A workload and machine categorization based resource allocation framework for load balancing and balanced resource utilization in cloud

[1]Avnish Thakur, [2]Major Singh Goraya
Sant Longowal Institute of Engineering and Technology, Longowal
[1]avi.himachal@gmail.com, [2]mjrsingh@yahoo.com

Abstract: This paper proposes a workload and machine categorization based resource allocation framework for balancing the load across active physical machines as well as utilizing their different resource capacities in a balanced manner. The workload, essentially independent and non-preemptive tasks are allocated resources on the physical machines whose resource availability complements the resource requirement of tasks. Simulation based experiments are performed using CloudSim simulator to execute three different set of tasks comprising 1×10^4, 2×10^4, and 3×10^4 number of tasks. The metric of load imbalance across active physical machines and the metric of utilization imbalance among their considered resource capacities (i.e., CPU and RAM) are measured in different scheduling cycles of a simulation run. Simulation results show that the proposed resource allocation method outperforms the compared methods in terms of balancing the load across active physical machines and utilizing their different resource capacities in a balanced manner.

Keywords: Cloud computing, Resource allocation, Load balancing, Workload categorization, Machine categorization

1. Introduction

Cloud computing [1]–[4] is an internet-based service delivery model to provide on-demand virtual access to the hardware and software computing resources of the data center. It minimizes the need of hardware and software computing resources at the consumer end. This significantly reduces the cost of ownership. The large consumer base of cloud services impose high load on data centers. This necessitates the efficient utilization of data center resources in order to provide quality services and to minimize the overall operational cost of the data center [5], [6]. The performance of computing machines is crucial to service quality whereas resource utilization primarily decides the operational cost. Load balancing plays an important role in the utilization and performance of computing machines in the data center.

Imbalanced distribution of workload may lead to the overloading of physical machines (PMs) which degrades the performance whereas imbalanced utilization of different resource capacities in a PM may lead to resource wastage. In literature [7]–[12], it is observed that virtual machine (VM) migration and task migration are the primarily adopted approaches to balance the load imbalance in cloud computing environment. The migration of VMs may lead to service downtime and high computational and network overheads. In the migration of tasks the major challenge is the availability of compatible VMs.

This paper proposes a resource allocation based proactive load balancing framework to avoid the need for VM/task migrations. In the proposed framework, the physical resources are allocated to the required VMs with an objective of balancing the load across active PMs and to utilize their different considered resource capacities (i.e., CPU and RAM) in a balanced manner. In the proposed resource allocation strategy the incoming workload (i.e., batch of independent tasks) and target PMs are categorized into the categories of CPU-intensive and RAM-intensive. The tasks and PMs of different categories are ordered on the basis of their affinity to the corresponding category and incongruity to the other category. The computing resources to deploy a VM for executing a task are first searched in the PMs of related category and only if

the resources are not found in the related category the search is extended to the PMs of other category. This categorization and ordering based resource allocation methodology serve to balance the load across active PMs as well as utilize their resource capacities in a balanced manner. The proposed approach is fast and easy to implement as it uses simple mathematical calculations using available data and does not use complex and computation/time intensive prediction system, optimization algorithms, etc.

Simulation-based experiments are performed to execute different number of tasks. The metric for load imbalance across active PMs and the metric of utilization imbalance among their resource capacities are computed to assess the performance of the proposed resource allocation strategy. Simulation results show that the proposed workload and machine categorization based resource allocation leads to comparatively better load balancing and balanced resource utilization. The rest of this paper is organized as follows: Section 2 discusses the related literature. Section 3 discusses the architecture of the proposed resource allocation framework. Section 4 explains the implementation of the proposed workload and machine categorization based resource allocation. Section 5 first discusses the simulation results and then presents the comparative statistical analysis of the obtained results. Section 6 concludes the work presented in this paper.

2. Related work

In literature, numerous heuristic and metaheuristic based procedures have been proposed for balancing the load. In cloud, load balancing follows reactive/proactive approaches to balance the load. This section primarily discusses heuristic based proactive load balancing strategies proposed in literature for cloud.

In reactive approach, the load balancing procedures are triggered when an overload/underload occurs. The load balancing strategy proposed in [13] determines the over utilized VMs on the overloaded PMs and transfer their excess tasks onto compatible low utilization VMs on low utilization PMs. Simulation results show reduction in overall makespan and energy consumption. The load balancing frameworks proposed in [14], [15] aims to minimize makespan time and task rejection ratio in cloud. To resolve the problem of overutilization/underutilization in VMs, a task migration strategy is adopted. The number of VMs are increased/decreased on the basis of monitoring data corresponding to previous intervals in order to minimize the percentage of underloaded VMs and rejected tasks. The load balancing framework proposed in [16] balances the overloaded PMs by migrating the VMs whose resource usage pattern of different resource capacities is similar to that of their host machine. This also helps in utilizing different resource capacities in a balanced manner. The destination PM in the process of VM migration is selected such that the requirement of different resource capacities matches the availability of resources on the machine. Framework also aims to keep the VMs with higher communication rates on the same PM. The VM migration is triggered only when the overload state of a PM persists at least for the specified time length. This prevents unnecessary VM migrations due to the transient overload state of PMs. Simulation results show that the proposed framework leads to a better load balancing effect and less number of VM migrations.

