Received: May 10, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Run Time Virtual Machine Task Migration Technique for Load Balancing in Cloud

Geetha Megharaj [1]* Mohan Kabadi [2]

[1]*Department of Computer Science and Engineering,*
Sri Krishna Institute of Technology, Bangalore, Karnataka, India
[2]*Department of Computer Science and Engineering,*
Presidency University, Bangalore, Karnataka, India
* Corresponding author's Email: geethagvit@yahoo.com

Abstract: Load Balancing is an important aspect of cloud service centers for optimizing the resource utilization. Consumption of excess power in cloud centers can result in monetary wastage. It is critical that, the resources in the cloud center are utilized optimally; so that, both monetary savings and client satisfaction can be achieved. One of the most popular techniques to achieve load balancing is the Virtual Machine (VM) migration technique; wherein, some of the VMs from overloaded Physical Machines (PMs) are migrated to lightly loaded PMs; however, this technique requires excessive time and monetary cost. Recently, a load balancing technique which migrates VM tasks instead of the actual VM was proposed in the literature. This technique was able to overcome some of the limitations of VM migration technique. Here, the overloaded VM does not accept any new task; however, the new tasks are migrated to lightly loaded VMs. Even though this technique migrates extra tasks to achieve VM load balancing, the already overloaded VMs are not relieved from their existing task burden. If some of the existing and suitable tasks in overloaded VMs are migrated, it could improve efficiency of load balancing. In this work, a new run time VM task migration technique is proposed, which migrates tasks from overloaded VMs. The suitable tasks for migration are selected through a discriminant function, which identifies heavy resource consuming and limited execution progressed tasks for migration. Since, it has been shown in the literature that, optimal task-resource mapping is NP-hard, Particle Swarm Optimization (PSO) based solution search technique is proposed. This proposed technique substantially reduces computing load, and achieves good power/energy conservation in the overloaded VMs, when compared with the contemporary VM task migration and VM migration techniques.

Keywords: Cloud computing, Load balancing, Virtual machine, Task migration.

1. Introduction

The Cloud Computing framework provides 3 classes of services namely— Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The first service class provides the required hardware computational resources for the requesting clients. The second class provides the necessary software platforms for the client to build the required applications. Finally, the complete application service is provided by SaaS model.

The cloud computing framework has been successful in cutting down the client computational costs by providing on-demand computational service, which helps the client from avoiding procurement of computational resources. Also, it provides new business opportunities through the establishment of commercial cloud centers. One of the significant advantages of Cloud Computing framework is scalability. The cloud centers are not usually confined to a single geographical location, but, the cloud computational resources are distributed in different geographical locations, and the services are provided through distributed framework.

The cloud centers usually contain numerous computing devices or Physical Machines (PM). Usually, each PM provides computing services to multiple users through logical system called as the Virtual Machine (VM). The computing resource details of the PM are abstracted from the VM user, and the user may feel that a single and independent computing system is providing the requested service of the user. The computing resources of a PM are divided among the different VMs built over it, and this resource division may be mutually exclusive w.r.t. different VMs.

Due to the large number of tasks and heavy computational workload, load balancing in cloud centers becomes extremely essential. Load balancing aids in increasing the efficiency of computing machines, and reduces the power expenditure cost.

The VM migration technique is a load balancing technique, which migrates overloaded VMs from one PM to the other, which is lightly loaded. It is one of the most popular load balancing technique in the cloud. However, there are important drawbacks for this approach of load balancing:

1. VM migration requires significant memory consumption, and excessive task execution downtime can occur because of stopping the functionality of the VM which is about to be migrated.
2. Customer activity information can be lost in the VM migration process, and it can increase the monetary expenditure of the users.
3. Significant increase in dirty memory can be seen after VM migration.

1.1 Research issues

To counter demerits due to VM migration technique, another load balancing strategy called as VM Task Migration was proposed in [1, 2]. These techniques do not migrate the overloaded VMs, but, migrate the extra tasks which are submitted to overloaded VMs. Even though, these techniques are able to counter most of the problems associated with VM migration technique, however, only address the overloading problem that occurs due to extra tasks; however, the overloaded VM is not subjected to existing load reduction. It is also important to identify suitable tasks which are currently running inside the VM and subject some of these tasks to migration in order to achieve efficient load distribution. Performing existing VM task migration also has certain challenges: the identified tasks for migration might have completed executing significant portion of their execution data; also, the identified tasks should reduce significant computational load in their original VM - when migrated; otherwise, the task migration itself becomes ineffective.

In-order to achieve even better task load reduction as achieved in [1]; in this work, current running tasks from overloaded VMs which are not only resource heavy in their execution requirements, they should have also completed only limited part of their entire execution cycle are identified to be migrated along with extra tasks of these overloaded VMs. Since, only resource heavy tasks which are currently being executed in the overloaded VMs are evicted, clearly, this mechanism can result in substantial task load reduction in the overloaded VMs; also, the computational effort wastage is limited, because the evicted tasks have only finished limited part of their execution cycle. Clearly, the proposed scheme can achieve substantial load reduction when compared to [1], because it migrates, not only the extra tasks of overloaded VMs as in [1], it also migrates the most resource heavy task running in each overloaded VM.

