Cluster Computing (2021) 24:1955–1974
https://doi.org/10.1007/s10586-021-03232-4

OPSA: an optimized prediction based scheduling approach for scientific applications in cloud environment

Gurleen Kaur[1] · Anju Bala[1]

Received: 25 June 2019 / Revised: 5 December 2020 / Accepted: 4 January 2021 / Published online: 27 January 2021

Abstract

Cloud computing has attracted scientists to deploy scientific applications by offering services such as Infrastructure-as-a-service (IaaS), Software-as-a-service (SaaS), and Platform-as-a-Service (PaaS). The research community is able to get access to resources on-demand within a short period of time. But, as the demand for cloud resources is dynamic in nature, this affects resource availability during scheduling. Hence, there is a need for efficient management of resources so that tasks can be scheduled based on their execution requirements. To provide a solution, a resource prediction based scheduling approach has been introduced in this paper which automates the resource allocation for scientific applications in a virtualized cloud environment. This research work focuses on the design of an optimized prediction based scheduling approach which maps the tasks of scientific application with the optimal VM by combining the features of swarm intelligence and TOPSIS. The proposed approach minimizes the execution time, cost, and SLA violation rate in comparison to existing scheduling heuristics.

Keywords Resource prediction · Resource scheduling · Cloud environment · Virtual machine · Ensembling · Machine learning · Quality of service

1 Introduction

Cloud computing offers unlimited resources to its users in the form of services like Infrastructure-as-a-service (IaaS), Software-as-a-service (SaaS), and Platform-as-a-service (PaaS). Virtualization is a key process in cloud computing that segregates the resources of a physical machine (PM) to create more than one execution environment and enable the concept of multi-tenancy. These peculiar characteristics of the cloud environment lead to certain major challenges such as load-balancing, fault-tolerance, and scheduling. Task scheduling (TS) is a multi-objective NP hard optimization problem whose objective is to achieve successful mapping between tasks and VMs by minimizing the execution time, cost, and service level agreement (SLA) violations between cloud user and provider.

✉ Gurleen Kaur
gurleen.kaur@thapar.edu

[1] Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala 147003, India

Virtualized cloud systems have become popular in hosting complex scientific applications such as montage, cybershake, inspiral, sipht, etc. Cloud clients deploy the applications onto the VMs in the cloud with dedicated resource requirements for performance guarantee, which is specified in terms of Service Level Agreement (SLA). The workload of VMs varies all the time and some may exhibit weekly or seasonal variability. To guarantee good performance at periods of peak demand, VMs processing capacity is often over-provisioned. This leads to poor scheduling and cloud providers are unable to exploit the benefits out of cloud services. Thus, it is still a challenging problem for cloud providers to schedule the virtualized resource adaptively in order to handle variable workloads without SLA violation. The significant prediction of resource usage is essential to achieve optimal resource scheduling for cloud computing [1]. Hence, prediction based scheduling is the motivation behind this research work.

This research work focuses on the design of an Optimized Prediction based Scheduling Approach (OPSA) which maps the tasks of scientific application with the optimal VM. The existing approaches have not focused on predicting the set of resources required for executing a scientific application and simultaneously scheduling the resources based on the prediction in an optimized manner. The proposed work takes the predicted set of resources as input and schedules these scientific applications by the efficient utilization of resources while simultaneously reducing the SLA violations at runtime, thereby achieving cost-effectiveness and desired performance. To handle the problem of multi-objective task scheduling, a multi-criteria decision making algorithm ''TOPSIS" is incorporated. TOPSIS is a mathematical multi-criteria decision based algorithm which supports the working of swarm intelligence algorithm by finding local optimum solutions. Here, TOPSIS is used to compute the fitness value of tasks which is further utilized by swarm intelligence algorithm to schedule the tasks of scientific application.

2 Related work

Various researchers have proposed prediction and scheduling techniques but to perform prediction based scheduling is still a challenging problem. Prediction is basically used to enhance the effectiveness of scheduling algorithms. Zhiping et al. [2] have proposed a Deep-Q network based online resource scheduling framework that combines the capabilities of prediction and scheduling. The authors have used Google Cluster Usage Traces for conducting the experiments. The focus of paper is on two optimization objectives namely task makespan and energy consumption. Bo et al. [3] in their survey paper has stated that cloud is a cost-efficient way to address the issue of resource scarcity for meeting the peak demands of its users. Resource utilization, cost, SLA violation are few major challenges which researchers need to address. Fatemah and Seyed [4] proposed an autonomic task scheduling algorithm for executing the dynamic workloads in a cloud environment. They used the combination of prediction technique ANFIS and autonomous load balancing technique for minimizing the response time and maximizing the utilization of available cloud resources. Mahmood, and Kamran [5] introduced BCFramework with focus a on provisioning and scheduling of public cloud resources for big data. The proposed approach minimizes the average tuple latency and public cloud resource utilization cost but it does not consider a prediction model for handling big data fluctuations. Haion et al. [6] proposed a multi-prediction based scheduling approach which reduces the task failures and increases the resource utilization for hybrid workload in the cloud data center.

Table 1 summarizes the existing resource prediction based scheduling approaches. This table presents the prediction and scheduling techniques used, problem solved, computational requirements, outcomes, and forthcoming challenges. From the literature review, it can be inferred that a lot of work has been done towards prediction based scheduling of resources but none of the approaches is application specific. Moreover, there is need to improve the execution time, cost, and SLA violations as mentioned in the challenges of the surveyed research work. Therefore, there is a lot of scope to apply these approaches specifically for scientific applications in cloud environment and enhance the performance in terms of execution time, cost and SLA violations.

2.1 The main contributions of this research work are

- In this proposed research work (OPSA), the predicted CPU & memory usages have been taken as input.
- The application resource requirements for execution have been compared against the availability of resources on VMs.
- Applications are mapped to suitable VMs.
- The tasks of the mapped application were scheduled using a combination of swarm intelligence and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS).
- TOPSIS, a multi-criteria decision making algorithm, has been used to compute the fitness value of the tasks in the swarm algorithm to optimize the scheduling of resources.
- The fitness values calculated by TOPSIS have been assigned as local best (*pbest*) values of tasks.
- The task with the highest fitness value has been considered as global best (*gbest*) and it has been assigned to VM for execution and the process has been repeated until all the tasks were scheduled.

The key objectives of the proposed approach have been the reduction of execution time, cost, and SLA violation rate. The proposed approach has been validated by comparing the results with existing scheduling heuristics.

2.2 Problem formulation

The objective of the task scheduling algorithm in this research work is to solve the problem of scheduling n tasks of a scientific application on a set of m heterogeneous VMs to attain certain goals such as minimizing total execution time, minimizing cost, and reducing SLA violations. So, there is a need for an efficient scheduling algorithm which can take into consideration multiple objectives. Many parameters have been initialized as shown in Table 2 and various terms used in problem formulation are given in Table 3.

The following objective problems are taken into account for developing an optimal scheduling algorithm.

Table 1 Existing resource prediction based scheduling techniques

Author	Description	Prediction technique	Scheduling Technique	Platform	Tool	Computational requirements	Outcomes	Challenges
Zhiping et al. [2]	Deep Q-network based online resources	DQN algorithm	Deep reinforcement learning	Simulation	Python environment with tensorflow	Ubantu 16.04 OS, Intel Core i7-7800X processor, 16G, 3.50 GHz memory	Task makespan is improved and energy consumption is reduced	How to effectively represent cloud computing model? Necessary to design multi-learning model Effective load forecasting will help in better cloud resource man-agement and task scheduling strategy
Fatemah and Seyed [4]	Predicts the future load on VMs and balances the load using autonomous load balancing technique	ANFIS	Dynamic scheduling	Cloud Simulator	CloudSim	–	Response time is reduced and use of available cloud resources is increased	Multi-objective scheduling
Mahmood, and Kamran [5]	Provisioning and scheduling of public cloud resource for big data	–	Deadline aware scheduling	Simulator and Cloud	Storm and Amazon ECE2	XWindows 7 64bit, JDK1.8, CEPSim simulator, Intel Dual Core CPU 3.0 GHz 64 bit with 4 GB RAM	Average tuple latency and cloud resource utilization cost is minimized	Big data fluctuation prediction model need to be considered
Haion et al. [6]	Multi-Prediction-based Scheduling for hybrid work- loads in cloud environment	ARMA pre-diction model and feedback based online AR model	MPHW Scheduling	Cloud Simulator	CloudSim	–	Task failure rate is reduced and resource utilization rate is increased	Dynamic scheduling need to be incorporated Live migration
Micha et al. [7]	Predicts utilization of cloud resource on a per-task and per-resource level using prediction models	Artificial Neural Network	Task scheduling	Cloud	Open-source Java library Deeplearn- ing4	–	Increases prediction accuracy. Lacks renormalization	To refine the applied prediction techniques Nonsupervised techniques should be used
Jiangtian et al. [8]	ANN	Artificial Neural Networks (ANNs)	Modified Critical Path (MCP) Scheduling algorithm	Cloud	R prediction tool and DAG scheduler	Opt cluster, which has 16 2-way SMP nodes. Each node has a dual-core AMD	High prediction accuracy is achieved. Improvedruntime performance	Examine runtime prediction errors Investigate disjoint parameter spaces

Table 1 (continued)

