INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Resource utilization prediction with multipath traffic routing for congestion-aware VM migration in cloud computing

R GowriPrakash[1]*, R Shankar[2], S Duraisamy[2]

1 Ph.D, Research Scholar, Department of Computer Science, Chikkanna Government Arts College, Tirupur, Tamilnadu, India. Tel.: +918148188030
2 Assistant Professor, Department of Computer Science, Chikkanna Government Arts College, Tirupur, Tamilnadu, India

OPEN ACCESS

Received: 19.10.2020
Accepted: 23.02.2021
Published: 03.03.2021

Citation: GowriPrakash R, Shankar R, Duraisamy S (2021) Resource utilization prediction with multipath traffic routing for congestion-aware VM migration in cloud computing. Indian Journal of Science and Technology 14(7): 636-651. https://doi.org/10.17485/IJST/v14i7.1901

* **Corresponding author**.

Tel: +918148188030
prakashcs003@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Background: The Load Balancing (LB) schemes in cloud computing consider both current and future utilization of resources to decide the most suitable Virtual Machines (VMs) to be migrated to the most appropriate Physical Machines (PMs). But, the possibility of network congestion occurrence was high when increasing the bandwidth use between VMs within the cloud data centers. Also, a less-than-optimal migration of VMs can lead to high network traffic since it causes inter-VM traffic for traversing the bottleneck network routes. **Objective:** To enhance the efficiency of LB and reduce the possibility of congestion occurrence during VM migration in cloud data centers. **Methods**: Osmotic Hybrid artificial Bee and Ant Colony with Future Utilization Prediction with Multipath Traffic Routing (OH-BAC-FUP-MTR) mechanism are presented in this article to achieve the above objective. Initially, the OH-BAC-FUP mechanism is performed to decide the most suitable VMs to be migrated to the most suitable PMs. During VM migration, if any congestion exists due to high bandwidth use or traffic flows, then the MTR algorithm is applied to partition the flows and route them through multiple link-disjoint routes. Based on this, the congestion is avoided while ensuring the bandwidth and security grade demands. Also, the highest traffic on the path is applied as a congestion factor. Moreover, the current and future network states are taken into account for MTR to select the most optimal route from multiple routes with no consideration of the past use of the paths. **Findings**: The simulation outcomes demonstrate the OH-BAC-FUP-MTR mechanism consumes 8.74% of overall energy, 6.8% of Service Level Agreement violation Time per Active Host (SLATAH), 27.63% of Performance Degradation due to Migration (PDM), 22.98% of SLA Violation (SLAV), 27.27% of VM migrations and 30.77% of hosts shutdown compared to the OH-BAC-FUP using Linear Regression (LR) and Optimal Piecewise LR (OPLR).

Keywords: Cloud computing; load balancing; VM migration; OH-BAC-FUP; multipath traffic routing

1 Introduction

In general, the cloud computing paradigm possesses numerous problems as its usage is increasing exponentially. The most significant problem is the balancing of loads between resources [1,2]. To solve this problem, LB mechanisms are used which improves the overall system efficiency, robustness, availability and other characteristics in the cloud data centers. The objective of LB is to handle the unbalance of the workload between the datacenter to prevent overload and under overload situations. The LB mechanisms are enhanced by providing the Service Level Agreement (SLA) and customer satisfaction [3]. The development of powerful LB systems and applications is indeed a key element of cloud-based systems. For this reason, many algorithms have been developed in the field of LB with task scheduling processes in cloud services [4–8].

In recent years, overall development is looking into an innovative concept in osmotic computing followed by the substance osmotic characteristic concept. It is primarily applied for achieving the balanced utilization of resources in highly distributed systems.

In cloud-based services, it is introduced to make use of balanced VMs that are migrated in the cloud devices [9,10]. Among different LB algorithms, many optimization algorithms achieve better performance; however, most of them cannot have the ability to achieve better performance in all aspects. To tackle this issue, Gamal et al. [11] proposed an OH-BAC mechanism to minimize the power usage, the number of VM migrations and the number of shutdown hosts. Conversely, only the amount of active PMs was minimized depending on their present resource demands whereas the future resource demands were omitted. So, the undesired VM migrations were created and the percentage of SLA Violations (SLAV) was improved in the cloud storage.

As a result, OH-BAC-FUP was developed for minimizing the number of VM migrations and improving the LB efficiency. In this mechanism [12], the upcoming use of resources was also taken into account with the current use for transferring the VMs to the minimal number of operative PMs. For predicting the future use of resources such as CPU, bandwidth, storage size and memory of both VMs and PMs, LR and OPLR-based prediction algorithms were applied. After obtaining the predicted values, these were employed in the OH-BAC's fitness factor for selecting the suitable VM to migrate to the highly appropriate PM. But, the probability of the occurrence of network congestion was high while increasing the bandwidth utilization between VMs within the data center. Less than optimal positioning of VMs can lead inter-VM communication to connect bottleneck network paths which results in huge cross-network congestion. Primary node overprovisioning and unstable assignments in cloudlets can also contribute to long-lasting traffic. If there were a large number of congestions, the resource utilization performance may degrade significantly.

Many studies have considered VM migration with its impact on network traffic; however, the primary objective of minimizing the overall energy consumption in a data center. As data centers transmit a massive amount of traffic, system failures have severe impacts [13]. If resources are reserved with adequate backup bandwidth resources, then, 100% traffic prevention can be achieved. Due to the expense of reserving the backup resources, traffic can be prevented partially wherein traffic will obtain less bandwidth in the system failures. The partial prevention can guarantee service accessibility, but with less bandwidth and efficiency which could be acceptable for many applications. From this perspective, a prevention grade is considered as a measure of partial prevention in this paper for reducing traffic congestion in data centers.

Therefore, this study introduces an OH-BAC-FUP-MTR mechanism to achieve effective LB and reducing the probability of congestion occurrence in cloudlets. In this mechanism, the traffic is routed on multiple link-disjoint routes for preventing the failures and ensuring the availability of a minimum of one route for the traffic upon a link failure or congestion. At first, the OH-BAC-FUP is executed for deciding the most appropriate VMs to be migrated to the most suitable PMs. If any congestion occurs during migrating VMs to the PMs, then the MTR algorithm is performed that partitions flows or traffic into 2 and forwards them via multiple link-disjoint routes so the traffic is reduced when guaranteeing the bandwidth and security grade demands. Also, the highest traffic on the path is used as a congestion factor. Moreover, the current and future network states are accounting for MTR to select the most optimal route from multiple routes with no consideration of the past use of the paths. As a result, this OH-BAC-FUP-MTR mechanism is more suitable for intelligently migrating VMs onto the PMs with reduced traffic load on the network.

The rest of the article is prepared as follows: Section 2 studies the researches related to the VM migration in cloud computing. Section 3 describes the functioning of OH-BAC-FUP-MTR and Section 4 portrays its performance. Section 5 summarizes this research work and suggests future scope.

2 Literature Survey

Vu and Hwang [14] proposed a flow and energy-aware method for VM localization in the cloud storage systems intending to reduce the communication cost and energy consumption. In this method, a novel algorithm was proposed for finding the target PM to host selected VMs. If VMs were consolidated on PMs with higher CPU use per energy consumption, the overall energy

consumption was reduced. Also, if two VMs running network-aware applications were hosted on a PM, the traffic cost for that connection was reduced. However, the number of VM migrations was very high. Reguri et al.[15] proposed a power-efficient flow-aware VM migration based on dynamic VM migration techniques. It was performed via clustering the VMs depending on the transmission or flow. But, the SLAV created by over-loaded hosts was not reduced and also it does not consider the memory factor in the VM migration model.

