage of average CPU usage of PMs in each time interval	%
P_{ram}	Percentage of average memory usage of PMs in each time interval	%
P_{nic}	Percentage of Average NIC usage of PMs in each time interval	%
C_{cyl}	Node's CPUcapacity clock speed	GHz
C_{nic}	Bandwidth of PMs' network interface card	GB/s
C_{ram}	Memory capacity of PMs	GB

Since each parameter has a different effect on the cluster performance, it cannot be assumed that they all have equal impacts and weights. So as a result, finding the appropriate weight for each criterion is our main point in load balancing cluster through VM migration.

4 Parameters weighting model

Weights of parameters affect the optimal migration solution (identifying migratory VM(s), here). Besides, because of workload dynamism, parameters effectiveness may change over time slots. PMs host virtual machines running different workloads with dynamic specifications. Recourse utilization is a function of time and varies over time. Therefore, assuming parameters with fixed weights is impractical and inadequate. Thus, the question is which parameter is more important than other in each time slot.

Methods for finding weights can be categorized into two groups. Subjective weights are determined according to the preference of decision makers and the objective methods determine weights by solving mathematical models. In this paper, to determine physical machine parameters weight, neural network algorithm and post processing *enpo* function are applied. For VM parameters, weight resulted from group one is used.

Cluster response time nonlinearly changes with lots of parameters and moreover, there is no straight forward algebraic equation between them. Due to multiplicity of effective parameters and complexity of interactions among these parameters, statistical methods such as multivariate regression analysis (MVRA) cannot be fully appropriate because of suffering resulting equations from approximation and comparatively highly error of estimation. Here, ANN technique as an alternative to this conventional method is applied. It is an intelligent tool for simulating complex relationship between cluster response time and utilization parameters when the conventional approaches are unable. *enpo* operator, then, determines which attribute is more important than others and how much error in estimating response time grows up if a parameter is eliminated from processing.

4.1 IAMP Neural Network Architecture

A neural network can be defined using three fundamental modules including network architecture, transfer function and learning law. These components are defined according to the type of problem to be solved. Each network layer consists of a number of elementary processing units or neurons, which are connected to the next layer. The local weights given to different links play a major role in processing. Before ANN is put into actual operation, the network weight and bias values are to be estimated. The transfer functions are designed to map an input layer to its output. IAMP applies a novel algorithm over the supervised back propagation multi-layer perceptron neural networks (BPMLPNN) [15]. The IAMP load balancer is trained to learn the relationship between cluster inputs and output parameters. Then, the trained network is processed by *enpo* function $\varphi_{enpo}()$ to determining parameter's weight for making migration decision. This enables us to recognize the most effective factors (CPU or RAM or NET) affecting cluster output (average response time).

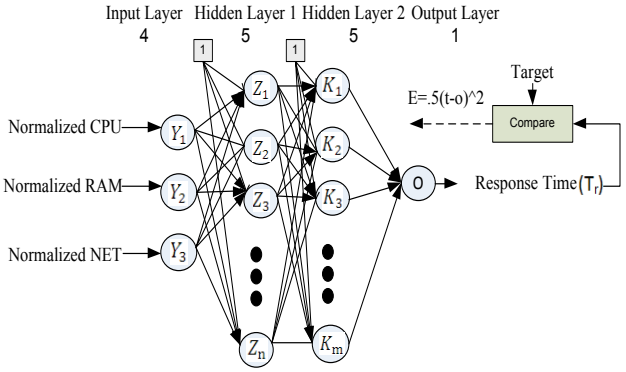

Figure 1 IAMP neural network architecture

IAMP neural network function $f_{nn}()$ was designed in order to establish relationships between percentage of CPU usage, Processor Clock Speed, memory utilization and capacity, Network bandwidth utilization, server's NIC bandwidth and the response time as the output parameter. The characteristics of the IAMP neural network architecture are shown in Fig. 1. The 4-layers perceptron ANN was found to be optimum with architecture of four neurons in input layer, 5 neurons in 1st

hidden layer and 5 neurons at the 2nd hidden layer. If we design the network with few layers and neurons, the network's ability for generalization lessens. In contradictory, a network with lots of layers puts a great processing burden on the IAMP algorithm.

The network computes simulated output using internal weights and thresholds. Among various algorithms available for training neural networks, we selected the back-propagation algorithm which provides the most efficient learning procedure.

During network training, the input layer receives input values, pass them on to the hidden nodes, which multiply the input by connection weights, add up such product and after attaching a bias, transform the result through a transfer function. Then, the actual output is compared with the desired output to determine the error. The error is calculated using the root mean squared error. RMSE is processed back through the network by updating individual connection weights of the connections and biases of each neuron. The process is repeated, as many epochs as needed, for all training pairs in the dataset, until the network predefined error is reached successfully [16].