Author	Description	Prediction technique	Scheduling Technique	Platform	Tool	Computational requirements	Outcomes	Challenges
Kang et al [9]	Dynamic Scheduling Strategy (DSS) integrates the Divisible Load Theory and node availability prediction technique	Linear regression	Dynamic Scheduling Strategy	Cloud	Cloud Simulator	Intel(X) Xeon(R) Processor E5-2640 2.5 GHz, 24 GB RAM	Reduces total execution time by 44.60%	Require prediction based scheduling strategy for handling BigData applications
Ghobaei-Arani et al. [10]	Hybrid resource provisioning approach for cloud services	Linear regression model	Time-shared scheduling	Cloud	CloudSim	Four VMs with core 1–8, RAM 2 GB-15 GB, Storage 160 GB-1690 GB	Increases the resource utilization and decreases the total cost, while avoiding SLA violations	To integrate the proposed approach with admission control strategies for multi-tier cloud applications
Choudhary. et al. [11]	Hybridization of Gravitational Search Algorithm (GSA) and Heterogeneous Earliest Finish Time (HEFT) to schedule workow applications	Gravitational Search Algorithm (GSA)	Heterogeneous Earliest Finish Time (HEFT)	Cloud	WorkflowSim	Intel(R) Core(TM) i5-2540 M CPU with 2.60 GHz and 4 GB RAM	Reduces makespan and he cost ofexecution	To deploy complex applications and check for VM failure
Zhang et al. [12]	A resource crowd-funding model to provide resource for cloud services	Genetic algorithm	Task based scheduling	Cloud Simulator	CloudSim	–	Increases the stability of task execution and reduce power consumption	To encourage idle resource to join the resource pool

Table 2 Initialization parameters

Parameter	Description
ET_n	Execution time of the jobs running in VMs on the nth node
ET_{nmk}	Execution time for k jobs running on m VMs on the nth node
EC	Execution cost of a job on a VM on the nth node
$SLAV$	Service Level Agreement Violation
$previousRequested$	Total amount of CPU MIPS and memory bytes requested by a job for execution
$previousAllocated$	Total amount of CPU MIPS and memory bytes already allocated to a VM to process a job
ACU_n	Average CPU utilization for nth node
JCT_{nmk}	CPU utilization of k jobs running on m VMs on the nth node
TCU_n	Total CPU utilization for the nth node
AMU_n	Average memory utilization for nth node
JMU_{nmk}	Memory utilization of k jobs running on m VMs on the nth node
TMU_n	Total memory utilization for the nth node

Table 3 Description of various terms

Terms used	Description
VM	Virtual Machine is an emulation of a computer system
Node	A computer system where VMs are running
Job	Jobs run in VM, which are created dynamically according to the job's requirement. Jobs need to be allocated across the node pool

2.2.1 Total execution time

The time taken by a job to execute on a particular VM is known as execution time [13]. ET_n is the execution time of the jobs running in VMs on the nth node and is defined as Eq. 1:

$$ET_n = \sum_{m=1}^{v_n} \sum_{k=1}^{j_n} ET_{nmk} \tag{1}$$

where ET_{nmk} is the execution time for k jobs running on m VMs on the nth node. Hence, the total execution time (ET) is Eq. 2:

$$ET = \sum_{n=1}^{M} ET_n \tag{2}$$

where M is the total number of nodes.

2.2.2 Total execution cost

The cost of executing a job on a VM is computed by Eq. 3:

$$EC = Processing\, cost\, per\, second * ET_n \tag{3}$$

2.2.3 SLA violation (SLAV)

The end users state the QoS requirements to the Cloud Service Providers (CSPs) in the form of Service Level Agreements (SLAs) [14]. It is the responsibility of the CSPs to make sure that an appropriate amount of resources are allocated to an application in order to fulfill the users' demands and minimize the SLA Violations (SLAV). The formula to compute SLAV is given in Eq. 4:

$$SLAV = \frac{prev\, Requested - prev\, Allocated}{prev\, Requested} \tag{4}$$

here, $prev\, Requested$ is the total amount of CPU MIPS and memory bytes requested/required by a job for execution. This is computed using the ensemble algorithm (Algorithm 1) which provides the predicted set of resources (CPU and Memory). $prevAllocated$ is the total amount of CPU MIPS and memory bytes allocated for the execution. These two parameters in the practical scenario are obtained using the cloudsim classes "getAvailableMIPS()" and "getCurrentsize()". The total amount of available CPU MIPS and available memory is recorded and allocated using the proposed prediction based scheduling algorithm (Algorithm 2).

2.2.4 Average CPU utilization

At any given time, for nth node, the CPU utilization ACU_n can be given as Eq. 5:

$$ACU_n = \sum_{m=1}^{v_n} \sum_{k=1}^{j_n} JCT_{nmk} \tag{5}$$

where v_n is the number of VMs running on the nth node and j_n is the number of jobs assigned to v_n VMs. JCT_{nmk} is the CPU utilization of k jobs running on m VMs on the nth node. The CPU utilization in percentage is calculated as Eq. 6:

$$ACU_n(\%age) = \frac{\sum_{m=1}^{v_n} \sum_{k=1}^{j_n} JCT_{nmk}}{TCU_n} * 100 \tag{6}$$

where,

$$Total\, CPU\, Utilization(TCU_n) = \frac{Clock\, Cycles\, per\, Instruction * Instruction\, Count}{Clock\, Rate}$$

2.2.5 Average memory utilization

At any given time, for nth node, the memory utilization AMU_n can be given as Eq. 7:

$$AMU_n = \sum_{m=1}^{v_n} \sum_{k=1}^{j_n} JMU_{nmk} \tag{7}$$

where v_n is the number of VMs running on the nth node and j_n is the number of jobs assigned to v_n VMs. JMU_{nmk} is the memory utilization of k jobs running on m VMs on the nth node. The memory utilization in percentage is calculated as Eq. 8:

$$AMU_n(\%age) = \frac{\sum_{m=1}^{v_n} \sum_{k=1}^{j_n} JMU_{nmk}}{TMU_n} * 100 \tag{8}$$

where TMU_n is the total memory utilization for the nth node.

2.3 Fitness value formulation

In this research work, the problem formulation for minimizing objective criterion mentioned in Eqs. 1–3 can be optimized. The Relative Closeness Score (RCS) for each task is calculated using a multi-criteria decision making algorithm, TOPSIS [15, 16]. This algorithm will enhance the proficiency of task scheduling by supporting multiple objectives. The RCS value computed by TOPSIS is taken as Fitness Value (FV) of the tasks for the proposed scheduling algorithm.

FV_{t1}	$= RCS_{t1}$
FV_{t2}	$= RCS_{t2}$
–	–
–	–
–	–
FV_{ti}	$= RCS_{ti}$

where RCS of tasks obtained using TOPSIS is $RCS_t = RCS_{t1}, RCS_{t2}, \ldots, RCS_{ti}$ which corresponds to the FV of tasks $FV_t = FV_{t1}, FV_{t2}, \ldots, FV_{ti}$, respectively. Fitness Value (FV) is unique to problems and is used to calculate particle efficiency. The search space represents the number of particles in the population. Particles are randomly initialized and each particle has a FV obtained using TOPSIS. The best results (i.e. FV) obtained so far by the particle is the *pbest* of a particle, while *gbest* is FV of the best particle in the search space. The TOPSIS algorithm for computing RCS of tasks is elaborated in the next section.

3 Resource prediction based scheduling framework

To achieve highly effective computations and the best Quality of Service (QoS) of the cloud, it is vital to perform the scheduling of tasks in an efficient manner. The mapping of the submitted applications and VMs are considered to be successful if attained minimum execution time, cost, SLA violations, and maximum utilization of resources has been attained. To solve the problem of multi-objective task scheduling an Optimized Prediction based Scheduling Approach (OPSA) has been proposed in this paper.

This section details the framework of the proposed RPS technique as portrayed in Fig. 1. This framework contains three modules: deployment, prediction, and scheduler which are explained further.

In the deployment module, the scientific application is executed on the WorkflowSim (a cloud simulator for scientific applications) [17]. Here, "Cybershake" and "Floodplain" are used as two different case studies in this research work. The scientific applications considered here are based on workflows and every workflow has a different number of execution levels. When level 1 is completed, the execution of level 2 is started, and so on. Therefore, each workflow is divided into execution levels. The tasks at every level have different execution requirements and those execution requirements are gathered using Microsoft Azure cloud instances and Algorithm 1 is used for future resource usage prediction.

