Optimized Load-balancing in Cloud Computing based on Traffic and Workload-aware VM Migration

GOWRIPRAKASH R[*1], SHANKAR R[*2], DURAISAMY S[*3]

Ph.D., Research Scholar[*1], *Department of Computer Science, Chikkanna Government Arts College, Tirupur,* Tamilnadu, India

Assistant Professor[*2&3], *Department of Computer Science, Chikkanna Government Arts College, Tirupur,* Tamilnadu, India

ABSTRACT

In cloud storage systems, Virtual Machine (VM) migration approaches play a major function in minimizing the power usage and balancing the tasks in the data centers. Owing to the massive amount of clients and applications trying to gain from cloud service, it makes it a complex process for the edge cloud servers to operate in an energy-saving mode. Also, while maximizing the bandwidth between VMs, the chance of traffic congestion presence is increased. To solve this problem, this article proposes an Osmotic Hybrid artificial Bee and Ant Colony with Future Utilization Prediction (OH-BAC-FUP) approach to lessen the number of VM migrations and enhance the load-balancing. First, both ongoing and upcoming resource consumption of PMs and VMs are determined using Linear Regression (LR) and Optimal Piecewise LR (OPLR) schemes. The determined values are given to the OH-BAC which computes the fitness value for choosing the optimal VM to be migrated to the optimal PM. Also, this is further enhanced by proposing a Multipath Traffic Routing (MTR) called OH-BAC-FUP-MTR approach which prevents the chance of congestion while migrating VM to the PM. If any congestion exists because of high bandwidth or traffic flows, then the flows are split into many segments and transfer them via multiple link-disjoint routes. Moreover, Merge-and-Split-based Coalitional Game-theoretic (MSCG) scheme is proposed with OH-BAC-FUP-MTR approach to achieve a proper tradeoff between task reliability and power conservation in heterogeneous data centers. This MSCG is used for splitting the PMs into many sets and choosing the members from those sets to generate effective coalitions. For each coalition, OH-BAC-FUP-MTR is executed which enhances the payoff of each coalition and sustains PMs to operate in a high energy-efficient state. Finally, the simulation results reveal the OH-BAC-FUP-MTR-MSCG accomplishes better efficacy than the classical VM migration approaches.

Keywords—Cloud computing, VM migration, Load-balancing, Resource consumption, Flow congestion, Osmotic computing.

I. INTRODUCTION

In general, the cloud computing model faces numerous challenges as its adoption gradually increases. The greatest significant problem in the cloud paradigm is managing facilities. To resolve this concern, load-balancing approaches have been utilized, which improves complete system efficiency, robustness, availability, and other characteristics in data centers. Load-balancing is primarily used to handle task inconsistency between data centers in order to mitigate over- and under-loading. The load-balancing approaches are strengthened by establishing the Service Level Agreement (SLA) and service quality. For this reason, effectual load-balancing frameworks are an important component of cloud computing. As a result, different approaches for load-balancing and task distribution processes in cloud servers have been developed [1-4].

In recent years, there has been a significant increase in the updated model of osmotic computing, which was accompanied by the concept of the substance osmotic characteristics. It is typically applied to ensure the long-term adoption of resources in significant cloud applications [5-6]. It is used in data centers to enable the use of stable VMs that are relocated to edge machines. Although various design methods ensure optimal performance in several load-balancing approaches, some of them lack the possibility to boost efficacy in every aspect. Accordingly, an OH-BAC approach [7] was designed to limit resource utilization, the ratio of VM migrations, and the percentage of host shutdowns. However, it only limits a small number of engaged PMs depending on the current available resources, leaving out forthcoming resource requirements. Due to this, unnecessary VM relocations took place, and the percentage of SLA Violations (SLAV) in data center nearly doubled.

As a result, this paper develops an OH-BAC-FUP approach to lower the percentage of VM migrations while improving load-balancing effectivity. In this approach, the upcoming and ongoing resource consumption are determined in order to relocate the VMs to the smaller engaged PMs. For this purpose, CPU, bandwidth, storage size, and RAM of both VMs and PMs are calculated. These calculated measurements are fed into the OH-BAC's fitness value, which is used to select the best PM. Conversely, if the bandwidth consumed between VMs within the data center is large, there is a great risk of overcrowding. Inadequate VM optimization leads to inter-VM distribution for connecting bottleneck system connections, leading to increased overcrowding.

Furthermore, main node over-provisioning and unstable distributions in data centers result in long-lasting traffic flow. When there is still a long queue, the resource use efficacy deteriorated quickly. Many studies concentrated on the effects of VM migration on traffic; however, the main goal is to minimize overall energy consumption in a data center [8]. Path failures possess serious consequences because data centers deliver a large amount of traffic. 100% traffic protection is also founded if resources will be involved with satisfactory recovery available bandwidth. Traffic is often ignored partially because reserving available resources is expensive. Even if limited protection will make sure accessibility, narrower bandwidth and feasibility are indeed optimal for a variety of applications.

As a result, an OH-BAC-FUP-MTR approach is proposed for proper load-balancing and reducing the possibility of overcrowding in data centers. The flow is first distributed on several link-disjoint routes to prevent unwanted system failures and ensure the affordability of at least one route for a flow in the event of route damage or overcrowding. If there is heavy traffic while relocating VMs to PMs, the MTR protocol is used, which divides traffic into several forms and sends it via various link-disjoint routes. Furthermore, the flow rate on the link is used as a congestion level. But, the trade-off between task reliability and power conserving in heterogeneous data centers is not measured.

Hence, an OH-BAC-FUP-MTR-MSCG approach is further proposed to switch the idle PMs into hibernation mode, resulting in very less energy use. A common classification is used to ensure the balanced workload during distribution. The VM migration is targeted at consolidating the VMs depending on the workload to the less number of PMs for reducing the power use and encouraging green computing. Bu using this approach, the PMs are split into various groups depending on their task ranges and then a MSCG method is applied to select members from these groups to create efficient coalitions. Also, an OH-BAC-FUP-MTR is performed among the coalition members for maximizing the payoff of each coalition and PMs are maintained to operate in a high energy-efficient state. Thus, the tradeoffs between task reliability and energy use are balanced during VM migration significantly.

The rest of the article is prepared as follows: Section II studies the previous works related to the VM migration approaches. Section III explains the methodology of proposed approaches and Section IV illustrates their experimental results. Section V concludes the research work.

II. LITERATURE SURVEY

An adaptive scheduling algorithm [9] was presented to balance the workload across all VMs using flexible resource handling and de-provisioning based on the best possible k-time. The load at each VM and storage device was regularly estimated. If any of the VMs became congested, the workload migration strategy was used to move the task from the congested VM to the under-loaded

VM. But, the QoS parameters must be increased in order to guarantee high-priority requests. Efficient load-balancing scheme is needed to improving the performance of cloud computing. Several load-balancing algorithms in cloud computing have been proposed by different researchers in the past years. In this paper, some of them are surveyed with those merits and demerits to further enhance the load-balancing in cloud using recent algorithms.

