Computing (2021) 103:895–916
https://doi.org/10.1007/s00607-021-00936-8

REGULAR PAPER

Prediction based task scheduling approach for floodplain application in cloud environment

Gurleen Kaur[1] · **Anju Bala**[1]

Received: 20 September 2019 / Accepted: 3 March 2021 / Published online: 26 March 2021

Abstract
Natural and environmental sciences are one of the scientific domains which seek a lot of attention as it requires high performance computation and large storage space. Cloud computing is such a platform that offers a customizable infrastructure where scientific applications can provision the required resources prior to execution. The elasticity characteristic of cloud computing and it's pay-as-you-go pricing model can reduce the resource usage cost for cloud client's. The various services offered by the cloud providers and the extravagant developments in the domain of cloud computing has attracted many scientists to deploy their applications on cloud. The change in number of tasks of scientific application directly affects the demand of cloud resources. Therefore, to handle the fluctuating demand of resources, there is a need to manage the resources in an efficient manner. This research work focuses on the design of a prediction based scheduling approach which maps the tasks of scientific application with the optimal VM by combining the features of swarm intelligence and multi-criteria decision making approach. The proposed approach improves the accuracy rate, minimizes the execution time, cost and service level agreement violation rate in comparison to existing scheduling heuristics.

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

Mathematics Subject Classification 00A69

1 Introduction

Scientific computing uses the state-of-the-art of high performance computing capabilities to solve the complex problems in various scientific domains such as weather forecasting, earthquake, sub-atomic particle behavior, turbulent flows and

✉ Gurleen Kaur
gurleen.kaur@thapar.edu

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

manufacturing processes etc. As the demand of resource requirements for solving the scientific problems is dynamic, so there is a need for a flexible platform which can handle the above-mentioned challenges in scientific applications concerning data storage and computation.

Cloud computing provides a dynamic environment for deploying scientific applications by offering services such as infrastructure, platform and software. Various other features such as on-demand service, resource pooling, pay-as-per-use, elasticity, etc. has attracted the scientists to deploy scientific applications on cloud. For effective utilization of virtualized resources in cloud, there is a need for efficient prediction based scheduling of tasks inorder to maximize performance and minimize execution time. Therefore, it is essential to first predict the resource requirements for scientific applications and then schedule them appropriately to meet the Quality of Service (QoS) requirements of the scientific users by taking SLA violations into consideration.

This research work focuses on the design of a prediction based scheduling approach which maps the tasks of scientific application with the optimal VM by combining the features of swarm intelligence and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) approach. The proposed approach improves the accuracy rate, minimizes the execution time, cost and SLA violation rate in comparison to existing scheduling heuristics.

2 Related work

Various researchers have proposed prediction and scheduling techniques but to perform prediction based scheduling, is still a challenging problem. A brief overview of the existing resource prediction based scheduling techniques is illustrated through Table 1.

Clovis et al. [1] designed Kalman filter based predictive grid scheduling framework for scheduling Condor jobs. The authors suggested that there is a need to develop flexible prediction mechanism for heterogeneous and distributed environment.

Shen [2] combined Support Vector Machine (SVM) with scheduling heuristic which predicted and scheduled the virtual resources within local scope. Gemm et al. [3] presented self-adjusting and analytical predictor along with First-Come-First-Serve (FCFS), Shortest-Job-First (SJF) and Earliest-Deadline-First (EDF) approaches for handling the translation problem of scheduling. But, the error rate while predicting resource usage was too high. Therefore, the authors recommended to reduce the error rate and predict the resource requirements for web workloads. Qingjia et al. [4] proposed a Prediction based Dynamic Resource Scheduling (PDRS) technique which reduced the SLA violation rate and improved resource utilization. The authors suggested predicting resource usage requirements for large-scale scientific applications. Micha et al. [5] employed Artificial Neural Network (ANN) for predicting utilization of resources on task level and schedule those tasks while Jiangtian et al. [6] combined ANN with Modified Critical Path (MCP) scheduling algorithm to improve prediction accuracy and runtime performance. Kang et al. [7] integrated Linear