Once the application is deployed, a resource usage dataset is generated and passed onto the prediction module for further processing. The proposed method used the simulation programming to the scientific application being

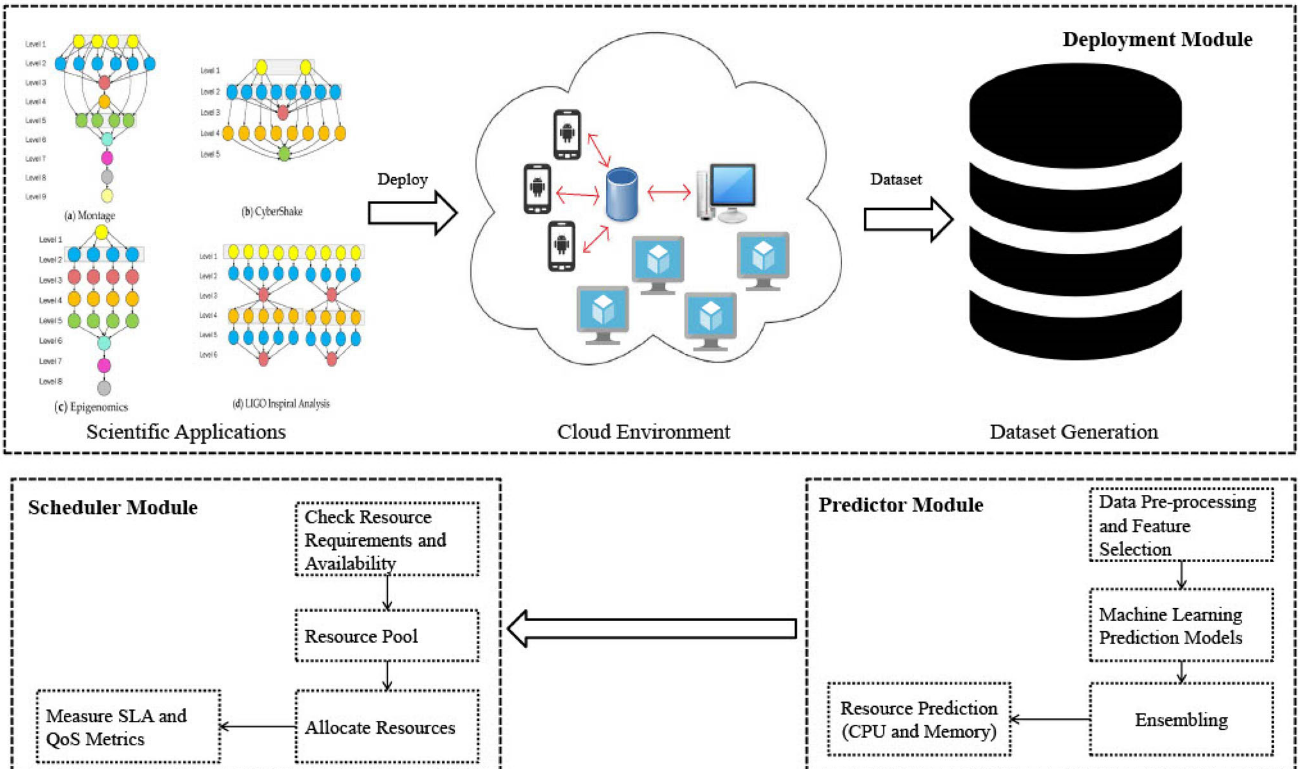

Fig. 1 Proposed RPS framework

deployed on the cloud and corresponding resource usage data from the Microsoft Azure Cloud instances rented in real time, then extracts data features to form the sample dataset and finally makes the resource usage prediction. The prediction module performs the preprocessing of data so that there are no null values and converts the alphabetic values to numeric for smooth processing. It also selects the relevant features using a Genetic Algorithm, a meta-heuristic feature selection approach. Finally, this module predicts the usage of resources by implementing an ensemble algorithm [18] as proposed in our previous research work.

3.1 Machine learning models

Machine learning is a process of modeling and analyzing learning processes that enhance system efficiency and improve the performance of an application. Machine learning can be categorized as supervised, unsupervised, and reinforcement learning. Numerous applications of machine learning involve only those tasks that can be employed as supervised. It enables the systems to learn from data, identify the hidden patterns, and make decisions with the least human interference.

- Bayesian ridge regression: Bayesian ridge regression (BRR) [19] model comprises special cases like t-test and anova. This model was intended to fit parametric regression models utilizing distinctive kinds of shrinkage techniques. BRR is formulated as depicted in Eq. 9:

$$Y = XB + e \tag{9}$$

here, Y is the dependent variable, X is the independent Variable, B is the regression coefficient, e is the Error. B is calculated as: $B_{OLS} = (X^T\ X) - 1\ X^T\ Y$ [OLS: Ordinary Least Squares] where $X^TX = R$ and R is a Correlation Matrix.

- Bayesian regularized neural network: The need for lengthy cross-validation is eliminated or reduced by Bayesian regularized neural networks (BRNN) [20, 21] as they are better than standard back-propagation nets. In this mathematical method of Bayesian regularization, non-linear regression is converted to a "well-posed" statistical problem in the manner of ridge regression. The formula to compute BRNN is shown in Eq. 10:

$$Y_i = g(X_i) + e_i = \sum_{k=1}^{s} w_k g_k \left(b_k + \sum_{j=1}^{p} X_{ij} \beta_j^{[k]} \right) + e_i; \quad i = 1, \ldots, n \tag{10}$$

here, $e_i \sim N(0,\sigma_e^2)$, s is number of neurons, w_k is weight of the kth neuron, k = 1,…….,s, b_k is a bias for the kth neuron, k = 1,……,s, $\beta_j^{[k]}$ is the weight of the jth input to the net, j = 1,……,p, $g_k(.)$ is the activation function, $g_k(x) = (\exp(2x) - 1)/(\exp(2x) + 1)$, The software will minimize $F = \beta E_D + \alpha E_w$ where$E_D = \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2$. Error sum of squares, E_w is the sum of squares of network parameters (weights and biases), $= \beta = 1/(2\sigma_e^2, \alpha = 1/(2\sigma_\theta^2)$, σ_θ^2 is a dispersion parameter for weights and biases.

- Neural network: Neural networks (NN) are distinguished under various types such as artificial neural network, recurrent neural network, recursive neural network, and so on. These are statistical learning models that deal with neurons similar to biological neural networks [20]. These neurons are interconnected to each other which exchange messages and the values are calculated using supervised or unsupervised learning. The connections within the network can be systematically adjusted based on the inputs and outputs and we can process them using various propagation techniques. NN is formulated as shown in Eq. 11:

$$P_j(t) = \sum\nolimits_i^n O_i(t) w_{ij} \tag{11}$$

here, $P_j(t)$ is input to the neuron j, $O_i(t)$ is output of the predecessor neurons (used to calculate P_j), i is the number of neurons in a level, w is the assigned weight and t is the value of the neuron.

- Support vector machine: Support Vector Machine (SVM) [22] is a machine learning algorithm which is used for both regression and classification problems. The main principle used is the optimal separation. The one which is a good classifier is the one with the maximum distance between data points of different classes. The output of the algorithm is a hyperplane that is used for categorizing new data. SVM is formulated as depicted in Eq. 12:

$$\min_{\alpha,\alpha^*} \frac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + Z^T (\alpha_i - \alpha_i^*) \tag{12}$$

- Subject to $0 \le \alpha_i, \alpha_i^* C; e^T(\alpha - \alpha^*) = 0; e^T(\alpha + \alpha^*) = C_v$here, e is the unity vector, C is the upper bound, Q is l by l positive semi-definite matrix, $Q_{ij} = y_i y_j K(x_i, x_j), Kx_i, x_j = \theta(x_i)^T \theta(x_j)$.
- Decision tree (regression tree): Decision tree (DT) [23] classifies instances by starting at the root of the tree and moving through it till a leaf node. We will calculate the probability of occurrence of all the events at each level using the ID3 algorithm. This consists of a decision node which specifies each attribute, edge which splits one attribute into many. We also have a leaf node which tells us about the target attribute and its probability of occurrence. We also have a path that specifies the attributes to make a final decision.
- For the given data: $y \in R^n, x \in R^{n*p}$; each observation $(y_i, x_i) \in R_{p+1}$; i = 1, ………, n. Suppose we have partition of R_p into m regions R_1, ……….,R_m. We predict the response using a constant on eachR_i.The formulation for DT is shown in Eq. 13:

$$f(x) = \sum_{i=1}^{m} C_i . 1_{(x \in R_i)} \tag{13}$$

- Inorder to minimize $\sum_{i=1}^{n} (y_i - f(x_i))^2$ one needs to choose: $(\widehat{C}_i) = ave(y_j : x_j \in R_i)$. Consider splitting variable $j \in 1, \ldots, p$ and splitting point $S \in R$. Define two half planes: $R_1(j,s) = \{x \in R^p : x_j \le s\}$ and $R_2(j,s) = \{x \in R^p : x_j > s\}$.
- Extreme learning machine: This model [24] is a learning algorithm for the single hidden layer neural networks used in classification and regression [25]. The ELM used for single hidden layer feedforward neural network training can adaptively set the hidden layer node number and it can randomly assign the input weights so that output layer weights obtained by the least square method, the whole learning process completed with very little error(minimum number of error). The training speed compared with the traditional BP algorithm based on experiments is improved.

  For N arbitrary distinct samples $(x_i, t_j) \in R^n * R^m$

  Standard SLFNs with L hidden nodes and activation function g(x) are mathematically modeled as $H\beta = T$ which is equivalent to Eq. 14,

$$\sum\nolimits_{i=i}^{L} \beta_i G(a_i, b_i, x_j) = t_j; \quad j = 1, \ldots, N \tag{14}$$

  here, a_i is the input weight vector connecting the ith hidden node and the input nodes, β_i is the weight vector connecting the ith hidden node and the output node, b_i is the threshold or impact factor of the ith hidden node and H is hidden layer output matrix.