Proactive load balancing aims to avoid overloading/underloading of the computing resources. The load balancing framework proposed in [17] learns the present computing capability of servers on the basis of their throughput in previous scheduling cycles. The workload is distributed among servers proportional to their estimated capabilities in order to balance the throughput which in turn balances the load across active servers. Simulation results show improvement in response time, makespan and throughput. Authors in [18] proposed a multi-resource aware bin-packing approach for resource allocation in cloud to utilize different resource capacities of PMs in a balanced manner. Resources are allocated on a PM whose availability of different resources best matches the resource requirements of given task. Proposed strategy improves

performance and resource utilization. Authors in [19] proposed a resource allocation based load balancing framework. Tasks are classified into different categories in accordance to their relative requirements of different resources. The target PMs are sorted in different lists according to the availability of different resource capacities. For allocating physical computing resources to execute a task the related sorted list of PMs is searched. Simulation results show improvement in response time, deadline violation rate, load balancing, and resource utilization. Authors in [20] proposed a honey bee behavior based resource allocation framework to avoid overloading/underloading of physical/virtual machines. Tasks are distributed in accordance to the load on physical and virtual machines. The tasks which are allocated VMs for execution informs the other tasks waiting to be scheduled about the load and resource availability information of their respective host VM/PM. Simulation results show relatively low average response time, low average execution time, low makespan, and low degree of load imbalance. Authors in [21] proposed a fuzzy-based prediction system to predict the future load of VMs on the basis of monitored data, and adopts an appropriate load distribution strategy to prevent overloading/underloading of the VMs. Simulation results show low makespan, less response time, improvement in resource utilization, less number of task migrations, and a relatively low degree of load imbalance.

Load balancing frameworks may also implement a blend of reactive/proactive approaches to balance the load. Authors in [22] proposed a prediction-based resource allocation system to avoid overloading/underloading of PMs. It also aims to utilize different resource capacities of the active PMs in a balanced manner. It uses VM migration to resolve the overload/underload of the PMs. Simulation results show reduction in the number of active PMs, reduced number of VM migrations, and a better load balancing effect. Authors in [23] proposed a load balancing framework using a regression based prediction model. The monitoring agents monitor the resource usage of VMs and their corresponding physical hosts. The physical resource allocation for the required VMs aims to minimize resource wastage. A warning threshold is used to limit the allocation of resources on PMs which may otherwise get overloaded. The VM migration is triggered only when the current load and the future load prediction for a minimum specified time length indicates overloading/underloading of an active PM. This helps in avoiding unnecessary VM migrations due to transient overload/underload scenarios. Simulation results show reduction in resource wastage, improvement in resource utilization, and reduction in the number of VM migrations.

Following observations are inferred from the related literature:

- Reactive approaches primarily use VM/task migrations to mitigate load imbalances which could be time and resource intensive, and may lead to slowdown and service downtime.
- Sophisticated load balancing frameworks have used different complex and resource intensive components, e.g., prediction models, fuzzy-based systems, etc.
- Most of the above discussed load balancing frameworks aims to balance the load across active PMs. Only a few target to utilize different resource capacities of PMs in a balanced manner.

To overcome the above mentioned issues, this paper propose a workload and machine categorization based proactive load balancing framework to balance the load across active PMs and utilize their different resource capacities in a balanced manner. The proposed framework is simple yet effective in achieving the desired objectives.

3. System model

In this research work, the datacenter is comprised of heterogeneous PMs with multi-core processors. A physical machine, PM_i is characterized by the processing power of its CPU, PM_i^{CPU} measured in million instructions per second (MIPS) and the capacity of its RAM, PM_i^{RAM} measured in gigabyte (GB). A PM

can host multiple VMs. A dedicated processor core is allocated to each VM. The *j*th VM on the *i*th PM is characterized by the processing power of CPU core (VM_{ij}^{CPU}) allocated to it and the maximum allowed limit of RAM (VM_{ij}^{RAM}) assigned to it for usage.

Independent and non-preemptive tasks with heterogeneous resource requirements are assumed to arrive in batches following poisson distribution. A task is characterized by its arrival time, deadline time, and its length measured in million instructions (MI).

3.1 Architecture of the proposed resource allocation framework

The proposed resource allocation framework is composed of five main components: 1) a *local resource manager* for each PM; 2) a *global resource manager*; 3) a *task handler*; 4) a *resource handler*; and 5) a *scheduler*. The *local resource managers* of different PMs report the resource information of their respective PMs to the *global resource manager*. When a batch of tasks arrives, the *task handler* module deduces the resource requirement information and sends it to the *resource handler* module. The *resource handler* on receiving resource requirements fetches the resource availability information from *global resource manager* to deduce a target set of PMs and report the mean resource availability in target PMs to the task handler module. The *task handler* and *resource handler* modules categorize tasks and target PMs, respectively, as CPU-intensive/RAM-intensive. The tasks/target PMs are ordered in their respective categories on the basis of their affinity to the related category and incongruity to the other category. The scheduler module is responsible for mapping tasks onto appropriate target PMs by first searching the related category. The deployment of VMs for executing the given set of tasks is realized by the *global resource manager* using the *local resource managers* of the designated target PMs.

