J Supercomput (2016) 72:926–960
DOI 10.1007/s11227-016-1626-x

# Resource provisioning and scheduling in clouds: QoS perspective

**Sukhpal Singh**[1] · **Inderveer Chana**[1]

Published online: 25 January 2016


**Abstract** Resource provisioning of appropriate resources to cloud workloads depends on the quality of service (QoS) requirements of cloud applications and is a challenging task. In cloud environment, heterogeneity, uncertainty and dispersion of resources encounter a problem of allocation of resources, which cannot be addressed with existing resource management frameworks. Resource scheduling, if done after efficient resource provisioning, will be more effective and the cloud resources would be scheduled as per the user requirements (QoS) on provisioned resources. Execution of cloud workloads should be as per QoS parameters to fully satisfy the cloud consumer. Therefore, based on QoS parameters, it is mandatory to predict and verify the resource provisioning before actual resource scheduling. In this paper, a resource provisioning and scheduling framework has been presented which caters to provisioned resource distribution and scheduling of resources. Cloud workloads have been re-clustered using *k*-means-based clustering algorithm after firstly clustering them through workload patterns to identify the QoS requirements of a workload, and then based on identified QoS requirements resources are provisioned before actual scheduling. Further, scheduling has been done based on different scheduling policies. Finally, the performance of the proposed framework has been evaluated in both real and simulated cloud environment and experimental results show that the framework provisions and schedules resource efficiently by considering energy consumption, execution cost and execution time as QoS parameters.

✉ Sukhpal Singh
ssgill@thapar.edu

Inderveer Chana
inderveer@thapar.edu

[1] Computer Science and Engineering Department, Thapar University, Patiala, Punjab 147004, India

**Keywords** Cloud computing · Resource provisioning · Resource scheduling · Cloud workloads · Quality of service · Service level agreement · Aneka cloud application platform · Cloud metrics · Workload patterns

## 1 Introduction

Cloud computing offers pay per usage model-based services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1]. The services offered to the users consist of a set of components, which may be offered by different providers. To satisfy the request of customers, services must be provided in accordance with the required level of quality of service (QoS) as described by the user in the form of Service Level Agreement (SLA) [2]. One of the major challenges in the current cloud solutions is to provide the required services according to the QoS level (minimum execution cost and execution time) expected by the user [3]. Another challenge is the consumption of a tremendous amount of energy by these cloud resources which leads to high operational costs and contributes toward carbon footprints to the environment. Cloud service providers want to confirm that sufficient amount of resources are provisioned to ensure that QoS requirements of cloud users such as deadline, execution time and budget restrictions are met. Literature reports that the complexity of management of resources in clouds is increasing day by day; so an efficient resource management technique is required. To solve this problem, there should be a focus on resource provisioning before actual resource scheduling. Further scheduling of resources distributes workloads to the provisioned resources based on scheduling decisions at runtime. To address this problem, an efficient resource provisioning and scheduling framework should be developed which provisions and schedules resources efficiently by considering energy consumption, execution cost and execution time as QoS parameters.

In our earlier work [1–6], we have identified various research issues related to QoS and SLA for management of cloud resources [1–4], and based on these issues we have developed a QoS-based resource provisioning technique (Q-aware) to map the resources to the workloads based on user requirements [5]. Further, a resource scheduling framework (QRSF) has been proposed, in which the provisioned resources have been scheduled using different resource scheduling policies [6]. Both Q-aware and QRSF have been verified using cloud-based simulated environment. In QRSF, there was direct scheduling of resources which further needs lot of work every time to schedule resources to execute workloads by fulfilling their QoS requirements (Table 2) due to the absence of QoS-based resource provisioning before actual scheduling of resources. To solve that problem, the concept of QRSF has been further extended by proposing QoS-based resource provisioning and scheduling (QRPS) framework in this research work. In the QRPS framework, QoS-based resource provisioning technique (Q-aware) is used for resource provisioning in which: (1) clustering of workloads is done through workload patterns, (2) $k$-means-based clustering algorithm is used for re-clustering of workloads after assigning weights to quality attributes of each workload and (3) QoS requirements of clustered workloads are identified and resources are provisioned by the resource provisionor based on their QoS requirements. Further, four

different resource scheduling policies (cost, time, cost–time and bargaining based) are used to schedule the provisioned resources (decision tree is used to select the appropriate policy) based on QoS requirements described by the cloud consumer through SLA. Further, the proposed framework (QRPS) is also implemented on a real cloud environment using Aneka to validate and optimize QoS parameters such as execution time, execution cost and energy consumption.

The motivation of this research work emerges from the challenge of finding the best resource workload pair according to user requirements by minimizing the execution time, execution cost and energy consumption and at the same time meeting the cloud workload deadline which improves user satisfaction. By using the proposed framework, QoS-based resource provisioning can be provisioned in an effective way before resource scheduling. Thus the queuing time and over- and underutilization of resources can be avoided or be assuaged. The paper is structured as follows: in Sect. 2, related works of resource provisioning and scheduling along with paper contribution are presented. The resource provisioning and scheduling framework is presented in Sect. 3. The experimental setup and results are presented in Sect. 4. In Sect. 5, conclusions and the future scope are presented.

## 2 Related work

Existing literature [7–12] reported that provisioning and scheduling of resources in a cloud environment is challenging due to two main reasons: (1) dynamic resources spread over geographical area and (2) on-demand scalability. Presently, provisioning of resources can be done in two ways: On-demand (providing resources quickly to urgent workloads) and long-term reservation (reserving resources for later usage). In on-demand criteria, executing too many workloads on a single resource will cause the problem of interference which leads to performance degradation and over provisioning. In long term reservation, many resources are in idle state which leads to underprovisioning. Underprovisioning and overprovisioning of resources lead to wastage of time and resources that increase execution cost and energy consumption. Existing resource provisioning and scheduling frameworks are presented briefly in this section.

### 2.1 QoS-based resource provisioning

Andres et al. [13] proposed a distributed framework to determine resource provisioning of workload on innovativeness clouds. To handle with erroneous request for resource that causes to over utilization of resources delivered by cloud workload request, their approach has revealed a design-based technique for approximating the workload execution time specified it's scheduling but the behavior of a particular workload was not identified. Christian et al. [14] described deadline-based resource provisioning technique using dynamic reallocation for execution of scientific workloads through Aneka in hybrid cloud environment. This technique decreases the makespan of workloads only without considering the execution cost of resources. Nikolas et al. [15] described a self-adaptive resource provisioning framework to find the appropriate predicting ways

for a given perspective through the use of the decision tree. This approach optimizes QoS parameters like relative error and SLA violation, but does not consider execution time and execution cost as a QoS parameter. Hamid et al. [16] proposed a dynamic behavior-based resource provisioning framework to assign an adequate number of resources to workloads. This approach reduces the execution cost of resources only without considering the execution time.

### 2.2 QoS-based resource scheduling

Varalakshmi et al. [17] 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 framework shows that a slight improvement in resource utilization is attained, but it did not consider cost and energy as QoS parameters. Wang et al. [18] presented an ant colony optimization (ACO) -based job scheduling framework, which adapts to the dynamic characteristics of cloud computing and incorporates particular benefits of ACO in NP-hard problems. This framework reduced only job completion time based on pheromone. Topcuoglu et al. [19] 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. Then workloads in the workflow are well ordered on a rank function. 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 completes the workload at the earliest time, but it did not consider cost and time as QoS parameters. Pandey et al. [20] introduced a particle swarm optimization (PSO)-based heuristic framework to schedule the applications to cloud resources that proceed with 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, but it does not consider the execution time of workloads. Nicola et al. [21] proposed the optimal minimum energy scheduler (OMES) for the dynamic online joint allocation of the task sizes, computing rates, communication rates and communication powers in virtualized networked data centers (NetDCs) that operates under hard per-job delay constraints. The referred NetDC's infrastructure is composed of multiple frequency-scalable virtual machines (VMs) that are interconnected by a bandwidth and power-limited switched local area network (LAN). Further, Nicola et al. [22] proposed an energy-efficient technique (EET) by extending OMES for adaptive networked data centers for real-time applications by considering energy efficiency as the QoS parameter. Both the techniques (EET and OMES) consider energy as a QoS parameter, but do not consider execution cost and time as QoS parameters. The proposed QoS-based resource provisioning and scheduling framework (QRSP) has been compared with existing resource provisioning and scheduling frameworks as described in Table 1.

Literature reported that none of the existing research work considers heterogeneous cloud workloads and QoS parameters (execution cost, energy consumption and execution time) simultaneously along with different types of resource scheduling policies

**Table 1** Comparison of QoS-based resource provisioning and scheduling framework with existing frameworks

| Framework | Mechanism | Workload type | QoS parameters |
|---|---|---|---|
| Distributed [13] | Provisioning | Homogenous | Execution time |
| Dynamic Reallocation [14] | Provisioning | Homogenous | Execution time |
| Self-adaptive [15] | Provisioning | Homogenous | Relative error |
| Dynamic behavior [16] | Provisioning | Homogenous | Execution cost |
| OWS [17] | Scheduling | Homogenous | Execution time |
| ACO [18] | Scheduling | Homogenous | Completion time |
| HEFT [19] | Scheduling | Homogenous | Communication time |
| PSO [20] | Scheduling | Homogenous | Communication cost |
| OMES [21] | Scheduling | Homogenous | Energy |
| EET [22] | Scheduling | Homogenous | Energy |
| QRPS (proposed) | Provisioning and scheduling | Homogenous and heterogeneous | Energy, cost and execution time |

in a single cloud framework. Due to this, the current resource management services have become complex. Consequently, when workloads are struggling for resources, the number of conflicts can increase leading to complexity. In addition, our novel QoS-based resource provisioning and scheduling framework executes the heterogeneous cloud workloads using clustering and optimizing QoS parameters such as execution cost, energy consumption and execution time.

### 2.3 Our contributions

In this research work, we have presented QoS-based resource provisioning and scheduling framework (QRPS). This work is an extension of our previous research work [5,6]. The contribution of this research paper is twofold. Firstly, QoS-based resource provisioning for cloud is presented in which heterogeneous cloud workloads based on different QoS requirements have been identified and analyzed. Further, various workload patterns have been identified and then clustering of workloads has been done based on these workload patterns. After that, re-clustering of clustered workloads has been done through $k$-means-based clustering algorithm by assigning weights to every quality attribute in each workload through QoS metrics. Secondly, scheduling of resources is done based on four resource scheduling policies (compromised cost–time-based (CCTB) scheduling policy, time-based (TB) scheduling policy, cost-based (CB) scheduling policy and bargaining-based (BB) scheduling policy). Mapping and execution of cloud workloads to the corresponding resources are done using these resource scheduling policies. Decision tree-based scheduling criteria are used to select the scheduling policy based on the consumer requirements. This framework focuses on how to map the cloud workload to reduce cost, time and energy consumption using clustering.

We have demonstrated the ability of the proposed framework by implementing it within simulation-based cloud environment using CloudSim toolkit along with its empirical evaluation. Aneka application development platform is used as a scalable cloud middleware to make interaction between SaaS and IaaS to deploy the QRPS framework on a real cloud environment. The performance of QRPS has also been tested on cloud testbed using synthetic workloads for different QoS parameters. We have then compared the experimental results of the proposed framework with existing frameworks for resource provisioning and scheduling. The main contribution of this paper is: (1) QoS-based resource provisioning and scheduling framework (QRPS) for effective management of resources is proposed, (2) the Aneka application development platform is used to deploy QRPS, (3) the performance of QRPS has been evaluated in the cloud environment by using purely open source technologies, (4) optimizing QoS parameters such as execution cost, energy consumption and execution time and (5) by improving customer satisfaction and queuing time, over and underutilization of resources can be avoided or assuaged.