Table 1 Existing resource prediction based scheduling techniques

References	Description	Prediction technique	Scheduling technique	Tool	Outcomes	Challenges
Clovis et al. [1]	Kalman filter based predictive grid scheduling framework was designed	Kalman filter prediction techniques	Condor job scheduling	GridSAM	Scheduling quality is improved	Need to develop flexible prediction mechanisms for heterogeneous and distributed environments
Shen [2]	Predicts and schedules the virtual resources within a local scope	Support vector machine (SVM)	SLA based scheduling algorithm	CloudSim	Improved prediction accuracy and stability of virtual resources	Scheduling resources globally
Gemm et al. [3]	Online prediction system is used for handling the translation problem of scheduling	Self-adjusting predictor and analytical predictor	FCFS, SJF, EDF, LSTF, HV, HD, HRU	WEKA toolkit and Cloud simulator	Error rate while predicting resource usage is too high	Predicting the resource requirements for the web workload Need to predict the requirements of jobs phases
Qingjia et al. [4]	Prediction-based dynamic resource scheduling (PDRS)	ARIMA pre-diction model	ARIMA-prediction-based scheduling, State-prediction-based scheduling	CloudSim	SLA violation rate is dramatically reduced, and resource saving ratio is increased	Should take large scale scientific applications for implementation
Micha et al. [5]	Predicts utilization of cloud resources on a per-task and per-resource level	Artificial neural network	Task scheduling	Open-source Java library Deeplearning4	Increases prediction accuracy. Lacks renormalization	To refine the applied Prediction techniques Nonsupervised Techniques should be used
Jiangtian et al. [6]	ANN	Artificial neural networks (ANNs)	Modified Critical Path (MCP) scheduling algorithm	R prediction tool and DAG scheduler	High prediction accuracy is achieved; Improved runtime performance	Examine runtime prediction errors Investigate disjoint parameter spaces

Table 1 (continued)

References	Description	Prediction technique	Scheduling technique	Tool	Outcomes	Challenges
Kang et al. [7]	Dynamic Scheduling Strategy (DSS) integrates the Divisible Load Theory and node availability prediction technique	Linear regression	Dynamic Scheduling Strategy	Cloud Simulator	Reduces total execution time by 44.60%	Require prediction based scheduling strategy for handling BigData applications
Ghobaei-Arani et al. [8]	Hybrid resource provisioning approach for cloud services	Linear regression model	Time-shared scheduling	CloudSim	Increases resource utilization and decreases total cost, while avoiding SLA violations	To integrate the proposed approach with admission control strategies for multi-tier cloud applications
Choudhary et al. [9]	Hybridization of Gravitational Search Algorithm (GSA) and Heterogeneous Earliest Finish Time (HEFT) to Schedule workflow applications	Gravitational search algorithm (GSA)	Heterogeneous Earliest Finish Time (HEFT)	WorkflowSim	Reduces makespan and the cost of execution	To deploy complex applications and check for VM failure
Zhang et al. [10]	A resource crowd-funding model to provide resources for cloud services	Genetic algorithm	Task based scheduling	CloudSim	Increases the stability of task execution and reduce power consumption	To encourage idle resources to join the resource pool
Proposed approach	Prediction based scheduling approach for scientific applications in cloud environment	Regressive ensemble approach for prediction	PSO-TOPSIS	R prediction tool, Cloudsim	Improves accuracy rate, minimizes execution time, cost and SLA violation rate	To show dependency among components of the scientific applications

Regression (LR) and Dynamic Scheduling Strategy (DSS) to predict node availability and schedule the tasks accordingly. The authors suggested that there is a need of prediction based scheduling strategy for handling scientific applications.