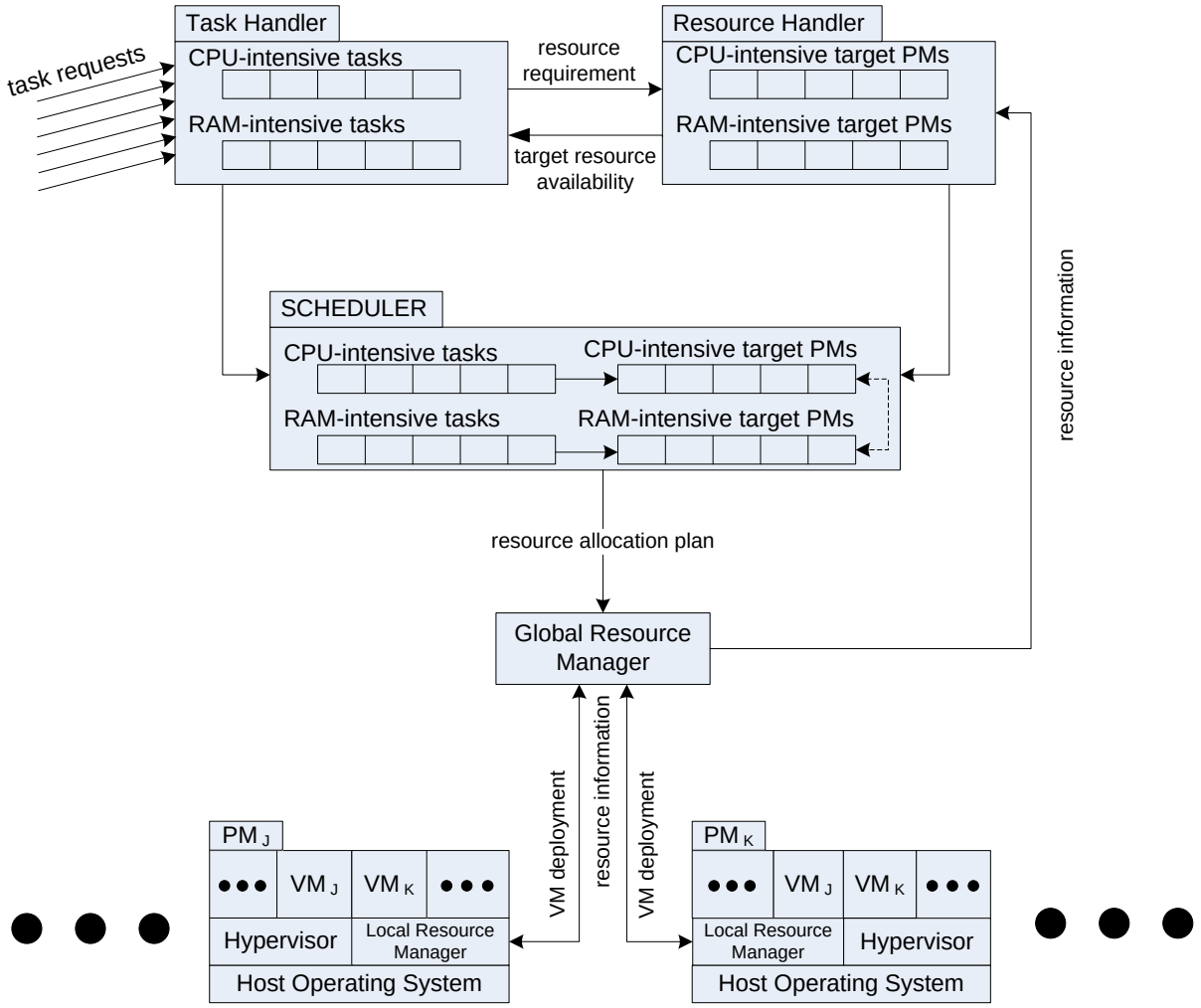

Fig. 1. Architecture of the proposed resource allocation framework

4. Proposed workload and machine categorization based resource allocation

In the proposed resource allocation framework, tasks and target PMs are categorized as CPU-intensive and RAM-intensive to produce a better load balancing effect across active PMs and to utilize their different considered resource capacities in a balanced manner.

Tasks are assumed to arrive in batches following poisson distribution. A target set of PMs is prepared on the basis of tasks' resource requirements. The target PMs and the tasks in the current batch are categorized as CPU-intensive and RAM-intensive on the basis of mean CPU/RAM availability in target PMs. A dedicated VM is allocated to each task in accordance to its resource requirements. The mean CPU and RAM usage in a VM depends on a number of factors [24]–[26] such as memory read/write operations, page faults, cache miss rate, etc. For the sake of simplicity this paper considers the mean resource usage of a VM equal to the mean resource requirement of the assigned task.

The category of a task t_k is deduced as follows:

$$category(t_k) = \begin{cases} CPU - intensive, & if\ CIS_{t_k} \geq RIS_{t_k} \\ RAM - intensive, & if\ CIS_{t_k} < RIS_{t_k} \end{cases} \quad (1)$$

where CIS_{t_k} and RIS_{t_k}, respectively, denotes CPU-intensive score and RAM-intensive score of a task t_k, calculated as follows:

$$CIS_{t_k} = \frac{t_k^{CPU\_requirement}}{\mathcal{H}_{target}^{mean\_CPU\_availability}} \tag{2}$$

$$RIS_{t_k} = \frac{t_k^{RAM\_requirement}}{\mathcal{H}_{target}^{mean\_RAM\_availability}} \tag{3}$$

where $t_k^{CPU\_requirement}$ and $t_k^{RAM\_requirement}$ denotes CPU and RAM requirements, respectively, of task t_k; $\mathcal{H}_{target}^{mean\_CPU\_availability}$ and $\mathcal{H}_{target}^{mean\_RAM\_availability}$, respectively, denotes the average availability of CPU and RAM in the PMs of the target host list, $\mathcal{H}_{target}$. The tasks are ordered in their respective categories in the descending order of their relative affinity score. The relative affinity score of a task t_k is computed as

$$RAS_{t_k} = \left|CIS_{t_k} - RIS_{t_k}\right| \tag{4}$$

where RAS_{t_k} denotes the relative affinity score of the task t_k.