## 3 Resource provisioning and scheduling framework

In this section, QoS-based resource provisioning and scheduling framework (QRPS) is presented in which the user interacts with the framework for their workload execution. The main objectives of QRPS are: (1) to understand the expectations and requirements of the cloud user, (2) workloads are analyzed and clustered based on workload patterns, (3) workloads are re-clustered based on QoS metrics using k-means-based clustering algorithm, (4) it provisions the resources and maps the resources to the workloads and (5) it schedules the resources for workload execution on appropriate resources with minimum energy consumption, execution time and execution cost. The QoS-based resource provisioning and scheduling framework, as shown in Fig. 1, consists of two subunits: (1) resource provisioning unit and (2) resource scheduling unit.

The cloud workload details are gathered through the Cloud Workload Management Portal (CWMP) from the cloud consumer. The bulk of workloads (BoWs) come for execution and are processed and stored in the workload queue. The types of workloads that have been identified during workload analysis are websites, technological computing, endeavour software, performance testing, online transaction processing, e-com, central financial services, storage and backup services, production applications, software/project development and testing, graphics oriented, critical Internet applications and mobile computing services [5,6]. In SLA Measure, the QRPS framework receives the information from the suitable SLA. After studying and confirming the various QoS constraints which the workload has required, the QRPS framework checks the availability of resources. QoS Metric data contains the information regarding QoS metrics used to calculate the weight for clustering of workloads. The different cloud workloads have different sets of QoS requirements and characteristics. All the workloads submitted to the QRPS framework are analyzed by the workload analyzer based on their QoS requirements. For QoS, the required workload patterns are identified for clustering of workloads and then the QoS metrics required to assign the weights based on the level of measurement described in QoS requirements specified

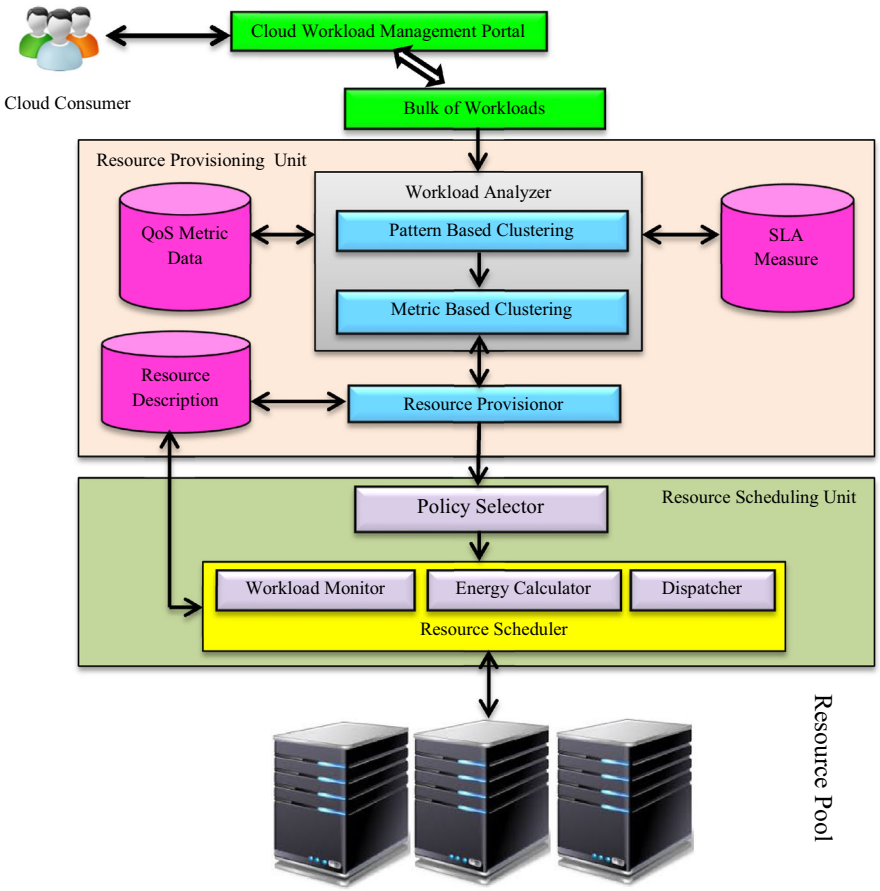


**Fig. 1** Resource provisioning and scheduling framework

in SLA are identified. $K$-means-based clustering algorithm is used for re-clustering the workloads for execution on different sets of resources. Resource description contains resource details like the number of CPUs used, size of memory, cost of resources, type of resources and number of resources. All the common resources are stored in the resource pool. Resource provisioner provides the demanded resources to the workload for their execution in a cloud environment only if the required resources are available in the resource pool. If the required resources are not available according to QoS requirement, then the QRPS framework asks to resubmit the workload with new QoS requirement in the form of SLA after re-negotiation (in which execution cost may be increased). After the provisioning of resources, the resource scheduler asks to submit the workload for resources provisioned. After this, the resource scheduler will send back the results (resource information) to the resource provisioner.

In the policy selector, decision tree is used to select the appropriate policy based on workload details described by the cloud consumer (Sect. 3.2.2). Based on the

scheduling policy, the scheduler allocates the resources to the cloud workloads by optimizing the QoS parameters. Resource scheduler is used to schedule the cloud workloads and map the cloud workloads with available provisioned resources based on the policy defined by user. Scheduler uses minimum number of resources to serve the cloud workloads within specified budget and deadline. Energy is also calculated and compared with the threshold energy value at different values of resources using (Eqs. 4–8). The workload is dispatched by the dispatcher for execution only, if the workload is executed within the described budget and deadline and actual energy consumption is less than the threshold energy value; otherwise SLA will be re-negotiated.

Resource provisioning unit (RPU) accomplishes the responsibilities in QRPS: management of (1) service-level agreement, (2) resource information and (3) workload information. RPU continually checks the status of resources provisioned, workloads queued and SLA deviation. The objective of QRPS is to provision the resources to cloud consumer without violation of SLA. The workloads submitted should be executed within their budget and deadline along with other QoS parameters as described by the cloud consumer. QRPS provisions and schedules the resources to the workloads as shown in Fig. 2. Workload submitted by the user to QRPS is stored into BoWs for their execution. All the submitted workloads are analyzed based on their QoS requirements described in terms of SLA. The workload patterns are identified for better classification of workloads and then pattern-based clustering of workloads is done. The QoS metrics for every QoS requirement of each workload are identified. Based on the importance of the attribute, weights for every cloud workload are calculated. After that, workloads are re-clustered based on $k$-means-based clustering algorithm for better execution. We calculate the value of the fitness function ($\mathrm{RPS}_{\mathrm{QoS}}$) for every workload and then compare it with the value calculated ($\mathrm{RPS}_{\mathrm{Non-QoS}}$) by the non-QoS-based RP (without considering QoS requirements). If the value of $\mathrm{RPS}_{\mathrm{QoS}}$ (QoS-based RP) is lesser than $\mathrm{RPS}_{\mathrm{Non-QoS}}$ (non-QoS-based RP), then it will provision; otherwise it analyses the workload again after resubmission of SLA by the cloud consumer through re-negotiation.

After successful provisioning of resources, the resource scheduling unit (RSU) takes the information from the appropriate workload after analyzing the various workload details which the cloud consumer demands. RSU then collects the information of available resources from Resource description (RD). RD also contains details of all the resources available in the resource pool and reserve resource pool. Based on the cloud consumer details, RSU assigns resources and executes cloud workloads. During execution of a particular cloud workload, the RSU will check the current workload. If the resources are sufficient for execution, then it will continue with execution, otherwise request for more resources. RSU checks the policy conditions and energy consumption of resources. If the scheduling policy is deadline urgency (execution within minimum time and Actual energy consumption ($E_a$) is less than the threshold energy value ($E_t$)), then QRPS will execute workloads through rescheduling otherwise, RSU asks to change the policy through re-negotiation. After successful execution of cloud workloads, RSU releases the free resources to the resource pool and the QRPS framework is ready for execution of new cloud workloads.

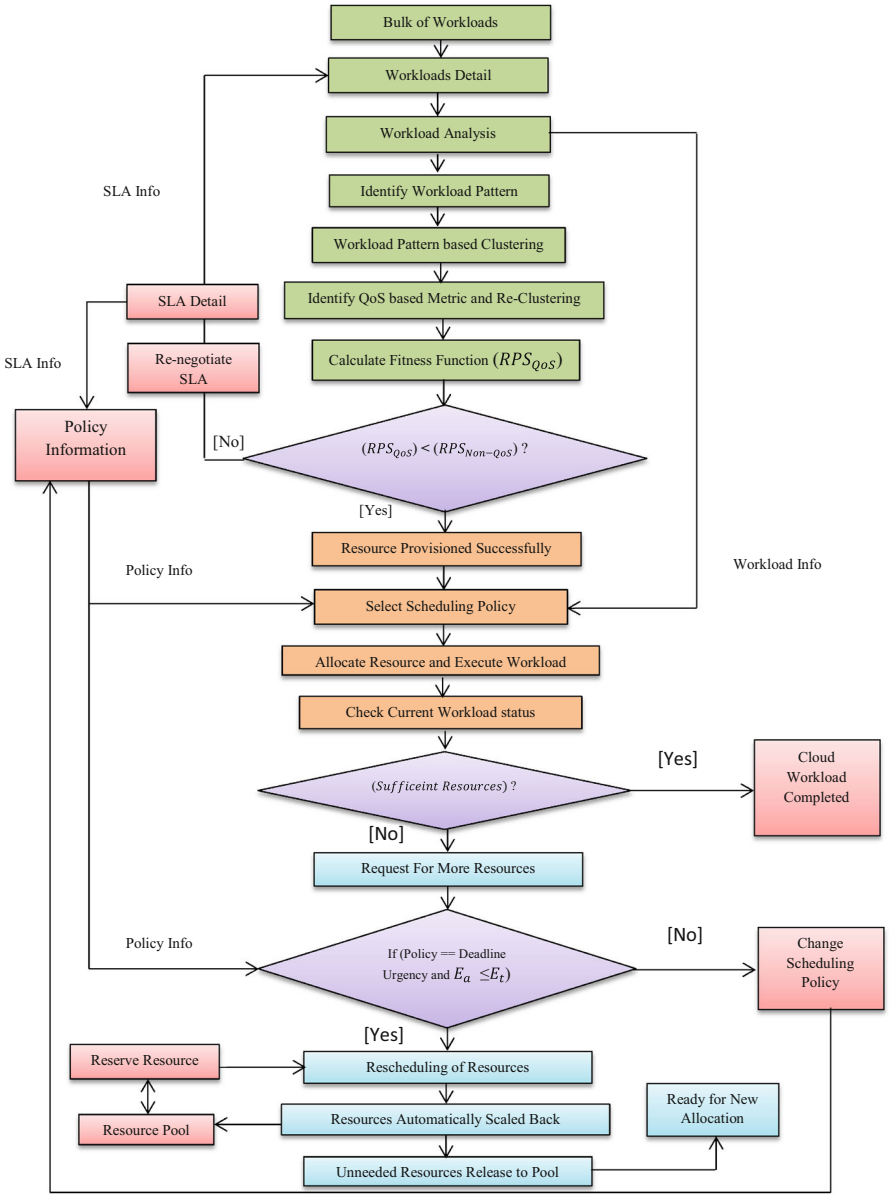


**Fig. 2** Flowchart of resource provisioning and scheduling framework

### 3.1 Resource provisioning unit

In this section, the design of RPU is presented. RPU contains the information about resources, QoS metrics and SLA to provision the resources for execution of workloads based on QoS requirements described by the cloud consumer. The QRPS framework