2.1 Gap analysis

Based on the literature survey following gaps have been identified:

- Due to the large size datasets of scientific applications, there is need for large computing capabilities [11].
- Non-functional information such as resource consumption information would lead to more rapid (reduced execution time) or less energy consuming (save cost) execution of scientific applications [11].
- Scientific applications have fluctuating resource demands which can be handled effectively when deployed on cloud [12]. Cloud Computing provide on demand service to users which guarantees convenience and effectiveness.
- Effective utilization of resources in cloud is very complex. A uniform scheduling method is needed for scientific applications to solve resource utilization issue [13]. Efficient prediction of resources can lead to effective scheduling. Hence, there is a need to design an efficient resource prediction technique which can predict a requisite set of resources for future and optimize resource deployment [14].
- To enhance scheduling efficiency for scientific applications of different size, it is important to have prior information of the resource requirements for executing the applications [15].
- Scheduling [16] can be described as the mapping and execution of the workload of cloud users based on the resource prediction outcomes of the application. Considering the large execution time and cost of resources required by scientific applications, resource scheduling has become a research challenge in cloud. Efficient scheduling [15] can reduce the execution time, cost and power consumption taking into consideration QoS factors. So there is a need to develop new solutions to handle the scheduling problems.
- Effective scheduling can help in improving the application efficiency, reduce the cost, increase the resource utilization and minimize the SLA violations [13].
- Hybridization of existing scheduling algorithm with other metaheuristic optimization techniques should be explored. This can enhance the scheduling effectiveness for scientific applications [17].

Therefore, this research work is focused towards developing prediction based task scheduling approach for scientific applications. The motive of proposed approach is to improve the accuracy rate, minimizes the execution time, cost and Service Level Agreement (SLA) violation rate.

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

3 Problem formulation

The objective of task scheduling algorithm in this research work is to solve a 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 of 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.

3.1 Total execution time

The time taken by a job to execute on a particular VM is known as execution time [18]. 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.

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

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

3.2 SLA violation (SLAV)

The end users state the QoS requirements to the Cloud Service Providers (CSPs) in the form of Service Level Agreements (SLAs) [19]. 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:

$$SLAV = \frac{prevRequested - prevAllocated}{prevRequested} \tag{4}$$

Here, *prevRequested* is the total amount of CPU MIPS and memory bytes requested/ required by a job for execution and *prevAllocated* is the total amount of CPU MIPS and memory bytes allocated for the execution.

3.3 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

$$\text{Total CPU utilization}\left(TCU_n\right) = \frac{\text{Clock cycles per instruction * instruction count}}{Clock\ rate}.$$

3.4 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.

3.5 Fitness value formulation

In this research work, the problem formulation for minimizing objective criterion mentioned in Eqs. 1, 3 and 4 can be optimized. The relative closeness score (RCS) for each task is calculated using multi-criteria decision making algorithm, TOPSIS [20, 21]. 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 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. The TOPSIS algorithm for computing RCS of tasks is elaborated in the next section.

4 Resource prediction based scheduling (RPS) framework

To achieve highly effective computations and best Quality of Service (QoS) of cloud, it is vital to perform scheduling of tasks in an efficient manner. The mapping of the submitted applications and VMs is considered to be successful if cloud has attained minimum execution time, cost, SLA violations and maximum utilization of resources. To solve the problem of multi-objective task scheduling, an Optimized Prediction based Scheduling Approach (OPSA) [22] 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 deployment module, the scientific application is deployed on actual cloud platform. Here, "Floodplain" as a scientific application is used as a case study in this research work. Floodplain application [23, 24] is committed to develop the 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, a broader coverage of parametric space is needed which also describes the characteristics of storms. The dynamic portion of the application is illustrated in Fig. 2 [23, 24]. The applications' instance executes in about a day, hence demanding large computational and storage resources.

Once the application is deployed, a resource usage dataset is generated and passed onto prediction module for further processing. The prediction module performs the preprocessing of data so that there are no null values and converts the