4.2 Neural network pre-processing

At the end of time slot *t*, each node PM^j calculates k samples $p^j_{cpu_i}$, $p^j_{ram_i}$ and $p^j_{nic_i}$. To convert heterogonous data to meaningful and comparable information, CPU-usage data samples ($p^j_{cpu_i}$) are divided by the CPU-cycle (C^j_{cyl}) to form new parameter $cpu^j_{s_i}$ with dimension $\% * \text{GHz}^{-1}$. It represents the sample s_i of CPU-usage in each GHz of node PM^j processing capacity. $cpu^j_{s_1}$, …., $cpu^j_{s_k}$ are sent to the cluster master node. Eq. (1) shows how to calculate $cpu^j_{s_i}$. $ram^j_{s_i}$ and $net^j_{s_i}$ samples are generated in the same way.

$$cpu^j_{s_i} = \frac{p^j_{cpu_i}}{C^j_{cyl}} \text{ for } i = 1, \dots, k \quad (\% * \text{GHz}^{-1}). \tag{1}$$

Master node collects samples from all forwarder nodes. After averaging samples (2) and (3), we have four vector of samples cpu_{s_i}, net_{s_i},ram_{s_i} and rt_{s_i}. In fact, IAMP creates a logical Meta model of the cluster. That is, we assume cluster of nodes as a single physical machine that has one CPU, NET and RAM resource and 3K samples. cpu_{s_i} and ram_{s_i} are calculated in the same way.

$$net_{s_i} = \frac{1}{n}\sum_{j=1}^{m} net^j_{s_i}, \tag{2}$$

$$rt_{s_i} = \frac{1}{n}\sum_{j=1}^{m} rt^j_{s_i}. \tag{3}$$

Neural network should be trained with normalized data. Therefore, samples are divided by maximum value to yield normalized samples.

4.3 IAMP Neural Network Calculations

The processing is performed during training and test phases. Sample matrix $\vec{S}(t)_{[4\times K]}$ is the vector consisting of K samples of each PM^j parameters in time slot t (4). For example, the first row demonstrates k cpu samples cpu_{s_1}, … , cpu_{s_i}, …, cpu_{s_k} and the 4th row indicates response time samples $rt_{s_1}, \dots, rt_{s_i}, \dots, rt_{s_K}$.

$$\vec{S}(t)_{[4\times K]} = \begin{bmatrix} cpu_{s_1} & cpu_{s_i} & cpu_{s_K} \\ ram_{s_1} & ram_{s_i} & ram_{s_K} \\ net_{s_1} & \cdots\ net_{s_i}\ \cdots & net_{s_K} \\ rt_{s_1} & rt_{s_i} & rt_{s_K} \end{bmatrix}. \tag{4}$$

Training of the *IAMP* neural function is a process of arriving at an optimum local weight space of the network. Learning is achieved through back propagation without momentum. Fast algorithm $trainlm$ is used for training to reduce processing time. The learning rate is 0,05. The training process is stopped as soon as one of the stopping criteria is satisfied. A root mean square error rate of less than 0,01 is used as a terminating condition. Nonlinear functions tangent sigmoid are used as transfer functions. The training is discontinued when RMSE as small as 0,01 or when a very large number of iterations (here, 1000 epochs) are completed. After a training phase, the network is tested with the validation data. Here, $K = 200$ is sufficient. That is, the training data-set contains 180 samples and validation data-set contains the remaining 20 samples. The output of IAMP neural network is a non-revealed function that relates average response time $\overline{R_t}$ to cluster resources usage.

$$\overline{R_t} = f(cpu_t, ram_t, net_t). \tag{5}$$

This relationship indicates Performance–Utilization function. $\overline{R_t}$ shows overall performance of the cluster and cpu_t, ram_t and net_t parameters illustrate utilization of resources. Neural network does not explicitly reveal the exact algebraic relation between them.

4.4 Parameter weighting using *ENPO* algorithm

Here, we introduce *Error Number of Parameter Omission (ENPO)* concept as a new method to indicate weights and parameters priorities. A useful concept of sensitivity analysis in neural network called RSE was introduced in [17]. Here, to identify the significance of each parameter on the factor response time, trained neural network is processed by *enpo* Analysis. This enables our model to recognize the most important factors affecting cluster response time in each time slot. IAMP creates weight vector by ANN and *enpo* function for time slot $+\tau$, Eqs. (6) and (7).

$$\vec{W}(t+\tau) = \left(w^{t+\tau}_{cpu}, w^{t+\tau}_{ram}, w^{t+\tau}_{net}\right), \tag{6}$$

$$\vec{W}(t+\tau) = \varphi_{enpo}\left(f_{nn}\left(\vec{S}(t)\right)\right). \tag{7}$$

Function $\varphi_{enpo}()$ is the *enpo* operator and $f_{nn}()$ is nonlinear neural network function. Function φ_{enpo}

operates on the output of the neural network to determine the parameters weight vector for the next time interval $t+\tau$. The neural network is first trained with samples matrix and then, the RMSE error (e_t) is computed.

$$e_t = 0{,}07\sqrt{\sum_{s_i=1}^{180}\left(rt_{s_i} - \widehat{rt_{s_i}}\right)^2}. \quad (8)$$

Which $\widehat{rt_{s_i}}$ is the predicted response time by $f_{nn}()$ and rt_{s_i} is the real value. When the designed BPNN has been trained successfully, the neural network is no longer allowed to adapt. Then, for each criterion C_i, parameter's samples is omitted from the matrix $\vec{S}(t)$ and error e_{no-C_i} is calculated again. For example, $\vec{S}_{\mathrm{no-cpu}}(t)$ indicates sample vector with no CPU samples (9) and e_{no-cpu} is the error of the neural network in test phase when cpu samples are ignored (10).