Deshpande & Keahey[16] proposed a flow-sensitive real-time VM migration algorithm by using a mixture of pre-copy and post-copy methods for transferring the co-situated VMs. The choice of migration algorithms was depending on VM's network traffic profiles. The network traffic on each host was monitored for generating the per-VM system flow summary. After that, each combination of pre-copy and post-copy was weighed and selected for all the VMs. But, the issue of migrating VMs from an identical origin host to an identical target host was not addressed.

Liu et al.[17] proposed an accurate Failure Detection based on Weibull distribution called WD-FD for preventing any congestion or failures in the cloud network. In WD-FD, a sliding window was considered for maintaining more fresh data to estimate the Weibull distribution factors. By using these data, the suspicion level was computed for matching the recent network condition. But, it needs to further reduce the mistake rate for improving the detection accuracy. Cui et al.[18] proposed a novel method for migrating VM via dynamically creating adaptive topologies depending on VM requirements. First, a flow-aware VM migration dilemma was formulated. After that, a new progressive-decompose-rounding algorithm was proposed for periodic traffic to plan VM migration in a polynomial period with a verified estimate rate. Also, an Online Decision-Maker (ODM) strategy was proposed with a verified efficiency limits for highly dynamic flows. However, the computational complexity of this method was high and it does not improve resource utilization.

Fu et al.[19] proposed a layered VM migration scheme for reducing resource use and network congestion. Initially, the cloud data center was split into many regions using the inter- and intra-area VM migration algorithm. These were performed based on the ratio of bandwidth used by the hosts. After that, the resources of all areas were balanced by VM migrations. But, the number of VM migrations was not reduced efficiently and the cost of this algorithm was high since the VMs were migrated several times in intra-regions.

Vakilina[20] proposed a hybrid optimization of energy use of system, transmission and migration costs including traffic and system heterogeneity concerning the resource and bandwidth limits. The optimization problem was devised as an Integer Quadratic Program (IQP) using quadratic limits for small-scale systems. Also, this can be developed as an Integer Linear Programming (ILP) resolved by the column creation in large-scale systems. Also, network bandwidth constraints were considered for preventing traffic congestion. However, the computational complexity and runtime were high (Afzal & Kavitha)[21]. This algorithm IMDLB takes consideration of the proficiency-related entity Vms and complicity of each requested task to assign relevant virtual machines. This work consider as migration cost of quality of service Qos parameter and validates its efficiency by providing the most minimal optimized solution in term of migration cost.

Hejja & Hesselbach[22] presented a novel energy-aware resource allocation method that supports PM consolidations integrated with the VMs consolidation for reducing the data centers overall costs for an offline scenario. Also, this method was combined with an elective standalone traffic migration scheme which was triggered based on certain criteria. But, its efficiency was less since it does not consider LB and traffic predictions. Hsieh et al.[23] suggested the VM consolidation method which considers the current and future use of resources by the host overload and host under-load identification. The future use of resources was precisely identified by the gray-Markov-based scheme. But, it does not consider the traffic flow during migration to solve any conflicts. Afzal & Kavitha[24] developed a hybrid multiple parallel queuing methods for improving the Quality-of-Service (QoS) in cloud computing. But, it assumes the traffic arrival rate and service rate were constant.

3 Proposed Methodology

This part explains the OH-BAC-FUP-MTR mechanism in detail. Initially, the OH-BAC-FUP is performed to choose the highly appropriate VMs to be transferred to the highly appropriate PMs based on the current and future resource utilization demands. During migration, the MTR algorithm is applied if any congestion occurs due to high traffic. The overview processing and flow diagram of the OH-BAC-FUP-MTR mechanism are portrayed in Figures 1 and 2, respectively.

In Figure 1, the VM migration through switches is shown which reduces the probability of congestion in the network. The use of switches can regulate the traffic on a link during VM migration. If VM_1 in PM_1 needs to transmit a flow to VM_2 in PM_5, the possible route between PM_1 and PM_5 is $S_1 - S_2 - S_3 - S_4$ with different traffic loads. It may split the traffic loads to avoid congestion based on the following processes:

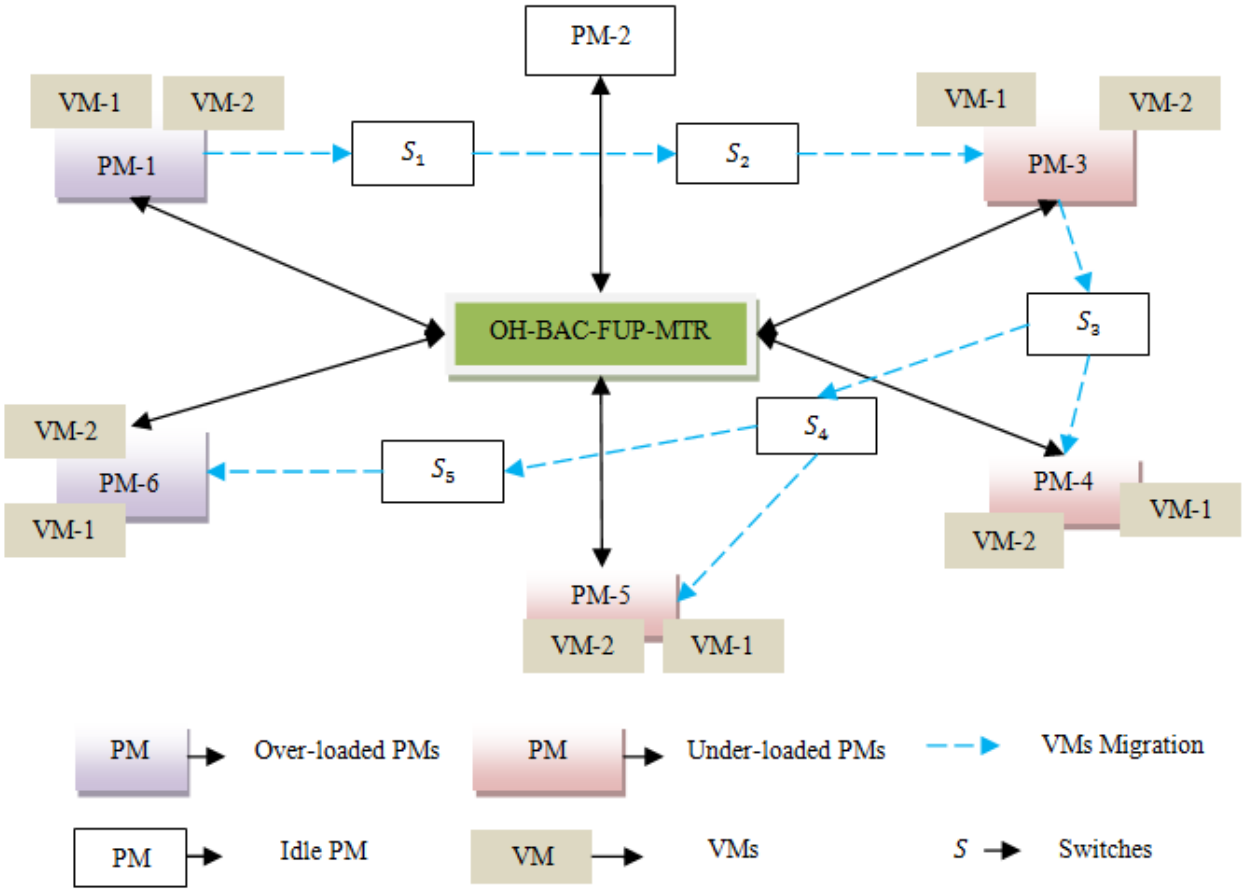

Fig 1. The overall process of OH-BAC-FUP-MTR mechanism

3.1 Problem formation

Let a set of x PMs $P = (p_1, p_2, \ldots, p_x\}$ and a set of y VMs $V = (v_1, v_2, \ldots, v_y\}$. A PM host p_i has a specific amount of resources represented as $p_i = \{p_{c_i}, p_{m_i}, p_{b_i}, p_{ss_i}\}$ where p_{c_i} is the CPU usage, p_{m_i} is the capacity of memory, p_{b_i} is the bandwidth and p_{ss_i} is the capacity of storage size. A VM flow is initiated at a VM and terminated at another VM. A task's resource demand is specified as a vector $J = (r_1, r_2, \ldots, r_y\}$ where r_k is the amount of VMs of type v_k.