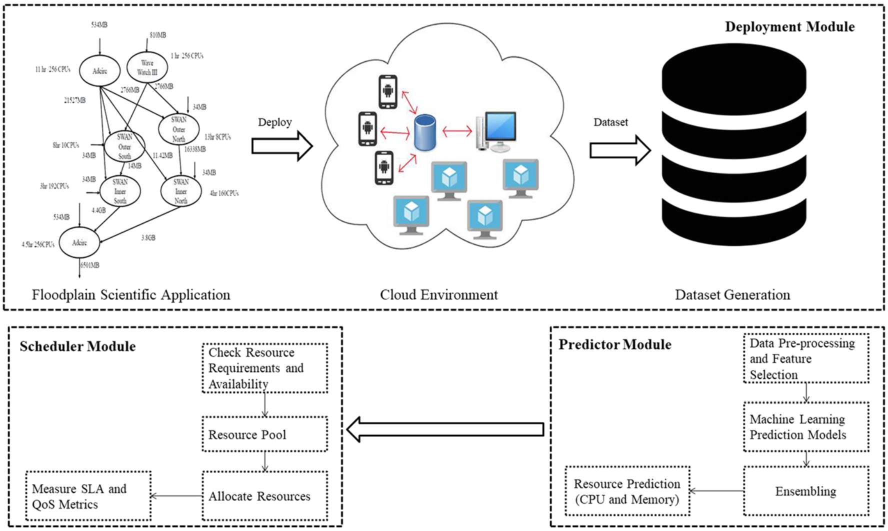

Fig. 1 Proposed RPS framework

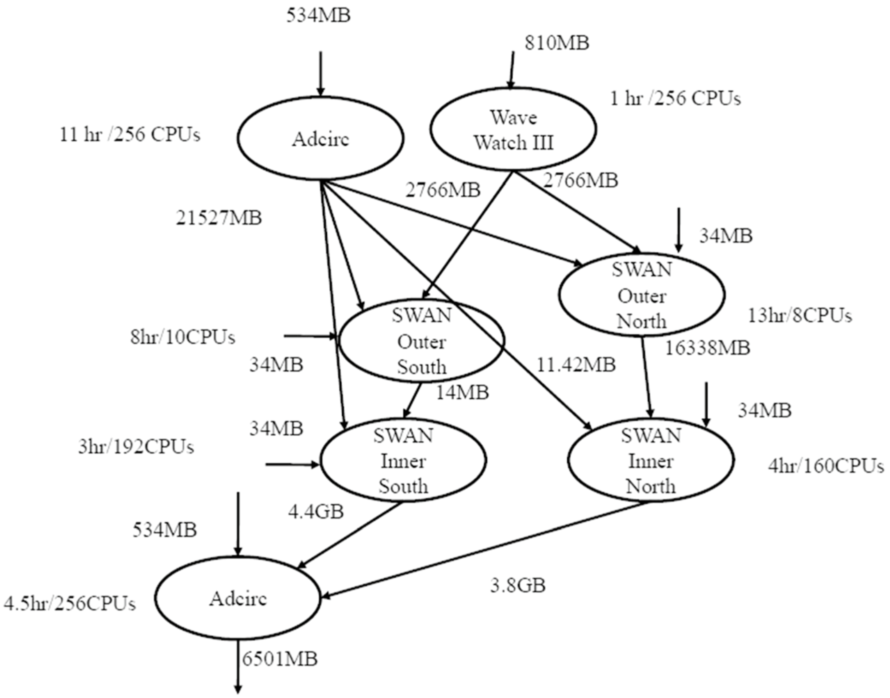

Fig. 2 Structure of floodplain

alphabetic values to numeric for smooth processing. It utilizes a meta-heuristic feature selection approach, Genetic Algorithm, to select relevant set of features. The advantage of using GA for feature selection is that it is a proven algorithm for solving problems of combinatory and optimization. In machine learning, one of the uses of GA is to pick up the right number of variables in order to create a predictive model. The idea of GA is to combine the different solutions generation after generation to extract the best genes (variables/features) from each one.

Table 4 illustrates the features which were selected to perform further experiments.

Finally, this module predicts the usage of resources by implementing ensemble algorithm [25] as proposed in our previous research work.

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 generated dataset for predicting resource usage. The brief detail of the methods with the required packages and their tuning parameters is described in Table 5.

These methods are available in R open source software [26] which is licensed under GNU GPL. To obtain better results, parameters of the models need to be tuned. These models are further grouped based on the proposed ensemble Algorithm 1.