A meta-heuristic algorithm known as the dominant firefly algorithm [10] was formulated to solve imbalance problems on data centers. An organic computing model was implemented in this algorithm based on the assumption that a firefly from the cluster of fireflies to the nearer percentage consumes a less power in its tail. But, the client performance and total CPU use were ineffective. Hejja and Hesselbach [11] developed an integrated energy-aware resource distribution with VM consolidation technique that supports PM relocations in order to reduce data centre overall costs in an offline scenario. This method was also merged with an optional standalone traffic migration scheme that was activated based on certain criteria. However, its efficiency was reduced because it did not take into account LB and traffic predictions.

Hsieh et al. [12] proposed a VM consolidation strategy that takes into account the ongoing and prospective usage of available resources by identifying the overloaded and under-loaded hosts. The gray-Markov-based scheme was used to accurately predict the resource usage in the future. However, it does not take into account flow of traffic during relocation to resolve any difficulties. Afzal and Kavitha [13] created a hybrid multiple parallel queuing model to enhance cloud platform Quality-of-Service (QoS). However, it supposes that the flow arrival rate and service rate seem to be fixed.

Gholipour et al. [14] created a novel cloud resource management process based on multi-criteria decision-making. The Joint VM and Container Multi-criteria Migration Decision (JVCMMD) method was implemented. After this, to address the joint VM and container migration problem, an innovative system was established. In addition, a multi-criteria migration decision strategy was required to evaluate whether VMs or containers should be relocated, which simultaneously controls energy consumption, the amount of VM and container migrations. But, other criteria such as memory, network bandwidth, and RAM must be considered in order to improve the VM consolidation effectiveness.

Yun et al. [15] created an adaptive harmony search scheme to determine data centre energy consumption while incentivizing a stable process through VM consolidation. This investigation was aimed for a virtualized environment that distributes PM resources to a large number of VMs while accounting for CPU, memory, and network resources. In addition, a migration cost was calculated in order to reduce energy consumption and avoid VM migration for sustainable assignments. However, if boosting resource utilization, then it does not predict the tasks in order to preserve the various qualities of VMs.

III. PROPOSED METHODOLOGY

In this section, the proposed VM migration approaches are described in detail. First, LR and OPLR schemes are applied to determine the ongoing and upcoming resource uses for both VMs and PMs. Also, OH-BAC is performed to choose the most suitable VMs and PMs for migrating the VMs properly. Then, MTR algorithm is introduced during VM migration to prevent the traffic congestion in the network or to handle the path failures effectively. Moreover, MSCG scheme is proposed to split the PMs into different coalitions based on their tasks and the OH-BAC-MTR approach is applied in each coalition for reducing the difficulty of VM migration and enhancing the resource utilization efficiency. Figure 3.1 portrays the overall block diagram of proposed VM migration-based load balancing system in the heterogeneous cloud computing.

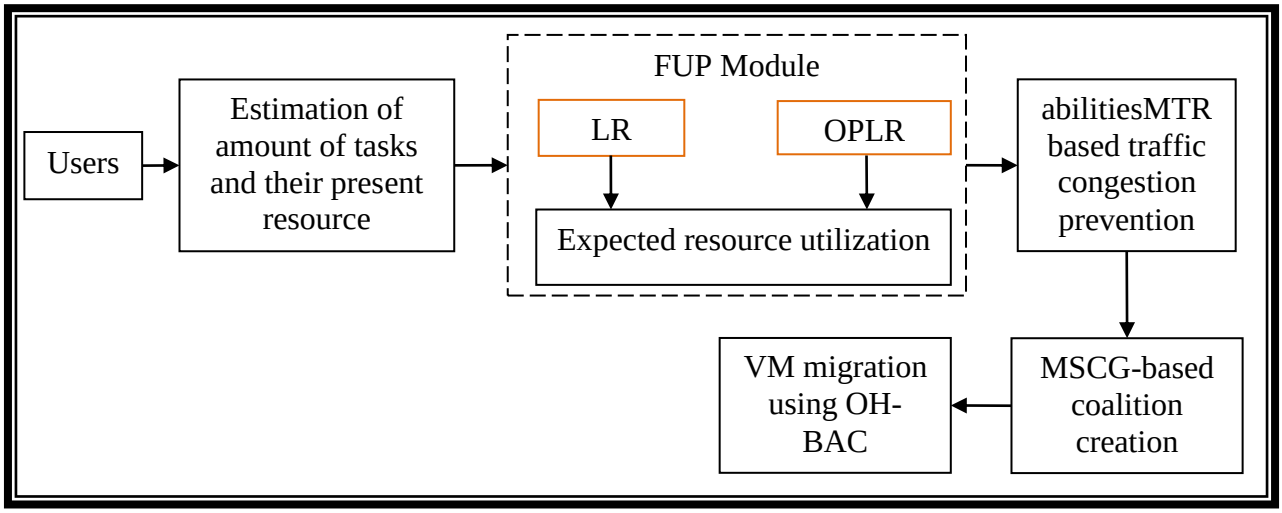

Figure1: Overall Block Diagram of Proposed VM Migration-based Load-balancing System in Heterogeneous Cloud Platform

3.1 Current & Future Resource Utilization Prediction

Consider a whole PM in cloud server, $\mathcal{P} = \{p_1, p_2, \dots, p_i\}$ where i is the number of PMs and a whole VM, $\mathcal{V} = \{v_1, v_2, \dots, v_j\}$ where j is the number of VMs. Every $\mathcal{P}$ include ψ type of resources i.e., CPU, bandwidth, memory and storage size. A client may request v VMs which are migrated to the PMs at a specific duration. To optimize this migration, OH-BAC-FUP is first applied. All p comprises a n-dimensional total potential vector $\mathcal{C}_p = \{\mathcal{C}_p^1, \mathcal{C}_p^2, \dots, \mathcal{C}_p^n\}$ where $\mathcal{C}_p^n$ is the overall n^{th} resource facility of p. Let the ongoing potential vector of p, $\mathcal{U}_p = \{\mathcal{U}_p^1, \mathcal{U}_p^2, \dots, \mathcal{U}_p^n\}$ where $\mathcal{U}_p^n$ is the ongoing facility of n^{th} resource. In the same manner, consider a total potential vector of v, $\mathcal{C}_v = \{\mathcal{C}_v^1, \mathcal{C}_v^2, \dots, \mathcal{C}_v^n\}$ where $\mathcal{C}_v^n$ is the overall n^{th} resource usage by v and n-dimensional consumed potential vector $\mathcal{U}_v = \{\mathcal{U}_v^1, \mathcal{U}_v^2, \dots, \mathcal{U}_v^n\}$ where $\mathcal{U}_v^n$ is the consumed n^{th} resource by v.