The resource necessity of v_i is denoted as $v_i = \{v_{c_i}, v_{m_i}, v_{b_i}, v_{ss_i}\}$ where v_{c_i} is the necessary CPU usage, v_{m_i} is the necessary memory, v_{b_i} is the necessary bandwidth and v_{ss_i} is the necessary storage size. Consider that there is a mixture of traffic flows in a data center requiring a different amount of bandwidth. So, it is essential to consider different types of VMs. Though it is complex to compute an accurate bandwidth requirement of traffic between VMs, consider that occupiers encompass a better estimation of bandwidth demands according to the types of purposes and their efficiency demands.

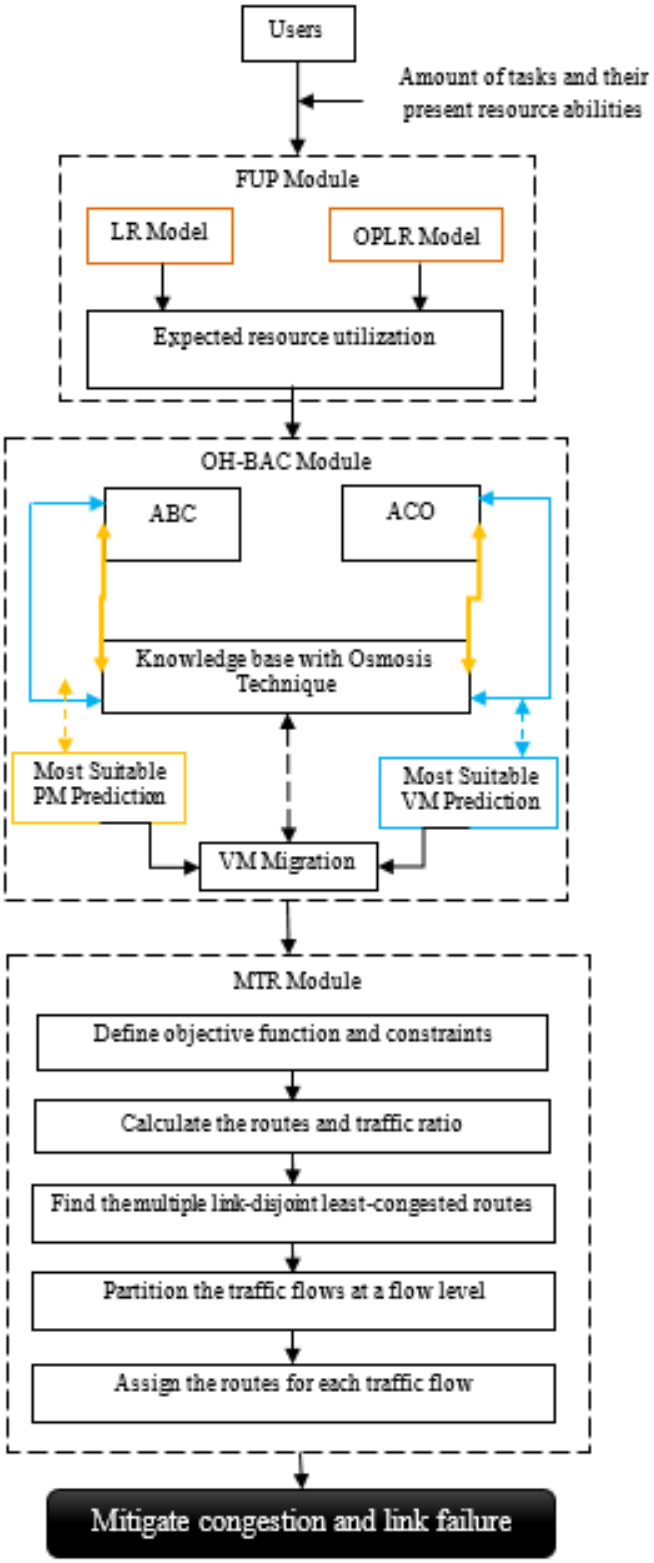

Fig 2. Flow diagram of OH-BAC-FUP-MTR-based LB in cloud computing

In this mechanism, the difficulty is to discover a mapping between VMs and PMs that fulfills the VM's resource necessities when aiming to reduce the overcrowding and offering prevention assurance of rank Υ, i.e., the ratio of bandwidth assured to be accessible for traffic upon a path collapse.

The highest traffic on a path is used as a congestion factor. A cloud storage system is modeled by a directed graph $G = (V, E)$ wherein $v \in V$ is a system controller, PM or the outside user and $(u, v) \in E$ is a substantial path connecting the controllers or/and the PMs. Every path (u, v) contain a nonnegative ability $ct\,(u, v) \geq 0$. Each path transmits single or multiple traffics. On $(u, v) \in E$, bandwidth $b_{i,u,v}$ is reserved for flow i. For each task, the PMs are selected for supporting its substantial resource demands and a group of routes in G to forward flows between VMs.

Every route has a group of connections that intersect controllers and PMs. In this mechanism, MTR is considered in which the flow is mapped to multiple routes having a specific flow rate to reduce the overcrowding when offering a certain Υ. When traffic requirement (i) including the bandwidth demand (b) are divided and transmitted across n link-disjoint routes at the ratio of b_{max} (maximum bandwidth), the least Υ is given by $(b - b_{max}] / (b]$ that occurs if a connection on the route having b_{max} is unsuccessful.

The goal is to reduce the highest flow on a path when offering Υ as a minimum of Γ for every requirement. The fitness factor and the restraints associated with VM migration and MTR are as follows:

$$min\,(L_{max}) \tag{1}$$

Restraints:

- The maximum load L_{max} is defined by $L_{max} \geq g_{u,v},\ \forall u, v \in V$.
- The overall bandwidth or traffic used on (u, v) is computed as the total of traffic i passing through the path.

$$g_{u,v} = \sum_i b_{i,u,v},\ \forall u, v \in V \tag{2}$$

- Ability restraints: The overall resource demand of the VMs situated on the host must not surpass its ability. For PM j, the total of the resource demands of each VM situated on it must be lower than or similar to the host's overall accessible ability.

$$\sum_i v_{c_i} * p_{ij} \leq p_{c_j},\ \forall j \tag{3-a}$$

$$\sum_i v_{m_i} * p_{ij} \leq p_{m_i},\ \forall j \tag{3-b}$$

$$\sum_i v_{b_i} * p_{ij} \leq p_{b_i},\ \forall j \tag{3-c}$$

$$\sum_i v_{ss_i} * p_{ij} \leq p_{ss_i},\ \forall j \tag{3-d}$$

- Migration restraint: Each VM is migrated to one PM and all VMs are migrated.

$$\sum_j p_{ij} = 1,\ \forall i \tag{4}$$

- Bandwidth restraint: The total bandwidth utilized on (u, v) must not surpass its $ct\,(u, v)$.