Table 4 Illustration of the features

Features	Description
Id	The unique id number of the job being executed
Count	Total number of tasks being executed in background
name2	This attribute contains 6 names—Swan Inner North, Wave WatchIII, Swan Inner South, Adcirc, Swan Outer North and Swan Outer South. For processing purpose these names are given a numeric values 1, 2, 3, 4, 5, 6 respectively
Runtime	It is the execution time taken by a particular jobID. The units used to denote runtime is microseconds (ms)
File	Name of the les being executed
Cores	Total number of CPU cores utilized for execution
Link	It indicates whether the file is an input file or an output file
Size	Defines the size of the job. The unit used to represent size is bytes
CPU usage	The total percentage of CPU being utilized while executing a particular job
Memory usage	Total bytes of memory used by each job during execution

Table 5 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

Algorithm 1 **Proposed Ensemble Model Algorithm**

```
Start
    Set BestAcc = 0
    Set BestMSet= NULL
    Set ModelList=[m_1, m_2, m_3, m_4 ... m_n]
    Set pd= PredictedDataSet
    Set Actual= pd[1]
        for each i in 1,...,n do
            Set   x ← rand(m_2 : m_n)
            Set   s ← sample((m_1, m_j), 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
```

In the proposed algorithm, a variety of combinations is formed for different models and mean 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 pro-posed algorithm is to generate a 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 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.

