sciendo

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **19**, No 3
Sofia • 2019 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2019-0027

Enhanced Evolutionary Computing Assisted Robust SLA-Centric Load Balancing System for Mega Cloud Data Centers

P. Govardhan, P. Srinivasan
School of Information Technology and Engineering, VIT University, Vellore 632014, Tamil Nadu, India
E-mails: *govardhan.p2016@vitstudent.ac.in* *srinivasan.suriya@vit.ac.in*

***Abstract*:** *Considering significance of a robust and Quality of Service* (*QoS*) *centric cloud computing, virtualization assisted load-balancing has been found a potential solution. However, assuring optimal Virtual-Machine* (*VM*) *migration with minimum violation of Service-Level-Agreement* (*SLA*) *and QoS degradation has been the challenge for academia-industries. VM allocation or scheduling being an NP-hard problem has been solved by numerous heuristic approaches such as classical Genetic Algorithm* (*GA*), *Ant Colony Optimization* (*ACO*), *etc. However they have been found confined due to local minima and convergence issues, especially for Mega Data Centres* (*MDCs*). *To alleviate such issues, in this paper an enhanced Evolutionary Computing algorithm named Adaptive Re-sampling GA* (*ARGA*) *algorithm has been developed that in conjunction with a stochastic prediction based dynamic load–measurement and Maximum Correlation* (*MC*) *assisted VM selection perform optimal load balancing over IaaS MDC infrastructures. The proposed ARGA VM allocation model with dual-level dynamic threshold assisted load estimation and MC based VM selection has exhibited lower SLA violation, performance degradation, downtime and minimum VM migration as compared to classical ACO based load balancing.*

***Keywords*:** *Load-balancing, virtualization, Adaptive Re-sampling Genetic Algorithm* (*ARGA*), *SLA provision.*

1. Introduction

The exponential rise in internet technologies and allied advanced computing systems has given rise to a broadened technological horizon called Cloud computing that facilitates decentralized scalable services for different purposes. Cloud computing has up surged significantly in the last few years to meet remote data or system access irrespective of geographical boundaries and hardware dependencies [2]. Increasing demands need providing more and more computing and storage infrastructures, often termed as data centers. Cloud service providers require ensuring Service-Level-Agreement (SLA) by sticking to the QoS delivery to the customers while ensuring cost-effective, eco-friendly and reliable services [4]. Some of the key challenges in

current cloud computing scenarios are SLA provision, QoS delivery, cost-efficient services, etc. Exploring in depth one of the key issues in modern cloud computing arena is load balancing that ensures fault-tolerant and reliable resource provision to the tasks or services continuously. Virtualization employs Virtual Machines (VMs) as computing resources to solve the problem of Dynamic Resource Management (DRM) in cloud computing environment [5]. To achieve optimal load balancing under non-linear use patterns virtualization enables live migrations between VMs and the Physical Machines (PMs), often called hosts on the basis of the change of load. Different VM monitors like XEN [6], KVM [7] can help efficient VM migrations on the different hosts. In practice, the load on each node of VM might vary according to the service type and user demand density [8], and hence random selection of hosts might lead into overload condition. On contrary, there can be the possibility that certain node or host might undergo under load condition with low resource consumption. Such load imbalance issue often leads either SLA violation, especially downtime or energy exhaustion and hence requires certain efficient load balancing solution [9]. Such load balancing problems become prevalent and more complicate in case of highly dynamic and heterogeneous resource demands [5]. Under such scenarios to augment resource utilization, VMs are required to be effectively allocated and hence SLA sensitive load balancing must be guaranteed [5].

As depicted in Fig. 1, the scheduler predominantly comprises two distinct elements named central controller and Local Migration Controller (LMC). Here, the central controller retrieves the information about entire PM resources utilization and initiates VM migration on the basis of certain predefined policies so as to perform load balancing and high resource utilization. On the other hand LMC controller sends information about its physical resources utilization to the center controller and retrieves instructions to initiate migration. However, this approach often undergoes adversaries when to decide where to migrate the task while ensuring SLA intact and QoS provision as goal. Improper migration scheduling can force host nodes to undergo overload or underload again and again and hence can adversely affect the performance. To solve such issues approaches like overload detection and distributed computing approaches have been explored in a number of researches.

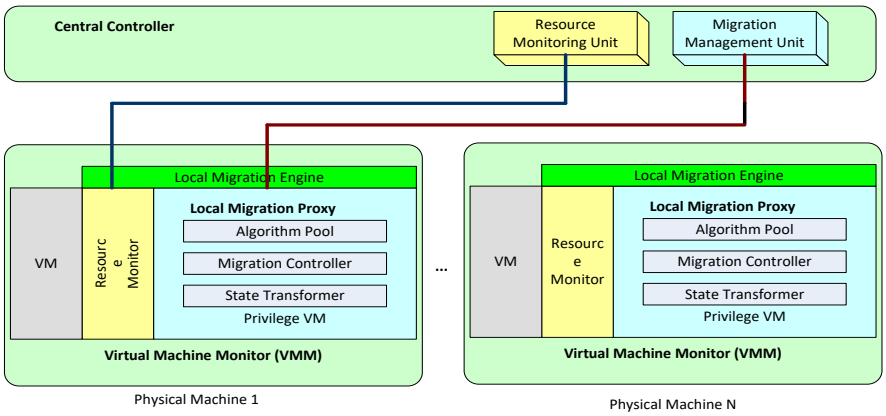

Fig. 1. VM live migration for load balancing in cloud computing

To solve this problem numerous meta-heuristic approaches like Particle Swarm Optimization (PSO) [11-15], Genetic Algorithm (GA) [16-19], Ant Colony System (ACS) [20-22], Cuckoo Search Optimization (CSO) [24], Honey Bee Foraging (HBF) [7, 10, 23, 25], etc., have been proposed. Undeniably, these approaches have made effort to augment VM placement policy however lacks dealing with local minima and convergence, which is predominant in major meta-heuristic approaches. In addition, most of the approaches use static threshold based CPU utilization for triggering VM migration, which seems confined due to dynamic nature of resource demands and varying utilization patterns of the allied tasks. Such problems can be dealt with certain predictive overload detection system that in conjunction with certain optimal VM selection policy such as the Minimum Migration Time (MMT) or the Maximum Correlation (MC) can achieve SLA sensitive VM migration and load balancing. Furthermore, improving genetic parameter selection (i.e., crossover probability and mutation probability) by applying dynamic fitness value for solution (search space) re-sampling can effectively alleviate local minima and convergence issue. Hence, here a highly robust enhanced Evolutionary Computing (EC) based load-balancing model is developed that exploits virtualization and live migration concept to offload tasks from a host undergoing overload condition. Our proposed approach employs dynamic load assessment method using stochastic prediction method followed by MMT based VM selection and Adaptive Re-sampling Genetic Algorithm (ARGA) for VM migration scheduling or VM allocation. Unlike classical GA based migration policy, ARGA applies dynamic PC and PM (value) estimation by exploiting the number of PMs having same fitness value. Noticeably, the concept of ARGA originates from the fact that during real-time computation, especially in Mega Data Centers (MDC) there probability may exist that the multiple hosts can have the same fitness value and hence assigning VM to such equi-fitness nodes becomes NP-hard. Moreover, higher iterations (say, generation size in GA) often overburden computations to exhaust energy as well as response time. In such cases, ARGA can exploit the information about the number of hosts with same fitness to dynamically calculate PC and PM values that as a result could help reducing both population size as well as iterations to perform timely and computationally efficient VM migration. This approach can achieve higher SLA provision even for large scale data centres. In this paper, the overall proposed system has been developed using JAVA programming language, which has been simulated over CloudSim simulator with PlanetLab [10] cloud trace data.

