Multi-level Hierarchical Controller Assisted Task Scheduling and Resource Allocation in Large Cloud Infrastructures

Jyothi S, B S Shylaja
Department of Information Science & Engineering
Dr.Ambedkar Institute of Technology, Bengaluru, India

***Abstract*—The high-pace emergence in Cloud Computing technologies demands and alarmed academia-industries to attain Quality-of-Service (QoS) oriented solutions to ensure optimal network performance in terms of Service Level Agreement (SLA) provision as well as Energy-Efficiency. Majority of the at-hand solutions employ Virtual Machine Migration to perform dynamic resource allocation which fails in addressing the key problem of SLA-sensitive scheduling where it demands timely and reliable task-migration solution. Undeniably, VM consolidation may help achieve energy-efficiency along with dynamic resource allocation where the classical heuristic methods which are often criticized for its local minima and premature convergence doesn't guarantee the optimality of the solution, especially over large cloud infrastructures. Considering these key problems as motivation, in this paper a highly robust and improved meta-heuristic model based on Ant Colony System is developed to achieve Task Scheduling and Resource Allocation. CloudSim based simulation over different PlanetLab cloud traces exhibited superior performance by the proposed task-scheduling model in terms of negligible SLA violence, minimum downtime, minimum energy-consumption and higher number of migrations over other heuristic variants, which make it suitable towards realistic Cloud Computing purposes.**

***Keywords*—Task-scheduling; VM-migration; improved ant colony system; SLA assurance; energy-efficient consolidation**

I. Introduction

In the last few years, the high-pace rise in advanced software systems and decentralized computing environments has broadened the horizon for a state-of-art new paradigm named cloud computing. Cloud computing has emerged as a potential technology serving decentralized scalable services to the significantly large number of users for respective data and/or query driven computation and information services. Cloud computing technology can be characterized as an array of network-enabled services facilitating quality-of-service (QoS) assured scalable and personalized (computing) solutions, even at the inexpensive cost [1-3]. The potential to serve decentralized data or (computing) infrastructure, independent of the geographical boundaries makes cloud computing an inevitable need to meet contemporary or even NextGen industrial as well as personal computing demands [2]. Based on the usage of the Cloud it is understood that it has been applied as a key technology to serve civic purposes, financial sector, industries, government agencies, scientific community, diverse business houses, etc. Noticeably, to serve aforesaid stakeholders, cloud services are classified into three key types; Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Irrespective of the service types, fulfilling QoS in cloud computing has always remained a challenge. To meet aforesaid service demands industry requires providing decentralized storage infrastructure, often called data centers; however, with exponential rise in computing demands with non-linear (demand or use) patterns, the at-hand solutions often undergo disrupted performance or connectivity. This as a result impacts overall QoS performance. Typically, delivering Service Level Agreement (SLA) by Cloud Service Providers ensures to provide QoS support to its customers while maintaining reliable services with higher scalability, reliability and continuity over operating periods [4, 5]. It is a challenging task to retain SLA over highly dynamic load demands and use patterns across a gigantically large user-base, located around the globe.

A cloud infrastructure mainly encompasses physical machines, also called servers, virtual machines (VMs) and allied controllers. Noticeably, hosts of the physical machines primarily acts as the component serving computing ability and memory, while VMs function as containers possessing different independent tasks. A huge cloud infrastructure may consist of multiple hosts, where each host can have multiple VMs, carrying different parallel-computing tasks. In this case, due to dynamism in resource demands by each task a VM might undergo an exceedingly large resource demand, which could not be facilitated by the currently attached physical machine or host. In such a case, a VM carrying multiple tasks is required to be migrated to the suitable host, which could provide sufficient resources to the associated task for SLA assurance and QoS provision. However, it may take significantly large traversal time or allocation scheduling related delay, impacting downtime and hence overall performance. Being an uncertain demand scenario, the tasks or allied VMs can have to traverse across the network as per at-hand overloading and under-loading scenario. Undeniably, it can increase downtime as well as QoS violation. On the other hand, cloud being an energy-exhaustive technique requires addressing energy-minimization needs and therefore simultaneous dynamic resource allocation, task scheduling and energy minimization turn out to be a complex NP-hard problem [1-5]. In sync with cloud with the heterogeneous demand types, the load pertaining to each VM might vary as per task-types and demand-density over the operating period.

Therefore, merely random host selection concepts or even the classical bin-packing models, cannot be appropriate. Such classical methods might give rise to the overloading or underloading condition, and hence can impact both SLA as well as energy-efficiency.

With this context, the research work proposes a "Multi-Level Hierarchical Controller Assisted Dynamic Task Scheduling and Resource Allocation Model for Large Cloud Infrastructures" which involves a hybrid evolutionary concept named Improved Ant Colony System (I-ACS) to achieve SLA with energy efficiency to meet cloud demands. The proposed model is developed using CloudSim platform, where simulation over PlanetLab cloud trace data revealed superiority of the proposed model over major existing approaches in terms of downtime, SLA violation, number of migrations and energy-consumption.

The further sections of the presented document are given as follows. Section II discusses the Literature Survey pertaining to SLA oriented and Energy-Efficient task-scheduling methods, Section III discusses the proposed method followed by Section IV which provides Results and Discussion. The overall research Conclusion and allied inferences are presented in Section V. References followed in this research are provided at the end of the manuscript.

II. Related Work

Afzal et al. [6] focuses on Load balancing based heuristic assisted task scheduling concept under static or dynamic load conditions. However, unlike classical static resource allocation that employs a first-come-first-servemethod, it can't be suitable under dynamic load conditions. Pradhan et al. [7] discusses about modifying especially round robin methods, which authors applied in their research to reduce the waiting time. Mogeset al. [8]focused on energy efficiency as the key concept to perform task scheduling. To reduce energy-exhaustion, authors proposed VM consolidation concept, which was performed to shut-down underutilized hosts and by removing hotspots. However, the classical use of bin-packing based consolidation could not address latency and QoS degradation issues. In addition to the power enhancement, the work suggested to perform consolidation scheduling in such a manner that it could retain lower task response time to meet SLA demands. To achieve it, authors suggested to focus on modified bin-packing based consolidation.

Syed Arshad Ali et al. [9] implemented task scheduling using Resource aware min- min algorithm where task-scheduling was performed on the basis of the load of the servers to minimize makespan. Mosa et al. [10] on the other hand emphasized on load balancing in the cloud by distributing the workload dynamically across the cloud infrastructure with multiple nodes. Authors applied utility functions and GA heuristic model to optimize VM allocation, Energy consumption and SLA violations. Jyothi S et.al. [18] Bhaskar R et.al [19] discussed numerous key challenges in dynamic load management in heterogeneous cloud environments. Authors proposed a heterogeneity- aware dynamic application provisioning model to reduce energy consumption in cloud environments.

Doppaet al. [11] designed a self-aware framework to adjust or optimize resource and SLA. However, the use of DVFS based methods can't be suitable for a heterogeneous cloud network with dynamic load conditions. In addition to the SLA expectations, authors [12 -13] focused on resource allocation while maintaining lower computation and energy-exhaustion. Liet al. [12] designed a directed acyclic graph (DAG) model to perform priority bound task scheduling. Here, in DAG construction the nodes characterize the tasks, while the edges represent the allied messages among jobs [14-16]. Tang et al.[14] applied DAG-based workflow where tasks were prioritized based on respective sizes to perform resource allocation. Zhu et al. [17] Jyothi et al. [18] performed task scheduling on the different multiprocessing environment, which can be solved using NP-hard optimization. Considering this as motivation, dynamic task-scheduling and resource allocation is performed by applying the concepts of co-evolution system and multi-population strategy for Meta-heuristic method such as ACO is considered.

III. System Model

This discussion primarily discusses the proposed model and its implementation including the multi-controller assisted overload and underload detection, VM selection and the proposed Improved Ant Colony System (I-ACS) based task scheduling.

The task scheduling or allied VM migration can be inducted as per the task-(heterogeneous) demands' and hence a controller can migrate one or multiple VMs to the suitable hosts (via consolidation) while retaining SLA performance and energy-efficiency. The proposed model introduces multi-layered controller units to dynamically monitor the VMs and allied task's demand to stochastically predict the demands and accordingly the global controller performs scheduling in advance to avoid any SLA violation, QoS-compromise or even energy-exhaustion.