$$\vec{S}_{no-cpu}(t) = \begin{bmatrix} 0 & 0 & 0 \\ ram_{s_1} & ram_{s_i} & ram_{s_K} \\ net_{s_1} \cdots & net_{s_i} & \cdots net_{s_K} \\ rt_{s_1} & rt_{s_i} & rt_{s_K} \end{bmatrix}, \quad (9)$$

$$e_{no-cpu}(t) = error\left(f_{nn}\left(\vec{S}_{no-cpu}(t)\right)\right). \quad (10)$$

We assigned zero to cpu samples in $\vec{S}_{no-cpu}(t)$ and fed this matrix to the $f_{nn}()$ just for test phase. Then *enpo* number that indicates parameters weight for the next time interval $t+\tau$ calculated as below:

$$\varphi_{enpo}(C_i) = w_{C_i}^{t+\tau} = \frac{\left|e_{no-C_i} - e_t\right|}{\sqrt[2]{\sum_{i=1}^{3}\left|e_{no-C_i} - e_t\right|}}. \quad (11)$$

For $i = 1,2,3$, and $C_i = cpu_t, ram_t$ and net_t. *enpo* number is between 0 and 1. If in a specific time interval, the processor is the bottleneck resource, after omitting it, the error e_{no-cpu} gets rises and consequently, the *Error Number of* CPU *Omission* increases. We use this weight vector as an input to rating algorithm for sorting PMs. If a parameter has no effect on the response time in a specific time interval, then $e_{no-C_i} = e_t$ and corresponding *enpo* number will be 0.

4.5 IAMP Sorting Algorithm

Decision engine tries to find the best source and destination PMs among cluster nodes. For solving this multi-dimensional problems, fast version of TOPSIS method (FTOPSIS), has been developed [18]. In FTOPSIS, to transform the various attribute dimensions into non-dimensional attributes, which allows comparison across the attributes, each value is divided by maximum (linear normalization) value instead of root mean square of all values (vector normalization). The latter is time consuming and not suitable for live processing.

The basic policy of the sorting algorithm is that the chosen PM should have the shortest distance from the ideal physical machine (PM^+) and the farthest distance from the negative ideal physical machine (PM^-). The ideal positive PM is formed as a composite of the best performance values exhibited by any PM for each parameter. PM^- is the composite of the worst performance values. Proximity to each of these performance poles is measured in the Euclidean sense. The higher value of closeness coefficient indicates that a PM is closer to PM^+ and farther from PM^- simultaneously. IAMP has m alternatives physical machines $PM^1, \ldots, PM^j, \ldots$, and three decision parameters cpu, ram and net. All the necessary information assigned to the PMs in time slot t form a decision matrix $\overrightarrow{IAMP}_{[\mathrm{m}\times 3]}(\mathrm{t}+\tau)$.

$$\overrightarrow{IAMP}(t+\tau) = \begin{matrix} \\ cpu^1(t) \\ \vdots \\ cpu^j(t) \\ \vdots \\ cpu^m(t) \end{matrix} \begin{matrix} cpuramnet \\ \begin{bmatrix} ram^1(t) & net^1(t) \\ \vdots & \vdots \\ ram^j(t) & net^j(t) \\ \vdots & \vdots \\ ram^m(t) & net^m(t) \end{bmatrix} \end{matrix}. \quad (12)$$

$cpu^j(t)$, $ram^j(t)$ and $net^j(t)$ are the average of K *cpu, ram* and *net* samples in PM^j at time slot *t*. Then the migration matrix is made dimensionless by dividing each entry by maximum value of each column. Also the weight vector is used to have weighted normalized migration matrix $\overrightarrow{IAMP}_{wn}(t+\tau)$ (13), (14).

$$\begin{bmatrix} \frac{w_{cpu}^{t+\tau} cpu^1(t)}{cpu_{\max}(t)} & \frac{w_{ram}^{t+\tau} ram^1(t)}{ram_{\max}(t)} & \frac{w_{net}^{t+\tau} net^1(t)}{net_{\max}(t)} \\ \vdots & \vdots & \vdots \\ \frac{w_{cpu}^{t+\tau} cpu^j(t)}{cpu_{\max}(t)} & \frac{w_{ram}^{t+\tau} ram^j(t)}{ram_{\max}(t)} & \frac{w_{net}^{t+\tau} net^j(t)}{net_{\max}(t)} \\ \vdots & \vdots & \vdots \\ \frac{w_{cpu}^{t+\tau} cpu^m(t)}{cpu_{\max}(t)} & \frac{w_{ram}^{t+\tau} ram^m(t)}{ram_{\max}(t)} & \frac{w_{net}^{t+\tau} net^m(t)}{net_{\max}(t)} \end{bmatrix}. \quad (13)$$

$$ram_{\max}(t) = \max\left(ram^j(t)\right) j = 1, \ldots, m, \quad (14)$$

$cpu_{\max}(t)$ and $net_{\max}(t)$ are calculated in the same way.