Here, the OH-BAC-FUP is applied to migrate VMs to the most appropriate PMs optimally by using LR and OPLR algorithms [16]. These algorithms can estimate the consumption of CPU, bandwidth, memory and storage size for VMs as well as PMs. First, the LR exploits the details on the prior consumption of resources and determines the association between the ongoing and upcoming consumption of resources as:

$$\chi_p = a + b\mathcal{U}_p \quad (1)$$

In Eq. (1), χ_p and $\mathcal{U}_p$ are the predicted and ongoing potential vector of p, accordingly, a and b are the regression variables which characterize the Y-intercept and slope of the line, accordingly. Also, the resource consumption of the VM is determined as:

$$\chi_v = a + b\mathcal{U}_v \quad (2)$$

In Eq. (2), χ_v and $\mathcal{U}_v$ stand for the predicted and ongoing potential vector of v, accordingly. In the same way, the OPLR is applied to determine the upcoming consumption of resources as:

$$\chi_p = \left(\sum_p \mathcal{W}_p \mathcal{U}_p\right) + \mathcal{B}_p, \forall p \quad (3)$$

In Eq. (4), $\mathcal{B}_p$ and $\mathcal{W}_p$ indicate the regression variables and the resource consumption of the VM is determined as:

$$\chi_v = (\sum_v \mathcal{W}_v \mathcal{U}_v) + \mathcal{B}_v, \forall v \quad (4)$$

Afterwards, the standard variance (σ) is calculated for all p using the scout beewhich recognizes both under and over-loaded hosts. The mean task of j^{th} v in i^{th} p $\left(\bar{\mathcal{V}}_{ij}\right)$ is calculated as:

$$\bar{\mathcal{V}}_{ij} = \left(\mathcal{U}_{c_j} \cdot \chi_{c_j}\right) + \left(\mathcal{U}_{m_j} \cdot \chi_{m_j}\right) + \left(\mathcal{U}_{\mathcal{b}_j} \cdot \chi_{\mathcal{b}_j}\right) \quad (5)$$

In Eq. (5), $\mathcal{U}_{c_j}, \mathcal{U}_{m_j}$ and $\mathcal{U}_{\mathcal{b}_j}$ are the ongoing CPU, memory and bandwidth consumptions of j^{th} v, accordingly. Similarly, χ_{c_j}, χ_{m_j} and $\chi_{\mathcal{b}_j}$ are the expected CPU, memory and bandwidth consumptions of j^{th} v, accordingly. If the task variance of $\mathcal{P}_i$ is lower than the least threshold, then $\mathcal{P}_i$ is called under-loaded host, otherwise, it is over-loaded host. After that, scout bee moves in the knowledge base to alert every swarm regarding whether the given p is under-loaded or over-loaded host. Then, the fresh host's record is forwarded to the ACO to obtain the suitable p among each osmotic host and shifting v from it. In this step, the ACO determines its fitness (θ_x) depending on the over and under-loaded hosts as:

$$\theta_x = \frac{(\rho_x)^k * (\omega_x)^l}{\sum_{x=1}^{j} (\rho_x)^k * (\omega_x)^l} \quad (6)$$

$$\theta_x = \frac{\left(1/\rho_x\right)^k * (\omega_x)^l}{\sum_{x=1}^{j} \left(1/\rho_x\right)^k * (\omega_x)^l} \quad (7)$$

In Eqns. (6) & (7), k and l offer a relative importance between pheromone ρ_x and edge weight ω_x (bandwidth). Here, ρ_x is defined by the task of $\mathcal{P}_i$. So, the most suitable $\mathcal{P}_i$ is decided and alert each swarm by knowledge base to assign this $\mathcal{P}_i$ for VM migration process. Consequently, ρ_x is modified as:

$$\rho_x = (1 - e)\rho_x + \Delta\rho_x \quad (8)$$

In Eq. (8), e $(0 < e < 1)$ is the pheromone evaporation ratio and $\Delta\rho_x$ is calculated as:

$$\Delta\rho_x = \frac{1}{(\omega_x)^{\varepsilon}} + \rho_x \quad (9)$$

In Eq. (9), ε is the variable to describe the proportional importance of the heuristic ratio associated with the task criteria. Moreover, employed bees perform its estimations to decide the suitable VM to be shifted to another host by the Least Migration Interval (LMI) scheme which defines the % of memory consumed for the VM to the additional bandwidth available for the $host_x$ as:

$$\frac{(\mathcal{U}_{m_s} \cdot \chi_{m_s})}{E\mathcal{b}_i} \leq \frac{(\mathcal{U}_{m_u} \cdot \chi_{m_u})}{E\mathcal{b}_i}, \forall s, u \in \mathcal{V}_j \quad (10)$$

In Eq. (10), $\mathcal{V}_j$ is the set of VMs recently migrated to $host_x$ and $E\mathcal{b}_i$ is the additional bandwidth available for $host_x$. Also, $\mathcal{U}_{m_s}$ and $\mathcal{U}_{m_u}$ indicate the memory quantity recently consumed by $\mathcal{V}_s$ and $\mathcal{V}_u$, accordingly, χ_{m_s} and χ_{m_u} indicate the expected memory quantity consumed by $\mathcal{V}_s$ *and* $\mathcal{V}_u$, accordingly. Further, the fitness value of ACO is computed to get the optimal mapping association between the selected VMs to be collaborated with the most suitable PM as:

$$\theta(\mathcal{V}_j, \mathcal{P}_i) = \frac{(\mathcal{U}_{c_i} \cdot \chi_{c_i}) - (\mathcal{U}_{c_j} \cdot \chi_{c_j})}{(\mathcal{U}_{c_j} \cdot \chi_{c_j})} \cdot \frac{(\mathcal{U}_{m_i} \cdot \mathcal{U}_{m_i}) - (\mathcal{U}_{m_j} \cdot \chi_{m_j})}{(\mathcal{U}_{m_j} \cdot \chi_{m_j})} \cdot \frac{(\mathcal{U}_{E\mathcal{b}_i} \cdot \chi_{E\mathcal{b}_i}) - (\mathcal{U}_{E\mathcal{b}_j} \cdot \chi_{E\mathcal{b}_j})}{(\mathcal{U}_{E\mathcal{b}_j} \cdot \chi_{E\mathcal{b}_j})} \cdot \frac{(\mathcal{U}_{ss_i} \cdot \chi_{ss_i}) - (\mathcal{U}_{ss_j} \cdot \chi_{ss_j})}{(\mathcal{U}_{ss_j} \cdot \chi_{ss_j})} \quad (11)$$

In Eq. (11), $\mathcal{U}_{c_i}, \mathcal{U}_{m_i}, \mathcal{U}_{E\mathcal{b}_i}$ and $\mathcal{U}_{ss_i}$ are the recently consumed VM's resources i.e., CPU use, memory, additional bandwidth and storage size, accordingly and $\chi_{c_i}, \chi_{m_i}, \chi_{E\mathcal{b}_i}$ and χ_{ss_i} are the expected VM's resources use. In the same manner, $\mathcal{U}_{c_j}, \mathcal{U}_{m_j}, \mathcal{U}_{E\mathcal{b}_j}$ and $\mathcal{U}_{ss_j}$ indicate the recently consumed PM's resources and $\chi_{c_j}, \chi_{m_j}, \chi_{E\mathcal{b}_j}$ and χ_{ss_j} indicate the expected PM's resources use. Next, onlooker bees can accept the VM information from employee bees and decide the suitable PM for VM migration. Eventually, onlooker bees perform VM reassignment through migrating it to the suitable PM.