The overall proposed model encompasses four key steps. They are:

Step-1 Hierarchical Multi-layered controller assisted cloud monitoring,

Step-2 Underload and Overload detection,

Step-3 Minimum Migration Time (MMT) oriented VM selection,

Step-4 Improved ACS (I-ACS) assisted S-DTS

The details of the overall proposed model are given in the subsequent sections.

Hierarchical Multi-layer Controller assisted Cloud Monitoring.

An illustration of the different controller and its respective task is given in Fig. 1.

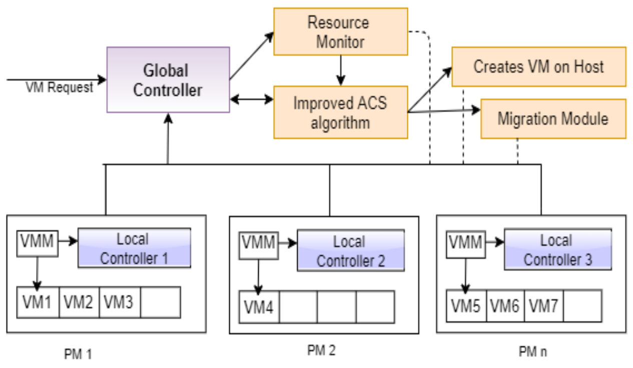

Fig. 1. Proposed Multi-controller Assisted Cloud Monitoring and Task-Scheduler.

Typically, cloud infrastructures that often accommodate a significantly large number of independent tasks operating or executed onto assigned VMs, undergo exceedingly high demand-dynamism. In other words, the different tasks connected to each VM undergo non-linear traffic demands, and therefore might require dynamic resource to continue its operation. Under such scenario, a VM encompassing single or multiple tasks might exhibit non-linear resource demand, influencing a host or physical machine to undergo under-utilization or overloading. Consequently, it might significantly impact the overall performance and SLA-reliability of the system. Considering this fact, performing demand-sensitive resource allocation or task-scheduling is must. To achieve it in the proposed method, a state-of-art new Hierarchical Multi-Layered Controller (HMLC) design is applied, which especially monitors demands or resource utilization pattern at each task connected to a VM. The proposed HMLC model encompassed a local controller and a global controller, especially designed to perform task-scheduling or dynamic resource allocation so as to preserve SLA, QoS as well as energy-efficiency. To perform task-level resource utilization assessment, local controller (LC) is applied that measures resource utilization per VM and updates the same to the global controller (GC), dynamically so as to make stochastic prediction-based task-scheduling decision in advance.

As shown in Fig. 1, the proposed local controller unit operates over each VM, accommodating multiple tasks. Here, it acts as an autonomous VMM manager that measures resource utilization dynamically and updates to the global controller so as to make dynamic task reallocation. Additionally, the proposed controller mechanism enables dynamic underload/overload detection and (proactive) avoidance. Once detecting any hotspot or any PM undergoing overload, the local controller executes VM selection mechanism (discussed in subsequent section) and selects the VM to be unloaded from the at-hand overloaded hosts. However, recalling the SLA assurance to the offloaded VM and allied tasks, the proposed model introduces a state-of-art new and robust dynamic VM scheduling model which guarantees optimal task-scheduling and allied VM migration, without affecting SLA performance. To achieve it, the proposed global controller model retrieves VM's and hosts' information proactively from the local controller and executing the proposed I-ACS concept it schedules VM placement or migration in advance so as to retain SLA intact. Once traversing or offloading the suitable VM from a host, the local controller updates the node-parameters and updates the same to the global controller for further decision making. To achieve SLA-assurance and energy-efficiency, at first, a dynamic threshold-based underload and overload detection unit is applied. The details are given as follows.

A. *Underload and Overload Detection*

To cope up with the dynamic resource demands and allied scheduling tasks, the work is carried out which examines the load condition of each task and associated host that helps in identifying under-loaded and overloaded nodes in the network. To ensure SLA-sensitive and energy-efficient scheduling, once detecting a node as under-loaded either certain specific VM (including all connected tasks) or all VMs are off-loaded, which are then migrated to the other suitable hosts. This approach not only helps in optimal resource allocation, but also preserves significant energy. On the other hand, detecting a host undergoing overload, the proposed model offloads tasks or allied VM(s) and migrates them to the other suitable host, while ensuring that the migration doesn't cause overload on another host (say, target host) or impacts SLA performance.

1) Underload detection: The proposed model discusses a host with load lower than a predefined minimum workload condition or resource utilization is referred as an underload host. In order to preserve energy, once identifying a host with under-utilized resources, it's connected VMs or allied tasks are migrated to the other host(s) strategically. However, this scheduling or migration takes place in such a manner that it doesn't cause overload on other nodes or hosts. In sync with the concept of VM consolidation, once migrating all VMs to the other host, successfully, it shuts down the host to preserve the energy. Here the task-migration or allied resource allocation strategy schedules the migration in such a manner that neither it causes SLA violation nor energy exhaustion or any possible overload situation on the target host. To guarantee SLA provision, the source host remains active or ON, until all allied tasks and the target host(s) holds the migrated connected VMs.

2) Adaptive threshold-sensitive host overload detection: To detect the overloaded VM (containing independent tasks), a stochastic prediction assisted approach is applied. In this each host node performs periodic load assessment of each host which eventually assists detecting an overloaded node. Here, each host's resource (i.e., CPU or MIPS) utilization is measured to assess the host node whether it is overloaded or not. Most of the existing approaches towards task-scheduling apply a static threshold method to detect an overloaded host. Unfortunately, IaaS which often undergoes dynamic loads over the operating period and the different tasks consume different resources at the varied time-instant. Therefore, the use of the static threshold method can't be suitable for overload detection. Here, dynamic CPU utilization (cumulative CPU utilization per VM over multiple independently processing tasks) assessment method to perform overload detection is applied. More specifically, in this method, the CPU utilization threshold value is adjusted dynamically on the basis of the changes in continuous CPU utilization. It assumes that higher fluctuation in use-pattern can be stated as the lower upper CPU utilization (threshold).

In general, the higher value of such non-linear resource utilization indicates an overloaded condition, with 100% resource utilization. To cope up with the exceedingly high dynamism in the cloud network, a hybrid concept encompassing both inter-service (task) relation along with varying information to achieve dynamic thresholding is applied. Here, a state of art new hybrid concept to exploit task level resource utilization and their cumulative impact as eventual load to perform overload detection is designed.

More specifically, interquartile range (IQR) and modified local regression methods is applied to measure dynamic CPU utilization and eventually predict adaptive threshold. Here, IQR algorithm follows a statistical dispersion approach to represent association between the first and the third quartile, as depicted in equation (1). The value of IQR is estimated to employ the equation (2) to obtain the upper-threshold of the CPU utilization.

$$IQR = Q_3 - Q_1 \quad (1)$$

$$T_u = 1 - s.IQR \quad (2)$$

With the consideration of dynamic load conditions and fluctuations in resource utilization for the same ongoing task, there can be significant effect on the upper threshold estimation (2). Any possible inaccuracy in threshold estimation might cause wrong resource allocation and allied task migration activities that as a result can affect overall SLA performance. Realizing this fact, in this research paper a state-of-art new dual-level threshold estimation model is formulated, where at first it applies IQR based $T_u$estimation, while in the subsequent phase it applies local linear regression (LRR) method. Noticeably, in the proposed model, LRR exhibits fitting of the (utilization) trend polynomial to the preceding k CPU utilization values, obtained as per (3) for each observation value.

$$\hat{g}(x) = a + \hat{b}x \quad (3)$$

Now, measuring the observation values, the next observation value, $\hat{g}(x_{k+1})$ is estimated. Now, to perform offloading of a host, the following condition is applied.

$$s.\hat{g}(x_{k+1}) \geq 1 \quad (4)$$

$$x_{k+1} - x_k \leq t_m$$

In above conditions (4), $s \in R^+$ signifies the maximum level of tolerance by a host. Here, the maximum time required to migrate a VM (containing one of multiple independently executing tasks) from host d be t_m . The classical local regression concepts which are often found limited under higher dynamic value changes and allied regression estimation. Additionally, it performs inferior due to the outliers introduced by leptokurtic or heavy-tailed distributions. Considering this fact, modified the classical least square (LR) algorithm is applied by a bi-square model. Noticeably, LR improves iteratively so as to estimate the initial fitting for which the tricube weights are obtained using a Tricube Weight Function (TWF). Here, the obtained fitting parameter at x_i was applied to retain the fitted values using$\hat{y}_i$. In this manner, the residual value, signifying $\varepsilon_i = y_i - \hat{y}_i$ was estimated. Thus, with the estimated values of $x_i$andy_i, it was assigned in (5) to estimate a factor called robustness factor R_i.