1.2 Contributions

The following contributions are made in this work:

1. The overloaded VMs are identified through a novel discriminant function, which selects the overloaded VMs based on power and computational resource consumption. The suitable tasks which are identified for migration are selected on two criteria namely—the task execution status and task migration benefit. The first criteria, avoids those tasks which have completed executing significant portion of the task specific data. The second criteria, avoids those tasks which consume limited computational resources.
2. A scoring function is designed which assigns migration score for a set of VMs which is a probable destination to host the migrated VMs. The VM set which has the lowest score is considered the optimal choice. Particle Swarm Optimization (PSO) technique is designed to search the candidate solutions space, because the search complexity is non-polynomial [1]. This technique provides an opportunity to implement parallel solution search. The PSO based solution search technique is implemented in MATLAB and its relative merits in computing load reduction, and power/energy conservation are exhibited against contemporary VM task migration technique [1],

and contemporary VM migration techniques [2, 3].

1.3 Organization of the paper

This paper is organized as follows: section 2 describes the related work in this area. The proposed VM task migration technique is outlined in section 3. The empirical results are presented in section 4. Finally, the work is concluded along with future avenues in section 5.

2. Related work

VM migration has been one of the popular load balancing techniques in cloud computing. In [2], VM migration technique focused on load balancing in data centers having multi-rooted tree format. In [3], VM migration technique addressed load balancing in distributed cloud centers; wherein, cloud resources are distributed in different geographical location. In [4], rapid migration scheme for VM migration was proposed. As explained above, even though VM migration techniques have demonstrated load balancing efficiency, they suffer from expensive cost of migration and possible task execution latency delays.

Task scheduling for load balancing in distributed systems—including cloud servers, deal with the problem of distributing the submitted task load on available computational units; so that, maximum utilization of these computational units, and substantial reduction in task execution time can be achieved. It must be noted that, task scheduling does not involve evicting already running tasks, and only distributes the newly submitted tasks for efficient computation. Also, overloaded VM problem is usually not addressed in task scheduling, because the task distribution scheme hypothesizes that, overloading will usually not occur.

A novel programming platform for task scheduling in cloud was presented in [5]. Genetic algorithm based task scheduling techniques for cloud was presented in both [6, 7]. Task scheduling technique for geographically distributed cloud centers was presented in [8]. Survey on different load balancing techniques for cloud was presented in [9]. Similarly, survey on meta-heuristic techniques for load balancing in cloud was presented in [10]. In [11], future problems for task scheduling in cloud were comprehensively presented. Dynamic Collaboration in cloud involves collaborative framework through different participating cloud service providers, and in [12], task scheduling in this new framework was presented. In [13] task scheduling technique for IaaS based cloud centers was presented.

Task scheduling technique through user requirement modeling for computational grids—which can also be relevant to cloud—was presented in [14]. Similarly, PSO based task scheduling technique for computational grids and cloud was presented in [15]. Security based task scheduling technique for cloud using Swarm scheduling approach was presented in [16]. Multi objective task scheduling involves achieving multiple goals such as: minimizing task latency, reducing power consumption e.t.c., and this problem for cloud was addressed in [17]. In [18], another multi objective task scheduling technique for cloud using genetic algorithm was presented. In [19], Honey Bee optimization technique for task scheduling in cloud was presented. In [20], task scheduling in computational grids—which can also be extended to cloud—was also achieved through Honey Bee optimization technique. Task scheduling technique for cloud using Ant Colony optimization framework was presented in [21]. In [22], task scheduling for cloud using probabilistic modeling was presented. In [23], task scheduling technique for cloud using specialized bio-inspired algorithm called: Symbiotic Organism Search, was presented. Multi objective task scheduling technique for cloud using Ant Colony optimization framework was presented in [24]. Hybrid task scheduling algorithm for cloud through merging of two techniques namely: Cuckoo search algorithm and Oppositional based learning was presented in [25]. In [26], evolutionary genetic algorithm framework was utilized to achieve task scheduling in cloud. Similarly, fruit fly optimization framework was utilized in [27] to design task scheduling technique in cloud.

Even though, task scheduling is effective in load balancing for cloud, in some scenarios, the estimated resource consumption for a certain task, which is used as critical parameter in task scheduling techniques, can deviate substantially compared to actual resource utilization—which can burgeon rapidly. In such scenarios, VMs can easily become overloaded, and has to be relived from this computational burden. The VM extra task migration techniques presented in [1, 28] achieves load reduction from overloaded VMs through migrating extra tasks. As outlined above, to achieve even better load reduction as achieved in [1, 28], some of the suitable running tasks in the overloaded VMs need to be identified and migrated—along with extra tasks.