$$\sum_i b_{i,u,v} \leq ct\,(u, v),\ \forall u, v \in V \tag{5}$$

- No failure restraint: For each controller u which is not the origin controller s_i or the target controller t_i of i, the received bandwidth should be similar to the transmitted bandwidth.

$$\sum_{v \in V} b_{i,v,u} - \sum_{w \in V} b_{i,u,w} = 0, \ \forall i \ and \ \forall u \in V - \{s_i, t_i\} \tag{6}$$

- Traffic partitioning restraint: Traffic i is partitioned into n having bandwidth b to be migrated onto multiple routes.

$$b_{max} = b \tag{7}$$

- Traffic routing on link-disjoint routes: (u, v) is passed through by no one or only one of the n routes of i. Also, it guarantees that n routes utilize disjoint groups of paths.

$$x_{i,1,u,v} + \cdots + x_{i,n,u,v} \leq 1, \ \forall i \ \& \ \forall u, v \in V \tag{8}$$

$$b_{i,v,u} = x_{i,1,u,v} * b_{max} + \cdots + x_{i,n,u,v} * b_{max} \leq b, \ \forall i \ \& \ \forall u, v \in V \tag{9}$$

- Traffic partitioning at the origin: The overall flow ratio of s_i is similar to b.

$$\sum_{v \in V} b_{i,s_i,v} = b, \ \forall flow \ i \tag{10}$$

- Traffic combining at the target controller: The overall flow ratio of t_i is similar to b.

$$\sum_{v \in V} b_{i,v,t_i} = b, \ \forall flow \ i \tag{11}$$

- Prevention restraint: The rate of bandwidth accessible for i upon a path collapse is not less than Υ.

$$\frac{b_{max}}{b} \geq \Upsilon, \ \forall requirement \ i \tag{12}$$

3.2 Multipath traffic routing

In this step, routes and their flow rates are computed. The accessibility of multiple routes between origin VM s_j and target VM t_j is leveraged to share the flow across the routes and partly prevent the flow from a particular path collapse. For all couples of interacting VMs, multiple link-disjoint minimum-congested routes are determined. The minimum-congested route is the route having the minimum traffic. The route traffic is referred to as the highest traffic on a connection passed through the route. The traffic is partitioned only at the edge and it pursues the pre-assigned routes with no necessity of additional partitioning.

Design of Routing and Route Load

Two different cases of routing are considered for selecting routes for a flow such as intra- and inter-pod routing. Intra-pod routing occurs while a couple of interacting hosts is inside a similar pod whereas inter-pod routing occurs while the hosts are in dissimilar pods.

For instance, assume the scenario of intra-pod routing as illustrated in Figure 3 in which the components which are not involved in the transmission route are neglected and the connections are noticeable including the present flow rate. If VM_1 requests to transmit traffic to VM_2, both are inside a similar pod. Figure 3(a) depicts 2 promising link-disjoint routes between PM_1 and PM_2. When the route S_1-S_2-S_4 contains a flow rate of 12, the route S_1-S_3-S_4 contains a flow rate of 17.

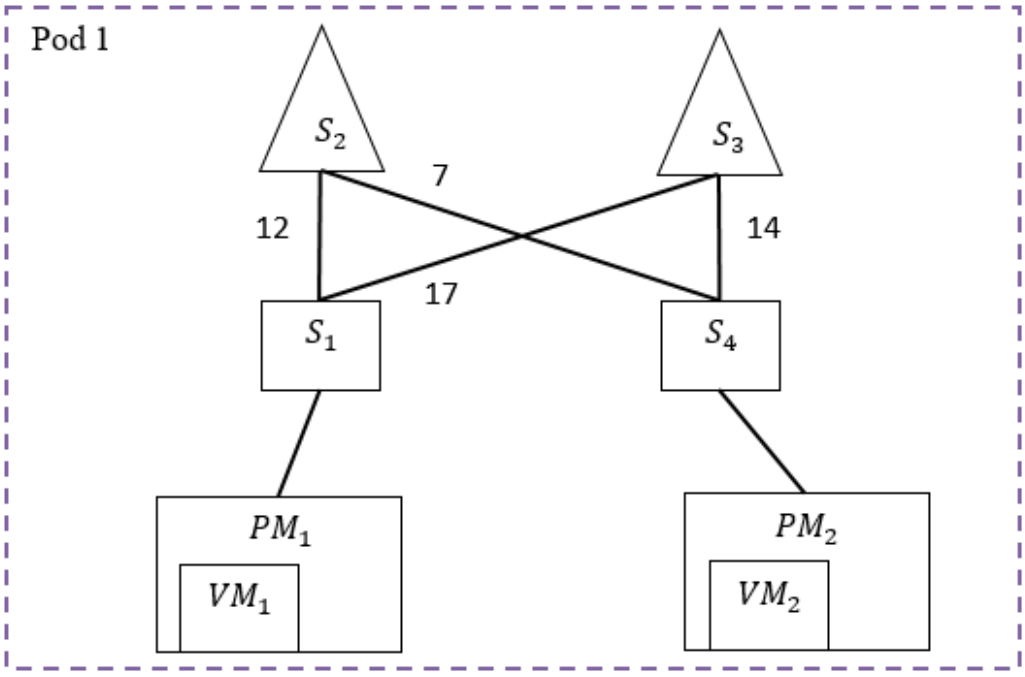

Fig 3. Example for intra-pod routing scenario

Inter-pod routing explores routes between dissimilar pods. For instance, assume a scenario of inter-pod routing as illustrated in Figure 3(b). In pod 1, VM_1 is included which requests to transmit traffic to VM_2 in pod 4. In this case, 2 promising routes are available between PM_1 and PM_2: S_1-S_2-S_4-S_8-S_{10} and S_1-S_3-S_6-S_9-S_{10} containing the flow rate of 12 and 17, accordingly.

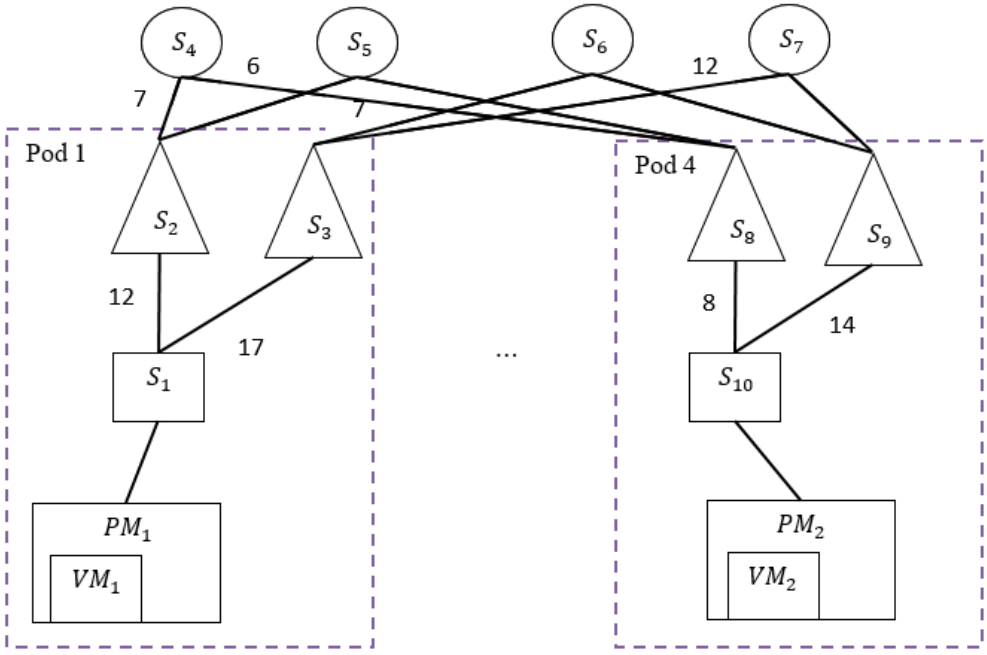

Fig 3(b). Example for inter-pod routing scenario

Traffic portioning policy

Assume the congestion and prevention constraint for selecting the traffic partitioning rate. Also, γ is used as a measure of the ratio of bandwidth assured to be accessible for the traffic on a path collapse. The highest γ is 0.5, while the flow is partitioned uniformly. If there is no traffic/flow partitioning, then γ is 0. Through this, it is observed that reducing the overcrowding is not essential leading to the highest γ.