$$R_i = B\left(\frac{\hat{\varepsilon}_l}{6s}\right) \quad (5)$$

Every observation value was allocated R_i . In (6), $B(.)$represents the bisquare weight function and s represents the Medium Absolute Deviation (MAD) to achieve least square fitting. Thus, obtaining $B(.)$As per (6).

$$B(.) = \{(1-u^2)^2 \quad if\ |u| < 1, 0 \quad Otherwise \quad (6)$$

In above derived equation (5), s was obtained as per (7).

$$s = mediun|\hat{\varepsilon}_i| \quad (7)$$

Thus, employing the above derived model (4) for the estimated trend line, the predicted possible value or instance, for any inequalities (with reference to the predicted value and the observed value), a host was identified as an overloaded host. Eventually, identifying the overloaded host, the local controller unit informs the global controller and meanwhile identifies the VMs to be migrated to the other resource-sufficient (optimal) host node. Though, in literature, researchers have randomly considered any VM to execute migrate; however, for SLA-sensitive task migration purposes, such approaches might undergo SLA-violation phase or QoS compromise, especially due to increased downtime and even complete task or transaction failure. Considering this fact, distinct unit called VM selection model is necessary. A snippet of the VM selection method applied is given as follows.

B. SLA Oriented Minimum Migration Time-based VM Selection

In order to preserve SLA, while guaranteeing minimum downtime, the minimum migration time (MMT) based VM selection method is applied. In other words, once identifying an overloaded host, only that specific VM is migrated, which takes minimum migration time. Hypothesizing the fact that higher downtime can lead higher losses, so maintaining lower downtime as favorable, MMT as a VM selection policy is considered. This approach can be suitable towards SLA preserving effort as well as reliable cloud service provision. Migration time for each task and allied VM connected to the overloaded host of PM is estimated. Thus, sorting the VMs based on their respective migration time, the VM is chosen with the minimum migration delay, at first to migrate towards the target host. Thus, applying this method, broadened the horizon for delay-resilient migration over cloud platforms. Now, once selecting the VM to be migrated, the local controller passes all allied details to the global controller, which employs a highly robust improved ACS heuristic concept to perform VM placement of migration. Though, the proposed VM migration or allied task scheduling concept resembles a VM consolidation problem; however, considering real-time tasks characteristics, classical meta-heuristics is improved to not only alleviate local minima and convergence but also ensure timely and SLA-centric task-migration or allied resource allocation.

The following section discusses the proposed I-ACS model for task-scheduling over a large Infrastructure as a Service (IaaS) cloud platform.

C. Task-Migration Problem Definition

Consider that the set of operating physical machines or the hosts be $PM = \{pm_1, pm_2, pm_3, \cdots, pm_m\}$ where, pm_i represents the specific host conditioned as$1 \leq i \leq m$. In the same manner, let the set of VMs encompassing or containing multiple autonomously operating tasks be $VM_i = \{vm_1, vm_2, vm_3, \cdots, vm_{n,i}\}$, where each VM is connected to certain host. $vm_{j,i}$be the $j-$th VM connected on the $i-$th host. The variable $x_{i,j}$ presents a binary variable signifying on host j connected by the i-th VM. Consider that $P_{r,i}$ be the resource capacity (in terms of the CPU utilization) of r on the $j-$th host and the resource demanded by the $j-$th VM be v_{rj}. In this manner the total load at that host can be characterized in the form of the total load caused by all VMs and allied tasks running onto it. Let, T be the time-period or observation period. Thus, the sub-gap can be estimated by splitting T into $q-1$ intervals $T = [(t_2 - t_1)(t_3 - t_2)\cdots(t_q - t_{q-1})]$. Noticeably, the time-slot $t_k - t_{k-1}$ represents the interval k. Thus, over k, the CPU utilization is estimated at a host using (8).

$$CPU_{i,Util}(k) = \sum_{j=1}^{n} vm_{CPU,j} \div pm_{CPU,j} \tag{8}$$

In (8), the parameter k refers to the CPU utilization which was collected for certain period. The average CPU utilization is estimated using (9).

$$pm_{i,AvgUtil} = \sum_{t-t_k}^{t_k-n} pm_{i,Util}(t) \div (q-1) \tag{9}$$

In (9), $(q-1)$ states the total amount of sub intervals or gap over T observation period. Let, $pm_{i,w}(k)$ be the power of $i-$th host over t_k span, then the power status can be obtained based on the CPU utilization value. $pm_i E^{(k)}$which signifies the energy consumption by the $i-$th host from the last time interval to the current time interval and hence is estimated as per (10).

$$pm_i E^{(k)} = pm_{iw}(k-1) + (pm_{i,w}(k-1) + (pm_{i,w}(k))(t_k - t_k - 1) \tag{10}$$

Based on host consumption hypothesis, for any host pm_j, employing CPU utilization, $CPU_{i,Util}(k)$ the energy consumed can be obtained as per (11).

$$E(pm_j) = K_j \, . \, e_j^{max} + (1 - k_j) \, .e_j^{max} \, .CPU_{i,Util}(k) \tag{11}$$

In (11), k_j signifies the portion of energy exhausted when the host (i.e., pm_j) is in idle state; while e_j^{max} refers the energy exhaustion of pm_j when being utilized completely. Moreover, the parameter $CPU_{i,Util}(k)$ presents the CPU utilization by pm_j over k duration. Thus, applying this mechanism, the resource utilization is estimated dynamically over each host and correspondingly the resource demand by each task or allied VM is estimated. Now, the resource consumption for all active hosts, $D_E(k)$ since the last or passed time interval to the current instant is estimated as per (12). The key dominant goal behind task migration or VM allocation problem is to obtain the set of VM-host mapping, where the proposed allocation model is supposed to place the targeted VM onto the suitable host, without impacting SLA performance or energy-exhaustion. Here, the resource allocation is performed in such manner that the proposed scheduling model attains minimal resource exhaustion$D_E(k)$, conditioned at:

$$\forall_i \sum_{j=1}^{m} x_{ij-1} \tag{12}$$

$$\forall_j \sum_{i=1}^{n} vm_{CPU,i} X_{ij} \leq pm_{CPU,j} \tag{13}$$

Thus, with the above derived motive, in this research work, a state-of-art new improved ACS heuristic model is developed for SLA-centric task-scheduling and allied VM or resource allocation strategy. The details of the planned I-ACS model is given in the following section.

D. Improved-ACS based Task Scheduling

Unlike classical heuristic models, a hybrid ACS algorithm for task scheduling or allied VM allocation is applied. The VM scheduling model proposed in this paper is based on a well-known heuristic model named ACO in which multiple agents estimate the solution-likelihood in iterative cycles. During this process, they converse ultimately by dropping the pheromone, which is a chemical substance called on respective paths they traverse. But, for research-intended task-scheduling or VM placement doesn't employ the notion of path, in the proposed model pheromone is deposited by the ants on each task (or VM) and within a pheromone matrix by the host pair. The ants retrieve VMs in each series, and starts forming local solutions by means of a probabilistic decision rule that signifies the attractiveness for an ant to select a specific VM (MMT based VM selection) as the next one to pack in its current host. In this mechanism, the higher the amount of pheromone deposition and higher information related to a VM-host pair, the probability that it will be selected for migration will also be higher. Fig. 2 presents the solution formation for a single ant. In this mechanism, the ant initiates with four VMs, calculates the probabilities for each of the VMs using the probabilistic decision rule, and begins allocating the (selected) VMs for each selected host as per the estimated probabilities. Once the host is completely occupied with the migrated task or allied VMs the proposed model identifies a new host on the basis of corresponding likelihood.

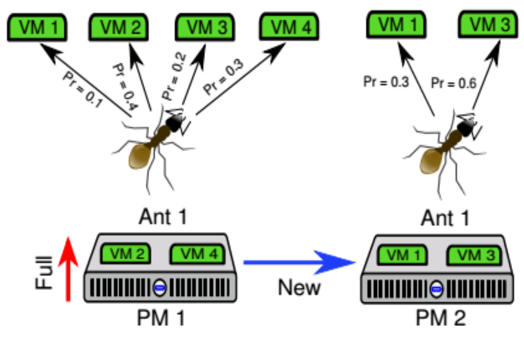

Fig. 2. ACS-assisted Task-scheduling or VM Placement.