In addition to this, consider a task's resource necessity $\mathcal{J} = \{r_1, r_2, \ldots, r_y\}$ where r_k refers to the number of v_j. The resource demand of v_j is $v_j = \{v_{c_j}, v_{m_j}, v_{\mathcal{b}_j}, v_{ss_j}\}$ where $v_{c_j}, v_{m_j}, v_{\mathcal{b}_j}$ and v_{ss_j} are the required CPU, memory, bandwidth and storage size. Assume

that there is a combination of flows in a cloud server requesting various bandwidths. Therefore, the exact bandwidth demand of flow between any VMs is not simple. To solve this problem, assume that tenants involve a good determination of bandwidth requirements based on the categories of applications and their competence.

The main problem in this OH-BAC-FUP is to find a mapping between VMs and PMs that satisfies the VM's resource requirements whilst intending to lessen the overcrowding and providing a protection guarantee of level Υ, i.e., the % of bandwidth guaranteed to be available for flow on a route failure. The maximum flow on a route is applied as a congestion variable. The cloud server is formulated using a directed graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ wherein $v \in \mathbb{V}$ is the PM and $(u, v) \in \mathbb{E}$ is a significant route linking the PMs. Each route (u, v) has a non-negative capacity $ct(u, v) \geq 0$ and forwards one or many flows. On $(u, v) \in E$, bandwidth $\mathscr{b}_{i,u,v}$ is engaged for flow i. For every task, the PMs are chosen to help its significant resource requirements and a set of paths in $\mathcal{G}$ for transferring flows between VMs.

3.2 Multipath Traffic Routing

As a result, (17) MTR is designed wherein the flow is mapped to many paths involving a particular flow percentage for minimizing the overcrowding whilst providing a specific Υ. If flow demand (i) with the bandwidth $(\mathscr{b})$ is partitioned and sent across n link-disjoint paths at the percentage of $\mathscr{b}_{max}$ (highest $\mathscr{b}$), then the minimum Υ is provided by $[\mathscr{b} - \mathscr{b}_{max}]/[\mathscr{b}]$ which happens when the path with $\mathscr{b}_{max}$ is failure.

The purpose of MTR is to lessen the maximum flow on the route while providing Υ as the least of Γ for all demands. The objective function and constraints related to the VM migration and MTR are:

$$min(L_{max}) \tag{12}$$

Constraints:

- The highest load L_{max} is $L_{max} \geq g_{u,v}, \forall u, v \in \mathbb{V}$.
- An entire bandwidth or flow applied on (u, v) is the overall flow i transmitting via the route.
- Capacity constraints: An entire resource necessity of the VMs positioned on the host should not go beyond its capacity. For PM j, an overall resource necessities of all VMs positioned on it have to be less than or equal to the host's total available capacity.
- Migration constraint: All VMs are migrated to single PM.
- Bandwidth constraint: An overall bandwidth applied on (u, v) have not to go beyond its $ct(u, v)$.
- No malfunction constraint: For all u which is not the actual PM s_i or the intended PM t_i of i, the accepted bandwidth have been equal to the forwarded bandwidth.
- Flow splitting constraint: Flow i is split into n with $\mathscr{b}$ to be shifted onto many paths.
- Flow routing on link-disjoint paths: (u, v) is transferred via only one of the n paths of i. As well, it ensures that n paths exploit disjoint sets of routes.
- Flow splitting at the source: A total flow % of s_i is equal to $\mathscr{b}$.
- Flow merging at the intended PM: A total flow % of t_i is equal to $\mathscr{b}$.
- Protection constraint: The % of bandwidth available for i on a route failure is not lower than Υ.

During MTR process [17], paths and their flow % are determined. The availability of many paths between s_j and VM t_j is leveraged to distribute the flow across the paths and partially protect the flow from a specific route failure. For each pair of cooperating VMs, many link-disjoint least-congested paths are estimated. Also, the flow is split only at the boundary and it follows the pre-selected paths with no demand of extra splitting.

Creation of Routing and Flow Splitting

Intra- and Inter-pod routing mechanisms are developed to choose the paths for a flow. Intra-pod routing exists if a pair of cooperating hosts is positioned in a same pod, but inter-pod routing exists if the hosts are in different pods. For the congestion and protection limit to choose the flow splitting %, Υ is considered as a measure of the % of bandwidth guaranteed to be available for the flow on a failed route. The utmost Υ is 0.5 if the flow is split evenly and it is zero if no flow is split. So, the congestion or path failure during VM migration is or routing flows between any VMs is avoided by splitting the flow on a certain path.

3.3 Merge-and-Split-based Coalitional Game-theoretic Scheme

For a data center of heterogeneous PMs and VMs, the network's operating condition information is forecasted by the cloud server's system software. After, each data is forwarded to the cloud server. Based on these data, the system can update the parameters, schedule the VMs and handle the PM's state like assigning the new PM or shutting down an idle PM [18]. The energy usage of a PM $\big(EC(u)\big)$ is assigned by its resource use u according to Eq. (13) where EC_{max} is the power used by an entirely-loaded PM & α is the fraction of inactive period of a PM.

$$EC(u) = \alpha EC_{max} + u(1-\alpha)EC_{max} \qquad (13)$$

Observe that the use of a CPU, memory, storage size and bandwidth are time-varying. Also, it depends on the workloads on it. So, $u(t)$ is used and the overall energy used is determined as:

$$\xi = \int_{t_0}^{t_0+T} EC\big(u(t)\big)dt \qquad (14)$$

In Eq. (14), t_0 is the initial time and T is the time during which a PM is active. It is considered that a cloud server has m categories of heterogeneous components, t_s refers to the interval that the VM consolidation initiates & t_e denotes the interval that VM consolidation terminates, f_k indicates the power used by the PM of category i per a given period that is in optimistic correlation with the traffic on it, called wl_k.

Consider b_k and a_k are the energy used by each machine of category k per unit interval before and after consolidation, respectively. Also, the energy used by VM migrations in a consolidation task is considered. This strategy is encouraged by the factor of migration-rate, (h). According to this considerations & settings, the dilemma is modeled as:

$$\max E_{conserve} = \int_{t_s}^{t_e} \sum_{k=1}^{m} (b_k - a_k) - h$$

$$\text{Subject to } \sum_{j=1}^{m} d_{ij} = 1, j = 1,2,\dots,u_j > 0 \qquad (15)$$

In Eq. (15), d_{ij} is a Boolean variable for deciding whether i^{th} VM is localized on j^{th} PM. If i^{th} VM is localized on j^{th} PM, then consider $d_{ij} = 1$; or else, $d_{ij} = 0$ and u_j denotes the use of PM_j, PM_j should not be an empty PM. Also, $E_{conserve}$ is the energy conserved by the VM consolidation strategy i.e., power preserved by the VM consolidation by means of the restrains that each VM is localized on single PM and there are no inactive PMs.