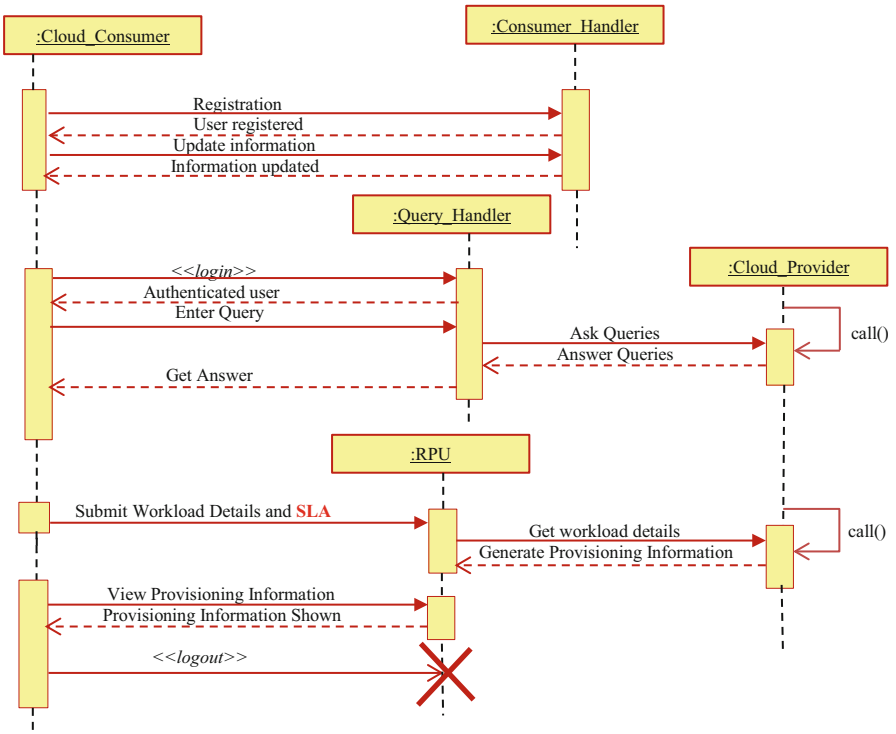


**Fig. 3** Sequence diagram of the resource provisioning

has been designed to gather the requirements from the cloud consumer and execute the workloads based on those requirements. Unified modeling language (UML) is used to study and analyze the different perspectives of the QRPS framework. Figure 3 shows the sequence model of the resource provisioning in QRPS. Firstly, successful registration of the cloud consumer has been demonstrated. After performing the task of the user's authentication and authorization, the home page of the user will be displayed. The cloud consumer writes their query regarding the cloud workloads and then submits the requirement of resources in the form of SLA. All the workloads submitted to RPU are analyzed based on their QoS requirements. The resources will be provisioned to the cloud consumer and then the cloud consumer can access the resource provisioning information.

### *3.1.1 QoS perspective of resource provisioning*

We have identified the QoS requirements for every workload and their corresponding QoS metrics has been designed [5,6] as described in Table 2. Based on QoS requirements, the required resources have been identified. Suppose a set of independent cloud workloads $\{w_1, w_2, w_3, \ldots, w_m\}$ to map on a set of resources $\{r_1, r_2, r_3, \ldots, r_n\}$ has been taken. $\mathrm{R} = \{r_k | 1 \leq \mathrm{k} \leq \mathrm{n}\}$ is the collection of resources identified and $n$ is the total number of resources. $\mathrm{W} = \{w_i | 1 \leq \mathrm{i} \leq \mathrm{m}\}$ is the collection of cloud work-

**Table 2** Cloud workloads and their QoS requirements

| 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 | 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 |

loads and $m$ is the total number of cloud workloads. In cloud computing, the provider wants to minimize the execution time and energy consumption, while the user wants to minimize the cost for cloud workload.

All the workloads submitted to the QRPS framework are analyzed by the workload analyzer based on their QoS requirements using Table 2. For QoS, the required workload patterns are identified for clustering of workloads which then identifies the QoS metrics required to assign the weights based on the level of measurement described in QoS requirements specified in SLA [5,6]. $K$-means-based clustering algorithm is used for re-clustering the workloads for execution on different sets of resources. The final set of workloads is shown in Table 3.

The goal of an objective function is to decrease the sum of product of cost and time expended for finishing all $n$ workloads. This objective function ($\mathrm{RPS}_{\mathrm{QoS}}$) successfully captures the compromise between execution cost and execution time as specified in (Eq. 1). Further, formally, the workload assignment problem with the cost and time function of each resource can be generally formulated as follows:

$$\mathrm{RPS}_{\mathrm{QoS}} = \sum_{i=1}^{n} (\mathrm{ET})_i \times (\mathrm{EC})_i, \tag{1}$$

**Table 3** 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 |

where $i$ is a current workload that is being executed, ET is execution time and EC is execution cost spent per hour. Execution time is calculated using (Eq. 2) and execution cost is calculated using (Eq. 3). The following metrics are selected from our previous work [5,6,23,24] to calculate the execution time, execution cost and energy consumption.

Execution time is a ratio of difference of workload finish time ($\mathrm{WF}_i$) and workload start time ($\mathrm{WStart}_i$) to the number of workloads. We have used the following formula to calculate the execution time (ET) (Eq. 2).

$$\mathrm{ET}_i = \sum_{i=1}^{n} \left( \frac{\mathrm{WF}_i - \mathrm{WStart}_i}{n} \right), \tag{2}$$

where $n$ is the number of workloads to be executed. Execution cost is defined as the total amount of cost spent per 1 h for the execution of workload and measured in grid dollars (G$). We have used the following formula to calculate the execution cost (EC) (Eq. 3).

$$\mathrm{EC} = \frac{\text{Total amount of cost spent}}{\text{Hour}}. \tag{3}$$

Energy consumption is calculated using (Eq. 4–8). The cloud workload will be executed only when the actual energy consumption denoted as $E_a$ is less than the threshold value of energy consumption ($E_t$). The energy model is devised on the basis that processor utilization has a linear relationship with energy ingestion. For a particular cloud workload, the information on its energy consumption and processor utilization is sufficient to measure the energy consumption for that cloud workload [22].

The overall energy consumption (EU) can be expressed in the following formula (Eq. 4):

$$\mathrm{EU} = \mathrm{EU}_{\mathrm{Datacenter}} + \mathrm{EU}_{\mathrm{Transceivers}} + \mathrm{EU}_{\mathrm{Memory}} + \mathrm{EU}_{\mathrm{Extra}}, \tag{4}$$

where $\mathrm{EU}_{\mathrm{Datacenter}}$ represents the datacenter's energy consumption and $\mathrm{EU}_{\mathrm{Transceivers}}$ represents the energy consumption of all the switching equipment. $\mathrm{EU}_{\mathrm{Memory}}$ repre-

sents the energy consumption of the storage device. $\mathrm{EU}_{\mathrm{Extra}}$ represents the energy consumption of other parts, including the fans, the current conversion loss and others. The above formula can be further disintegrated; for a cloud computing environment with $d$ datacenters, $t$ transceivers equipment and a centralized memory device, its energy consumption can be expressed as (Eq. 5):

$$\begin{aligned}\mathrm{EU} = {} & d(\mathrm{EU}_{\mathrm{Processor}} + \mathrm{EU}_{\mathrm{Primary\ storage}} + \mathrm{EU}_{\mathrm{secondary\ storage}} \\ & + \mathrm{EU}_{\mathrm{Motherboards}} + \mathrm{EU}_{\mathrm{Network\ cards}}) \\ & + \mathrm{t}\left(\mathrm{EU}_{\mathrm{Hardware}} + \mathrm{EU}_{\mathrm{LAN\ cards}} + \sum_{f=0}^{F} d_{\mathrm{connectors},f} + \mathrm{EU}_{f}\right) \\ & + \left(\mathrm{EU}_{\mathrm{Network\ analysis\ server}} + \mathrm{EU}_{\mathrm{Memory\ manager}} + \mathrm{EU}_{\mathrm{Network\ attached\ storage\ arrays}}\right) \\ & + \mathrm{EU}_{\mathrm{Extra}}. \end{aligned} \tag{5}$$

The energy consumed by a transceiver and all its ports can be defined as: $\mathrm{EU}_{\mathrm{Hardware}}$ is related to the power consumed by the transceiver, $\mathrm{EU}_{\mathrm{LANcards}}$ is the power consumed by any active network LAN card, $\mathrm{EU}_{f}$ corresponds to the power consumed by a connector (port) running at the frequency $f$. In the above Eq. (5), only the last component appears to be dependent on the link frequency, while other components, such as $\mathrm{EU}_{\mathrm{Hardware}}$ and $\mathrm{EU}_{\mathrm{LAN\ cards}}$, remain fixed for all the duration of transceiver operation. Therefore, $\mathrm{EU}_{Hardware}$ and $\mathrm{EU}_{\mathrm{LAN\ cards}}$ can be avoided by turning the transceiver off or putting it into sleep mode. For a particular cloud workload, the information on its energy consumption and processor utilization is sufficient to measure the energy consumption for that cloud workload [6]. $P_{t\prime,i}$ is the energy consumption (Eq. 6) of a cloud workload where $w_t$ is defined as:

$$P_{t',\,i}\,(r) = z \times \mathrm{EU}_{\max} + (1 - z) \times \mathrm{EU}_{\max} \times \mathrm{r}, \tag{6}$$

in which $\mathrm{EU}_{\max}$ is the maximum energy consumption, while the resource is fully utilized, $z$ is the fraction of power consumed by the idle resource and $r$ is CPU utilization. CPU utilization is change over time and is a function of time and presented as $\mathrm{r}\,(t)$. For a resource $r_t$ at a given time period ($t_1$ to $t_2$), the server utilization $U_t$ is defined as (Eq. 7):

$$U_t = \int_{t_1}^{t_2} \sum_{i=1}^{n} P_{t'i}(r(t))\ \mathrm{d}t, \tag{7}$$

where $n$ is the number of cloud workloads running at time $t$. The actual energy consumption $E_a$ of a resource $r_t$ at a given time $t$ is defined as (Eq. 8):

$$E_a = (\mathrm{EU}_{\max} - \mathrm{EU}_{\min}) \times U_t + \mathrm{EU}_{\min}, \tag{8}$$

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

### 3.2 Resource scheduling unit

In this section, the design of RPU is presented. After successful provisioning of resources, RSU takes the information from the appropriate workload after analyzing the various workload details which the cloud consumer demanded. Resource scheduler schedules the incoming cloud workloads based on the workload details. First of all, we get cloud workloads to schedule and then find appropriate and available resources through resource provisioning and cloud workloads mapped and scheduled efficiently based on the resource scheduling policies. All the incoming workloads are put into waiting queue for further processing. The mapping is saved for future purpose. The activity model in UML is used for modeling the dynamic aspects of resource scheduling. Activities ultimately result in some action, which is made up of executable atomic computations that result in a state of the resource scheduling. The activity model of resource scheduling is shown in Fig. 4. There are five different entities interacting simultaneously: Cloud Customer, Customer Handler, Resource Scheduling Unit, Query Handler and Cloud Provider. The Resource Scheduling Unit then collects the information available resources from Resource Description (RD). Based on cloud consumer details, the QRPS framework assigns resources and executes cloud workloads successfully.