This process continues till all tasks or VMs are assigned to the suitable hosts. During the optimal solution estimation, in each cycle or iteration, local solutions are assessed and the one demanding the minimum number of hosts is selected as the different optimal solution globally. Subsequently, it updates the pheromone matrix to estimate pheromone loss and reinforce VM-host pairs belonging to the set of the optimal or the best solution. To achieve it, ACS implements the pheromone update rule. In the proposed ACS based task-migration model, the proposed I-ACS gets triggered during VM assignment to the target host. It outputs a solution comprising VM to host (map) while maintaining no (or negligible) SLA violation or maximum host-shutdown (to achieve energy-efficiency).

In this research the emphasis has been made on optimizing classical ACO to avoid local minima, convergence and enhance solution diversity to meet dynamic cloud resource optimization. The proposed model encompasses some of the key optimizations such as multi-population strategy, co-evolution concept, dynamic pheromone update and dynamic pheromone diffusion. Such optimization measures enable the proposed model to retain optimal balance in between the convergence rate as well as solution diversity which helps perform better VM scheduling or placement decisions. Moreover, it helps perform swift computation which is effective towards large scale mega-cloud infrastructures. To achieve it, the proposed model is designed in such manner that it splits overall optimization problem into multiple sub-problems where to avoid local minima and convergence (i.e., to achieve local optima), the ant-population is split into two specific categories; Elite Population and Normal Ant-Population. This process not only increases computational efficiency (i.e., higher convergence rate) but also retains swift global-optima identification. Additionally, incorporating a dynamic pheromone update mechanism to enhance optimization ability over large network sizes. Similarly, the intent of pheromone diffusion was to make the pheromone released by ants at a certain point, which gradually affects a certain range of adjacent regions.

Realizing large scale cloud infrastructure, the concept of co-evolution (in reference to both populations based as well as diffusion based) is applied that helps interchanging local information amongst varied sub-population to achieve dynamic information sharing. Noticeably, each VM is hypothesized to be a component possessing or encompassing operating tasks, and therefore the concept of co-evolution can enable dynamic decision without imposing an SLA violation issue. Thus, the implementation of the overall proposed model can ensure optimal energy as well as QoS oriented task scheduling across a large-scale cloud-infrastructure.

Before discussing the overall proposed improved ACS model for the intended task-scheduling or VM migration, a ACO model is given as follows.

E. Probabilistic Decision Rule

In VM allocation strategy, at first ACS defines the likelihood of an ant to select a VM v for migrating it to a specific host p using (14).

$$Pr_p^v := \frac{\left[\tau_{v,p}\right]^{\alpha} \times \left[\eta_{v,p}\right]^{\beta}}{\sum_{u \in N_p} \left[\tau_{v,p}\right]^{\alpha} \times \left[\eta_{v,p}\right]^{\beta}}, \forall_v \in N_p \tag{14}$$

In (14), the parameter $\tau_{v,p}$ states the pheromone-based attractiveness to migrate or attack VM v onto the host p. Similarly, the parameter $\eta_{v,p}$ represents the VMs heuristic information. Moreover, the other variables $\alpha, \beta \geq 0$ are applied to either focus more on the pheromone or the vital heuristic information. Moreover, N_p states the total number of VMs encompassing single or multiple tasks which are suitable to be attacked or connected with the current host p. l_p states the overall utilized memory or capacity of the current host, which can be estimated as the sum of all requested resources by the connected VMs, i.e., $l_p := \sum_{\forall_v \in V} RC_v$. Here, the task scheduling or allied VM placement is accomplished by means of a parameter $\eta_{v,p}$, which is estimated as per (15).

$$\eta_{v,p} := \frac{1}{\left| TC_p - \left(T_C + RC_v\right) \right|_1} \tag{15}$$

Thus, once estimating the value of (15), a d-dimensional demand vector is created which is subsequently mapped in terms of a scalar value. Here, the L1-norm method is applied to perform mapping the VM and host so as to perform migration decisions.

F. Pheromone Trail Update

Once performing initial solution construction, the pheromone trails on all the pairs of VM-hosts are updated, which helps global solution retrieval. Classically ACS systems apply MAX-MIN Ant System (MMAS) and therefore the ant with the best solution in iteration is permitted to deposit pheromone. Thus, the pheromone update is performed as per (16).

$$\tau_{v,p} : (1-p) \times \tau_{v,p} + \Delta\tau_{v,p}^{best}, \forall (v,p) \in V \times P \tag{16}$$

In (16), $\rho, (0 \leq \rho \leq 1)$ plays a vital role in simulating the pheromone evaporation. Noticeably, the higher value of ρ results in an increased rate of evaporation. Additionally, a few pairs of the target VM and host pair require reinforcement and therefore $\Delta\tau_{v,p}^{best}$ is defined as the best pheromone amount deposited in each iteration by that VM-host pair. In other words, $\Delta\tau_{v,p}^{best}$ states the amount of pheromone added or deposited to the edge (v, p). In this manner, the VM-host pairs with S_{best} is reinforced that gains higher attraction.

$$\Delta\tau_{v,p}^{best} := \{\frac{1}{f\left(s_{v,p}\right)} \quad if \;\; x_{v,p} = 10 \qquad otherwise \tag{17}$$

In ACS based task-scheduling or resource allocation, VMs and host nodes are considered as input along with respective demanded resource capacity and total capacity RC_p andTC_p, respectively. Furthermore, certain parameters like α, β, ρ, g, τ_{max}, $nCycles, nAnts$ are initialized and the initial pheromone trails for VM (tasks)-host pairs is defined as τ_{max}. nCycles represents the number of iterations. In individual iteration an ant a initiates a host set PM_p and performs solution retrieval processS_a. Thus, with these initialized parameters, ACS model performs task-migration or VM allocation to the different suitable hosts, while maintaining optimal SLA and higher energy-efficiency. A snippet of the classical ACS based task scheduling is given as follows.

Algorithm 1 ACS-based VM scheduling or the task scheduling

```
**Input:** Declare VMs and hosts with respective resource requirement, $RC_v$and $TC_p$
**Output:** Best solution $S_{best}$
Assign initial pheromone value for the pair of VM-host with $\tau_{max}$
**for all** $q \in \{0 \ldots nCycles - 1\}$**do**
 **for all** $a \in \{0 \ldots nAnts - 1\}$**do**
 $IS := V; p := 0$
 $S_a := \left[x_{v,p} := 0\right], \forall v \in \{0, ..., m-1\}, \forall p \in \{0, ..., n-1\}$ **while** $IS \neq \emptyset$**do**
 $N_p := \left\{ v \sum_{p=0}^{n-1} x_{v,p} = 0 \Lambda l_p + RC_v \leq TC_p \right\}$
 **if** $N_p \neq \emptyset$ **then**
         Select $VM_v \in 2\ N_v$ as per the probability
         $Pr_p^v := \frac{\left[\tau_{v,p}\right]^\alpha \times \left[\eta_{v,p}\right]^\beta}{\sum_{u \in N_p}\left[\tau_{v,p}\right]^\alpha \times \left[\eta_{v,p}\right]^\beta}$
         $x_{v,p} := 1$
         $IS := IS\{v\}$
         $lp := lp + RCv$
         **else**
         $p := p+1$
 **end if**
 **end while**
**end for**
Estimate the solutions $S_a$as per the objective function and declare as $S_{cycle}$
**if** $q = 0 \vee IsBest(S_{cycle})$ **then**
 Declare iteration (cycle) as best solution with $S_{best}$ as the best solution.
**end if**
Estimate $\tau_{min}, andT_{max}$.
**for** all pair of VMs and hosts$(v, p) \in V \times P$ **do**
         $T_{v,p} := (1-p) \times \tau_{v,p} + \Delta\tau_{v,p}^{best}$
         **if** $\tau_{v,p} > \tau_{max}$**then**
         $T_{v,p} := T_{min}$
         **end if**
         **if** $T_{v,p} < T_{min}$**then**
         $T_{v,p} := T_{min}$
         **end if**
         **end for**
         **end for**
**return** $S_{best}$
```

As depicted in the above snippet, once identifying the best host solution, the proposed global controller schedules the VM (containing task(s)) to the selected host, and this process continues till all tasks or allied VMs are assigned a suitable host to continue respective functions. In classical ACS based optimization methods, ACS algorithm exploits the positive feedback and parallel computing concept to perform optimization. However, the majority of the ACS solutions undergo local minima and convergence problems, especially due to the complexity in estimating the optimal control parameter, etc. Though, a few efforts such as co-evolution, derived on the basis of the co-evolutionary phenomenon in nature have emerged as potential alternatives to the classical optimization solutions. These approaches employ the concept of decomposition and coordination to split a complex problem into multiple small but interacting optimization sub-problems. Such sub-problems are enhanced distinctly and perform as an eventual standalone solution. Thus, the strategic implementation of multi-population strategy along with co-evolution can improve overall performance. In sync with the ACS solution, the implementation of multiple generation, co-evolution, improved pheromone update concept and pheromone diffusion can achieve relatively better performance. Additionally, such approaches can greatly help avoiding local minima and convergence issues in the ACS system.