A coalitional game Υ has a group of players $N = \{1,2,\dots\}$ and a quality q which characterises the rate generated by various subgroups of the gamers i.e., the reward of a coalition C. At this point, increasing the reward $q(C)$ defines increasing the power-gains of C. Gamers of the competition select to link or not to link C through choosing whether high power-gains is realized. To

manage the CG over C of PMs, PMs are split as 3 sets: S, E and L which involve PMs with superfluous-, extremely- and lowly-loaded, accordingly, based on different workload thresholds as:

$$t_1 = Q_1 \text{ and } t_2 = Q_3 \quad (16)$$

In Eq. (36), $t_1 = Q_1$ indicates the primary quartile of the workloads localized on each PM and $t_2 = Q_3$ indicates the 3rd quartile of the traffic localized on each PM. In this strategy, the fuse-and-partition-based CG is conducted for increasing q of C i.e., reward as:

$$\max q \quad \text{where} \quad q = \frac{1}{n}\sum_{j=1}^{n} u_j$$

$$\text{Subject to } 0 < u_j \leq x_j, PM_j \notin S, PM_j \in C \quad (17)$$

The use of C is defined as q which equivalents the mean use of PMs in C excluding the PMs having superfluous traffic. In Eq. (37), u_j is the real-time use of PM_j, x_j is the highest use allowed of PM_j and n refers to the amount of PMs in C excluding the PMs having superfluous traffic. In CG, fuse function defines the merging many PMs into a one C whereas the partition function operates in the opposed way in which traffic from a superfluous-loaded PM is shared via many PMs.

The following Eqns. (18-a)-(18-d) indicate the prerequisite for merging a superfluous- & a lowly-loaded PM, the partition of a superfluous-loaded PM, the merging of lowly-loaded PMs and the merging of PMs with heavy-loaded, accordingly.

$$\forall PM_j \in S, PM_i \in L, C = \{PM_i, PM_j\}$$

$$q(C) > mean(u_j, u_i) \quad (18\text{-a})$$

$$\forall PM_j \in S, u_j < q(C)$$

$$C = \{PM_i, PM_k\}, PM_i, PM_k \in L/S \quad (18\text{-b})$$

$$\forall PM_j, PM_i \in L, C = \{PM_i, PM_j\}$$

$$q(C) > mean(u_j, u_i) \quad (18\text{-c})$$

$$\forall PM_j, PM_i \in E, C = \{PM_i, PM_j\}$$

$$q(C) > mean(u_j, u_i) \quad (18\text{-d})$$

Where u_i is the use of PM_i. Observe that the functions enabled by the Eqns. (18-a)-(18-d) prerequisites occur with the alphabetic manner of such prerequisites for guaranteeing that PMs having superfluous or lowly-loaded are controlled before those with the high workload. So, the objective of coalitional game is to create a PM set Y which includes PMs that are operating in a high-efficiency state to conserve more energy. Workload fairness is an essential metric for analyzing the VM consolidation method that denotes the network's resource use. Assume wf as a measure of workload fairness.

$$wf = {}^{(n_S + n_L)}/_{n_E} \quad (19)$$

In Eq. (19), n_S, n_L and n_E are the number of PMs in S, L and E, accordingly. So, a lower wf denotes better workload fairness. Therefore, the performance of VM migration is effectively improved considerably.

Algorithm for OH-BAC-FUP-MTR-MSCG:

Input: Set of PM and VM

Output: Suitable PM and VM in migration

Begin

$\boldsymbol{for}(each\ PM\ and\ VM)$

Estimate the usage of resources of both VM and PMs;

Perform LR & OPLR using past resource usages of VM and PM including regression coefficients;

Predict the relationship between the ongoing and upcoming usages of resources;

$\boldsymbol{for}(each\ PM)$

Initialize scout bee, employee bees and onlooker bees;

Compute σ by the scout bee using the average workload of PM;

Detect under and over-loaded hosts;

Scout bee dances in the knowledge base with osmosis technique to inform all swarms about the list of PMs (hosts);

Obtain the list of osmotic PMs;

$\boldsymbol{for}(List\ of\ osmotic\ PMs)$

Initialize pheromones;

Compute the fitness function using Eqns. (6) & (7);

Choose the suitable PM in migration;

Update the pheromones;

$\boldsymbol{end\ for}$

Inform all swarms about the suitable PM in migration by the knowledge base;

$\boldsymbol{end\ for}$

$\boldsymbol{for}(each\ VM)$

Execute the MMT policy by the employee bees;

Obtain the suitable VMs to be migrated;

$\boldsymbol{for}(Listofsuitable\ VMs\ in\ migration)$

Initialize pheromones;

Compute fitness function using Eq. (11);

Find the best mapping correlation between chosen VMs to be coordinated to the most suitable PM;

$\boldsymbol{end\ for}$

Inform all swarms about the most suitable VM to be migrated to the most suitable PM by the knowledge base;

Consider the mixture of traffic flows;

Estimate the bandwidth demands;

$\boldsymbol{if}(flow > bandwidth\ demands)$

Model $\mathcal{G} = (V, E)$;

$\boldsymbol{for}(each\ flow)$

Choose the PMs and construct the set of routes to send traffic among VMs;

Define a fitness factor and other restraints;

Compute the routes and flow ratio to be assigned;

$\boldsymbol{for}(each\ pair\ of\ communicating\ VM)$

Find the multiple link-disjoint minimum-congested routes;

Partition the flows at flow level;

Achieve the maximum Υ;

Assign the routes for each flow;

Prevent the bandwidth excess of controllers at top layers;

Prevent the congestion and link failure;

$\boldsymbol{end\ for}$

$\boldsymbol{end\ for}$

$\boldsymbol{end\ if}$

$\boldsymbol{end\ for}$

$\boldsymbol{for}(every\ PM_j\ in\ S)$

$\boldsymbol{for}(every\ PM_i\ in\ L)$

$\boldsymbol{if}(PM_i, PM_j\ are\ combined\ according\ to\ (18-a))$

Migrate the source VM from PM_j to PM_i by the onlooker bees;

$\boldsymbol{end\ if}$

$\boldsymbol{end\ for}$

$\boldsymbol{end\ for}$

$\boldsymbol{for}(every\ PM_j\ in\ S)$

$\boldsymbol{if}(PM_j\ is\ partitioned\ according\ to\ (18-b))$

Migrate the source VM from PM_j to a new PM by the onlooker bees;

$\boldsymbol{end\ if}$

end for

$\boldsymbol{for}(every\ PM_j, PM_i\ in\ L)$

$\boldsymbol{if}(PM_j, PM_i\ are\ combined\ according\ to\ (18-c))$

Migrate the source VM from PM_j to PM_i by the onlooker bees;

$\boldsymbol{if}(PM_j\ is\ empty)$

Shut down PM_j

end if

end if

end for

$\boldsymbol{for}(every\ PM_j, PM_i\ in\ E)$

$\boldsymbol{if}(PM_j, PM_i\ are\ combined\ according\ to\ (18-d))$

Migrate the source VM from PM_j to PM_i by the onlooker bees;

$\boldsymbol{if}(PM_j\ is\ empty)$

Shut down PM_j

end if

end if

end for

IV. SIMULATION RESULTS

This section simulates different methods such as OH-BAC-FUP, OH-BAC-FUP-MTR, and OH-BAC-FUP-MTR-MSCG in CloudSim API 3.0.3. Also, their efficiencies are compared with the OH-BAC method in the aspects of energy consumption, the number of VM migrations, the number of SLAV, SLATAH, PDM and the number of host's shutdowns. Table 1 shows the simulation environment and the parameters utilized in this analysis.