#### *3.2.1 QoS perspective of resource scheduling*

Four resource scheduling policies [compromised cost–time-based (CCTB) scheduling policy, time-based (TB) scheduling policy, cost- based (CB) scheduling policy and bargaining-based (BB) scheduling policy] have been considered based on different QoS parameters like cost, time and energy. Based on the scheduling policy, the resources are allocated to the cloud workloads. The information of the cloud workloads and computational resources is sent to the RSU. The RSU implements four resource scheduling policies: CCTB, TB, CB and BB scheduling policy. CWMP produces the cloud workloads and calculates workload deadline time. Each workload is characterized by their deadline, estimated budget and policy. The QoS of each cloud workload is also represented in the scheduling request of the cloud workload.

#### *3.2.2 Scheduling policy selection through decision tree*

In this section, rules for four resource scheduling policies [compromised cost–time-based (CCTB) scheduling policy, time-based (TB) scheduling policy, cost-based (CB) scheduling policy and bargaining-based (BB) scheduling policy] have been designed and decisions performed based on specified rules. Classification tree analysis can be used when the predicted outcome is the class to which the data belongs. Regression tree analysis can be used when the predicted outcome can be considered a real number.

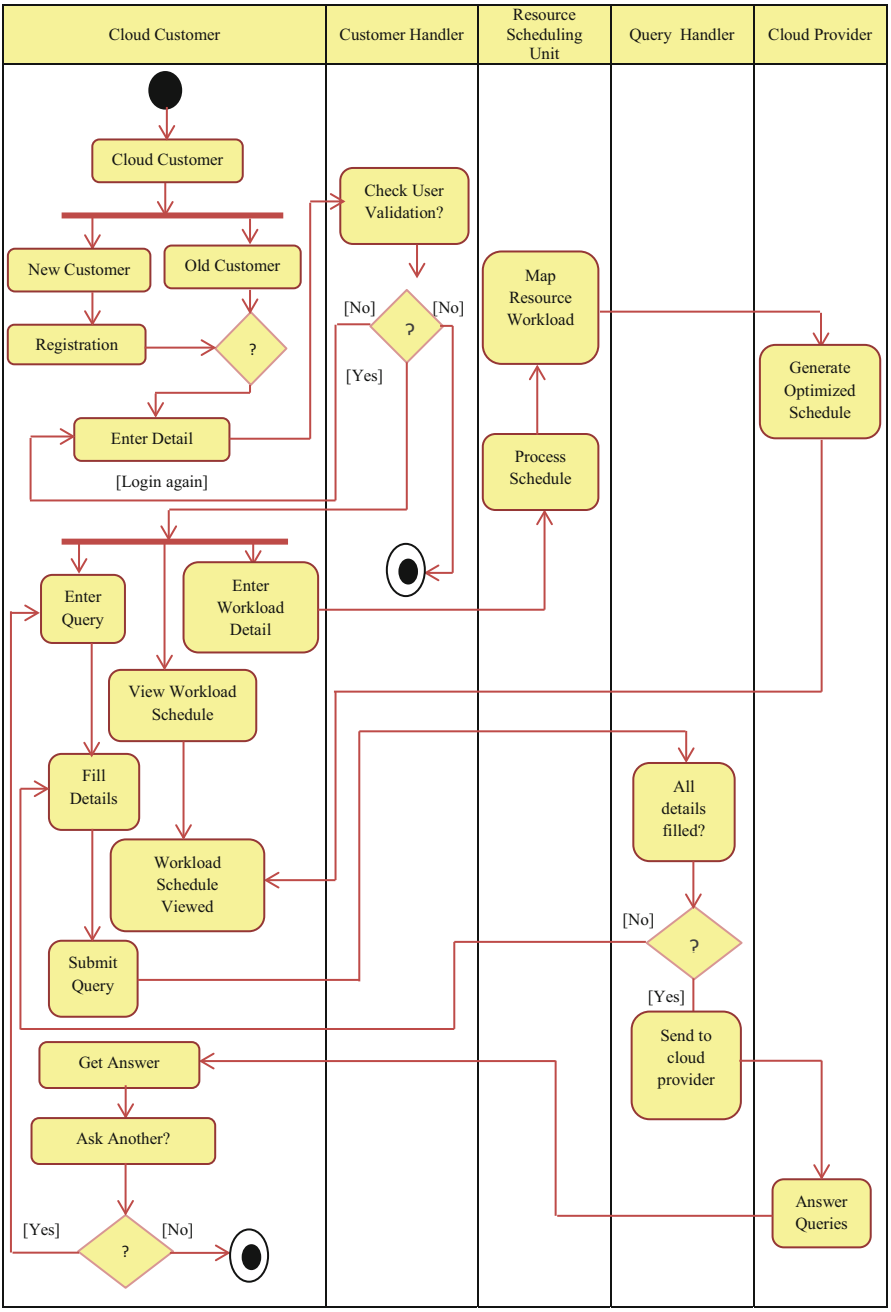


**Fig. 4** Working model of resource scheduling

**Table 4** Policies description

| Cost | Time | Policy |
|---|---|---|
| Minimum | Minimum | Compromised Cost-Time Based |
| Maximum | Minimum | Time Based |
| Minimum | Maximum | Cost Based |
| Cost agreement | Time agreement | Bargaining Based |

The classification and regression objective requires a set

$$I = \{I_1, \ldots, I_n\},$$

where $I_j$, $I \leq j \leq n$, $j \epsilon N$ is an object being reviewed and $n$ is a rational number.

The objects may be the information on executing policies in different values of cost and time (see Table 4). Each object is characterized by a set of variables:

$$I_i = \{x_1, \ldots x_m, y\},$$

where the $x_j$ are independent attributes, for which the values are known, and on which the value of the target variable y is based, and m is the total number of independent attributes. The independent attributes are: cost and time. The target variable is policy. A set of independent attributes is often defined as a vector:

$$X = \{x_1, \ldots x_m\}.$$

Every single variable $x_j$ can acquire values from a certain set:

$$C_j = \{c_{j1}, c_{j2} \ldots\}.$$

If the values of a variable are elements of a finite set, it is denoted as a categorical variable. For example, the variable policy acquires values from a range {compromised cost–time based, time based, cost based, bargaining based}.

If a range $C_y = \{c_1, c_2 \ldots\}$ of the variable y is finite, then the objective is referred to as classification objective; else the objective is referred to as regression objective. The policy details are shown in Table 4.

Decision trees comprise a way of presenting rules in a hierarchical, sequential structure. Figure 5 shows a decision tree for the data presented in Table 4.

In this case, the objects being reviewed are considered as points in an $(m + 1)$-dimensional space [6]. Then, the variables of an object $I_j = \{x_1, \ldots x_m, y\}$ are considered as coordinates, which are set in a relation in the following function:

$$\mathrm{y} = \{w_0 + w_1 x_1 + \cdots + w_m x_m\},$$

where $w_0 \ldots w_m$—weights of independent attributes, in which the determination of these weights is the central objective of finding a classification function. It is obvious

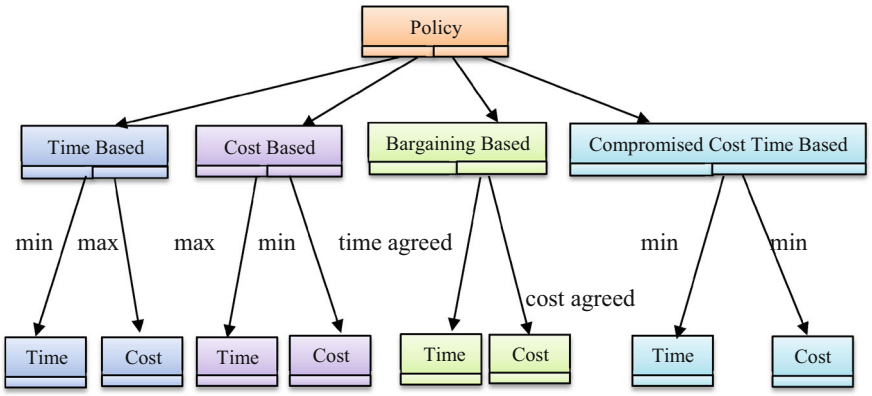


**Fig. 5** Decision tree of scheduling policies

that all variables should be presented as numeric parameters. There are different ways to transform logical and categorical variables into numeric ones. Logical types, as a rule, are encoded by Fig. 1.

The values of categorical variables are the names of the possible categories of an object being reviewed. Indeed, there may be more than two of such states. Their names should be mentioned and enumerated in a list. Each name from the list may be presented by its individual number. For the transformation into a numeric variable, for example, the value of the variable policy = {compromised cost–time based, time based, cost based, bargaining based} may be replaced by the values {0, 1, 2, 3}. It is observed from the results obtained in experimentation that the naive Bayes model is quite appealing because of its simplicity, elegance, robustness and effectiveness [25]. Such a classification can be made, via a naive Bayes algorithm, which applies the Bayes' formula for the calculation of a probability. The term naive is derived from the naive assumption that all attributes reviewed are independent of each other. Let us denote the probability that a certain object $I_j$ belongs to a class $C_j$ (that is $\mathrm{y} = c_j$) as $\mathrm{P}(\mathrm{y} = c_j)$. An event corresponding to the equality of independent attributes to specific values is denoted as $E$, with the probability of its occurrence as $P(E)$ . The idea of the algorithm is to calculate the conditional probability of affiliation of an object to $c_j$, with its independent attributes being equal to specific values. With the use of Bayes's theorem:

$$\mathrm{P}(\mathrm{y} = c_j|E) = \mathrm{P}\,(\mathrm{E}|\mathrm{y} = c_j)\mathrm{P}(\mathrm{y} = c_j)/\mathrm{P}\,(E)$$

In other words, rules are built with a conditional part that compares all independent attributes with the corresponding possible values. In the concluding part, all possible values of the target variable are present:

$$\text{If } [(x_1 = c_{j_1}) \text{ and } \ldots \text{and } (x_m = c_{j_m})] \text{ then } = (\mathrm{y} = c_j),$$

For each of these rules, Bayes' formula calculates a corresponding probability. Assuming that attributes acquire values independently of each other, let us perform the calculation of the probability P (E|y $= c_j$) through a multiplication of probabilities for each independent attribute:

$$P(E|y = c_j) = P(x_1 = c_{j_1}|y = c_j) \ldots P(x_m = c_{j(m)}|y = c_j).$$

Then, the probability for the whole rule can be defined by the formula:

$$P(y = c_j|\mathrm{E}) = \mathrm{P}(x_1 = c_{j_1}|y = c_j) \times \cdots \times \mathrm{P}(x_m = c_{j_m}|y = c_j) \times P(y = c_j)/P(E).$$

The probability of affiliation of an object to class $c_j$, provided that its attribute $x_j$ is equal to $c_{j_i}$, is defined by the formula:

$$P(x_j = c_{j_i}|\mathrm{y} = c_j) = \mathrm{P}(x_j = c_{j_i} \cap y = c_j) \,/P(y = c_j),$$

which is the ratio of the number of objects in the training sample, in which $x_j = c_{j_i}$ and $y = c_j$ are the number of objects belonging to class $C_j$. For example, for the objects from Table 4, the following probabilities for the values of the independent attribute cost and time have been obtained:

P (Cost = Minimum|Policy = Compromised Cost-Time Based) = 1/4;
P (Cost = Maximum|Policy = Time Based) = 1/4;
P (Cost = Minimum|Policy = Cost Based) = 1/4;
P (Cost = Cost Agreed|Policy = Bargaining Based) = 1/4;
P (Time = Minimum|Policy = Compromised Cost − Time Based) = 1/4;
P (Time = Minimum|Policy = Time Based) = 1/4;
P (Time = Maximum|Policy = Cost Based) = 1/4;
P (Time = Time Agreed |Policy = Bargaining Based) = 1/4.