The intended improvement in convergence rate can significantly help in avoiding local optimal value and hence more precise resource allocation can be accomplished. This approach can be well suited towards the large-scale task-scheduling and allied resource allocation problem in cloud (IaaS) infrastructure. In reference to the above stated ACS optimization requirement, the proposed model applies a multi-generation concept that splits the complete population or ants into two broad categories; elite ants and the common ants. Moreover, it introduces state-of-art new and robust pheromone update mechanism to enhance the optimization capacity of ACS to meet at-hand task scheduling and allied VM migration control. Subsequently, a novel pheromone diffusion model is applied that effectively controls the pheromone release by ants at specific points, which subsequently impacts adjacent regions to optimize solution faster.

On the other hand, the proposed co-evolution concept helps exchanging information amongst the varied sub-populations for better information sharing. These enhancement efforts intend to achieve more efficient, fast and accurate task-scheduling over cloud to meet real-world cloud demands. The detailed discussion of the above stated improvement and allied implementation towards S-DTS purpose is given in the subsequent sections.

G. Multi-Population Generation Mechanism

In the classical ACS model, as discussed in the previous section, it applies merely one kind of population (i.e., ants) to retrieve new solutions. In this process, these classical methods apply predefined fixed values of the ant-colony size, convergence parameter, and selection parameter to control the solution-estimation. However, under dynamic applications such as at-hand cloud computing problems, it is highly complex and challenging to estimate the suitable set of parameters to retrieve the enhanced performance with swift convergence rate. Such limitations often results in premature convergence, and hence seems inferior towards task-scheduling in cloud infrastructure. To alleviate such problems, a concept of multi-population is applied that splits the entire population of ants into two categories: elite ants and common ants. The elite ants retrieve information from the solution archive that eventually helps in generating solutions by implementing a Gaussian kernel function assisted likelihood selection model. More specifically, the proposed elite ants possess a set of distinct parameters that help them (i.e., elite ants) to enhance the convergence rate. On the contrary, the common-ants are employed to generate new solutions with relatively slower speed by means of a single Gaussian function. Noticeably, to achieve it, the common ants employ the mean value of each dimension that helps in avoiding local optima. The proposed model applies the following Gaussian function to generate common ants.

$$f_N^i(x) = \frac{1}{\sigma_{i,N}\sqrt{2\pi}} e^{-\frac{(x-\mu_{i,N},)^2}{2\sigma_{i,N}^2}} \tag{18}$$

$$\mu_{i,N} = \sum_{k=1}^{K} S_{i,k} \tag{19}$$

$$\sigma_{i,N} = \xi_N \sum_{e=1}^{K} \frac{\left|S_{i,e} - \underline{s_i}\right|}{K-1} \tag{20}$$

In (18), the parameter $f_N^i(x)$ represents the Gaussian function used for common ant generation in the $i-th$ dimension. The other parameter, $\mu_{i,N}$ represents the sample value while $\sigma_{i,N}$ refers to the obtained standard deviation. The average value of the solution in the i-th dimension is given by S_i . Here, ξ_N be the constant employed to control the convergence rate of the common ants. Thus, the proposed model enables common antsto increase the search space sufficiently large which eventually helps improve the global search ability.

H. Multi-level Pheromone Update

In the majority of the classical ACS solutions, the key challenge is the pheromone update. To alleviate such limitations in the proposed ACS solution, the two different pheromone update mechanisms; the local pheromone update and the global pheromone update is applied. A snippet of the proposed multi-level pheromone update method is given as follows:

I. Local Pheromone Update

In the proposed model, before executing the optimization (say, the first iteration of the optimization), the pheromone deposition on each edge (signifying the VM-host pair) remains the same and constant. The local pheromone model is executed on each (passed) VM-host pair's edge once any ant completes the current iteration. Similar to the classical ACS model, it updates the local pheromone using (21).

$$\tau_{x,y}^{(i)} = (1-\rho_G)\tau_{x,y}^{(i)} + \rho_L \Delta\tau_0^{(i)} \tag{21}$$

In (21), $\rho_L \in (0,1)$ refers the local pheromone evaporation coefficient, while $1-\rho_L$ be the pheromone residue factor. The other parameter, $\tau_0^{(i)}$ presents the initial pheromone value. For a node value as 1, $\tau_0^{(i)}$ used to be the small negative number, while the same as 0, indicates $\tau_0^{(i)} = 0$.

J. Global Pheromone Update

Once all ants complete one iteration and achieve a solution set, the passed nodes exhibit the global pheromone update. Unlike classical pheromone update model (17), the proposed model performs pheromone update as per (22).

$$\tau_{x,y}^{(i)} = (1-\rho_G)\tau_{x,y}^{(i)} + \rho_L \Delta\tau_0^{(i)} \tag{22}$$

$$\Delta\tau_G^{(i)} = \{F_G^{(i)}, (x, j) \\ \in Global\ optimal\ Solution\ F_I^{(i)}, \\ \in Iterative\ optimal\ Solution\ 0, Otherwise \tag{23}$$

In (22-23), $\rho_G \in (0,1)$ refers the global pheromone evaporation coefficient, while $(1-\rho_G)$ presents the residue factor, $F_G^{(i)}$ is global optimal solution and while $F_l^{(i)}$ signifies the iterative optimal solution.

K. Pheromone Diffusion

In Pheromone Diffusion process, the ants (agent) apply a single pheromone release mechanism. This approach can merely influence the subsequent ants with the passed same point; however doesn't guide the ant-search within a specific range of neighboring regions, and therefore influences the overall optimization performance. Based on the above discussed multi-layer pheromone update model, the pheromone diffusion concept to enhances the performance. The likelihood of superior solutions in the neighboring region used to be higher in comparison to the other neighboring regions. Hence, the pheromone diffusion concept can enable pheromone release by the agents at a certain point that slowly influences a specific range of the adjoining regions. On the other hand, the other ants (elite ants) intend to avoid making any search in its vicinity of the poor solution and often intend to search the solution near or in the neighborhood of the better solution. This as a result not only improves time performance but also accuracy of the selected solution in each iteration. Mathematically, the pheromone update and diffusion concept are presented as per (24-25).

$$\tau_{x,y}^{(i)} = (1-\rho_D)\tau_{x,y}^{(i)} + \rho_L \Delta\tau_{x,y}^{(i)} \tag{24}$$

$$\Delta\tau_{x,y}^{(i)} = \{\frac{1}{N+1} \times \frac{\tau_x^{(i)}}{d_r(o_x, o_y)}, d_r(o_x, o_y) < 10, Otherwise \tag{25}$$

In (25), N refers the total number of estimated solutions in current iteration, while $\tau_x^{(i)}$ refers the left guiding pheromone concentration on the source object o_x. The other parameter, $d_r(o_x, o_y) = 1/(f+1)$ represents the correlation distance in between the two maps or objects.

L. The Co-Evolution

Unlike classical evolutionary computing approaches, co-evolution is an improved concept that enables higher biological diversity, by emphasizing on certain reliance on intra-organisms (between organisms and organisms), inter-organisms (organisms and environment) during the evolution process. Functionally, it employs evolution theory to construct the competition relation or cooperation relation among two or more populations so as to enhance optimization performance by the interaction of multiple populations. It also focusses on exploiting at-hand interaction amongst the varied sub-populations, and eventually influences each other to co-evolve altogether to attain superior optimization performance. In proposed ACS solution, a co-evolution concept to realize the information interaction amongst the varied sub-population to yield better optimization performance. Thus, implementing the above stated improved ACS model dynamic task scheduling and allied resource allocation. The results obtained by carrying out simulation and its inferences are discussed in the following sections.