3. VM task migration technique

Let, VM_y indicate the y^{th} VM, c_y indicates the number of computing node in VM_y, m_y indicates the memory capacity of VM_y, t_{iy} indicates the i^{th} task present inside VM_y, c_{iy} is the CPU utilization ratio of t_{iy}, if t_{iy} is running on multiple CPUs, then, c_{iy} is the sum of CPU utilization ratio for every CPU on which t_{iy} is being executed, m_{iy} represents the memory utilization ratio of t_{iy} and p_{iy} represents the power consumption of t_{iy}, which is represented in Eq. (1)

$$p_{iy} = c_{iy} \times m_{iy} \quad (1)$$

The total power consumed by all the tasks present in VM_y is indicated by the variable p_y and it is represented in Eq. (2) Here, n_y represents the total number of tasks that are being executed in VM_y.

$$\boldsymbol{p}_y = \frac{\sum_{j=1}^{ny} p_{jy}}{c_y} \quad (2)$$

Two thresholds are defined to detect overloaded VMs. The first threshold is defined over CPU utilization ratio, which is indicated by T_c. The second threshold is defined over power consumption, which is indicated by T_p. The VM_y is decided as overloaded if the value of the function *overloaded* $(VM_y)=1$, otherwise if, *overloaded*(VM_y) = 0, then, VM_y is decided as not-overloaded. This case is represented in Eq. (3)

$$Overloaded(VMy) = \begin{cases} 1, & if\ T_c \le \dfrac{\sum_{j=1}^{Ny} c_{jy}}{c_y} \\ & or \\ & T_p \le py \\ \\ 0, & otherwise \end{cases} \quad (3)$$

3.1 Task migration methodology

To select the suitable running tasks for migration, it is important to select those tasks which have completed executing only small portion of their data. The task completion ratio of t_{iy} is represented in Eq. (4). Here, *task_completion* (t_{iy}) indicates the task completion ratio of t_{iy}, d_{iy} is the size of data used by t_{iy} and $\hat{d}_{iy}$ indicates the size of data already consumed by t_{iy}.

$$task\_completion(t_{iy}) = \frac{\hat{d}iy}{diy} \quad (4)$$

The suitable tasks for migration are identified through their task completion ratio, CPU utilization and consumed power. The selected task should have the task completion ratio within the specified threshold indicated by T_o. This case is represented in Eq. (5). Since, stopping already executing tasks and migrating them into different VMs along with their data, reduces the task execution efficiency, so, only a single task which provides the maximum benefit in load reduction is selected for migration.

$$task\_completion(t_{iy}) \le T_o \quad (5)$$

The task which has the maximum combined value of both CPU utilization ratio and consumed power is selected for migration, and this case is represented in Eq. (6)

$$task\ selected\ for\ migration = \max_{t_{iy}} (c_{iy} + p_{iy}) \quad (6)$$

3.2 Scoring function for task migration

Suppose that, t_{iy} has to be migrated from VM_y and VM_z is one of the possible VM to which t_{iy} has to be migrated. The score of the migration task is represented in Eq. (7). The value of the parameters exe_{iz}, $transfer(t_{iy}, VM_z)$, p_z, $g(T_{cz}, t_{iy})$ and $g(T_{pz}, t_{iy})$ are represented in Eqs. (8), (9), (10), (11), and (12), respectively. Here, $score(t_{iy}, VM_z)$ indicates the migration score, exe_{iz} indicates the cost of executing tiy in VM_z, $transfer(t_{iy}, VM_z)$ indicates the transfer cost of transferring t_{iy} from VM_y to VM_z, bw_{yz} indicates the bandwidth between VM_y and VM_z and $\hat{p_z}$ is the power consumed by VM_z when task t_{iy} is migrated to VM_z. The functions $g(T_c, t_{iy})$ and $g(T_p, t_{iy})$ ensure that, the migration of t_{iy} from VM_y to VM_z does not cause CPU utilization threshold and power consumption threshold violations.

Consider the situation where the set of tasks [t_{i1y1}, t_{i2y2}, … t_{isys}] which need to be migrated. One of the candidate solution is the VM set [VM_{z1}, VM_{z2} ,…. VM_{zs}], such that, t_{i1y1} will be migrated to VM_{z1} , t_{i2y2} will be migrated to VM_{z2} and so on t_{isys} will be migrated to VM_{zs} . There is no restriction that, the VM_s in the candidate solution set should be distinct. The migration score for this candidate solution is represented in Eq. (13). Here, $t_{ijyj} \rightarrow VM_{zj}$ $(1 \le j \le s)$ indicates that the task t_{ijyj} has already been assigned to VM_{zj} and is being executed inside it. The CPU and memory utilization ratio of t_{ijyj} in VM_{zj} is assumed to be same as observed when t_{ijyj} was executing inside VM_{zj}. The operator | is interpreted as such that.

$$score(t_{iy}, VM_z) = exe_{iz} + transfer(t_{iy}, VMz) + \hat{p}_z - (g(T_c, t_{iy}) + g(T_p, t_{iy})) \quad (7)$$

$$exe_{iz} = \frac{d_y}{c_z \times c_{iy} + m_{z\times} m_{iy}} \quad (8)$$

$$transfer(t_{iy}, z) = \frac{d_{iy}}{bw_{yz}} \quad (9)$$

$$p_z = p_z + \frac{p_{iy}}{c_z} \quad (10)$$

$$g(T_c, t_{iy}) = \begin{cases} T_c - \frac{\sum_{j=1}^{nz} c_{jz} + c_{iy}}{c_z}, & if\ T_c - \frac{\sum_{j=1}^{nz} c_{jz} + c_{iy}}{c_z} > 0 \\ -\infty, & if\ T_c - \frac{\sum_{j=1}^{nz} c_{jz} + c_{iy}}{c_z} \leq 0 \end{cases} \quad (11)$$