Thus, if it is necessary to define whether a policy would take place with the following values of independent attributes (Event E):

Cost = Maximum;
Time = Maximum.

One should calculate the following conditional probabilities:

P (Policy = Compromised Cost-Time Based|E)
= P(Cost = Maximum|Policy = Compromised Cost − Time Based)
× P(Time = Maximum|Policy = Compromised Cost − Time Based)/P(E);
P (Policy = Time Based|E) = P (Cost = Minimum|Policy = Time Based)
× P (Time = Maximum|Policy = Time Based)/P (E);

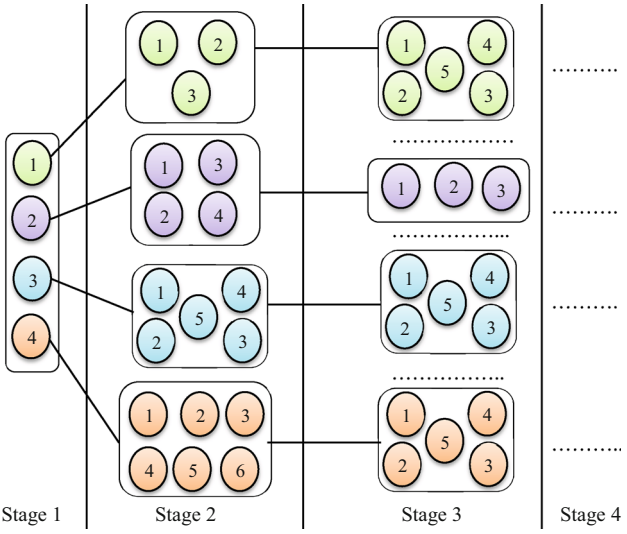


**Fig. 6** Decision tree-based branching

$$\begin{aligned}&\text{P (Policy = Cost Based|E) = P (Cost = Maximum|Policy = Cost Based)}\\&\quad\times \text{P (Time = Minimum|Policy = Cost Based)/P (E);}\\&\text{P (Policy = Bargaining Based|E) = P(Cost = Maximum|Policy = Bargaining Based)}\\&\quad\times\text{P (Time = Maximum|Policy = Bargaining Based)/P (E).}\end{aligned}$$

By inserting the corresponding probabilities, the following figure has been obtained:

$$\begin{aligned}&\text{P (Policy = Compromised Cost-Time Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/\text{P(E);}\\&\text{P (Policy = Time Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/\text{P (E);}\\&\text{P (Policy = Cost Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/\text{P (E);}\\&\text{P (Policy = Bargaining Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/\text{P (E).}\end{aligned}$$

The description of decisions for selection of scheduling policy is shown in Fig. 6.

The probability P(E) is not accounted for, as in the normalization of the probabilities for each of the possible rules it vanishes. The normalized probability for a rule is calculated by the formula:

$$P'(y = c_j|E) = P(y = c_j|E)/\sum P(y = c_j|E).$$

In this case, one may say that under specified conditions, the compromised cost–time-based, time-based, cost-based, bargaining-based policy will select with a probability of:

$$\begin{aligned}&\text{P (Policy = Compromised Cost-Time Based|E)}\\&\quad= 1/4 \times 1/4/(0.0625 + 0.0625 + 0.0625 + 0.0625) = 0.0625/0.25 = 0.25;\end{aligned}$$

**Table 5** Compromised cost–time-based rules

| Pending workload | TECT $\leq D_t$ | TEC $\leq B_E$ | $E_a \leq E_t$ | Policy |
|---|---|---|---|---|
| Yes | True | True | True | Yes |
| Yes | True | False | True | No |
| Yes | False | True | True | No |
| Yes | False | False | True | No |
| No | – | – | – | No |

$$\text{P (Policy} = \text{Time Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/0.25 = 0.25;$$
$$\text{P (Policy} = \text{Cost Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/0.25 = 0.25;$$
$$\text{P (Policy} = \text{Bargaining Based|E)} = 1/4 \times 1/4/\text{P (E)} = 0.0625/0.25 = 0.25.$$

Hence, it has been clearly proved that the probability distribution is uniform on the scheduling policies.

The rules for four resource scheduling polices along with the conditions are discussing below:

#### 3.2.2.1 Rules for CCTB scheduling policy

In this scheduling policy, the cloud provider minimizes cost as well as as execution time along with least energy consumption. It calculates the total expected cost, total expected completion time and time difference to allocate the resources. The allocation agents identify the missed deadlines and calculate the time difference for each workload then use the extra available time to the workloads with missed deadlines and execute all the cloud workloads within their corresponding deadlines.

The rules for CCTB scheduling policy are described in Table 5 along with their conditions.

where TECT is the total expected completion time, $D_t$ is the deadline time, TEC is the total expected cost, $B_E$ is the estimated budget, $E_{\text{Cloud}}$ is the actual energy consumption and $E_{\text{Threshold}}$ is the threshold energy value.

#### 3.2.2.2 Rules for TB scheduling policy

Time-based (TB) scheduling policy works as per the following: first, the allocation agent begins to compute the deadline time of the cloud workload in the given budget. Allocate resources based on time, the workload which has the shortest deadline time will execute first. If the two workloads have the same deadline time, then that workload will execute first the one that has lesser execution time. By default, processing speed (PS) =1. The allocation agent then schedules all the cloud workloads with the smallest execution time request to the resources that provide high QoS. If any deadline found is missed, then execution time will be recalculated by increasing the value of PS and it will increase the cost only. The rules for TB scheduling policy are described in Table 6 along with their conditions.

**Table 6** Time-based rules

| Workload pending | Urgency | Add resource | Workload |
|---|---|---|---|
| Yes | Yes | Reserve | Admit |
| Yes | No | Available | Admit |
| No | – | – | Finish |

**Table 7** Cost-based rules

| Workload pending | $R_A > 0$ | $E_t > W_d$ | $B_A > P_r$ | Status |
|---|---|---|---|---|
| Yes | True | True | True | Add resource |
| Yes | False | True | True | Add resource |
| No | – | – | – | Finish |
| Yes | True | False | True | Finish |
| Yes | True | True | False | Finish |

**Table 8** Bargaining-based rules

| Workload pending | Slot available | $Cur_t \leq$ NST | t.value () =! g.value() | TECT $\leq D_t$ | Mapping |
|---|---|---|---|---|---|
| Yes | Yes | True | True | True | Yes |
| No | Yes | True | True | True | No |
| Yes | No | True | True | True | No |
| Yes | Yes | False | False | True | No |
| Yes | Yes | True | True | False | No |

#### 3.2.2.3 Rules for CB scheduling policy

Cost-based (CB) scheduling policy works as follows: first, the allocation agent begins to compute the cost of each cloud workload and then sort, as priority is given to the cloud workload which has the maximum budget. If the two workloads have the same budget, then that workload will execute first the one that has lesser execution time. By default, PS = 1. The allocation agent then schedules all the workloads with high budget request to the resources that provide high QoS. Finally, all other workloads are scheduled on the available resources set. The rules for CB scheduling policy are described in Table 7 along with their conditions.

Where $R_A$ is the resource available, $E_t$ is the estimated time, $P_r$ is the resource price, $W_d$ is the desired deadline and $B_A$ is the available budget.

#### 3.2.2.4 Rules for BB scheduling policy

The implementation complies with the negotiation among the various resources and cloud workload producer along with different time slots. The allocation agent allocates the resources based on the bargaining between them. The mapping is done with the objective of best negotiation between the consumer and provider. The rules for BB scheduling policy are described in Table 8 along with their conditions.

Where $Cur_t$ is the current time, NST is the next scheduled time, t.value() is the workload price taken, g.value() is the workload price given, TECT is the total expected completion time and $D_t$ is the deadline time.
*Note* A detailed discussion of the formulas used in Sect. 3.2.2.1–3.2.2.4 to calculate the value of various parameters is discussed in our previous work [6].

## 4 Experimental setup and results

We have used simulations along with the empirical method to evaluate the the performance of QoS-based resource provisioning and scheduling framework (QRPS). QRPS has been verified in three different stages: (1) resource provisioning, (2) resource scheduling and (3) resource scheduling on cloud testbed. The first two stages have been verified using a cloud-based simulation environment, while the third stage has been verified using a real cloud environment. All the three experiments have been verified using QoS parameters such as energy consumption, execution cost and execution time. To test the performance of resource provisioning and scheduling framework, we have considered three QoS parameters: energy consumption, execution cost and execution time that are measured in kilowatt hour (KWh), grid dollars (G$) and seconds, respectively, using (Eqs. 1–8).

### 4.1 Simulation-based experimental results: resource provisioning

A variety of techniques and technologies exist for carrying out performance evaluation of resource provisioning using evaluation techniques like analytical and simulation. Some of the notable cloud tools are CloudSim and Cloud Analyst for simulation. Simulation allows the creation of large-scale virtual cloud environments and usage and availability scenarios that can be repeated and controlled. We have selected CloudSim [26] for simulation as it supports both system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource scheduling policies. CloudSim also supports modeling and simulation of cloud computing environments consisting of both single and inter-networked clouds (federation of clouds). Moreover, it exposes custom interfaces for implementing policies and scheduling techniques for allocation of VMs under inter-networked cloud computing scenarios. CloudSim has been installed along with its requirements using NetBeans, which provides cloud service. 3000 Independent cloud workloads were generated randomly in CloudSim as Cloudlets. User cloud workloads are modeled as independent parallel applications which are compute intensive. Thus, the data dependency among the cloud workloads in the parallel applications is negligible.

We have performed a different number of experiments by comparing QoS-based resource provisioning (RP) (QoS-based RP) used in resource provisioning and scheduling framework (QRPS) with non-QoS-based resource provisioning (non-QoS-based RP), in which no QoS parameter is considered. Non-QoS-based resource provisioning technique used for experimental evaluation in this paper has been designed by combining two traditional resource provisioning algorithms [First Come First Serve (FCFS) and Round Robin], in which resources are provisioned without

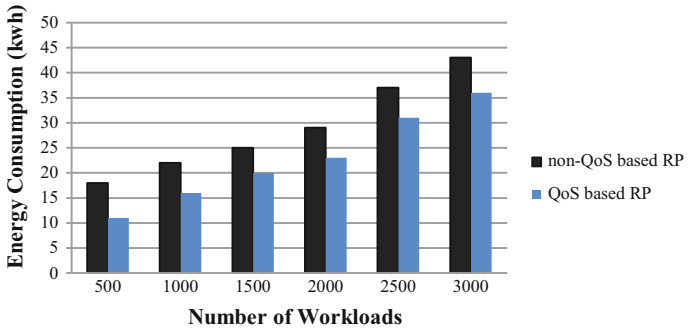


**Fig. 7** Effect of change in the number of workloads submitted on energy consumption

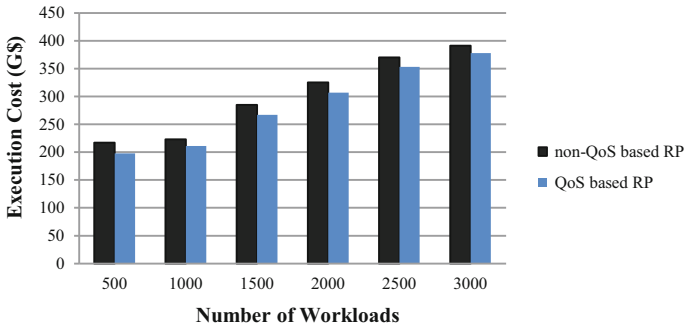


**Fig. 8** Effect of change in the number of workloads submitted on execution cost

considering QoS parameters. To measure the effect of growing the number of cloud workloads on the energy consumption, execution cost and execution time, the experiment has been performed.