IV. Results and Discussion

Ensuring SLA/QoS centric task migration while preserving energy-efficiency is a NP-hard problem, a state of art new Improved ACS model (I-ACS) for VM migration scheduling is applied. Unlike classical heuristic methods, including the conventional ACS or ACO, the proposed method applied multi-population with co-evolution and dynamic pheromone update capacity. This approach not only intended to improve overall scheduling efficiency but also intended to alleviate the problem of local minima and convergence. Thus, performing above stated activities achieves SLA-sensitive and energy-efficient task scheduling in large scale cloud infrastructure. The details of the simulation environment applied is given as follows.

A. Experimental Setup

To simulate the overall proposed model, CloudSim simulation environment and allied benchmark tool is considered. The overall programs were developed in Java programming language and emulation was performed over Java Eclipse platform. Noticeably, the higher scalability, ease of implementation and realistic problem realization was the foundation behind the selection of CloudSim based simulation.

In cloud configuration setup, each host is characterized in terms of corresponding utilization of memory and the performance of Central Processing Unit (CPU). The parameters are Million Instruction Per Second (MIPS), signifying the resource being used or demanded by each task and the resource available onto a host. Moreover, memory (RAM) utilization and bandwidth information of each host as well as VM, which are supposed to be monitored continuously to ensure QoSand SLA oriented task scheduling.

To consider the effectiveness of the proposed task-migration of the VM allocation model, the multiple real-time cloud-computing traces obtained from the CoMon data project, a PlanetLab simulation benchmark (cloud trace) dataset are used. The employed dataset comprised the cloud traffic and allied CPU utilization traces from 1000 plus VMs and allied autonomous tasks, where the different VMs were located at the different locations. The considered benchmark data encompassed the cloud traces over 10 randomly selected data in March and April, 2011. In the considered dataset, the CPU utilization measurement interval was fixed at five minutes. A simulation environment is considered with the system architecture consisting of two heterogeneous servers with dual-core CPUs, one HP ProLiant ML110 G5 with Intel Xeon 3040, 2 cores $\times$1860 MHz processors, armored with 4GB RAM. Additionally, it encompassed HP ProLiant ML110 G5 server with Intel Xeon 3075, 2 cores $\times$ 2660 MHz, 4 GBRAM) to represent a heterogeneous cloud environment. The server's frequency is mapped onto MIPS specifications where HP ProLiant ML110 G4 server was mapped with 1860 MIPS, while for HP ProLiant ML110 G5 server mapping with 2660 MIPS. Each server was armored with 1 Gbps network bandwidth. To assess the efficacy of the proposed task migration or VM allocation (say, resource allocation) model, the performance is obtained in terms of SLA violation (often called, SLAV), SLA downtime, number of migration and energy-consumption. Before discussing the empirical outcomes, a snippet of the different SLA sensitive performance variable is given as follows:

B. The Cost of Tack-Scheduling or VM Migration

Undeniably, the key intent behind the task-migration or allied VM migration is its QoS-affinity or SLA demands. Additionally, this mechanism demands the proposed scheduling model to ensure minimum SLA violation (SLAV), maximum migration with minimum downtime performance. Moreover, maintaining lower energy-consumption has always been the dominant demand from cloud infrastructures. Typically, the SLAV or downtime probability primarily rely on the key factors such as resource demand or memory expected by the different tasks operating onto the VMs, number of memory disks updated over varied execution periods, etc. Under dynamic workload scenarios, the average performance degradation caused due to the downtime is nearly 10% of the overall CPU utilization. Each VM migration introduces a certain SLAV and therefore the minimization of the migration while maintaining SLA performance can be vital. However, maintaining higher task migration without causing any SLAV can also be suitable towards real world application. It seems more realistic under resource constrained scenarios with exceedingly high dynamism. Practically, the migration period relies on the total amount of memory used by the tasks at a certain VM and the available network bandwidth. The migration period for a specific VM, say VM_j can be estimated as per (26).

$$T_{m_j} = \frac{M_j}{B_j} \tag{26}$$

In (26), the memory employed by VM_j is M_j, while the available bandwidth is given by B_j. Here, the focus is on reducing SLAV by maintaining MMT to avoid downtime. To assess performance, the overall performance degradation during the targeted task-scheduling was assessed as per (27).

$$U_{d_j} = 0.1 \cdot \int_{t_0}^{t_0+T_{m_j}} u_j(t)\,dt \tag{27}$$

In (27), the parameter U_{d_j} signifies the overall performance degradation during the task-migration or VM allocation from one host to another, t_0 be the initial migration (start) time, while T_{m_j} be the overall time exhausted during migration. The other parameter $u_j(t)$ is the overall CPU utilization by a nodeVM_j.

C. SLAV Metrics

Considering the SLA objective in cloud infrastructure, the performance of the proposed task scheduling or VM migration model in terms of the different SLAV parameters is examined. To meet QoS and SLA demands, migration model are required to be optimal in delivering minimum throughput and maximum response time. Functionally, these performance parameters change based on the application demands and allied scheduling modalities. The overall SLAV is defined as the disparity in between the demanded MIPS by the tasks or VMs $\left(U_{r_j}(t)\right)$ and the actual assigned MIPS $\left(U_{a_j}(t)\right)$ over the life time of VM (28).

$$SLA = \frac{\sum_{j=1}^{M}\int_t U_{r_j}(t) - U_{a_j}(t)\,dt}{\sum_{j=1}^{M}\int_t U_{r_j}(t)\,dt} \tag{28}$$

In (28), the total number of active VMs is given as M.This work considered MIPS information as well as CPU utilization. Noticeably, here the CPU utilization refers the memory demands which couldn't be assigned when demanded. In the proposed method, distinct two SLA metrics, one the duration through which the active host nodes have experienced 100% CPU utilization, called Overload Time Fraction (OTF); and the performance degradation by VMs (PDM) caused due to VMs migrations have been considered for performance analysis. Here, the value of OTF and PDM is estimated using the following equations (29-30).

$$OTF = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{29}$$

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{30}$$

In (29-30), N represents the total number of active hosts, while the number of active VMs is M. The other parameter T_{s_i} be the total time-period over which the $i-$th host experienced complete (i.e., 100%) resource utilization giving rise to the SLAV. Here, the total number of active hosts or servers are T_{a_i} and C_{d_j} be the performance degradation of VM_j due to migration. In the proposed model, the overall CPU demanded by the cumulative tasks at $VM_j$is C_{r_j}. Since, the above stated SLAV parameters or metrics, OTF and PDM represent SLAV distinctly, and therefore combining the both metrics as a unified performance parameter named SLAV, which is defined as (31).

$$SLAV = OTF.PDM \tag{31}$$

The detailed discussion of the simulated performance outcomes in terms of the above discussed SLA performance metrics, downtime and energy is given as follows: Unlike major classical researches such as [1-5], authors have focused on assessing resource scheduling performance based on the parameters like make span, scheduling time, etc.; however, could not assess whether their approach delivers SLA or not. Unlike the performance assessment in terms of makeover or scheduling time, a real-world cloud infrastructure, especially IaaS often demands ensuring minimum or even negligible downtime, SLAV, etc. Moreover, assessing their suitability in terms of energy is equally significant. Therefore, taking into consideration of this fact, in this research the performance of the proposed system is examined in terms of the following parameters:

No. of VM migrations,

SLA-Violation (SLAV),

SLA performance degradation,

SLA Violation per active host,

Host Shut-Down,

Energy-Consumption.

Amongst the above stated performance metrics, 2, 3, and 4 represents robustness of the scheduling methods towards SLA assurance or QoS. On the contrary, 1 and 5 presents scalability of the proposed cloud model, while 7 indicates swiftness. Though, 1, 3 and 5 are highly dependent. Similarly, 6th performance metrics indicate the energy-efficacy by the proposed model. Noticeably, for an SLA-oriented solution a task scheduler requires maintaining a greater number of migrations while maintaining negligible SLAV, SLAV per active host, and scheduling time. On the contrary, higher number of active hosts shut down indicates energy-convergence ability by the proposed model. To compare the performance by the proposed model i.e. I-ACS model, with other recent approaches as well; though these methods examined their performance in the different terms like make-span or time over varying tasks. Noticeably, scheduling methods are considered as the foundation and performed task-migration hypothesizing that each VM carries a single operating task, and hence the task migration can be realized as a classical VM-consolidation or migration problem. Thus, with this hypothesis, three different existing approaches as mentioned in [2], [3] and [4] are implemented.