Next $PM^+$and $PM^-$are determined.

$$PM^+ = \{cpu^+, ram^+, net^+\} = \mathrm{Max}\{cpu, ram, net\}, \quad (15)$$

$$PM^- = \{cpu^-, ram^-, net^-\} = \mathrm{Min}\{cpu, ram, net\}. \quad (16)$$

For example, $cpu^+ = \mathrm{Max}\left\{\frac{w_{cpu}^{t+\tau} cpu^j(t)}{cpu_{\max}(t)}\right\}$ for $j = 1, \ldots, m$ and $ram^- = \mathrm{Min}\left\{\frac{w_{ram}^{t+\tau} ram^j(t)}{ram_{\max}(t)}\right\}$ for $i = 1, \ldots, m$.

Then, the distance of each PM^j from the $PM^+$and $PM^-$is measured using the Euclidean distance. Finally, the relative closeness of each alternative to $PM^+$is calculated (17). It shows the rank of the PM^j; the bigger the $PM_{\mathrm{score}}^j(t+\tau)$, the better the alternative PM^j.

$$PM^{j}_{\text{score}}(t+\tau) = \frac{\sqrt{\sum_{j=1}^{3}(PM^{j}-PM^{-})^{2}}}{\sqrt{\sum_{j=1}^{3}(PM^{j}-PM^{-})^{2}}+\sqrt{\sum_{j=1}^{3}(PM^{j}-PM^{+})^{2}}}. \quad (17)$$

Weights of virtual machines parameters are assumed equal to weights of physical node parameters. To more illustration, both $w^{t+\tau}_{P_{vcpu}}$ and $w^{t+\tau}_{cpu}$ refer to processing resources of cluster. To find appropriate VM to migrate, a virtual machine decision matrix $\overrightarrow{VM}(t+\tau)$ is formed for VMs located on the PM^{j} with the maximum $PM^{j}_{\text{score}}(t+\tau)$ to decide which virtual machine should be migrated. Suppose that there are l virtual machines VM^{j}_{w} to $VM^{j}_{z}(l=z-w)$ which are running on the physical machine PM^{j}, we have:

$$\overrightarrow{VM}(t+\tau) = \begin{matrix} \\ VM_w \\ \vdots \\ VM_z \end{matrix} \begin{matrix} P_{vcpu} P_{vram} P_{vnic} \\ \begin{bmatrix} p^{w}_{vcpu} p^{w}_{vram} p^{w}_{vnic} \\ \vdots \quad \vdots \quad \vdots \\ p^{z}_{vcpu} p^{z}_{vram} p^{z}_{vnic} \end{bmatrix} \end{matrix}. \quad (18)$$

Like physical machines, we calculate $VM^{j}_{w_{\text{score}}}$ to $VM^{j}_{z_{\text{score}}}$.

5 Model evaluation - a case study

We are interested in minimizing cluster average response time as an aggregated SLA factor. Results show more accuracy, reduced useless migrations and better load balancing that causes improved average response time of cluster due to IAMP parameter weight-aware algorithm. We implemented a virtualized cluster server. Each server is connected with a high-speed 10 GB/s LAN network. Depending on VM's applications, different SLAs in form of desired service response time have been defined. We consider a discrete-time window in which time is slotted into intervals with equal length of τ. SLA will be broken if the service response time is greater than the agreed at a specific time duration. The idea is to monitor the resource demand during the current time window t in order to make decisions about the VM(s) reallocations in the next time window $t+\tau$. For all virtual machines, there are no reservations or limits. If the resource demand of any application exceeds the SLA, VM size auto-scaling facility is provided or migration is done.

5.1 Test -bed

For the performance evaluation, various experiments have been done over a heterogeneous cluster of 5 physical machines and 25 Virtual Machines. Cluster hardware specifications are illustrated in Tab. 3. Each node runs different NIC-intensive, CPU-intensive and RAM-intensive workloads varying over time such as SPEC web for Apache 2.2 Web server, Dbench for File server and Swing bench for Data base server. SAN storage is configured for the cluster to allow system to migrate a SCSI disk by reconnecting to the disk on the destination node. A number of client machines generate workloads for VMs while a desktop machine is used for automating the experiments and analyzing measurements. All nodes run Linux 2.6.16 and we will base our analysis on the XEN 3.0.3. migration infrastructure.

Table 3 Cluster node specifications

PM^{j}	CPU	RAM	NIC
PM^{1}	2×2,66 GHz	16 GB	10 GB/s
PM^{2}	4×2,8 GHz	32 GB	1 GB/s
PM^{3}	8×1,7 GHz	8 GB	1 GB/s
PM^{4}	2,66 GHz	64 GB	1 GB/s
PM^{5}	2×3,5 GHz	24 GB	10 GB/s

XEN is open-source and allows us to efficiently determine the migration sub-system design and implementation. The *Dom 0* is the manager domain launched and configures all other regular guest unprivileged domains *Dom U*. Initial distribution of virtual machines over cluster nodes was set randomly. IAMP runs every τ minute and collects cluster information consisting of resource samples for all parameters of each PM^{j} and calculates global parameters weight and feeds them into IAMP decision making module.