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
```

4.1 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. The advantage of using PSO for scheduling is that the number of iterations has been reduced leading to less computational effort to reach the global optimum. In this algorithm, the resources are scheduled on the basis of predicted set of resources by ensemble algorithm [25]. 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 requirement 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. The velocity *vi* and position *pi* of particles (tasks) are randomly initialized and further updated using formula 9 and 10.

$$V_{i[k+1]} = w * V_{i[k]} + c1 * rand1 * \left(pbest - P_{i[k]}\right) + c2 * rand2 * (gbest - P_{i[k]}) \tag{9}$$

$$P_{i[k+1]} = P_{i[k]} + V_{i[k+1]} \tag{10}$$

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 number whose value lie between 0 and 1. 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 interia 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. 4.2. 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 *pbest* are compared and highest *pbest* value is assigned as *gbest* value. Again, if current *gbest* value is less than current fitness value, then current fitness value is assigned as *gbest* value and task with highest *gbest* value is given to VM for execution. The same procedure is applied to rest of the tasks of the mapped application with different sizes. The time complexity of the nested loops is equivalent to the number of times the innermost expression is executed. Hence, the proposed algorithm have $O(n^4)$ time complexity.

4.2 TOPSIS: a multi-criteria decision making algorithm

Several heuristic techniques such as PSO, ACO, ABC, etc. have been utilized by various researchers for optimizing single criteria based problems. These optimization techniques lack the ability to handle decision making based on multiple criteria. Inorder to attain better optimized results for problems based on multiple criteria, a multi-objective decision making algorithm named "TOPSIS" is incorporated [20, 21]. 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 Next, the decision matrix is normalized using Eq. 11.

$$DM_n \leftarrow (DM[c][i]) / \sum \sqrt{i^2} \tag{11}$$

where i = {1, 2,…,t}, j = {1, 2,…,c} and $(DM_n[j][i])$ are the elements of the decision matrix corresponding to ith alternative and jth criteria. Further, the elements of $(DM_n[j][i])$ are multiplied by inertia weight as shown in Eq. 12, provided by the decision maker as per the importance of criteria in the scheduling process.

$$DM_{nw}[i][j] \leftarrow DM_n[i][j] * inertiaweight[j] \tag{12}$$

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.

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

Next step is to evaluate the separation measure for Att_p and Att_n for each attribute using Eq. 13 and 14.

$$SM\_Att_p[i] \leftarrow \left(\sum_j \left(Att_p[\mathrm{c}] - DM_{nw}[i][c] \right)^2 \right)^{\frac{1}{2}} \tag{13}$$

$$SM\_Att_n[i] \leftarrow \left(\sum_j \left(Att_n[\mathrm{c}] - DM_{nw}[i][c]\right)^2 \right)^{\frac{1}{2}} \tag{14}$$

Finally, compute the relative closeness score (*RCS*) using Eq. 15 and update the velocity of tasks (particles) in Algorithm 2 for determining the *gbest* value for scheduling.

$$RCS[i] \leftarrow SM\_Att_n[i]/SM_{Att_n[i]} + SM\_Att_p[i] \tag{15}$$

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

5 Experimental setup

The tools used to set up testbed for experiments include Netbeans IDE 8.2, CloudSim 3.0, WorkflowSim 1.0, Java SDK 8, and Microsoft Azure. WorkflowSim extends the features of CloudSim that facilitates to simulate cloud environment by creating datacenters, hosts, VMs, cloudlets, etc. This has been used to collect the resource usage requirements of scientific applications. Further, four heterogeneous virtual machines are created on cloud platform for parallel execution of application. The performance of proposed resource prediction model has been validated on actual cloud environment. The characteristics of the VMs are mentioned in Table 6 which clearly indicates that all the four VMs have different configuration which creates a distributed environment for deploying applications. There are four different sizes of Floodplain application 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 7.