- Linear regression model: Linear Regression (LR) is the most commonly used category of predictive analysis [19]. It is used to show relationship between two or more variables where there are two types of variables one is dependent and the other is explanatory. LR is formulated as depicted in Eq. 15:

$$Y = \beta X + A + e \tag{15}$$

here, β is slope of the line (predicted increase or decrease for Y scores for each unit increasing X), X is the independent variable, A is Y-intercept (level of Y when X is 0) and e is the random error term.

Table 4 Machine learning regression models

Model name	Method used	Package required	Tuning parameters
BRR	bridge	monomvn	T = 1000, lambda2 = 1
BRNN	brnn	Brnn	neurons = 2,mu = 0.005,mu_dec = 0.1,mu_inc = 10, mu_max = 1e10,min_grad = 1e-10
SVM	ksvm	kernlab	kernel = "rbfdot", type = "nu-svr"
DT	rpart	None	usesurrogate = 0, maxsurrogate = 0
ELM	elm	elmNN	nhid = 10, actfun = "sig"
LM	lm	None	None
NN	nnet	Nnet	maxit = 100, MaxNWts = 10,000
RF	randomForest	randomForest	ntree = 500, mtry = 2

- Random forest: Random forest (RF) is a learning method that works by developing a huge number of choice trees at time of training and outputs the mean forecast (regression) of the individual trees. The formula for computing RF is shown in Eq. 16:

$$\mathrm{h}_k(X) = h(X|\theta_k); \quad k = 1, \ldots, n \tag{16}$$

  here, θ_k: are independent identically distributed random vectors, X is input variable, n is the number of trees and h = {h_1 (X),……..,h_k (X)} ensemble of classifiers.

The methods are available in R open source software [26] which is licensed under GNU GPL. To obtain better results, the parameters of the models need to be tuned. The brief detail of the methods with the required packages and their tuning parameters is described in Table 4.

The selected machine learning models are further assembled using the proposed ensemble algorithm which is discussed further.

3.2 Ensembling

Ensembling is the process of stacking multiple machine learning models and improving the prediction accuracy or decrease variance, by combining the capabilities of models. The machine learning regression models are applied to the generated dataset for predicting resource usage. These models are further grouped based on the proposed ensemble Algorithm 1.

Algorithm 1 Proposed Ensemble Model Algorithm

```
Start
    Set BestAcc = 0
    Set BestMSet= NULL
    Set ModelList=[m1, m2, m3, m4 ... mn]
    Set pd= PredictedDataSet
    Set Actual= pd[1]
        for each i in 1,...,n do
            Set  x ← rand(m2 : mn)
            Set  s ← sample((m1, mj), x)
            e ← ensemble(s)
            acc ← mean(e == pd[, 1]) * 100
                if acc > BestAcc then
                    BestAcc ← acc
                    BestMSet ← s
                end if
        end for each
    return BestAcc
    return BestMSet
Stop
```

This algorithm explains how the application requirements are obtained. It is an ensemble algorithm which provides the best combination of machine learning models and enhances the prediction accuracy. This ensembled model is applied to the generated resource usage dataset from Microsoft Azure for predicting future resource usage. The output produced by the ensembled Algorithm 1 is used as an input parameter in Algorithm 2 which further schedules the resources in an optimized manner.

Algorithm 2 Optimized Prediction Based Scheduling (OPSA)

Input Predicted resource requirements from Algorithm 1 [ACU and AMU]
Application Tasks List, ATList= $[T_1, T_2, ..., T_n]$, List of VMs, VmList= $[Vm_1, Vm_2, ..., Vm_n]$
Output Optimal mapping between ATList and VMList
Reduced Execution Time, Cost and SLA violations
Begin

```
Set Ap_cp = ACU
Set Ap_mem = AMU
Set n = number of applications
Set pdim ← ATList size
for each i in 1,...,n do
    Set vmSize ← getVmList().size()
    Set firstIdleVm ← null
        for each j in 1,...,vmSize do
            Set vm ← getVmList().get(j)
                If vm.getState()== IDLE
                    ∧ Ap_i_cp < vm.getAvailableMips()
                    ∧ Ap_i_mem < vm.getCurrentSize()
                then
                    firstIdleVm ← vm
                    Assign application to firstIdleVm
                    Set firstIdleVmBUSY.setState() ← BUSY
                endIf
            endfor
    Randomly initialize the velocity v_i and position p_i of particles(tasks)
        for each t ∈ ATList
            Repeat
                Compute FV(RCS) for each particle using Algorithm 3 and
                update the values for p_i
                If FV_cur < pbest
                then Assign pbest ← p_i
                else Keep pbest
                endIf
                Compare all pbest
                Assign gbest ← highest pbest
                If gbest_cur < FV_cur
                then Assign gbest ← p_i
                else Keep previous gbest
                endIf
                Assign particle with highest gbest to VM for execution
            Update the velocity v_i and position p_i of particles
            Until stopping criteria is not satisfied
        endfor
    endfor
Return Execution Time, Cost and SLA violations
```

Stop

In the proposed algorithm, a variety of combinations is formed for different models and means accuracy *acc* is calculated for each combination. The computed accuracy rate is further compared with the best accuracy *BestAcc* already generated. If the calculated *acc* is better than *BestAcc* then *BestAcc* is replaced with the calculated *acc* and the provided combination of models is returned as the best model set to ensemble. The primary focus of the proposed algorithm is to generate the best set of models which can be assembled to enhance the performance of regression models for predicting the usage of resources.

The working of the scheduler module depends upon the output of the prediction module. In this module, the availability of the resources is checked from the resource pool. Then, the resources are scheduled efficiently based on the usage requirements of the application for further processing as discussed in Algorithm 2. The aim of this scheduling algorithm is to improve the performance in terms of execution time, cost, and SLA by efficiently allocating the resources to the tasks.

3.3 Proposed algorithm

The objective of this algorithm is to find an optimal solution by considering multiple criteria. Therefore, the features of swarm intelligence are combined with TOPSIS. The former technique is very quick at determining the optimal solutions and the latter helps to make a decision based on multiple criteria. *Swarm Intelligence* is a simple optimization technique that performs the parallel execution of tasks to handle global optimization issues. It works with a swarm of individuals also known as particles. Each particle is a representation of a candidate solution. Particles observe basic conduct: imitate the performance of adjacent particles and success achieved on its own. Therefore, the location of a particle is determined by the best particle in the neighborhood *pbest* and by the best solution found by all the particles in the whole population *gbest*. *TOPSIS* is a mathematical multi-criteria decision based algorithm which supports the working of swarm optimization by finding local optimum solutions. The traditional TOPSIS approach tries to pick alternatives that have the shortest distance from the ideal-positive solution at the same time and the farthest distance from the ideal-negative solution. The ideal-positive approach maximizes the benefits and minimizes the cost, while the negative optimal solution optimizes the cost criteria and the benefit criteria are minimized. TOPSIS makes good use of information on the attributes, offers a cardinal ranking of alternatives and does not require individual attribute preferences. To apply TOPSIS, the values of the attributes must be an integer (either in increasing order or decreasing).

In this algorithm, the resources are scheduled on the basis of a predicted set of resources by ensemble algorithm [18]. Initially, the average CPU utilization (AP_{cp}) and average memory utilization (AP_{mem}) requirement of a scientific application is checked against the available CPU and memory size of the firstIdleVm. If the CPU and memory requirements of the application are less than the available MIPS and current size of VM, then the application is mapped to that particular VM. Further, to schedule the tasks of mapped application, an optimization approach is followed. Each particle is represented by means of velocity (v_i) and position (p_i) that can be obtained using formula 17 and 18. Each particle determines its velocity (v_i) and position (p_i) according to its best position *pbest* and the best particle position in each generation *gbest*. The values assigned to each particle's dimensions reflect the computational resources allocated to VM. Therefore, a particle reflects the mapping of the tasks and available VM resources. The parameters are the tasks which are also allocated to the available VMs.

$$V_{i[k+1]} = w * V_{i[k]} + c1 * rand1 * \left(pbest - P_{i[k]}\right) + c2 * rand2 * \left(gbest - P_{i[k]}\right) \tag{17}$$

$$P_{i[k+1]} = P_{i[k]} + V_{i[k+1]} \tag{18}$$

where $V_{i[k+1]}$ is current velocity and $V_{i[k]}$ is the previous velocity of particle *i*. $P_{i[k+1]}$ is current position and $P_{i[k]}$ is the previous position of the particle *i*. $c1$ and $c2$ are acceleration coefficients whose value can be taken between 1 and 2. $rand1$ and $rand2$ are the random numbers whose values lie between 0 and 1. In the traditional swarm optimization algorithm, the random values $(rand1\ and\ rand2)$ are generated by a uniform distribution method in the range of [0,1] (U[0,1]). The probability of each random value is similar in the range. This random parameter plays an important role in the overall performance, as it avoids premature convergences, increasing the most likely global optima. Particle's best position is denoted by *pbest* and the position of the best particle in the entire population is denoted as *gbest*. *w* is the inertia weight usually lie between 0 and 1. Fitness Value (FV) is used as an evaluation tool to measure the performance of a particle. The FV for each task is computed using TOPSIS algorithm which is explained in Sect. 3.3. If the current fitness value of a task is less than its *pbest* value, then current fitness value is assigned as its *pbest* value and the same process is repeated for all the tasks. Next, all the personal best values (*pbest*) are compared and the highest *pbest* value is assigned as the global best value (*gbest*). Again, if the current *gbest* value is less than the current fitness value, then current fitness value is assigned as *gbest* value and the task with the highest *gbest* value is given to VM for execution. The same procedure is applied to the rest of the tasks of all the mapped applications.