Velliangiri et al. has focused on improving heuristic model to achieve better performance and local minima and convergence avoidance. In this regard, authors [2] designed a

Hybrid Electro Search with GA (HESGA) algorithm for task-scheduling. To achieve better performance, authors applied GA to obtain local optimal solution, while Electro Search algorithm was applied to improve global optima solution. However, authors failed in addressing the dynamism of the resource demands under uncertain predefined heterogeneous (dynamic) clouds. Recalling the fact, unlike [2], where authors applied static threshold-based hotspot detection, to cope up with the exceedingly dynamic cloud environment IQR-LRR based stochastic prediction concept for overloading detection is applied, which helped making task-scheduling on time and hence preserved SLA performance. Recently, an improved effort was made in [3], where Liu et al. [3] proposed an improved GA based collaborative scheduling concept for cloud infrastructure. With the same intend as [2], or the proposed I-ACS model, authors [3] targeted on avoiding local minima and convergence problems for better scheduling.

Xiang et al. [4] recently proposed the Greedy-ACO algorithm for workflow scheduling in heterogeneous cloud environments. To be noted, there are a large number of existing method or literatures discussing heuristic based task scheduling, VM consolidation and VM migration; however, considering these three key recent methods which not only intend to perform task-scheduling, but also address the existing drawbacks of the major existing methods such as local minima and convergence.

Recalling the fact that the considered cloud traces or benchmark data was taken from PlanetLab datasets, to examine or simulate the proposed model (as well as the existing methods [2-4] over the different datasets. More precisely, the proposed model is executed with the cloud traces obtained 03 March 2011, 06 March 2011 09 March 2011, 22 March 2011, 25 March 2011, 03 April 2011, 09 April 2011, 11 April 2011, 12 April 2011 and 20 April 2011. Thus, simulating the different methods, including the proposed I-ACS model obtains performance outputs in terms of 1-6 metrics. To generalize the performance over multiple test instances or cases, the average performance is considered. The outputs obtained in terms of the different SLA metrics is given as follows:

Fig. 3 presents the number of VM migrations by the different techniques. After the observations, the overall results obtained by the proposed I-ACS model show a higher number of task migration, exhibiting robustness towards superior scalability. It is further be identified in terms of the minimum SLA violation and downtime, as depicted in Fig. 4 to Fig. 6. Noticeably, literature hypothesizes that maintaining a lower number of migrations can avoid any likelihood of SLAV; however, the proposed model has exhibited on the contrary, affirming that one can achieve superior SLA performance even with a higher number of migrations. Since, in the proposed model, each VM was considered as one autonomously operating task, scheduling a larger number of tasks shows the superior scalability by the proposed method. It affirms robustness of the proposed model towards realistic mega data center applications.

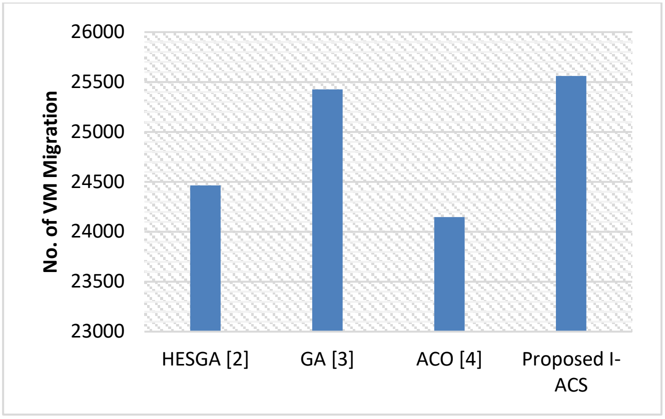

Fig. 3. Number of VM Migrations using different Techniques.

Fig. 4 presents the SLA violation, here called SLAV. The observations with overall results achieved by the proposed I-ACS model shows better than other existing approaches; however, its performance is far better than the classical ACO algorithms. This performance enhancement could be contributed because of multiple-generation, dynamic pheromone update and co-evolution concept. Statistically, I-ACS model has exhibited almost 0.03% of SLA violation, which shows its robustness. A similar performance was observed in terms of SLA performance degradation per host (Fig. 5). As depicted in Fig. 5, the proposed method performs superior over other heuristic based scheduling. To be noted, since HESGA [2] and improved GA [3] algorithms were developed similar to the proposed I-ACS concept, where the key focus was made on alleviating the at hand local minima and convergence and hence these approaches showed better performance than the classical ACO based scheduling. However, these methods [2][3], due to the lack of adaptive overloading or hotspot detection and dynamic scheduling (performed using multiple controller-based systems), were found inferior than the proposed model.

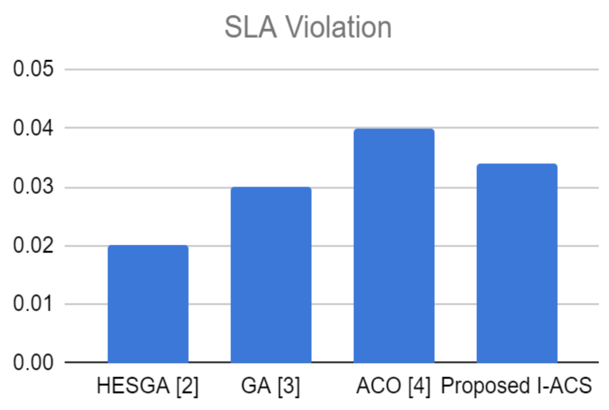

Fig. 4. SLA Violation (SLAV) Performance by the different Techniques.

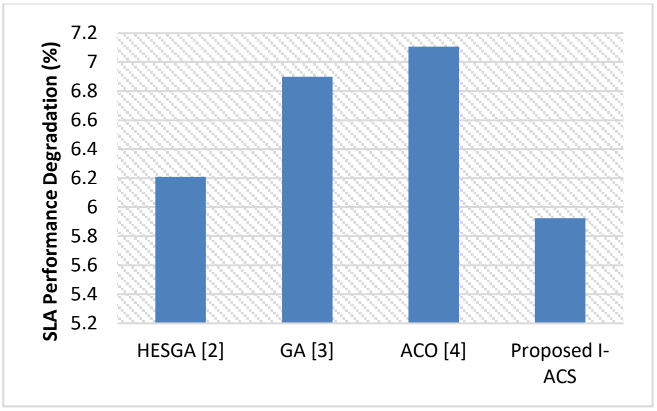

Fig. 5. SLA Performance Degradation by the different Techniques.

A similar performance was found in SLA per active host (Fig. 6). Observing overall performance, it can easily be found that the proposed multi-controller assisted I-ACS based task-scheduling model achieves better SLA performance and eventual QOS to meet major cloud computing demands. In terms of time of execution, Fig. 6 reveals that the proposed I-ACS model exhibits superior in terms of the SLA time per-active host (second), signifying very small or near tolerable downtime. The comparative outcomes too reveal that the proposed model shows almost 18% lower downtime than other heuristic based approaches.

Considering about the number of hosts shut-down, Fig. 7 reveals that the proposed I-ACS based task-scheduling model exhibits a higher number of host-shut down, signifying better energy-efficiency and optimal resource utilization.

Fig. 8 can be found in affirmation, where the proposed I-ACS model has exhibited almost 8% lower energy than the classical ACO based scheduling. Noticeably, in Fig. 8, the energy consumption by GA variants is relatively higher. This could be because of the predefined number of stopping criteria (considering 200 number of generations). It could have taken more time for computation and hence higher energy exhaustion. Thus, considering the overall performance outputs, it can be stated that the proposed I-ACS based model achieves superior performance than other existing (recent) heuristic based task-scheduling systems or resource allocation (say, VM migration) methods. The overall research conclusion and its related inferences are given in the subsequent sections.

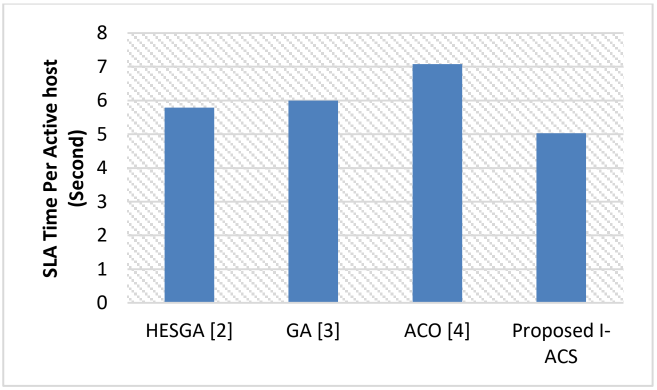

Fig. 6. SLA Time Per Active Host (sec.) by different Techniques.