The effect of partitioning on congestion and γ is shown in Figure 4. The routes R_1, R_2 and R_3 having different traffics are illustrated in Figure 4(a). If traffic having a flow ratio of 200 units requests to be transmitted. Assume 3 scenarios of partitioning such as 50:50, 60:40 and 55:45. While the flow is partitioned in the fraction of 50:50, the congestion is 120 with γ of 0.5 as illustrated in Figure 4(b).

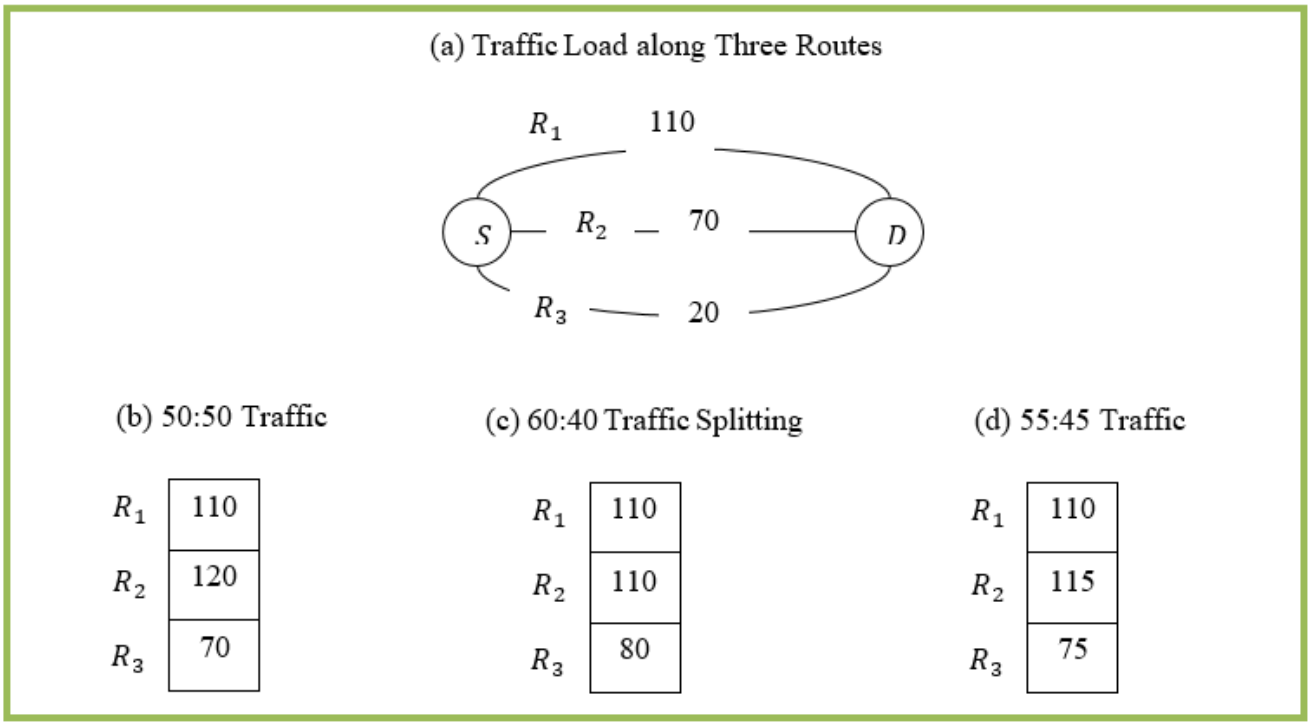

Fig 4. Illustration of traffic partitioning and prevention grade

While the flow is partitioned in the fraction of 60:40, the congestion is 110 with γ of 0.4 as illustrated in Figure 4(c). While the flow is partitioned in the fraction of 55:45, the congestion is 115 with γ of 0.45 as illustrated in Figure 4(d). If the aim is to maximize γ without reporting congestion, then the flow is partitioned into 50:50.

If the goal is to reduce the congestion without reporting γ, then the flow is partitioned into 60:40. If the aim is to reduce the congestion related to a specific constraint, e.g., at least 45% γ, then the flow is partitioned into 55:45. If the minimum γ required is 30%. Partitioning the flow 30:70 results in traffic of 140 and 50 on P_2 and P_3, accordingly with the congestion being 140. Here, partitioning flow 40:60 as in scenario 2 will result in congestion of 110 only. Initiate with partitioning the flow in the fraction $\gamma : (200 - \gamma)$ and the resulting congestion is computed. After that, the congestion for various ranges of γ is computed until $\gamma = 50$ and the partitioning ratio has been selected that results in less congestion.

Algorithm for OH-BAC-FUP-MTR:

Begin

for (*each VM and PM*)

Compute the resource usages;

Consider the mixture of traffic flows;

Estimate the bandwidth demands;

if (*congestion occurs*)

```
Model G = (V, E);
for (each traffic 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;
  for (each pair of communicating VM)
    Find the multiple link-disjoint minimum-congested routes;
    Partition the traffic 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;
  end for
 end for
end if
end for
```

The computational complexity of OH-BAC-FUP-MTR is $O\left(x^2\right) + O\left(w^2\right)$ where x and w are the number of PMs and switches (S) in the system.

4 Simulation Results

This section simulates the OH-BAC-FUP-MTR mechanism and compares its efficiency with the OH-BAC-FUP by LR and OPLR mechanisms. The analysis is carried out based on energy consumption, the number of VM migrations and the number of SLAV, SLATAH, PDM and the number of host's shutdowns. In this experiment, both mechanisms are implemented by using CloudSim API 3.0.3. Table 1 shows the simulation environment and the parameters used in OH-BAC-FUP.

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

Continued on next page

Table 1 continued

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 the 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
		0.8
		0.32
		0.8
		0.1
		0.8

4.1 Energy consumption

It is the overall energy consumption by PMs at a given period.

$$E = \int_t \left(k * P_{full} + (1-k) * P_{full} * u_i\right) \tag{13}$$

In Eq. (13), k is the % of idle PM's energy consumption, P_{full} is the energy consumption of PM at full load and u_i is the CPU usage of the PM.