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.

6 Discussion: results and limitations

6.1 Results: RPS for floodplain

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 cloud where VMs have been created for executing scienti c applications. Further, the jobs of scientific applications have been uploaded in scheduler job collections directory for execution. Finally, the resources are scheduled efficiently to the application for further processing as discussed in Algorithm 2.

Table 6 Configuration of VMs

VM	VM type	RAM (GiB)	Storage capacity (GB)	Data disks	VCPUs	Max IOPS	OS
VM1	Standard D2s V3	8	16	4	2	3200	Ubuntu server 18.04 LTS
VM2	Standard B2ms	8	16	4	2	2400	CentOS based 7.5
VM3	Standard B1s	0.5	4	1	1	200	Windows server 2016 Datacenter
VM4	Standard Ds1 V2	3.5	7	4	1	3200	Windows server 2012 R2 Data-center

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.

6.1.1 Case 1: Execution time

The performance of proposed scheduling approach has been analysed for flood-plain application with 10, 20, 30 and 50 jobs where every single job can comprise of hundred to thousand tasks. It is evident from Fig. 3 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 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 lowest execution time (48.84 ms) and (62.719 ms), wherein MCT and Max–Min gives the highest execution time (91.36 ms) and (109.53 ms), respectively. The experimental results shown in Fig. 3 clearly states that the execution time taken by the proposed prediction based scheduling approach are far less than the execution time taken by existing scheduling approaches.

6.1.2 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. 4.

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. Similarly, the cost incurred by proposed RPS approach for 20, 30 and 50 jobs is 153.73 INR,

Table 7 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

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 has least execution cost, therefore the proposed RPS approach is better in comparison to existing scheduling heuristics.

6.1.3 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 violation rate of 8.21%, followed by DataAware and MCT with 6.04% and 2.19%. The 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.

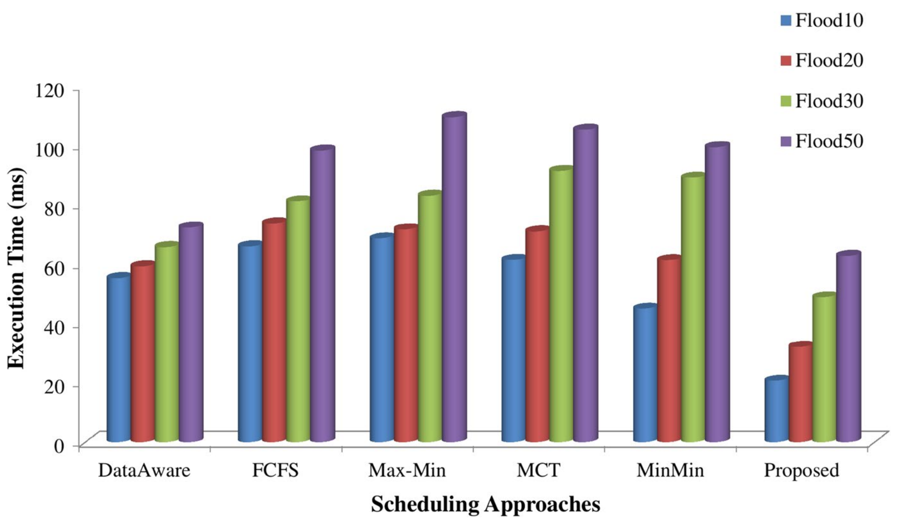

Fig. 3 Execution time comparison of existing and proposed scheduling approach

The graphical representation of the above mentioned results is depicted using Fig. 5. 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.

6.2 Limitations

Various aspects of scientific application execution such as scalability, band-width usage, response time, etc. have not been taken into consideration. These aspects can help in improving the accuracy rate, reducing the response time and enhance the scheduling efficiency. In future, the proposed prediction approach can be extended for predicting anomalies, peak resource usage period for improving the scheduling, load balancing and resource scaling.

7 Conclusion

This research work presented the case study of scientific application "Floodplain" considered for validating the proposed approach in the actual cloud environment. It elaborated the characteristics chosen through the feature selection approach and discusses a cloud test bed that was set up for testing and validating the proposed approach using the Microsoft Azure cloud platform. The results of proposed prediction based scheduling approach are validated for floodplain scientific application along with existing scheduling heuristics. The proposed RPS approach outperforms the existing approaches in terms of execution time, cost and SLA violation rate.

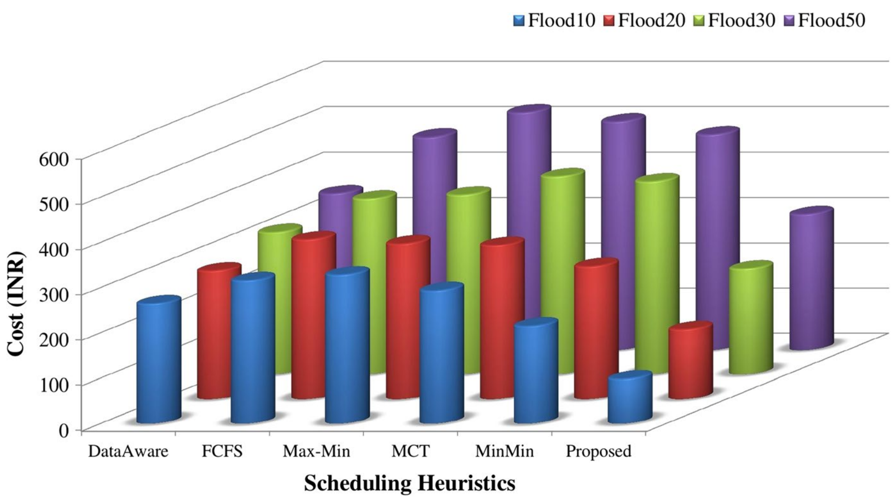

Fig. 4 Cost comparison of existing and proposed scheduling approach

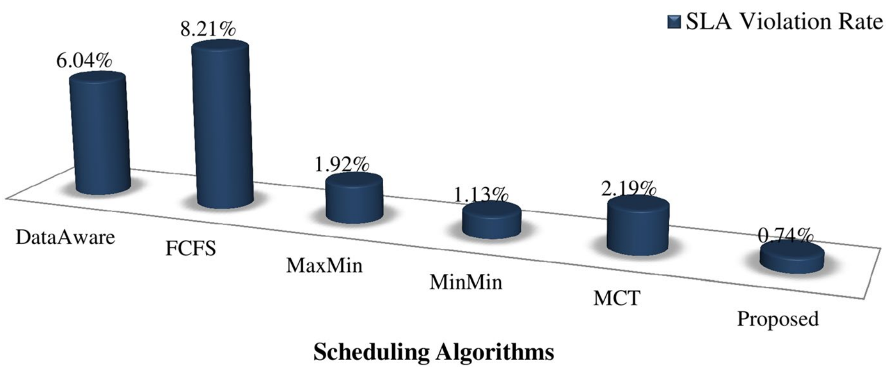

Fig. 5 SLA violation rate comparison of existing and proposed scheduling approach

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. Chapman C, Musolesi M, Emmerich W, Mascolo C (2007) Predictive resource scheduling in computational grids. In: 2007 IEEE international parallel and distributed pro-cessing symposium. IEEE, pp 1–10