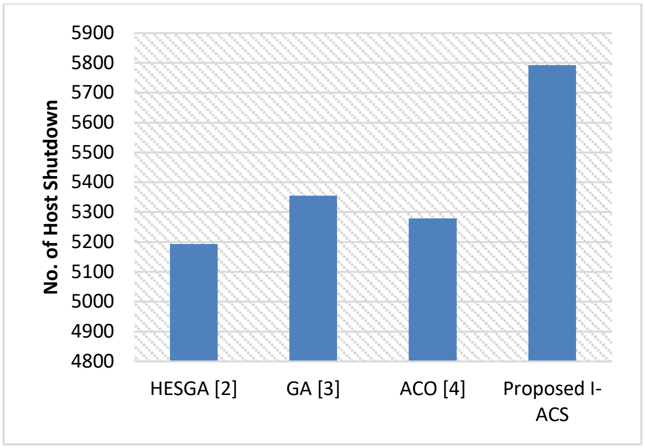

Fig. 7. No. of Host Shut-down by the different Techniques.

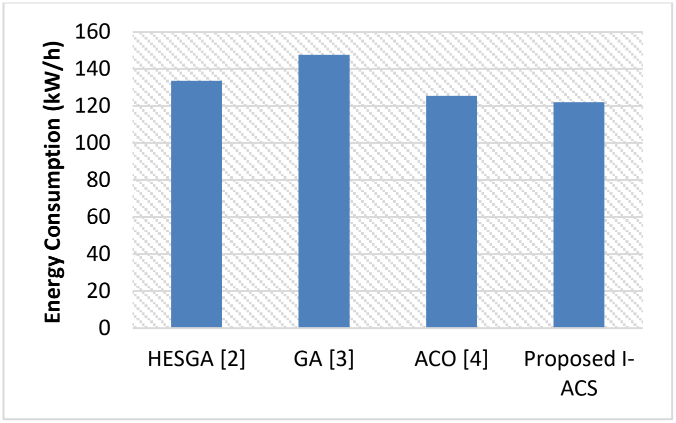

Fig. 8. Energy Consumption by the different Techniques.

V. Conclusion

The research work primarily focused on improving the task-scheduling and allied dynamic resource allocation to meet SLA-centric cloud services. To meet contemporary as well as future demands including QoS, SLA-agreement and energy-efficiency, the proposed work introduced multiple enhancement at the different levels of computation. The proposed model applied multi-controller strategies, where the use of local controllers enabled task-level resource utilization assessment and stochastic prediction-based overloading or underloading detection avoiding any possible downtime. The proposed local controller applied minimum migration time based VM selection strategy that greatly helped for timely task-migration scheduling. Eventually, exploiting the task and possible target host information, the proposed involves improved multi-population, adaptive or dynamic pheromone update and co-evolution-based I-ACS model which performs dynamic task-migration or allied resource scheduling. The overall proposed I-ACS model not only enabled superior task-migration but also avoided any possible local minima and convergence problem. This as a result affirmed optimality of the proposed solution exhibiting superior performance in terms of minimum SLA violation, minimum downtime, lower energy consumption and higher number of task-migration.

REFERENCES

[1] S. Pang, W. Li, H. He, Z. Shan and X. Wang, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing," in IEEE Access, vol. 7, pp. 146379-146389, 2019. DOI:10.1109/ACCESS.2019.2946216.

[2] S. Velliangiri, P. Karthikeyan, V.M. Arul Xavier, D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing", Ain Shams Engineering Journal, pp. 1-9; July 2020. DOI:10.1016/j.asej.2020.07.003.

[3] S. Liu and N. Wang, "Collaborative Optimization Scheduling of Cloud Service Resources Based on Improved Genetic Algorithm," in IEEE Access,vol.8,pp.150878-150890,2020. DOI:https://doi.org/10.1155/2021/5582646.

[4] B. Xiang, B. Zhang and L. Zhang, "Greedy-Ant: Ant Colony System-Inspired Workflow Scheduling for Heterogeneous Computing," in IEEE Access,vol.5,pp.11404-11412,2017.DOI: 10.1109/ACCESS.2017.2715279.

[5] S. G. Domanal, R. M. R. Guddeti and R. Buyya, "A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment," in IEEE Transactions on Services Computing, vol. 13, no. 1, pp. 3-15, 1 Jan.-Feb. 2020. DOI: 10.1109/TSC.2017.2679738.

[6] Afzal, S., Kavitha, G. Load balancing in cloud computing – A hierarchical taxonomical classification. J Cloud Comp 8, 22 (2019). https://doi.org/10.1186/s13677-019-0146-7.

[7] Pradhan, P., Behera, P.K. and Ray, B.N.B., 2016. Modified round robin algorithm for resource allocation in cloud computing. Procedia Computer Science, 85, pp.878-890.https://doi.org/10.1016/j.procs.2016.05.278.

[8] Moges, F., Abebe, S. Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework. J Cloud Comp 8, 2 (2019). https://doi.org/10.1186/s13677-019-0126-y.

[9] Syed Arshad Ali, Samiya Khan, MansafAlam, Resource-Aware Min-Min (RAMM) Algorithm for Resource Allocation in Cloud Computing Environment, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-3, September 2019. Pp 1863-1870 DOI: https://doi.org/10.35940/ijrte.c5197.098319.

[10] Mosa, A., Paton, N.W. Optimizing virtual machine placement for energy and SLA in clouds using utility functions. J Cloud Comp 5, 17 (2016). https://doi.org/10.1186/s13677-016-0067-7.

[11] J. R. Doppa, R. G. Kim, M. Isakov, M. A. Kinsy, H. Kwon and T. Krishna, "Adaptive many core architectures for big data computing: Special session paper," 2017 Eleventh IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Seoul, pp. 1-8, 21017. DOI: https://doi.org/10.1145/3130218.3130236.

[12] Z. Li, J. Ge, H. Hu, W. Song, H. Hu and B. Luo, "Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds," in IEEE Transactions on Services Computing, vol. 11, no. 4, pp. 713-726, 1 July-Aug. 2018. DOI: https://doi.org/10.1109/TSC.2015.2466545.

[13] K. Li, "Power and performance management for parallel computations in clouds and data centers," J. Comput. Syst. Sci., vol. 82, no. 2, pp. 174–190, Mar. 2016. DOI: https://doi.org/10.1016/j.jcss.2015.07.001.

[14] 28 -30 Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energyefficient task scheduling algorithm in DVFS-enabled cloud environment," J Grid Comput., vol. 14, no. 1, pp. 55–74, Mar. 2016. DOI: DOI:10.1007/s10723-015-9334-y.

[15] G. Xie, L. Liu, L. Yang, and R. Li, "Scheduling trade-off of dynamic multiple parallel workflows on heterogeneous distributed computing systems," Concurrency Comput.-Parctice Exp., vol. 29, no. 8, pp. 1–18, Jan. 2017. DOI:10.1002/cpe.3782.

[16] G. Zeng, Y. Matsubara, H. Tomiyama, and H. Takada, "Energy Aware task migration for multiprocessor real-time systems," Future Gen.Comput. Syst., vol.56, pp.220–228,Mar.2016. https://doi.org/10.1016/j.future.2015.07.008.

[17] Z. Zhu, G. Zhang, M. Li and X. Liu, "Evolutionary Multi-Objective Workflow Scheduling in Cloud," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 5, pp. 1344- 1357, 1 May 2016. DOI: 10.1109/TPDS.2015.2446459.

[18] Jyothi.S, Dr.B.S.Shylaja, "Efficient Approach for Resource Provisioning to manage Workload in Cloud Environment" in International Journal of engineering Research and Technology(IJERT),ISSN 2278-0181,Vol 9,Issue 06,June2020.DOI:http://dx.doi.org/10.17577/IJERTV9IS060979.

[19] Bhaskar, Shylaja.B.S (2019)"KBR Knowledge Based Reduction Method for Virtual Machine Migration in Cloud Computing", International Conference on Recent Trends in Advanced Computing 2019, ICRTAC-2019 published in Elsevier Procedia Computer Science 00 (2019) 000–000. DOI: https://doi.org/10.1016/j.procs.2020.01.026.