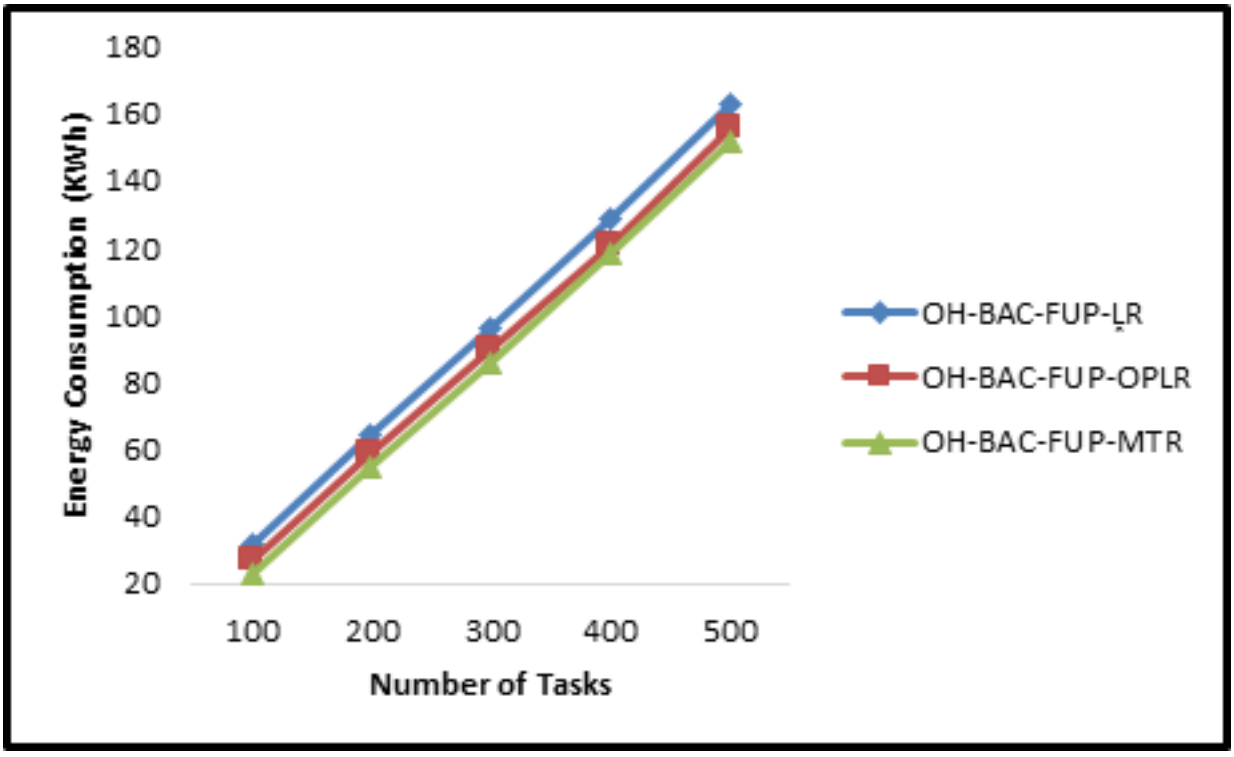

Fig 5. Energy consumption vs. No. of tasks

Figure 5 shows the energy consumption (in KWh) of OH-BAC-FUP-MTR and OH-BAC-FUP mechanisms under a varying number of tasks. From this analysis, it is observed that the OH-BAC-FUP-MTR minimizes energy consumption than the OH-BAC-FUP during VM migration. For example, energy consumption by PMs during 500 tasks for OH-BAC-FUP-MTR is 6.75% less than the OH-BAC-FUP-LR and 2.56% less than the OH-BAC-FUP-OPLR mechanisms.

4.2 SLATAH

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

$$SLATAH = \frac{1}{n}\sum_{j=1}^{n}\frac{T_{s_j}}{T_{as_j}} \tag{14}$$

In Eq. (14), n denotes the number of PMs, T_{s_j} is the period in which j^{th} PM uses 100% CPU and T_{as_j} is the total number of j^{th} PM that is in the active state.

Figure 6 shows the SLATAH (in %) for OH-BAC-FUP-MTR and OH-BAC-FUP mechanisms under the different number of tasks. This scrutiny notices that the OH-BAC-FUP-MTR attains the least SLATAH than the OH-BAC-FUP. For example, the SLATAH of OH-BAC-FUP-MTR for 500 tasks is 6.15% less than the OH-BAC-FUP-LR and 3.17% less than the OH-BAC-FUP-OPLR mechanisms.

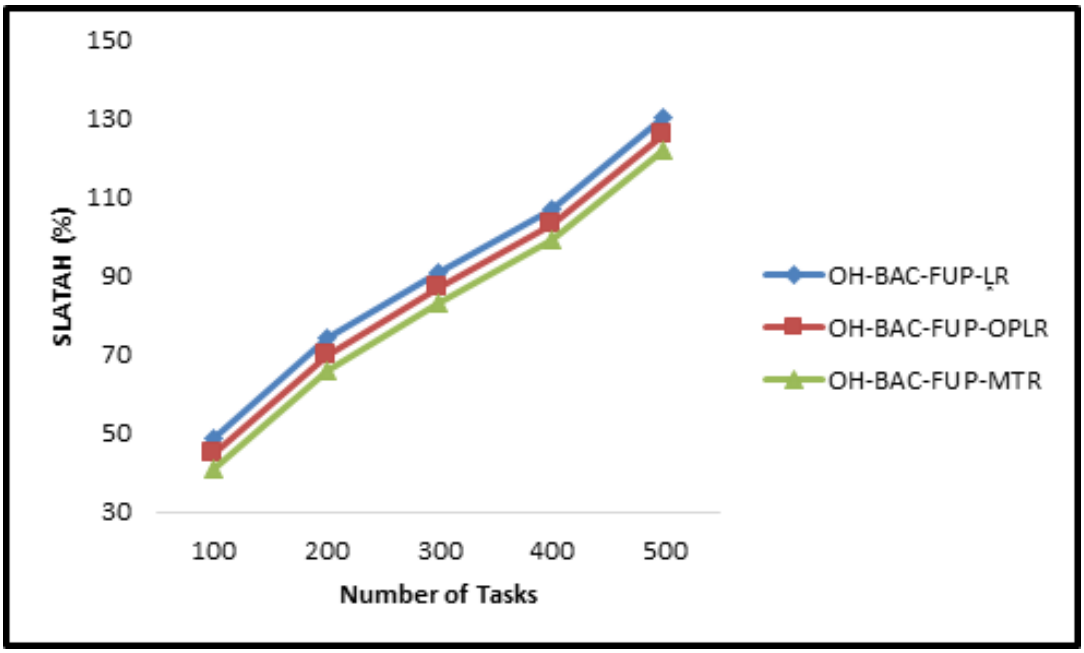

Fig 6. SLATAH vs. No. of Tasks

4.3 PDM

It is the overall performance degradation due to VM migrations and computed as:

$$PDM = \frac{1}{m}\sum_{i=1}^{m}\frac{C_{d_i}}{C_{r_i}} \tag{15}$$

In Eq. (15), m is the number of VMs, C_{d_i} denotes the estimate of performance degradation of i^{th} VM created by migrations and C_{r_i} is the overall CPU required by i^{th} VM. Consider C_{d_i} as 10% of the CPU usage in Million Instructions Per Second (MIPS) agreed on SLA in every migration of the i^{th} VM.

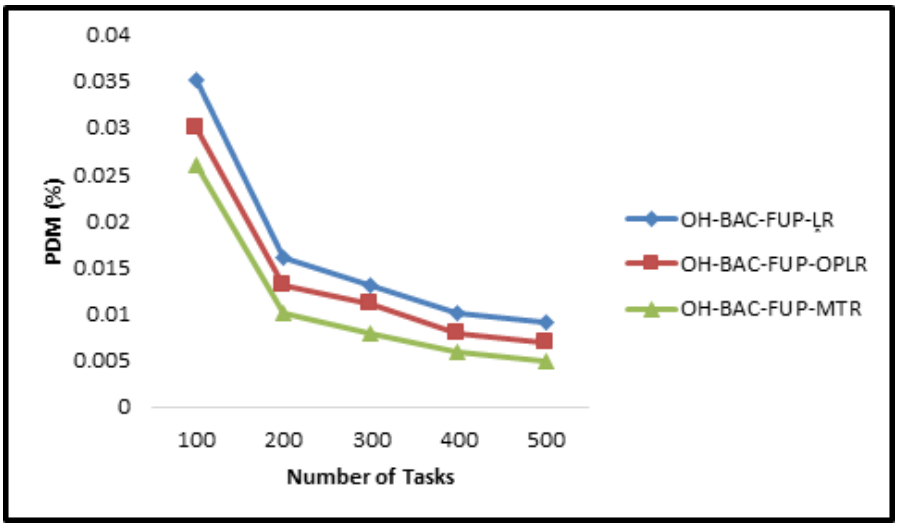

Fig 7. PDM vs. No. of Tasks

Figure 7 shows the PDM (in %) of OH-BAC-FUP-MTR and OH-BAC-FUP mechanisms under a varying number of tasks. This analysis indicates that the OH-BAC-FUP-MTR attains a minimum PDM and a maximum efficiency than the OH-BAC-FUP. For example, the PDM of OH-BAC-FUP-MTR for 500 tasks is 44.44% less than the OH-BAC-FUP-LR and 28.57% less than the OH-BAC-FUP-OPLR mechanisms.