5.2 IAMP algorithm results

Initial distribution of virtual machines over cluster nodes is shown in Tab. 4. Sampled data are transferred by physical machines to IAMP node.

Table 4 VMs initial distribution

PM^{j}	VM^{j}_{w}
PM^{1}	$VM^{1}_{1}, VM^{1}_{3}, VM^{1}_{4}, VM^{1}_{5}, VM^{1}_{6}, VM^{1}_{7}, VM^{1}_{8}$
PM^{2}	$VM^{2}_{1}, VM^{2}_{2}, VM^{2}_{3}$
PM^{3}	VM^{3}_{1}, VM^{3}_{2}
PM^{4}	$VM^{4}_{1}, VM^{4}_{2}, VM^{4}_{3}, VM^{4}_{4}, VM^{4}_{5}, VM^{4}_{6}, VM^{4}_{7}$
PM^{5}	$VM^{5}_{1}, VM^{5}_{2}, VM^{5}_{3}, VM^{5}_{4}, VM^{5}_{5}, VM^{5}_{6}$

After averaging, neural network operates over final 4× 200 samples. Data predicted by neural network in the test phase and actual data are compared in Fig. 2.

After omitting each parameter's samples from the process, the neural network predicted response times in absence of CPU, RAM, and NET are computed again and compared with actual response time as Fig. 3 indicates.

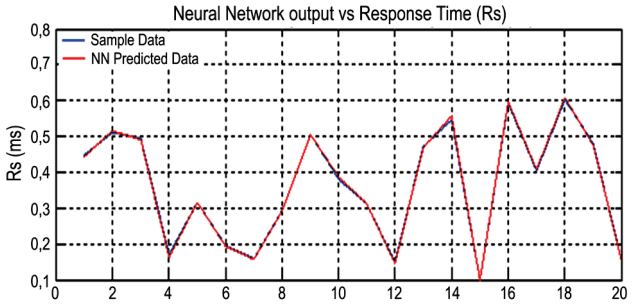

Figure 2 Observed and predicted data by the neural network

As shown, at time interval $t+2\tau$, the network and somehow processing resources are the bottleneck resources respectively because upon ignoring NET

samples from neural network processing, the error e_{no-net} got rises significantly. For memory, *enpo* errors are negligible. Cluster conditions support these results because data gathered 30 minutes ($\tau = 15$ min) after powering on the cluster indicates lots of NET-intensive file server virtual machines.

IAMP master node computes criteria weights $\vec{W}(t+\tau) = (0{,}19;\ 0{,}09;\ 0{,}72)$. These weights are used to construct the weighted normalized decision matrix $\overrightarrow{IAMP}_{wn}(t+\tau)$. Fig. 4 shows *enpo* number of weight of physical machine parameters for time slot $t + 2\tau$.

At the end of time interval $t + 2\tau$, control unit verifies whether the load on this node breaks the threshold defined by the administrator or not. That is, PM^j with one or more resources greater than threshold are included in *IAMP* load balancing algorithm. One snapshot of the cluster that consists of average usage of $PM^1, \ldots, PM^5$ resources in time slot $t + 2\tau$ is indicated in Tab. 5. For time slot $t + 3\tau$ (35 minutes after switching cluster on), Physical server PM^1 obtained the highest score 78,2 (on the top of the table) and PM^5 is the last entry (see Tab. 6). This means that this node and its virtual services are in danger and migration would be helpful.

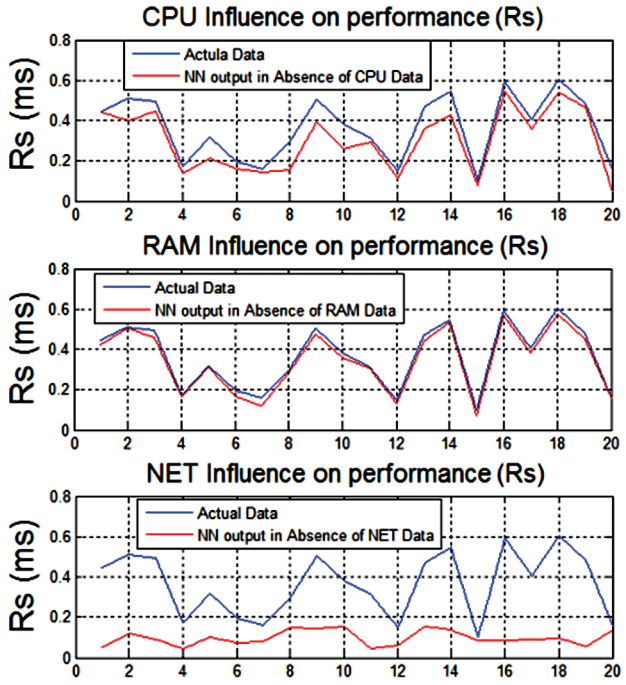

Figure 3 Neural network predicted response time in Absence of Parameters CPU, RAM, and NET

