J Netw Syst Manage (2018) 26:361–400
https://doi.org/10.1007/s10922-017-9419-y

BULLET: Particle Swarm Optimization Based Scheduling Technique for Provisioned Cloud Resources

Sukhpal Singh Gill[1] · **Rajkumar Buyya**[1] · **Inderveer Chana**[2] · **Maninder Singh**[2] · **Ajith Abraham**[3]

Received: 6 February 2016 / Revised: 15 July 2017 / Accepted: 26 July 2017 / Published online: 2 August 2017

Abstract Cloud resource scheduling requires mapping of cloud resources to cloud workloads. Scheduling results can be optimized by considering Quality of Service (QoS) parameters as inherent requirements of scheduling. In existing literature, only a few resource scheduling algorithms have considered cost and execution time constraints but efficient scheduling requires better optimization of QoS parameters. The main aim of this research paper is to present an efficient strategy for execution of workloads on cloud resources. A particle swarm optimization based resource scheduling technique has been designed named as BULLET which is used to execute workloads effectively on available resources. Performance of the proposed technique has been evaluated in cloud environment. The experimental results show that the proposed technique efficiently reduces execution cost, time and energy consumption along with other QoS parameters.

✉ Sukhpal Singh Gill
ssgill@thapar.edu

Rajkumar Buyya
rbuyya@unimelb.edu.au

Inderveer Chana
inderveer@thapar.edu

Maninder Singh
msingh@thapar.edu

Ajith Abraham
ajith.abraham@ieee.org

[1] Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Parkville, Australia

[2] Computer Science and Engineering Department, Thapar University, Patiala, Punjab, India

[3] Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Auburn, WA, USA

Keywords Cloud workload · Cloud computing · Resource scheduling · Quality of Service · Particle swarm optimization · Energy consumption · Resource provisioning

1 Introduction

Cloud computing enables resources (Infrastructure, Platform or Software) to be offered as services. These resources are provided using a pay-as-you-use pricing plan [1]. The services offered to the users consist of set of components, which may be offered by different providers. To satisfy the request of customers, service must be provided in accordance with the required level of Quality of Service (QoS). QoS is the capability to guarantee a definite level of performance based on the parameters described by consumer and Service Level Agreement (SLA) in an authorized agreement that describes QoS [2]. One of the major challenges in the current cloud solutions is to provide the required services according to the QoS level expected by the user. Cloud service providers want to confirm that sufficient amount of resources are provisioned to ensure that QoS requirements of cloud service consumers such as deadline, execution time and budget restrictions are met. However, executing too many workloads on a single resource will cause workloads to interfere with each other and result in degraded and unpredictable performance which, in turn, discourages the users [3]. The mapping of workloads to appropriate resources for execution in cloud environment is a complex task and it can be solved by using optimization algorithms. Through these techniques, effective scheduling of resources can be done after resource provisioning. Dispersion, heterogeneity and uncertainty of resources brings challenges to resource allocation, which cannot be satisfied with traditional resource allocation policies in Cloud [4]. Thus, there is a need to make cloud services and cloud-oriented applications efficient by taking care of these properties of the cloud environment. Resource scheduling aims to allocate appropriate resources at the right time to the right workloads, so that applications can utilize the resources effectively which leads to maximization of scaling advantages [5, 6]. The minimum amount of resources should be used for a workload execution to maintain a desirable level of QoS, or minimize workload completion time of a workload. To address this problem, efficient solution should be developed which schedules the provisioned resources efficiently by considering energy consumption, execution cost and execution time as important QoS parameters.

In our earlier work [7–9], we have identified various research issues related to QoS and SLA for cloud resource scheduling and based on these challenges, we have developed a QoS based resource provisioning technique (Q-aware) to map the resources to the workloads based on user requirements. The main aim of Q-aware is to analyze the workloads, categorize them on the basis of common patterns and then provision the resources for execution of cloud workloads before actual resource scheduling. For resource scheduling, resource scheduling framework (QRSF) has been proposed, in which resources have been scheduled by using different resource scheduling policies (cost, time, cost-time and bargaining based). Earlier research work [7, 8] have been implemented in simulated cloud environment and only focus on two QoS parameters (execution cost and time).

The motivation of our research work emerges from the challenge of finding the best resource workload pair according to customer requirements. In real life situations, there are three main QoS constraints that need to be considered for efficient utilization of resources: (1) minimizing the execution time and energy consumption, (2) minimizing the execution cost and at the same time meeting the cloud workload deadline and (3) increasing user satisfaction. In this research work, we have extended our previous research work [7] by proposing *particle swarm optimization **B**ased resource provisioning and sched**UL**ing technique in c**L**oud **E**nvironmen**T*** called ***BULLET*** which focuses on other QoS parameters (availability, resource utilization, latency and reliability) also along with energy consumption, execution cost and time and requires minimum user involvement during execution of workload. The main objectives of proposed technique are: (1) identifying the QoS requirements of a workload, (2) clustering of workloads is done through workload patterns, (3) *k-means-based* clustering algorithm is used for re-clustering of workloads after assigning weights to quality attributes of each workload and (4) resources are provisioned for clustered workloads by the resource provisioner based on their QoS requirements before actual resource scheduling. Further, proposed technique is modeled and simulated a cloud environment using CloudSim to validate and optimize QoS parameters.

Initially, resource provisioning takes slight more time to identify the best resources based on QoS requirements of a particular workload, but later on it improves overall efficiency of resource management. Thus, the queuing time and over-utilization and under-utilization of resources can be avoided or be assuaged. Further, proposed technique outperforms as it adjusts the resources at runtime according to the QoS requirements of workload. The paper is structured as follows: in Sect. 2, related work of resource scheduling along with paper contribution has been presented. PSO based resource scheduling technique has been presented in Sect. 3. Experimental setup and results has been presented in Sect. 4. In Sect. 5, conclusions and the future scope have been presented.

2 Related Work

Scheduling of workloads in a cloud environment is challenging due to dynamic and heterogeneous resources spread over geographical area. Most of the reported research deals with workload management systems in a cloud computing environment on the basis of resource requirements. Cloud computing offers dynamic and flexible resource allocation for reliable and guaranteed services in pay according to use fashion. Many cloud consumers can demand number of cloud services concurrently [10, 11]. Subsequently, there is a need to provide all the resources to requesting cloud consumer in a well-organized way to fulfill their requirements.

2.1 Cloud Resource Scheduling

Varalakshmi et al. [12] described an Optimal Workflow based Scheduling (OWS) framework to discover a solution that tries to meet the user-desired QoS constraints i.e. execution time. This research shows little improvement in resource utilization and it does not consider cost as one of the QoS parameters. Xing et al. [13] presented an ant colony optimization (ACO) based job scheduling framework, which adapts to dynamic characteristics of Cloud computing and incorporates particular benefits of ACO in NP-hard problems. This approach reduced only job completion time based on pheromone. Topcuoglu et al. [14] presented the Heterogeneous Earliest Time First (HEFT) framework to discover the average execution time of each workload and also the average communication time among the resources of two workloads. The workload with higher rank value is given higher priority. In the resource selection stage workloads are scheduled in priorities and each workload is allocated to the resource that complete the workload at the earliest time. The framework has not been designed to reduce cost and time. El-kenawy et al. [15] proposed a RASA based scheduling framework to select the jobs based on execution time instead of overall completion time. This technique shows achieving schedules with comparable lower execution time as compared to original Max–Min and Resource Aware Scheduling Algorithm (RASA) by considering only provider's benefit. Lin et al. [16] suggested compromised cost time based resource scheduling policy which considers cost-constrained workflows and taking execution time and cost as QoS parameters. This approach meets user designed deadline and achieve lower cost simultaneously but not considering heterogeneous workflow instances. Verma and Kaushal [17] presented Deadline and Budget Distribution-based Cost-Time Optimization (DBD-CTO) workflow scheduling framework that minimizes execution cost while meeting deadline without considering energy consumption and heterogeneous cloud workloads.

2.2 PSO Based Cloud Resource Scheduling

Pandey et al. [18] introduced a particle swarm optimization (PSO) based heuristic framework (PSO-H) to schedule the applications to Cloud resources that proceeds both computation and data transmission cost. It is used for workflow applications by changing its computation and communication costs. The assessment results show that PSO can reduce the cost and good sharing of workload onto resources. They did not consider execution time of workloads. Somasundaram and Govindarajan [19] presented PSO based resource scheduling mechanism (PSO-HPC) to reduce makespan, price, job rejection ratio and maximize jobs meeting deadline for HPC applications. MATLAB programming environment is used to simulate the HPC applications and resources and verified this technique on Eucalyptus-based cloud environments and results depicted that this technique is efficient in reducing job rejection ratio and execution cost, and improves user's satisfaction without focusing energy consumption. Netjinda et al. [20] described PSO based scheduling technique (PSO-SW) to achieve scientific workflow execution within the particular deadlines. This approach is used to identify the configuration requirements with minimum cost

Table 1 Comparison of proposed technique with existing resource scheduling techniques

Technique	Provisioning based scheduling	Workload type	Clustering of workloads	QoS parameters
OWS [12]	×	Homogenous	×	Execution time
ACO [13]	×	Homogenous	×	Completion time
HEFT [14]	×	Homogenous	×	Communication time
RSA [15]	×	Homogenous and heterogeneous	×	Execution time
CTC [16]	×	Homogenous	×	Communication cost
DBD-CTO [17]	×	Homogenous	×	Execution time
PSO-H [18]	×	Homogenous	×	Computation and communication cost
PSO-HPC [19]	×	Homogenous and heterogeneous	×	Execution cost
PSO-SW [20]	×	Homogenous	×	Execution time
PSO-DVFS [21]	×	Homogenous	×	Energy
BULLET (Proposed)	√	Homogenous and heterogeneous	√	Execution time, cost, energy, availability, resource utilization, latency and reliability

to execute the particular workflow application and executed the applications with minimum cost without degradation in performance but execution time is not considered as a QoS parameter. Yassa et al. [21] described Dynamic Voltage and Frequency Scaling (DVFS) and PSO based scheduling policy (PSO-DVFS) for scientific workloads to reduce consumption of power in which different levels of voltage supply workloads are used through sacrificing clock frequencies. This multiple voltage involves a compromise between the quality of schedules and energy but execution time and cost are not considered as a QoS parameter. Proposed technique (BULLET) has been compared with existing resource scheduling techniques as described in Table 1.

These research works have considered only one of the QoS parameters from energy, cost and time but all the three parameters have not been considered simultaneously in any of the existing work to the best of the knowledge of the authors. Moreover, most of the existing work considers homogeneous cloud workloads. PSO based resource scheduling technique considers the basic features of cloud computing in order to execute the heterogeneous cloud workloads with minimum execution cost, time and energy consumption along with other QoS parameters.

2.3 Our Contributions

We present a PSO based resource scheduling technique for both homogenous and heterogeneous cloud workloads. This is an extension of our previous work

[7]. The proposed technique focuses on how to map the cloud workload in order to improve execution cost, time and energy along with other QoS parameters (availability, resource utilization, latency and reliability). The proposed technique has been evaluated in simulated cloud environment using CloudSim. The performance of proposed technique has also been tested on cloud testbed using synthetic workloads for different QoS parameters. We have then compared the experimental results of proposed technique with existing PSO based resource scheduling techniques. The main contribution of this paper is: (1) scheduling technique for effective management of resources is proposed, (2) performance of proposed technique has been evaluated in cloud environment using CloudSim, (3) optimized important QoS parameters such as execution cost, energy consumption and execution time and (4) improved the customer satisfaction and queuing time, over and under-utilization of resources can be avoided or be assuaged.

3 BULLET: Proposed PSO Based Resource Scheduling Technique

In cloud computing, resource scheduling is the core of resource management system. It essentially indicates mapping of cloud workloads to the appropriate resources from the available resource pool. This process searches the best resource and maps with cloud workload based on consumer requirements. Process of resource scheduling comprises of four steps. In first step, workloads are analyzed and clustered based on their requirements. Second step, identifies the required set of resources from resource pool. Third step, maps the cloud workloads with appropriate resources based on QoS requirements as specified by user. In Final step, schedule the resources to execute workloads therefore further guaranteeing near optimal satisfaction of QoS requirements. Need of optimized resource scheduling in cloud is achieved using proposed technique. For example, assume that a customer wants to purchase some items from grocery store, then salesman would ask the requirements in terms of budget etc. and then salesman will display the items accordingly. Based on the money they want to spend and other requirements and constraints, select the particular item among all the displayed items. Figure 1 shows the architecture of proposed technique (BULLET).

3.1 Resource Provisioning

The resource provisioning technique comprises of following units:

1. *Bulk of Workloads* Bulk of Workloads (BoW) are coming for execution and are processed and stored in workload queue.
2. *Workload Resource Manager* Workload Resource Manager (WRM) contains the information about resources, QoS metrics and SLA to provision the

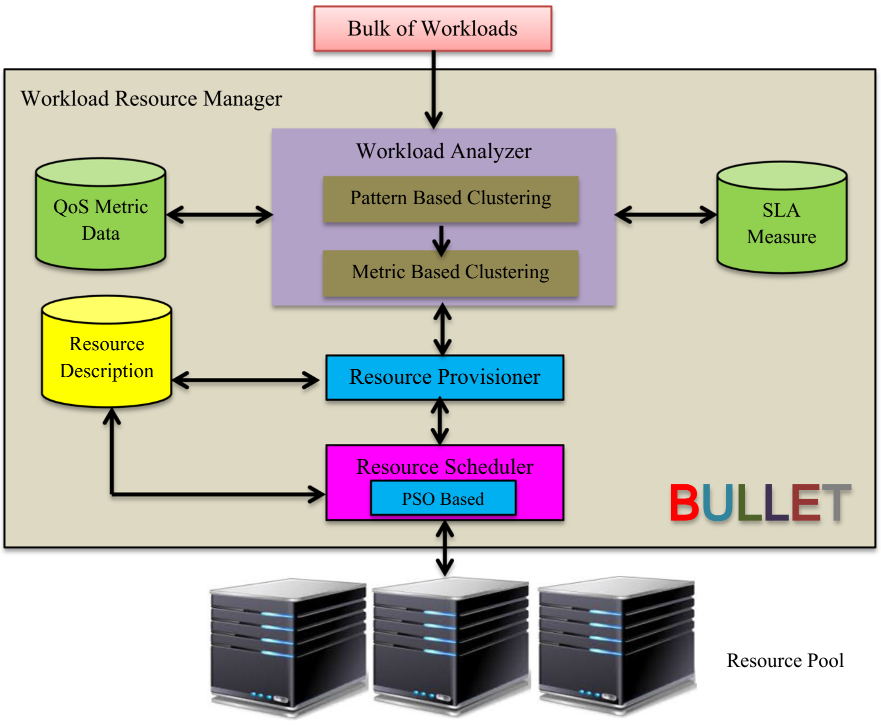

Fig. 1 Architecture of BULLET

resources for execution of workloads based on QoS requirements described by cloud consumer.

3. *SLA Measure* WRM receipts the information from the suitable Service Level Agreement (SLA). After studying and confirming the various QoS constraints which the workload has required, WRM checking the availability of resources.
4. *QoS Metric Data* It contains the information regarding QoS metrics used to calculate weight for clustering of workloads.
5. *Workload Analyzer* The aim of Workload Analyzer is to look at different characteristics of a cloud workload to determine the feasibility of porting the application in the cloud. The different cloud workloads have different set of QoS requirements and characteristics. All the workloads are submitted to WRM are analyzed based on their QoS requirements. For QoS, the required workload patterns are identified for clustering of workloads then identifies the metrics required to assign the weights based on level of measurement (Sect. 3.1.1) described in QoS requirements specified in SLA. K-Means based clustering algorithm is used for re-clustering the workloads for execution on different set of resources.
6. *Resource Information* The resource details include the number of CPU using, size of memory, cost of resources, type of resources and number of resources. All the common resources are stored in resource pool.

7. *Resource Provisioner* It provides the demanded resources to the workload for their execution in cloud environment only if required resources are available in resource pool. If the required resources are not available according to QoS requirement then the WRM asks to resubmit the workload with QoS requirement in the form of SLA. After the provisioning of resources, workloads are submitted to resource scheduler. Then the resource scheduler will ask to submit the workload for resources provisioned. After this resource scheduler send back the results to WRM, cloud workload contains the resource information.
8. *Resource Scheduler* It will execute all the workloads on provisioned resources efficiently and described in Sect. 3.2.

Cloud workload is an abstraction of work of that instance or set of instances executing on the appropriate resources with different QoS requirements submitted by cloud consumer as a type of application. The types of workload that have been considered for this research work are: websites, technological computing, endeavor software, performance testing, online transaction processing, e-commerce, central financial services, storage and backup services, production applications, software/project development and testing, graphics oriented, critical internet applications and mobile computing services [7, 8].

3.1.1 Clustering of Workloads

Based on the important features of Cloud workloads and workload patterns the clustering of Cloud workloads has been done and process of clustering has been described in our previous research work in detail [7]. The outcome of pattern based workload clustering is shown in Table 2.

Further, *K-means* based clustering algorithm is used for re-clustering the workloads for execution on different set of resources and process of clustering using *K-means* based clustering algorithm has been described in our previous research work in detail [8]. Final set of workloads is shown in Table 3.

3.2 Resource Scheduling

We have designed particle swarm optimization (PSO) based resource scheduling algorithm by considering different QoS parameters (execution time, cost and energy consumption).

3.2.1 Requirements

Following are some important requirements to design an efficient resource scheduling algorithm:

Efficiency Provisioning of resources offers the facility to reduce the cloud overheads which requires QoS based efficient management of resources.

Table 2 Cloud workloads and their QoS requirements after pattern based clustering

Workload	QoS requirements
Websites	Reliable storage, high network bandwidth, high availability
Technological computing	Computing capacity, reliable storage
Endeavour software	Security, high availability, customer confidence level, correctness
Performance testing	Execution time, energy consumption and execution cost
Online transaction processing	Security, high availability, internet accessibility, usability
E-Com (E-commerce)	Variable computing load, customizability
Central financial services	Security, high availability, changeability, integrity
Storage and backup services	Reliability, persistence
Productivity applications	Network bandwidth, latency, data backup, security
Software/project development and testing	User self-service rate, flexibility, creative group of infrastructure services, testing time
Graphics oriented	Network bandwidth, latency, data backup, visibility
Critical internet applications	High availability, serviceability, usability
Mobile computing services	High availability, reliability, portability

Table 3 K-means based clustering of workloads

Cluster	Cluster name	Workloads
C1	Compute	Technological computing, performance testing
C2	Storage	E-Com and storage and backup services
C3	Communication	Websites, critical internet applications, mobile computing services
C4	Administration	Endeavour software, online transaction processing, central financial services, productivity applications, software/project development and testing and graphics oriented

Efficient Resource Usage Efficient scheduling of resources should minimize wastage of the resources. Different cloud workloads are waiting for execution should be executed with maximum resource utilization of resources and optimizing QoS parameters (execution time and cost).
Fair Scheduling The number of resources allotted to every consumer should be independent of number of cloud workloads each user submits.
Adaptability and Scalability A smart scheduler adapts as per the resources, i.e. whenever resources join or leave (dynamically), it manages the resources and workloads' execution process efficiently.

3.2.2 Problem Formulation

Cloud resource scheduling is a tedious task due to the problem of finding the best match of resource-workload pair based on the user QoS requirements. The goal of Cloud workload analyzer is to categorize the workloads and the goal of resource scheduler is to map and schedule the workloads effectively and efficiently. The resources and Cloud workloads can leave and join the Cloud dynamically. Cloud resources are heterogeneous and dynamic in nature. In this work, independent Cloud workloads have been considered to handle the realistic scenarios as there are many scenarios in which the need of scheduling Cloud workloads arises. Firstly, this problem is suitable to Cloud systems because of the nature of Cloud customers, who submit Cloud workloads in an independent manner to the system. Secondly, Cloud systems are most useful for massive parallel processing, in which large amounts of data are processed independently. In this work, the scheduling of workloads has been considered from both the Cloud customer and Cloud provider's point of view. The user wants to minimize the cost whereas the Cloud provider wants to minimize the execution time and energy consumption. In this problem, the most popular and extensively studied optimization criteria, i.e. the minimization of the execution time has been considered. Execution time is used to indicate the general productivity of the Cloud systems. Smaller values of execution time and energy consumption indicate that the scheduler is planning the Cloud workloads in an efficient manner. Cost is another optimization criterion, which refers to the total cost of the Cloud workload execution on a particular resource. The problem has been derived to get an optimal solution.

The problem can be expressed as: to consider this problem, a set of independent Cloud workloads $\{w_1, w_2, w_3, \ldots, w_m\}$ to map on a set of heterogeneous and dynamic resources $\{r_1, r_2, r_3, \ldots, r_n\}$ has been taken. For continuous problem, $\mathrm{R} = \{r_k | 1 \leq \mathrm{k} \leq \mathrm{n}\}$ is the collection of resources and n is the total number of resources. $\mathrm{W} = \{w_i | 1 \leq \mathrm{i} \leq \mathrm{m}\}$ is the collection of Cloud workloads and m is the total number of Cloud workloads. The estimated time to compute the value of each Cloud workload on each resource is assumed to be given by the consumer-supplied information (data-driven). Under the Predictable Time to Compute (PTC), the following assumptions have been considered:

1. Each Cloud workload to be scheduled for application's execution has a unique *workload id.*
2. Cloud workloads are independent.
3. Arrival of Cloud workloads for execution of application is random and Cloud workloads are placed in a queue of unscheduled Cloud workloads.
4. The processing speed of the resources is measured in Multiple Instructions Per Second (MIPS) as per the Standard Performance Evaluation Corporation (SPEC) benchmark.
5. The processing requirement of a Cloud workload is measured in Million Instructions (MIs).
6. Execution time for every Cloud workload on a resource is obtained from objective function. [*Number of workloads* × *number of resources*] for every workload on resources is calculated from PTC matrix. Columns of PTC matrix demonstrate the estimated execution time for a specific resource while rows on PTC matrix demonstrate the execution time of a workload on every resource. PTC (w_i, r_k) is the expected execution time of workload w_i and the resource r_k.

3.2.3 Objective Function

In Cloud computing, provider wants to minimize the execution time and energy consumption while user wants to minimize the cost for Cloud workload. The goal of an objective function is to optimize the QoS parameters (execution cost, time and energy consumption) for finishing all n workloads of a given Bulk of Workloads (BoW). This objective function successfully captures the compromise among QoS parameters as specified in Eq. (1). Further formally, the workload assignment problem with the energy, cost and time function of each resource r can be generally formulated as follows:

$$Fitness\,value = \alpha\, execution_{cost} + \beta\, execution_{time} + \gamma\, energy_{consumption} \tag{1}$$

where $0 \leq \alpha < 1,\ 0 \leq \beta < 1\ \mathit{and}\ 0 \leq \gamma < 1$ are weights to prioritize components of fitness function.

(a) *Execution Cost* ($execution_{cost}$) It is the cost spend to execute workload and measured in terms of Cloud Dollars (C$):

$$execution_{cost} = min(c(r_k, w_i)) \quad for\, 1 \leq k \leq n\, and\, 1 \leq i \leq m \tag{2}$$

where $c(r_k, w_i)$ is the cost of workload w_i which executes on resource r_k as defined below:

$$\mathrm{c}(r_k, w_i) = \sum_{w_i \in W} \left(\frac{completion(w_{i,} r_k)}{completion_{m(w_i)} \times W} \right) \tag{3}$$

W is the collection of cloud workloads.

$$\mathrm{c}(r_k, w_i) = \frac{1}{W} \sum_{w_i \in W} \left(\frac{completion(w_{i,} r_k)}{completion_{m(w_i)}} \right) \tag{4}$$

where as:

$$completion_{m(w_i)} = \max_{w_i \in W, r_k \in R} completion(w_{i,} r_k) \tag{5}$$

$$execution_{time} = min(L_{w_i}) w_i \in W \tag{6}$$

(b) *Execution Time* ($execution_{time}$) It is the finishing time L_w of the latest workload and can also be represented as PTC workload w_i on resource r_k. Before estimation of execution time, completion time of a resource should be defined. Completion time can be defined as the time in which resource can finish the execution of all the previous workloads in addition to the execution of workload w_i on resource r_k as described as:

$$completion(r_k) = available\_time_{r_k} \pm PTC_{m(w_i)} \tag{7}$$

where as:

$$PTC_{m(w_i)} = \max_{w_i \in W, r_k \in R} PTC(w_{i,} r_k) \tag{8}$$

(c) *Energy Consumption* (*EC*) The energy model is devised on the basis that resource utilization has a linear relationship with energy consumption [22]. Energy Consumption of using resources can be expressed as the following formula (Eq. 9):

$$EC = EC_{Datacenter} + EC_{Transceivers} + EC_{Memory} + EC_{Extra} \quad (9)$$

$EC_{Datacenter}$ represents the datacenter's energy consumption, $EC_{Transceivers}$ represents the energy consumption of all the switching equipment. EC_{Memory} represents the energy consumption of the storage device. EC_{Extra} represents the energy consumption of other parts, including the fans, the current conversion loss and others. $EC_{t\cdot i}$ is the energy consumption at given time t is defined in (Eq. 10):

$$EC_{t,i}(r) = q \times EC_{max} + (1 - q) \times EC_{max} \times ru \quad (10)$$

where EC_{max} is maximum energy consumption while resource is fully utilized, q is fraction of energy consumed by idle resource and ru is resource utilization. Resource utilization is change over time and it is function of time and presented as $ru(t)$. For a resource r_t at given time t, the resource utilization $ResU_t$ is defined as (Eq. 11):

$$ResU_t = \sum_{i=1}^{n} EC_{t,i}(ru(t))dt \quad (11)$$

where n is the number of cloud workloads running at time t. The actual energy consumption EC_{actual} of a resource ru_t at given time t is defined as (Eq. 12):

$$EC_{actual} = (EC_{max} - EC_{min}) \times ResU_t + EC_{min} \quad (12)$$

where EC_{max} is the energy consumption at the peak load (or 100% utilization) and EC_{min} is the minimum energy consumption in the active/idle mode (or as low as 1% utilization).

3.2.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is a group based intelligence algorithm which is inspired by the social behavior such as school of fish defending themselves from a hunter (predator) or group of birds finding a source of food [23]. Population in PSO algorithm is defined as the total number of particles in a problem space and particles in population are initialized randomly. In every generation, fitness value of every particle is estimated by a fitness function to be improved. Both the positions of the particle is known: (1) best position (local best i.e. L_{BP}) of a particle and (2) global best [the best position so far among the whole group of particles (G_{BP})]. G_{BP} is s the best particle in terms of fitness in an whole population, whereas L_{BP} of a particle is the best result (fitness value) so far reached by the particle. Particle's position and velocity is updated in every generation by using (Eq. 1).

PSO optimization technique which works based on global search. There is no straight re-combination of individuals of the population in algorithm of PSO like other population-based algorithms (or evolutionary algorithms) such as GA (Genetic Algorithm) etc. Algorithm of PSO is dependent on particle's social behavior. Every individual particle regulates its path based on the position of the best particle (global best) of the whole population and its best position (local best) in every generation. Stochastic nature of the particle increases due to this property of PSO and touches rapidly to global minima with a realistic noble solution [23]. PSO has become prevalent due to its easiness and its usefulness in extensive range of application with little cost of computation. Applications like pattern recognition, reactive data mining etc. have used PSO mostly. Along with these applications PSO has been solved various NP-Hard problems like workload allocation and resource scheduling.

PSO Terminology The PSO terminology used in this research paper is described below:

(a) *Particle* A particle in particle swarm optimization is similar to a fish or bird flying through a search space (problem area). Every particle's movement is synchronized by a velocity which has both direction and magnitude. Position of every particle at any time instance is influenced by its best position (L_{BP}) and the position of the best particle (G_{BP}) in a search space. Fitness value is used to measure the performance of a particle, which is problem oriented. For this research work, a workload is considered as a particle.
(b) *Population Size* In this research work, size of set of workloads (number of workloads considered as a population size).
(c) *Random Velocity* Every particle's movement is the composition of an initial random velocity and two randomly weighted effects: (1) the affinity to return to the best previous position of particle and (2) the affinity to move towards the best previous position of neighborhood. Based on these two affinities of workload, the mapping of workload with resources is done. Workload will be executed only with that resource which has high value of fitness.
(d) *Particle Velocity* It is calculated based on the probability distribution for the particle position, that is, the particle (workload) position in a dimension is randomly generated using that distribution.
(e) *Particle Position* Current state of particle (workload), state may be submission state, waiting state, ready state, execution state and completion state.
(f) *Global Best Position* (G_{BP}) Best position of particle (workload) among the whole group of particles (set of workloads).
(g) *Local Best Position* (L_{BP}) Best position of particle (workload) as reached by the particle (workload regulates its path based on its best resource which executes workload with minimum fitness value).

PSO Based Resource Scheduling Algorithm In this section, we present the pseudo code of PSO-based algorithm for resource scheduling in the Cloud environment. Each particle in genome is a partial solution and is represented as a resource identifier (e.g. select, move, swap, drop) or a sequence of resource identifiers. The non-PSO based resource identifier can be simple or complex and are implemented as follows:

(a) Workload selection and scheduling: the resource identifiers select workload from the unscheduled list and schedule it into the best available resource.
(b) Try for the best combination of all workloads and resources until the best combination is found.
(c) Move workload (w_i) from its current resource/schedule.
(d) Swap workloads: select the workloads randomly which can swap.
(e) Remove a randomly selected workload from workload queue already scheduled.

This is the only heuristic which will move the search into an infeasible region because any workload may be unscheduled. We make sure that the search can move back into its feasible region by un-scheduling workload that has other valid resources so that it can move into the next iteration. The non-PSO based resource identifier is then applied so as to find an optimal solution of the problem instance. The objective of PSO is to find the best resource identifier that generates the best solution for resource scheduling problem. The selection process of non-PSO based resource identifier stops after a pre-defined number of iterations. We set a fixed number of iterations to keep the computation time low. The particle rejects the new solution if it is poorer than the current solution. The pseudo code of PSO based resource scheduling algorithm in Fig. 2.

- A resource list is then obtained from the resource provisioning unit after provisioning of user's workloads [7]. Once the resource list has been obtained, a workload list and a random feasible solution are initialized.
- The task to choose the best heuristic from low-level heuristics is started.
- We have a number of workloads, each of which represents a resource identifier supplied with an initial solution in the solution space and an access to the evaluation function.
- Workload's position and Workload's velocity would be randomly initialized.
- It will then select a low-level heuristic at each workload position and compute its fitness function i.e. Fitness (L_{BP}).
- If at P_p, Fitness (L_{BP}) is better than Fitness (G_{BP}) then G_{BP} takes the value of L_{BP}.
- We will try to find the Fitness value at best global position of the workload.
- After a workload has been chosen from the population, its position and velocity would be updated using (Eq. 1). Then, its fitness at the new position is calculated and compared with its previous position.
- If it is better than the local best value then we will assign workload's current position to the local best value.
- Now, we will compare fitness at L_{BP} and G_{BP}. If the fitness at L_{BP} is better than at G_{BP} then we will assign the value of L_{BP} to G_{BP}.
- After selection of a low-level heuristic, it is then applied to the problem. Resource scheduling is performed till there are no unscheduled jobs in the queue.

```
Algorithm 1: PSO based Resource Scheduling Algorithm
1.  Input Data: Number of workloads and number of available resources.
2.  Result: Mapping of the each workloads to the resources.
3.  start
4.      initialize Resource list [Number of Resources]
5.      initialize Workload List [Number of Jobs]
6.      initialize a random feasible solution
7.      S= The number of particles in the population
8.      PS = Population Size
9.      RV = Random Velocity
10.     $P_v$ = Particle Velocity
11.     $P_p$ = Particle Position
12.     $P_{op}$ = Population
13.     $G_{BP}$ = Global Best Position
14.     $L_{BP}$ = Local Best Position
15.     for $i = 1$ To Populationsize do
16.         $P_v \leftarrow RV()$
17.         $P_p \leftarrow$ Random_Position ($PS$)
18.         $L_{BP} \leftarrow P_p$
19.
20.         For each particle, calculate the fitness value using [Eq. 1]
21.         if Fitness ($G_{BP}$) $\geq$ Fitness ($L_{BP}$) then
22.             $G_{BP} \leftarrow L_{BP}$
23.     while maximum iteration is not satisfied do
24.         for $P \in P_{op}$ do
25.         $P_v \leftarrow$ UpdateVelocity ($P_v$, $G_{BP}$, $L_{BP}$)
26.         $P_p \leftarrow$ UpdatePosition ($P_p$, $P_v$)
27.         if Fitness ($P_p$) $\leq$ Fitness($L_{BP}$)
28.         then
29.            $L_{BP} \leftarrow P_p$
30.           if ($L_{BP}$) $\leq$ Fitness($G_{BP}$) then
31.               $G_{BP} \leftarrow L_{BP}$
32.     Return ($G_{BP}$)
33.     while there are unscheduled workloads in the queue do
34.         for every resource is in resource list do
35.             get the next workload from queue
36.             schedule the workload on the resource on the basis of fitness
37.     Repeat each and every step till all the workloads are allocated
38. End
```

Fig. 2 PSO based resource scheduling algorithm

3.3 Cloud Workload Execution Design

There are eight different classes created to represent the interaction among different workload related entities. Workloads are scheduled to the appropriate resources based on the workload description by taking care of QoS. Fitness value for every workload is calculated and analysed. Schedule the resources to cloud workloads and execute within the defined budget and desired deadline with minimum energy consumption. The interaction among different classes for workload execution is shown in Fig. 3. In the course of its lifetime, a workload passes through many states as is outlined in Fig. 4. A workload is an input to the scheduler which allocates it to a set of provisioned resources based on its requirements. The workload's status is then changed to SCHEDULED. During the STAGE_IN state, input files and executables required for the workload are staged to the available provisioned

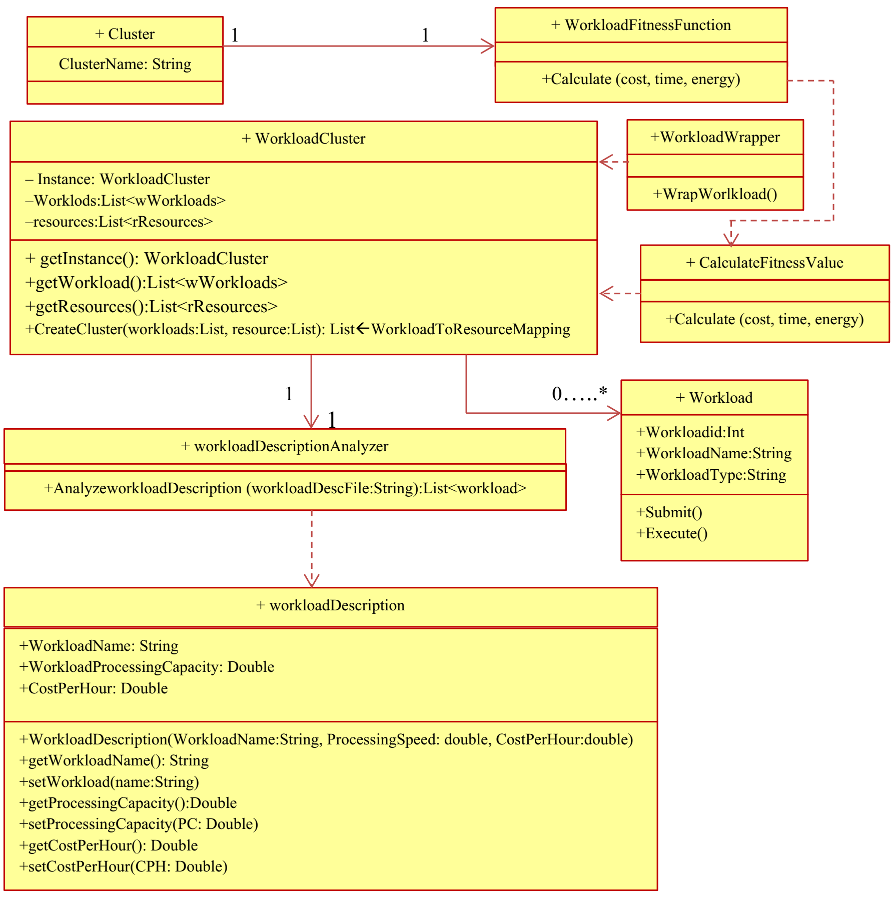
+ Cluster
ClusterName: String
1
1
+ WorkloadFitnessFunction
+Calculate (cost, time, energy)
+ WorkloadCluster
– Instance: WorkloadCluster
–Worklods:List<wWorkloads>
–resources:List<rResources>
+ getInstance(): WorkloadCluster
+getWorkload():List<wWorkloads>
+getResources():List<rResources>
+CreateCluster(workloads:List, resource:List): List←WorkloadToResourceMapping
+WorkloadWrapper
+WrapWorlkload()
+ CalculateFitnessValue
+Calculate (cost, time, energy)
1
1
0…..*
+ workloadDescriptionAnalyzer
+AnalyzeworkloadDescription (workloadDescFile:String):List<workload>
+ Workload
+Workloadid:Int
+WorkloadName:String
+WorkloadType:String
+Submit()
+Execute()
+ workloadDescription
+WorkloadName: String
+WorkloadProcessingCapacity: Double
+CostPerHour: Double
+WorkloadDescription(WorkloadName:String, ProcessingSpeed: double, CostPerHour:double)
+getWorkloadName(): String
+setWorkload(name:String)
+getProcessingCapacity():Double
+setProcessingCapacity(PC: Double)
+getCostPerHour(): Double
+setCostPerHour(CPH: Double)

Fig. 3 Interaction among different classes for workload execution

resource. When this process is completed successfully, then a workload is considered to be SUBMITTED. The workload may be queued while waiting for an available processor and its state changes to PENDING. When the workload starts its execution, it is considered ACTIVE. After the workload has finished executing, it enters the STAGE_OUT stage where its output files are transferred back to the broker. If all its outputs are received and are as expected by the task requirements, then the workload is considered as "DONE". If one of state transition fails on the available side or the workload is completed on the available side but has not produced the expected result files, then it is considered FAILED and is reset and marked for re-scheduling.

3.4 QoS Metrics

The following other metrics (Eqs. 13–18) are selected from our previous work [7–9] to measure the value of QoS parameters other than energy, cost and time.

Availability (A) It is a ratio of Mean Time Between Failure (MTBF) to addition of Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR). We have used following formula to calculate availability (Eq. 13).

$$A = \frac{MTBF}{MTBF + MTTR} \tag{13}$$

where Mean Time Between Failure (MTBF) is ratio of total uptime to number of breakdowns (Eq. 14).

$$MTBF = \frac{Total\ Uptime}{Number\ of\ Breakdowns} \tag{14}$$

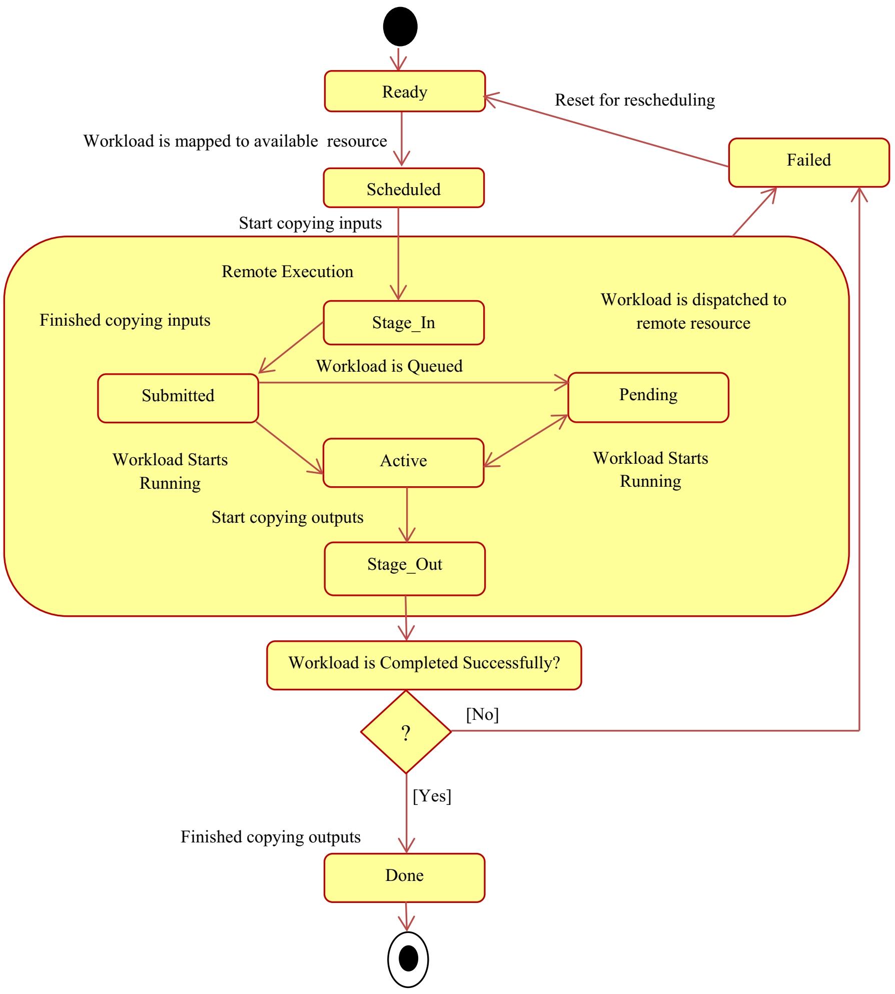

Fig. 4 Different stages for cloud workload

where Mean Time To Repair (MTTR) is ratio of total downtime to number of breakdowns (Eq. 15).

$$MTTR = \frac{Total\,Downtime}{Number\,of\,Breakdowns} \tag{15}$$

Reliability (re) Reliability of the resource has to be checked for scheduling of the resources. With the help of reliability parameter, we can check the fault tolerance of the resource. Reliability of the resource is calculated with the following formula (Eq. 16) as:

$$re = e^{-\lambda t} \tag{16}$$

re = reliability of resource, e = exponential function, t = time for resource to deal with its request for any workload's execution and λ = the failure rate of the resource at the give time.

Resource Utilization (RU) It is a ratio of execution time of a workload executed by a particular resource to total uptime of that resource. We have used following formula to calculate resource utilization (Eq. 17).

$$\mathrm{RU} = \sum_{i=1}^{n}\left(\frac{execution\,time\,of\,a\,workload\,executed\,ith\,resource}{total\,uptime\,of\,ith\,resource}\right) \tag{17}$$

where n is number of workloads.

Latency (L) It is defined as a difference between expected execution time and actual execution time. We have used following formula to calculate Latency (Eq. 18):

$$\mathrm{L} = \sum_{i=1}^{n}(Expected\,Execution\,Time_i - Actual\,Execution\,Time_i) \tag{18}$$

where n is number of workloads.

4 Experimental Setup and Results

We modeled and simulated a cloud environment using CloudSim [6]. We simulated computing nodes that resembles configuration of resources shown in Table 4. The workload is modeled as processing of images to convert from one format to another (e.g., converting from JPEG to PNG format). The characteristics of resources and cloudlets that have been used for all the experiments has been described in our previous research work [7, 8]. User cloud workloads are modeled as independent parallel applications which are computation intensive. Thus the data dependency among the cloud workloads in the parallel applications is negligible. Each cloud workload is parallel and is hence considered to be independent of any other cloud workload. In this experimental setup, three different cloud platforms are used: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) as shown in Fig. 5.

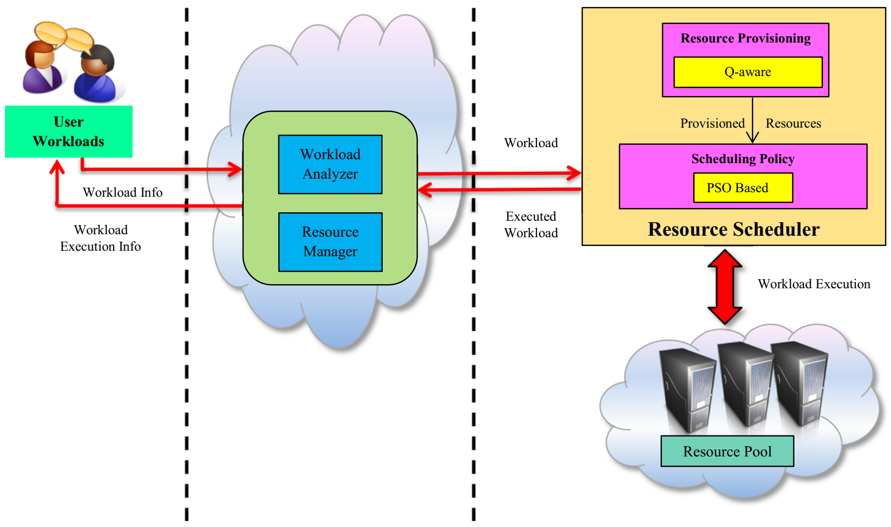

Fig. 5 Cloud testbed

At ***SaaS level***, Microsoft Visual Studio 2010 is used to develop *Cloud Workload Management Portal (CWMP)* to provide user interface in which user can access service from any geographical location. At ***PaaS level***, Resource Manager [9, 23] is used as a scalable cloud middleware to make interaction between IaaS and SaaS, and continually monitor the performance of the system. A task is a single unit of work processed (request) in a node. It is independent from other tasks that may be executed on the same or any other node at the same time. At ***IaaS level***, three different servers (consist of virtual nodes) have been created and SQL Server has been used for data storage. Scheduler as shown in Fig. 5, runs at IaaS level on Server. Computing nodes used in this experiment work are further categorized into three categories as shown in Table 4.

Table 4 Configuration details

Resource_Id	Configuration	Specifications	Operating system	Number of virtual node	Number of ECs	Price (C$/EC time unit)
R1	Intel Core 2 Duo—2.4 GHz	1 GB RAM and 160 GB HDD	Windows	6	18	2
R2	Intel Core i5-2310—2.9 GHz	1 GB RAM and 160 GB HDD	Linux	4	12	3
R3	Intel XEON E 52407—2.2 GHz	2 GB RAM and 320 GB HDD	Linux	2	6	4

The execution cost is calculated based on user workload and deadline (if deadline is too early (urgent) it will be more costly because we need a greater processing speed and free resources to process particular workload with urgency. There individual price is fixed (artificially) for different resources because all the resources are working in coordination manner to fulfill the demand of user (demand of user is changing dynamically). Experiment setup using 3 servers in which further virtual nodes (12 = 6 (Server 1) + 4 (Server 2) + 2 (Server 3)) are created. Every virtual node has different number of Execution Components (ECs) to process user workload and every EC has their own cost (C$/EC time unit (Sec)). Table 4 shows the characteristics of the resources used and their Execution Component (EC) access cost per time unit in Cloud Dollars (C$) and access cost in C$ is manually assigned for experimental purposes. The access cost of an EC in C$/time unit does not necessarily reflect the cost of execution when ECs have different capabilities. The execution agent needs to translate the access cost into the C$ for each resource. Such translation helps in identifying the relative cost of resources for executing user workloads on them.

Due to limited number of resources, cost increases with increase in user workloads. Cost is varying in two different cases: (1) relaxed deadline and (2) tight deadline. In both cases, when the deadline is low (e.g. 200 s), the number of user workloads processed increases as the budget value increases. When a higher budget is available, the execution agent uses expensive resources to process more user workloads within the deadline. Alternatively, when scheduling with a low budget, the number of user workloads processed increases as the deadline is relaxed. Execution agent allocates as many user requests as the first cheapest resource can complete by the deadline, and then allocates the remaining user workloads to the next cheapest resources. When the deadline is tight (e.g. 100), there is high demand for all the resources in a short time. All the resources are used up so long as budget is available to process all user workloads within the deadline. However, when the deadline is relaxed (e.g. 700 s), it is likely that all user workloads can be completed using the first few cheapest resources. As the deadline increases, execution agent schedules user workloads on the available resources to finish earlier as possible. The aim of this performance evaluation is to demonstrate that it is feasible to implement and deploy the proposed technique on real cloud resources. The key components of the cloud environment are: user interface (CWMP), workload analyzer and resource scheduler. Figure 6 enables the understanding of the cloud based environment in which the proposed technique is implemented.

4.1 Performance Evaluation

In order to evaluate the performance of BULLET, we have compared the value of QoS parameters (execution time, cost, energy consumption, reliability, availability, latency and resource utilization) of BULLET with existing PSO based cloud based resource scheduling techniques (PSO-HPC [19], PSO-SW [20] and PSO-DVFS [21]) and all the three existing techniques have been described in Sect. 2.2. Formals used to measure the value of QoS parameters have been described in Sect. 3. We have performed experiments to determine the effect of an increase in number of

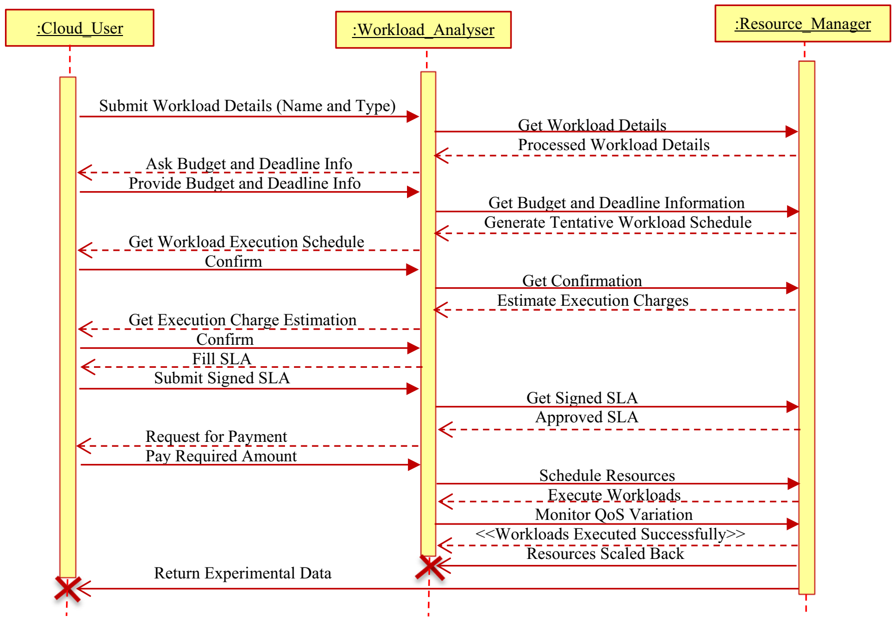
:Cloud_User
:Workload_Analyser
:Resource_Manager
Submit Workload Details (Name and Type)
Get Workload Details
Processed Workload Details
Ask Budget and Deadline Info
Provide Budget and Deadline Info
Get Budget and Deadline Information
Generate Tentative Workload Schedule
Get Workload Execution Schedule
Confirm
Get Confirmation
Estimate Execution Charges
Get Execution Charge Estimation
Confirm
Fill SLA
Submit Signed SLA
Get Signed SLA
Approved SLA
Request for Payment
Pay Required Amount
Schedule Resources
Execute Workloads
Monitor QoS Variation
<<Workloads Executed Successfully>>
Resources Scaled Back
Return Experimental Data

Fig. 6 Workload execution

workloads and resources on QoS parameters such as energy consumption, cost and time. All the experiments were started with workload name: Performance Testing [Processing Larger Image File of Size 713 MB], in which BULLET converts an image file from JPEG format to PNG format. Conversion of a single JPEG file into PNG is considered as a single workload. We have created the PTC matrix which is computed as ratio of workload and computing capacity of virtual machines.

4.2 Experimental Results

Experiment has been conducted with different number of cloud workloads (15–90) for verification of QoS parameters.

Test Case 1: Energy Consumption Versus Number of Workloads By increasing the number of cloud workloads, the value of energy consumption is increasing. The minimum value of energy consumption is 69 kWh at 15 cloud workloads for BULLET. BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS in terms of energy consumption at different number of cloud workloads as shown in Fig. 7. The average value of energy consumption in BULLET is 7.61, 11.45 and 17.19% lesser than PSO-HPC, PSO-SW and PSO-DVFS respectively.

Test Case 2: Execution Cost Versus Number of Workloads With the increase in number of workloads, execution cost rises as shown in Fig. 8. As per the number of workloads increases, BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS. BULLET outperforms as it adjusts the resources at runtime according to the

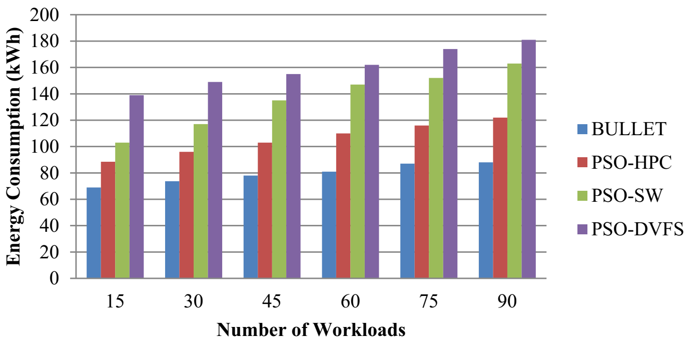

Fig. 7 Effect of change in number of workloads submitted on energy consumption

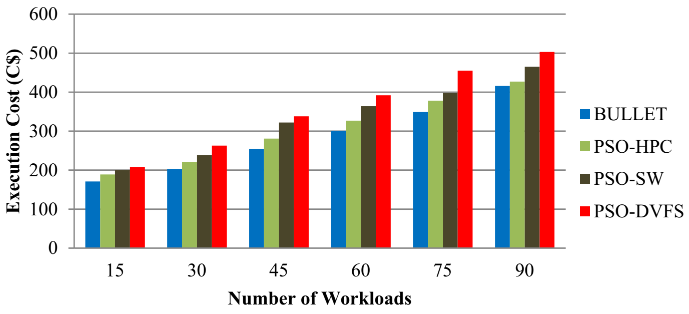

Fig. 8 Effect of execution cost with change in number of workloads

QoS requirements of workload. The minimum cost used in BULLET is 171 C$ at 15 workloads and maximum is 416 C$ at 90 workloads. The average value of execution cost in BULLET is 3.16, 4.72 and 9.16% lesser than PSO-HPC, PSO-SW and PSO-DVFS respectively.

Test Case 3: Execution Time Versus Number of Workloads As shown in Fig. 9, the execution time increases with increase in number of workloads. At 30 workloads, execution time in BULLET is 6.69% lesser than PSO-HPC, 7.12% lesser than PSO-SW and 7.59% lesser than PSO-DVFS. At 90 workloads, execution time in BULLET is 8.72% lesser than PSO-HPC, 11.39% lesser than PSO-SW and 14.79% lesser than PSO-DVFS. Figure 9 show that execution time varies in same ratio but BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS.

The number of workloads considered for Test Case 4, Test Case 5 and Test Case 6 is 90.

Test Case 4: Energy Consumption Versus Number of Resources By increasing the number of resources, the value of energy consumption increases. The minimum value of energy consumption is 22 kWh at 6 resources for BULLET. BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS in terms of energy consumption at different number of resources as shown in Fig. 10. The average

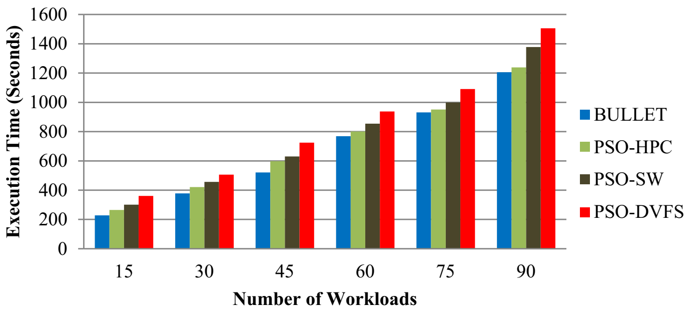

Fig. 9 Effect of execution time with change in number of workloads

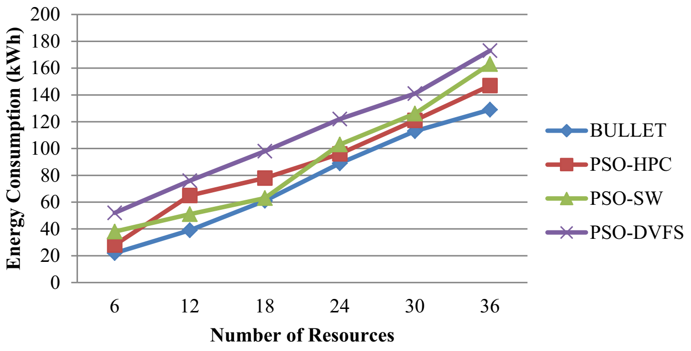

Fig. 10 Effect of change in number of resources submitted on energy consumption

value of energy consumption in BULLET is 4.41, 8.54 and 12.36% lesser than PSO-HPC, PSO-SW and PSO-DVFS respectively.

Test Case 5: Execution Cost Versus Number of Resources With the increase in number of resources, execution cost rises as shown in Fig. 11. As per the number of resources increases, BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS. The cause is that BULLET adjusts the resources at runtime according to the QoS requirements of workload. The minimum cost is used in BULLET is 45 C$ at 6 resources and maximum is 137 C$ at 36 resources. The average value of execution cost in BULLET is 5.41, 6.26 and 8.78% lesser than PSO-HPC, PSO-SW and PSO-DVFS respectively.

Test Case 6: Execution Time Versus Number of Resources As shown in Fig. 12, the execution time decreases with increase in number of resources. At 18 resources, execution time in BULLET is 7.42% lesser than PSO-HPC, 8.91% lesser than PSO-SW and 12.52% lesser than PSO-DVFS. At 36 resources, execution time in BULLET is 3.49% lesser than PSO-HPC, 4.64% lesser than PSO-SW and 7.93% lesser than PSO-DVFS. Figure 12 show that execution time varies in same ratio but BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS.

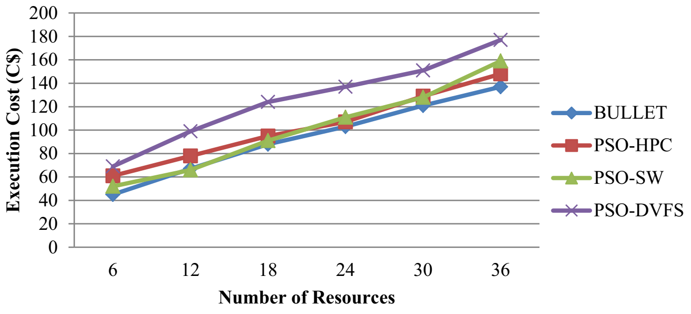

Fig. 11 Effect of execution cost with change in number of resources

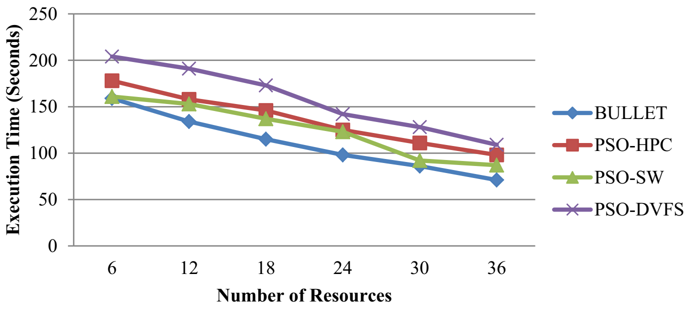

Fig. 12 Effect of execution time with change in number of resources

Test Case 7: Availability We have calculated the percentage of availability for BULLET and existing scheduling algorithms (PSO-HPC, PSO-SW and PSO-DVFS) with different number of cloud workloads. With increasing the number of cloud workloads, the percentage of availability is decreasing. The percentage of availability in BULLET is more as compared to PSO-HPC, PSO-SW and PSO-DVFS at different number of cloud workloads as shown in Fig. 13. The maximum percentage of availability is 88.7% at minimum number of cloud workloads. At 75 workloads, percentage of availability in BULLET is 7.42% more than PSO-HPC, 9.91% more than PSO-SW and 13.72% more than PSO-DVFS.

Test Case 8: Reliability We have calculated the percentage of reliability for BULLET and existing scheduling algorithms (PSO-HPC, PSO-SW and PSO-DVFS) with different number of cloud workloads. By increasing the number of cloud workloads, the percentage of reliability is decreasing. The percentage of reliability in BULLET is more as compared to PSO-HPC, PSO-SW and PSO-DVFS at different number of cloud workloads as shown in Fig. 14. The maximum percentage of reliability is 94.65 at 15 cloud workloads. At 60 workloads, percentage of reliability in BULLET is 4.91% more than PSO-HPC, 8.76% more than PSO-SW and 17.22% more than PSO-DVFS.

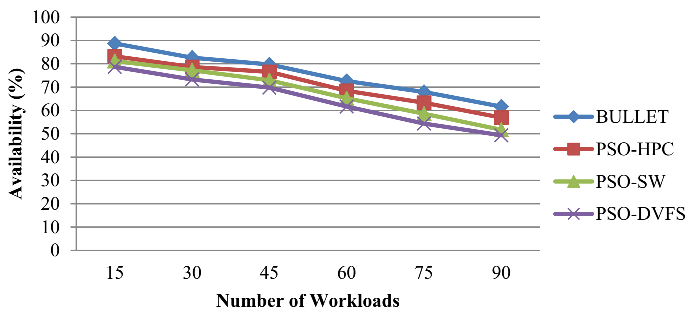

Fig. 13 Effect of change in number of workloads submitted on availability

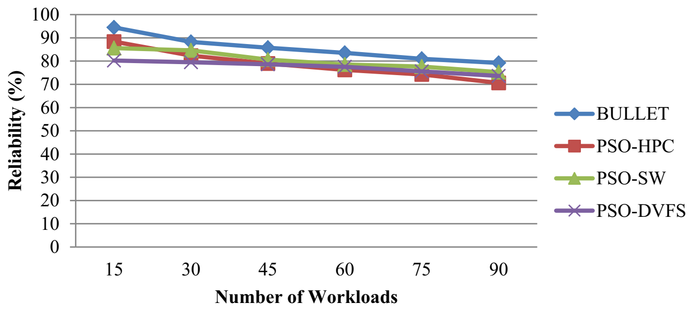

Fig. 14 Effect of change in number of workloads submitted on reliability

Test Case 9: Resource Utilization With increasing number of cloud workloads, the percentage of resource utilization is increasing. The percentage of resource utilization in BULLET is more as compared to PSO-HPC, PSO-SW and PSO-DVFS at different number of cloud workloads as shown in Fig. 15. The maximum percentage of resource utilization is 88.4% at 90 cloud workloads and minimum percentage is 71.96 at 15 workloads in BULLET but BULLET performs better than PSO-HPC, PSO-SW and PSO-DVFS for any number of workloads.

Test Case 10: Latency With increasing number of cloud workloads, the value of latency is increasing. The value of latency in BULLET is lesser as compared to PSO-HPC, PSO-SW and PSO-DVFS at different number of cloud workloads as shown in Fig. 16. The minimum value of latency is 1.24 s at 15 cloud workloads and maximum is 9.13 s at 90 cloud workloads in BULLET. At 15 workloads, latency in BULLET is 2.23% lesser than PSO-HPC, 1.91% lesser than PSO-SW and 5.66% lesser than PSO-DVFS but at 90 workloads, latency in BULLET is 6.11% lesser than PSO-HPC, 14.92% lesser than PSO-SW and 17.59% lesser than PSO-DVFS.

Test Case 11: Convergence of PSO Figure 17 plots the convergence of total cost computed by PSO over the number of iterations for different value of Resource Utilization (RU): 85, 75 and 65% by executing different number of workloads.

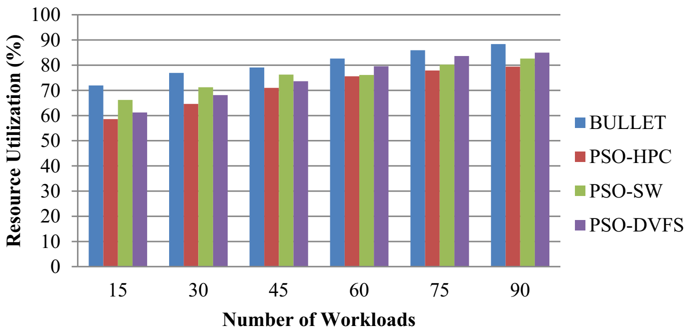

Fig. 15 Effect of change in number of workloads submitted on resource utilization

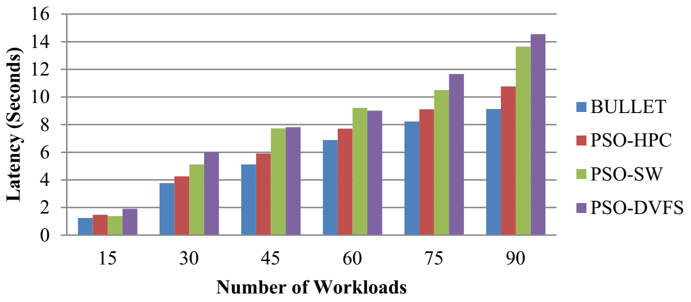

Fig. 16 Effect of change in number of workloads submitted on latency

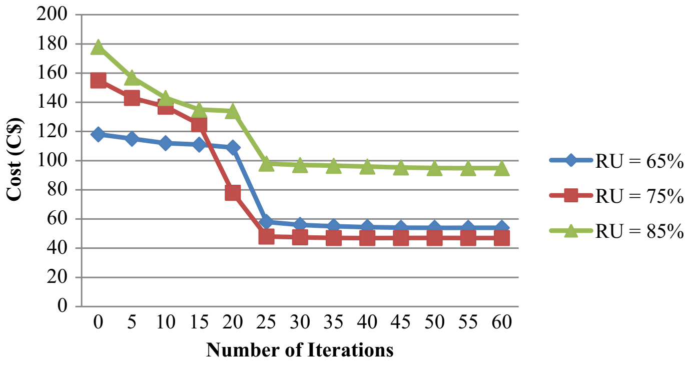

Fig. 17 Convergence curve of total cost

Initially the workloads are randomly initialized. Therefore, the total initial cost is very high at 0th iteration. As the algorithm progresses, the convergence is drastic and achieves global minima very quickly. The number of iterations required for the convergence is seen to be 50–60, for our cloud environment.

Table 5 describes the comparison of execution cost, execution time and energy consumption used to process same number of workloads (50 workloads of same type) on cloud environment for PSO-HPC [19], PSO-SW [20] and PSO-DVFS [21]. In this experiment, we have considered three different cloud infrastructures with different processor configurations (2 core processor, 4 core processor, 8 core processor and 16 core processor) to measure the variation of execution cost, execution time and energy consumption.

Figures 18, 19, 20, 21, 22, 23 and 24 describes the comparison of QoS parameters (execution cost, execution time, energy consumption, availability, resource utilization, latency and reliability) used to process different number of workloads (45 and 90) on cloud environment for BULLET with different number of Virtual Machines (VMs). The number of VMs used to execute the workloads was incremented gradually showing how the QoS parameters are optimized when more VMs were added to the cloud. As shown in Figs. 18, 19, 20, 21, 22, 23 and 24, with one virtual node running on Server R1, execution of 45 workloads finished in 636.12 s. With 12 virtual nodes (6 running on R1, 4 running on R2 and 2 running on R3), the application took 476.16 s. We note that the execution time is reduced by adding additional virtual nodes.

The value of reliability, availability, resource utilization, latency, execution time, execution cost and energy consumption has been calculated for 45 and 90 cloud workloads with different number virtual machines (VM nodes). By increasing the number of VMs, the percentage of reliability is increasing. The percentage of reliability with 45 workloads is more as compared to 90 Cloud workloads as shown in Fig. 18. The maximum percentage of reliability is 94.65 at 12 VMs.

By increasing the number of VMs, the percentage of availability is increasing as shown in Fig. 19. The percentage of availability with 90 workloads is lesser as compared to 45 cloud workloads. The maximum percentage of reliability is 93.46 for 45 workloads and 92.44 for 90 workloads at 12 VMs.

As shown in Fig. 20, the percentage of resource utilization with 90 workloads is more as compared to 45 workloads. The maximum percentage of resource utilization is 89.68 at 90 workloads and minimum percentage is 88.41 at 45 workloads with 12 VMs.

As shown in Fig. 21, the execution time decreases with increase in number of VMs. At 45 workloads, execution time is lesser than 90 workloads. Figure 21 shows that the execution time reduces rapidly in 90 workloads as compared to 45 workloads.

With the increase in number of VMs, execution cost rises as shown in Fig. 22. The minimum cost used is 47.58C$ at 45 workloads and 322.31 C$ at 90 workloads with 1 VM. The average value of execution cost at 45 workloads is 22.16% lesser than 90 workloads.

With increasing the number of VMs, the value of latency is decreasing as shown in Fig. 23. Initially, the value of latency is more for 90 workloads with 1 VM. At 12 VMs, maximum resources are utilized and value of latency for both 45 and 90 workloads is approximately same.

By increasing the number of VMs, the value of energy consumption increases as shown in Fig. 24. The minimum value of energy consumption is 20.25 kWh for 45

Table 5 Summary of experimental statistics on real cloud environment

Configuration	QoS parameter											
	Execution cost (C$)				Energy consumption (kWh)				Execution time (s)			
	PSO-HPC	PSO-SW	PSO-DVFS	BULLET	PSO-HPC	PSO-SW	PSO-DVFS	BULLET	PSO-HPC	PSO-SW	PSO-DVFS	BULLET
2 Core Processor	102	78	49	37	145	118	77	55	321	356	301	229
4 Core Processor	310	289	155	83	178	152	141	125	669	587	671	491
8 Core Processor	451	371	299	151	415	372	237	195	1391	1423	1293	908
16 Core Processor	522	439	391	203	564	456	389	225	1820	2109	2201	1423

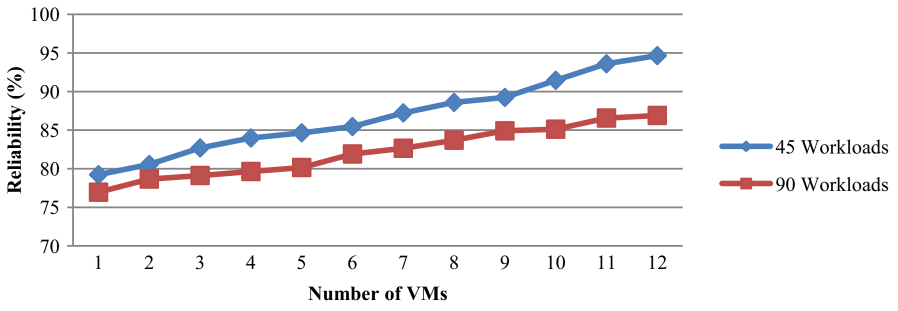

Fig. 18 Effect of change in number of VMs on reliability

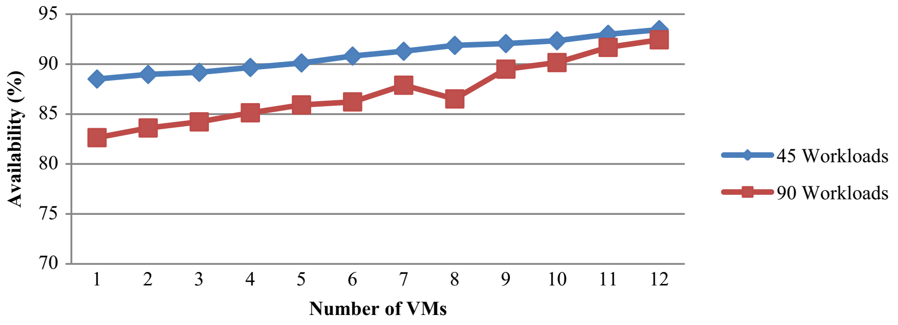

Fig. 19 Effect of change in number of VMs on availability

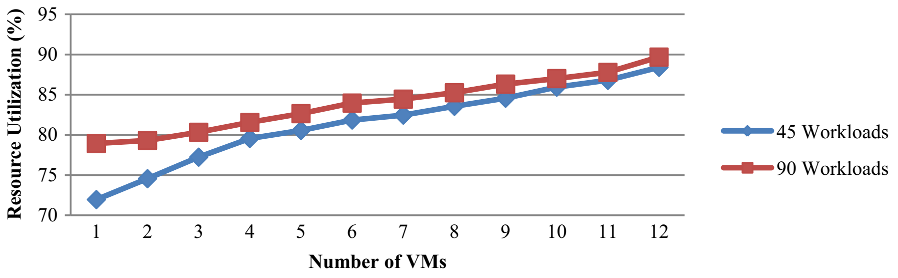

Fig. 20 Effect of change in number of VMs on resource utilization

workloads at 1 VM. The average value of energy consumption with 45 workloads is 22.36% lesser 90 workloads.

4.3 Statistical Analysis

Statistical significance of the results has been analyzed by Coefficient of Variation (c_v), a statistical method. c_v is statistical measure of the distribution of data about the mean value. c_v is used to compare to different means and furthermore offer an overall analysis of performance of the technique used for creating the statistics. It

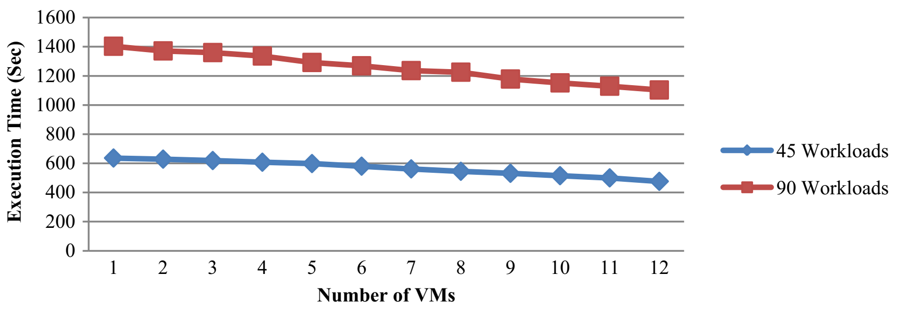

Fig. 21 Effect of change in number of VMs on execution time

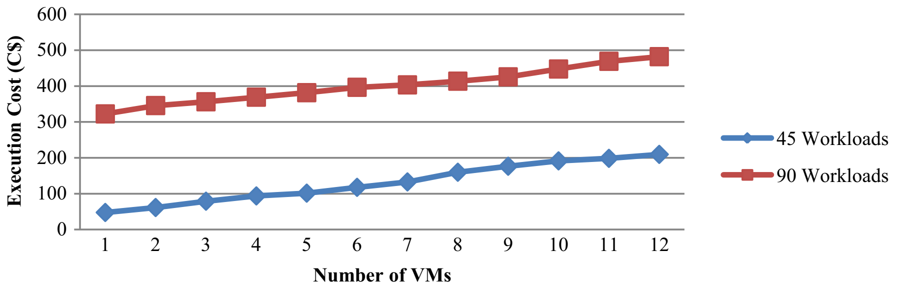

Fig. 22 Effect of change in number of VMs on execution cost

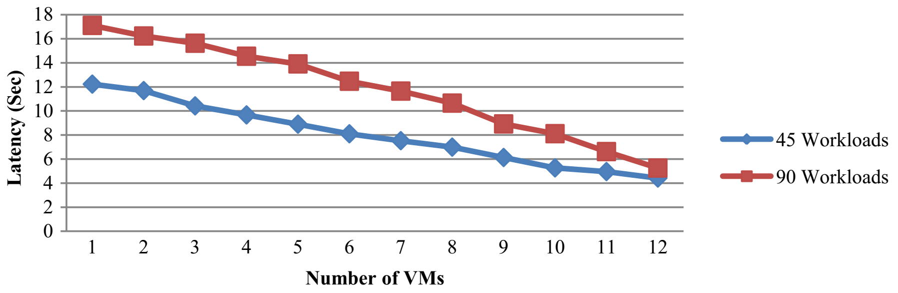

Fig. 23 Effect of change in number of VMs on latency

states the deviation of the data as a proportion of its average value, and is calculated as follows (Eq. 19):

$$c_v = \frac{SD}{\mathrm{M}} \times 100 \tag{19}$$

where *SD* is a standard deviation and *M* is mean. c_v of execution time, cost and energy consumption has been studied of Cloud workload of proposed technique (BULLET) and existing algorithms (PSO-HPC, PSO-SW and PSO-DVFS) as shown in Figs. 25, 26 and 27.

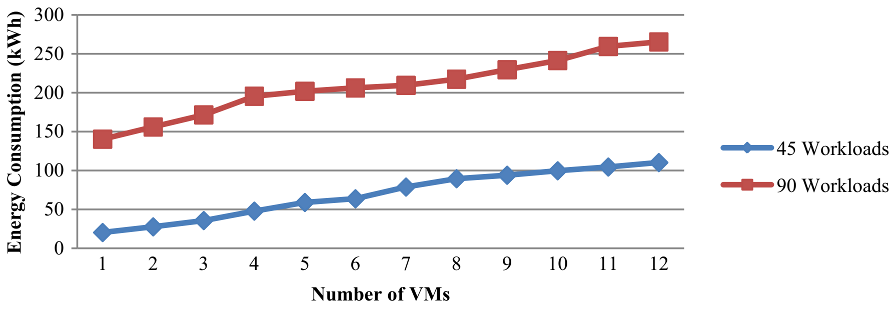

Fig. 24 Effect of change in number of VMs on energy consumption

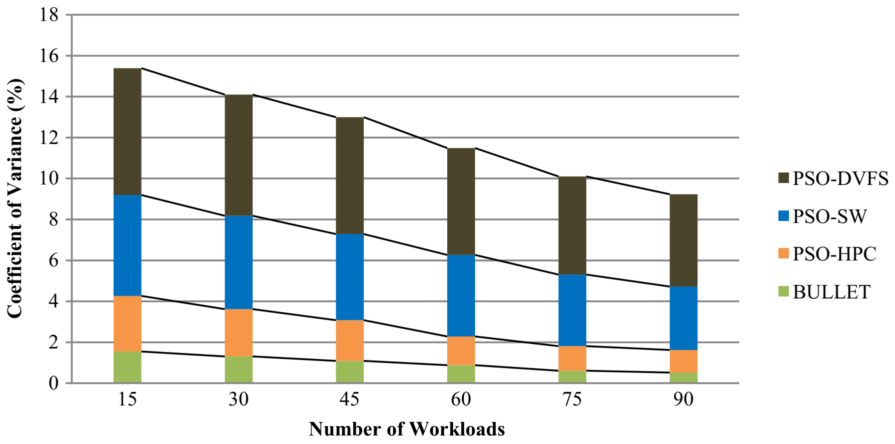

Fig. 25 c_v for execution time with each scheduling algorithm

c_v calculated for execution time and cost results attained by proposed algorithm and existing algorithms. Range of c_v (0.25–1.69%) for execution time (0.37–1.96%) for cost and (0.61–2.47%) for energy consumption approves the stability of BULLET as shown in Figs. 25, 26 and 27. Small value of c_v signifies BULLET is more efficient in resource scheduling in the situations where the number of cloud workloads has changed. Value of c_v decreases as the number of workloads is increasing. Statistical analysis demonstrates the BULLET outperforms existing scheduling algorithms for large numbers of cloud workloads. With small value of c_v system is more stable and BULLET attained the best results in the cloud for cost and execution time as QoS parameters.

The statistical analysis of QoS parameters (Figs. 18, 19, 20, 21, 22, 23, 24) is described in Table 6. The number of samples considered is 12. The value of confidence interval is calculated using IBM SPSS 24. Table 6 lists the 95% Confidence Intervals and the estimations of the medians of the differences of the values of QoS parameters of the servers with 45 and 90 workloads. From the Table 6, we can see that the estimated error (the pseudo-median of the differences)

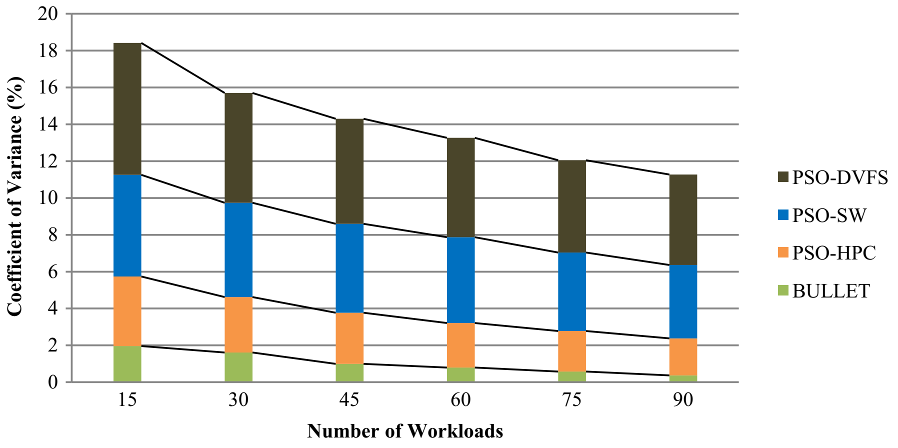

Fig. 26 c_v for execution cost with each scheduling algorithm

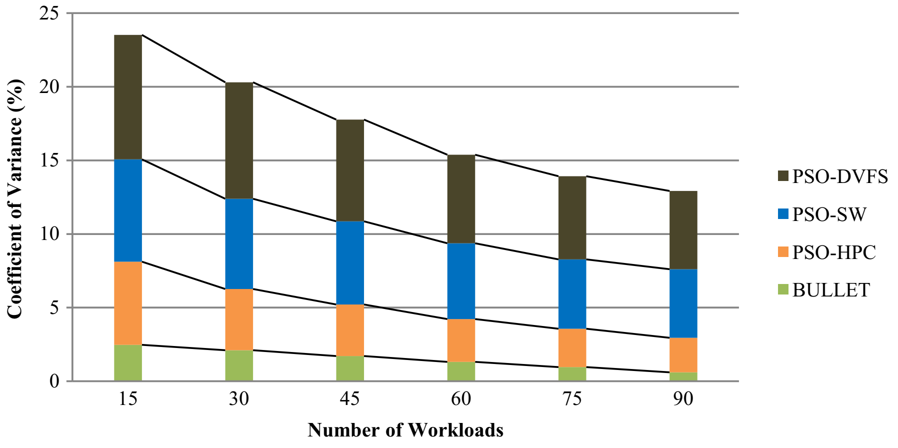

Fig. 27 c_v for energy consumption with each scheduling algorithm

is less than 3% for all characteristics. This is representative of the fact, the mean value of all the QoS parameters with 95% confidence interval from its lower value to upper value.

The variation of mean value of execution cost with different number of workloads and number of VMs is shown in Fig. 28. With increasing the number of VMs, the execution cost increases. It is clearly shown that the execution cost for 15 workloads is 31.82% lesser than 90 workloads with 2 VMs. For 60 workloads, execution cost with 12 VMs is 22.61% more than 2 VMs.

Figure 29 shows the variation of mean value of latency with different number of workloads and number of virtual nodes. It clearly shows the latency for 15 workloads is 42.52% lesser than 90 workloads with 2 VMs. With increasing the

Table 6 Statistical analysis of QoS parameters

QoS parameter	Number of workloads	SD	Standard error mean	95% confidence interval of the difference		Mean value
				Lower	Upper	
Execution time (s)	45	53.34805	15.40026	532.6884	600.4799	566.5842
	90	100.56821	29.03154	1190.6220	1318.4180	1254.52
Execution cost (C$)	45	55.75	16.09364	958.2748	166.1185	130.6967
	90	49.34614	14.24500	369.6428	432.3489	400.9958
Latency (s)	45	2.61545	0.75502	6.3633	9.6869	8.0251
	90	3.90714	1.127889	9.2759	14.2408	11.7583
Availability (%)	45	1.65032	0.47641	89.8914	91.9886	90.94
	90	3.19012	0.92091	85.1339	89.1877	87.1608
Reliability (%)	45	4.92129	1.42065	83.6493	89.9030	86.7762
	90	3.29977	0.95256	80.6493	84.2875	82.1909
Resource utilization (%)	45	4.97449	1.43601	78.2951	84.6164	81.4557
	90	3.45959	0.99870	81.7377	86.1340	83.9358
Energy consumption (kWh)	45	31.37344	9.05673	49.2559	89.1234	69.1897
	90	38.66417	11.16138	183.3145	232.4465	207.8805

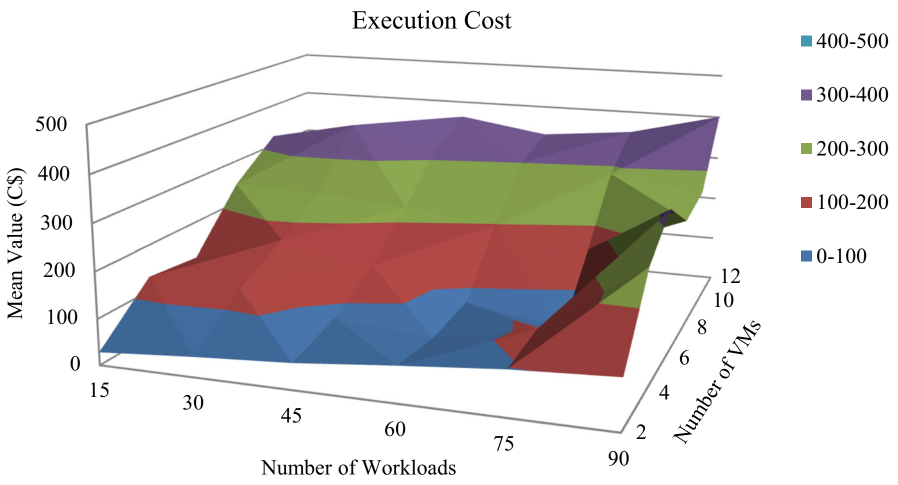

Fig. 28 Statistical analysis of execution cost

number of workloads, the latency increases but due to increasing in number of VMs, latency decreases. For 90 workloads, latency with 12 VMs is 27.92% lesser than 2 VMs.

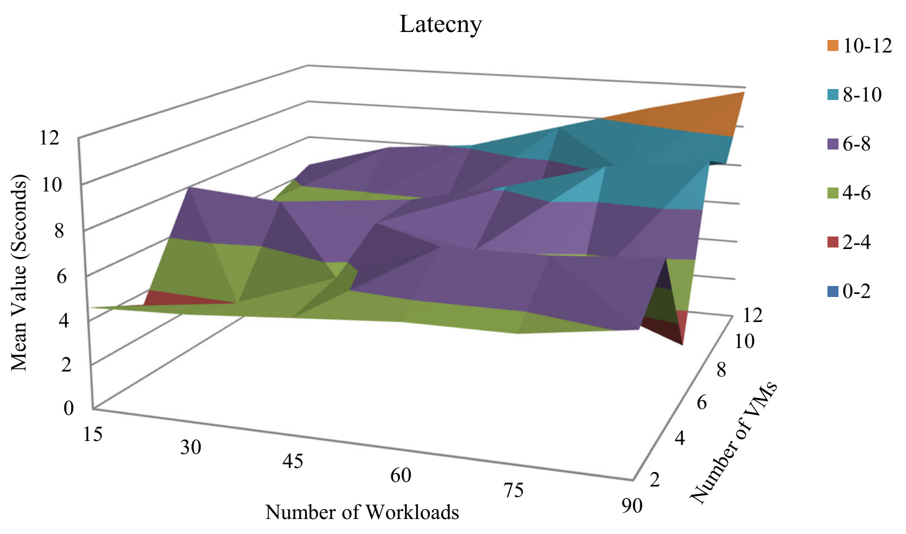

Fig. 29 Statistical analysis of latency

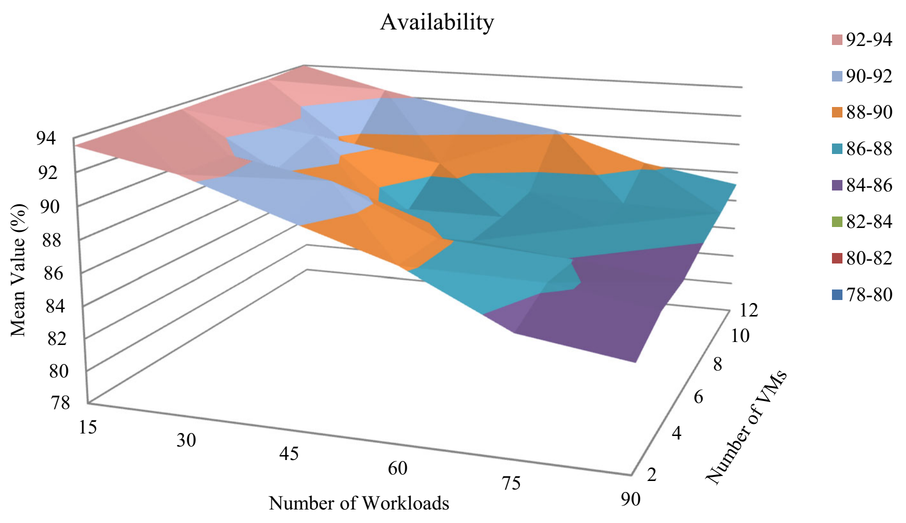

Fig. 30 Statistical analysis of availability

The variation of mean value of availability with different number of workloads and number of VMs is shown in Fig. 30. With increasing the number of VMs, the availability increases. It is clearly shown that the availability with 12 VMs is 4.25% more than 2 VMs for 15 workloads. With increasing the number of workloads, the availability decreases. The availability for 90 workloads is 7.52% lesser for 45 workloads with 12 VMs.

Figure 31 shows the variation of mean value of reliability with different number of workloads and number of virtual nodes. With increasing the number of VMs, the reliability increases. It is clearly shown that the availability with 12 VMs is 7.61% more than 2 VMs for 30 workloads. With increasing the number of workloads, the reliability decreases. The reliability for 75 workloads is 6.96% lesser for 15 workloads with 12 VMs.

The variation of mean value of resource utilization with different number of workloads and number of VMs is shown in Fig. 32. With increasing the number of

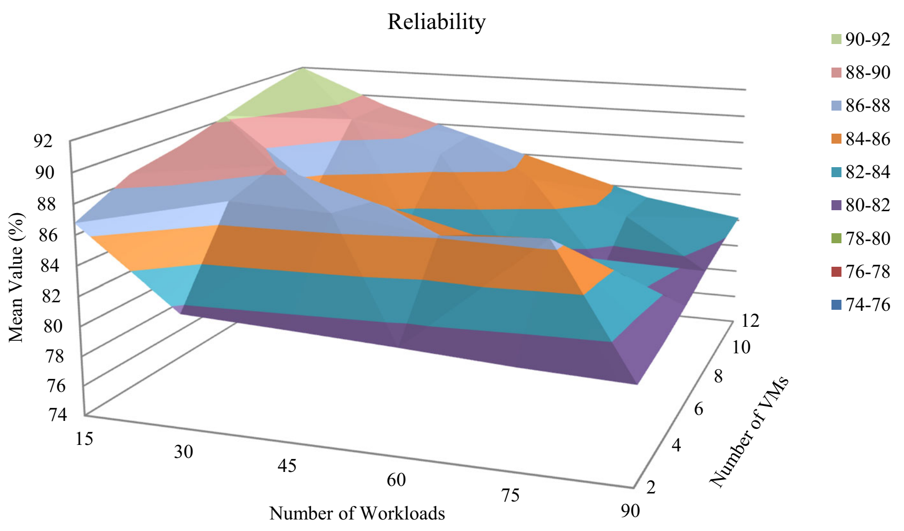

Fig. 31 Statistical analysis of reliability

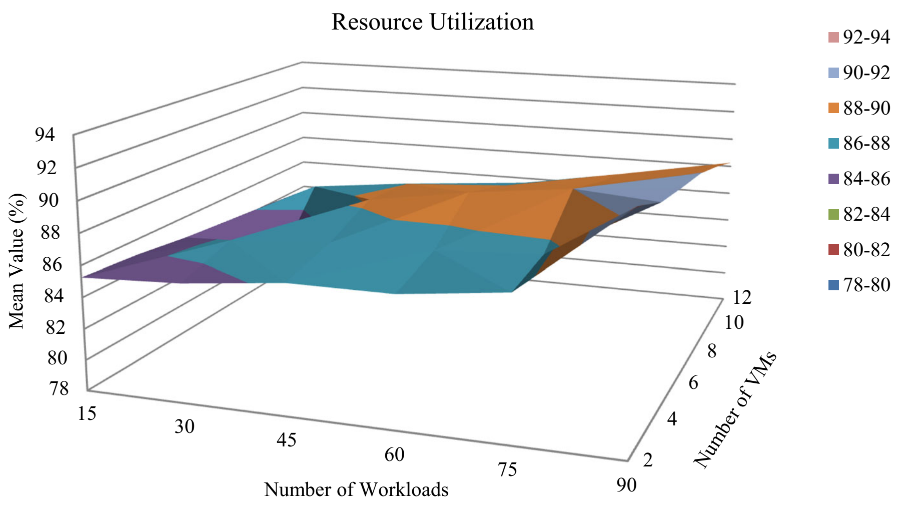

Fig. 32 Statistical analysis of resource utilization

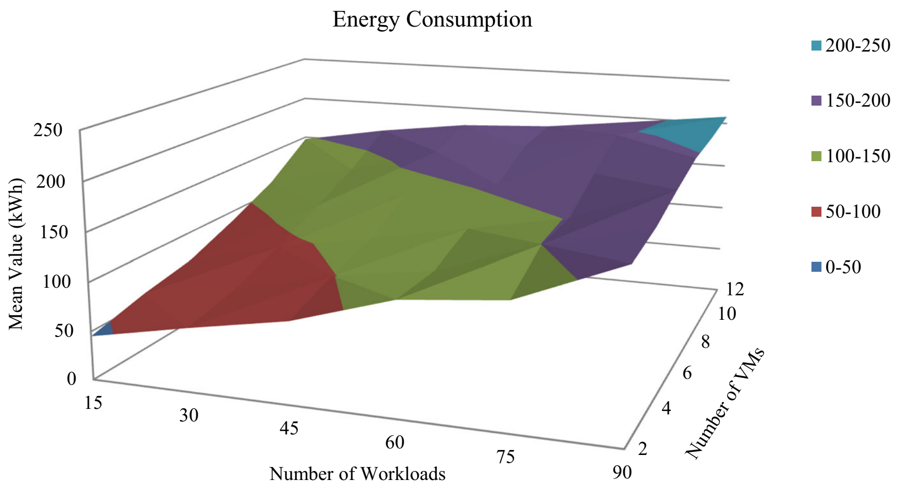

Fig. 33 Statistical analysis of energy consumption

VMs, resource utilization decreases. It is clearly shown that the resource utilization for 30 workloads is 9.42% more than 75 workloads with 6 VMs. With increasing the number of workloads, resource utilization increases. For 60 workloads, resource utilization with 12 VMs is 4.61% lesser than 2 VMs.