2. Problem formulation

In this paper the concept of virtualization or live VM-migration has been applied to alleviate the aforementioned issues. Noticeably, unlike classical virtualization based approaches where certain predefined threshold also called static threshold based overloading estimation is used, in our proposed load balancing model a predictive or stochastic load prediction model is developed that in conjunction with a suitable VM selection and placement policy ensures SLA centric dynamic load balancing. Considering NP-hard issue of the classical bin packing based VM allocation strategies

numerous meta-heuristic models such as GA, PSO, ACO, BFA etc. have been developed where these heuristic models predominantly perform VL allocation. In other words, it identifies the suitable host to migrate tasks or VMs. However, most of the classical EC algorithms undergo local minima and convergence, in addition to the significantly huge computational time. Considering this fact as motivation, in this research paper a novel EC algorithm named Adaptive Genetic Algorithm with iterative re-sampling has been developed. The proposed VM migration model ensures time-efficient migration thus supporting SLA provision. Noticeably, the overall process of virtualization based load balancing encompasses three consecutive phases, including overload detection, VM selection and VM migration.

In this research we have focused on augmenting each participating phase where at first we have intended to implement a stochastic or predictive model for load estimation at each node (say VM). In the following phase of implementation considering deadline sensitive as well as demand sensitive resource management Minimum Migration Time (MMT) and Maximum Correlation (MC) based VM selection model has been developed. To ensure SLA and QoS oriented load balancing, our proposed ARGA heuristic model has been applied for VM migration scheduling or VM placement purposes that intends to map VM (to be migrated) and optimal possible host in such manner that it enables minimum migration and SLA Violation (SLAV). Here, the use of ARGA significantly alleviates the problem of local minima and convergence even with the large scale MDC and consequently strengthens QoS and SLA provision for mega cloud infrastructures. The detailed discussion of the proposed method is given in the subsequent sections.

3. Our contribution

This section primarily discusses the proposed enhanced evolutionary computing assisted SLA centric load balancing model.

3.1. Multi-controller assisted dynamic cloud monitoring and information exchange

In practice, cloud computing can be stated as a highly dynamic computing platform with varying load or demands conditions. In such cases updating (cloud) network conditions and making adaptive migration decision is of paramount significance. Considering this fact, in this paper multi-controller assisted dynamic cloud monitoring and information exchange model has been developed. In our model being proposed two controllers named Local VM Monitoring Controller (LVMC) and Global Migration Controller (GMC) with distinct (dual) roles have been applied. The detailed discussion of the incorporated controllers is given in this section. Before discussing the proposed controller model, a snippet of the cloud configuration and components is given as it is follow.

With intention to develop load balancing model for a large scale cloud data centre or infrastructure, also called Mega Data Centers (MDC), in our model we consider an IAAS with N heterogeneous hosts or PMs, where each PM has been assessed in terms of its memory utilization and CPU performance. Noticeably, these node characteristics

(parameters) have been defined in terms of the Millions Instructions Per Second (MIPS). Furthermore, in addition to the MIPS, bandwidth ad RAM utilization has been obtained for each node (VMs as well as PMs) which have been monitored to perform load-balancing decision. The considered MDC system does not have specific inbuilt storage provision; we have applied Network Attached Storage (NAS) or Storage Area Network (SAN) that helps enabling live VM migration for intended load balancing. In this model it is hypothesized that there exists no significant information under exchange or migration (pertaining to the task or application under process at the VMs). It signifies the proposed cloud model to be like application-agnostic paradigm. In function, there can be numerous simultaneously working users requesting for M heterogeneous VMs to access or use certain needed Processing Elements (PEs) or MIPS. In this way, the load balancing or VM migration can be the result of the need of user's application demand and therefore load balancing by migrating tasks or VMs to a single host can be the consolidation problem for the multiple simultaneously working users with heterogeneous resource demands. During computation the VMs can migrate from one host to another so as to accommodate ongoing task and allied resource demands, however retaining intact operating environment during migration turns into indefinite condition thus needs certain optimal VM migration model. On the other hand, during migration there can be a situation where one host might get significantly huge workloads thus making it overloaded. Meanwhile, a host can also be undergoing under load condition where it can be functional even with very low or without any ongoing task execution. In such scenarios, developing a robust load balancing model becomes inevitable that could identify a host under overload or under load condition and can migrate tasks to the appropriate host without causing any significant SLA violation or QoS loss. Towards this goal, the implementation of multiple controllers at the different layers of the IaaS cloud infrastructure can be vital.

As already indicated, our proposed virtualization based load balancing model employs two distinct controllers termed LVMC and GMC at the different layers of the cloud architecture where it intends to augment overall scalability and timely event-decision control under dynamic load conditions. A snippet of the proposed controller model and its specific task towards SLA-centric load balancing is depicted in Fig. 1. As depicted here, LVMC functions on each node or VM where it operates as VMM manager. It performs continuous CPU/memory utilization monitoring over operating nodes that eventually helps enabling dynamic load (overload or under load) detection on host nodes. Once detecting any host undergoing overload condition, LVMC triggers VM selection to estimate or identify the VM(s) to be offloaded from that host to reduce overloading. Now, considering the fact that the improper migration scheduling could cause iterative overload condition on the hosts and therefore it becomes inevitable to develop certain robust VM migration scheduling scheme that could dynamically identify the optimal host to which the VM could be migrated without affecting ongoing task or SLA. With this motive, in this paper we have applied GMS that continuously gathers node's information from the comprising VMs and executes optimal VM placement process. Once moving the selected VM from the overloaded host, VMM updates the node characteristics and exchanges information to the GMC to continue optimal load balancing and SLA assurance. The detailed

discussion of the proposed virtualization assisted load balancing is given in the subsequent sections.

3.2. Stochastic prediction assisted load assessment

In this phase, our proposed load balancing model assesses and estimates load condition of each host node and identifies under-loaded and overloaded nodes. Considering the prime objective of efficient load balancing once identifying a node as under-loaded it is intended to off-load either all VMs and migrate it to other host so as to preserve resource or migrate other VMs from other host to it (current under-loaded host) so as to use optimal resource. On contrary, in case of overloaded node, it is intended to offload VMs and migrate it to other hosts so as to alleviate SLA violation probability on that host.

3.2.1. Host under-load detection

Here, a PM with minimum or significantly less workload (i.e., resource utilization) as compared to certain defined threshold or in comparison to other PMs, is identified as under load host. Identifying a host or PM as under-loaded, its connected VMs are migrated to the other PM(s) strategically while assuring that such migration does not impose overloading on target PM or host. Noticeably, once migrating overall connected VMs of the under loaded host, it can either be switched-off or can be availed for other VMs to continue SLA-centric task execution and thus can significantly enhance energy-efficiency of the cloud system. Our proposed EC assisted load-balancing model ensures that the migration of VMs does not impose any further overloading on the target host. Furthermore, the source host remains active until all VMs are migrated and accommodated to the target host(s), so as to assure SLA assurance, especially to avoid any probable downtime.