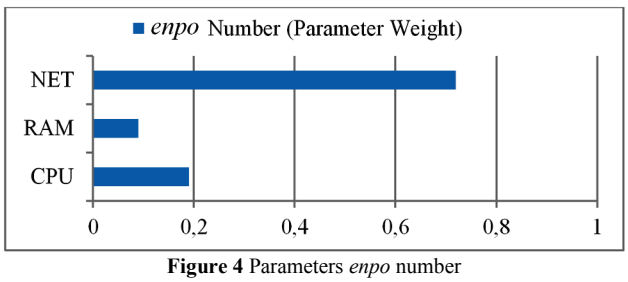

Figure 4 Parameters *enpo* number

In next stage, IAMP runs over a different set of parameter to find the virtual machine causing hotspot in PM^1 (Tab. 7). As shown, virtual machine VM_5^1 is the first candidate for migration. But, calculations indicated that by this replacement UF increases. Therefore, IAMP algorithm went to the next potential candidate VM_3^1 and relocated it from PM^1 to PM^5 at the beginning of 3rd time slot $t + 3\tau$. Without enabled IAMP over cluster, this node will be overloaded in the next time slot $t + 4\tau$.

Table 5 Averaged resource usage in time slot $t + 2\tau$ (before migration)

PM^j	cpu	ram	net
$Weight$	19	09	72
PM^1	55	72	71
PM^2	35	78	58
PM^3	70	20	45
PM^4	25	35	77
PM^5	73	15	45

Table 6 Ranking physical nodes

PM^j	Rank	Score (PM^j_{score})
PM^1	1	78,2
PM^2	3	42,5
PM^3	5	28,4
PM^4	2	67,7
PM^5	4	29,5

Table 7 Ranking virtual machines

VM_w^1	Rank	Score ($VM^1_{w_{\text{score}}}$)
VM_8^1	4	50
VM_3^1	2	73,4
VM_1^1	3	70
VM_5^1	1	75,8
VM_6^1	5	24,3
VM_7^1	7	5,5
VM_4^1	6	15,4

Average resource utilization of $PM^1$and PM^5after migration of VM_3^1 are presented in Tab. 8. After migration, UF was decreased from 0,28 to 0,22 which shows % 20,3 improvement in balancing cluster workloads.

Table 8 Averaged resource usage in time slot $t + 3\tau$ (after migration)

PM^j	cpu	ram	net
PM^1	50	62	51
PM^2	35	78	58
PM^3	70	20	45
PM^4	25	35	77
PM^5	78	25	65

6 Performance analysis

To analyse performance of our algorithm, we also measured the average response time of the cluster over hundreds of time slots under dynamic workloads and compared it with DRS and sandpiper algorithms. For more illustration, comparisons are characterized in two scenarios as follows:

6.1 Scenario one

To clarify IAMP ability over VMware DRS, we started the cluster with only network intensive virtual machines. In the other words, 3 heavily loaded, 2

moderately loaded and 2 lightly loaded NET-intensive virtual machines were distributed over 5 node cluster randomly. VMware Distributed Resource Scheduler could not keep system under-controlled because it is not *NIC* aware load balancing algorithm. Saturating network bandwidth on three nodes PM^1, PM^2 and PM^3 leads to increasing response time of these physical machines and as a result, average response time of whole cluster nodes increased from 25 ms to 150 ms only after 2 time slots. DRS can do nothing because it is network-unaware migration algorithm. DRS cluster overflew to 185 ms in time slot $t + 4\tau$ and caused *SLA* violations for virtual machines VM_1^1, VM_2^2 and VM_1^3. Fig. 5 shows average response time of the cluster for three algorithms from time slot t to $t + 12\tau$.

IAMP scheduling algorithm could control cluster *average response time* around 85 ms after *one* time slot. Load balancer engine initiates 3 migrations from PM^1 and PM^2 to underloaded machine PM^5. In the other hand, Sandpiper-enabled cluster decreased the *response time* to 75 ms. In this scenario, NET parameter has the highest weight because we do not have CPU and RAM intensive virtual machines over the cluster. Therefore, performances of Sandpiper and IAMP algorithms are close to each other.

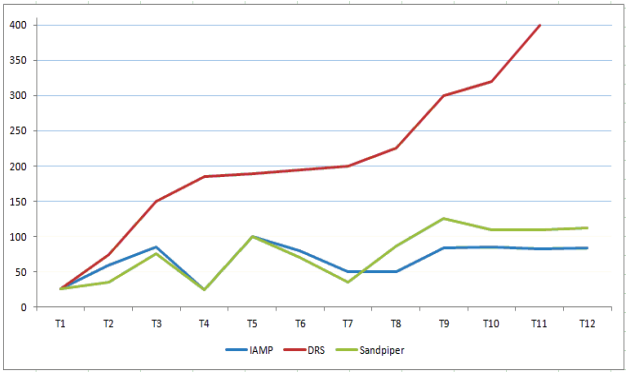

Figure 5 Average response time of the cluster with three algorithms

6.2 Scenario two

In time slot $t + 8\tau$, we added 15 memory intensive virtual machines to the cluster. Sandpiper cluster assumed all parameters equally weighted. After rising *average response time* to 125 ms in time slot $t + 9\tau$, Sandpiper detects hotspot because PM^2 encountered memory shortage and the migration initiates to decrease average response time to 110 ms in time slot $t + 11\tau$. Physical machinePM^3 is determined by Sandpiper as destination host. In contradictory, our algorithm selected PM^4 as a highly loaded server because it considers parameters weight and degree of resource impact on cluster *average response time*. With this new workload pattern, RAM weight is increased to 65 %, CPU stayed around 10% and NET decreased down to 25 %. Fig. 5 depicts this situation.