A physical machine, PM_i is categorized as follows:

$$category(PM_i) = \begin{cases} CPU - intensive, & if\ CIS_{PM_i} \geq RIS_{PM_i} \\ RAM - intensive, & if\ CIS_{PM_i} < RIS_{PM_i} \end{cases} \tag{5}$$

where CIS_{PM_i} and RIS_{PM_i}, respectively, denotes CPU-intensive score and RAM-intensive score of PM_i, calculated as follows:

$$CIS_{PM_i} = \frac{PM_i^{CPU\_availability}}{\mathcal{H}_{target}^{mean\_CPU\_availability}} \tag{6}$$

$$RIS_{PM_i} = \frac{PM_i^{RAM\_availability}}{\mathcal{H}_{target}^{mean\_RAM\_availability}} \tag{7}$$

where $PM_i^{CPU-availability}$ and $PM_i^{RAM-availability}$ denotes the availability of CPU and RAM, respectively, in PM_i. The PMs are ordered in their respective categories in the descending order of their relative affinity score. The relative affinity score of a physical machine, PM_i is computed as

$$RAS_{PM_i} = \left|CIS_{PM_i} - RIS_{PM_i}\right| \tag{8}$$

where RAS_{PM_i} denotes the relative affinity score of PM_i.

The PM for deploying a required VM to execute a task is first searched in the related category of target PMs. Further, if a suitable PM is not found in the related category, the search is extended to other category, in reverse order. The relative affinity based resource allocation using categorization and ordering of tasks

and target PMs serve to map the required VMs onto the PMs whose resource availability matches the resource requirements. This serves to minimize the load imbalance across active PMs and utilization imbalance among their considered resource capacities. The stepwise procedure of the proposed resource allocation is presented as Algorithm 1, which in turn uses the proposed workload and machine categorization algorithm given as Algorithm 2.

Algorithm 1. Workload and machine categorization based resource allocation

Input: Batch of tasks, $T = \{t_1, t_2, \ldots, t_k, \ldots, t_{|T|}\}$
Output: VM placement for executing the tasks in T.

1 For each task in T
2 Deduce CPU/RAM requirements.
3 End for
4 Generate target list of PMs.
5 Categorize tasks in T and PMs in $\mathcal{H}_{target}$, as CPU/RAM intensive using Algorithm 2.
6 For all tasks in T do
7 Search a suitable PM in related category target PMs using first-fit approach.
8 Deploy the required VM on the selected PM.
9 End for

Algorithm 2. Workload and machine categorization

Input: Batch of tasks, T and target set of PMs, $\mathcal{H}_{target}$
Output: Categorized (CPU/RAM intensive) tasks/target PMs.

1 For each task in T
2 Categorize tasks using Eqs. (1)-(3).
3 End for
4 Sort tasks in each category using Eq. (4).
5 For each PM in $\mathcal{H}_{target}$
6 Categorize PMs using Eqs. (5)-(7).
7 End for
8 Sort PMs of each category using Eq. (8).

The proposed resource allocation approach comprises of three main phases: categorization of tasks and target PMs, ordering of categorized tasks and target PMs in their respective categories, and mapping of required VMs onto appropriate PMs for resource allocation. The time-complexity of categorizing tasks into the categories of CPU/RAM intensive is $O(n_t)$, where n_t is the number of tasks in a batch of tasks. The time-complexity of categorizing target PMs is $O(n_p)$, where n_p is the number of target PMs. The best and the worst case time-complexities for ordering tasks in their respective categories are $O(n_t)$ and $O(n_t^2)$, respectively, whereas the best and the worst case time-complexities for ordering PMs are $O(n_p)$ and $O(n_p^2)$, respectively. The time-complexity of mapping required VMs onto appropriate PMs for resource allocation is $O(n_t n_p)$.

5. Simulation results and discussion

Simulation experiments are performed using CloudSim simulator [27]. The performance of the proposed workload and machine categorization based approach for the considered resource allocation problem is

compared with First-Fit [28] and a multiqueue based resource allocation framework proposed in [19]. A brief description of the compared approaches is as follows:

- In First-Fit based approach, the list of target PMs is searched from beginning for each task request and the required VM is deployed on the first found appropriate PM.
- In multiqueue based approach, tasks are categorized into the categories of CPU-intensive and RAM-intensive on the basis of their relative requirement for different resources. The PMs in the target host list are arranged in two lists in the ascending order of CPU and RAM utilization. For each task, the related list of target PMs is searched from beginning to deploy the corresponding VM on the first found suitable PM.

In this work, the data center facility is assumed to host 200 multi-core processor PMs of heterogeneous CPU/RAM configurations. A PM can host multiple heterogeneous VMs. A dedicated processor core is assigned to each VM. The different discrete PM and VM configurations considered in this work are given in tables 1 and 2, respectively.