3.2.2. Dual-level dynamic thresholding assisted host overload detection

To identify overloaded host, here a stochastic prediction model has been developed. In this approach, each host executes an algorithm that periodically estimates (current) load condition at the host that helps identifying overloaded node to support SLA provision. In our proposed method each host's CPU utilization has been estimated so as to detect whether the host is overloaded or not. In majority of the existing works [11-15, 20-22], authors have applied static threshold to identify overloaded node. However, in practice, cloud infrastructure, especially IaaS undergoes dynamic load condition and therefore static threshold based overload node identification can be an optimal solution. In this paper, we have used dynamic CPU utilization to perform overload detection. Noticeably, in our proposed threshold scheme dynamically adjusts the CPU utilization threshold based on the variations of CPU utilization. The proposed method hypothesizes that more deviation can be referred as lower upper CPU utilization (threshold). Typically, higher deviation in use pattern raises the likelihood of almost all or 100% CPU utilization and in such cases the probability of SLA violation gets significantly increased. To alleviate such issues a hybrid model has been developed that considers both inter-service (task) relation as well as variational

information to perform dynamic thresholding. We have derived a hybrid model where both InterQuartile Range (IQR) [17] and modified Local Regression (LRR) approaches have been applied to estimate dynamic CPU threshold. IQR which is a statistical dispersion method signifies the in between the first and the third quartile (1). Thus, estimating the IQR value, we have applied (2) to estimate the upper threshold of CPU utilization per node:

(1) $$\text{IQR} = \mathcal{Q}_3 - \mathcal{Q}_1,$$

(2) $$\mathcal{T}_u = 1 - \mathcal{s}.\text{IQR}.$$

However, considering the dynamism of load conditions and probable (resource utilization) fluctuations in the same ongoing task, there can be significant affect on the upper threshold estimation (2). Such inaccuracy might lead wrong decision and hence causing SLA violation. Considering this fact, in this paper a dual-level thresholding mechanism has been applied, where in this first level IQR based $\mathcal{T}_u$ has been estimated, while in the second level of thresholding an enhanced LRR model has been applied. In our proposed thresholding model LRR performs fitting of a trend polynomial to the preceding $\mathcal{k}$ CPU utilization values, obtained using equation $\hat{\mathcal{g}}(x) = \hat{\mathcal{a}} + \hat{\mathcal{b}}\, x$ each new observation values. Estimating the observation values we have derived the value of the next observation $\hat{\mathcal{g}}(x_{\mathcal{k}+1})$. In order to offload a host undergoing overloading we have used following two conditions:

(3) $$\mathcal{s}.\, \hat{\mathcal{g}}(x_{\mathcal{k}+1}) \geq 1,$$
$$x_{\mathcal{k}+1} - x_{\mathcal{k}} \leq \mathcal{t}_m.$$

In (3), $\mathcal{s} \in \mathbb{R}^+$ refers the highest tolerance of a host node, while the maximum time needed to migrate a VM from host d $\mathcal{t}_m$ represents the maximum time required for migrating VM from the host node.

As already stated the classical Loess method [17] based LRR often gives inferior performance thus to alleviate this issue, in this paper the classical Least-Square (LR) model has been augmented to the bisquare. The LR enhancement has been performed in iterative manner so as to calculate the initial fitting where we have estimated tricube weights using a Tricube Weight Function (TWF). We estimated fitting parameter at x_i to retrieve the fitted values by using $\hat{y}_i$. Thus, the residual value is estimated as $\varepsilon_i = y_i - \hat{y}_i$. Now, once estimating the values of x_i and y_i, has been assigned to a robustness factor $\mathcal{R}_i$ that primarily relies on the value of ε_i. We have used the next equation, to estimate the robustness factor $\mathcal{R}_i$:

(4) $$\mathcal{R}_i = \mathcal{B}\left(\frac{\hat{\varepsilon}_1}{6\mathcal{s}}\right).$$

Here, each observation values are assigned $\mathcal{R}_i$, where $\mathcal{B}(.)$ states the bisquare weight function and $\mathcal{s}$ signifies the Mediun Absolute Deviation (MAD) to perform least square fitting. Mathematically $\mathcal{B}(.)$ is estimated using (5); in (4), the value of $\mathcal{s}$ is estimated using (6):

(5) $$\mathcal{B}(.) = \begin{cases} (1-\mathcal{u}^2)^2 & \text{if } |\mathcal{u}| < 1, \\ 0 & \text{otherwise,} \end{cases}$$

(6) $$\mathcal{s} = \text{mediun}\, |\hat{\varepsilon}_i|.$$

In this manner, applying (3) for the estimated trend line, the subsequent has been predicted. In case of inequalities, host can be identified as overloaded host. Once identifying the overloaded host, the local controller identifies the VM to be migrated

from the overloaded host so as to retain SLA provision; this process is defined as VM selection, given in the next section.

3.3. SLA centric VM selection

In this paper we have applied two methods MMT and Maximum Correlation (MC) to select VMs so as to off load it from the over-loaded host. A snippet of these is given as follows:

3.3.1. Minimum Migration Time (MMT) model

As name signifies MMT policy recommends selecting a VM which could be migrated to a target host in the minimum possible time. This is because moving a VM with minimum migration period can avoid downtime for that VM specific task and thus the overall SLA could be retained optimal. Undeniably, MMT based VM selection can help migrating certain task or associated VM to a near possible or suitable host; however this approach doesn't consider on-going application specific scenario. On the other hand, exploiting correlation in between the CPU utilization and the resource consumption, correlation between tasks consuming resource and resource available on host can be an effective and potential solution to retain SLA assurance as well as computational efficiency and/or reliability. With this motive, in this paper Maximum Correlation (MC) based VM migration model has been developed.

A brief of the proposed MC based VM migration model is given in the next section.

3.3.2. Maximum Correlation (MC) model

In general, MC based approach considers that the high correlation in between CPU utilization by VM on certain oversubscribed host does reflect higher likelihood of becoming overloaded. In this paper a VMs with the highest correlation between the CPU utilization patterns have been migrated from the current host to the targeted host so as to preserve task-reliability and minimum downtime. Here, we have incorporated the approach of the multiple correlation coefficients that helps assessing intra-VM correlation on a host and its CPU utilization pattern. In our proposed method the (multiple) correlation coefficients to be used are highly related to the squared correlation in between the actual and the predicted values of the dependent variable. Let, $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$ be the CPU utilization patterns for n VMs assisted or connected to a host. Consider that $\mathcal{Y}$ be the VM to be migrated. Here, the arbitrary independent variables are $n-1$ and $\mathcal{Y}$ is the independent variable. Now, with this configuration, our model intends to calculate the correlation in between $\mathcal{Y}$ and $n-1$. With these two variables a matrix can be derived as $(n-1) \times n$. With these values, the observation vector $(n-1) \times 1$ for $\mathcal{Y}$ be y. In other words, the observation vectors derived for the CPU utilization of all connected hosts can be obtained using

$$\mathcal{X} = \begin{bmatrix} 1 & \mathcal{X}_{1,1} & \cdots & \mathcal{X}_{1,n-1} \\ \vdots & \vdots & & \ddots \ \vdots \\ 1 & \mathcal{X}_{n-1,1} & \cdots & \mathrm{x}_{n-1,n-1} \end{bmatrix}, \quad \mathcal{Y} = \begin{bmatrix} \mathcal{Y}_1 \\ \vdots \\ \mathcal{Y}_{\mathrm{n}} \end{bmatrix}. \tag{7}$$

Noticeably, for any ongoing task the first column as indicated in (7) would be 1 for all instances. In our model, the predicted values of $\mathcal{Y}$ is $\hat{\mathcal{Y}}$, which has been estimated as the product of $\mathcal{X}$ and $\mathcal{b}$. Mathematically,

(8) $$\hat{\mathcal{Y}} = \mathcal{X}\mathcal{b},$$

where $\mathcal{b} = \left(\mathcal{X}^{\mathcal{T}}\mathcal{X}\right)^{-1}\mathcal{X}^{\mathcal{T}}\mathcal{Y}$.