Table 1. Cloud Simulation Parameters

Type	Parameter	Value
Host	Number of hosts	100
	Types of hosts	HP ProLiant ML110 G4
		HP ProLiant ML110 G5
HP ProLiant ML110 G4	Number of Processing Elements (PEs) per host	4
	Bandwidth	3Gbps
	Host memory	8GB
	MIPS of PE	2060
HP ProLiant ML110 G5	Number of PEs per host	4
	Bandwidth	3Gbps
	Host memory	8GB
	MIPS of PE	3560
VM	Number of VMs	450
	Types of VMs	High-CPU Medium Instance
		Extra Large Instance
		Small Instance
		Micro Instance
High-CPU Medium Instance	MIPS of PE	2500
	Number of PEs per VM	5
	VM memory	1GB
	Bandwidth	118Mbps
Extra Large Instance	MIPS of PE	2000
	Number of PEs per VM	4
	VM memory	4GB
	Bandwidth	118Mbps
Small Instance	MIPS of PE	1000
	Number of PEs per VM	3
	VM memory	2GB
	Bandwidth	118Mbps
Micro Instance	MIPS of PE	500
	Number of PEs per VM	2
	VM memory	1.5GB
	Bandwidth	118Mbps
Cloudlets	Number of tasks	500
	Length of task (Million Instructions (MI))	2500*simulation limit
	Number of PEs per demand	2
OH-BAC-FUP	Number of iterations	100
	Number of ants	5
	Number of honeybees	15
	α	0.8
	β	0.32
	γ	0.8

	ρ	0.1
	c	0.8

4.1. Energy Consumption

It is the overall energy consumption by PMs at a given period. Figure 2 shows the energy consumption (in KWh) of different VM migration methods under a varying number of tasks. From this analysis, it is observed that the OH-BAC-FUP-MTR-MSCG minimizes energy consumption than all the methods during VM migration i.e., a mean energy consumption of OH-BAC-FUP-MTR-MSCG is 39.5% less than the OH-BAC, 35.7% less than the OH-BAC-FUP-LR, 31.3% less than the OH-BAC-FUP-OPLR and 28.3% less than the OH-BAC-FUP-MTR methods.

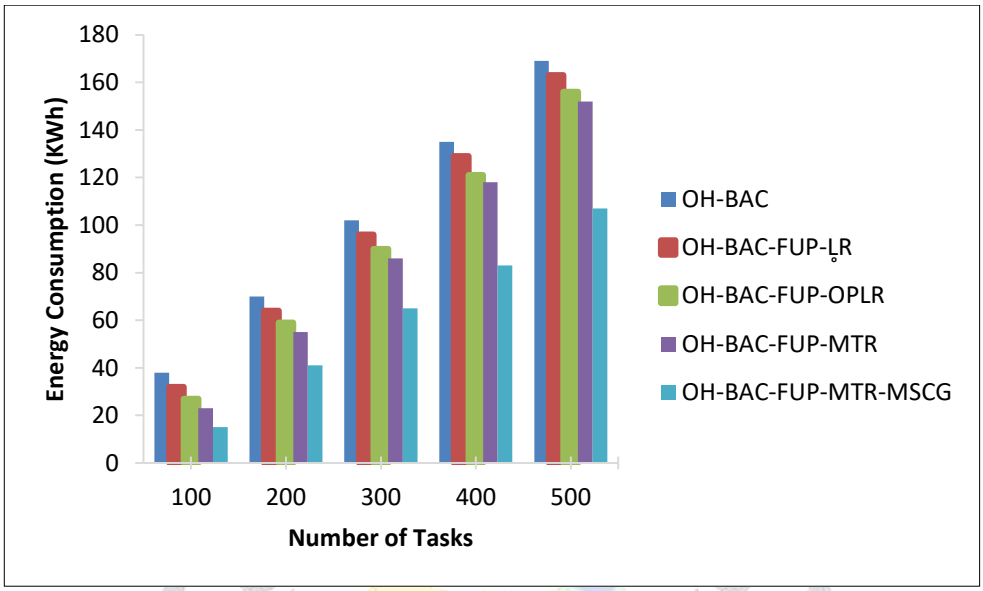

Figure 2: Energy Consumption vs. No. of Tasks

4.2. SLATAH

It refers to the proportion of time in which the operative host uses 100% CPU.

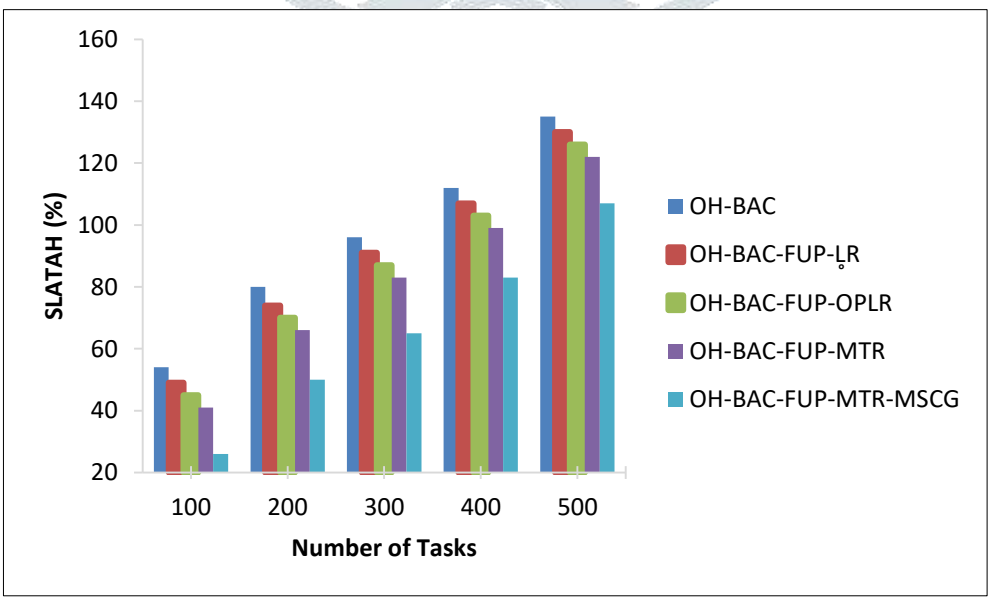

Figure 3:SLATAH vs. No. of Tasks

Figure 3 shows the SLATAH (in %) for different VM migration methods under the different number of tasks. This scrutiny notices that the OH-BAC-FUP-MTR-MSCG attains the least SLATAH than all other methods i.e., a mean SLATAH of OH-BAC-FUP-MTR-MSCG is 30.6% less than the OH-BAC, 26.6% less than the OH-BAC-FUP-LR, 23.2% less than the OH-BAC-FUP-OPLR and 19.5% less than the OH-BAC-FUP-MTR methods.