$$g(T_p, t_{iy}) = \begin{cases} T_p - (p_z + \frac{p_{iy}}{c_z}), & if\ T_p - (p_z + \frac{p_{iy}}{c_z}) > 0 \\ -\infty, & if\ T_p - (p_z + \frac{p_{iy}}{c_z}) \leq 0 \end{cases} \quad (12)$$

$$migration\_score((t_{i1y1}, VM_{z1}), (t_{i2y2}, VM_{z2}), \ldots (t_{isys}, VM_{zs})) = score(t_{i1y1}, VM_{z1} | t_{i2y2} \rightarrow VM_{z2}, t_{i3y3} \rightarrow VM_{z3}, \ldots t_{isys} \rightarrow VM_{zs}) + score(t_{i2y2}, VM_{z2} | t_{i1y1} \rightarrow VM_{z1}, t_{i3y3} \rightarrow VM_{z3}, \ldots t_{isys} \rightarrow VM_{zs}) + \ldots\ldots score(t_{isys}, VM_{zs} | t_{i1y1} \rightarrow VM_{z1}, t_{i2y2} \rightarrow VM_{z2}, \ldots t_{i(s-1)y(s-1)} \rightarrow VM_{z(s-1)}) \quad (13)$$

The most beneficial candidate solution is the one which satisfies the optimization condition which is represented in Eq. (14).

$$optimization\ condition = \min_{(VM_{z1}, VM_{z2}, \ldots VM_{zs})} migration\_score((t_{i1y1}, VM_{z1}), (t_{i2y2}, VM_{z2}), \ldots (t_{isys}, VM_{zs})) \quad (14)$$

4. PSO technique for VM task migration

PSO is a meta-heuristic technique [28] which provides an approximate solution to the optimization problems, and it is inspired by the social behavior of the birds. The search for optimal solution is carried out by group of particles; wherein, each particle has an exclusive zone in candidate solution space, and union of all particle zones is equal to the candidate solution space. Each point in candidate solution space represents a candidate solution vector. The particles are continuously moving in their corresponding candidate solution space to identify optimal solution, and are involved in continuous communication for exchanging their locally discovered best solution, which in-turn decides the corresponding velocity of the particle for navigation. The particles continue their search until acceptable solution is obtained.

The PSO utilizes multiple search particles, which are collectively involved in discovering near optimal candidate solution for optimization problem. Here r search particles are assumed. The current position of the i[th] particle at iteration t be $\overrightarrow{X_i}(t)$. The position for the next iteration is indicated by $\overrightarrow{X_i}(t+1)$, which is calculated as represented in Eq. (15) Here, $\overrightarrow{V_i}(t)$ indicates the velocity of the i[th] particle for *t + 1* iteration, and it is calculated as represented in Eq. (16). Here, D1 and D2 indicate the degree of particle attraction towards individual and group success respectively, $\vec{x}_{\text{gbest and}}$ $\vec{x}_{\text{pbesti}}$ indicate the global best solution obtained by all the particles until the current iteration respectively, W indicates a control variable, and r1,r2 ∈ [0, 1] are the random factors.

$$\overrightarrow{X_i}(t+1) = \overrightarrow{X_i}(t) + \overrightarrow{V_i}(t+1) \quad (15)$$

$$\overrightarrow{V_i}(t+1) = W\overrightarrow{V_i}(t) + D1r1(\vec{x}_{pbesti} - \overrightarrow{X_i}(t)) + D2r2\ (\vec{x}_{gbest} - \overrightarrow{X_i}(t)) \quad (16)$$

The proposed PSO based VM task migration technique for load balancing is outlined in Algorithm 1. Here, *initialize_PSO(P)* divides the candidate solution space among the *r* search particles indicated by P=p1, p2,....pr, and assigns each particle to some arbitrary positions in their corresponding candidate solution space. Each particle calculates its candidate solution for the corresponding current position through *compute_score(**$\overrightarrow{X_i}(t)$**)*, which utilizes Eqs. (15) and (16). The values for $\vec{x}_{\text{gbest}}$ and $\vec{x}_{\text{pbesti}}$ are calculated through *local_best(score$_i$)* and *global_best(P, $\vec{x}_{pbesti}$)* respectively. The particles continue to search until the acceptable solution is found, and which is calculated through *acceptable($\vec{x}_{gbest}$)*.

Algorithm 1 PSO Algorithm for VM task migration

```
P=p1, p2 ...pr
initialize_PSO (P)
flag=0
t=0
While flag==0 do
        t=t+1
        For i=1 to r do
                score_i= compute_score (X_i(t))
```

$\vec{x}_{pbesti}$ = *local_best(score$_i$)*
$\vec{x}_{gbest}$ = *global_best(P, $\vec{x}_{pbesti}$)*
If *acceptable($\vec{x}_{gbest}$)* then
flag=1
end if
end for
t=t+1
end while

5. Experimental results and discussions

The proposed PSO technique for VM task migration is simulated on MATLAB. The simulation parameter settings are presented in Table 1. Each PSO search particle is assumed to be executing on an exclusive computing node, so that, maximum parallelism is exploited from both these techniques.