Once estimating the predicted values, we have estimated the correlation coefficients, $\mathcal{R}^2{}_{y,1,\ldots,\, n-1'}$, using

(9) $$\mathcal{R}^2{}_{y,x_1,\ldots,x_{n-1}} = \frac{\sum_{i=1}^{n}(y_i - m_y)^2\left(\hat{y}_i - m_{\hat{y}}\right)^2}{\sum_{i=1}^{n}(y_i - m_y)^2 \sum_{i=1}^{n}\left(\hat{y}_i - m_{\hat{y}}\right)^{2'}}.$$

In (9), the variables m_y and $m_{\hat{y}}$ signify the observation means of $\mathcal{Y}$ and $\hat{\mathcal{Y}}$ correspondingly. We have obtained multiple correlation coefficients for all instances $\mathcal{X}_i$ as $\mathcal{R}^2{}_{x_i,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n}$, an thus the VM selection has been performed using

(10) $$v \in \mathrm{VM}_j | \forall_a \in \mathcal{V}_j,\ \mathcal{R}^2{}_{x_{vm},x_1,\ldots,x_{vm-1},x_{v+1}x_n} \geq \mathcal{R}^2{}_{x_v,x_1,\ldots,x_{a-1},x_{a+1},\ldots,x_n}.$$

3.3.3. Enhanced AGA based VM migration for SLA-centric load balancing

Once performing VM selection one of the critical tasks surfaces is migrating the (selected) VM to the other host without imposing overload (on target host) and incurring the downtime that adversely affects the QoS and/or SLA provision. In some of the existing works VM migration has been considered as the problem of VM consolidation where "Bin-packing" approach [63] is applied to migrate VMs to the nearest target host up to its maximum capacity (until it doesn't undergo overload). However, this approach cannot be an effective method only because it might cause iterative overloading at the host and hence can adversely affect the SLA. On the other hand, in a large-scale cloud infrastructure often called MDC there can be "Heterogeneous Cloud Configuration (HCC)" that can have hosts with different memory capacity. Similarly, the VMs or allied tasks too can have different demands or bin sizes. In such cases merely using CPU utilization based migration policy can't be effective to achieve SLA centric load-balancing. Literatures [21] reveal that migrating VM to a suitable host or PM is a NP-hard problem, which can be significantly solved using meta-heuristics models such as Evolutionary Computing (EC). Towards this objective a few efforts have been made to use PSO [11-15], ACS [20-22], GA [16-19], HBF [23], CS [24], etc., where these approaches have been applied mainly for identifying the sub-optimal solution (i.e., suitable host). Undeniably, these approaches require certain Objective Functions (OF) to maintain optimal migration scheduling, for which CPU availability has been a most-used approach, though, in this paper we intend to use dynamic node (host) parameters such as bandwidth availability as well as CPU utilization at VM to perform migration decision (say, OF). VM migration or allocation policy signifies a problem of bin packing with varied host configuration (often called bin sizes (CPU utilization), items (VMs with distinct ongoing processes) and prices (it can be computational cost, energy consumption or SLA). Being an NP-hard problem, VM allocation requires exploiting multiple network parameters to select certain optimal host so as to accommodate VMs to be migrated from overloaded host. As stated, to solve the problem of NP-hard

numerous EC algorithms or meta-heuristic models, such as ACO, PSO, CS, HBF, GA, etc., have been applied. However, majority of these classical EC algorithms suffers local minima and convergence issues. Furthermore, the use of static "stopping criteria" and significantly huge "iterations" make these existing approaches undergo resource and time exhaustion and hence confined for online decision purposes. To solve such issues, in this paper a novel and robust EC model named Adaptive Genetic Algorithm (AGA) which has been further augmented as Adaptive Re-sampling Genetic Algorithm (ARGA) has been developed. The proposed ARGA model intends to alleviate the local minima and convergence issues in addition to the Adaptive Genetic Parameter Scheduling (AGPS) that estimates the probability of crossover and mutation dynamically by exploiting the number of chromosomes having same fitness value. This method can enable optimal solution retrieval even with lower iterations without causing any local minima and convergence problem. In ARGA, re-sampling refers an iterative method where based on the "number of chromosomes having same fitness value (i.e., OF)" it re-samples search space or solution set and thus reduces the size of search space. This makes overall computation more accurate and swift. The detailed discussion of the proposed ARGA based VM migration and load-balancing is given in the subsequent sections.

3.3.4. ARGA assisted VM placement and Load balancing

Since, our proposed virtualization based VM migration model incorporates multiple heterogeneous hosts and functional VMs simultaneously it has been simulated over CloudSim, a well-known simulator. Here, each host possesses one or multiple Processing Elements (PEs) and functional VMs on a host can have one or multiple (functional) cloudlets. To perform load balancing in our proposed model, user requests are presented as cloudlets, where the resource needed or demanded by a user (say, cloudlet) is defined in terms of Million Instructions Per Second (MIPS). Thus, considering the demands of each autonomously functional VMs, PMs and allied tasks, our proposed ARGA model requires identifying a suitable host to accommodate each demands without causing any significant SLA violation. In this approach ARGA exploits all connected hosts as input and generates dynamic resource use-map, where the overall MIPS is split into different network components (i.e., VMs and hosts). ARGA performs VM migration based on the upper threshold values as it enables satisfying transient variations in resource demand by functional VMs on a host. To perform VM migration or load-balancing scheduling CPU utilization pattern of the VMs in conjunction with host's resource availability have been taken into consideration.

Recalling the functional paradigm of the native GA which at first generates (random) population as the initial solution, we intend to identify the optimal or sub-optimal host to accommodate a VM's MIPS demands while assuring minimum number of migrations, minimum SLA violation and downtime. In our proposed model, ARGA at first generates a set of population randomly where each population element can be stated to be a tree encompassing GMC as the root, hosts from the next level nodes and VMs as the Child host nodes (Fig. 2).

In this method, for individual mapping ARGA calculates the total resource consumption (in terms of MIPS) across the MDC deployed. Now, ARGA considers previous patterns of the VM migration mapping onto the hosts, CPU demand pattern (in MIPS), number of migrations, probable VM mapping, hosts resource availability as the population, where it intends to identify a host with sufficient MIPS and minimum migration as well as SLAV. Once generating the initial population, ARGA selects two VMs mapping possessing minimum migration time and number of migrations, which have been further processed for GA functions such as crossover, fitness estimation, mutation, and selection, etc. Now, the mapping obtained for demands (i.e., VMs demands) onto the host(s) is appended to the population based on the fitness values or CPU availability. In our proposed mode the crossover operator (P_c) exhibits host selection by assessing CPU utilization pattern of each host and allied VMs mapping information. In our proposed model, GA parameters P_c and P_m signifying crossover and mutation probability, respectively intends to reduce the solution space (or the host nodes) by migrating offloaded (from over-loaded host) and/or accommodating VMs to the hosts with minimum migration counts and SLAV.