*Test Case 1: energy consumption* We have calculated the value of energy consumption in kilowatt hour (kwh) for both QoS-based resource provisioning (QoS-based RP) and non-QoS-based resource provisioning (non-QoS-based RP) with different number of cloud workloads using formulas discussed in Sect. 3 (Eqs. 4–8). On increasing the number of cloud workloads, the value of energy consumption increases. The minimum value of energy consumption is 11 kWh at 500 cloud workloads and the maximum 36 kWh at 3000 cloud workloads for QoS-based RP. QoS-based resource provisioning performs better than non-QoS-based resource provisioning in terms of energy consumption at different number of cloud workloads as shown in Fig. 7.

*Test Case 2: execution cost* We have used (Eq. 3) to calculate the value of execution cost. With the increase in the number of workloads, execution cost rises as shown in Fig. 8. As per the number of workloads increases, QoS-based resource provisioning performs better than non-QoS-based resource provisioning. The cause is that QoS-based resource provisioning adjusts the resources at runtime according to the QoS requirements of the workload. The minimum cost used in QoS-based resource provisioning is 198 G$ at 500 workloads and the maximum is 378 G$ at 3000 workloads.

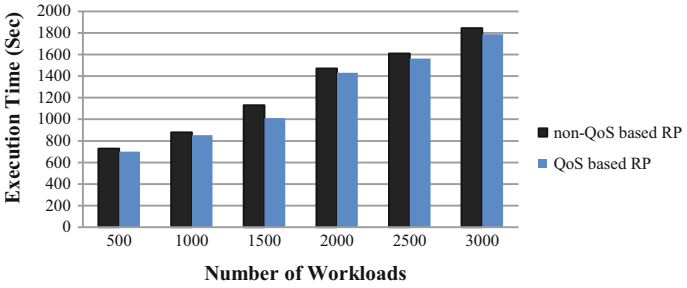


**Fig. 9** Effect of change in the number of workloads submitted on execution time

*Test Case 3: execution time* We have used (Eq. 2) to calculate the value of execution time. As shown in Fig. 9, the execution time increases with increase in the number of workloads. At 1500 workloads, the execution time in the QoS-based resource provisioning is 14.41 % lesser than non-QoS-based resource provisioning. Figure 9 shows that the execution time varies from 2000 workloads to 3000 workloads in the same ratio, but QoS-based resource provisioning performs better than non-QoS-based resource provisioning.

### 4.2 Simulation-based experimental results: resource scheduling

Tools used for setting up cloud-based simulation environment for verification of resource scheduling in QRPS are NetBeans IDE 7.1.2, Oracle Java SDK V.6, CloudSim Toolkit, SQL Server 2008, Microsoft Visual Studio and IntegratedNETJavaWeb. Cloud user interacts with Resource provisioning and scheduling framework through the Cloud Workload Management Portal (CWMP) to submit the workload details. User information, workload detail and resource detail are stored in the database through the SQL server. CWMP is implemented in .NET framework and framework is run in Microsoft Visual Studio. NetBeans is used to execute the CloudSim toolkit in which resource scheduling policies have been implemented. The workload details gathered from various users are transferred in a specific format from .NET framework to NetBeans through the use of IntegratedNETJavaWeb. The integration of multiple environments used to conduct experiments is described in detail in our previous research work [6].

To evaluate the overall performance of resource scheduling of QoS-aware resource provisioning and scheduling framework, we have considered energy consumption, execution cost, execution time, throughput and waiting time as QoS parameters and compared the values of QoS parameters of QoS-based Resource provisioning and scheduling framework (RP-QRSF, i.e., resource provisioning-based resource scheduling) with non-QoS-based resource scheduling [CTC (compromised time cost (CTC) scheduling algorithm) [27]] and QRSF (QoS-aware resource scheduling framework [6]).

*Test Case 1: energy consumption* We have calculated the value of energy consumption in kilowatt hour (kWh) for QoS-based resource provisioning and scheduling

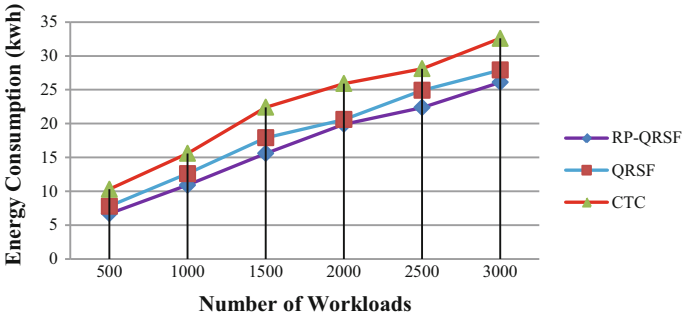


**Fig. 10** Comparison of energy consumption of different scheduling policies

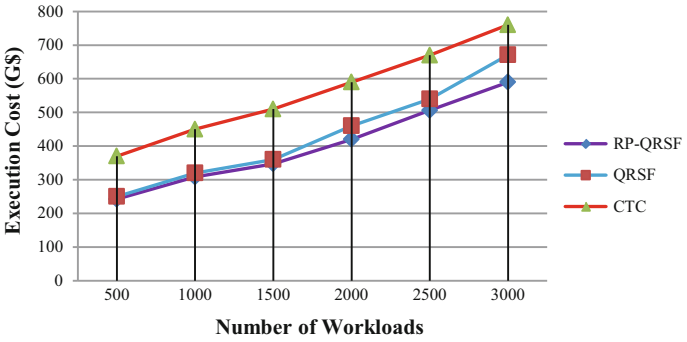


**Fig. 11** Comparison of execution cost of different scheduling policies

framework (RP-QRSF), non-QoS-based resource scheduling (CTC) and QoS-based resource scheduling framework (QRSF) with different number of cloud workloads using formulas discussed in Sect. 3 (Eqs. 4–8). With increasing the number of cloud workloads, the value of energy consumption increases. The minimum value of energy consumption is 6.7 kWh at 500 cloud workloads for RP-QRSF. RP-QRSF scheduling policy performs better than CTC and QRSF in terms of energy consumption at different number of cloud workloads as shown in Fig. 10. At 2500 workloads, energy consumption in RP-QRSF is 15.22 % lesser than CTC and 10.49 % lesser than QRSF.

*Test Case 2: execution cost* We have used (Eq. 3) to calculate the value of execution cost. With the increase in the number of workloads, the execution cost rises as shown in Fig. 11. As per the number of workload increases, RP-QRSF performs better than QRSF and CTC. The cause is that RP-QRSF adjusts the resources at runtime according to the QoS requirements of workload. The minimum cost used in RP-QRSF is 242 G$ at 500 workloads and the maximum is 590 G$ at 3000 workloads.

*Test Case 3: execution time* We have used (Eq. 2) to calculate value of the execution time. As shown in Fig. 12, the execution time increases with increase in the number of workloads. At 2500 workloads, the execution time in RP-QRSF is 21.60 % lesser than CTC and 3.72 % lesser than QRSF. Figure 12 shows that the execution time varies in the same ratio, but RP-QRSF performs better than CTC and QRSF.

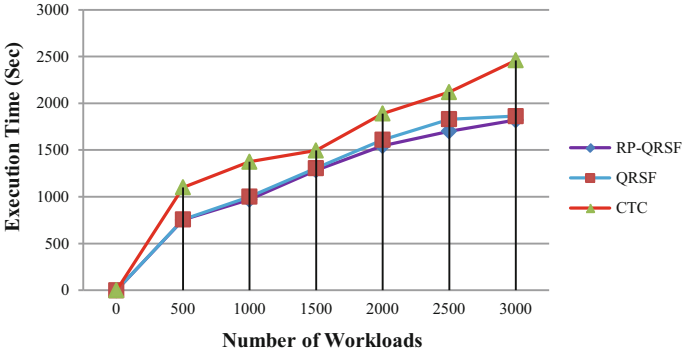


**Fig. 12** Comparison of execution time of different scheduling policies

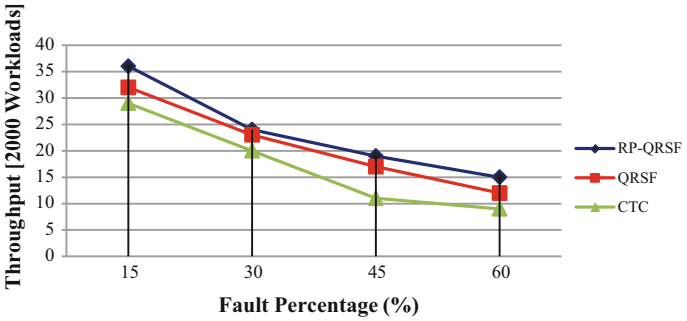


**Fig. 13** Throughput (2000 workloads) vs. fault percentage (%)

*Test Case 4: throughput* It is a ratio of the total number of workloads to the total amount of time required to execute the workloads. We have used the following formula to calculate the throughput (Eq. 9).

$$Throughput = \frac{\text{Total number of workloads } (W_n)}{\text{Total amount of time required to execute the workloads } (W_n)} \quad (9)$$

We have injected a number of software, network and hardware faults (fault percentage) to verify the throughput of the RP-QRSF with 2000 workloads. Figure 13 shows the comparison of the throughput of RP-QRSF, CTC and QRSF at 2000 workloads and it is clearly shown that RP-QRSF performs better than CTC and QRSF. In our experiment, we found the maximum value of throughput at a fault percentage 30 %, i.e., RP-QRSF has 16 % more throughput than QRSF and 19 % more than CTC.

*Test Case 5: waiting time* It is a ratio of the difference of workload execution start time ($\text{WE}_i$) and workload submission time ($\text{WS}_i$) to the number of workloads. We have used the following formula to calculate waiting time (Eq. 10):

$$\text{Waiting Time}_i = \sum_{i=1}^{n} \left( \frac{\text{WE}_i - \text{WS}_i}{n} \right), \quad (10)$$

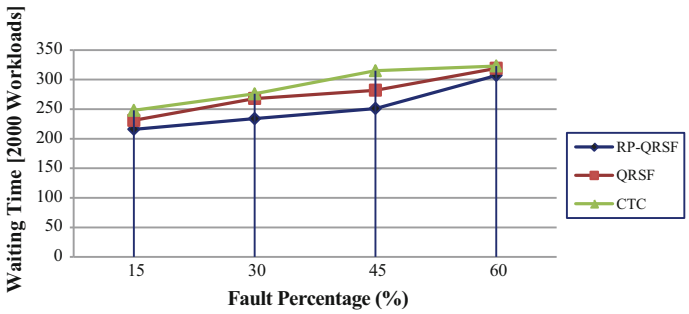


**Fig. 14** Waiting time (2000 Workloads) vs. fault percentage

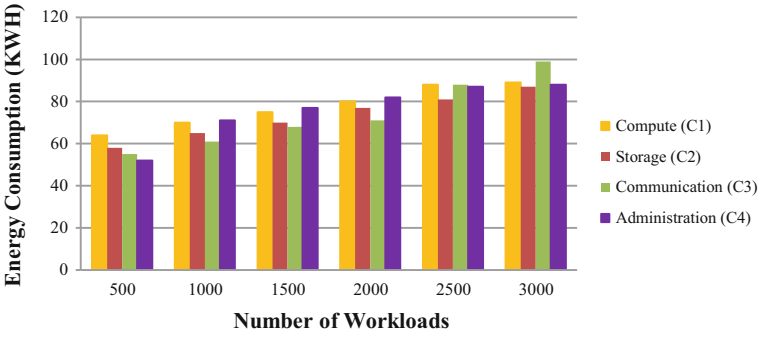


**Fig. 15** Comparison of energy consumption with different number of workloads of different clusters

where $n$ is the number of workloads. We have injected failures to verify the performance in terms of waiting time of workloads in RP-QRSF with different fault percentages (15–60 %). Figure 14 shows the comparison of waiting time of RP-QRSF, CTC and QRSF at 2000 workloads and it is clearly shown that RP-QRSF performs better than CTC and QRSF. In our experiment, we found the maximum difference in waiting time with fault percentage (45 %).

### *4.2.1 Variation of QoS parameters with different number of clusters*

We have clustered the workloads through $k$-means-based clustering algorithm in our previous work [5,6], namely: (a) compute (C1), (b) storage (C2), (c) communication (C3) and (d) administration (C4) as described in Table 3 in Sect. 3. Experiment has been performed to know the variation in energy consumption, execution cost and execution time with different number of clusters. Figure 15 shows the energy consumption of different number of workloads (500–3000) with resource provisioning and scheduling framework for different number of clusters. It is clearly shown that the cluster C3 (communication) consumes more energy than the other three clusters at 3000 cloud workloads.

Figure 16 shows the execution cost with different number of workloads (500–3000) with resource provisioning and scheduling framework for different number of

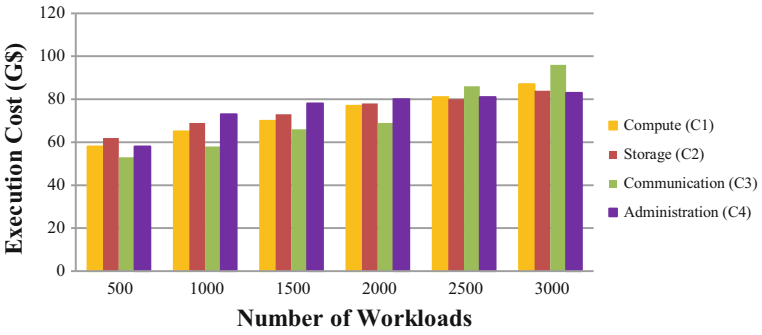


**Fig. 16** Comparison of execution cost with different number of workloads of different clusters

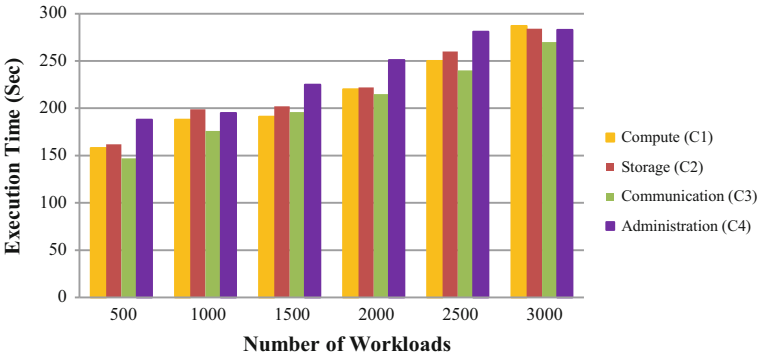


**Fig. 17** Comparison of execution time with different number of workloads of different clusters

clusters. It is clearly shown that all the four clusters have the same execution cost at 2400 workloads and cluster C2 and C4 performs better than C1 and C3 at 3000 workloads, but C3 performs excellently in terms of execution cost between 500 to 2400 workloads.

Figure 17 shows the execution time with different number of workloads (500–3000) with resource provisioning and scheduling framework for different number of clusters. It is clearly shown that cluster C1 takes a larger execution time than C2, C3 and C4 to execute different number of workloads, and clusters C2, C3 and C4 have almost the same execution time from 1500 workloads to 2500 workloads.

### 4.3 Empirical evaluation: resource scheduling on cloud testbed

The tools used for setting up cloud environment for empirical evaluation are Aneka, Microsoft Visual Studio 2010 (to create user interface), SQL Server 2008 and JADE Platform (for agents). JADE is used to establish the communication among devices and exchanging information for updates, and all the updated information is stored in a centralized database. Future usage and backup of corresponding updates is also maintained in case of failure of database. Aneka has been installed along with its

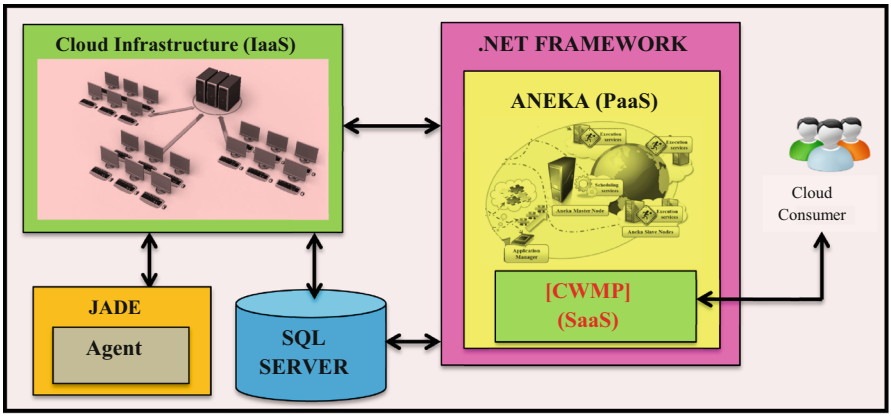


**Fig. 18** Cloud environment at Thapar University

requirements on all the nodes which are participating to provide cloud service. Nodes in this system can be added or removed based on the requirement. We have verified the proposed framework (QRSP) in a cloud environment. The integration of multiple environments used to conduct experiments is shown in Fig. 18. QRSP is installed on the main server and tested on a virtual cloud environment that has been established at the High Performance Computing Laboratory at Thapar University, India. We installed different number of virtual machines on different servers, and deployed the QRSP framework to measure the variations. 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). At software level, Microsoft Visual Studio 2010 is used to develop Cloud Workload Management Portal (CWMP) to provide user interface in which the user can access service from any geographical location.

At the platform level, Aneka cloud application platform [28] is used as a scalable cloud middleware to make interaction between IaaS and SaaS and continually monitor the performance of the system. Aneka is a framework for development, deployment and management of cloud applications. It consists of a scalable cloud middleware that is deployed on top of heterogeneous computing resources and an extensible collection of services coordinating the execution of applications, monitoring the status of the cloud and providing integration with existing cloud technologies. At infrastructure level, three different servers (consisting of computing nodes) have been used to test the resource scheduling algorithm which allocates resources to process the different user requests at runtime. Computing nodes used in this experimental work are further mentioned in Table 9.

The aim of this empirical study is to demonstrate that it is feasible to implement and deploy the QRPS framework on real cloud resources. The key components of the cloud environment are: user interface (CWMP), Aneka, resource scheduler and resources. A detailed discussion of the implementation using Aneka can be found in [24]. However, the following details enable the understanding of the cloud-based environment in which the QoS-based resource provisioning and scheduling framework (QRSP) is implemented:

**Table 9** Configuration details of Thapar cloud

| Configuration | Specifications | Operating system | Node |
|---|---|---|---|
| Intel Core 2 Duo-2.4 GHz | 1 GB RAM and 160 GB HDD | Window | 6 |
| Intel Core i5-2310-2.9GHz | 1 GB RAM and 160 GB HDD | Linux | 4 |
| Intel XEON E 52407-2.2 GHz | 2 GB RAM and 320 GB HDD | Linux | 2 |

1. Cloud consumer submits its request to the user interface (CWMP) that contains the workload description [workload name, workload type, budget, deadline and policy (cost based or time based or cost–time based or bargaining based)].
2. CWMP is deployed on Aneka platform (used as a scalable cloud middleware to make interaction between SaaS and IaaS).
3. Resource configuration is identified to schedule the number of workloads based on QoS requirements as described by the cloud consumer in the form of SLA.
4. Resource scheduler schedules the resources to the workloads based on the resource scheduling policy.
5. Resource manager measures the performance of the system in terms of QoS to avoid violation of SLA and the updated information is exchanged between all the nodes through JADE.
6. After successful execution of workloads, this further returns the resources to the resource pool.
7. At the end, the workload analyzer returns the updated experimental data along with the processed workload back to the cloud consumer.

To evaluate the performance of QoS-based Resource Provisioning and Scheduling Framework, we have compared execution time, cost and energy consumption of QoS-based resource provisioning and scheduling framework (RP-QRSF, i.e., resource provisioning-based resource scheduling) with non-QoS-based resource scheduling (CTC (compromised time cost (CTC) Scheduling algorithm) [27]) and QRSF (QoS-aware resource scheduling framework [6]). We have also performed experiments to determine the effect of an increase in the number of workloads on execution cost, energy consumption and execution time. All the experiments were started with workload name: Performance Testing (processing larger image file of size 713 MB). The experiment was conducted with different number of cloud workloads (10–60) for verification of energy consumption, execution cost and execution time.

*Test Case 1: Energy consumption* We have calculated the value of energy consumption in kilowatt hour (kWh) for QoS-based Resource provisioning and scheduling framework (RP-QRSF), non-QoS-based resource scheduling (CTC) and QoS-aware resource scheduling framework (QRSF) with different number of cloud workloads using formulas discussed in Sect. 3 (Eqs. 4–8). With increasing the number of cloud workloads, the value of energy consumption increases. The minimum value of energy consumption is 88.5 kWh at 10 cloud workloads for RP-QRSF. RP-QRSF scheduling policy performs better than CTC and QRSF in terms of energy consumption at different number of cloud workloads as shown in Fig. 19.

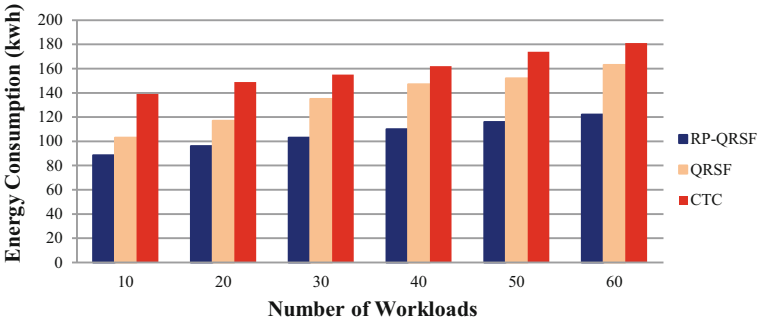


**Fig. 19** Effect of change in the number of workloads submitted on energy consumption

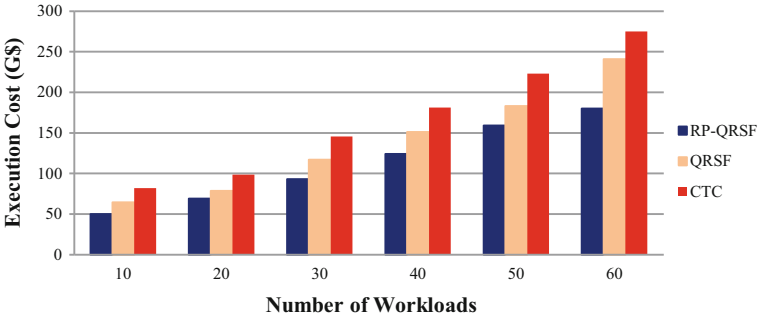


**Fig. 20** Effect of execution cost with change in the number of workloads

*Test Case 2: Execution cost* We have used (Eq. 3) to calculate the execution cost. With the increase in the number of workloads, the execution cost rises as shown in Fig. 20. As per the number of workloads increases, RP-QRSF performs better than QRSF and CTC. The cause is that RP-QRSF adjusts the resources at runtime according to the QoS requirements of workload. The minimum cost used in RP-QRSF is 50 G$ at 10 workloads and the maximum is 180 G$ at 60 workloads.