To perform relative performance evaluation, the proposed VM runtime task migration technique is coupled with VM extra task migration technique[1], and for the ease of reference, this coupled scheme will be referred as *VM_run*. Here, *VM_run* identifies the overloaded VMs, and migrates the extra and suitable runtime tasks from the overloaded VMs. For the ease of reference, the VM extra task migration technique [1] will be referred as *VM_extra*, and this technique only migrates the extra tasks of overloaded VMs. Both, *VM_run* and *VM_extra* are subjected to relative performance evaluation.

Two performance metrics are defined for performing simulation analysis. The first metric is - *Average CPU Utilization Ratio (ACPUR)*, which indicates the average CPU utilization ratio of all overloaded VMs after task load reduction is performed by either *VM_run* or *VM_extra*. The metric *ACPUR* is represented in Eq. (17). Here, OS represents the set of overloaded VMs. Also, *ACPUR(VM_run)* and *ACPUR(VM_extra)* indicate the *ACPUR* score achieved by *VM_run* and *VM_extra* respectively.

$$ACPUR = \frac{\sum_{VM \in OS}^{|OS|} cpur(VM_y)}{|OS|} \quad (17)$$

where $cpur(VM_y) = \frac{\sum_{j=1}^{ny} c_{jy}}{c_y}$

The second performance metric is: *Average Power Utilization Ratio (APUR)*, which indicates the average power utilization of all overloaded VMs after task load reduction is performed by either *VM_run* or *VM_extra*. The metric *APUR* is represented in Eq. (18). Also, *APUR(VM_run)* and *APUR(VM_extra)* indicate the *APUR* score achieved by *VM_run* and *VM_extra* respectively.

Table 1. Simulation parameter settings

Simulation Parameter	Values
Number of VMs considered	Varied between 1000- 5000
Number of computing nodes/CPUs in each VM	Varied between 5 to 20
Main memory capacity for each VM	Varied between 4GB/ 8GB/16GB
Number of tasks executing in each VM	Varied between 10 to 50 tasks
Bandwidth between any 2 VMs	Varied between 100mbps to 500mbps
CPU utilization ratio for any task	Varied between 0.2 to 0.8
Memory utilization of each task	Varied between 0.2 to 0.8
Number of PSO search particles	Varied between 5 – 25
Number of Computing nodes allotted for each PSO particle	1
Threshold T_c	0.7
Threshold T_p	0.6
Size of task data	Varied between 1GB to 10GB
Threshold T_o	Varied between 0.05 – 0.25

$$APUR = \frac{\sum_{VM \in OS}^{|OS|} P_y}{|OS|} \quad (18)$$

VM_run is also compared against the contemporary VM migration technique presented in [2] and [3]. For the ease of reference VM migration techniques presented in [2] and [3] are denoted as *VM_M_1* and *VM_M_2* respectively. Since, optimal scheme for VM migration is NP hard [2], approximation algorithm is utilized in [2] for VM migration. In [3], Bin Packing framework is used to model VM migration problem. Again, optimal Bin Packing is NP hard [3]. Hence, approximate Bin Packing technique is utilized for VM migration in [3]. In-order to perform empirical analysis between *VM_run*, *VM_M_1* and *VM_M_2*, the metric AEXE is utilized, which represents the average execution time of all the tasks—including extra and running tasks—from overloaded VMs--after either of *VM_run*, *VM_M_1* or *VM_M_2* is executed. The metric AEXE is represented in Eq. (19). Here, $(t_1, t_2, \ldots . t_K)$ represent the set of all extra and running tasks $\in OS$. Here, $exe(t_i)$ ($1 \leq i \leq K$), represents the execution time of t_i after either of

VM_run, *VM_M_1* or *VM_M_2* is executed, and this metric is calculated as represented in Eq. (8). Also, *AEXE(VM_run)*, *AEXE(VM_M_1)* and *AEXE(VM_M_2)* represent the AEXE value for *VM_run*, *VM_M_1* and *VM_M_2* respectively.

$$AEXE = \frac{\sum_{i=1}^{K} exe(t_i)}{K} \quad (19)$$

The first experiment analyzes the performance of *VM_run* and *VM_extra* w.r.t. *ACPUR*, when the number of VMs used in simulation is varied. The result of this experimental analysis is illustrated in Fig.1 Here, *VM_run* considerably outperforms *VM_extra*, because existing running tasks inside overloaded VMs have no impact from migration of only extra tasks by *VM_extra;* however, *VM_run* selects the most resource consuming task running inside overloaded VM, and subjects the task to migration. Hence, appreciable reduction in *ACPUR* is observed for *VM_run*. Also, as the number of overloaded VMs increase, it creates a tendency to produce more overloaded VMs. Some of these overloaded VMs might have high resource consuming tasks, and evicting these tasks creates more resource release. Hence, *ACPUR(VM_run)* improves as the number of VMs increase.

The analysis result of first experiment w.r.t. *APUR* is illustrated in Fig. 2 Similar results seen in Fig. 1 can be observed here, and for the same reasons outlined for Fig. 1. The execution latency of both *VM_run* and *VM_extra* for first experiment is illustrated in Fig. 3. Here, *VM_run* is slightly expensive due to the extra component of performing VM runtime task migration.