IAMP detected hotspot on PM^4 and through migrating VM_1^4 from physical machine $PM^4$to the least loaded serverPM^5, workload redistributed more efficient and *average response time* of the cluster decreased to 84 ms. Needless to say that workload of physical machine PM^5 increases because of hosting the migratory virtual machine VM_1^4. Fig. 6 indicates parameters weight over time slots. These weights are calculated by ENPO approach at the first of each time intervals.

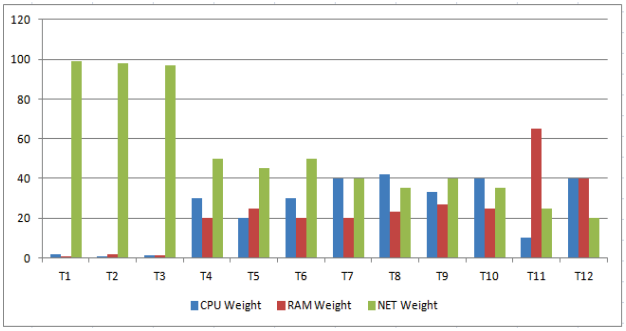

Figure 6 Parameters weight over 12 time intervals

Notice that Sandpiper does not talk about the weights of parameters in the decision process of migration; therefore its result differs from our algorithm. Fig. 7 shows physical machine's score with two algorithms (IAMP versus Sandpiper). To make comparisons more realistic, we normalized the score produced by IAMP and Sandpiper algorithms.

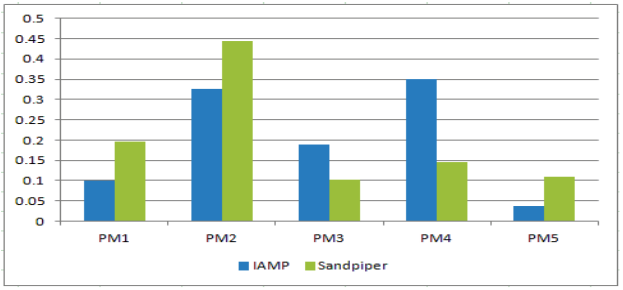

Figure 7 Physical node score under *IAMP* and *Sandpiper* algorithms

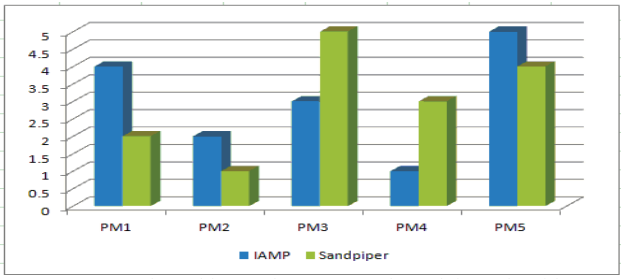

Figure 8 Node ranking under IAMP and Sandpiper algorithms

Fig. 8 indicates physical machines ranking. As shown, the results are completely different. That is, in Sandpiper, server PM^2 takes the first rank and physical host PM^3 is the last one. Therefore, node PM^2 is chosen as the source of migration and machinePM^3 as the destination, while by applying IAMP migration algorithm, PM^4 is chosen as the source of migration and server PM^5 receives migratory virtual machine. That is, when we consider the weights of the parameters as indicator of their importance, the results will be different and another physical machine takes the first rank and therefore is chosen as source candidate in migration process.

6.3 Algorithm performance discussion

As scenarios 1 and 2 show, IAMP algorithm reacts to the cluster hotspots quickly and more efficient. Migration of virtual machines decreases average response time of the cluster. The algorithm indicates bottlenecked resource precisely depending on system workload profile. It helps datacentre managerial board to run multi-demand services for business customers without SLA degradation risk.

This advantage leads to fewer penalties and more incomes. Administrators can keep the clusters load under control because this algorithm tags each resource with a weight number and monitors the most important ones.

Moreover, reservation of extra capacity and physical resources is not necessary. So, as a result, hardware cost will be decreased. IAMP engine could be applied for large clusters and datacentres.

7 Conclusion and future work

The purpose of the current study was to design an intelligent adaptive migration algorithm to provision and control dynamic and unpredictable customer services. We introduced IAMP algorithm as a live migration tool and studied its effect on reduction cluster *average response time* using XEN virtualization platform.

The proposed load balancing approach addressing completely dynamic data center environment in which, lots of services- encapsulated in VM containers- run over hundreds of physical servers and clusters. Demand profile of clusters change rapidly. That is, sometimes, average cpu usages of cluster nodes increase because of CPU-intensive applications, while in the next time, some memory-intensive services may be activated for new customers. Therefore, cluster bottleneck may change from CPU to RAM. Current static weighting approaches imprecisely remain on CPU as a bottleneck. To address this challenge, our algorithm considers variable impact factors for each resource in each time window.