2. Shen Y (2015) Virtual resource scheduling prediction based on a support vector machine in cloud computing. In: 2015 8th international symposium on computational intelligence and design (ISCID), vol 1. IEEE, pp 110–113
3. Reig G, Alonso J, Guitart J (2010) Prediction of job resource requirements for deadline schedulers to manage high-level slas on the cloud. In: 2010 9th IEEE international symposium on network computing and applications. IEEE, pp 162–167
4. Huang Q, Shuang K, Xu P, Li J, Liu X, Su S (2014) Prediction-based dynamic resource scheduling for virtualized cloud systems. J Netw 9(2):375–383
5. Borkowski M, Schulte S, Hochreiner C (2016) Predicting cloud resource utilization. In: 2016 IEEE/ACM 9th international conference on utility and cloud computing (UCC). IEEE, pp 37–42
6. Li J, Ma X, Singh K, Schulz M, de Supinski BR, McKee SA (2009) Machine learning based online performance prediction for runtime parallelization and task scheduling. In: 2009 IEEE international symposium on performance analysis of systems and software. IEEE, pp 89–100
7. Kang S, Veeravalli B, Aung KMM (2018) Dynamic scheduling strategy with efficient node availability prediction for handling divisible loads in multi-cloud systems. J Parallel Distrib Comput 113:1–16
8. Ghobaei-Arani M, Jabbehdari S, Pourmina MA (2018) An autonomic resource pro-visioning approach for service-based cloud applications: a hybrid approach. Futur Gener Comput Syst 78:191–210
9. Choudhary A, Gupta I, Singh V, Jana PK (2018) A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Futur Gener Comput Syst 83:14–26
10. Zhang N, Yang X, Zhang M, Sun Y, Long K (2018) A genetic algorithm-based task scheduling for cloud resource crowd-funding model. Int J Commun Syst 31(1):e3394
11. Ober I, Palyart M, Bruel J-M, Lugato D (2016) On the use of models for high-performance scientific computing applications: an experience report. Softw Syst Model 1–24:2016
12. Monteiro AFG (2015) HPC management and engineering in the hybrid cloud. Doctoral dissertation, Universidade de Aveiro, Portugal, pp 1–153

13. Liu J, Pacitti E, Valduriez P, Mattoso M (2015) A survey of data-intensive scientific workflow management. J Grid Comput 13(4):457–493
14. Mastelic T, Fdhila W, Brandic I, Rinderle-Ma S (2015) Predicting resource allocation and costs for business processes in the cloud. In: 2015 IEEE world congress on services. IEEE, pp 47–54
15. Bala A, Chana I (2015) Intelligent failure prediction models for scientific workflows. Expert Syst Appl 42(3):980–989
16. Pandey S, Wu L, Guru SM, Buyya R (2010) A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 2010 24th IEEE international conference on advanced information networking and applications. IEEE, pp 400–407
17. Lati MSA, Abdul-Salaam G, Madni SHH et al (2016) Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. PLoS ONE 11(7):1–18
18. Kansal NJ, Chana I (2015) Artificial bee colony based energy-aware resource utilization technique for cloud computing. Concurr Comput Pract Exp 27(5):1207–1225
19. Emeakaroha VC, Netto MA, Calheiros RN, Brandic I, Buyya R, De Rose CA (2012) Towards autonomic detection of SLA violations in cloud infrastructures. Futur Gener Comput Syst 28(7):1017–1029
20. Behzadian M, Otaghsara SK, Yazdani M, Ignatius J (2012) A state-of the-art survey of topsis applications. Expert Syst Appl 39(17):13051–13069
21. Jahanshahloo GR, Lot FH, Izadikhah M (2006) An algorithmic method to extend topsis for decision-making problems with interval data. Appl Math Comput 175(2):1375–1384
22. Kaur G, Bala A (2021) OPSA: an optimized prediction based scheduling approach for scientific applications in cloud environment. Clust Comput 2021:1–20
23. Ramakrishnan L, Gannon D (2008) A survey of distributed work ow characteristics and resource requirements. Indiana University, pp 1–23
24. Ramakrishnan L, Plale B (2010) A multi-dimensional classification model for scientific workflow characteristics. In: Proceedings of the 1st international workshop on work-flow approaches to new data-centric science. ACM, p 4
25. Kaur G, Bala A, Chana I (2019) An intelligent regressive ensemble approach for pre-dicting resource usage in cloud computing. J Parallel Distrib Comput 123:1–12
26. Team RDC (2018) The R project for statistical computing. https://www.r-project.org/