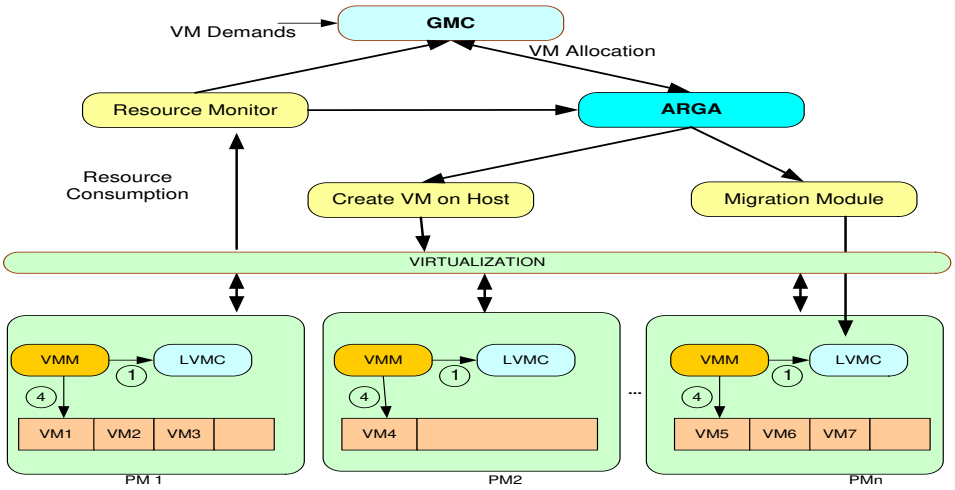

Fig. 2. Implementation model of the proposed ARGA assisted load balancing

Unlike traditional GA algorithm, our proposed ARGA VM placement policy employs dynamic or adaptive genetic parameters (i.e., P_c and P_m) that not only alleviates that problem of local minima and convergence (over-fitting and under-fitting) but also reduces search space (say, solution space) and hence eventually achieves time-efficient load balancing. Unlike classical GA where a predefined stopping criteria such as the number of generations or iterations (often selected as 500 or more) ARGA considers the number of chromosomes (solutions) having same fitness value. This contribution can be of utmost significance, especially in the large-scale cloud infrastructure or MDC where hosts can in large numbers and hence search space or solution set might keep on bulging over generations. In addition, multiple solutions with the same fitness value could force model to undergo pre-mature convergence. In such cases the implementation of our proposed ARGA scheduling

model with adaptive parameter (i.e., P_{c} and P_{m}) selection can achieve optimal migration scheduling. Mathematically, the dynamic update of P_{c} and P_{m} is given by

$$(P_{\mathrm{c}})_{k+1} = (P_{\mathrm{c}})_k - \frac{C_1 * N_{\mathrm{SSF}}}{7}, \tag{11}$$

$$(P_{\mathrm{m}})_{k+1} = (P_{\mathrm{m}})_k - \frac{C_2 * N_{\mathrm{SSF}}}{7}.$$

In above equation the variables $(P_{\mathrm{c}})_{k+1}$ and $(P_{\mathrm{m}})_{k+1}$ signify the updated crossover and mutation probability, while its current probability is given by $(P_{\mathrm{c}})_k$ and $(P_{\mathrm{m}})_k$. Furthermore, we have considered static values for the coefficient parameters c_1 and c_2 as 0.1 and 0.01, respectively. The parameter N_{SSF} signifies the total chromosome (set of solution) having similar fitness value. Now, to implement ARGA model for VM placement, recalling the fact that any heuristic algorithm (including our proposed ARGA algorithm (modified GA)) requires certain goal-centric objective function to perform computation and retrieve optimal solution, we have estimated OF by exploiting previous VM mapping information and host's CPU utilization pattern. The detailed discussion of the estimation model is given as follows:

Consider that the hosts across MDC be $\mathcal{PM} = \{\mathcal{pm}_1, \mathcal{pm}_2, \mathcal{pm}_3, \cdots, \mathcal{pm}_m\}$. Let, $\mathcal{pm}_i$ be the host node $1 \leq i \leq m$ signifying i-th PM. Similarly, the set of VMs operating across the MDC be $\mathrm{VM}_i = \{\mathcal{vm}_1, \mathcal{vm}_2, \mathcal{vm}_3, \cdots, \mathcal{vm}_{n,i}\}$ and the i-th VM connected to the j-th host be $\mathcal{vm}_{j,i}$. Let, $x_{i,j}$ be the binary factor stating whether the i-th VM is connected to the j-th host. Now, let $\mathcal{P}_{r,i}$ be the resource availability of MIPS available at the j-th host. Now, let the MIPS or CPU demanded by the i-th VM be $\mathcal{v}_{r,j}$. With this operating scenario, the total load on a host can be defined as the cumulative load pertaining to all connected VMs (at that host). Let, $\mathcal{T}$ be the duration of the previous (event) observations. In such manner, to perform time-series analysis the sub-intervals can be retrieved by dividing total span $\mathcal{T}$ into $q - 1$ sub-intervals. Mathematically,

$$\mathcal{T} = \left[(t_2 - t_1)(t_3 - t_2) \cdots \left(t_q - t_{q-1}\right)\right], \tag{12}$$

where the time slot $(t_k - t_{k-1})$ refers the duration k, where for each slot (i.e., k), the total CPU utilization at a host can be obtained using (13). Mathematically,

$$\mathrm{CPU}_{i,\mathrm{Util}}(k) = \sum_{j=1}^{n} \mathcal{vm}_{\mathrm{CPU},j} / \mathcal{pm}_{\mathrm{CPU},j}, \tag{13}$$

where k states the duration for which the MIPS or CPU utilization has been collected. Here, we have estimated average CPU utilization to make scheduling appropriate or to avoid any over-fitting or under-fitting probability. Mathematically, the average CPU utilization is obtained using

$$\mathcal{pm}_{i,\mathrm{AvgUtil}} = \sum_{t-t_k}^{t_k - n} \mathcal{pm}_{i,\mathrm{Util}}(t)/(q - 1). \tag{14}$$

Here the denominator component $q - 1$ signifies the total number of sub-intervals in $\mathcal{T}$ time period.

Let, $\mathcal{pm}_i$,be the CPU available of j-th host during t_k, and hence the resource available can be obtained for the using current CPU utilization information at a host. Let, $\mathcal{pm}_i C^{(k)}$ be the CPU consumption at the j-th host during the previous time interval. Mathematically,

$$\mathcal{pm}_i C^{(k)} = \mathcal{pm}_{iw}(k - 1) + (\mathcal{pm}_{iw}(k - 1) + \\ +(\mathcal{pm}_{iw}(k))(t_k - t_k - 1). \tag{15}$$

The CPU utilization $C(\mathcal{pm}_j)$ at a host $\mathcal{pm}_j$ can be obtained in terms of the CPU usage $\text{CPU}_{i,\text{Util}}(\mathcal{k})$. Mathematically,

(16) $$C(\mathcal{pm}_j) = \mathcal{K}_j \,.\, c_j^{\max} + (1 - \mathcal{k}_j) \,.\, c_j^{\max} \,.\, \text{CPU}_{i,\text{Util}}(\mathcal{k}).$$