4.4 SLAV

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

$$SLAV = SLATAH \times PDM \tag{16}$$

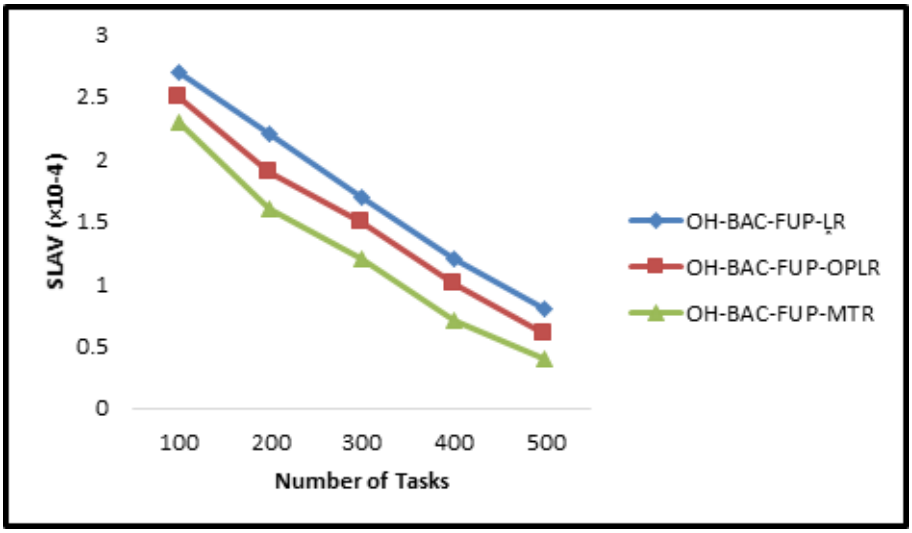

Fig 8. SLAV vs. No. of Tasks

Figure 8 shows the SLAV ($\times 10^4$ %) of OH-BAC-FUP-MTR and OH-BAC-FUP mechanisms under the different number of tasks. This scrutiny indicates that the OH-BAC-FUP-MTR realizes the minimum SLAV than the OH-BAC-FUP. For example, the SLAV of OH-BAC-FUP-MTR for 500 tasks is 50% less than the OH-BAC-FUP-LR and 33.33% less than the OH-BAC-FUP-OPLR mechanisms.

4.5 Number of VM migrations

It is the number of migrations created in the remapping phase.

Figure 9 shows the number of VM migrations for OH-BAC-FUP-MTR and OH-BAC-FUP mechanisms under the varying number of tasks. This scrutiny indicates that the OH-BAC-FUP-MTR attains less number of VM migrations compared to the OH-BAC-FUP. For example, the number of VM migration for OH-BAC-FUP-MTR during 500 tasks is 2 whereas OH-BAC-FUP-LR and OH-BAC-FUP-OPLR mechanisms have 4 and 3 VM migrations, respectively.

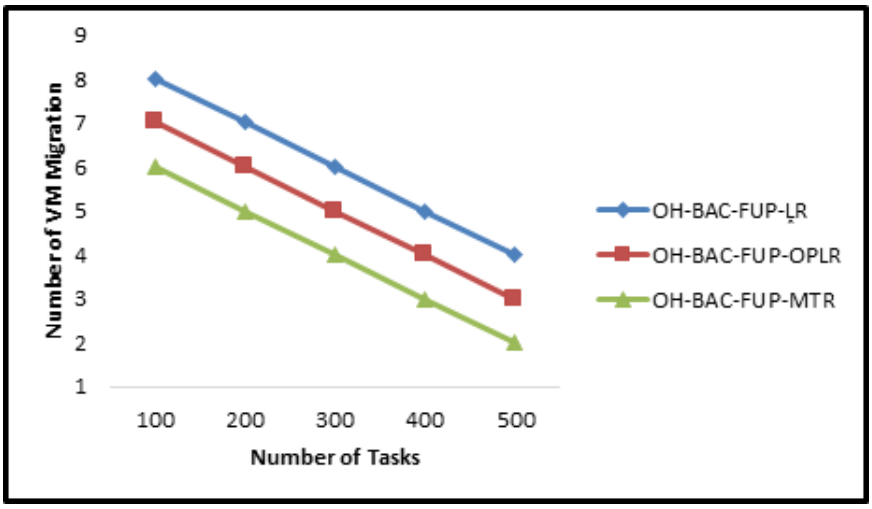

Fig 9. No. of VM Migrations vs. No. of Tasks

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 is moved, then the host is shut down to reduce energy use. But, the host is becoming operative when a VM has moved to it again.

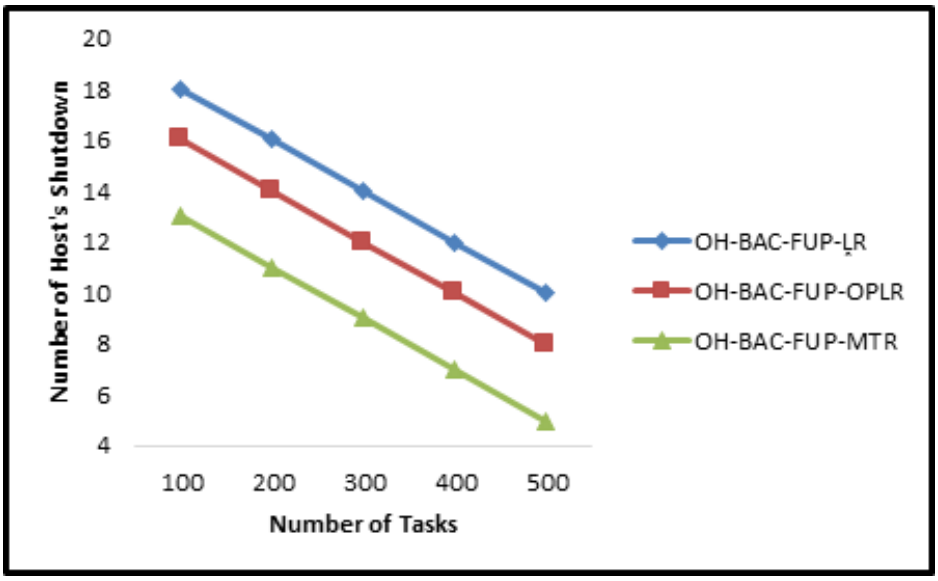

Fig 10. No. of Host's Shutdowns vs. No. of Tasks

Figure 10 shows the number of host shutdowns for OH-BAC-FUP-MTR and OH-BAC-FUP mechanisms under the different number of tasks. This scrutiny indicates that the OH-BAC-FUP-MTR realizes the minimum number of active hosts than the

OH-BAC-FUP. It means if a host is shut down, then it continues this stage for a while; therefore the effectiveness of OH-BAC-FUP-MTR is maximized. For example, the number of host's shutdowns for OH-BAC-FUP-MTR during 500 tasks is 5 whereas OH-BAC-FUP-LR and OH-BAC-FUP-OPLR mechanisms have 10 and 8 host's shutdowns, respectively.