3.4 TOPSIS- a multi-criteria decision making algorithm

Several heuristic techniques such as Particle Swarm Optimization (PSO), ANT Colony Optimization (ACO), Artificial Bee Colony (ABC), etc. have been utilized by various researchers for optimizing single criteria based problems. PSO is a stochastic evolutionary algorithm (EA) search process, based on population, modeling the behavior of bird flocks. This type of algorithm is suitable for solving problems where a point of optimal solution is within multidimensional parameter space. ACO is an optimization technique in which the task is to find the best possible path along with a graph. It basically works on the behavior of the ants looking for a path between the source of food and their colony. In ACO, the solutions are built on the basis of two factors (a) attractiveness: desire to take move for state transition and, (b) pheromone trail: social interaction among agents to follow the path. ABC simulates the smart foraging behavior of a swarm of honeybees. This algorithm comprises of three main components (a) food sources: a possible solution to the optimization problem, (b) employed foragers, and (c) unemployed foragers are the number of possible solutions for the given optimization problem. These optimization techniques lack the ability to handle decision making based on multiple criteria. In order to attain better optimized results for problems based on multiple criteria, a multi-objective decision making algorithm named "TOPSIS" is incorporated [15, 16]. This method takes multiple factors into consideration while computing the fitness value for tasks. Algorithm 3 depicts the overall process followed by TOPSIS algorithm.

Initially, a decision matrix is constructed of size $t * c$, where t are the number of tasks (alternatives) and c represents the number of criterion as shown in Table 5.

Next, the decision matrix is normalized using Eq. 19.

$$DM_n \leftarrow (DM[c][i]) / \sum \sqrt{i^2} \tag{19}$$

where i = {1,2,…,t}, j = {1,2,…,c} and $(DM_n[j][i])$ are the elements of the decision matrix corresponding to i^{th} alternative and j^{th} criteria. Further, the elements of $(DM_n[j][i])$ are multiplied by inertia weight as shown in Eq. 20, provided by the decision maker as per the importance of criteria in the scheduling process.

$$DM_{nw}[i][j] \leftarrow DM_n[i][j] * inertiaweight[j] \tag{20}$$

Now, calculate the Att_p and Att_n, where Att_p are the set of attributes that have positive impact and Att_n are those set of attributes which have negative impact on the solution.

Next step is to evaluate the separation measure for Att_p and Att_n for each attribute using Eqs. 21 and 22.

$$SM\_Att_p[i] \leftarrow \Big(\sum_j (Att_p[\mathrm{c}] - DM_{nw}[i][c])^2\Big)^{\frac{1}{2}} \tag{21}$$

$$SM\_Att_n[i] \leftarrow \Big(\sum_j (Att_n[\mathrm{c}] - DM_{nw}[i][c])^2\Big)^{\frac{1}{2}} \tag{22}$$

Algorithm 3 TOPSIS Algorithm to Compute RCS

Input m alternatives, c criterion and inertia weight for each task
Output Relative Closeness Score (RCS)
Begin

1 Construct Decision Matrix(DM)
  DM[Execution Time] $ET_t = \sum_{m=1}^{v_n} \sum_{k=1}^{j_n} ET_{nmk}$
  DM[Execution Cost] $EC_t = Processingcostpersecond * ET_t$
2 Calculate normalized DM $[DM_n]$
  for each i in alternative
    for each c in criteria
      $\sum \sqrt{c^2} \leftarrow (\sum (DM[i][c])^2)^{\frac{1}{2}}$
      $DM_n \leftarrow (DM[c][i]) / \sum \sqrt{i^2}$
    endfor
  endfor
3 Determine weighted normalized DM $[DM_{nw}]$
  for each i in alternative
    for each c in criteria
      $DM_{nw} \leftarrow DM_n[i][c] * inertiaweight[c]$
    endfor
  endfor
4 Calculate Att_p and Att_n
  $Att_p = setofpositiveattributes$
  $Att_n = setofnegativeattributes$
5 Evaluate the separation measures from Att_p and Att_n for each attribute
  for each i in alternative
    for each c in criteria
      $SM\_Att_p[i] \leftarrow (\sum_j (Att_p[c] - DM_{nw}[i][c])^2)^{\frac{1}{2}}$
      $SM\_Att_n[i] \leftarrow (\sum_j (Att_n[c] - DM_{nw}[i][c])^2)^{\frac{1}{2}}$
    endfor
  endfor
6 Compute RCS
  for each i in alternative
    $RCS[i] \leftarrow SM\_Att_n[i] / SM\_Att_n[i] + SM\_Att_p[i]$
  endfor
7 Return RCS
Stop

Finally, compute the relative closeness score (*RCS*) using Eq. 23 and update the velocity of tasks (particles) in Algorithm 2 for determining the *gbest* value for scheduling.

$$RCS[i] \leftarrow SM\_Att_n[i] / SM_{Att_n[i]} + SM\_Att_p[i] \tag{23}$$

The final computed RCS is shown in Table 6. The score is sent as FV for the tasks in Algorithm 2 for scheduling.

4 Experimental setup and results

The tools used to set up a testbed for experiments include Netbeans IDE 8.2, CloudSim 3.0, WorkflowSim 1.0, Java SDK 8, Microsoft Azure. WorkflowSim extends the features of CloudSim that facilitates simulating cloud environments by creating datacentres, hosts, VMs, cloudlets, etc. This has been used to collect the resource usage requirements of scientific applications. OpenStack, an open source software platform for cloud computing, is installed on the server (HP DL380) to setup a cloud environment.

Further, four heterogeneous virtual machines are created for parallel execution of application. The performance of the proposed resource prediction model has been validated in a cloud environment.

The characteristics of the VMs are mentioned in Table 7 which clearly indicates that all the four VMs have different configurations which creates a distributed environment for deploying applications. The resource usage of the VMs before deploying the application is shown in Table 8.

4.1 Scenario 1: Cybershake scientific application

CyberShake represents the aleatory variability in wave excitation through conditional hypocenter distributions and conditional slip distributions, and it characterizes the epistemic uncertainty in the wavefield calculations in terms of alternative 3D seismic-velocity models [27]. There are four different CyberShake applications namely cyber30, cyber50, cyber100, and cyber1000, which vary in number of instances. The resource usage requirement of these applications is shown in Table 9. The application requirements have been obtained using the prediction algorithm (Algorithm 1). In this algorithm, the previous usage of resources along with application size and various other factors has been taken into input. Finally, based on the historical data and by applying our ensemble algorithm, the application resource requirements are predicted.

The proposed prediction based scheduling approach has been compared with the existing heuristics namely DataAware, FCFS, MaxMin, MinMin, and MCT on the basis of execution time and cost. The results are also validated on the basis of SLA violation rate and the comparative analysis is shown between proposed and existing scheduling approaches.

Table 5 Decision matrix

Cloudlet ID	Time	Cost
30	0.11	8028.03
13	137.03	4478.1
20	25.11	130.13
18	29.22	142.46
16	46.15	193.25
14	47.44	197.33
22	31.06	148.27
21	1.36	4.08
19	1.36	4.08
17	1.53	4.59
2	198.22	4668.46
15	1.53	4.59
23	1.43	4.29
26	23.99	126.88

4.1.1 Results: Cybershake scientific application

- **Case 1: execution time**

The performance of proposed approach has been analyzed with Cybershake application with 30, 50, 100 and 1000 tasks. The time taken by existing and proposed scheduling approach for executing cyber30, cyber50, cyber100 and cyber1000 can be seen in Fig. 2.

The time taken by RPS approach for executing Cyber30 is 30.18 ms while the existing heuristics DataAware, FCFS, MaxMin, MCT and MinMin executed the applications in 62.59 ms, 84.005 ms, 125.76 ms, 71.63 ms and 49.04 ms, respectively. Similarly, for Cyber50, Cyber100 and Cyber1000, the proposed approach took 42.63 ms, 53.70 ms and 73.03 ms respectively. In comparison, the DataAware approach executed Cyber50, Cyber100 and Cyber1000 in 76.83 ms, 83.78 ms, and 94.45 ms respectively. Further, FCFS executed Cyber50 in 137.08 ms, Cyber100 in 154.46 ms and Cyber1000 in 189.99 ms. Also, the execution time taken by MaxMin is 154.73 ms, 155.16 ms and 169.81 ms for Cyber50, Cyber100 and Cyber1000, respectively. Next, MCT accomplished the execution of Cyber50, Cyber100 and Cyber1000 in 85.37 ms, 95.36 ms and 150.40 ms, respectively. At last, MinMin executed Cyber 50 in 63.41 ms, Cyber100 in 93.89 ms and Cyber1000 in 103.49 ms.

The results shown in Fig. 2 clearly states that the proposed prediction based scheduling approach took far less execution time when compared to existing scheduling approaches. The overall execution time is curtailed by 35.59% using the proposed approach.