The second experiment analyzes the performance of *VM_run* and *VM_extra* w.r.t. *ACPUR* and *APUR*, when the parameter T_o is varied. The result of second experiment analysis is presented in Fig. 4 and Fig. 5 respectively. Here, *VM_run* outperforms *VM_extra* for the same reasons outlined for first experiment. Increase in T_o values creates an opportunity to select the suitable tasks for migration from a more expanded set of qualified tasks. Hence, some of these qualified tasks for migration - usually are more resource consuming, their eviction creates better resource release. Hence, *ACPUR (VM_run)* improves as T_o increases. However, higher values of T_o can lead to the problem of significantly wasting the existing computed result of the migrated tasks.

The third experiment explores the influence of the number of particles on the execution latency of *VM_run*. The result of final experimental analysis is illustrated in Fig. 6. The increase in number of particles leads to the reduction in the search space size for each particle; hence, execution efficiency of *VM_run* improves considerably.

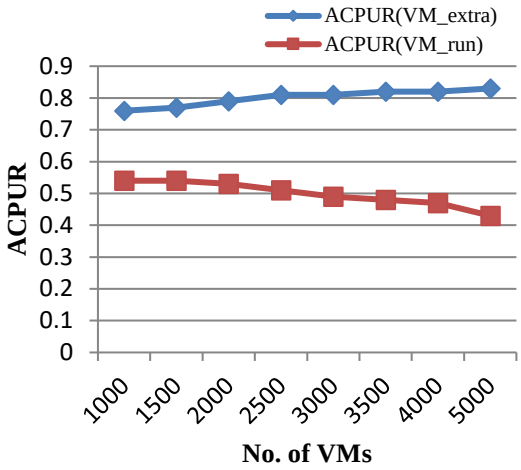

Figure.1 No. of VMs vs *ACPUR*

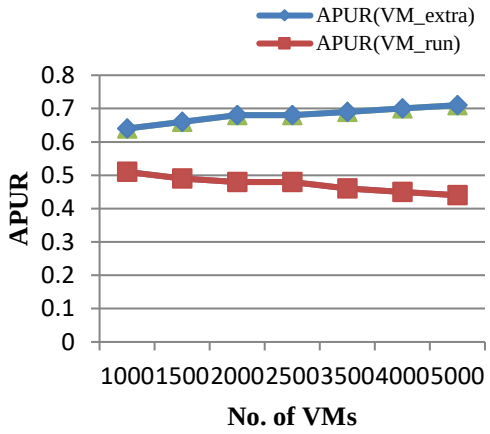

Figure 2. No. of VMs vs *APUR*

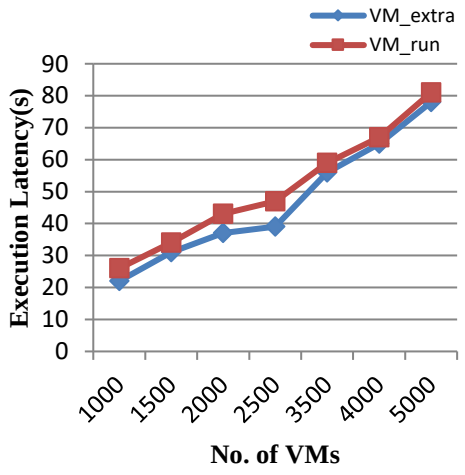

Figure. 3 No. of VMs vs Execution latency

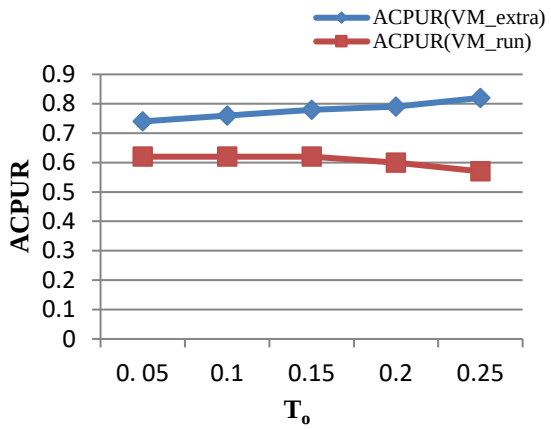

Figure. 4 T_o vs *ACPUR*

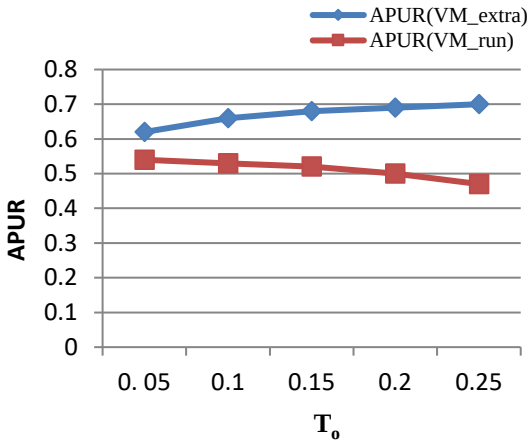

Figure. 5 T_o vs *APUR*

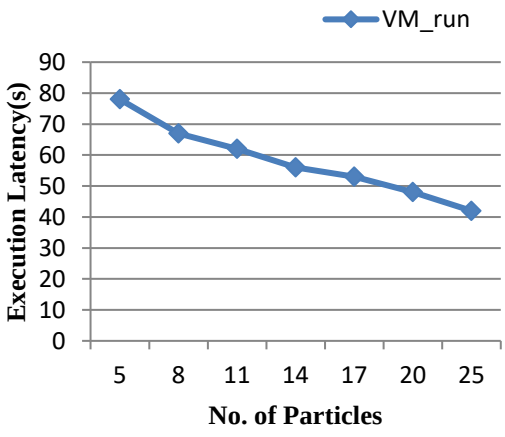

Figure. 6 No. of Particles vs Execution Latency

The final experiment analyzes the relative performance of *VM_run*, *VM_M_1* and *VM_M_2* through *AEXE* metric. The result of the final experimental analysis when number of VMs and T_o are varied is represented in Fig. 7 and Fig. 8. Here, *VM_run* substantially out performs the other two VM migration techniques, because only extra tasks and a single running task from overloaded VM are

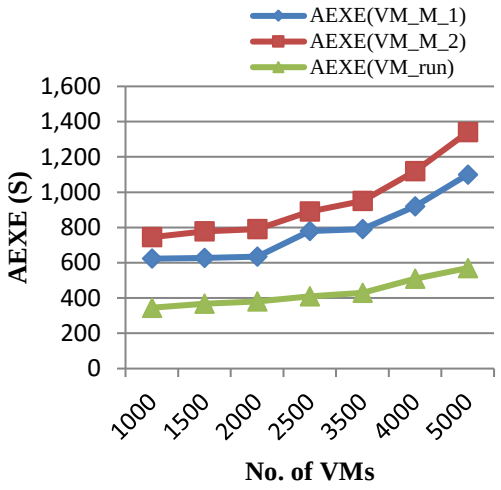

Figure. 7 No. of VMs vs *AEXE*

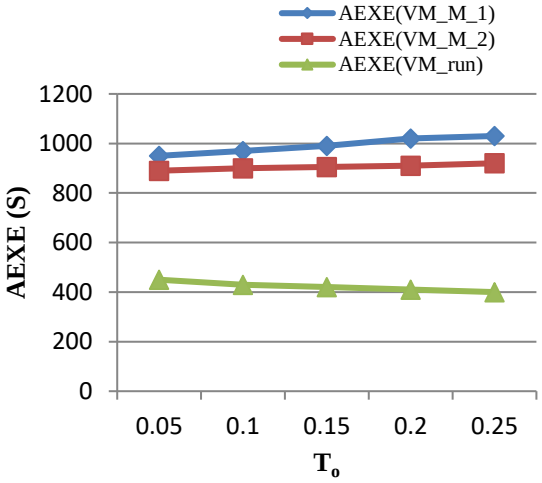

Figure. 8 T_o vs *AEXE*

migrated in *VM_run*; however, all the extra and running tasks from overloaded VM are migrated in *VM_M_1* and *VM_M_2*, which causes substantial task execution delays in migrated Physical Machines due to resource competition from already running tasks in those machines.

6. Conclusion