Table 1
PM configurations

CPU processing power (MIPS)	RAM (GB)
2000	2
2500	2.5
3000	3

Table 2
VM configurations

Processing power (MIPS)	RAM (GB)
250	{0.25, 0.5}
500	{0.25, 0.5}
750	{0.5, 0.75}

In this paper, tasks of heterogeneous resource requirements are assumed to arrive in batches following poisson distribution with mean inter arrival time of 100s. The number of tasks in a batch and the length of tasks are assumed to be uniformly distributed in the range [50, 100] and [1×10^5, 2×10^5]MI, respectively. Each task is allotted a dedicated VM for execution.

Simulation experiments are performed to execute the artificially generated set of tasks, comprising of 1×10^4, 2×10^4, and 3×10^4 number of tasks. In this paper, 30 independent simulation runs are performed for different number of tasks, to test the robustness and effectiveness of the proposed approach. The considered resource allocation problem is formulated as a minimization problem. Following metrics are measured in each scheduling cycle of a simulation run to evaluate the performance of the considered resource allocation approaches.

- Mean load imbalance metric, $Mean_{LI}$ measures the mean of load imbalance across active PMs over different scheduling cycles in a simulation run.

$$Mean_{LI} = \frac{\sum_{k=1}^{n_s} LI_k}{n_s} \tag{9}$$

where LI_k denotes the load imbalance across active PMs in the *k*-th scheduling cycle and n_s denotes the total number of scheduling cycles in a simulation run. The load imbalance, *LI* across active PMs in a scheduling cycle is measured as follows:

$$LI = \frac{1}{n_a} \times \left(\delta_1 \times \sum_{\forall active\ PMs} \left| CPU_{APM_i} - meanCPU_{active\ PMs} \right| + \delta_2 \times \sum_{\forall active\ PMs} \left| RAM_{APM_i} - meanRAM_{active\ PMs} \right| \right) \quad (10)$$

where n_a denotes the number of active PMs in a scheduling cycle; CPU_{APM_i} and RAM_{APM_i} denotes the percent utilization of CPU and RAM, respectively, in an *i*-th active PM, APM_i; $meanCPU_{activePMs}$ and $meanRAM_{activePMs}$ denotes the percent mean utilization of CPU and RAM, respectively, in active PMs; δ_1 and δ_2 are the weightage constants such that $\delta_1 + \delta_2 = 1$. In this work, $\delta_1 = \delta_2$, to give an equal weightage for CPU and RAM.

- Mean resource capacity imbalance metric, $Mean_{RCI}$ measures the mean of utilization imbalance among considered resource capacities in active PMs over different scheduling cycles in a simulation run.

$$Mean_{RCI} = \frac{\sum_{k=1}^{n_s} RCI_k}{n_s} \quad (11)$$

where RCI_k denotes the resource capacity utilization imbalance in active PMs in the *k*-th scheduling cycle. The resource capacity imbalance, *RCI* is measured as follows:

$$RCI = \frac{\sum_{\forall active\ PMs} \left| CPU_{APM_i} - RAM_{APM_i} \right|}{n_a} \quad (12)$$

5.1 Load imbalance across active physical machines

The load imbalance across active PMs in a simulation run is measured using $Mean_{LI}$ metric. The average of $Mean_{LI}$ metric over 30 independent simulation runs corresponding to the proposed and other compared resource allocation approaches is presented graphically in Fig. 2, and the corresponding numerical values of $Mean_{LI}$ are given in table 3. Results show that the proposed workload and machine categorization based resource allocation approach is comparatively more effective and reliable in balancing the load across active PMs. The proposed approach has a relative mean improvement of 10.68% and 12.22% over first-fit and multiqueue-based approaches, respectively.

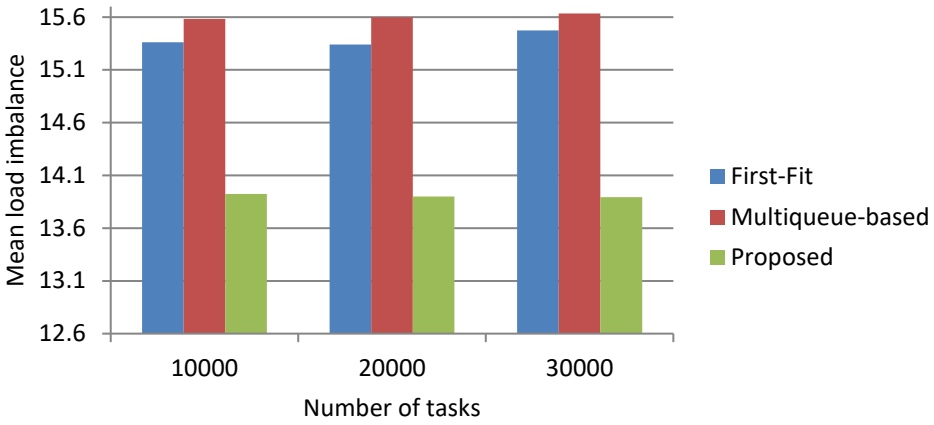

Fig. 2. Mean load imbalance corresponding to different resource allocation approaches

Table 3
Mean$_{LI}$ comparison of different resource allocation approaches

Number of tasks	Proposed	First-Fit	Multiqueue-based
10000	**13.925**	15.362	15.582
20000	**13.901**	15.340	15.599
30000	**13.893**	15.473	15.636

The box plot of *Mean*$_{LI}$ metric for 30 independent simulation runs corresponding to different algorithms is presented in Fig. 3. The box plot clearly shows that the proposed approach achieves fittest best and least worst value among compared approaches in executing different number of tasks. This shows the relative potential of the proposed resource allocation approach. The best and worst values of *Mean*$_{LI}$ over 30 independent simulation runs for different number of tasks are given in tables 4 and 5, respectively.