We involved multi factors (CPU, RAM, NET) to find critical servers. These parameters non-linearly impact cluster response time. Moreover, impact factor of parameters is different and not static. This paper applied Neural Network approach to determine weights in each time window. An MLP neural network with 3 input neurons and 1 neuron in the output layer is developed to estimate parameter weights.

These measurements are typically in a time series format containing times stamped records of CPU, memory, and I/O bandwidth consumptions of physical servers. Performance of IAMP was evaluated along heterogeneous cluster environments. Based on performance analysis and real applications, this managerial framework improves the reaction time and performance of the virtualized cluster system around 20 % compared to current algorithms.

For small time windows, this method suffers from computational overhead and time-consuming calculations. In coming studies, developing effective parameters and using faster weighting methods such as entropy are considered. We also plan deploying our model for effective migration virtual machines over clouds and WAN connections.

Acknowledgement

The authors would like to acknowledge the Faculty of Electrical & Electronics, University of Tehran Polytechnic, and the High Performance and Cloud Computing laboratory for their support and contribution to this study.

8 References

[1] Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Wareld, A. Xen and the art of virtualization. // In SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles, New York, USA, ACM (2003), pp. 164-177.

[2] Gupta, D.; Gardner, R.; Cherkasova, L. QoS monitoring and performance profiling tool. // Technical Report HPL, HP Labs, (2005).

[3] Gupta, D.; Cherkasova, L.; Gardner, R.; Vahdat, A. Enforcing Performance Isolation across Virtual Machines in XEN. // In Proceedings of Middleware '06, October (2006).

[4] Nelson, M.; Lim, B.; Hutchins, G. Fast Transparent Migration for Virtual Machines, In Proceedings of USENIX, (2005).

[5] Clark, C.; Fraser, K.; Hand, S.; Hansen, J. G.; Jul, E.; Limpach, C.; Pratt, I.; Warfield, A. Live migration of virtual machines. // In Proceedings of NSDI (2005).

[6] XenMotion. Citrix Systems: http://www. xenserver5.com/xenmotion.php

[7] VMware Dynamic Resource Scheduler, http://www.vmware.com/products/vi/vc/drs.html.VMWare, Inc. drs_datasheet.pdf

[8] Hacking S.; Hudzia, B. Improving the live migration process of large enterprise applications, // in Proc. ACM International Workshop on Virtualization Technologies in Distributed Computing (VTDC'09), New York, USA, (2009), pp. 51-58.

[9] Menasce, D. A.; Bennani M. N. Autonomic virtualized environments, // IEEE International Conference on Autonomic and Autonomous Systems, (2006).

[10] Wood, T.; Shenoy, P.; Venkataramani, A.; Yousif, M. Black-box and Gray-box Strategies for Virtual Machine Migration, // Computer Networks Journal Special Issue on Virtualized Data Centers, (2009).

[11] Tarighi, M.; Motamedi, S. A.; Sharifian, S. A new model for virtual machine migration in virtualized cluster server based on Fuzzy Decision Making, // International Journal of telecommunications,1, 1(2010).

[12] Tarighi, M.; Motamedi, S.A.; Arianyan, E. Performance Improvement of Virtualized Cluster Computing System Using TOPSIS Algorithm, The 40th International Conference on Computers & Industrial Engineering (CIE-40), (2010), pp. 25-31.

[13] Andreolini, M.; Colajanni, M.; Nuccio, M. Kernel Based web Switches Providing Content Aware Routing. In proc. of IEEE Network computing & application symposium (NCA'03).

[14] Casalicchio E.; Colajanni, M. A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services". In Proc. Of Int'l World Wide Web Conference, May (2001).

[15] Alsmadi, M. K.; Bin Omar, K.; Noah, S. A.; Almarashdah, I. Performance Comparison of Multi-layer Perceptron algorithms in Neural Networks. Advance Computing Conference. // IACC 2009. IEEE International Digital Object Identifier. (2009), pp. 296-299.

[16] Patterson, D. W. Artificial neural networks. // Prentice Hall, New York, (1996).

[17] Monjezi, M.; Dehghani, H. Evaluation of effect of blasting pattern parameters on back break using neural networks. // International Journal of Rock Mechanics & Mining Sciences, 45, 8(2008), pp. 1446-1453.
[18] Hwang, C. L.; Yoon, K. Multiple attribute decision making: Methods and applications. // Berlin, Springer, (1981).
[19] Kikuchi, S.; Matsumoto, Y. Performance Modeling of Concurrent Live Migration Operations in Cloud Computing Systems Using PRISM Probabilistic Model Checker. International Conference on Cloud Computing (CLOUD), July 2011, pp. 49-56.
[20] Hsiang-Yao, C.; Kuan-Lung, H.; Chao-Tung, Y. A Dynamic Resource Allocation Model for Virtual Machine Management on Cloud. // Grid and Distributed Computing Communications in Computer and Information Science. 261, (2011), pp. 581-590.

Authors' addresses

Mohsen Tarighi
Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran

Seyed Ahmad Motamedi *(corresponding author)*
Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran

Saeed Sharifian
Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran