Future Generation Computer Systems 34 (2014) 47–65

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud

Thamarai Selvi Somasundaram, Kannan Govindarajan *

Madras Institute of Technology, Anna University, Chennai, India

HIGHLIGHTS

- We propose scheduling framework for managing HPC applications in Science Cloud (SC).
- Introduce Particle Swarm Optimization based scheduling for allocation of resources.
- Designed Discrete Event Simulator in Matlab to simulate the proposed research work.
- Embarrassingly parallel and Iterative parallel application execution in SC.
- Minimize makespan, cost, job rejection ratio and maximize jobs meeting deadline.

ARTICLE INFO

Article history:
Received 9 January 2013
Received in revised form
12 October 2013
Accepted 3 December 2013
Available online 22 December 2013

Keywords:
Cloud computing
High Performance Computing (HPC)
CLOUD Resource Broker (CLOUDRB)
Resource allocation
Job scheduling
Particle Swarm Optimization (PSO)
Science cloud

ABSTRACT

In recent years, the Cloud environment has played a major role in running High-Performance Computing (HPC) applications, which are computationally intensive and data intensive in nature. The High-Performance Computing Cloud (HPCC) or Science Cloud (SC) provides the resources to these types of applications in an on demand and scalable manner. Scheduling of jobs or applications in a Cloud environment is NP-Complete and complex in nature due to the dynamicity of resources and on demand user application requirements. The main motivation behind this research study is to design and develop a CLOUD Resource Broker (CLOUDRB) for efficiently managing cloud resources and completing jobs for scientific applications within a user-specified deadline. It is implemented and integrated with a Deadline-based Job Scheduling and Particle Swarm Optimization (PSO)-based Resource Allocation mechanism. Our proposed approach intends to achieve the objectives of minimizing both execution time and cost based on the defined fitness function. It is simulated by modeling the HPC jobs and Cloud resources using the Matlab programming environment. The simulation results prove the effectiveness of the proposed research work by minimizing the completion time, cost and job rejection ratio and maximizing the number of jobs completing their applications within a deadline and meeting the user's satisfaction. The proposed work has been tested in our Eucalyptus-based cloud environments by submitting real-world HPC applications and observed the improvements in performance.

1. Introduction

Cloud Computing provides on demand and scalable delivery models to the users. The Cloud service delivery models are categorized into three major types viz., Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The IaaS service delivery model plays a major role in hosting SaaS, PaaS and Scientific applications or web applications in Cloud data centers. Furthermore, it provides the computational and storage resources to the scientific and business community using the open-source Cloud middleware tools such as Nimbus [1], Eucalyptus [2], OpenNebula [3] and Hadoop [4] in a dynamic manner. Cloud Computing has been extended to HPC applications as well. HPC applications are mainly focused on scientific applications that may fall into any one of the categories, specifically parallel, iterative parallel, embarrassingly parallel, data intensive and workflow applications. The HPC applications are complex in nature; they require large amounts of computational cycles that also involves vast amount of data processing. In this research work, our focus is mainly on parallel iterative and embarrassingly parallel (parallel sweep) types of jobs. Originally, Cluster Computing has evolved to solve those scientific application problems in terms of High Performance Computing (HPC). The introduction of Grid Computing allowed a collaborative problem solving technique to solve the complex problems using the resources which are globally distributed. However, the

* Corresponding author.
E-mail addresses: stselvi@annauniv.edu (T.S. Somasundaram), kannan.gridlab@gmail.com (K. Govindarajan).
URL: http://www.annauniv.edu/care (K. Govindarajan).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.024

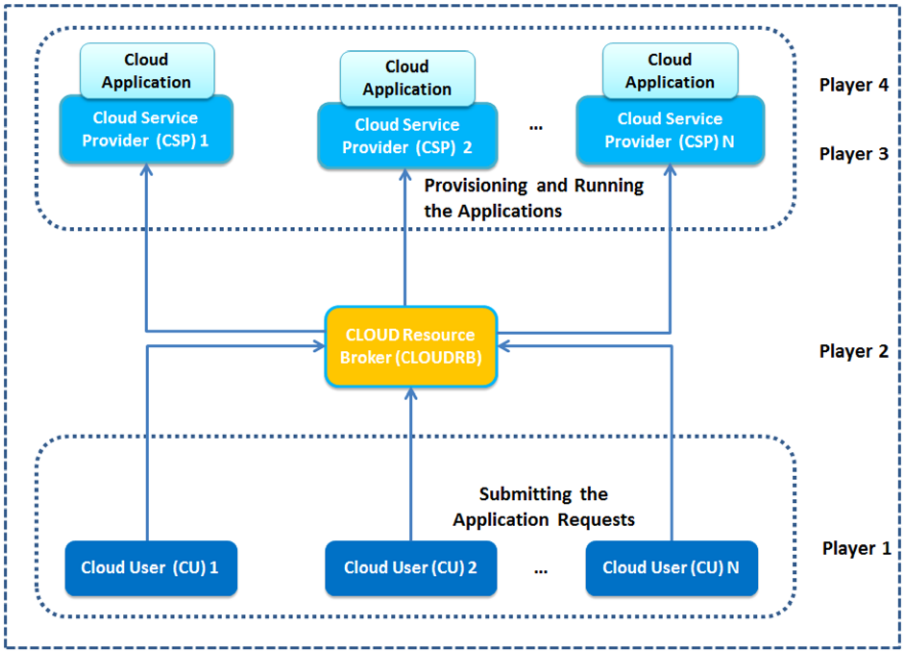


Fig. 1. Four major players of IaaS.

Grid Computing was unable to provision the resources as per the requirement of jobs in need of the software environment to execute the jobs. The Virtualization concept helped in provisioning Infrastructure from the Cloud resources in the form of IaaS. Thus, the emerging technology of Cloud Computing provides Cloud resources such as High Performance Computing Clouds (HPCCs) or Science Clouds (SCs). HPCCs provide the customized execution environment for scientific applications by employing the Virtualization Technology and Service Oriented Architecture (SOA) concepts in a dynamic manner.

1.1. Four major players in Cloud

The four major players in the Cloud environment are: (1) Cloud Users (CUs) (2) Cloud Service Providers (CSPs) (3) Cloud Applications (CAs) and (4) CLOUD Resource Brokers (CLOUDRBs). The interaction between the four major players is shown in Fig. 1. The CUs submit the jobs specifying the requirements of Software, Hardware and Quality of Service (QoS) parameters. The requirements vary in terms of Hardware (Processor Speed, RAM Memory, Bandwidth, etc.), Software (Mpich-1.2.7, Charm++ 3. X, FFTW-3. X, etc.) and QoS (Deadline, Throughput, etc.). The CSPs are responsible for managing the physical heterogeneous Cloud resources to host the Cloud applications which require computation, storage and network resources. The CAs may be of different types varying from simple web applications to complex high-performance computing applications. The Cloud Users are particular about completing their application's execution within a specified deadline and cost. To meet the objectives of CAs, CUs and CSPs we have proposed CLOUDRB. The Proposed CLOUDRB helps the CUs by selecting appropriate Cloud resources and enables the CSPs to deliver more profit with maximum resource utilization. To select the suitable Cloud resource a scheduling algorithm has been proposed and integrated with our proposed CLOUDRB to successfully complete the execution of tasks within the stipulated deadline with minimal time and cost.

Scheduling in Cloud Computing environments is considered to be an NP-complete problem [5]. The biologically inspired techniques are classified [6] into Evolutionary Technique, Swarm Intelligence (SI) and Ecology Technique. In recent years, the biologically inspired techniques are receiving greater attention to solve such types of NP-complete problems. Several researchers have investigated the SI-based biologically inspired algorithms such as Genetic Algorithm [7], Ant Colony Optimization [8] and Particle Swarm Optimization (PSO) [9] for solving optimization problems. PSO is one of the most popular biologically inspired optimization algorithms which come under the family of Swarm Intelligence (SI). It has been developed by Eberhart and Kennedy [10] in 1995. It uses the principle of social behavior of birds flocking or fish schooling. Moreover, it achieves a faster convergence rate [11,12] and global optimum solution [13] within minimal time as compared to ACO and GA. Some of the researchers [14–16] have investigated earlier for solving the job scheduling problem in grids and clouds using PSO because of the advantages over other algorithms mentioned above. However, most of the research papers are primarily focused on minimizing time [17], cost [15], energy [15,18] or time and cost [12] for workflow types of applications. Nevertheless, our proposed approach is mainly targeted to minimize the makespan and cost together for embarrassingly parallel and parameter sweep applications. Also, we have looked at the deadline as one of the important factors in our proposed job scheduling mechanism. Therefore, in this research work, we have made use of the advantages of Particle Swarm Optimization (PSO) for scheduling of jobs to the cloud resources for getting a near optimal solution. The jobs are first prioritized based on their deadline and resource selection is carried out by employing the population-based PSO algorithm. The proposed algorithm generates the possible schedule, and it builds all possible combinations of job requests with available cloud resources with the objectives of minimizing makespan and cost within the user specified deadline. We first formulate the scheduling problem in a cloud environment as an NP-complete problem; subsequently we have described the cloud-based architecture design of CLOUDRB. The proposed approach is simulated to test the efficacy of various performance metrics. Furthermore, we have tested the same with real-world HPC scientific applications in our cloud testbed. In brief, the contributions of the research study are summarized below:

- We have designed and developed a Deadline-based Job Scheduling algorithm for prioritizing the user job requests. (A)

- We have formulated and developed a Particle Swarm Optimization (PSO)-based Resource Scheduling algorithm for allocating the user requests to the Cloud resources in a near optimal manner. (B)
- We have simulated the proposed scheduling algorithm to validate the effectiveness of the proposed approach. (C)
- We have integrated (A) and (B) with CLOUD Resource Broker (CLOUDRB) for dynamic provisioning of Cloud resources to HPC applications in an effective way. (D)
- Finally, we have tested the proposed research work in our Cloud testbed by submitting real-world HPC applications. (E)

The rest of the paper is organized as follows: Section 1 starts with an introduction and motivation of the proposed work, Section 2 describes the related works that are closely related and provides background knowledge to our research work. Section 3 discusses the notations, problem conceptualization, problem statement, and scheduling model of our proposed work. Section 4 explains the proposed PSO scheduling algorithm, Section 5 presents the system architecture of the proposed framework; Section 6 describes the simulation and the results to validate our proposed work. Section 7 describes the implementation details, experimental setup and implementation of real-world applications. Section 8 concludes the proposed work and explores the feasibility of future employment.

2. Related works

Scheduling in Cloud Computing environments is an NP-complete problem [5]. However, in recent years, many heuristics and metaheuristic algorithms have been devised for doing effective scheduling in distributed computing environments. Some of the metaheuristic algorithms are nature-inspired, and some of them are non-nature inspired. Abramson et al. [19] proposed the economy based grid scheduling mechanism for parameter sweep applications in computational grid environments. It considers the resource cost, price and deadline. It has been integrated with the grid resource broker, namely NIMROD-G, to know the strength of the proposed approach. Nevertheless, in their proposed approach scheduling depends heavily on cost and deadline specified by the user. Biao Song et al. [20] proposed a task selection and resource allocation framework in a cloud environment. The resource allocation problem is classified into two types, namely, heavy workload and light workload condition. The threshold value is fixed for allocating tasks to resources with an objective to increase the resource utilization. QoS and resource utilization are the major factors considered in the heavy workload and light workload condition respectively. Hai Zhong et al. [21] proposed the optimized resource scheduling mechanism for open-source cloud systems using the Improved Genetic Algorithm (IGA). The main contributions of their research work are to derive the fitness function using dividend policy mechanism for improving the utilization of the Cloud resources and minimize the energy consumption. They have compared their proposed approach to First Fit and Round Robin scheduling mechanisms.

Pandey et al. [12] presented a Particle Swarm Optimization (PSO)-based scheduling mechanism for data-intensive workflow types of applications. Their scheduling mechanism considers the execution cost and data-transfer cost and they have compared their proposed approach with the Best Resource Selection (BRS) technique. The results make it evident that their proposed approach saves three times more cost than BRS. However, our proposed approach is considered for HPC applications which are iterative parallel and embarrassingly parallel, where the jobs are independent in nature. The fitness function we have derived balances both the cost and time. Our proposed approach has been compared with the Rank-Based Allocation and other popular heuristic approach of Ant Colony Optimization (ACO). However, their proposed approach is minimizing the makespan of tasks available in workflow applications; whereas ours is minimizing the makespan of independent jobs submitted to resource broker per schedule. Moreover, our proposed approach considers deadline as one of the important factors for prioritizing the jobs submitted to cloud resource broker. Byuna et al. [22] proposed a Partition Time Balance Schedule (PBTS) based scheduling mechanism for optimal performance of workflow applications in a cloud environment with minimal cost. Their scheduling mechanism schedules the tasks by minimizing the number of cloud instances in each time slot. Mezmaz et al. [13] proposed the parallel bi-objective hybrid genetic algorithm. Their approach is proposed to minimize the makespan and energy consumption for precedence-constraint parallel applications. Sonmez et al. [23] proposed a novel economy driven job scheduling and the simulation-based experiments have been established to prove the effectiveness of the proposed approach for parameter sweep and sequential workflow type of applications. Jin Xu et al. [24] implemented the metaheuristic technique of Chemical Reaction Optimization (CRO) for Task Scheduling in the dynamic Grid environment. They have minimized the makespan and flowtime by taking the workload and reliability of the resources into account. They have simulated their proposed approach using both permutation and vector based representations of job-resource mapping. The simulation results have shown that the performance of CRO based scheduling is comparatively better in vector-based representation than permutation-based representation.

Liang Hu et al. [25] proposed the Parallel Hybrid Particle Swarm Optimization (PH-PSO) algorithm for resource prediction system in Grid computing environments. PH-PSO is hybridized from both continuous PSO and binary PSO. It is employed with inputs of hyper parameter selection and feature selection for enhancing the accuracy and efficiency prediction subsystem. Their algorithm starts with a population of random particles and searches a multidimensional solution space for an optimal solution by updating the generation of particles. Our proposed approach is different from their approach by considering the precise particles (i.e. matched resource list for jobs) with continuous PSO for near optimal resource selection in the Cloud environment. Siriluck Lorpunmanee et al. [26] proposed an ACO based job scheduling mechanism in Grid environments. The proposed approach is aimed to effectively utilize the processor and minimize the tardiness (delay time) for the submitted jobs. Jia Yu and Rajkumar Buyya [27] have proposed the GA-based scheduling mechanism for scientific workflow applications on Utility Grids by considering the deadline and budget constraints. Fengguang Tian and Keke Chen [28] proposed the optimal resource provisioning that is mainly aimed to minimize the cost and time for running MapReduce programs in public Cloud environments. The price is calculated based on read/write, partitioning and sorting operations. Chaisiri et al. [29] proposed the Optimal Cloud Resource Provisioning (OCRP) algorithm based on the stochastic integer programming model. They have classified the cost into two categories: cost for on demand plan and cost for advance reservation plans. The total resource provisioning cost is computed by summing these two cost values and the resources are taken based on the minimal cost. Tasgetiren et al. [30] have proved that PSO is able to improve 57 out of 90 best-known solutions provided by other well-known algorithms to solve the sequencing problems. Our proposed approach considers both the deadline (time) and budget (cost) for iterative parallel and embarrassingly parallel (parameter sweep) applications with the faster convergence rate in a dynamic Cloud environment.

Mohammad Izuddin Nordin [31] proposed the Goal-based Cloud Resource Broker for managing the job requests related to

medical applications. The main idea of their proposed work is to optimize the usage of Cloud resources in an efficient manner. CARE Resource Broker (CRB) [32] is a Grid metascheduler or resource broker that is deployed over the Globus Toolkit 4 (GT4) [33] middleware to manage the Grid and Virtualization enabled Grid resources. It addresses several scheduling scenarios encountered when Virtualization technology integrates with the Grid environment. It has the support for creation and management of virtual resources in Virtualization enabled Grid resources. Elastic Site Manager [34] is a resource manager, and it is able to dynamically provision the resources for running scientific applications from both public as well as private Clouds. The ASKALON [35] Grid has extended to support the workflow application execution in both Grid and Cloud that can be either public or private. Their proposed Workflow Toolkit is capable of performing the discovery, data access, service invocation, and execution of workflows in the heterogeneous resources. Huang et al. [36] have proposed that virtual machines are most useful for providing users a portion of system resources and different execution/software environment than the host machine. They have added that virtual machines are helpful for HPC applications when the required environment is different from the host environment. The performance of applications running in the virtual machines is almost equal to the real physical machines. Christian Vecchiola et al. [37] proposed the resource provisioning approach using hybrid clouds. Their proposed approach is tested using the Aneka platform with a Multi-Objective Evolutionary Optimizer called EMO for CPU-intensive tasks.

3. Problem statement and problem formulation

We have stated the scheduling problem in Cloud as "*Jobs and Resources need to be allocated and scheduled in such a way that CUs can complete their jobs with minimal time (within the deadline) and cost and maximize the user satisfaction and throughput of CRP*". In a broader sense, the user is expected to complete his/her jobs with minimal time and minimal cost. However, faster job execution requires high-end computational resources (costlier resources). Hence, the objective of completing applications with minimal time and cost in Cloud scheduling becomes a multi-objective problem and complex in nature. The CLOUDRB receives N number of job requests $J_1, J_2, J_3, \ldots, J_N$ which are iterative parallel and independent in nature. Every job has Job Length (JL_{J_i}) that is expressed as Millions of Instructions (MI) and $Deadline_{J_i}$. Every job requires p number of processing nodes, q amount of processor speed, r amount of RAM Speed, $s$amount of Storage Space (Hard disk space) and t amount of bandwidth. The Cloud resources registered with CLOUDRB has M number of resources $R_1, R_2, R_3, \ldots, R_M$ which are heterogeneous in nature in terms of Processor Speed, RAM memory, Hard disk Memory, Bandwidth, etc. Based on the job requirements and availability of Cloud resources, CLOUDRB generates the matched resource list (ϕ) for all the jobs available in a schedule (S_i) that may contain all the 'M' resources or less than that. If a resource R_j is matched for the job J_i then the decision variable $x_{J_iR_j}$ is 1 else it is 0. CLOUDRB computes two matrices namely (i) $EET_{J_iR_j \in \Phi_{J_iS_p}}$ and (ii) $EECost_{J_iR_j \in \Phi_{J_iS_p}}$ and the dimension of each matrix is $\left|\Phi_{J_iS_p}\right| \times 1$.

$$EET_{J_iR_j \in \Phi_{J_iS_p}} = \left(EECost_{J_iR_1} EECost_{J_iR_2} \cdots EECost_{J_iR_M}\right)^T$$

$$EECost_{J_iR_j \in \Phi_{J_iS_p}} = \left(ET_{J_iR_1} ET_{J_iR_2} \cdots ET_{J_iR_M}\right)^T.$$

Here, $\left|\Phi_{J_iS_p}\right|$ represents the number of resources in the matched resource list Φ for the job J_i. The initial workload of the Cloud resources $W_1, W_2, W_3, \ldots, W_M$ for the matched resources $R_1, R_2, R_3, \ldots, R_M$ is $W_{R_j} = 0$. The notations and their descriptions used in our proposed system are represented in Table 1.

Table 1
Notations and its description.

Notations	Description
$S_1, S_2, S_3, \ldots, S_K$	Represents the schedule from 1 to K
S_p	Represents pth schedule
$J_1, J_2, J_3, \ldots, J_N$	Job request 1–N submitted for schedule
$R_1, R_2, R_3, \ldots, R_M$	Cloud resource 1–M available for schedule
$MIPS$	Millions of instructions per second
$\Phi_{J_iS_p}$	Represents a list of matched resources for job J_i for schedule S_p
$\left\|\Phi_{J_iS_p}\right\|$	Represents the number of matched resources for job J_i for schedule S_p
$BT_{J_iR_j \in \Phi_{J_iS_p}}$	Boot time for job J_i on resource R_j in the matched cloud resource $\Phi_{J_iS_p}$
$TT_{J_iR_j \in \Phi_{J_iS_p}}$	Transfer time for job J_i on resource R_j in the matched cloud resource $\Phi_{J_iS_p}$
$ET_{J_iR_j \in \Phi_{J_iS_p}}$	Execution time for job J_i on resource R_j in the matched cloud resource $\Phi_{J_iS_p}$
$EET_{J_iR_j \in \Phi_{J_iS_p}}$	Expected execution time for resource R_j to process job J_i
$TET\ (S_p)$	Total execution time for the jobs available in schedule S_p
$ECost_{J_iR_j \in \Phi_{J_iS_p}}$	Execution cost for the job J_i on R_j in the matched cloud resource $\Phi_{J_iS_p}$
$TCost_{J_iR_j \in \Phi_{J_iS_p}}$	Transfer cost for the job J_i on R_j in the matched cloud resource $\Phi_{J_iS_p}$
$EECost_{J_iR_j \in \Phi_{J_iS_p}}$	Expected execution cost for resource R_j to process job J_i
$TECOST\ (S_p)$	Total execution cost for the jobs available in the schedule S_p
JL_{J_i}	Length of job J_i in $MIPS$
FS_{J_i}	File size required for the job J_i
PC_{R_j}	Processing capability of the resource R_j in MI
BW_{R_j}	Bandwidth available in the resource R_j
D_{R_j}	Delay rate for the resource R_j
$x_{J_iR_j}$	The decision variable to represent whether the resource R_j is in the matched resource list for the job J_i
W_{R_j}	Workload of the resource R_j
N_k	Number of applications submitted by the user k
JM_{S_p}	Number of jobs for which matched resources have been found in S_p
JS_{S_p}	Number of jobs which has been scheduled in S_p
US_k	User satisfaction for the user k
DoD_k	Degree of satisfaction for the user k based on deadline
DoC_k	Degree of satisfaction for the user k based on cost
$Budget_{J_i}$	Actual cost for the job J_i
$Deadline_{J_i}$	Deadline specified by the job J_i
$FinishTime_{J_i}$	Finishing time of the job J_i
WT_{J_i}	Waiting time of the job J_i
$ActualCostIncurred_{J_i}$	Actual cost to be spent or incurred to process the job J_i

3.1. Objective functions and fitness function

The objective functions implemented with the proposed scheduling algorithm attempt to maximize or minimize the value. We have defined two objective functions with the aim of minimizing time and cost and it is integrated with the scheduling algorithm. The first objective is the minimization of total execution time of jobs $J_1, J_2, J_3, \ldots, J_N$ submitted in a particular schedule S_p and it is represented using (1).

$$Objective\ function\ 1 : Min\ TET(J_i) = EET_{J_iR_j \in \Phi_{J_iS_p}} \quad (1)$$

$$ET_{J_iR_j \in \Phi_{J_iS_p}} = JL_{J_i}/PC_{R_j} \quad (2)$$

$$TT_{J_iR_j \in \Phi_{J_iS_p}} = FS_{J_i}/2 * D_{R_j} * BW_{R_j} \quad (3)$$

$$EET_{J_iR_j \in \Phi_{J_iS_p}} = ET_{J_iR_j} + TT_{J_iR_j} + BT_{J_iR_j} \quad (4)$$

$$\Phi_{J_iS_p} \quad \text{where } \left|\Phi_{J_iS_p}\right| \leq M. \quad (5)$$

Here, $EET_{J_iR_j \in \Phi_{J_iS_p}}$ is the sum of execution time $\left(ET_{J_iR_j \in \Phi_{J_iS_p}}\right)$, transfer time $\left(TT_{J_iR_j \in \Phi_{J_iS_p}}\right)$ and booting time $\left(BT_{J_iR_j \in \Phi_{J_iS_p}}\right)$. The matched resource list generated for each job J_i is represented using (5) that consists of resources less than or equal to the total number of available Cloud resources 'M'.

The second objective is the minimization of the total execution cost $TECOST(S_p)$ of jobs $J_1, J_2, J_3, \ldots, J_N$ submitted in a particular schedule S_p and it is represented using (6)

$$Objective\ function\ 2 : Min\ TECOST(J_i) = EECost_{J_iR_j \in \Phi_{J_iS_p}} \tag{6}$$

$$ECost_{J_iR_j \in \Phi_{J_iS_p}} = ET_{J_iR_j \in \Phi_{J_iS_p}} * (Cost) \tag{7}$$

$$TCost_{J_iR_j \in \Phi_{J_iS_p}} = TT_{J_iR_j \in \Phi_{J_iS_p}} * (Cost) \tag{8}$$

$$EECost_{J_iR_j \in \Phi_{J_iS_p}} = ECost_{J_iR_j \in \Phi_{J_iS_p}} + TCost_{J_iR_j \in \Phi_{J_iS_p}}. \tag{9}$$

Here, $ECost_{J_iR_j \in \Phi_{J_iS_p}}$ is the sum of execution cost using (6) and data transfer cost using (7). The fitness function $f(w)$ (10) is defined to solve the above-said scheduling multi-objective optimization problem in Cloud.

Fitness function:

$$Min\, f(R_j) = w1 * x_{J_iR_j} * EET_{J_iR_j} + w2 * x_{J_iR_j} * EECost_{J_iR_j}. \tag{10}$$

Subject to constraints

$$\Phi_{J_iS_p} \leq \{M\} \tag{11}$$

$$R_j \in \Phi_{J_iS_p}, \quad x_{J_iR_j} = 1 \tag{12}$$

$$R_j \notin \Phi_{J_iS_p}, \quad x_{J_iR_j} = 0 \tag{13}$$

$$w1 + w2 = 1 \tag{14}$$

where $w1$, $w2$ represents the weight assigned to execution time, execution cost.

The time required for completing the jobs running in the Cloud resource is derived using (15), and the job will be assigned to the resource only if it satisfies (16).

$$RT_{J_iR_j \in \Phi_{J_iS_p}} = Deadline_{J_i} - W_{R_j \in \Phi_{J_iS_p}} \tag{15}$$

$$ET_{J_iR_j \in \Phi_{J_iS_p}} \leq RT_{J_iR_j \in \Phi_{J_iS_p}}. \tag{16}$$

Once the jobs are allocated to the Cloud resources the workload of the resources is calculated using (17).

$$W_{R_j} = W_{R_j} + ET_{J_iR_j}. \tag{17}$$

The total time for Resource R_j to complete all the jobs allocated to the particular resource is represented using (18).

$$ms_{R_j} = \sum_{J_i \in \varphi_{R_j}} EET_{J_iR_j}. \tag{18}$$

Here, φ_{R_j} represents the list of jobs assigned to resource R_j. The makespan value or total time taken for a schedule to complete all the jobs is represented using (19).

$$Makespan = \max\{ms_{R_j}\}. \tag{19}$$

3.2. Scheduling model

Pugliese et al. [38] have analyzed the requirements of Grid Resource Management and classified the schedulers into various types. In addition to that, they have defined the model for Grid Scheduling that comprises Job Model, Resource Model, Performance Metrics, Scheduling Policy and Programming Model. We have inherited and held out their scheduling model for our proposed scheduling approach in a Cloud environment as shown in Fig. 2.

A. *Job model*—It provides the abstraction of jobs and its requirements. The requirements are classified into Hardware, Software and QoS. The jobs are classified into various types such as sequential, parallel, parameter sweep, iterative parallel, data intensive and workflow jobs. The jobs are modeled as

<JobID, ExecutableID, InputFileID, OutputFileID, HParametersID, SParametersID, QoSParametersID, OSParametersID>

Here, ExecutableID represents the executable files required for application execution. InputFileID and OutputFileID represents the input and output files required/generated for application execution. HParametersID, SParametersID, QoSParametersID and OSParametersID represent the hardware requirements (processor speed, RAM memory, number of nodes etc.), software requirements (Mpich-1.2.7, FFTW-3. 2. X etc.), QoS requirements (deadline 25-09-2012 etc.) and operating system requirements (Cent OS 5.5 etc.) respectively.

B. *Resource model*—It describes the characteristics and performances of data centers, hosts, operating system images, software libraries and network links. The resource model maintains the metadata that contains resource characteristics in terms of properties and value.
   - A datacenter is modeled as <DataCenterID, Description>. Here, Description represents the datacenter name, datacenter locality (availability zone) and its associated hosts.
   - A host is modeled as <HostID, Description, DataCenterID>. Here, Description represents the hostname, processor speed, RAM memory, Hard disk Memory, Operating System, etc.
   - An operating system image is modeled as <OperatingSystemImageID, Description, HostID, DataCenterID>. Here, Description represents the operating system name, version and release. HostID represents the Cloud host where the operating system image is residing.
   - A Software Library component is modeled as <SoftwareID, Description, OperatingSystemImageID, HostID, DataCenterID>. Here, Description represents the software name, software version, associated operating system, host and datacenter.

C. *Scheduling policy*—The scheduling policy is implemented with Scheduler based on specific goals of optimization of time or cost or both. It operates based on characteristics of resource and workload data. It is possible to integrate heuristics or metaheuristic based scheduling. It is separated into two types such as job scheduling and resource scheduling policy. The proposed work consists of Deadline-based Job Scheduling policy and PSO-based Resource Scheduling policy.

D. *Performance metrics—It is used to evaluate the performance improvements gained by the proposed scheduling model.* In our proposed work, we have evaluated the various performance metrics such as makespan, number of job's meeting deadline, execution time, execution cost, job rejection ratio and user satisfaction.

E. *Programming model*—The programming model is helpful for providing the interface to the scheduler. In our proposed approach, we have made use of web services based programming model for interfacing the scheduler with other components.

4. Proposed particle swarm optimization (PSO) algorithm

Particle Swarm Optimization (PSO) is a population-based artificial intelligence mechanism that is inspired by the social behavior of a swarm of birds. The PSO algorithm is employed with the swarm of particles, and every particle is responsible for tracking the fitness of each particle. Every particle is associated with corresponding velocity that helps the particle to move on to best position and every particle in the swarm searching the multidimensional solution space to search for a better solution. Let us consider there are T particles where $T \leftarrow \left|\Phi_{J_iS_p}\right|$ in a D–Dimensional search space.

$$\Phi_{J_iS_p} = \{R_1, R_2, R_3, \ldots, R_k\} \quad \text{where } k \leq M. \tag{20}$$

Here, $\Phi_{J_iS_p}$ contain '$\left|\Phi_{J_iS_p}\right|$' number of Cloud resources which are satisfying the user application requirements and each resource has dimensions, i.e. $D = 1, 2$.

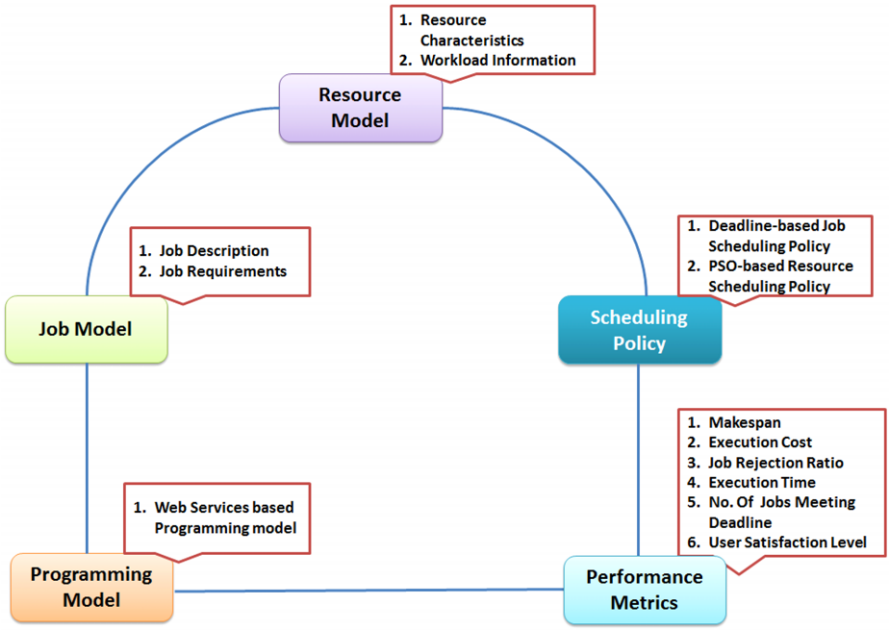


Fig. 2. Scheduling model for cloud environment.

1. The size of the swarm is $T \times D$ where each row represents the time and cost for the cloud resources present in $\Phi_{J_iS_p}$.

$$\text{Swarm}_{T\times D} = \begin{pmatrix} \text{EET1} & \text{ECT1} \\ .. & .. \\ \text{EET}_T & \text{ECT}_T \end{pmatrix}.$$

2. Each particle in the swarm is initialized with the velocity where the size of the velocity matrix is the same as that of the size of the swarm.

$$\text{Velocity}_{TXD} = \begin{pmatrix} V_{11} & V_{12} \\ .. & .. \\ V_{T1} & V_{T2} \end{pmatrix}.$$

3. $\text{PBest}_{T\times D}$ represent the personal best position of the swarm where the size of the personal best position is same as that of the size of the swarm.

$$\text{PBest}_{T\times D} = \begin{pmatrix} \text{Pb}_{11} & \text{Pb}_{12} \\ .. & .. \\ \text{Pb}_{T1} & \text{Pb}_{T2} \end{pmatrix}.$$

4. $GBest_{Swarm}$ represent the global best position of the swarm.

$$GBest_{Swarm} = (G1\ G2)\,.$$

In each iteration i, the velocity of each particle in each dimension is calculated using (21). The particle new position in each dimension is determined based on the Eq. (22).

$$\begin{aligned} V(p, i+1) &= V(p, i) + w * (rand_1 + C1 \\ &\quad * (PBestPosition(p, i) - R(p, i))) \\ &\quad + w * (rand_2 + C2 * (GBestPosition(p, i) \\ &\quad - R(p, i))) \end{aligned} \tag{21}$$

$$R(p, i+1) = R(p, i) + V(p, i). \tag{22}$$

The parameter 'w' represents the inertia factor and it plays a major role for convergence. $C1$, $C2$ represent the personal learning and social learning factory respectively. $rand_1$ and $rand_2$ are random numbers which are $\in_u \{0, 1\}$. The pseudo code for the particle swarm optimization algorithm is illustrated in Algorithm 1.

Algorithm 1: Particle Swarm Optimization (PSO) Algorithm

Input: $|\Phi_{J_iS_p}|$ number of matched resources for the job J_i on the schedule S_p
Output: Resource R_j that is near optimal for job J_i
For each resource in $\Phi_{J_iS_p}$
  Personal Best Value $R_j \in \Phi_{J_iS_p} \leftarrow \infty$
  Global Best Value for the swarm $\Phi_{J_iS_p} \leftarrow \infty$
  Personal Best Position, $R_j \in \Phi_{J_iS_p}$ having two dimensions $\leftarrow (0, 0)$
  Global Best Position for the swarm $\Phi_{J_iS_p} \leftarrow (0, 0)$
  Velocity for each R_j in the swarm
    $\Phi_{J_iS_p} \leftarrow -rand \in_u \{-1, 0\}$
  Assign for each R_j in the swarm
    $\Phi_{J_iS_p} \leftarrow \left(EET_{J_iR_j\in\Phi_{J_iS_p}}, EECost_{J_iR_j\in\Phi_{J_iS_p}}\right)$
End
  Repeat until Max iteration
    For each particle R_j in $\Phi_{J_iS_p}$
    $f(R_j \in \Phi_{J_iS_p}) \leftarrow w1 * x_{J_iR_j} * EET_{J_iR_j} + w2$ $*x_{J_iR_j} * EECost_{J_iR_j}$
    $pbestval_{R_j} \leftarrow \min\left\{f(R_j \in \Phi_{J_iS_p}), pbestval_{R_j}\right\}$
    $pbestpos_{R_j} \leftarrow \left(EET_{J_iR_j\in\Phi_{J_iS_p}}, EECost_{J_iR_j\in\Phi_{J_iS_p}}\right)$
    If $ET_{J_iR_j\in\Phi_{J_iS_p}} < RT_{J_iR_j\in\Phi_{J_iS_p}}$ then
      $gbestval_{\Phi_{is}} \leftarrow \min\left\{f\left(R_{j\in\Phi_{J_iS_p}}\right), gbestval_{\Phi_{J_iS_p}}\right\}$
      $gbestpos_{\Phi_{is}} \leftarrow \left(EET_{J_iR_j\in\Phi_{J_iS_p}}, EECost_{J_iR_j\in\Phi_{J_iS_p}}\right)$
    End
  End
  For each particle R_j in $\Phi_{J_iS_p}$
    $\overrightarrow{v(t+1)_{R_j}} = \overrightarrow{v(t)_{R_j}} + w$ $* \left(rand + C_1 * \left(\overrightarrow{pbestpos_{R_j}} - \overrightarrow{x_{R_j}}\right)\right)$ $+w * \left(rand + C_2 * \left(\overrightarrow{gbestpos_{\Phi_{is}}} - \overrightarrow{x_{R_j}}\right)\right)$
    $\overrightarrow{x_{R_j}(t+1)} = \overrightarrow{x_{R_j}(t)} + \overrightarrow{v_{R_j}(t+1)}$
  End
End (of max iteration)

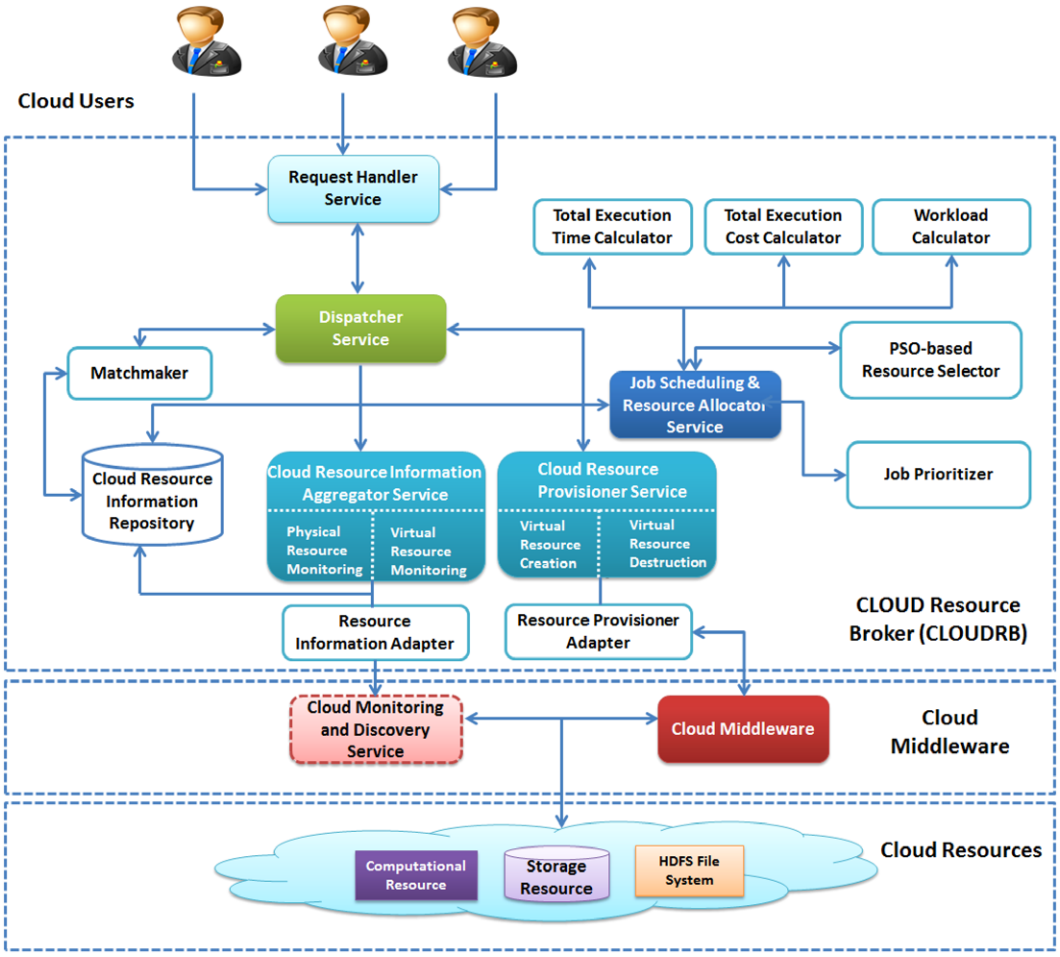


Fig. 3. Cloud based architecture design of CLOUDRB.

5. Architecture design of cloud resource broker (CLOUDRB)

In this section, we present the proposed Cloud based architectural design of CLOUDRB and various components as shown in Fig. 3.

5.1. Request handler service

The user submits the job requirements as an XML file or through a Graphical User Interface (GUI). The parser in the request handler parses the job requirements and the parsed information is updated in the job queue. Consequently, it invokes the matchmaker to match the user job requirements with the available cloud resources. The pseudo code for matchmaking Algorithm 2 is integrated with request handler that filters the potential resources that can create virtual resources capable of accomplishing the job. In the end, it generates the list of matched resources, and the same is sent to the Job Scheduling and Resource Allocator Service.

Algorithm 2: Pseudo Code for Matchmaking Algorithm

1. For each $J_i \in \{J_1, J_2 \ldots J_N\}$
2. $\Phi_{J_i S_p} \leftarrow \{\}$
3. For each $R_j \in \{R_1, R_2 \ldots R_M\}$
4. If J_i. GetNodeCount () * J_i. GetRAMSpace () $\leq R_j$. GetFreeRAMSpace () && J_i. GetNodeCount () * J_i. GetHardDiskSpace () $\leq R_j$. GetFreeHardDiskSpace () && J_i. GetProcessorSpeed () $\leq R_j$. GetProcessorSpeed () then $\Phi_{J_i S_p} = \Phi_{J_i S_p} \cup \{R_j\}$
5. End
6. End
7. Return the matched resource list $\Phi_{J_i S_p}$
8. End

5.2. Job scheduling and resource allocator service

This service is implemented with major components viz., Total Execution Time Calculator, Total Execution Cost Calculator, Workload Calculator, Job Prioritizer and PSO based Resource Selector. Workload Calculator calculates the workload of the resources based on the number of jobs and the remaining time to complete the assigned/running jobs. Job Prioritizer prioritizes the jobs which have matched resources. The priority is calculated based on the deadline which is given by users. The PSO based resource allocator allocates the resource to the jobs has four-fold processes such as (i) Initial job assignment in the matched resource list (ii) Calculation of Expected Execution Time (EET), Expected Execution Cost (EECost) and Workload for the matched resource list (iii) Calculate the fitness value and (iv) Final job assignment

Table 2
Job details.

Job Id	Number of nodes	RAM (GB)	Hard disk (GB)	Deadline	Length of job (MI)	File size (GB)
1–*N*	5–500	2–200	10–500	60–150	100 000–250 000	2–100

Table 3
Cloud resource details.

Resource Id	Number of nodes	RAM (GB)	Hard disk (GB)	Processing capability (MIPS)	Bandwidth (MBPS)	Delay (s)
1–*M*	100–500	20–200	1000–5000	2000–3000	100 000–250 000	0.01–1

to the cloud resource which has minimal fitness value. Let ϕ_{R_j} represent the list of jobs assigned to resource R_j and it is an empty set initially. The pseudo code for generating the job schedule is given in Algorithm 3.

Algorithm 3: Pseudo Code for Jobs Generation and Allocation of Cloud Resources

1. Sort the jobs based on the Deadline
2. For each job $J_i \in \{J_1, J_2 \ldots J_N\}$
3. $\varphi_{R_j} \leftarrow \{\}$
4. For each $R_j \in \Phi_{J_i S_p}$
5. $ET_{J_i R_j} = JL_{J_i}/PC_{R_j}$
6. $EECost_{J_i R_j} = ECost_{J_i R_j} + TCost_{J_i R_j}$
7. End
8. $R_j \leftarrow PSO(\,)$
9. $W_{R_j} = W_{R_j} + ET_{J_i R_j}$
10. $\varphi_{R_j} = \varphi_{R_j} \cup \{J_i\}$
11. End

5.3. Dispatcher service

The dispatcher is the core element that acts like a controller for CLOUDRB. It receives the requests from/to the following services such as Request Handler, Job Scheduling and Resource Allocator, Cloud Resource Provisioner (CRP) and Cloud Resource Information (CRI) Aggregator. It invokes the appropriate services in the CLOUDRB based on the request received or the event occurred.

5.4. Cloud resource information aggregator service

This component invokes the CMDS service in a periodic interval of approximately (∼45 s) to aggregate the available cloud resource's static and dynamic information. The static attributes represent the processor speed, operating system, etc., and the dynamic attributes represent the processor load, latency, number of virtual machines running, etc. The aggregated information is updated in the Cloud Resource Information Repository (CRIP).

5.5. Cloud resource provisioner service

Resource provisioning is the process of provisioning of virtual resources from the cloud environment. In our proposed work, we have made use of the private cloud resources created by using Eucalyptus middleware. Eucalyptus is an open-source software framework that is helpful for creating IaaS clouds. It interacts with the Cloud middleware to provision the virtual machine instances. It fetches the virtual machine request with the parameters such as type of instances, RAM capacity and the number of instances to be created to run the job. The Eucalyptus API [39] is compatible with Amazon EC2 [40], and it is primarily used for managing private cloud infrastructure. It supports Xen and KVM hypervisors in the open-source version. Typical APIs [41] are used for interfacing the Eucalyptus cloud middleware with the resource broker to retrieve the resource information, upload the images and provision the virtual resources.

5.6. Cloud monitoring and discovery service (CMDS)

The CMDS [42] is deployed along with Eucalyptus Cloud Controller to monitor and discover the resource information. It is implemented as SOAP based web service with the Web Service Resource Framework (WSRF) properties to maintain the cloud resources information in a stateful manner. It acts as a registry server and it is deployed in the cloud controller node. It is responsible for collecting the available resource information from cluster controllers. It makes use of the two popular open source external information providers such as Ganglia and Network Weather Service (NWS). The collected information is validated using Cloud Resource Information Schema (CRIS) which is based on Grid Uniform Laboratory Extension (GLUE) [43] schema mainly used in a distributed environment. The collected information is updated in the Cloud Resource Repository (CRP) in CLOUDRB.

6. Simulation and its inferences

6.1. Simulation details

We have simulated the proposed scheduling algorithm with CLOUDRB explained in this paper. The results are compared with other popular metaheuristic techniques of Ant Colony Optimization (ACO) and Genetic Algorithm (GA) and the greedy technique of the Rank based Allocation (RBA) method. The experimental model is implemented by mapping one CLOUDRB with multiple Cloud resources. The experimentation randomly generates the Cloud resources of 100–500 with different capabilities available in terms of number of CPUs, RAM, Bandwidth, Delay, Processing Capability, Processor Speed and Hard disk Memory. Each cloud resource has N_h number of hosts, and every host is capable of creating N_v number of virtual machine instances. The proposed system is accessed by N_u numbers of users arriving at a regular interval of λ based on the Poisson distribution. The job parameters are length of the job (JL_{J_i}), number of nodes required ($Nodes_{J_i}$), number of jobs (N_k) per user, deadline ($Deadline_{J_i}$), Processor Speed (PC_{J_i}), RAM (RAM_{J_i}) and Hard Disk Memory (HD_{J_i}) are generated using the Fietelson model [38]. The random number generator used in the simulation approach generates 200–1000 job requests in a random fashion that selects the resources based on the capacity of generating Cloud resources. The generated HPC jobs, Cloud resources and Cost details of the usage of Cloud resources are shown in Tables 2–4 respectively.

The parameters used for the PSO algorithm are shown in Table 5. We set the maximum number of iteration values at 1000. The number of particles considered in our proposed approach is not a random value and it is the dynamic value that depends upon the number of matched resources per job. We fixed the value of $c1 + c2 \leq 4$, i.e., social learning and personal learning factor [44,45] should be less than or equal to 4.

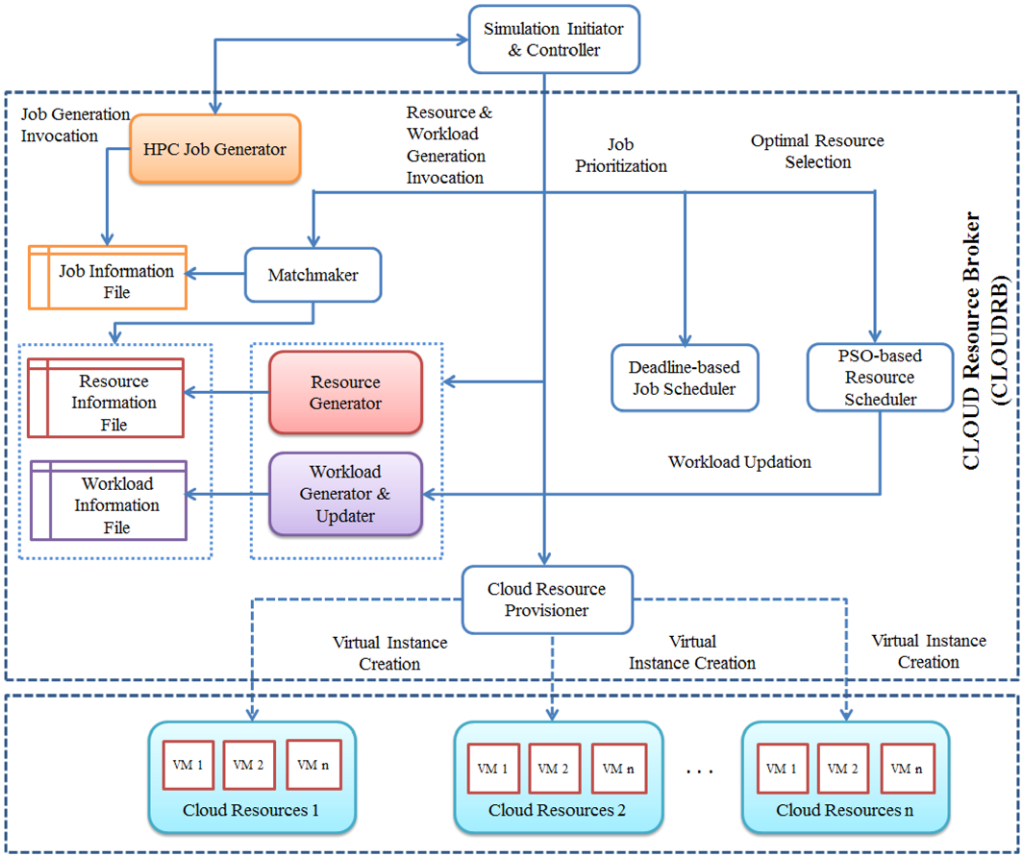


Fig. 4. Simulation architecture.

Table 4
Cost details.

Execution cost (per hour)	Transfer cost (per hour)	Storage cost (per GB/hour)
0.35$–1$	0.1$–1$	0.45$–1$

Table 5
PSO parameter details.

Max. number of generations	Number of particles	Social learning factor $c1$	Personal learning factor $c2$	Inertia weight $w1$	Inertia weight $w2$
1000	$\lvert\Phi_{J_i S_p}\rvert$	2	2	0.5	0.5

6.2. Simulation architecture

The simulation architecture implemented to test our proposed work using the Matlab programming environment is depicted in Fig. 4.

The Simulation Initiator and Controller is the core component which initiates and controls the whole simulation process. It first invokes the HPC Job Generator for generating the job requests, subsequently it invokes the Resource and Workload Generator for generating the Cloud resources and workload information. Once the job, resource and workload information have been generated, it invokes the matchmaker for performing the job resource mapping. The matchmaker performs the matching process for the job requests with available Cloud resource information and generates the matched resource list. The jobs that are completed by the matchmaking process are sent to the deadline-based job scheduler to prioritize the jobs. Then, the prioritized jobs with their matched resource list is sent to the PSO-based resource scheduler. The PSO-based resource scheduler computes the execution time, execution cost and workload value. The computed values are given as input to the proposed PSO algorithm. It selects the Cloud resource for provisioning of the virtual instance, which has a minimal fitness value. Once the jobs have been allocated to the Cloud resource the workload information is updated through the workload generator and updater component.

6.3. Simulation results and discussion

In this section, we have discussed the simulation results that include the minimization of time (makespan), cost and job rejection ratio and maximization of the number of jobs meeting deadline and user satisfaction. The derived fitness function assigns equal weights to execution time and cost during the optimization process. There is always a trade-off between time and cost. Our proposed research work solves the trade-off issue between time and cost by assigning an appropriate weighting to those two factors. The performance of the proposed PSO is compared with most popular metaheuristic techniques of Genetic Algorithm (GA) and Ant Colony Optimization (ACO) and Rank-Based Allocation (RBA), a greedy technique. Ant Colony Optimization (ACO) [46] is a population-based metaheuristic algorithm and the main principles are guided search i.e. an ant is always updating its own local pheromone and global pheromone. GA [47] works on survival of the fittest principle. In GA, the solutions are represented as chromosomes; it is evaluated and ranked based on the sorted fitness value. The operation such as selection, crossover, and mutation plays a significant role in converging local optimum values rather than finding a global optimum value. The time to find an optimal solution increases in a non-linear manner when the size of the population increases. Our proposed PSO performs

Table 6
Simulation details of job, resource and makespan.

Schedule #	Total number of jobs submitted	Total number of resources	Makespan value in PSO (h)	Makespan value in ACO (h)	Makespan value in GA (h)	Makespan value in RBA (h)
Schedule 1	200	100	154.87	163.27	165.91	168.12
Schedule 2	400	200	157.51	169.51	173.87	175.78
Schedule 3	600	300	166.29	174.03	177.16	179.69
Schedule 4	800	400	170.81	181.74	184.78	188.79
Schedule 5	1000	500	179.47	186.90	189.45	193.58

Table 7
Simulation details of job, resource and cost.

Schedule #	Jobs	Resources	PSO max. cost (in $/h)	ACO max. cost (in $/h)	GA max. cost (in $/h)	RBA max. cost (in $/h)
Schedule 1	200	100	124.69	171.72	184.49	245.89
Schedule 2	400	200	133.84	182.99	197.10	384.81
Schedule 3	600	300	137.61	193.20	219.37	449.29
Schedule 4	800	400	151.53	245.84	269.98	539.9
Schedule 5	1000	500	162.44	257.394	287.41	612.94

Table 8
Simulation details of job, resource and jobs meeting deadline.

Schedule #	Jobs	Resources	# of jobs meeting deadline in PSO	# of jobs meeting deadline in ACO	# of jobs meeting deadline in GA	# of jobs meeting deadline in RBA
Schedule 1	200	100	156	128	121	92
Schedule 2	400	200	306	251	238	178
Schedule 3	600	300	448	368	347	258
Schedule 4	800	400	588	484	432	332
Schedule 5	1000	500	750	596	529	409

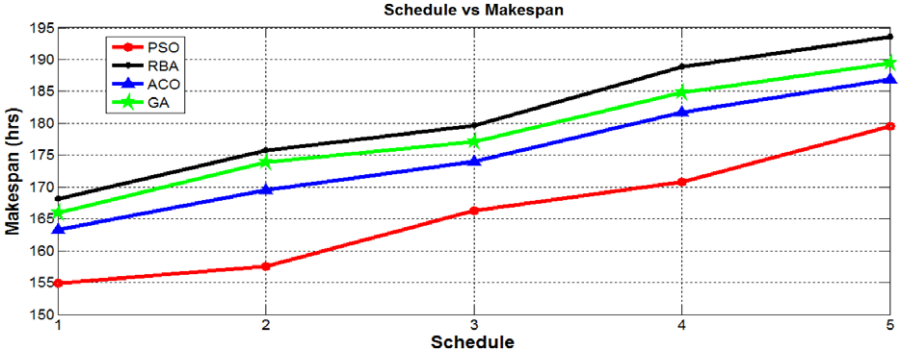


Fig. 5. Comparison of makespan for proposed PSO vs. ACO, GA and RBA.

both exploration and exploitation in the search space and selects a resource which is having the minimal execution time and cost. From the simulation results, it is evident that the proposed PSO minimizes the makespan, cost and rejection ratio and maximizes the number of jobs meeting deadline and user satisfaction. Furthermore, unlike GA and ACO, PSO achieves faster convergence speed. The detailed simulation results are discussed below:

6.3.1. Comparison of minimization of makespan

One of the primary goals of this paper is minimization of execution time of jobs submitted to CLOUDRB. The first set of experiments is carried out to compare the makespan value for different algorithms. We have investigated the effect of the proposed PSO based scheduling approach by varying the number of jobs and resources in each schedule. This simulation process is conducted by generating five schedules. Each schedule is varied with 200–100 jobs and resources in the range of 100–500 as presented in Table 6. The proposed resource allocation mechanism selects an optimal cloud resource for the job requests using the fitness function defined in Section 3. The makespan value envisioned during the simulation process is presented in Table 6. Fig. 5 shows a comparison between the makespan values of different algorithms for five schedules. From the simulation results, it is apparently evident that PSO achieves minimum makespan. Even though, when we increase the number of jobs up to a maximum value of 1000, makespan is minimal in proposed PSO based scheduling approach compared to the other three algorithms. Even though there is a slight variation in makespan, the number of jobs successfully executed in PSO is comparatively better than ACO, GA, and PSO. PSO executes nearly 75% of the jobs, ACO executes nearly 60% of jobs, GA executes nearly 53% of jobs and RBA executes 40% of jobs within the job deadline.

6.3.2. Comparison of maximum cost

This performance metric is computed for analyzing the maximum cost for the jobs per schedule. The maximum cost is calculated based on the number of jobs meeting deadlines per schedule that includes execution cost and transfer cost. The maximum cost value for five schedules is depicted in Table 7. The proposed PSO based scheduling approach selects the resource which is having the minimal execution time and cost. ACO selects the resource which is having the maximum pheromone value. If all the ants select a resource which is having the maximum pheromone value, the resource becomes overloaded which leads to an increase in the exe-

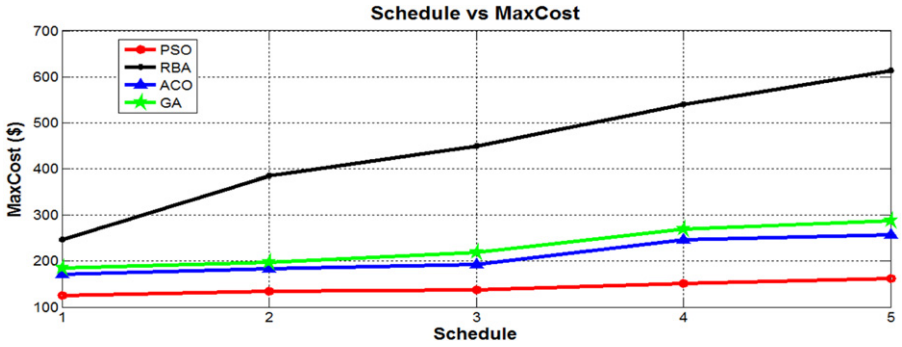


Fig. 6. Comparison of maximum cost for proposed PSO vs. ACO, GA and RBA.

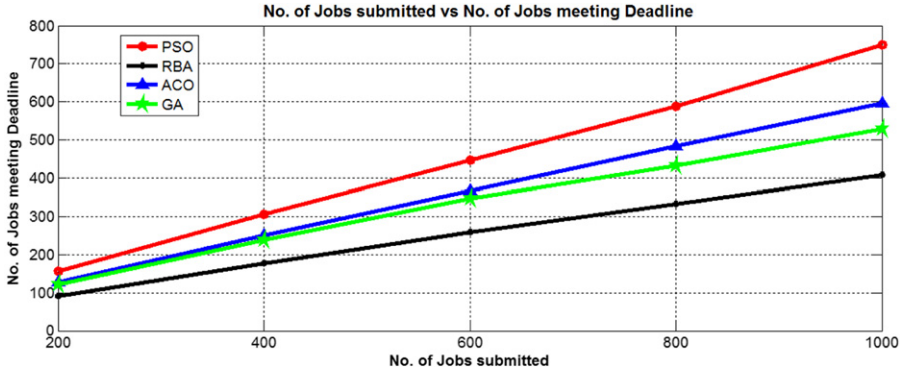


Fig. 7. Comparison for number of jobs meeting deadline for proposed PSO vs. ACO, GA and RBA.

Table 9
Comparison of number of jobs meeting deadline for proposed PSO vs. ACO, GA and RBA.

Schedule #	Jobs	Resources	Job rejection ratio in PSO	Job rejection ratio in ACO	Job rejection ratio in GA	Job rejection ratio in RBA
Schedule 1	200	100	0.235	0.34	0.41	0.445
Schedule 2	400	200	0.27	0.33	0.365	0.447
Schedule 3	600	300	0.253	0.298	0.316	0.47
Schedule 4	800	400	0.265	0.291	0.31875	0.476
Schedule 5	1000	500	0.25	0.279	0.281	0.491

cution time and cost of the jobs submitted per schedule. RBA computes the rank based on processor speed and RAM memory and allocates the jobs to the cloud resources based on the highest rank value; it does not have the objective to minimize the cost. The RBA increases the makespan value of the jobs submitted to the resource broker; obviously that increases the total cost of jobs submitted per schedule. From the simulation results, it is evident that PSO achieves minimal makespan and cost in all the five schedules. Fig. 6 represents the maximum cost acquired for the jobs in five schedules by PSO, ACO, GA and RBA. The maximum cost acquired for PSO varies from 124.69 to 162.44, ACO varies from 171.72 to 257.394, GA varies from 184.49 to 287.41 and RBA varies from 245.89 to 612.94. In other words, the cost of ACO increases by a factor of 1.5%, 1.7% in GA and it is up to 4% in RBA.

6.3.3. Comparison of number of jobs meeting deadline

This performance metric is computed for analyzing the number of jobs completed on or before the deadline specified by the user. The jobs and resources taken for simulation and the number of jobs meeting the user-specified deadline are shown in Table 8. Our proposed PSO based scheduling approach considers the deadline as one of the important factors in addition to the fitness function value. However, the ACO and GA tries to find the best resource based on the fitness value, whereas, RBA selects the resource which has the highest rank. Moreover, it does not consider the deadline as a constraint that leads to reducing the number of jobs meeting the user-specified deadline. Fig. 7 presented that the number of jobs completed before the deadline is comparatively more in PSO, ACO GA and RBA. The percentage of jobs meeting the deadline in an average of PSO is 74.93%, ACO is 60.9%, GA is 55.566% and the RBA is 42.3% only. In other words, PSO is comparatively 1.23 times better than ACO, 1.3 times better than GA and 1.77 times better than RBA. Hence, our proposed PSO based scheduling approach performs better than the other three algorithms in all the five schedules.

6.3.4. Comparison of job rejection ratio

This performance metric is computed based on the ratio of the number of jobs getting rejected versus the total number of jobs submitted to resource broker per schedule. The job rejection ratio is calculated using Eq. (23). The jobs and resources taken for simulation and the rejection ratio of simulated jobs are shown in Table 9. The job rejection ratio falls into two following cases:

1. The submitted job spends more time in queue (waiting time) exceeds or equal to the deadline of the jobs, i.e. $WT_{J_i} \geq Deadline_{J_i}$.
2. The waiting time of a job is less than the deadline but the finishing time of the job exceeds the deadline, i.e. $Deadline_{J_i} - WT_{J_i} > FinishTime_{J_i}$.

Let the total number of jobs submitted by the user k be N_k. The number of job requests that receive matched resources list per

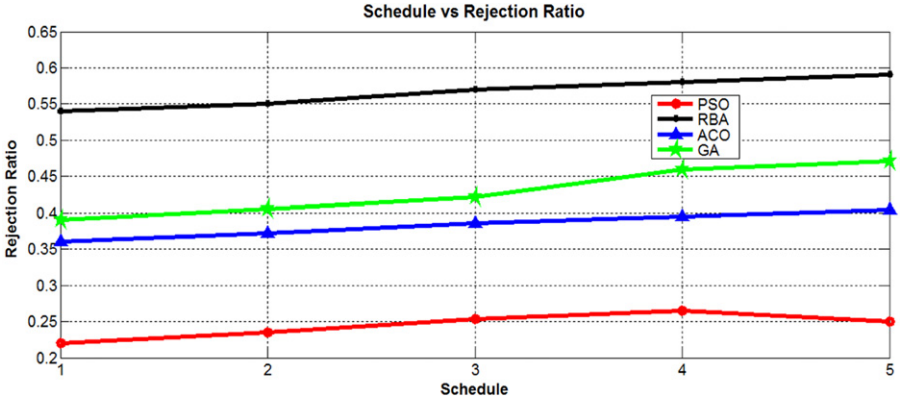


Fig. 8. Comparison of job rejection ratio for proposed PSO vs. ACO, GA and RBA.

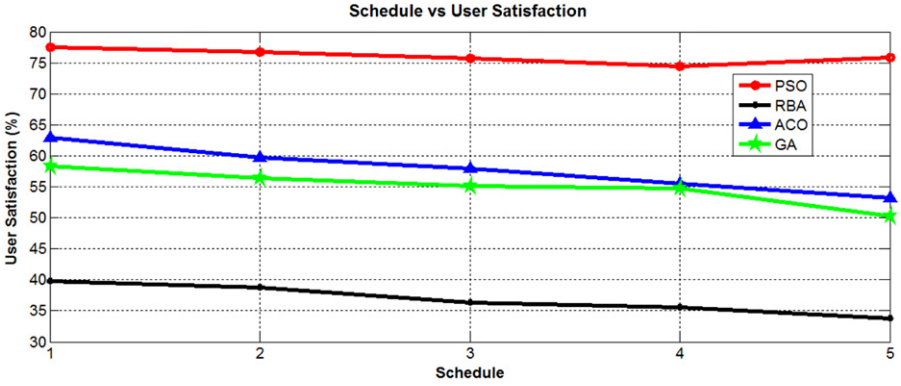


Fig. 9. Comparison of user satisfaction for proposed PSO vs. ACO, GA and RBA.

Table 10
Simulation details of job, resource and user satisfaction.

Schedule #	Jobs	Resources	User satisfaction in PSO (%)	User satisfaction in ACO (%)	User satisfaction in GA (%)	User satisfaction in RBA (%)
Schedule 1	200	100	77.5	62.95	58.32	39.845
Schedule 2	400	200	76.8	59.75	56.41	38.812
Schedule 3	600	300	75.66	57.92	55.19	36.324
Schedule 4	800	400	74.5	55.45	54.74	35.541
Schedule 5	1000	500	75.83	53.16	50.26	33.72

schedule S_p is JM_{S_p}. The number of job requests which have been successfully scheduled per schedule S_p is JS_{S_p}.

$$JRR_{S_p} = \frac{\left(1 - JM_{S_p}\right)}{N_k} + \frac{\left(1 - JS_{S_p}\right)}{JM_{S_p}}. \tag{23}$$

The job rejection ratio calculated using (23) for five schedules is presented in Fig. 8. The simulation results show that the rejection ratio in PSO is around 25%, in ACO it is nearly 40%, in GA it is approximately 50% and in RBA the rejection ratio is nearly 60%. From the results, it is evident that our proposed strategy outperforms GA, ACO and RBA algorithms in all the five schedules.

6.3.5. Comparison of user satisfaction

This performance metric represents the degree of satisfaction based on deadline and cost. For example, if the user submits the job at 10 A.M. and the deadline for the job is six hours, if the execution of the job is completed on or before 4 P.M. it increases the user satisfaction, else, it increases the user dissatisfaction. The simulation experiment is carried out in five schedules, and each schedule is varied with jobs ranging from 200 to 1000. The degree of satisfaction based on deadline (DoD_k) for the user k is calculated using (24) and the degree of satisfaction based on cost (DoC_k) for the user k is calculated using (25). The overall user satisfaction is calculated using Eq. (26).

$$DoD_k \leftarrow \sum_{i=1}^{n_k} w_i \left(Deadline_{J_i} - FinishTime_{J_i}\right). \tag{24}$$

Here, n_k represents the number of jobs submitted by user k, w_i represents the weight assigned to the job J_i.

$$DoC_k \leftarrow \sum_{i=1}^{n_k} w_i \left(Budget_{J_i} - ActualCostIncurred_{J_i}\right) \tag{25}$$

$$US_k \leftarrow DoD_k + DoC_k. \tag{26}$$

The calculated user satisfaction values are presented in Table 10. The comparison of user satisfaction values for PSO, ACO, GA and RBA is presented in Fig. 9. The simulation results are evidence that the user satisfaction is comparatively 12% better than ACO, 19% better than GA and 25% better than RBA.

6.3.6. Simulation 2: convergence Rate for 100, 500 and 1000 Cloud resources

This simulation process is carried out to find the convergence rate of proposed PSO for the resource selection process. Initially,

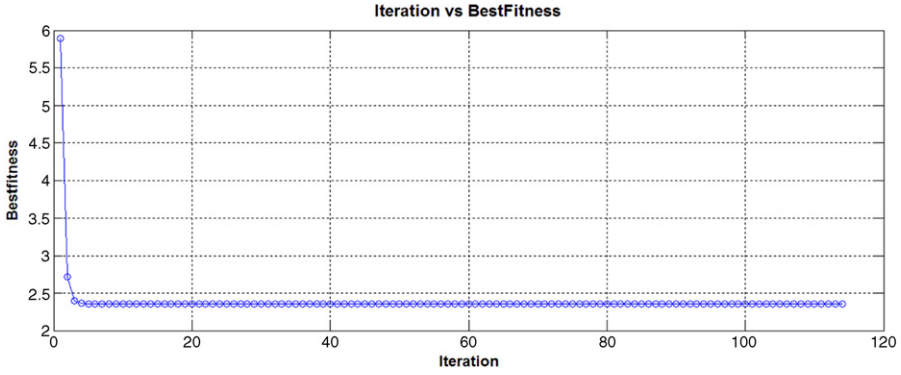


Fig. 10. Convergence rate for proposed PSO with 100 resources.

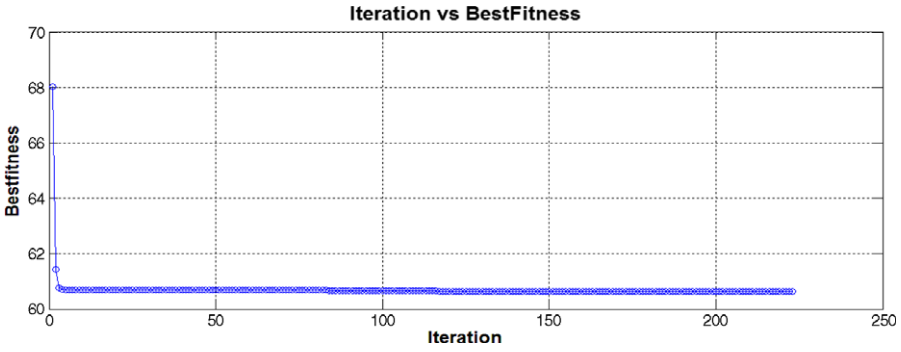


Fig. 11. Convergence rate for proposed PSO with 500 resources.

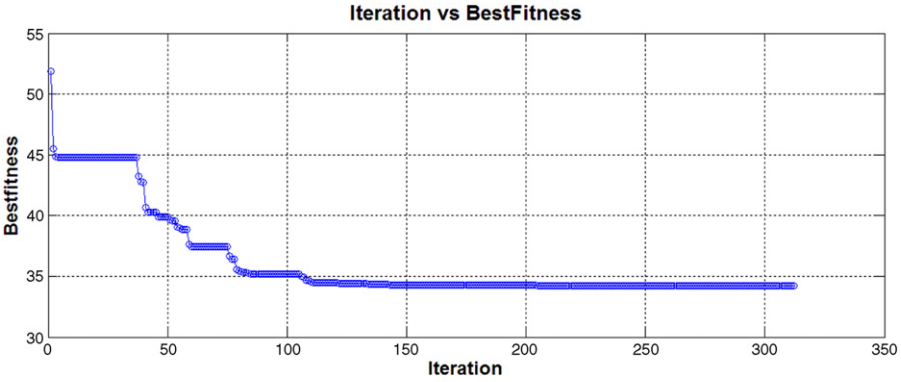


Fig. 12. Convergence rate for proposed PSO with 1000 resources.

PSO starts with computed values of *EET* and *EECost*. The algorithm does gradient descent to find an optimal solution. PSO tries to find an optimal value as well as it adjusts the velocity and position of every particle in all iterations. Finally, the algorithm converges to an optimal solution after '*N*' iterations. We simulated this experiment after fixing the maximum number of iterations as 1000 and stopping criteria as a fitness value remains constant for consecutive 100 iterations. The first simulation experiment is carried out for a job with 100 resources. The convergence rate value for 100 resources is shown in Fig. 10. From Fig. 10, it is evident that for 100 resources, the algorithm converges from iteration number 14 and remains constant for the next 100 iterations.

The second experiment is carried out for a job with 500 resources. The convergence rate is shown in Fig. 11. From Fig. 11, it is evident that for 500 resources, the algorithm converges from iteration number 123 and remains constant for the next 100 iterations.

The third experiment is carried out for a job with 1000 resources to find the convergence rate. From Fig. 12, it is evident that for 1000 resources, the algorithm converges from iteration number 212 and remains constant for the next 100 iterations.

7. Implementation details, experimental setup and application execution

The proposed work is carried out and implemented as part of research work in the Centre for Advanced Computing Research and Education (CARE), Anna University, Chennai, India.

7.1. Implementation details

The proposed architecture has been implemented using SOAP-based and RESTful (REpresentational State Transfer) [48] web services. The services are developed using NetBeans IDE 7.X and deployed in Sun Glassfish Server 3.0.1 and MySQL 5.0.77 is used as the database for storing the information about users and maintaining the job history for later use. The implementation of services is classified into two levels such as broker level and

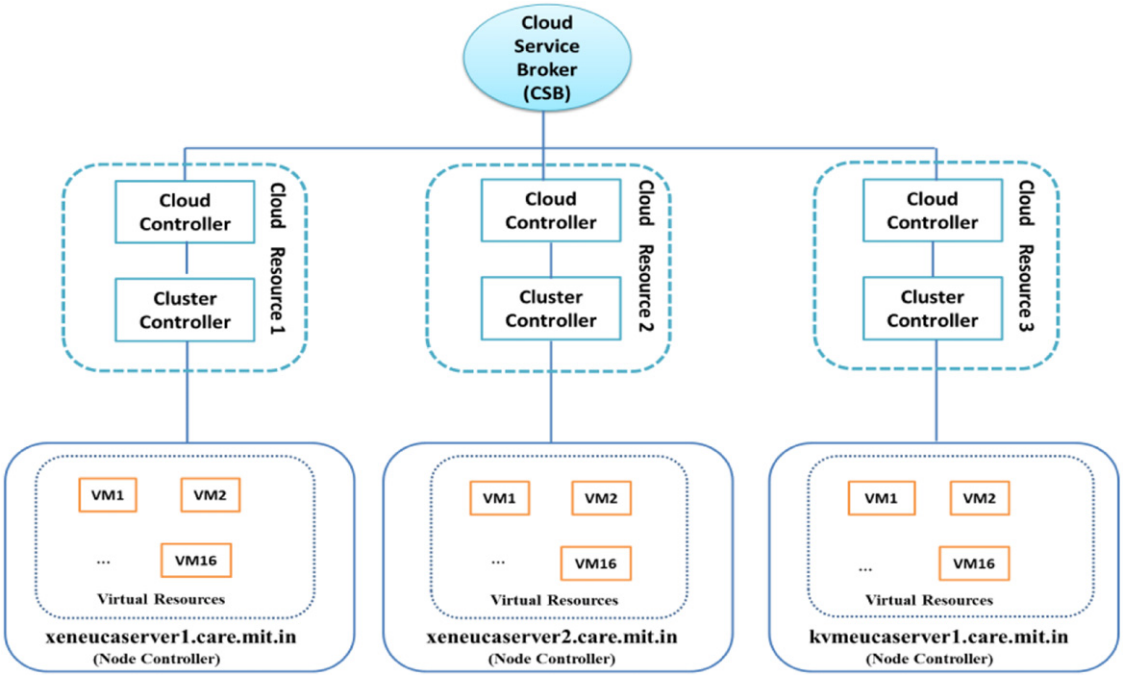


Fig. 13. Experimental setup.

Table 11
Service details and its deployment level.

Service name	Broker level	Infrastructure level
Request handler service	Yes	No
Job scheduling and resource allocator service	Yes	No
Dispatcher service	Yes	No
Cloud resource information aggregator service	Yes	No
Cloud resource provisioner service	Yes	No
Cloud monitoring and discovering service	No	Yes

infrastructure level and the implemented services are shown in Table 11. The Cloud Monitoring and Discovery Service (CMDS) are deployed in the middleware level to collect the static and dynamic attributes of the Cloud resources. The Cloud Resource Information Aggregator Service (CRIAS) has been implemented in push mode and deployed in broker level. It collects the resource information of static as well as dynamic attributes of the Cloud resources registered with CLOUDRB. The aggregated information is updated in the Cloud Resource Information Repository (CRIP).

7.2. Experimental setup

To evaluate the performance of our proposed work using real-world application execution, the Eucalyptus Cloud middleware based experimental setup is made in our department as shown in Fig. 13. The experimental setup is created with three Cloud clusters, namely xeneucaserver1.care.mit.in, xeneucaserver2.care.mit.in and kvmeucaserver1.care.mit.in. Each Cloud cluster has '*N*' number of Node Controllers. The Node controllers are managed by a Cluster Controller. The Cluster controllers are managed by a Cloud Controller. The Xen based Cloud cluster is installed with Xen-3.2.3 as a hypervisor and a KVM based Cloud cluster is installed with kvm-0.6.15 as the hypervisor.

7.3. Case study

In this section, we have discussed the testing of real-world HPC applications using our proposed approach in our cloud testbed. The first case study is Surface Detection of Defects in Fruits (Image Processing) application, which is embarrassingly parallel (parameter sweep) that helps to identify the defects in the fruits. The second case study is analyzing the structure of protein applications named NAMD [49], which is iterative parallel in nature. Both the case studies have been tested on a Eucalyptus based private Cloud infrastructure and analyzed for the various performance metrics of makespan, execution time and cost.

7.3.1. Embarrassingly parallel (parameter sweep)—image processing application

Central Electronics Engineering Research Institute (CEERI), India has developed an image processing application namely Surface Detection of Defects on fruits. Detection of defects in the fruits is a complex task due to stem areas and diverse types of defects present, as well as the natural variability of skin color. Defect detection, in particular, requires the precise segmentation of the defects. Also, the identification of surface defects in fruits helps in automatic rejection of fruits. This application identifies various defects in fruits including black spot surface defects in green and red mangoes and peel, mechanical and line defects in apples. Among these defects, black spots on mangos were detected using marker controlled watershed segmentation. It is implemented and tested for mangoes covering different species viz., Senthura, Thothapuri, Neelam and Rumani. The defects are quantified in terms of the number of black spots and percentage of the total black spot area in relation to the mango's surface area for grading the mangos. Peel defects in apples were detected using the RGB thresholding method; whereas the mechanical defects are detected by the Bit-Plane slicing method and Line Defects are detected using sub-image thresholding. In order to improve the execution time, we have parallelized this application as a parameter sweep application that takes the pictures of fruits in four distinct directions, which are coming through the conveyor belt. These images are compared with test images, which are stored in the database to identify whether there is any black spots surface defect in green and red mangoes. The real testing of this application requires a huge amount of computational resources in an on demand manner. Hence, we have made the experimental

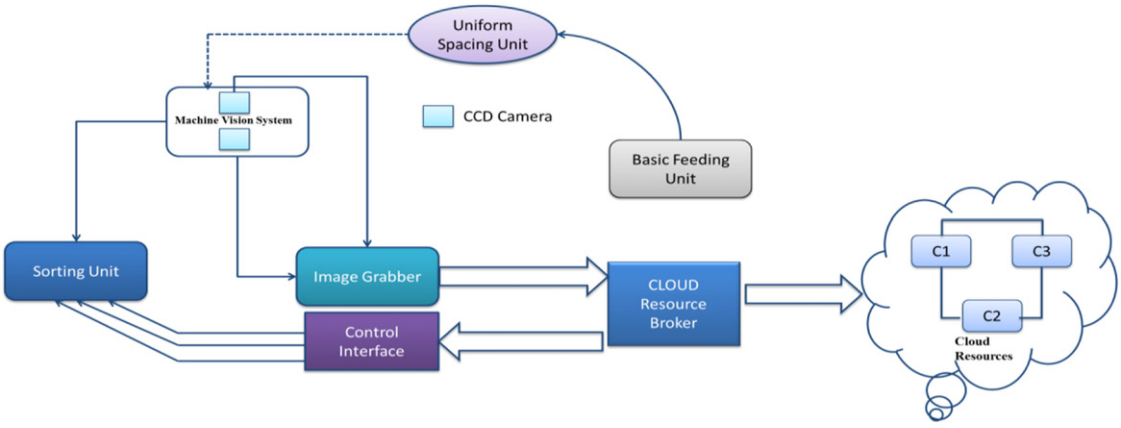


Fig. 14. Schematic representation of surface detection of defects.

setup as shown in Fig. 14 for capturing the images of fruit and processing those images using Eucalyptus based Cloud resources.

The experimental setup consists of a feeding unit, fruit uniform spacing unit, machine vision system and sorting unit. The feeding unit transports the fruits to the uniform space conveyor. Then the fruits are fed into the machine vision system for inspection of defects. The sorting unit is responsible for performing the grading operation to accept/reject the fruits. The machine vision system is included with the cup type conveyor, lighting chamber for the desired spectrum and light distribution for fruit illumination, two cameras, and an image grabbing card with four input channels inserted in a microcomputer. The image sensors in the cameras had an actual resolution of 580 horizontal and 350 vertical TV lines. Each camera is synchronized to another timing source with a variable electronic shutter. In addition to that, Identical 8.5 mm focal length C-mount lenses are attached to the cameras, with interference band-pass optical filters (840 nm) which are attached to the outside of each lens. The images are retrieved by Image Grabber and it is fed into the CLOUDRB. The matchmaker in CLOUDRB identifies the potential resources for processing of application requests. Once the resources have been identified, the dispatcher invokes the Deadline-based Job Scheduler to prioritize the job requests. The dispatcher invokes the PSO-based Resource Scheduler to select the near optimal cloud resource. It invokes the Cloud Resource Provisioner to instantiate the virtual cluster creation with the required capability as per the user requirements. Finally, the input files (images) are transferred to the provisioned virtual resource for application execution. At present, we have captured 40–60 fruits per instant and successfully processed 160–240 images using our cloud testbed. The time taken by parallel execution is 60% less than by sequential execution.

There are various edge detection algorithms available such as Canny, Prewitt, Roberts, Zero-cross, Sobel, etc. Among these, Sobel can detect all the black spots. However, it shows that some of the black spots have no closed boundary due to the nature of gray levels of black spots. Hence, watershed segmentation is used along with Sobel edge detection. The algorithm first captures the fruit images; remove the background and noise using the minimal filter as shown in Fig. 15(a)–(b). Subsequently, the images are converted into Gradient image and the magnitude of gradient image is calculated using a Sobel filter. Next, the regional minima are calculated which will act as Internal Markers corresponding to defects, then distance transform is computed that will act as External markers correspond to background. Watershed ridge lines have been found using Internal and External markers and those wrinkles have been imposed on Gradient images as shown in Fig. 15(c)–(e). Finally, pixels are counted in each region to identify the defected and non-defected portion in the fruits. The identified

Table 12
Variable matrix.

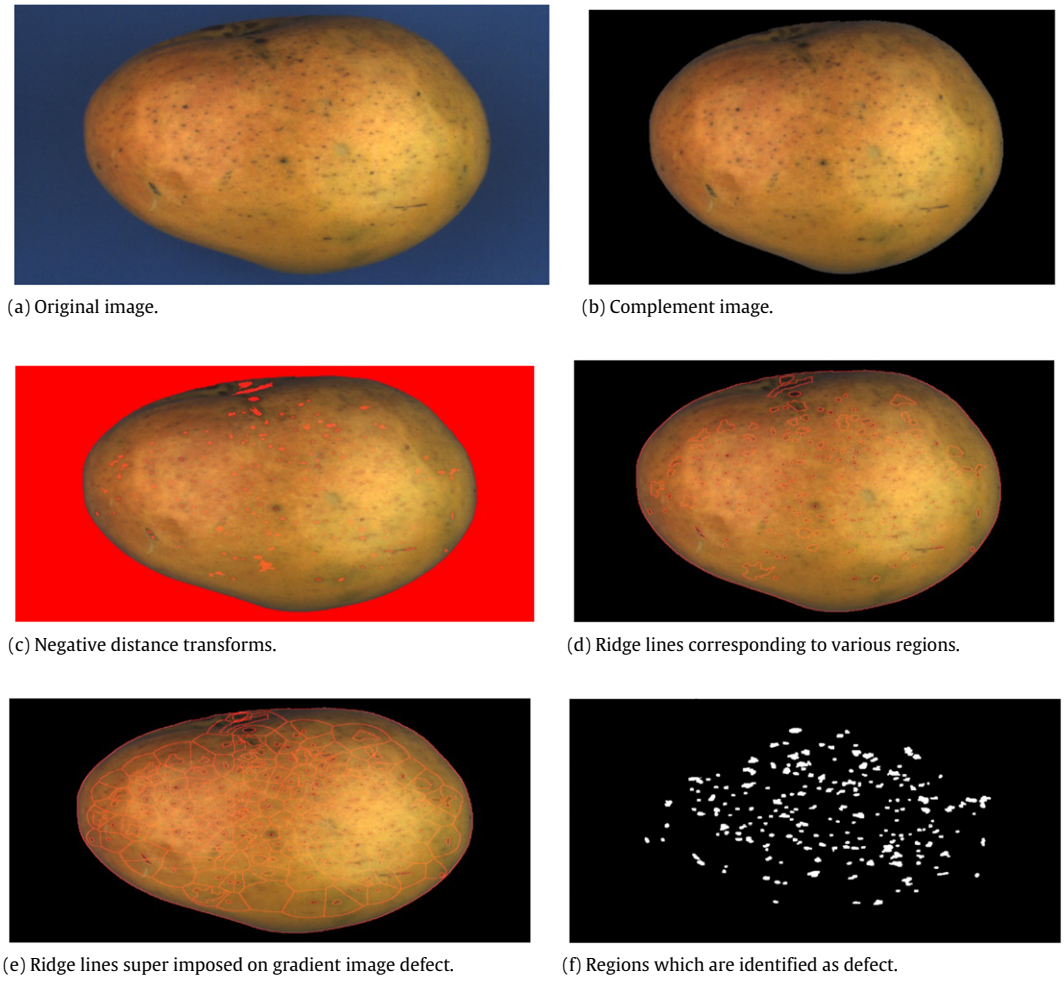
A1S1	A2S1	A1S2	A2S2
A3S1	A4S1	A3S2	A4S2
A1S3	A2S3	A1S4	A2S4
A3S3	A4S3	A3S4	A4S4

Table 13
Time taken by sequential and parallel cloud cluster.

Iteration	Execution time (in minutes)		% of time saved
	Sequential	Parallel	
1	27.83056	11.1853	59.80931
2	30.93951	11.95596	61.35698
3	30.3973	14.27329	53.04421
4	27.43112	11.37914	58.51741
5	27.59889	11.83616	57.11363
6	28.4858	12.18261	57.23271
7	28.05725	12.79112	54.41064
8	29.83515	11.35939	61.92614
9	29.26784	13.98613	52.21332
10	33.20827	12.91121	61.12049

defects in fruit images are depicted in Fig. 15(f). The resulting regions related to black spots in mango have been counted by human vision and proposed techniques. After analyzing the results carefully, we have identified that the global thresholding method failed to detect many black spots whereas the marker controlled watershed segmentation method identified the black spots count closer to the human eye as shown in Fig. 16. Due to the presence of noise in original image such as very high contrast and rough surface, the proposed technique shows some extra black spots but it is negligible.

This application requires four different views of a mango image. Each of mango image is sent to four different processors. Finally, the output is aggregated from those processors. The area of each defected region is calculated and returned from all four processors in four different variables, each belonging to separate fruit. The result returned by a processor is represented as a matrix and it is shown in Table 12. The total pixels defected for each fruit is calculated using (27). Here, A_iS_j represent the number of pixels defected in jth view of ith mango. The application is tested in sequential as well as in parallel mode for ten iterations and execution time is noted in Table 13. The percentage of time saved using our proposed approach is shown in Table 13. The graph shown in Fig. 17 shows the impact of parallelization of the marker controlled watershed segmentation algorithm and running the application in cloud resources in a parallel manner. It reduces



(a) Original image.

(b) Complement image.

(c) Negative distance transforms.

(d) Ridge lines corresponding to various regions.

(e) Ridge lines super imposed on gradient image defect.

(f) Regions which are identified as defect.

Fig. 15. Surface detection of defects.

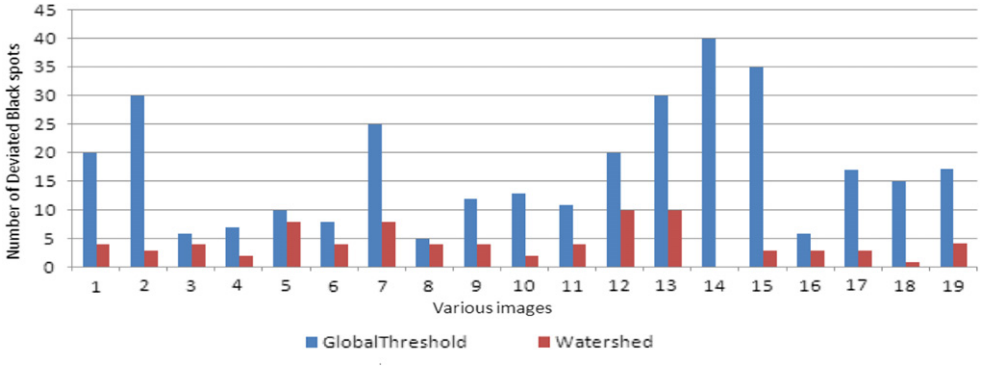


Fig. 16. Deviation from human vision.

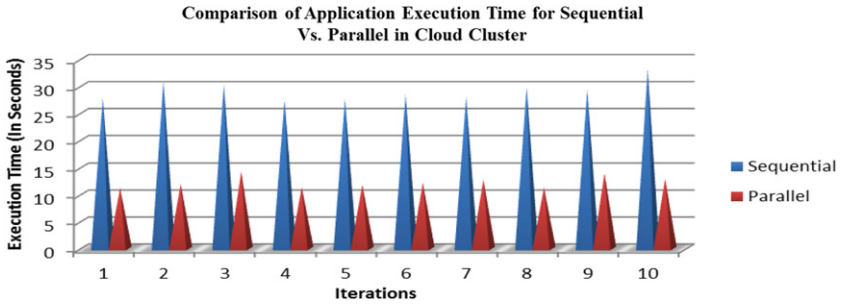


Fig. 17. Comparison of application execution time in sequential versus parallel mode.

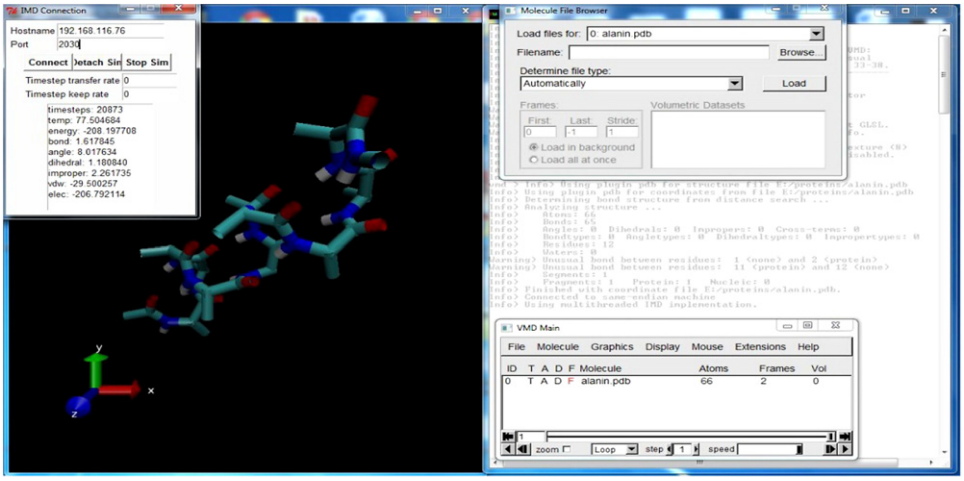


Fig. 18. NAMD simulation.

Table 14
Input files for NAMD simulation.

Protein database & input file names	Number of steps
alanin.pdb alanin.psf alanin.config alanin.params	100–10 000 000 steps

Table 15
Application execution time in cloud resources.

Number of nodes	Iteration 1 (execution time in seconds)	Iteration 2 (execution time in seconds)	Average time taken (execution time in seconds)
10	57.500	57.33583	57.41783
20	52.188	52.833	52.5105
30	47.877	48.0555	47.966
40	44.733	44.888	44.8105

the execution time up to 60% when compared to sequential performance.

$$\text{Total Pixels Defected in } i\text{th fruit} = \sum_{j=1}^{4} A_i S_j. \tag{27}$$

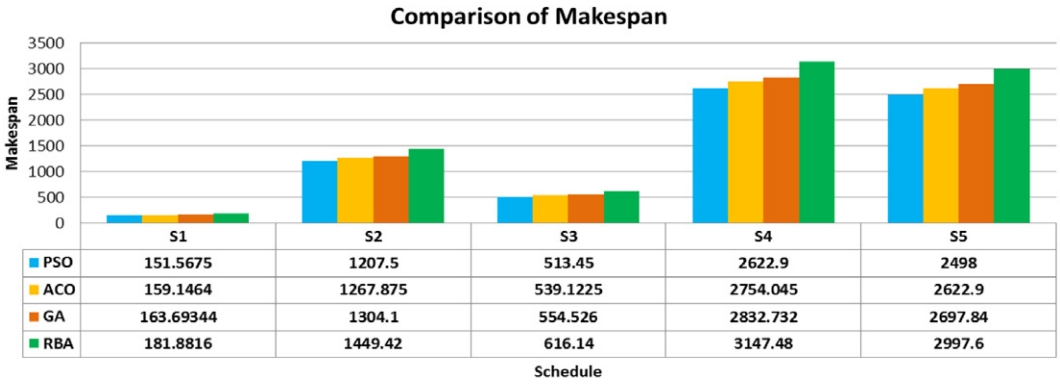
7.3.2. Iterative parallel—bio-informatics application

NAMD is scalable molecular dynamics software, and it is primarily designed for high-performance simulation of large Bio-molecular systems. Molecular Dynamics (MD) simulation is a popular tool in modern molecular modeling technology. It helps to understand the structure, dynamics and motion of individual atoms. The two most commonly used methods are energy minimization and molecular dynamics that optimize the structure and simulate the natural motion of biological macromolecules. It converts molecular coordinates from a Protein Data Bank (PDB) file into its internal format. The NAMD simulation process first generates the configuration file. The actual simulation process starts once the configuration file is created, that produces a trajectory file describing the movements of atoms over time. The generated trajectory file is helpful to understand the structure of protein. The protein file is downloaded from this site [50] [http://www.pdb.org/pdb/]. The input files required to run the simulation process are Protein Structure File (PSF), force field parameter file and configuration file as shown in Table 14. The PSF file is used to store the structural information of protein and various types of bonding interactions. The force field parameter file consists of a mathematical expression of the potential in which atoms in the system will experience. The number of steps is an important parameter to analyze the structure of protein and derives the accurate results about the protein structures. Furthermore, if we increase the number of steps that plays a direct role in increasing the execution time of applications as well as the accuracy of the analysis process. We have pre-configured the operating system images and those images have uploaded in cloud resources. The experiment is carried out by generating the job requests by varying the number of nodes, iterations and the number of steps in a random manner as depicted in Table 14.

The generated job requests are submitted to CLOUDRB with hardware, software and Quality of Service (QoS) parameters. It first prioritizes the job requests, and then it selects the cloud resources that satisfy the user application requirements with the objectives of minimizing execution time and cost. Once the cloud resources are selected for virtual resource creation, the virtual resource provisioner invokes the cloud middleware for virtual instance creation. Our eucalyptus cloud testbed is configured in system mode; hence, the virtual instance acquires the IP address in a dynamic manner through a DHCP server. The acquired IP address has been sent back to CLOUDRB. Once the IP address is received, it transfers the executable and input files to the virtual instance through the Secured SHell (SSH) passwordless communication mode. The experiment is carried out in two iterations to find out the consistency of the application execution time. The median time taken to run the application in a cloud environment is depicted in Table 15. Once the simulation is completed successfully, the generated trajectory files are viewed and analyzed using Visual Molecular Dynamics (VMD) [51] as shown in Fig. 18. VMD is the visualizer tool that is helpful to watch the trajectory files and analyze the NAMD simulation results. Moreover, the cloud user can execute the simulation process from her machine by establishing IMD connection using the virtual instance IP address and default port of 2030. This approach is helpful for the researchers in biotechnology doing research in the field of bio-molecular systems to examine the motion of molecules in water, iron, etc., for pharmaceutical and other applications.

7.3.3. Evaluation of proposed work

We have generated five schedules and each schedule has two different types of job requests files, namely (1) DefectDetection-Simulation.xml and (2) NAMDSimulation.xml. The job requests files consist of hardware, software and Quality of Service (QoS) parameters. The generated job requests are submitted to the resource broker. It first prioritizes the job requests, and then



	S1	S2	S3	S4	S5
PSO	151.5675	1207.5	513.45	2622.9	2498
ACO	159.1464	1267.875	539.1225	2754.045	2622.9
GA	163.69344	1304.1	554.526	2832.732	2697.84
RBA	181.8816	1449.42	616.14	3147.48	2997.6

Fig. 19. Comparison of makespan for PSO, ACO, GA and RBA.

it selects the cloud resources that satisfy the user application requirements. It schedules the jobs, if it is possible to complete the jobs within the deadline specified by the user. The proposed research work is mainly aimed to complete the jobs within minimal time (makespan) and cost. We have tested and compared our proposed approach with the other three types of resource scheduling algorithms such as Ant Colony Optimization (ACO), Genetic Algorithm (GA) and Rank-Based Allocation (RBA) in CLOUDRB. In the first strategy, the resource selection is carried out based on the Particle Swarm Optimization (PSO) algorithm. The resource selection is based on Ant Colony Optimization (ACO) in the second strategy. The resource selection is carried out based on the Genetic Algorithm (GA) in the third strategy. In the fourth strategy, the resource selection is carried out by calculating the rank of resources using processor speed and RAM speed. Finally, the scheduler schedules the job to the selected resource for virtual resource creation and job execution. The results of three different resource selection strategies are taken in a day at different times. The comparison has been carried out by generating the job requests with short execution time. The total execution time is calculated based on execution time, transfer time and booting time. The proposed PSO based scheduling strategy achieves minimal makespan value; this can be seen in Fig. 19 that represents the total time taken for the execution of jobs generated in a particular schedule. The reason behind PSO achieving minimal makespan is that it chooses a resource which has minimal execution time and cost, in ACO an optimal schedule is selected from an initial random allocation process, whereas in the GA an optimal schedule is selected based on the ranking of chromosomes and applying the selection, crossover and mutation operations. As it is known, RBA selects a resource which has the highest rank.

8. Conclusion and future work

In this paper, we first formulate the NP-complete problem of scheduling in the Cloud environment. Then we have discussed the proposed Particle Swarm Optimization (PSO) mechanism for achieving efficient resource allocation and job scheduling. In the next section, we have discussed the Cloud based architectural design and components of CLOUDRB along with working principle. The integration of deadline-based job scheduling with PSO-based resource scheduling mechanism prioritizes the user application requests based on deadline and selects the cloud resources that complete the jobs with minimal time and cost by considering the deadline as one of the important factor that leads to increase the user satisfaction level. The proposed research work is evaluated with the help of Matlab simulation, and it is compared with Ant Colony Optimization (ACO), Genetic Algorithm (GA) and Rank-Based Allocation (RBA) mechanism. The same has been tested in a Eucalyptus based Cloud testbed by submitting bio-informatics and image processing applications, which are computationally intensive in nature. The performance metrics calculated from the simulation experiments are minimizations of the makespan, cost and job rejection ratio and maximizations of the number of jobs meeting a deadline and user satisfaction level. It has been observed from the applications tested in our Cloud testbed that the proposed CLOUDRB is able to successfully complete the execution of jobs within the deadline with a reduction in time and cost. The future work will explore further the integration of semantic resource discovery and Service Level Agreement (SLA) to improve the job mapping process and Quality of Service (QoS).

Acknowledgments

The authors sincerely thank the Ministry of communication and Information Technology, Government of India, Delhi for financially supporting the Centre for Advanced Computing Research and Education (CARE) in Anna University Chennai, India. The authors also thank Ms. Rajalakshmi Shenbaga Moorthy, a M.E student of MIT Campus, Anna University for spending her valuable time to implement the prototype of the PSO algorithm in the Matlab environment.

References

[1] Nimbus: open source infrastructure-as-a-service cloud computing software, Kate Keahey, in: Workshop on Adapting Applications and Computing Services to Multi-Core and Virtualization, CERN, Switzerland, June 2009.
[2] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, The Eucalyptus open source cloud-computing system, in: Proceedings of Cloud Computing and its Applications, October 2008.
[3] OpenNebula: The Open Source Toolkit for Cloud Computing, 2011. http://opennebula.org/start.
[4] Hadoop: http://hadoop.apache.org/.
[5] Jeffrey D. Ullman, NP-complete scheduling problems, J. Comput. System Sci. 10 (3) (1975) 384–393.
[6] S. Binitha, S.Siva Sathya, A survey of bio inspired optimization algorithms, Int. J. Soft Comput. Eng. (IJSCE) (ISSN: 2231-2307) 2 (2) (2012).
[7] Genetic Algorithm, J.H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM J. Comput. 2 (2) (1973) 88–105.
[8] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.
[9] Particle Swarm Optimization (PSO), http://www.swarmintelligence.org/.
[10] J. Kennedy, R. Eberhart, Particle swarm optimization (PSO), in: Proc. IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
[11] Lizheng Guo, Shuguang Zhao, Shigen Shen, Changyuan Jiang, Task scheduling optimization in cloud computing based on heuristic algorithm, J. Netw. 7 (3) (2012) 547–553. http://dx.doi.org/10.4304/jnw.7.3.547–553.
[12] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, Rajkumar Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments, in: AINA'10 Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE Computer Society, Washington, DC, USA, ISBN: 978-0-7695-4018-4, ©2010, pp. 400–407. http://dx.doi.org/10.1109/AINA.2010.31.

[13] M. Mezmaz, Choon Lee Young, N. Melab, E.-G. Talbi, A.Y. Zomaya, A bi-objective hybrid genetic algorithm to minimize energy consumption and makespan for precedence-constrained applications using dynamic voltage scaling, in: 2010 IEEE Congress on Evolutionary Computation, CEC, 18–23 July 2010.
[14] Hongbo Liu, Ajith Abraham, Aboul Ella Hassanien, Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, Future Gener. Comput. Syst. (ISSN: 0167-739X) 26 (8) (2010) 1336–1343. http://dx.doi.org/10.1016/j.future.2009.05.022.
[15] Jing Liu, Xing Guo Luo, Xing Ming Zhang, Fan Zhang, Job scheduling algorithm for cloud computing based on particle swarm optimization, in: Advanced Materials Research, Vol. 662, 2013, p. 957.
[16] Shaobin Zhan, Hongying Huo, Improved PSO-based task scheduling algorithm in cloud computing, J. Inform. Comput. Sci. 9 (13) (2012) 3821–3829.
[17] Hongli Zhang, Panpan Li, Zhigang Zhou, Xiang Zhan Yu, A PSO-based hierarchical resource scheduling on cloud computing, in: ISCTCS 2012, CCIS 320, 2013, pp. 325–332. http://dx.doi.org/10.1007/978-3-642-35795-4_41.
[18] Rajarathinam Jeyarani, N. Nagaveni, Rajarathinam Vasanth Ram, Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence, Future Gener. Comput. Syst. 28 (5) (2012) 811–821.
[19] D.A. Abramson, R. Buyya, J. Giddy, A computational economy for grid computing and its implementation in the Nimrod-G resource broker, Future Gener. Comput. Syst. 18 (8) (2002) 1061–1074. Elsevier Science BV, Amsterdam, The Netherlands.
[20] Biao Song, Mohammad Mehedi Hassan, Eui-nam Huh, A novel heuristic-based task selection and allocation framework in dynamic collaborative cloud service platform, in: CloudCom 2010, pp. 360–367.
[21] Hai Zhong, Kun Tao, Xuejie Zhang, An approach to optimized resource scheduling algorithm for open-source cloud systems, in: The Fifth Annual China Grid Conference, 2010. http://dx.doi.org/10.1109/ChinaGrid.2010.37.
[22] Eun-Kyu Byuna, Yang-Suk Keeb, Jin-Soo Kimc, Seungryoul Maeng, Cost optimized provisioning of elastic resources for application workflows, Future Gener. Comput. Syst. 27 (2011) 1011–1026.
[23] O.O. Sonmez, A. Gursoy, A novel economic-based scheduling heuristic for computational grids, Int. J. High Perform. Comput. Appl. 21 (1) (2007) 21–29.
[24] Jin Xu, Albert Y.S. Lam, Victor O.K. Li, Chemical reaction optimization for task scheduling in grid computing, IEEE Trans. Parallel Distrib. Syst. 22 (10) (2011) 1624–1631.
[25] Liang Hu, Xi-Long Che, Si-Qing Zheng, Online system for grid resource monitoring and machine learning-based prediction, IEEE Trans. Parallel Distrib. Syst. 23 (1) (2012) 134–145.
[26] Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, Chai Chompoo-inwai, An ant colony optimization for dynamic job scheduling in grid environment, Int. J. Comput. Inf. Eng. 1 (2007) 8.
[27] Jia Yu, Rajkumar Buyya, Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms, Sci. Program. J. (ISSN: 1058-9244) 14 (3–4) (2006) 217–230. IOS Press, Amsterdam, The Netherlands.
[28] Fengguang Tian, Keke Chen, Towards optimal resource provisioning for running mapreduce programs in public clouds, in: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, 2011, pp. 155–162, July 04–09.
[29] S. Chaisiri, Bu-Sung Lee, D. Niyato, Optimization of resource provisioning cost in cloud computing, IEEE Trans. Serv. Comput. 5 (2) (2012) 164–177. http://dx.doi.org/10.1109/TSC.2011.7.
[30] M.F. Tasgetiren, Y.-C. Liang, M. Sevkli, G. Gencyilmaz, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European J. Oper. Res. 177 (3) (2007) 1930–1947.
[31] Mohamad Izuddin Nordin, Azween Abdullah, Mahamat Issa Hassan, Goal-based request cloud resource broker in medical application, World Acad. Sci. Eng. Technol. 50 (2011).
[32] Thamarai Selvi Somasundaram, Balachandar R. Amarnath, R. Kumar, P. Balakrishnan, K. Rajendar, R. Rajiv, G. Kannan, G. Rajesh Britto, E. Mahendran, B. Madusudhanan, CARE resource broker: a framework for scheduling and supporting virtual resource management, Future Gener. Comput. Syst. 26 (3) (2010) 337–347. http://dx.doi.org/10.1016/j.physletb.2003.10.071.
[33] Globus Toolkit, http://www.globus.org/toolkit/.
[34] P. Marshall, K. Keahey, T. Freeman, Elastic site: using clouds to elastically extend site resources, in: IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2010, May 2010.
[35] R. Duan, T. Fahringer, R. Prodan, J. Qin, A. Villazon, M. Wieczorek, Real world workflow applications in the askalon grid environment, in: European Grid Conference 2005, EGC 2005, in: LNCS, Springer-Verlag, 2005, Copyright (C).
[36] Wei Huang, Jiuxing Liu, Bulent Abali, Dhabaleswar K. Panda, A case for high performance computing with virtual machines, in: Proceedings of the 20th Annual International Conference on Supercomputing ICS'06, ACM, New York, NY, USA, ISBN: 1-59593-282-8, ©2006, pp. 125–134. http://dx.doi.org/10.1145/1183401.118342.
[37] Christian Vecchiola, Rodrigo N. Calheiros, Dileban Karunamoorthy, Rajkumar Buyya, Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka, Future Gener. Comput. Syst. 28 (1) (2012) 58–65. http://dx.doi.org/10.1016/j.future.2011.05.008.
[38] Andrea Pugliese, Domenico Talia, Ramin Yahyapour, Modeling and supporting grid scheduling, J. Grid Comput. 6 (2008) 195–213.
[39] Eucalyptus API, http://www.apihub.com/api/eucalyptus-api.
[40] Amazon EC2, http://aws.amazon.com/ec2/.
[41] Typica API, http://code.google.com/p/typica/.
[42] T.S. Somasundaram, K. Govindarajan, Cloud monitoring and discovery service (CMDS) for IaaS resources, in: 2011 Third International Conference on Advanced Computing, ICoAC, 14–16 December 2011, pp. 340–345. http://dx.doi.org/10.1109/ICoAC.2011.6165199.
[43] Grid Uniform Laboratory Extension (GLUE), http://www.ogf.org/documents/GFD.147.pdf.
[44] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[45] H. Liu, A. Abraham, Fuzzy adaptive turbulent particle swarm optimization, in: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, 2005, pp. 445–450.
[46] M. Dorigo, T. Stutzle, Ant Colony Optimization, MIT Press, Cambridge, 2004.
[47] D.E. Goldberg, Genetic Algorithm: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Corporation, Inc., Reading, 1989.
[48] REST (REpresentational State Transfer), https://www.ibm.com/developerworks/webservices/library/ws-restful/.
[49] NAMD, http://www.ks.uiuc.edu/Research/namd/.
[50] Protein DataBase (PDB), http://www.pdb.org/pdb/.
[51] Visual Molecular Dynamics (VMD), http://www.ks.uiuc.edu/Research/vmd/.

Thamarai Selvi Somasundaram is working as a Professor & Dean in the MIT Campus, Anna University, Chennai, India. She received her Ph.D. in Computer Science and Engineering from Manonmaniam Sundaranar University, Tirunelveli. She has 28 years of teaching and research experience. Her area of interest includes artificial intelligence, neural networks, grid computing, grid scheduling, semantic technology, virtualization and Cloud computing. She has authored/co-authored five books. She has received the best IBM faculty award in the year of 2009. She has produced ten Ph.D.scholars under her guidance. She received the Active Researcher Award in 2011 from Anna University, Chennai. She has received patents for her inventions received and also she is one of the inventors for two patents which have been filed and are under consideration. She has authored/co-authored more than 100 research publications to her credit. She established the Grid Computing Research Laboratory in the Department of Information Technology at MIT Campus, Anna University. She is the Principal Investigator for the project titled "Centre for Advanced Computing Research and Education (CARE)" funded by Department of Information Technology, Ministry of Communication and Information Technology, New Delhi. She has received Staff Exchange for Education and Research Initiative, under the UKERI scheme for visiting the University of Dundee, Scotland. She has visited various countries such as Singapore, Malaysia, Australia, New Zealand, Kuwait, Oman, the UAE, Germany, France, the UK, Switzerland, the USA and Canada. She is the reviewer for Future Generation Computing Systems (FGCS), Journal of Grid Computing, Journal of Parallel and Distributed Computing, Transactions on Mobile Computing and IEEE Transactions on Data and Knowledge Engineering.

Kannan Govindarajan has received the B.E. degree in Information Technology from Bharathidasan University. He has completed his M.S. (by research) degree in Computer Science and Engineering from Anna University, Chennai in the area of Grid Scheduling in a Virtualized Grid Environment. He is currently doing his Ph.D. in Computer Science and Engineering at Anna University. He has seven years of research and development experience. His area of interest includes grid computing, grid scheduling, semantic technology, virtualization and Cloud computing. He is currently working as a Senior Research Associate in the Centre for Advanced Computing Research and Education (CARE), MIT Campus, Anna University. He has received an APAN fellowship from ERNET, India in 2011. Recently, in 2012 he has been invited to visit Athabasca University, Canada for research interaction and development under the Pro Development Initiative Grant (PDIG) funded by the Indo-Canadian Shastri Institute. He has received patents for his inventions and is also one of the inventors for two patents which have been filed and are under consideration. He has published two journal papers and in more than twenty National/International conferences and two reputed journals.