In this work, the benefits of migrating suitable running tasks along with extra tasks from overloaded VMs were outlined. The contemporary VM task migration technique, migrates only the extra tasks from overloaded VMs. However, the proposed technique—in-order to achieve superior load balancing--migrates the most resource consuming tasks along with extra tasks from these overloaded VMs. Also, the evicted running task is also ensured to have completed limited part of its execution work load, which prevents wastage of

substantial computational effort. The proposed technique—as expected—provides superior load balancing merits when compared empirically with contemporary VM task migration and VM migration techniques.

In future, the merits of applying VM task migration schemes for Distributed Cloud Center in which, the cloud center is distributed in different geographical locations need to be analyzed. Task migration in this new setting faces multiple challenges, because migrating tasks to different locations in the cloud center can result in performance limitation due to large geographical distances.

References

[1] F. Ramezani, J. Lu, and F. K. Hussain, "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization", *International Journal of Parallel Programming*, Vol. 42, No. 5, pp. 739-754, 2014.

[2] N. Jain, I. Menache, J. Naor, and F. Shepherd, "Topology Aware VM Migration in Bandwidth Oversubscribed Datacenter Networks", In: *Proc. of the 39th International Colloquium*, pp. 586-597, 2012.

[3] S. Kumaraswamy and K.N. Mydhilli, "Virtual Machine Placement in Distributed Cloud Centers using Bin Packing Algorithm", *International Journey of Grid and Utility Computing*, 2018.

[4] C.P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M.S. Lam, and M. Rosenblum, "Optimizing the Migration of Virtual Computers", *ACM SIGOPS Oper. Syst. Rev*. 36 (SI), 377390, 2002.

[5] A. Whitaker, R.S. Cox, M. Shaw, and S.D. Gribble, "Constructing Services with Interposable Virtual Hardware", In: *Proc. of the 1st Symposium on Networked Systems Design and Implementation*, pp. 169-182, 2004.

[6] A.Y. Zomaya and Y.-H. Teh, "Observations on Using Genetic Algorithms for Dynamic Load Balancing", *IEEE Transactions on Parallel Distributed Systems*, Vol. 12, No. 9, pp. 899-911, 2001.

[7] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent Tasks Scheduling based on Genetic Algorithm in Cloud Computing", In: *Proc. of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-4, 2009.