Figure 33 shows the variation of mean value of energy consumption with different number of workloads and number of virtual nodes. It is clearly shown that the energy consumption increases with increasing the number of VMs and number of workloads. Energy consumption for 15 workloads is 14.75% lesser than 90 workloads with 2 VMs. With 12 VMs, energy consumption for 90 workloads is 17.56% more than 15 workloads.

Figure 34 shows the variation of mean value of execution time with different number of workloads and number of virtual nodes (VMs). It clearly shows the execution time for 15 workloads is 34.152% lesser than 90 workloads with 2 VMs. With increasing the number of VMs, the execution cost decreases. For 90 workloads, execution time with 12 VMs is 25.44% lesser than 2 VMs.

4.4 Discussions

The performance of proposed PSO based resource scheduling technique (BULLET) has been compared with the existing scheduling algorithms (PSO-HPC, PSO-SW and PSO-DVFS). The performance of BULLET has been analyzed with different number of cloud workloads and resources. The performance of BULLET has been evaluated with respect to execution time, cost, energy and other QoS parameters like availability, reliability, latency and resource utilization. Execution cost permits the evaluation for selection of resources whereas duration of workload execution evaluates by execution time. The workload execution using the BULLET performs better as shown by all the experimental results. The overall cost for cloud consumer's workload execution is less. With the increase in budget, the more

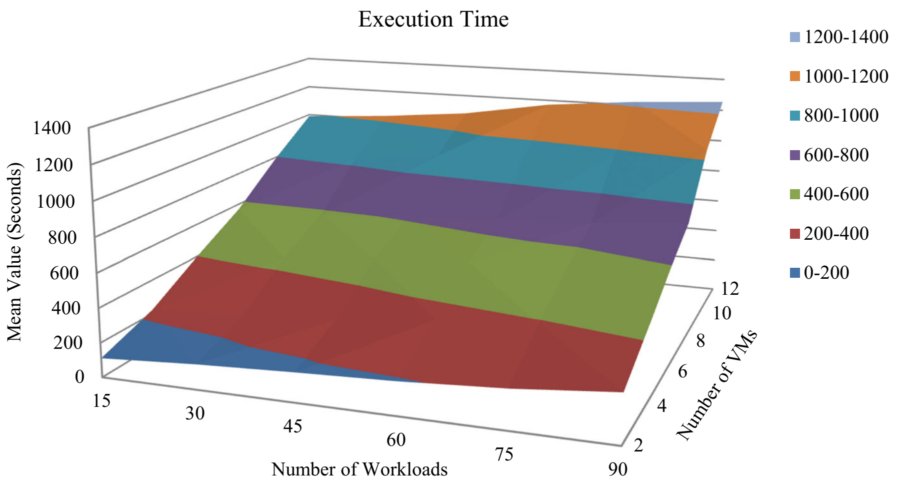

Fig. 34 Statistical analysis of execution time

number of resources are provided to reduce the execution time. BULLET executes the same number of cloud workloads at maximum availability and reliability. The average value of energy consumption in BULLET is 7.61, 11.45 and 17.19% lesser than PSO-HPC, PSO-SW and PSO-DVFS respectively. The minimum cost used in BULLET is 171 C$ at 15 workloads and maximum is 416 C$ at 90 workloads. The average value of execution cost in BULLET is 3.16, 4.72 and 9.16% lesser than PSO-HPC, PSO-SW and PSO-DVFS respectively.

At 30 workloads, execution time in BULLET is 6.69% lesser than PSO-HPC, 7.12% lesser than PSO-SW and 7.59% lesser than PSO-DVFS. At 90 workloads, execution time in BULLET is 8.72% lesser than PSO-HPC, 11.39% lesser than PSO-SW and 14.79% lesser than PSO-DVFS. The maximum percentage of availability is 88.7% at minimum number of cloud workloads. At 75 workloads, percentage of availability in BULLET is 7.42% more than PSO-HPC, 9.91% more than PSO-SW and 13.72% more than PSO-DVFS. The maximum percentage of reliability is 19.2 at 15 cloud workloads. At 60 workloads, percentage of reliability in BULLET is 4.91% more than PSO-HPC, 8.76% more than PSO-SW and 17.22% more than PSO-DVFS. The maximum percentage of resource utilization is 88.4% at 90 cloud workloads and minimum percentage is 71.96 at 15 workloads in BULLET. The minimum value of latency is 1.24 s at 15 cloud workloads and maximum is 9.13 at 90 cloud workloads in BULLET. At 15 workloads, latency in BULLET is 2.23% lesser than PSO-HPC, 1.91% lesser than PSO-SW and 5.66% lesser than PSO-DVFS but at 90 workloads, latency in BULLET is 6.11% lesser than PSO-HPC, 14.92% lesser than PSO-SW and 17.59% lesser than PSO-DVFS. Considering all these QoS parameters (execution time, cost, energy, availability, reliability, latency and resource utilization) and experimental result outcomes, it is shown that BULLET delivers a superior solution for heterogeneous cloud workloads and approximate optimum solution for challenges of resource scheduling.

5 Conclusions and Future Scope

In this paper, we have proposed PSO based resource scheduling technique called BULLET for scheduling of workloads in cloud environment so as to minimize the execution cost, time and energy. Experimental results demonstrate that BULLET is effective in decreasing the execution time, cost and energy consumption of cloud workloads along with other QoS parameters like availability, reliability, latency and resource utilization. Proposed scheduling technique provides effective outcomes as compared to existing PSO based scheduling algorithms at different levels of cost, time and energy as shown in test cases. Thus, resources can be scheduled easily and workloads can be executed effectively through proposed scheduling algorithm and this will further reduce queuing time which leads to effective resource execution. Proposed scheduling technique maps and executes the workloads based on workload details given by user and resource details given by providers. In future, we will further develop an autonomic resource management technique that efficiently schedules the provisioned cloud resources and maintains the SLA based on user's QoS requirements to reduce the above mentioned dependency. IaaS providers can use these results to quickly assess possible reductions in execution time and execution cost, hence having the potential to save energy. This framework can also be extended by identifying relationship between workload (patterns) and the resource demands (demands for compute, storage, and network resources) in the cloud.

Acknowledgements One of the authors, Dr. Sukhpal Singh Gill (Post Doctorate Fellow), gratefully acknowledges the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Australia, for awarding him the Fellowship to carry out this research work. This research work is partially supported by Melbourne-Chindia Cloud Computing Research Network funded by the University of Melbourne.

References

1. Moens, H., Truyen, E., Walraven, S., Joosen, W., Dhoedt, B., De Turck, F.: Cost-effective feature placement of customizable multi-tenant applications in the cloud. J. Netw. Syst. Manag. **22**(4), 517–558 (2014)
2. Singh, S., Chana, I.: QoS-aware autonomic resource management in cloud computing: a systematic review. ACM Comput. Surv. **48**(3), 1–46 (2015)
3. Singh, S., Chana, I., Singh, M.: The journey of QoS based autonomic cloud computing. IT Prof. Mag. **19**(2), 42–49 (2017)
4. Singh, S., Chana, I.: Resource provisioning and scheduling in clouds: QoS perspective. J. Supercomput. **72**(3), 926–960 (2016)
5. Chiang, M.L.: Efficient diagnosis protocol to enhance the reliability of a cloud computing environment. J. Netw. Syst. Manag. **20**(4), 579–600 (2012)
6. Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Soft: Pract. Exper. **41**(1), 23–50 (2011)
7. Singh, S., Chana, I.: Q-aware: quality of service based cloud resource provisioning. Comput. Electr. Eng. **47**, 138–160 (2015)
8. Singh, S., Chana, I.: QRSF: QoS-aware resource scheduling framework in cloud computing. J. Supercomput. **71**(1), 241–292 (2015)

9. Singh, S., Chana, I., Buyya, R.: STAR: SLA-aware autonomic management of cloud resources. IEEE Trans.Cloud Comput. (2017). doi:10.1109/TCC.2017.2648788
10. Daniel, L.A., Madeira, E., Medhi, D.: On makespan, migrations, and QoS workloads' execution times in high speed data centers. IEICE Trans. Commun. **98**(11), 2099–2110 (2015)
11. Lago, D., Madeira, E., Medhi, D.: High speed network impacts and power consumption estimation for cloud data centers. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 615–620. ACM (2015)
12. Varalakshmi, P., Ramaswamy, A., Balasubramanian, A., Vijaykumar, P.: An optimal workflow based scheduling and resource allocation in cloud. In: Advances in Computing and Communications, pp. 411–420. Springer, Berlin (2011)
13. Xing, L.-N., Chen, Y.-W., Wang, Pe, Zhao, Q.-S., Xiong, J.: A knowledge-based ant colony optimization for flexible job shop scheduling problems. Appl. Soft Comput. **10**(3), 888–896 (2010)
14. Topcuoglu, H., Hariri, S., Wu, M.-Y.: Task scheduling algorithms for heterogeneous processors. In: Heterogeneous Computing Workshop (HCW'99). San Juan (1999)
15. El-kenawy, E.-S.T., El-Desoky, A.I., Al-rahamawy, M.F.: Extended max-min scheduling using petri net and load balancing. Int. J. Soft Comput. Eng. (IJSCE) **2**(4), 198–203 (2012)
16. Liu, K., Jin, H., Chen, J., Liu, X., Yuan, D., Yang, Y.: A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. Int. J. High Perform. Comput. Appl. **24**(4), 445–456 (2010)
17. Verma, A., Kaushal, S.: Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud. In: IJCA Proceedings on International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT 2012) (2012)
18. Pandey, S., Wu, L., Guru, S., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Advanced Information Networking and Applications (AINA), 24th IEEE International Conference, Perth (2010)
19. Somasundaram, T.S., Govindarajan, K.: CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud. Future Generation Computer Systems **34**, 47–65 (2014)
20. Netjinda, N., Sirinaovakul, B., Achalakul, T.: Cost optimal scheduling in IaaS for dependent workload with particle swarm optimization. J. Supercomput. **68**(3), 1579–1603 (2014)
21. Yassa, S., Chelouah, R., Kadima, H., Granado, B.: Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. Sci. World J. (2013). doi:10.1155/2013/350934
22. Singh, S., Chana, I., Singh, M., Buyya, R.: SOCCER: self-optimization of energy-efficient cloud resources. Cluster Comput. **19**(4), 1787–1800 (2016)
23. Chen, G., Yu, J.: Particle swarm optimization algorithm. Inf. Control-Shenyang **34**(3), 318 (2005)

Sukhpal Singh Gill is faculty at Computer Science and Engineering Department, Thapar University, Patiala, India. Dr. Gill obtained Master Degree (Gold Medalist) and Doctoral Degree (DST Inspire Fellow) from Thapar University. Presently, Dr. Gill is working as a Post Doctorate Fellow at Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia. His research interests include Software Engineering, Cloud Computing, Internet of Things and Fog Computing. He has more than 40 research publications in reputed journals and conferences.

Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercialising its innovations in Cloud Computing. He has authored over 500 publications and four text books. He is one of the highly cited authors in computer science and software engineering worldwide (h-index 113+, 61,000+ citations). He has served as the founding Editor-in-Chief (EiC) of IEEE Transactions on Cloud Computing and now serving as Co-EiC of Journal of Software: Practice and Experience.

Inderveer Chana joined Computer Science and Engineering Department of Thapar University, Patiala, India, in 1997 as Lecturer and is presently serving as Professor in the department. She is Ph.D. in Computer Science with specialization in Grid Computing. She has more than 100 research publications in reputed Journals and Conferences.

Maninder Singh received his Bachelor's Degree from Pune University in 1994, and holds a Master's Degree, with honors in Software Engineering from Thapar Institute of Engineering and Technology, as well as a Doctoral Degree specialization in Network Security from Thapar University. Dr. Singh is currently working as Professor in Computer Science and Engineering Department at Thapar University.

Ajith Abraham received the Ph.D. degree in Computer Science from Monash University, Melbourne, Australia. He is currently the Director of Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, USA. He is an author/co-author of 900+ peer reviewed publications, h-index 68 and has over 25,000+ citations.