5 Conclusion and future work

This study presents an OH-BAC-FUP-MTR mechanism for enhancing the performance of LB and reducing the chance of congestion occurrence in the cloud data centers. At first, an OH-BAC-FUP is applied to select the most suitable VMs to be migrated to the most appropriate PMs. During migration, if any congestion occurs because of using high bandwidth or traffic flows, then the MTR algorithm is performed to partition the flows and route them via multiple link-disjoint routes. By partitioning the traffic flows, the congestion is prevented when guaranteeing the bandwidth and security demands. As well, the highest traffic on a path is applied as a congestion factor. Moreover, the current and future network states are taken into account for MTR to select the most optimal route from multiple routes with no consideration of the past use of the paths. Finally, the simulation outcomes exhibited that the OH-BAC-FUP-MTR enhances the efficacy of VM migration than the OH-BAC-FUP. Scope of the OH-BAC-MTR enhances the performance of VM migration, the trade-off between load fairness and energy-saving in heterogeneous cloud computing was not effective. So, the future extension of this work could be focused on introducing workload-aware VM consolidation to measure the trade-off between load fairness and energy-saving in heterogeneous cloud scenarios.

References

1) Nashaat H, Ashry N, Rizk R. Smart elastic scheduling algorithm for virtual machine migration in cloud computing. *The Journal of Supercomputing*. 2019;75(7):3842–3865. Available from: https://dx.doi.org/10.1007/s11227-019-02748-2.
2) Shawish A, Salama M. Cloud computing: paradigms and technologies. In: Inter-cooperative Collective Intelligence: Techniques and Applications. Springer. 2014;p. 39–67. Available from: https://doi.org/10.1007/978-3-642-35016-0_2.
3) Afzal S, Kavitha G. Load balancing in cloud computing – A hierarchical taxonomical classification. *Journal of Cloud Computing*. 2019;8(1). Available from: https://dx.doi.org/10.1186/s13677-019-0146-7.
4) Sui X, Liu D, Li L, Wang H, Yang H. Virtual machine scheduling strategy based on machine learning algorithms for load balancing. *EURASIP Journal on Wireless Communications and Networking*. 2019. Available from: https://doi.org/10.1186/s13638-019-1454-9.
5) Pourghebleh B, Hayyolalam V. A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things. *Cluster Computing*. 2020;23(2):641–661. Available from: https://dx.doi.org/10.1007/s10586-019-02950-0.
6) Dong E, Fu X, Xu M, Yang Y. Low-Cost Datacenter Load Balancing With Multipath Transport and Top-of-Rack Switches. *IEEE Transactions on Parallel and Distributed Systems*. 2020;31:2232–2247. Available from: https://doi.org/10.1109/tpds.2020.2989441.
7) Rajeshkannan R, Aramudhan M. Comparative Study of Load Balancing Algorithms in Cloud Computing Environment. *Indian Journal of Science and Technology*. 2016;9(20):1–7. Available from: https://dx.doi.org/10.17485/ijst/2016/v9i20/85866.
8) Ghomi EJ, Rahmani AM, Qader NN. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*. 2017;88:50–71. Available from: https://dx.doi.org/10.1016/j.jnca.2017.04.007.
9) Villari M, Fazio M, Dustdar S, Rana O, Ranjan R. Osmotic Computing: A New Paradigm for Edge/Cloud Integration. *IEEE Cloud Computing*. 2016;3(6):76–83. Available from: https://dx.doi.org/10.1109/mcc.2016.124.
10) Villari M, Celesti A, Fazio M. Towards osmotic computing: Looking at basic principles and technologies. In: Conference on Complex, Intelligent, and Software Intensive Systems. Springer. 2017;p. 906–915. Available from: https://doi.org/10.1007/978-3-319-61566-0_86.
11) Gamal M, Rizk R, Mahdi H, Elnaghi BE. Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing. *IEEE Access*. 2019;7:42735–42744. Available from: https://dx.doi.org/10.1109/access.2019.2907615.
12) Prakash RG, Shankar R, Fupa DS. Future Utilization Prediction Algorithm based Load Balancing Scheme for Optimal VM Migration in Cloud Computing. *IEEE Fourth International Conference on Inventive Systems and Control*. 2020;p. 638–644. Available from: https://doi.org/10.1109/ICISC47916.2020.9171059.
13) Cziva R, Jouet S, Stapleton D, Tso FP, Pezaros DP. SDN-Based Virtual Machine Management for Cloud Data Centers. *IEEE Transactions on Network and Service Management*. 2016;13(2):212–225. Available from: https://dx.doi.org/10.1109/tnsm.2016.2528220.
14) Vu HT, Hwang S. A Traffic and Power-aware Algorithm for Virtual Machine Placement in Cloud Data Center. *International Journal of Grid and Distributed Computing*. 2014;7(1):21–32. Available from: https://dx.doi.org/10.14257/ijgdc.2014.7.1.03.
15) Reguri VR, Kogatam S, Moh M. Energy efficient traffic-aware virtual machine migration in green cloud data centers. *IEEE 2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing and IEEE International Conference on Intelligent Data and Security*. 2016;p. 268–273. Available from: https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.55.
16) Deshpande U, Keahey K. Traffic-sensitive Live Migration of Virtual Machines. *Future Generation Computer Systems*. 2017;72:118–128. Available from: https://dx.doi.org/10.1016/j.future.2016.05.003.
17) Liu J, Wu Z, Wu J, Dong J, Zhao Y, Wen D. A Weibull distribution accrual failure detector for cloud computing. *PloS One*. 2017;12. Available from: https://doi.org/10.1371/journal.pone.0173666.
18) Cui Y, Yang Z, Wang XS, Yan X, S. Traffic-aware virtual machine migration in topology-adaptive DCN. *IEEE/ACM Transactions on Networking*. 2017;25(6):3427–3440. Available from: https://doi.org/10.1109/TNET.2017.2744643.
19) Fu X, Chen J, Deng S, Wang J, Zhang L. Layered virtual machine migration algorithm for network resource balancing in cloud computing. *Frontiers of Computer Science*. 2018;12(1):75–85. Available from: https://doi.org/10.1007/s11704-016-6135-9.

20) Vakilinia S. Energy efficient temporal load aware resource allocation in cloud computing datacenters. *Journal of Cloud Computing*. 2018;7(1). Available from: https://doi.org/10.1186/s13677-017-0103-2.
21) Afzal S, Kavitha G. Optimization of Task Migration Cost in Infrastructure Cloud Computing using IMDLB Algorithm. In: International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET). 2018;p. 1–6. Available from: https://doi.org/10.1109/ICCSDET.2018.8821193.
22) Hejja K, Hesselbach X. Evaluating impacts of traffic migration and virtual network functions consolidation on power aware resource allocation algorithms. *Future Generation Computer Systems*. 2019;101:83–98. Available from: https://dx.doi.org/10.1016/j.future.2019.06.015.
23) Hsieh SY, Liu CS, Buyya R, Zomaya AY. Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. *Journal of Parallel and Distributed Computing*. 2020;139:99–109. Available from: https://dx.doi.org/10.1016/j.jpdc.2019.12.014.
24) Afzal S, Kavitha G. A Hybrid Multiple Parallel Queuing Model to Enhance QoS in Cloud Computing. *International Journal of Grid and High Performance Computing*. 2020;12(1):18–34. Available from: https://dx.doi.org/10.4018/ijghpc.2020010102.