- **Case 2: cost**

With every action during the application execution there is a cost associated with it, for example- cost for resource usage, data transfer cost, and execution cost. The cost incurred by existing and proposed approaches is depicted in Fig. 3. The cost obtained by the proposed approach for executing 30 jobs of Cybershake is 144.54 INR which is least amongst existing scheduling heuristics whereas, MaxMin attained the highest cost of 602.21 INR. For executing 50 jobs, RPS approach obtained cost of 204.16 INR while MaxMin executed the jobs with highest cost of 740.92 INR, FCFS with 656.40 INR. Similarly, for executing 100 & 1000 jobs of cybershake the proposed RPS approach incurred the minimal cost of 257.14 INR and 349.72 INR whereas MaxMin (742.98 INR) and FCFS (909.75 INR) obtained the highest cost to execute 100 & 1000 jobs of cybershake. Hence, the proposed approach is better than the existing approaches as it has minimum execution cost.

Table 6 Relative closeness score

Cloudlet ID	Run time	Start time	Finish time	Cost	Score
30	0.11	0.1	0.21	8028.03	0.4761452
13	137.03	0.21	137.24	4478.1	0.3840374
20	25.11	137.24	162.35	130.13	0.9160464
18	29.22	137.24	166.46	142.46	0.9027774
16	46.15	137.24	183.39	193.25	0.8494419
14	47.44	137.24	184.69	197.33	0.8454663
22	31.06	162.35	193.41	148.27	0.8968708
21	1.36	193.41	194.77	4.08	0.9957131
19	1.36	194.77	196.13	4.08	0.9957131
17	1.53	196.13	197.66	4.59	0.9951354
2	198.22	0.21	198.43	4668.46	0.2794519
15	1.53	197.66	199.19	4.59	0.9951354
23	1.43	198.43	199.86	4.29	0.9954752
26	23.99	183.39	207.39	126.88	0.9196805

- **Case 3: SLA violation rate**

It is very important that there should be minimal violation of SLAs so that cloud providers are able to retain their users. Another major goal of the proposed approach was to reduce the SLA violation. The results of the SLA violation rate can be seen in Fig. 4.

FCFS has the highest SLA violation rate of 10.32%, followed by DataAware and MCT with 5.44% and 2.25%. The MaxMin and MinMin have very minute difference between SLA violation, the former attained 1.14% while the latter obtained 1.70%. The proposed approach has 0.91% of SLA violation rate, which is the least amongst existing scheduling approaches. It can be clearly seen that the proposed approach has the minimum rate of SLA violation. Therefore, the proposed prediction based scheduling approach is better than the existing approaches.

Table 8 Resource usage of VMs

VM	Average CPU usage	Average memory usage
VM1	38.31%	12.56%
VM2	35.45%	28.06%
VM3	43.69%	35.89%
VM4	45.61%	40.87%

Table 9 Resource usage requirement of Cybershake

Application	Average CPU required	Average memory required
cyber_30	48.03%	50.89%
cyber_50	52.07%	56.06%
cyber_100	62.61%	62.48%
cyber_1000	78.15%	59.38%

4.2 Scenario 2: floodplain scientific application

Floodplain application [28, 29] is committed to developing an accurate simulation for frequent surges in storms at North Carolina's coastal regions. Currently, a four model system is deployed which comprises of different models namely Hurricane Boundary Layer which is focused on winds, ADCIRC is for surges in the storms, SWAN and Wavewatch III are directed towards waves generated by winds at near-shore along with oceans. To achieve accuracy in analysis and floodplain mapping in a given region, broader coverage of parametric space is needed which also

Table 7 Configuration of VMs

VM	vm_id	Ram	Storage capacity	Processor name	Mips	Bw(GBps)	OS	Vmm	Graphics card
VM1	1	4 GB	58 GB	i7 7700 k	1000	19.9	CentOS	Xen	Nvidia GeForce G7X 1080
VM2	2	8 GB	256 GB	i7 8700 k	1000	21.2	Windows	KVM	AMD Radeon Pro WX7100
VM3	3	16 GB	500 GB	i7 6700 k	1000	37.0	CentOS	Xen	AMD Radeon Pro 560
VM4	4	32 GB	500 GB	i7 7900x	1000	41.6	CentOS	KVM	NvidiaQuadro M620

describes the characteristics of storms. The application's instance executes in about a day, hence demanding large computational and storage resources. There are four different sizes of Floodplain applications namely flood10, flood20, flood30, and flood50, which vary in number of jobs. The resource usage requirement of Floodplain application with different number of jobs like 10, 20, 30, and 50 is shown in Table 10.

The proposed prediction based scheduling approach has been compared with the existing heuristics namely DataAware, FCFS, MaxMin, MinMin, and MCT on the basis of execution time and cost. The results are also validated on the basis of SLA violation rate and the comparative analysis is shown between proposed and existing scheduling approaches. The proposed RPS approach has been executed and tested on Microsoft Azure Cloud by incorporating Azure Scheduler. Firstly, a resource group has been set up in the cloud where VMs have been created for executing scientific applications. Further, the jobs of scientific applications have been uploaded in the scheduler job collections directory for execution. Finally, the resources are scheduled efficiently to the application for further processing as discussed in Algorithm 2.

4.2.1 Results: floodplain scientific application

- **Case 1: execution time**

The performance of the proposed scheduling approach has been analyzed for floodplain application with 10, 20, 30, and 50 jobs where every single job can comprise of hundred to thousand tasks. It is evident from Fig. 5 that the proposed scheduling approach has minimal execution time (20.65 ms) for flood application with 10 jobs, whereas Max–Min has the maximal execution time of (68.65 ms). The performance of the proposed approach is also verified by incrementing the size of application to 20, 30, and 50 jobs.

The proposed approach obtains the least execution time of (32.106 ms) for 20 jobs, while FCFS attains the highest execution time (73.68 ms). Similarly, for 30 and 50 jobs the proposed approach has the lowest execution time (48.84 ms) and (62.719 ms), wherein MCT and Max–Min give the highest execution time (91.36 ms) and (109.53 ms), respectively. The experimental results shown in Fig. 5 clearly states that the execution time taken by the proposed prediction based scheduling approach is far less than the execution time taken by existing scheduling approaches.

- **Case 2: cost**

With every action during the application execution there is a cost associated with it, for instance- cost for execution, resource usage and data transfer. The cost incurred by existing and proposed approaches is depicted through Fig. 6. The proposed approach obtained the cost of 98.87 INR for executing flood application with 10 jobs which is least amongst existing approaches, whereas Max–Min scheduling approach incurred highest cost of 328.71 INR.