Here the variable $\mathcal{K}_j$ signifies the MIPS consumed when the host $\mathcal{pm}_j$ was in idle case, while $c_j^{\max}$ is the MIPS used (at $\mathcal{pm}_j$) when it was being used completely. $\text{CPU}_{i,\text{Util}}(\mathcal{k})$ states the CPU utilization by $\mathcal{pm}_j$ at certain time instant $\mathcal{k}$. Estimating above parameters, the CPU utilization at each host has been obtained. Now, the overall CPU utilization at a host $H_C(\mathcal{k})$ during overall process duration can be obtained using the following equations:

(17) $$\mathcal{D}_C(\mathcal{k}) = \sum_{i=1}^{m} \mathcal{pm}_i C^{(\mathcal{k})}.$$

Here, in our proposed ARGA assisted load-balancing model VM migration problem has been solved by deriving a set of mapping from $\mathcal{vm}$ to the host(s) $(\mathcal{pm})$ while reducing of (18), in such manner that

(18) $$\forall_i \sum_{j=1}^{m} x_{ij-1},$$

(19) $$\forall_j \sum_{i=1}^{n} \mathcal{vm}_{CPU,i} \mathcal{X}_{ij} \leq \mathcal{pm}_{\text{CPU},j}.$$

Noticeably, in our proposed ARGA assisted VM migration model once the stopping criteria has been met, the list of sorted VMs which need to be migrated and the list of PMs having sufficient resources to accommodate aforesaid VMs are obtained. Now, obtaining this list ARGA matches these two lists where it intends to identify most suitable PM where to migrate the VM. To achieve an optimal mapping relationship we have derived a function called Fitness-Function (FF), which is obtained using

(20) $$\text{FF}(\mathcal{vm}_i, \mathcal{pm}_i) = \frac{\mathcal{pm}_{\text{CPU},i} - \mathcal{vm}_{\text{CPU},i}}{\mathcal{vm}_{\text{CPU},i}}.$$

Here $\mathcal{pm}_{\text{CPU},j}$ represents the surplus resources available at the j-th host (PM), while $\mathcal{vm}_{\text{CPU},i}$ states the resources demanded by ith VM, $\mathcal{vm}_{\text{CPU},i}$. In case of $\text{FF} < 0$, it signifies that the host does not have sufficient resource to accommodate the VM's demands and hence cannot be suitable to migrate *i*-th VM onto that specific *j*-th PM or host. Thus, each VM compares with all the available hosts or PMs so as to identify the optimal host to accommodate the task. This process continues until all VMs (from overloaded hosts) are migrated to the suitable host(s) without influencing SLA provision. The simulation results obtained and their respective inferences are presented in the subsequent section.

4. Results and discussions

This section primarily discusses the experimental design and allied simulation outcomes. The results obtained and their inferences are also discussed in this section.

4.1. Experimental design

Considering the experimental evaluation need for the proposed load balancing model, we have applied CloudSim toolkit package, which is one of the best known virtualization simulator, especially industrially acknowledged for its generalization

and scalable simulation abilities towards numerous application specific simulation. It can provide varied significant performance metrics including energy efficiency or consumption, SLA violation metrics, etc. To perform simulation over real-time cloud event traces, we used CoMon data project, a significant cloud monitoring infrastructure for PlanetLab. Considering configuration or the cloud infrastructures, we used benchmark data encompassing CPU utilization traces by thousands of servers located at the different geographical locations. The cloud monitoring traces of encompasses the tasks information collected during 10 randomly chosen days in between March and April 2011. Noticeably, the considered PlanetLab data or cloud traces contain the trace-values or information (CPU utilization) at the interval of 5 minutes. As hardware configurations two distinct and autonomously operating server configurations have been considered. The first server is armoured with the dual-core CPUs, single HP ProLiant ML110 G5 server with Intel Xeon 3040 and 2 cores ×1860 MHz processors. The other server was configured with HP ProLiant ML110 G5 with Intel Xeon 3075, dual cores × 2660 MHz operating processors. Both servers were armoured with a 4GB RAM to support real-time computation. Noticeably, each server was having a 1 GB/s network bandwidth and to illustrate the efficacy of our proposed load balancing model the resource utilization of an application was generated stochastically where the overall RAM utilization was maintained at 50% to avoid any probable hanging or pre-mature shut-down condition.

4.2. Experimental results

To assess the efficiency of the proposed ARGA assisted load balancing model, we simulated different set of algorithms and allied configurations to perform load balancing. Hence, we have incorporated the best recommended models in conjunction to our novel proposed load-sensitive migration policy. We compared the simulation outputs of our proposed model with static THReshold (THR) and classical IQR-RS (IQR with Random Selection (RS) Policy) based approaches [2] with safe-parameter of 1.5. Additionally, IQR with MS selection policy and ARGA VM-migration scheduling has been considered and to assess relative performance, MMT selection policy has also been taken into consideration. Noticeably, in IQR+MS model, the safe parameter of 1.2 is considered. To examine effectiveness of our proposed model a few QoS centric performance variables have been derived and a snippet of these is given as follows:

To enable QoS centric cloud-infrastructure solution, performance can be visualized in the form of SLA constraints that refer enterprise service-level requirements for data center. It can be characterized in terms of the minimum latency or downtime, maximum response time, minimum migration counts, etc., (Beloglazov and Buyya [2]) too recommended SLAV metrics (SLA Violations) to assess the SLA delivered by a VM in an IaaS cloud infrastructure. It can be characterised in terms of either the SLA Violations caused because of Over-utilization (SLATAH) (21) or due to the Performance Degradation in Migrations (PDM) (23). Mathematically,

$$\text{SLATAH} = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{sj}}{T_{aj}}. \tag{21}$$

Here the variable N signifies the total hosts or PMs; T_{si} states the overall span in which the j-th PM underwent 100% resource utilization and hence imposing SLA violation. The other variable T_{aj} presents the total i-th PM being in the active state.

$$\text{SLATAH} = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{di}}{C_{ri}}. \tag{22}$$

In (22), M represents the total VMs; C_{di} refers the performance degradation pertaining to the i-th VM caused by migrations and C_{ri} states the overall CPU capacity demanded by the i-th VM during its lifetime. Eventually, the overall SLA violation can be mathematically derived as

$$\text{SLAV} = \text{SLATAH} \times \text{PDM}. \tag{23}$$

The above equation reveals that both SLATAH as well as PDM contribute similar significance towards SLA violations. In this paper we have estimated SLAV, number of migration and downtime as the performance parameters. To ensure SLA centric and QoS oriented reliable computation it is inevitable to maintain minimum downtime. In addition to the above stated performance variables, we have examined our proposed load balancing in terms of the total number of migration. Undeniably, higher number of migrations imposes high response time and even increases the probability of downtime [3]. In our proposed ARGA assisted load balancing we intend to reduce the number of migrations so as to achieve QoS as well as SLA provision. The results obtained for the above stated variables are given as follows:

We have examined the performance of our proposed load balancing model with two VM selection policies MMT and MC. Similarly, two VM migration policies have also been applied including ACA (as existing meta-heuristic model) and ARGA as our proposed placement policy. Noticeably, in the simulation scenarios, we have applied IQR and LRR as the stochastic load detection approach. In numerous existing works authors have compared their respective performance in reference to the static thresholding based load detection and RS based VM placement. However, such approaches cannot be assured to have optimality due to dynamic nature of load variations (i.e., application specific resource demands variations) and iterative overload conditions due to classical RS approach. Even the native RS method does not consider micro-level SLA objective to perform consolidation. Therefore, in this paper we emphasized simulating our proposed model with an enhanced stochastic prediction assisted load measurement (proposed dual-level dynamic threshold assisted load estimation), where it is hypothesized to outperform classical IQR or LRR individually. The simulation outcomes are obtained with the proposed dual-level dynamic threshold assisted load estimation model, which has been followed by MMT and MC based VM selection and ARGA assisted VM migration. Observing key characteristics that MMT which considers migration time as a factor to identify a VM to be migrated or offloaded from the overloaded host is not concerned about the application demands scenario or resource access/demands type. On the other hand MC type VM selection approach considers hosts as well as VMs resource needs and based on the correlation between the resource demands patterns and thus based on optimal correlation identify a VM or VMs to migrate from the overloaded host. Since, this method considers application specific resource demand patterns; scheduling migration accordingly can help retaining maximum reliable offloading. In other words, MC based VM selection can help retaining higher reliability and minimum

possible vulnerability of suffering pre-mature task's downtime. We have compared ARGA based migration scheduling with ACO algorithm. Though, ACO is also a heuristic approach that intends to solve Travelling Salesman Problem (TSP) by exploiting Ant's movement or path planning phenomenon, its efficacy can be limited for the optimization problem with a large number of search space or population with dynamic non-linearity. Figs 3-6 presents the simulation results obtained with respect to MMT selection policy.

Fig. 3 presents the SLAV performance by our proposed ARGA VM-migration policy and the existing ACO based migration. Noticeably, here we used MMT as the VM selection policy to offload VM from the overloaded host. Observing the results it can be found that the proposed ARGA based model exhibits lower SLAV than the existing ACO based load balancing model.

Fig. 4 presents the SLA performance degradation with MMT selection policy. Result (Fig. 4) affirms efficiency of the proposed ARGA model over ACO based migration. Fig. 5 presents the SLA time per active host, where we observed that the proposed model exhibits lower SLA (downtime or loss) per active host than the classical ACO based scheduling. As already indicated that to achieve QoS delivery and SLA assurance maintaining lower migration count is a must. With this motive, the simulation performed exhibited that in our proposed model ARGA based scheduling incurs lower migration counts than the ACO (Fig. 6). Figs 7-10 presents the results obtained with MC selection policy. Observing these results, it can be found that the proposed ARGA assisted VM migration scheduling or load balancing model outperforms ACA based approaches in terms of SLAV (Fig. 7), SLA Performance degradation (Fig. 8), SLA time per active host (Fig. 9) and the number of migrations (Fig. 10).

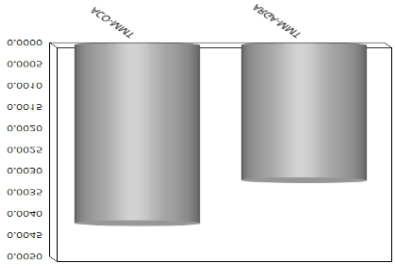

Fig. 3. SLAV analysis with MMT selection policy

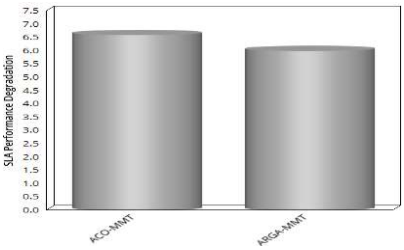

Fig. 4. SLA Performance degradation analysis with MMT selection policy

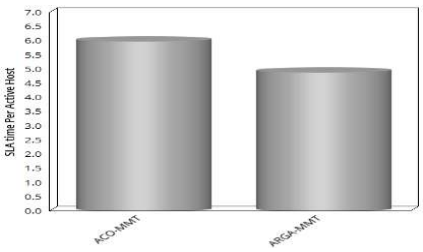

Fig. 5. SLA time per active host with MMT selection policy

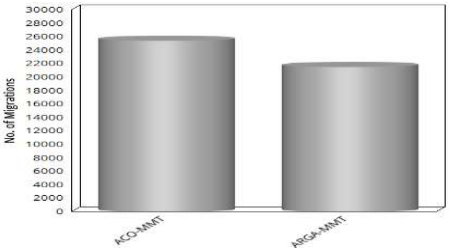

Fig. 6. No of Migration analysis with MMT selection policy

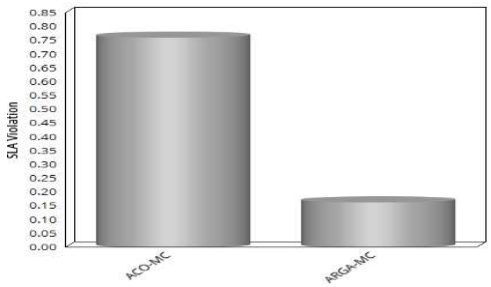

Fig. 7. SLA violation analysis with MC selection policy

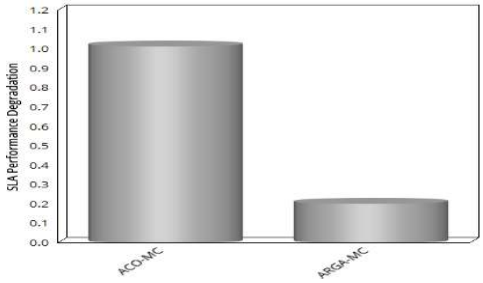

Fig. 8. SLA Performance degradation analysis with MC selection policy

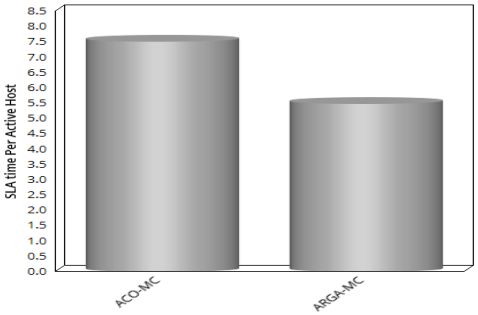

Fig. 9. SLA time per active host with MC selection policy

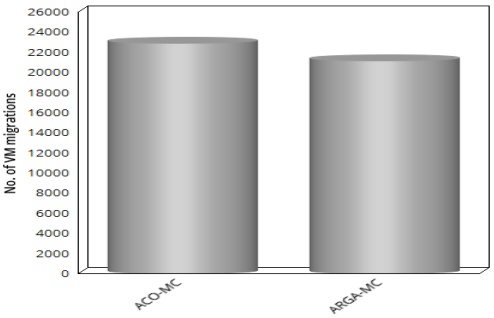

Fig. 10. No of Migration analysis with MC selection policy

Observing the performance results (Figs 3-8), it can be found that the proposed ARGA load balancing model can be more effective with dual-level dynamic threshold assisted load estimation overload detection and MC based selection policy.

The comparative performance results are depicted in Table 1. The overall results illustrate that the proposed ARGA based load balancing (particularly as VM migration policy) can be optimal with our proposed dual-level dynamic threshold assisted load estimation and MC based VM selection. This approach cannot only ensure SLA provision but can also augment QoS assurance over dynamic IaaS cloud infrastructure.

Table 1. ARGA performance with MMT and MC VM selection policy

QoS Parameters	ARGA-MMT	ARGA-MC
SLA Violation	0.00315	0.16416
SLA Performance Degradation	6	0.203
SLA time Per Active Host	4.88666	5.479
Number of VM migrations	21465	21146

Thus, with the obtained results and allied inferences the following cloud load balancing model can be derived.