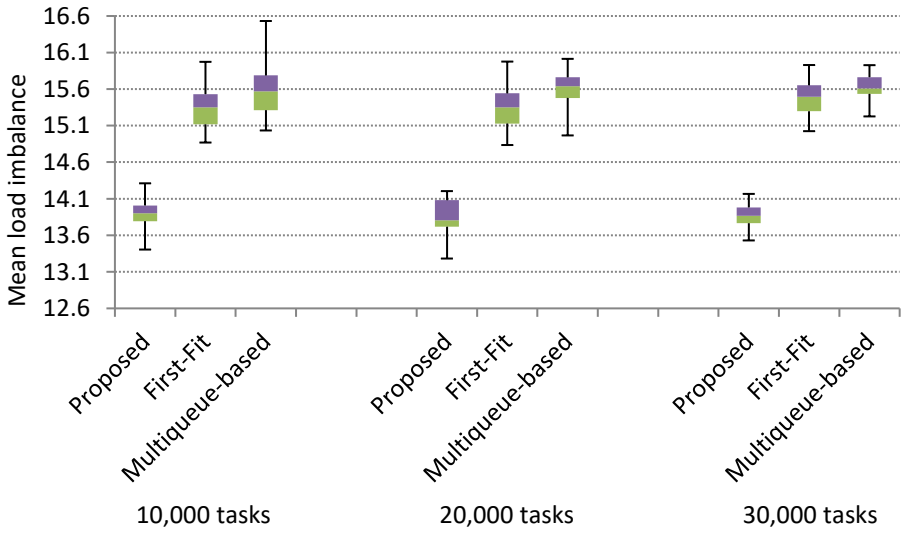

Fig. 3. Box plot of *Mean*$_{LI}$ for 30 simulation runs

Table 4
Mean$_{LI}$ best values over 30 simulation runs

Number of tasks	Proposed	First-Fit	Multiqueue-based
10000	**13.405**	14.869	15.034
20000	**13.282**	14.835	14.966
30000	**13.530**	15.025	15.226

Table 5
Mean$_{LI}$ worst values over 30 simulation runs

Number of tasks	Proposed	First-Fit	Multiqueue-based
10000	**14.311**	15.973	16.532
20000	**14.204**	15.976	16.014
30000	**14.166**	15.929	15.928

5.2 Mean resource capacity utilization imbalance

The mean of utilization imbalance among different resource capacities in active PMs is measured using *Mean*$_{RCI}$ metric. The average of *Mean*$_{RCI}$ metric over 30 independent simulation runs corresponding to the proposed and other compared resource allocation approaches is presented graphically in Fig. 4, and the corresponding numerical values of *Mean*$_{RCI}$ are given in table 6. Results show that the proposed workload and machine categorization based resource allocation approach is comparatively more effective and reliable in balancing the utilization of different resource capacities in active PMs. The proposed approach has a relative mean improvement of 4.83% and 1.56% over first-fit and multiqueue-based approaches, respectively.

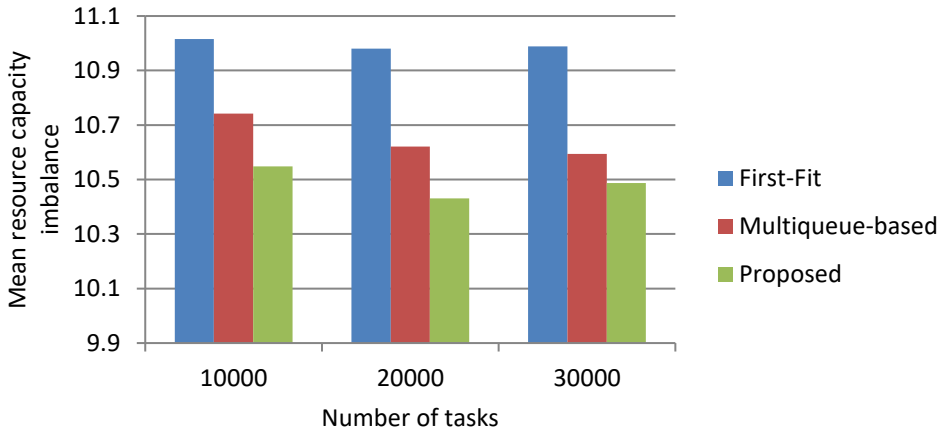

Fig. 4. Mean resource capacity imbalance corresponding to different resource allocation approaches

Table 6
Mean$_{RCI}$ comparison of different resource allocation approaches

Number of tasks	Proposed	First-Fit	Multiqueue-based
10000	**10.548**	11.016	10.742
20000	**10.431**	10.981	10.621
30000	**10.488**	10.988	10.594

The box plot of $Mean_{RCI}$ metric for 30 independent simulation runs corresponding to different algorithms is presented in Fig. 5. The box plot clearly shows that the proposed approach achieves the fittest best and least worst value among the compared approaches in executing different number of tasks. This shows the relative potential of the proposed resource allocation approach. The best and worst values of $Mean_{RCI}$ over 30 independent simulation runs for different number of tasks are given in tables 7 and 8, respectively.