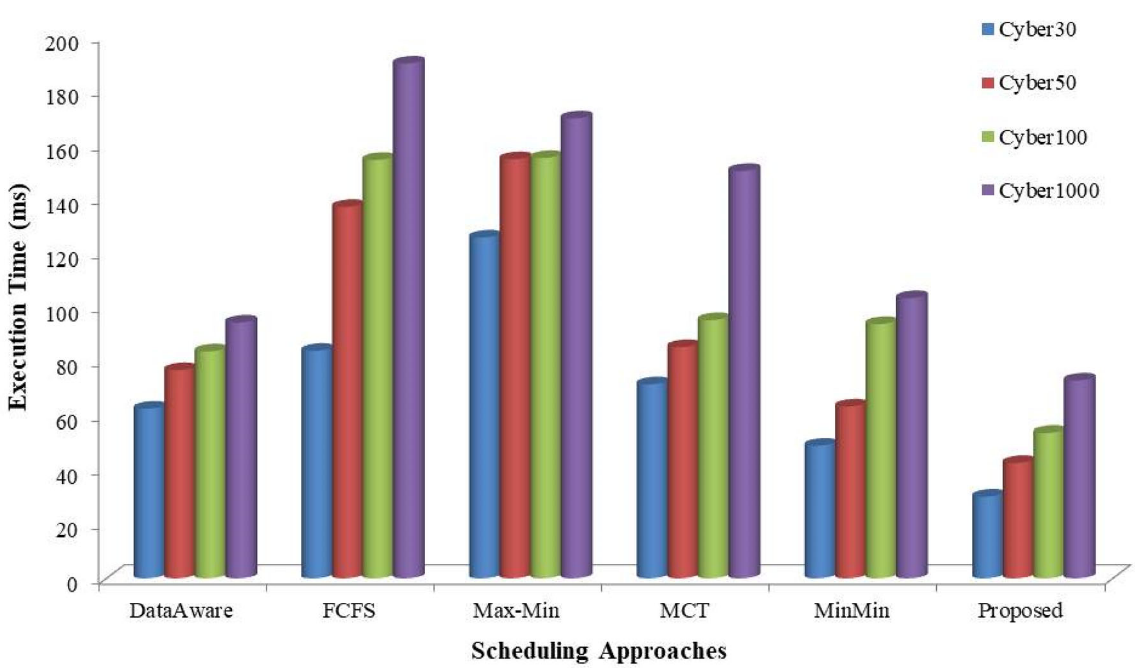

Fig. 2 Execution time comparison of existing and proposed scheduling approach

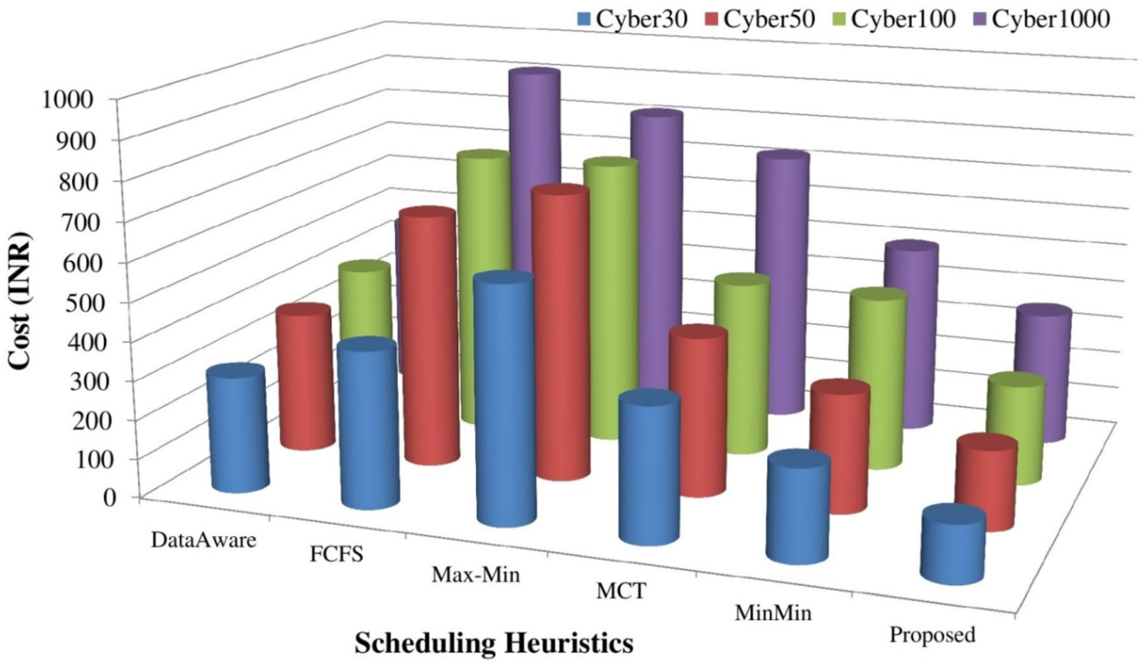

Fig. 3 Cost comparison of existing and proposed scheduling approach

Similarly, the cost incurred by proposed RPS approach for 20, 30 and 50 jobs is 153.73 INR, 233.85 INR and 300.31 INR respectively. On the contrary, FCFS attained the maximum cost of 352.80 INR for flood application with 20 jobs, MCT incurred highest expense of 437.45 INR for flood application with 30 jobs and Max–Min obtained the cost of 524.45 INR for flood application with 50 jobs, respectively. It is apparent that for all the different sizes of application the proposed approach have least execution cost, therefore the proposed RPS approach is better in comparison to existing scheduling heuristics.

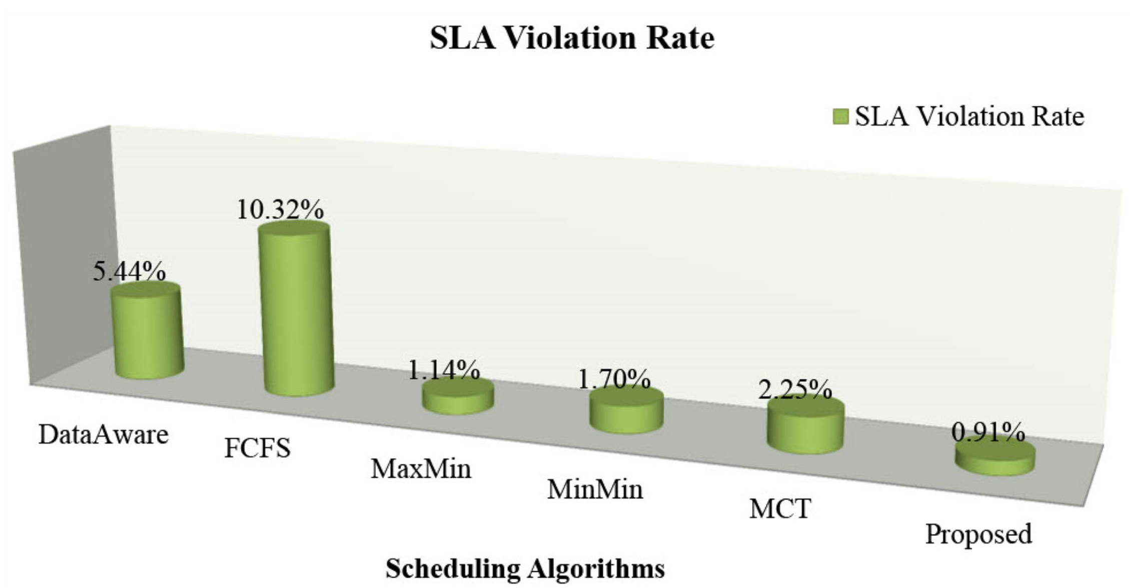

Fig. 4 SLA Violation rate comparison of existing and proposed scheduling approach

Table 10 Resource usage requirement of Floodplain

Application	Average CPU required	Average memory required (MB)
flood10	2.39%	4.4
flood20	4.695%	8.162
flood30	8.72%	11.008
flood50	14.33%	14.23

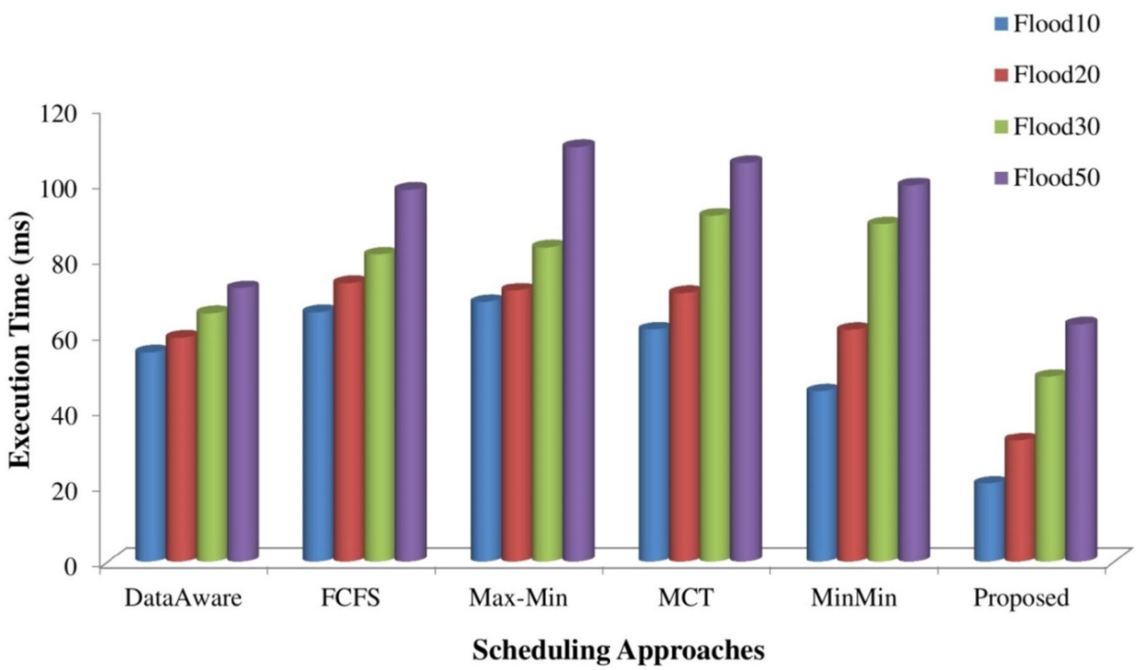

Fig. 5 Execution time comparison of existing and proposed scheduling approach

- **Case 3: SLA violation rate**

It is very important that there should be minimal violation of SLAs so that cloud providers are able to retain their users. Another major goal of the proposed approach was to reduce the SLA violation. FCFS has the highest SLA violaton rate of 8.21%, followed by DataAware and MCT with 6.04% and 2.19%. Thr MaxMin and MinMin have very minute difference between SLA violation, the former attained 1.92% while the latter obtained 1.13%. The proposed approach has 0.74% of SLA violation rate, which is least amongst existing scheduling approaches.

The graphical representation of the above mentioned results is depicted using Fig. 7. It can be clearly seen that the proposed approach has the minimum rate of SLA violation. Therefore, the proposed prediction based scheduling approach is better than the existing approaches.

5 Conclusion and future scope

This research work focused on the importance of optimized prediction based scheduling approach for scientific applications in a cloud environment. It elaborated the characteristics chosen through the feature selection approach and discusses a cloud testbed that was set up for testing and validating the proposed approach. The results of the proposed prediction based scheduling approach are validated for “Cybershake” and “Floodplain” scientific applications along with existing scheduling heuristics. The proposed OPSA approach outperforms the existing approaches in terms of execution time, cost and SLA violation rate. In the future, the proposed research work can be extended for different applications such as montage, epigenomics, weather prediction, and predicting anomalies.