*Test Case 3: Execution time* We have used (Eq. 2) to calculate the execution time. As shown in Fig. 21, the execution time increases with increase in the number of workloads. At 60 workloads, the execution time in RP-QRSF is 12.61 % lesser than CTC and 5.64 % lesser than QRSF. Figure 21 shows that the execution time varies in the same ratio, but RP-QRSF performs better than CTC and QRSF.

Table 10 describes the comparison of execution cost, execution time and energy consumption used to process the same number of workloads (15 workloads of the same type) on a real cloud environment for RP-QRSF, QRSF and CTC. In this experiment, we have considered three different cloud infrastructures with different processor configurations (2 core processor, 4 core processor and 8 core processor) to measure the variation of execution cost, execution time and energy consumption.

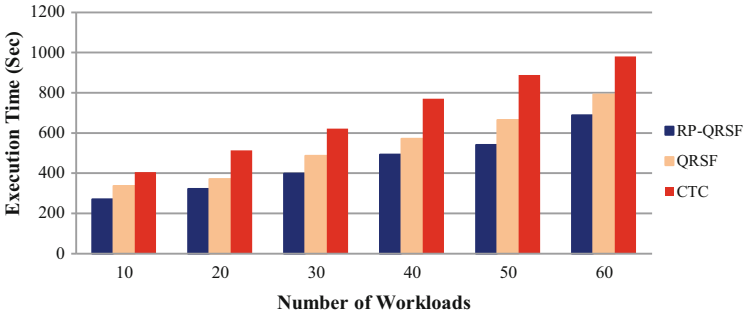


**Fig. 21** Effect of execution time with change in the number of workloads

**Table 10** Summary of experiment statistics on real cloud environment

| Configuration | QoS parameter | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Execution cost (G$) | | | Energy consumption (kWh) | | | Execution time (s) | | |
| | CTC | QRSF | RP-QRSF | CTC | QRSF | RP-QRSF | CTC | QRSF | RP-QRSF |
| 2 Core Processor | 98 | 65 | 50 | 133 | 107 | 88 | 362 | 311 | 270 |
| 4 Core Processor | 329 | 241 | 180 | 168 | 147 | 122 | 871 | 722 | 688 |
| 8 Core Processor | 477 | 323 | 370 | 391 | 347 | 209 | 2301 | 1881 | 1432 |

## 4.4 Discussions

The performance of QoS-based resource provisioning and scheduling framework (QRPS) has been compared with existing frameworks. The performance of the proposed framework has been analyzed with different number of cloud workloads, and QoS parameters such as execution time, execution cost and energy consumption to analyze the effect of QoS parameters on QRPS. The performance of QRPS and existing frameworks has been evaluated on the same cloud environment. QRPS has been verified in three different stages: (1) resource provisioning, (2) resource scheduling and (3) resource scheduling on Cloud Testbed. The minimum value of energy consumption is 11 kWh at 500 cloud workloads and maximum 36 kWh at 3000 cloud workloads in QoS-based resource provisioning. QoS-based resource provisioning performs better than non-QoS-based resource provisioning in terms of energy consumption at different number of cloud workloads. The minimum cost used in QoS-based resource provisioning is 198 G$ at 500 workloads and the maximum is 378 G$ at 3000 workloads in QoS-based resource provisioning. At 1500 workloads, execution time in QoS-based resource provisioning is 14.41 % lesser than non-QoS-based resource provisioning.

To evaluate the overall performance of resource scheduling of the QRPS framework, we have considered energy consumption, execution cost, execution time, throughput and waiting time as QoS parameters and compared the values of QoS parameters of the QRPS framework with existing resource scheduling frameworks (CTC and QRSF). At 2500 workloads, energy consumption in RP-QRSF is 15.22 % lesser than

CTC and 10.49 % lesser than QRSF. The minimum cost used in RP-QRSF is 242 G$ at 500 workloads and maximum is 590 G$ at 3000 workloads. At 2500 workloads, execution time in RP-QRSF is 21.60 % lesser than CTC and 3.72 % lesser than QRSF. We found the maximum value of throughput at fault percentage 30 %, i.e., RP-QRSF has 16 % more throughput than QRSF and 19 % more than CTC and the maximum variation in waiting time with fault percentage (45 %). For resource scheduling, the QRPS framework executes the same number of cloud workloads at a minimum execution time, execution cost and energy consumption as compared to existing resource scheduling frameworks.

Empirical study demonstrates that it is feasible to implement and deploy the QoS-based resource provisioning and scheduling framework (QRPS) on real cloud resources. In the empirical experiment, we have considered cloud infrastructures with 2, 4 and 8 core processors to measure the variation of execution time, execution cost and energy consumption. The minimum value of energy consumption is 88.5 kWh at 10 cloud workloads in RP-QRSF. As the number of workloads increase, RP-QRSF performs better than QRSF and CTC. The cause is that RP-QRSF adjusts the resources at runtime according to the QoS requirements of workload. The minimum cost used in RP-QRSF is 50 G$ at 10 workloads and the maximum is 180 G$ at 60 workloads. At 60 workloads, the execution time in RP-QRSF is 12.61 % lesser than CTC and 5.64 % lesser than QRSF. Considering all these QoS parameters, simulation outcomes and empirical evaluation, it is shown that the QRPS framework delivers a superior solution for heterogeneous cloud workloads and approximate optimum solution for challenges of resource management.

## 5 Conclusions and future scope

A QoS-based resource provisioning and scheduling framework for the cloud computing environment has been presented in this paper. The main goal of QoS-based resource provisioning is to reduce the complexity of provisioning for workload execution in cloud. We determine the effectiveness and usefulness of metrics to develop resource provisioning of workloads. Thus, resources can be managed easily and workloads can be executed effectively through QoS-based resource provisioning; further provisioned resources are scheduled and workloads can be executed on appropriate resources based on QoS requirements through different resource scheduling policies. This framework provides an effective outcome as compared to existing frameworks as shown in test cases. This research work also gives an insight into the problem of making decisions based on QoS parameters such as execution time, execution cost and energy consumption constraints in cloud computing. Resource provisioning and scheduling framework has been evaluated in both simulated and real cloud environment. The experimental results demonstrate that the proposed framework has better performance in terms of execution time, execution cost and energy consumption as compared to existing frameworks.

Our presented framework maps and executes the workloads based on workload details given by the user and resource details given by providers. In the 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.

**Acknowledgments** One of the authors, Sukhpal Singh (SRF-Professional), acknowledges the Department of Science and Technology (DST), Government of India, for awarding him the INSPIRE (Innovation in Science Pursuit for Inspired Research) Fellowship (Registration/IVR Number: 201400000761 [DST/INSPIRE/03/2014/000359]) to carry out this research work. We would like to thank all the anonymous reviewers for their valuable comments and suggestions for improving the paper. We would also like to thank Dr. Maninder Singh [EC-Council's Certified Ethical Hacker (C-EH)] for his valuable suggestions

## References

1. Singh S, Chana I (2015) QoS-aware autonomic resource management in cloud computing: a systematic review. ACM Comput Surv 48(3):1–46
2. Singh S, Chana I (2016) Cloud resource provisioning: survey, status and future research directions. Knowl Inform Syst 44:1–50
3. Singh S, Chana I (2016) A survey on resource scheduling in cloud computing issues and challenges. J Grid Comput 14:1–50
4. Chana I, Singh S (2014) Quality of service and service level agreements for cloud environments: issues and challenges. Cloud computing-challenges, limitations and R&D solutions. Springer International Publishing, pp 51–72
5. Singh S, Chana I (2015) Q-aware: quality of service based cloud resource provisioning. Comput Electr Eng 47:138–160. doi:10.1016/j.compeleceng.2015.02.003
6. Singh S, Chana I (2015) QRSF: QoS-aware resource scheduling framework in cloud computing. J Supercomput 71(1):241–292
7. Lee YC, Zomaya AY (2012) Energy efficient utilization of resources in cloud computing systems. J Supercomput 60(2):268–280
8. Kliazovich D (2012) GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. J Supercomput 62(3):1263–1283
9. Wang WJ, Chang YS, Lo WT, Lee YK (2013) Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments. J Supercomput 66(2):783–811
10. Li C (2012) Optimal resource provisioning for cloud computing environment. J Supercomput 62(2):989–1022
11. Ergu D, Kou G, Peng Y, Shi Y, Shi Y (2013) The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. J Supercomput 64(3):835–848
12. Lee HM, Jeong YS (2014) Performance analysis based resource allocation for green cloud computing. J Supercomput 69(3):1013–1026
13. Quiroz A, Kim H, Parashar M, Gnanasambandam N, Sharma N (2009) Towards autonomic workload provisioning for enterprise grids and clouds. In: Grid Computing, 2009 10th IEEE/ACM International Conference on. IEEE, pp 50–57
14. Vecchiola C, Calheiros RN, Karunamoorthy D, Buyya R (2012) Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka. Future Gener Comput Syst 28(1):58–65
15. Herbst NR, Huber N, Kounev S, Amrehn E (2014) Self-adaptive workload classification and forecasting for proactive resource provisioning. Concurrency Comput Pract Exp 26(12):2053–2078
16. Qavami HR, Jamali S, Akbari MK, Javadi B (2014) Dynamic resource provisioning in cloud computing: a heuristic markovian approach. In: Cloud computing, lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, vol 133. Springer International Publishing, pp 102–111

17. Varalakshmi P, Ramaswamy A, Balasubramanian A, Vijaykumar P (2011) An optimal workflow based scheduling and resource allocation in cloud. Advances in computing and communications. Springer, Berlin, Heidelberg
18. Li K, Xu G, Zhao G, Dong Y, Wang D (2011) Cloud task scheduling based on load balancing ant colony optimization. In: Chinagrid Conference (ChinaGrid), 2011 Sixth Annual. IEEE, pp 3–9
19. Topcuoglu H, Hariri S, Wu M-Y (1999) Task scheduling algorithms for heterogeneous processors. In: Heterogeneous computing workshop, (HCW'99), San Juan, Puerto Rico
20. Pandey S, Wu L, Guru S, Buyya R (2010) 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, Australia
21. Cordeschi N, Shojafar M, Baccarelli E (2013) Energy-saving self-configuring networked data centers. Comput Netw 57(17):3479–3491
22. Cordeschi N, Shojafar M, Amendola D, Baccarelli E (2014) Energy-efficient adaptive networked datacenters for the QoS support of real-time applications. J Supercomput 71(2):448–478
23. Singh S, Chana I (2015) EARTH: Energy-aware autonomic resource scheduling in cloud computing. J Intell Fuzzy Syst Syst:1–20. IOS Press doi:10.3233/IFS-151866
24. Singh S, Chana I, Buyya R (2015) Agri-Info: cloud based autonomic system for delivering agriculture as a service. Technical Report CLOUDS-TR-2015-2, cloud computing and distributed systems laboratory, the University of Melbourne, pp 1–31. Retrieved from http://www.cloudbus.org/reports/AgriCloud2015.pdf
25. Singh S, Chana I (2013) Efficient cloud workload management framework. Masters Dissertation, Thapar University, India. Retrieved From: http://dspace.thapar.edu:8080/dspace/bitstream/10266/2247/1/sukhpal_singh_me_thesis.pdf
26. Calheiros RN, Rajiv R, De Rose César AF, Rajkumar B (2009) CloudSim: a novel framework for modeling and simulation of cloud computing infrastructures and services. Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia
27. Liu K, Jin H, Chen J, Liu X, Yuan D, Yang Y (2010) 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
28. Calheiros RN, Vecchiola C, Karunamoorthy D, Buyya R (2012) The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. Future Gener Comput Syst 28(6):861–870