[8] E. Juhnke, T. Dornemann, D. Bock, and B. Freisleben, "Multi Objective Scheduling of BPEL Workflows in Geographically Distributed Clouds", In: *Proc. of the 4th IEEE International Conference on Cloud Computing*, pp.412-419, 2011.

[9] M. Geetha and K.G. Mohan, "A Survey on Load Balancing Techniques for Cloud Computing", *IOSR Journal of Computer Engineering*, Vol. 18, No. 2, pp. 55-61, 2017.

[10] S. Poonam, D. Maitreyee and A. Naveen, "A Review of Task Scheduling Based on Metaheuristics Approach in Cloud Computing", *Journal Knowledge and Information Systems*, pp 1-51, 2017.

[11] M.A. Sadeghi and N.N. Jafari, "Load Balancing Mechanisms and Techniques in the Cloud Environments", *Journal of Networks and Computer Applications*, Vol. 71, pp 86-98, 2016.

[12] B. Song, M.M. Hassan, and E. Huh, "A Novel Heuristic-based Task Selection and Allocation Framework in Dynamic Collaborative Cloud Service Platform", In: *Proc. of the 2nd IEEE International Conference on Cloud Computing Technology and Science*, pp.360-367, 2010.

[13] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online Optimization for Scheduling Preemptable Tasks on IaaS Cloud Systems", *Journal of Parallel and Distributed Computing*, Vol. 72, No. 5, pp. 666-677, 2012.

[14] J. Kolodziej and F. Xhafa, "Modern Approaches to Modeling User Requirements on Resource and Task Allocation in Hierarchical Computational Grids", *International Journal of Applied Mathematics and Computer Science*, Vol. 21, No. 2, pp. 243-257, 2011.

[15] Z. Lei, C. Yuehui, S. Runyuan, J. Shan, and Y. Bo, "A Task Scheduling Algorithm based on PSO for Grid Computing", *Int. J.Comput. Intell. Res*., Vol. 4, No. 1, pp. 37-43, 2008.

[16] H. Liu, A. Abrahan, V. Snasel, and S. McLoone, "Swarm Scheduling Approaches for Workflow Applications with Security Constraints in Distributed Data-intensive Computing Environments", *Journal Information Sciences*, Vol. 192, pp. 228-243, 2012.

[17] F. Ramezani, J. Lu, and F. Hussain, "Task Based System Load Balancing Approach in Cloud Environments", *Knowledge Engineering and Management*, pp. 31-42, 2014.

[18] F. Ramezani, L. Jie, T. Javid, and H.F. Khadeer, "Evolutionary Algorithm-based Multi-objective Task Scheduling Optimization Model in Cloud

Environments", *Journal World Wide Web*, Vol. 2015, pp 1737-1757, 2015.
[19] D. B. LD, and P. V. Krishna, "Honey Bee Behavior Inspired Load Balancing of Tasks in Cloud Computing Environments", *Applied Soft Computing Journal*, Vol. 13, No. 5, pp. 2292-2303, 2013.
[20] J. Taheri, Y.C. Lee, A.Y. Zomaya, and H.J. Siegel, "A Bee Colony based Optimization Approach for Simultaneous Job Scheduling and Data Replication in Grid Environments", *Comput. Oper. Res.,* Vol. 40, No. 6, pp. 1564-1578, 2013.
[21] J.-F. Li, J. Peng, X. Cao, and H.-Y. Li, "A Task Scheduling Algorithm based on Improved Ant Colony Optimization in Cloud Computing Environment", *Energy Procedia,* Vol. 13, pp. 6833-6840, 2011.
[22] R. Shiva, N. A. Habibizad, R. A. Masoud, and H. Mehdi, "Probabilistic Modeling to Achieve Load Balancing in Expert Clouds", *Journal Ad Hoc Networks*, pp 12-23, 2017.
[23] M. Abdullahi, M.A. Ngadi, and S.M. Abdulhamid, "Symbiotic Organism Search Optimization Based Task Scheduling in Cloud Computing Environment", *Future Generation Computing Systems*, Vol. 56, pp 640-650, 2016.
[24] G. Reddy, N. Reddy, and S. Phanikumar, "Multi Objective Task Scheduling Using Modified Ant Colony Optimization in Cloud Computing", *International Journal of Intelligent Engineering and Systems*, Vol.11, No.3, pp. 242-250, 2018.
[25] P. Krishnadoss and P. Jacob, "OCSA: Task Scheduling Algorithm in Cloud Computing Environment", *International Journal of Intelligent Engineering and Systems*, Vol.11, No.3, pp. 271-279, 2018.
[26] A.B.A. Muthu and S. Enoch, "Optimized Scheduling and Resource Allocation Using Evolutionary Algorithms in Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol.10, No.5, 2017.
[27] M. LawanyaShri, S. Subha and B. Balusamy, "Energy-Aware Fruit-fly Optimisation Algorithm for Load Balancing in Cloud Computing Environments", *International Journal of Intelligent Engineering and Systems*, Vol.10, No.1, pp. 75-85, 2017.
[28] M. Geetha and G.K. Mohan, "Metaheuristic Based Virtual Machine Task Migration Technique for Load Balancing in Cloud", In: *Proc. of the Second International Conference on Integrated Intelligent Computing, Communication and Security*, 2018.