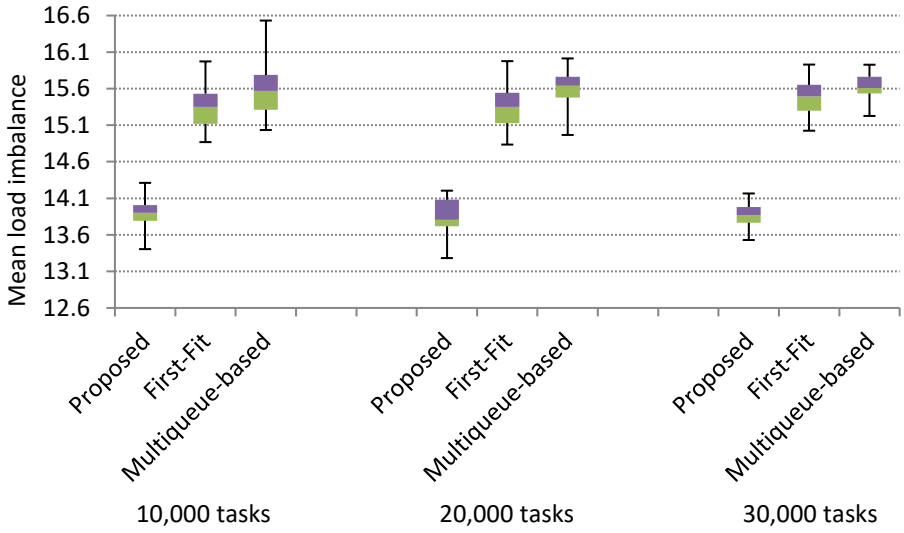

Fig. 5. Box plot of $Mean_{RCI}$ for 30 simulation runs

Table 7
$Mean_{RCI}$ best values over 30 simulation runs

Number of tasks	Proposed	First-Fit	Multiqueue-based
10000	**10.024**	10.521	10.144
20000	**10.003**	10.374	10.143
30000	**9.976**	10.533	10.088

Table 8
$Mean_{RCI}$ worst values over 30 simulation runs

Number of tasks	Proposed	First-Fit	Multiqueue-based
10000	**10.983**	11.570	11.163
20000	**11.046**	11.404	11.207
30000	**10.713**	11.525	11.097

5.3 Statistical analysis

In this section, statistical analysis is performed to deduce the relative efficacy of the proposed resource allocation approach over compared approaches. Friedman test is performed to compute the relative ranks of considered approaches with respect to $Mean_{LI}$ and $Mean_{RCI}$ metrics. The relative ranks given in table 9 demonstrate the relative superiority of proposed approach over other considered approaches in balancing load across active PMs and utilizing their resource capacities in a balanced manner.

Table 9
Relative ranks of considered approaches in Friedman test

Number of tasks	Metrics	Proposed	First-Fit	Multiqueue-based
10000	$Mean_{LI}$	**1.00**	2.27	2.73
	$Mean_{RCI}$	**1.53**	2.53	1.93
20000	$Mean_{LI}$	**1.00**	2.20	2.80
	$Mean_{RCI}$	**1.30**	2.87	1.83
30000	$Mean_{LI}$	**1.00**	2.33	2.67
	$Mean_{RCI}$	**1.17**	2.90	1.93

The non-parametric hypothesis tests: Wilcoxon sign rank test and *t*-test, are performed to check whether the relative improvement is significant or not, at a significance level of 0.05. The null hypothesis, H_0 is: 'The difference between the results obtained by the given two approaches is not significant', and the alternate hypothesis, H_a is: 'The difference between the results is significant'. The *p*-values of Wilcoxon sign rank test and *t*-test with respect to $Mean_{LI}$ and $Mean_{RCI}$ metrics are given in tables 10 and 11, respectively.

Table 10
p-values in Wilcoxon sign rank test corresponding to $Mean_{LI}$ and $Mean_{RCI}$ (Proposed versus other compared approaches)

Number of tasks	Metrics	First-Fit	Multiqueue-based
10000	$Mean_{LI}$	3.02E-11	3.02E-11
	$Mean_{RCI}$	1.71E-07	0.01
20000	$Mean_{LI}$	1.69E-17	1.69E-17
	$Mean_{RCI}$	1.29E-09	3.00E-04
30000	$Mean_{LI}$	1.69E-17	1.69E-17
	$Mean_{RCI}$	1.69E-10	3.00E-04

Table 11
p-values in *t*-test corresponding to $Mean_{LI}$ and $Mean_{RCI}$ (Proposed versus other compared approaches)

Number of tasks	Metrics	First-Fit	Multiqueue-based
10000	$Mean_{LI}$	1.00E-05	1.00E-05
	$Mean_{RCI}$	1.00E-05	7.60E-03
20000	$Mean_{LI}$	1.00E-05	1.00E-05
	$Mean_{RCI}$	1.00E-05	5.83E-04
30000	$Mean_{LI}$	1.00E-05	1.00E-05
	$Mean_{RCI}$	1.00E-05	9.40E-05

The *p*-values given in tables 10 and 11, are all less than the considered significance level of 0.05. Hence, the null hypothesis, H_0 is rejected and the alternate hypothesis, H_a is accepted corresponding to each compared approach. This shows that the results obtained by the proposed approach are significantly better than obtained by the compared approaches.