4.3. PDM

It is the overall performance degradation due to VM migrations.

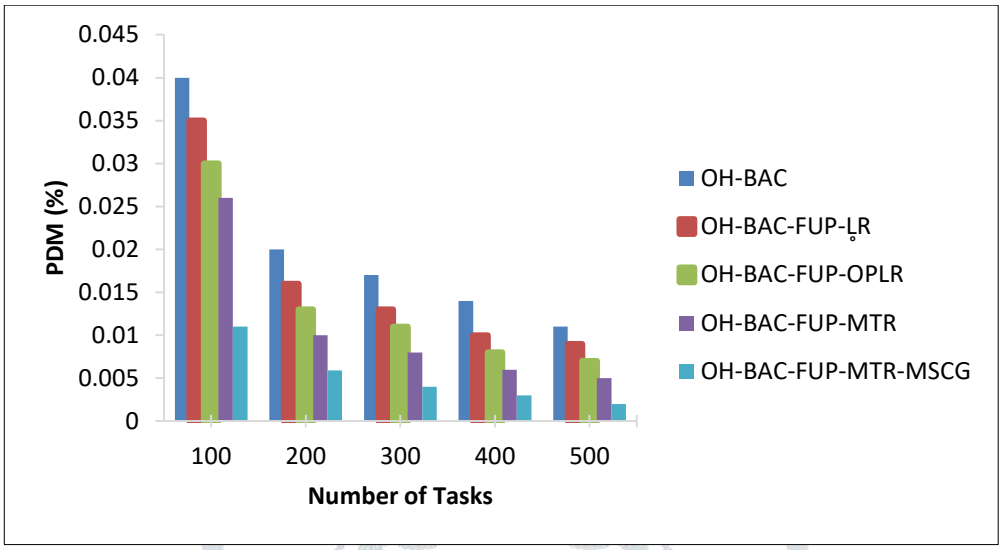

Figure 4:PDM vs. No. of Tasks

Figure 4 shows the PDM (in %) of different VM migration methods under a varying number of tasks. This analysis indicates that the OH-BAC-FUP-MTR-MSCG attains a minimum PDM and a maximum efficiency than all other methods i.e., a mean PDM of OH-BAC-FUP-MTR-MSCG is 74.6% less than the OH-BAC, 68.8% less than the OH-BAC-FUP-LR, 62.5% less than the OH-BAC-FUP-OPLR and 52.9% less than the OH-BAC-FUP-MTR methods.

4.4. SLAV

It is considered for evaluating the SLA delivered by a VM in the IaaS cloud.

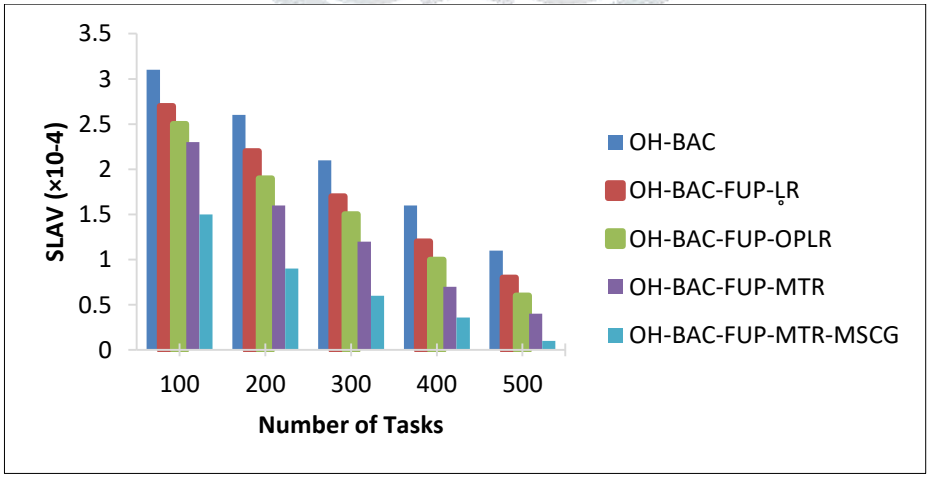

Figure 5: SLAV vs. No. of Tasks

Figure 5 shows the SLAV ($\times 10^4$ %) of different VM migrations under the different number of tasks. This scrutiny indicates that the OH-BAC-FUP-MTR-MSCG realizes the least SLAV than all other methods i.e., the mean SLAV of OH-BAC-FUP-MTR-MSCG is

67% less than the OH-BAC, 59.8% less than the OH-BAC-FUP-LR, 53.9% less than the OH-BAC-FUP-OPLR and 44.2% less than the OH-BAC-FUP-MTR methods.

4.5. Number of VM Migrations

It is the number of migrations created in the remapping phase.Figure 6 shows the number of VM migrations for different VM migration methods under the varying number of tasks. This scrutiny indicates that the OH-BAC-FUP-MTR-MSCG attains less number of VM migrations compared to all other methods i.e., a mean number of VM migration for OH-BAC-FUP-MTR-MSCG is 85% reduced than the OH-BAC, 80% reduced than the OH-BAC-FUP-LR, 76% reduced than the OH-BAC-FUP-OPLR and 70% reduced than the OH-BAC-FUP-MTR methods.

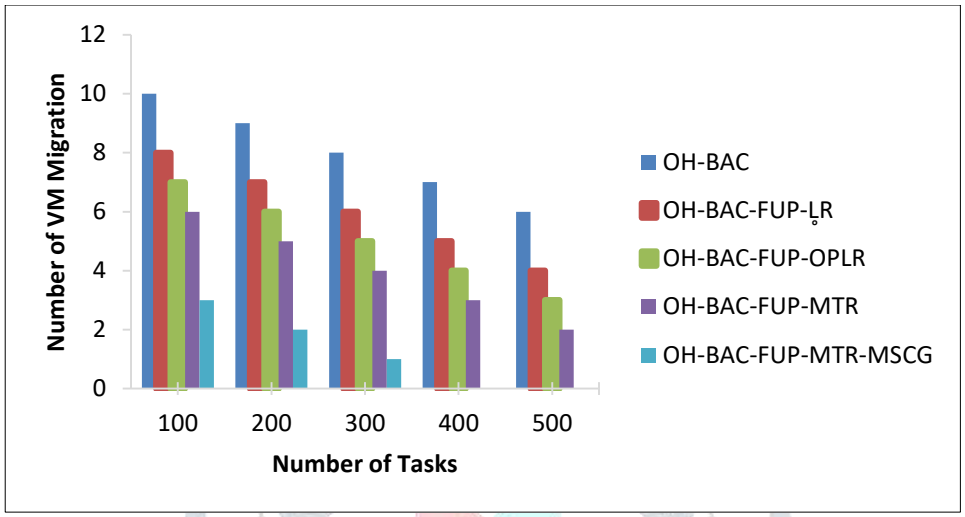

Figure 6: No. of VM Migrations vs. No. of Tasks

4.6. Number of Host's Shutdowns

It decides which hosts are operating, then shutdown. The host is shutdown after VM migration. If all VMs in a certain host are moved, then the host is shut down to reduce the energy use. But, the host is becoming operative when a VM has moved to it again.