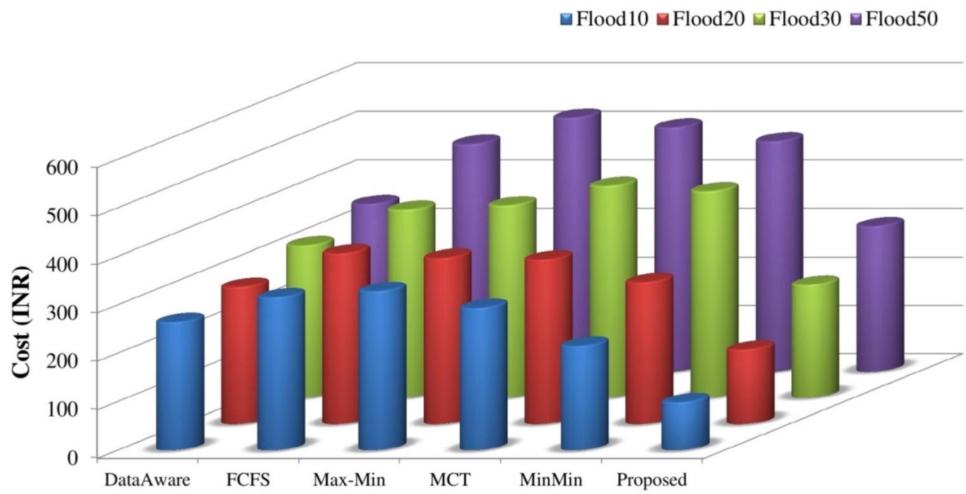

Fig. 6 Cost comparison of existing and proposed scheduling approach

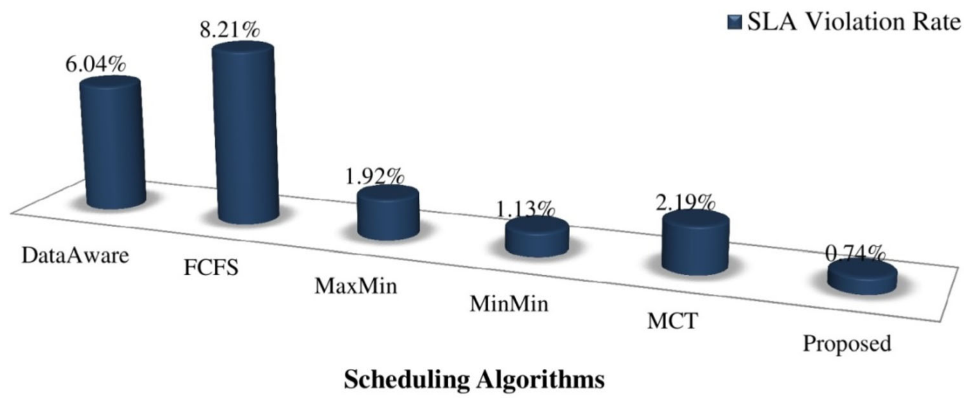

Fig. 7 SLA Violation rate comparison of existing and proposed scheduling approach

Acknowledgements One of the authors, Gurleen Kaur, acknowledges the Maulana Azad National Fellowship, UGC, Government of India, for awarding the scholarship which helped to avail the required resources to carry out this research work.

References

1. Kaur, G., Bala, A.: A survey of prediction-based resource scheduling techniques for physics-based scientific applications. Mod. Phys. Lett. B **32**(25), 1850295 (2018)
2. Peng, Z., Lin, J., Cui, D., Li, Q., He, J.: A multi-objective trade-off framework for cloud resource scheduling based on the deep q-network algorithm. Clust. Comput. (2020). https://doi.org/10.1007/s10586-019-03042-9
3. Wang, B., Wang, C., Song, Y., Cao, J., Cui, X., Zhang, L.: A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03048-8
4. Ebadifard, F., Babamir, S.M.: Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03177-0
5. Mortazavi-Dehkordi, M., Zamanifar, K.: Efficient deadline-aware scheduling for the analysis of big data streams in public cloud. Clust. Comput. **23**(1), 241–263 (2020)
6. Jiang, H., Haihong, E., Song, M.: Multi-prediction based scheduling for hybrid workloads in the cloud data center. Clust. Comput. **21**(3), 1607–1622 (2018)
7. Borkowski, M., Schulte, S., Hochreiner, C.: Predicting cloud resource utilization. In: 2016 IEEE/ACM 9th international conference on utility and cloud computing (UCC), pp. 37–42, IEEE (2016).
8. Li, J., Ma, X., Singh, K., Schulz, M., de Supinski, B.R., McKee, S.A.: Machine learning based online performance prediction for runtime parallelization and task scheduling. In: 2009 IEEE international symposium on performance analysis of systems and software, pp. 89–100, IEEE.
9. Kang, S., Veeravalli, B., Aung, K.M.M.: Dynamic scheduling strategy with efficient node availability prediction for handling divisible loads in multi-cloud systems. J. Parallel Distrib. Comput. **113**, 1–16 (2018)
10. Ghobaei-Arani, M., Jabbehdari, S., Pourmina, M.A.: An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach. Fut. Gener. Comput. Syst. **78**, 191–210 (2018)
11. Choudhary, A., Gupta, I., Singh, V., Jana, P.K.: A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Fut. Gener. Comput. Syst. **83**, 14–26 (2018)
12. Zhang, N., Yang, X., Zhang, M., Sun, Y., Long, K.: A genetic algorithm-based task scheduling for cloud resource crowd-funding model. Int. J. Commun. Syst. **31**(1), e3394 (2018)

13. Kansal, N.J., Chana, I.: Artificial bee colony based energy-aware resource utilization technique for cloud computing. Concurr. Comput.: Pract. Exp. **27**(5), 1207–1225 (2015)
14. Emeakaroha, V.C., Netto, M.A., Calheiros, R.N., Brandic, I., Buyya, R., De Rose, C.A.: Towards autonomic detection of SLA violations in cloud infrastructures. Fut. Gener. Comput. Syst. **28**(7), 1017–1029 (2012)
15. Behzadian, M., Otaghsara, S.K., Yazdani, M., Ignatius, J.: A state-of the-art survey of topsis applications. Expert Syst. Appl. **39**(17), 13051–13069 (2012)
16. Jahanshahloo, G.R., Lot, F.H., Izadikhah, M.: An algorithmic method to extend topsis for decision-making problems with interval data. Appl. Math. Comput. **175**(2), 1375–1384 (2006)
17. Chen, W., Deelman, E.: Workflowsim: a toolkit for simulating scientific workflows indistributed environments. In: 2012 IEEE 8th international conference on E-science, pp. 1–8, IEEE
18. Kaur, G., Bala, A., Chana, I.: An intelligent regressive ensemble approach for predicting resource usage in cloud computing. J. Parallel Distrib. Comput. **123**, 1–12 (2019)
19. Fumo, N., Biswas, M.R.: Regression analysis for prediction of residential energy consumption. Renew. Sustain. Energy Rev. **47**, 332–343 (2015)
20. Zhang, W., Duan, P., Yang, L.T., Xia, F., Li, Z., Lu, Q., Gong, W., Yang, S.: Resource requests prediction in the cloud computing environment with a deep belief network. Softw. Pract. Exp. **47**(3), 473–488 (2017)
21. Takai, S., Yang, T., Cafeo, J.A.: A Bayesian method for predicting future customer need distributions. Concurr. Eng. **19**(3), 255–264 (2011)
22. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Fut. Gener. Comput. Syst. **28**(1), 155–162 (2012)
23. Gupta, N., Ahuja, N., Malhotra, S., Bala, A., Kaur, G.: Intelligent heart disease prediction in cloud environment through ensembling. Expert Syst. **34**(3), e12207 (2017)
24. Huang, G., Huang, G.-B., Song, S., You, K.: Trends in extreme learning machines: a review. Neural Netw. **61**, 32–48 (2015)
25. Ismaeel, S., Miri, A.: Using elm techniques to predict data centre vm requests. In: 2015 IEEE 2nd international conference on cyber security and cloud computing, pp. 80–86, IEEE
26. R. D. C. Team: The R project for statistical computing. (2018). https://www.r-project.org/. Accessed 5 Dec 2020
27. Maechling, P., Deelman, E., Zhao, L., Graves, R., Mehta, G., Gupta, N., Mehringer, J., Kesselman, C., Callaghan, S., Okaya, D., et al.: Scec cybershake workflows automating probabilistic seismic hazard analysis calculations. In: Workflows for e-Science, pp. 143–163. Springer, London (2007)
28. Ramakrishnan, L., Gannon, D.: A survey of distributed workflow characteristics and resource requirements, pp. 1–23. Indiana University, Bloomington (2008)
29. Ramakrishnan, L., Plale, B.: A multi-dimensional classification model for scientific workflow characteristics. In: Proceedings of the 1st international workshop on workflow approaches to new data-centric science, p. 4, ACM

Dr. Gurleen Kaur completed her Ph.D. degree in the research area of Cloud Computing from Thapar Institute of Engineering and Technology, Patiala and her Master's degree in Computer Applications from Punjab Agricultural University. She worked as an Assistant Professor for 3 years and Teaching Assistant for 4 years. She has more than seven research publications in scientific journals and international conferences.

Dr. Anju Bala is working as an Assistant Professor in the Department of Computer Science and Engineering, Thapar University, Patiala, India. She received her B.E. in Computer Science and Engineering and MTech from Punjabi University and Ph.D. in the research area of Cloud computing from Thapar Institute of Engineering and Technology, Patiala. She has more than 50 research publications in reputed journals and conferences and guided more than 35 ME thesis in the same area.