6. Conclusion

This paper proposes a resource allocation framework to proactively balance the load across PMs and utilize their considered resource capacities in a balanced manner. The categorization and ordering of workload and machines is used to map workload onto appropriate PMs. The results of considered performance evaluation

metrics show that the proposed workload and machine categorization based resource allocation leads to comparatively better load balancing across active PMs and balanced utilization of their resource capacities. The statistical analysis also validates the relative superiority of the proposed approach.

As future work, we plan to extend this work for other cloud computing problems, e.g., energy conservation, resource utilization, and cost optimization.

Acknowledgements

We are extremely thankful to the editor and all the anonymous reviewers for their precious time and valuable suggestions. We acknowledge the Ph.D. institute fellowship received from Sant Longowal Institute of Engineering and Technology, India to carry out the research work.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms : Vision , hype , and reality for delivering computing as the 5th utility," *Futur. Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," Gaithersburg, MD, USA, 2011.

[3] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing — The business perspective," *Decis. Support Syst.*, vol. 51, no. 1, pp. 176–189, 2011.

[4] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Netw. Comput. Appl.*, vol. 66, pp. 106–127, 2016.

[5] M. A. Alworafi and S. Mallappa, "An Enhanced Task Scheduling in Cloud Computing Based on Deadline-Aware Model," *Int. J. Grid High Perform. Comput.*, vol. 10, no. 1, pp. 31–53, 2018.

[6] N. Garg, D. Singh, and M. Singh, "Energy and resource efficient workflow scheduling in a virtualized cloud environment," *Cluster Comput.*, vol. 24, no. 2, pp. 767–797, 2021.

[7] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud environments : Systematic literature review and future trends," *J. Netw. Comput. Appl.*, vol. 71, pp. 86–98, 2016.

[8] A. Thakur and M. S. Goraya, "A Taxonomic Survey on Load Balancing in Cloud," *J. Netw. Comput. Appl.*, vol. 98, pp. 43–57, 2017.

[9] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurr. Comput. Pract. Exp.*, vol. 29, no. 12, pp. 1–16, 2017.

[10] P. Kumar and R. Kumar, "Issues and Challenges of Load Balancing Techniques in Cloud Computing : A Survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–35, 2019.

[11] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment : A review," *J. King Saud Univ. - Comput. Inf. Sci.*, 2021.

[12] A. Thakur and M. S. Goraya, "RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment," *Simul. Model. Pract. Theory*, vol. 116, p. 102485, 2022.

[13] F. Ramezani, J. Lu, J. Taheri, and A. Y. Zomaya, "A Multi-Objective Load Balancing System for Cloud Environments," *Comput. J.*, vol. 60, no. 9, pp. 1316–1337, 2017.

[14] M. Kumar, K. Dubey, and S. C. Sharma, "Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing," *Procedia Comput. Sci.*, vol. 125, pp. 717–724, 2018.

[15] M. Kumar and S. C. Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment," *Comput. Electr. Eng.*, vol. 69, pp. 395–411, 2018.

[16] H. Shen and L. Chen, "A Resource Usage Intensity Aware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 17–31, 2020.

[17] V. R. Chandakanna and V. K. Vatsavayi, "A sliding window based Self-Learning and Adaptive Load Balancer," *J. Netw. Comput. Appl.*, vol. 56, pp. 188–205, 2015.

[18] M. Sheikhalishahi, R. M. Wallace, L. Grandinetti, J. L. Vazquez-poletti, and F. Guerriero, "A multi-dimensional job scheduling," *Futur. Gener. Comput. Syst.*, vol. 54, pp. 123–131, 2016.
[19] L. Zuo, S. Dong, L. Shu, C. Zhu, and G. Han, "A Multiqueue Interlacing Peak Scheduling Method Based on Tasks' Classification in Cloud Computing," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1518–1530, 2016.
[20] W. Hashem, H. Nashaat, and R. Rizk, "Honey Bee Based Load Balancing in Cloud Computing," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 12, pp. 5694–5711, 2017.
[21] F. Ebadifard and S. M. Babamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment," *Cluster Comput.*, vol. 24, no. 2, pp. 1075–1101, 2021.
[22] Z. Xiao, W. Song, and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, 2013.
[23] R. K. Gupta and R. K. Pateriya, "Balance Resource Utilization (BRU) Approach for the Dynamic Load Balancing in Cloud Environment by Using AR Prediction Model," *J. Organ. End User Comput.*, vol. 29, no. 4, pp. 24–50, 2017.
[24] P. Findeisen, "DETERMINING PROCESSOR USAGE BY A THREAD," US 7426731 B2, 2008.
[25] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and Dynamicity of Clouds at Scale : Google Trace Analysis," in *SoCC '12: Proceedings of the Third ACM Symposium on Cloud Computing*, 2012, pp. 1–13.
[26] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2647–2660, 2014.
[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, A. F. D. R. César, and R. Buyya, "CloudSim : A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. - Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.
[28] P. Silva, C. Perez, and F. Desprez, "Efficient Heuristics for Placing Large-Scale Distributed Applications on Multiple Clouds," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2016, pp. 483–492.