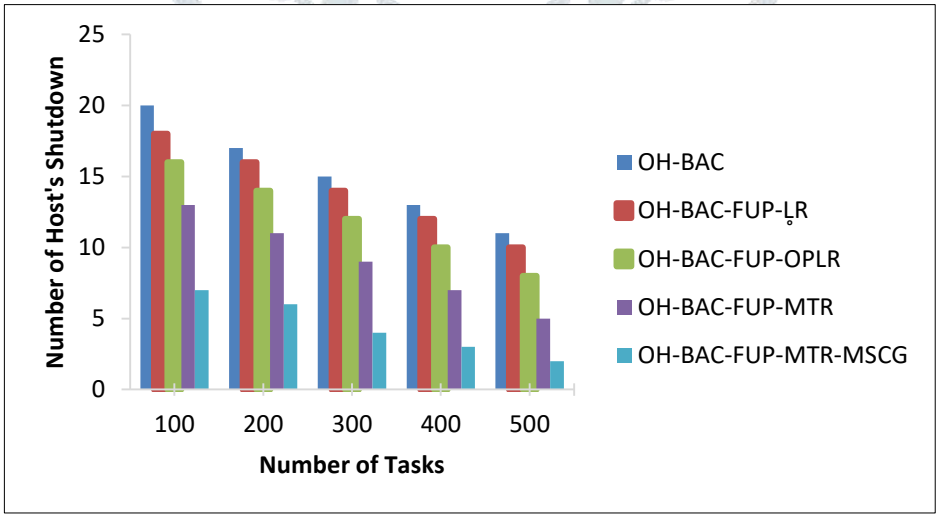

Figure 7. No. of Host's Shutdowns vs. No. of Tasks

Figure 7 shows the number of host's shutdowns for different VM migration methods under the different number of tasks. This scrutiny indicates that the OH-BAC-FUP-MTR-MSCG realizes the minimum number of active hosts than all other methods i.e., a mean number of host's shutdowns for OH-BAC-FUP-MTR-MSCG is 71.1% less than the OH-BAC, 68.6% less than the OH-BAC-FUP-LR, 63.3% less than the OH-BAC-FUP-OPLR and 51.1% less than the OH-BAC-FUP-MTR methods.

V. CONCLUSION

In this study, an OH-BAC-FUP approach using LR and OPLR schemes was initially developed to determine both ongoing and upcoming resource consumption of PMs and VMs. These estimated values are fed to OH-BAC for deciding the most apt VM to be migrated and the most apt PMs. Then, MTR scheme is introduced during VM migration to protect the chance of congestion. If any congestion exists because of high bandwidth or traffic flows, then the flows are split into many segments and transfer them via multiple link-disjoint routes. Also, MSCG scheme is applied for splitting the PMs into many sets and choosing the members from those sets to generate effective coalitions. For each coalition, OH-BAC-FUP-MTR is executed which enhances the payoff of each coalition and sustains PMs to operate in a high energy-efficient state. To conclude, the findings proved that the OH-BAC-FUP-MTR-MSCG has a higher performance compared to all other approaches in terms of energy consumption, SLATAH, SLAV, PDM, number of VM migrations and host shutdowns.

REFERENCES

[1] De Donno, M., Tange, K., & Dragoni, N. (2019). Foundations and evolution of modern computing paradigms: cloud, iot, edge, and fog. *IEEE Access*, *7*, 150936-150948.

[2] Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing–a hierarchical taxonomical classification. *Journal of Cloud Computing*, *8*(1), 1-24.

[3] Tripathi, A., & Singh, S. (2017). A literature review on algorithms for the load balancing in cloud computing environments and their future trends. *Mathematical and Computer Modelling*, *21*(1), 64-73.

[4] Mohammadian, V., Navimipour, N. J., Hosseinzadeh, M., & Darwesh, A. (2020). Comprehensive and systematic study on the fault tolerance architectures in cloud computing. *Journal of Circuits, Systems and Computers*, *29*(15), 1-42.

[5] Villari, M., Fazio, M., Dustdar, S., Rana, O., & Ranjan, R. (2016). Osmotic computing: a new paradigm for edge/cloud integration. *IEEE Cloud Computing*, *3*(6), 76-83.

[6] Villari, M., Celesti, A., & Fazio, M. (2017). Towards osmotic computing: Looking at basic principles and technologies. In *Conference on Complex, Intelligent, and Software Intensive Systems*, Springer, Cham, pp. 906-915.

[7] Gamal, M., Rizk, R., Mahdi, H., & Elnaghi, B. E. (2019). Osmotic bio-inspired load balancing algorithm in cloud computing. *IEEE Access*, *7*, 42735-42744.

[8] Cziva, R., Jouët, S., Stapleton, D., Tso, F. P., & Pezaros, D. P. (2016). SDN-based virtual machine management for cloud data centers. *IEEE Transactions on Network and Service Management*, *13*(2), 212-225.

[9] Kumar, M., & Sharma, S. C. (2018). Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. *Computers & Electrical Engineering*, *69*, 395-411.

[10] Sekaran, K., Khan, M. S., Patan, R., Gandomi, A. H., Krishna, P. V., & Kallam, S. (2019). Improving the response time of m-learning and cloud computing environments using a dominant firefly approach. *IEEE Access*, *7*, 30203-30212.

[11] Hejja, K., & Hesselbach, X. (2019). Evaluating impacts of traffic migration and virtual network functions consolidation on power aware resource allocation algorithms. *Future Generation Computer Systems*, *101*, 83-98.

[12] Hsieh, S. Y., Liu, C. S., Buyya, R., & Zomaya, A. Y. (2020). Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. *Journal of Parallel and Distributed Computing*, *139*, 99-109.

[13] Afzal, S., & Kavitha, G. (2020). A hybrid multiple parallel queuing model to enhance QoS in cloud computing. *International Journal of Grid and High Performance Computing*, *12*(1), 18-34.

[14] Gholipour, N., Arianyan, E., & Buyya, R. (2020). A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers. *Simulation Modelling Practice and Theory*, *104*, 1-17.

[15] Yun, H. Y., Jin, S. H., & Kim, K. S. (2021). Workload stability-aware virtual machine consolidation using adaptive harmony search in cloud datacenters. *Applied Sciences*, *11*(2), 1-23.

[16] R.Gowriprakash, Shankar. R & Duraisamy.S. (2020). FUPA: future utilization prediction algorithm based load balancing scheme for optimal VM migration in cloud computing. In *IEEE Fourth International Conference on Inventive Systems and Control*, pp. 638-644.

[17] R.Gowriprakash, Shankar.R & Duraisamy, S. Resource utilization prediction with multipath traffic routing for congestion-aware VM migration in cloud computing. Year: 2021, Volume: 14, Issue: 7 ,Indian journals of science and technology.

[18] R.Gowriprakash, Shankar.R & Duraisamy.S The impact on load balancing in cloud computing International journals of scientific & technology research Volume9,Issuse 01,January 2020