Table 2. Proposed load balancing model for IaaS

Parameters	Solution
Cloud/Service type	IaaS
Data type	Dynamic cloud trace over definite period
Load Measurement	Dual-level dynamic threshold assisted load estimation (IQR+LRR with safe parameter 1.2)
VM Selection	MC
VM Migration scheduling	ARGA (Proposed algorithm)

5. Conclusion

To achieve a robust SLA and/or QoS centric load balancing it is inevitable to augment the above stated three phases. Considering this as motivation in this research the focus was made on augmenting each participating phases where at first early overloading prediction model was implemented that helped making timely migration decision. On the other hand, the consideration of MC model as VM selection strategy significantly assisted migrating only that task or VM, which could reach to the targeted host in minimum possible time. This approach significantly reduced the probability of any (high) downtime or SLA violation. In addition, the use of ARGA helped achieving optimal VM migration or scheduling where the tasks were migrated successfully without causing any significant violation in SLA provision. Noticeably, unlike classical meta-heuristic based VM migration the use of ARGA alleviated the problem of local minima and convergence that consequently strengthened the proposed load balancing method to achieve QoS and SLA centric load balancing over large-scale cloud data centers. Since the proposed load balancing method implements optimization at all comprising functions such as prediction based load sensing, SLA-sensitive VM selection and ARGA assisted migration scheduling, it can be applied for any number of VMs and hosts and therefore its robustness gets confirmed. The simulation results with different load conditions and configurations too have confirmed that the proposed load balancing model outperforms state of art ACO based approach. Considering overall approach it can be stated that the proposed load balancing model can be easily implemented with real-time purposes where it can enable timely and QoS oriented cloud computing services or processes.

References

1. W a n g, L., G. L a s z e w s k i, M. K u n z e, J. T a o. Cloud Computing: A Perspective Study. – J. New Generation Computing, 2010, pp. 1-11.
2. B e l o g l a z o v, A., R. B u y y a. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints. – IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. **24**, 2013, No 7, pp. 1366-1379.
3. M e l l, P., T. G r a n c e. The NIST Definition of Cloud Computing. 2009. Accessed April 2010. **Http://www.wheresmyserver.co.nz/storage/ media /faq-files/clouddef-v15.pdf**
4. G u, J.-H., J.-H. H u, T.-H. Z h a o, G.-F. S u n. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. 2010, pp. 89-96.

5. Sahu, Y., R. Pateriya, R. K. Gupta. Cloud Server Optimization with Load Balancing and Green Computing Techniques Using Dynamic Compare and Balance Algorithm. 2013, pp. 527-531.
6. Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warifield. Xen and the Art of Virtualization. – ACM SIGOPS Operating Systems Review, Vol. **37**, 2003, pp. 164-177.
7. Kivity, A., Y. Kamay, D. Laor, U. Lublin, A. Liguori. KVM: The Linux Virtual Machine Monitor. – In: Proc. of Linux Symposium, No 1, 2007, pp. 255-230.
8. Megharaj, G., K. G. Mohan. A Survey on Load Balancing Techniques in Cloud Computing. – IOSR Journal of Computer Engineering (IOSR-JCE), Vol. **18**. March-April 2016, Issue 2, Ver. I. e-ISSN: 2278-0661, p-ISSN: 2278-8727.
9. Kaur, J., S. Kinger. A Survey on Load Balancing Techniques in Cloud Computing. – International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064, Vol. **3**, June 2014, Issue 6.
10. Park, K. S., V. S. Pai. CoMon: A Mostly-Scalable Monitoring System for Planet-Lab. – ACM SIGOPS Operating Systems Review, Vol. **40**, 2006, No 1, pp. 65-74.
11. Yao, L. G. W., J. Ren, Y. Zhu, Y. Li. Guaranteeing Fault-Tolerant Requirement Load Balancing Scheme Based on VM Migration. – The Computer Journal, Vol. **57**, February 2014, No 2, pp. 225-232.
12. Tao, Z. L., B. Wang. Load Feedback-Based Resource Scheduling and Dynamic Migration-Based Data Locality for Virtual Hadoop Clusters in Openstack-Based Clouds. – Tsinghua Science and Technology, Vol. **22**, April 2017, No 2, pp. 149-159.
13. Yang, D., Y. Yang. A Load Balancing and Multi-Tenancy Oriented Data Center Virtualization Framework. – IEEE Transactions on Parallel and Distributed Systems, Vol. **28**, 1 August 2017, No 8, pp. 2131-2144.
14. Ningning, S., G. Chao, A. Xingshuo, Z. Qiang. Fog Computing Dynamic Load Balancing Mechanism Based on Graph Repartitioning. – China Communications, Vol. **13**, March 2016, No 3, pp. 156-164.
15. Cheng, L., F. C. M. Lau. Offloading Interrupt Load Balancing from SMP Virtual Machines to the Hypervisor. – IEEE Transactions on Parallel and Distributed Sys., Vol. **27**, 1 November 2016, No 11, pp. 3298-3310.
16. Laredo, J. L. J., F. Guinand, D. Olivier, P. Bouvry. Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing. – IEEE Transactions on Parallel and Distributed Systems, Vol. **28**, 1 February 2017, No 2, pp. 517-529.
17. Islam, M. A., S. Ren, G. Quan, M. Z. Shakir, A. V. Vasilakos. Water-Constrained Geographic Load Balancing in Data Centers. – IEEE Trans. on Cloud Computing, Vol. **5**, April-June 2017, No 2, pp. 208-220.
18. Paya, A., D. C. Marinescu. Energy-Aware Load Balancing and Application Scaling for the Cloud Ecosystem. – IEEE Transactions on Cloud Computing, Vol. **5**, January-March 2017, No 1, pp. 15-27.
19. Chen, Y. J., L. C. Wang, M. C. Chen, P. M. Huang, P. J. Chung. SDN-Enabled Traffic-Aware Load Balancing for M2M Networks. – IEEE Internet of Things Journal, Vol. **5**, June 2018, No 3, pp. 1797-1806.
20. Aslanzadeh, S., Z. Chaczko. Load Balancing Optimization in Cloud Computing: Applying Endocrine-Particale Swarm Optimization. – In: 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, 2015, pp. 165-169.
21. Pei, Y., J. Wu, X. Wang. Virtual Machine Placement Strategy Based on Particle Swarm Optimization Algorithm. – Computer Engineering, Vol. **38**, 1 2012–2 2013, No 16, p. 16.
22. Velde, V., B. Rama. An Advanced Algorithm for Load Balancing in Cloud Computing Using Fuzzy Technique. – In: 2017 International Conference on Intelligent Computing and Control Sys., Madurai, 2017, pp. 1042-1047.
23. Tiwari, K., S. Joshi. Dynamic Weighted Virtual Machine Live Migration Mechanism to Manages Load Balancing in Cloud Computing. – In: 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, 2016, pp. 1-5.

24. L a g w a l, M., N. B h a r d w a j. Load Balancing in Cloud Computing Using Genetic Algorithm. – In: 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, 2017, pp. 560-565.
25. M a k a s a r w a l a, H. A., P. H a z a r i. Using Genetic Algorithm for Load Balancing in Cloud Computing. – 2016 8th International Conference on Electronics, Computers and Artificial Intelligence, Ploiesti, 2016, pp. 1-6.

Received: 08.03.2019; Accepted: 03.07.2019